Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM Lorraine

Systèmes de reconnaissance de la parole revisité : Réseaux Bayesiens dynamiques et nouveaux paradigmes

THÈSE

présentée et soutenue publiquement le ?/10/2004

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Murat Deviren

Composition du jury

Président :	René Schott, Professeur Université Henri Poincaré-Nancy I
Rapporteurs :	Steve Young, Professeur University of Cambridge Régine André-Obrecht, Professeur Université Paul Sabatier de Toulouse
Examinateur :	Kamel Smaïli, Professeur Université Nancy II
Directeurs :	Khalid Daoudi, Chargé de Recherche INRIA Jean-Paul Haton, Professeur Université Henri Poincaré-Nancy I

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Abstract

In this thesis we focus on four principle components of a speech recognition system : acoustic modeling, language modeling, speech feature extraction and noise compensation. We propose novel modeling approaches for acoustic and linguistic modeling within the Bayesian networks formalism. Bayesian networks are a subset of probabilistic graphical models that include the most widely used probability models in speech recognition. Therefore rethinking the modeling problems in this formalism provides new perspectives that was not considered previously. Besides novel modeling approaches we also address new speech feature extraction schemes. Our main motivation in this direction is to seek for robust features that are not bound to be used in classical hidden Markov modeling (HMM) approach. Finally, we address the robustness problem for varying application conditions and propose a novel supervised compensation scheme.

Acoustic Modeling : State-of-the-art automatic speech recognition (ASR) systems are based on probabilistic modeling of the speech signal using Hidden Markov Models (HMM). In this thesis, we reformulate the acoustic modeling problem in speech recognition within the probabilistic graphical models (PGM) formalism. Dynamic Bayesian networks (DBN) are a subset of PGM which include HMM as a special case. One of the principle weakness of HMMs is the independence assumptions on the observed and hidden processes of speech. We propose to use the DBN setting to extract the proper dependence structure for speech modeling rather than limiting ourselves with HMMs. The proposed approach is based on structure learning paradigm in DBN framework [36, 34]. This approach has the advantage to guaranty that the resulting model represents speech with higher fidelity than HMM.

As a first step we test our approach on isolated digit task [36, 37]. Next, we extend this approach for continuous speech recognition. In this task, one of the main problems was the decoding algorithm. Although Viterbi algorithm provides an efficient solution in HMM framework, in the proposed DBN setting each acoustic unit may have different conditional independence properties. In the first place, we develop a technique to construct a meta model that represents all acoustic units with the same graph structure such that standard Bayesian network inference algorithms can be applied directly [39]. However this approach is computationally demanding. Therefore in [41, 38], We develop a fast decoding algorithm based on dynamic programming that exploits the structural properties of the acoustic models proposed.

Language Modeling : Statistical language models attempt to construct a representation of language using the empirical probability distribution of linguistic events. Briefly the specification of a language model involves the definition of implicit and/or explicit variables of language. The dynamic evolution of language is controlled by these variables and their interactions through time. Each variable interacts with a certain number of factors that constitute its context. In probabilistic terms the context of a linguistic unit is defined with conditional independence properties. Given its context each linguistic unit is assumed to be independent of other linguistic events.

On the other hand, conditional independence is the core property of probabilistic graphical models, it is indeed the exploitation of this property that leads to efficient and generic inference algorithms of graphical models. Moreover, n-gram and n-class models (and other language modes) are very particular instances of *dynamic Bayesian networks* (DBNs) which are *oriented* probabilistic graphical models. Thus, it is a natural idea to rethink language models within the

general framework of DBNs and seek potential benefits from this rethinking.

In this respect we use the DBNs framework in order to achieve a better exploitation of each linguistic unit considered in modeling. We develop a unifying approach that processes each of these units in a unique model and construct new data-driven language models with improved performances. The principle of this approach is to construct DBNs in which a variable (word, class or any other linguistic unit) may depend on a set of context variables. We evaluate our approach using several datasets [42, 44, 43].

Speech feature extraction : Another direction of my thesis is on new acoustic feature extraction algorithms for ASR. Existing feature extraction algorithms are developed considering the properties of HMM formalism. Therefore there is a need for new directions in this field to be able to extract informative features that can later be processed within the DBN setting without the limiting constraints of HMMs. With this motivation in mind we review wavelet analysis based feature extraction schemes for speech recognition and propose a new technique for frequency localized robust feature extraction using wavelets [40]. The proposed scheme is based on the projection of mel-scale filter-bank energies on a wavelet basis. This technique is indeed a reformulation of the frequency filtering approach that emphasizes formant frequencies with a simple linear filter and achieves robust frequency localized features. Using a wavelet basis improves decorrelation property and allows a controlled localization in frequency domain.

Noise compensation : In this field we propose a new predictive compensation scheme which makes no assumption on how the noise sources alter the speech data and which do not rely on clean speech models. Rather, this new scheme makes the (realistic) assumption that speech databases recorded under different background noise conditions are available. The philosophy of this scheme is to process these databases in order to build a "tool" which will allow it to handle new noise conditions in a robust way. The performance of this new compensation scheme is evaluated on a connected digits recognition task [24, 25].

Keywords: automatic speech recognition, acoustic modeling, language modeling, dynamic Bayesian networks, speech feature extraction, wavelet analysis, noise robustness

Remerciements

Les remerciements.

Je dédie cette thèse
au soleil de Nancy,
qui n'ose pas
se montrer très souvent.

Table des matières

Partie I Résumé étendu

Chapitre 1
Introduction

1.1 Reconnaissance de la parole . . . 3
1.2 Modélisation acoustique . . . 5
1.3 Modélisation du langage . . . 6
1.4 Paramétrisation du signal acoustique . . . 6
1.5 Compensation du bruit . . . 7

Chapitre 2
Modélisation acoustique par des réseaux Bayésiens dynamiques

2.1 Introduction . . . 9
2.2 Réseaux Bayésiens dynamiques . . . 10
2.3 Classe de recherche de structures . . . 11
2.4 Apprentissage de structures . . . 12
2.5 Algorithme de décodage . . . 13
2.6 Expériences . . . 14
2.7 Conclusion . . . 15

Chapitre 3
Modélisation du langage par des réseaux Bayésiens dynamiques

3.1 Introduction . . . 17
3.2 Réseaux Bayésiens dynamiques . . . 18
3.3 RBDs pour la modélisation du langage . . . 19
3.3.1 Structures graphiques utilisées . . . 20
3.3.2 Expérimentations et évaluation . . . 21
3.4 Conclusion et perspectives . . . 23

Chapitre 4
Compensation prédictive pour la robustesse au bruit

4.1 Introduction . . . 25
4.2 Architecture de la compensation supervisée . . . 26
4.3 Cadre expérimental . . . 28
4.4 Résultats et discussion . . . 28
4.5 Conclusion . . . 30

Chapitre 5
Conclusion

Partie II Revisiting speech recognition systems : dynamic Bayesian networks and new computational paradigms

Chapitre 1
Introduction

1.1 Statistical speech recognition . . . 37
1.2 Acoustic Modeling . . . 39
1.3 Language Modeling . . . 40
1.4 Speech feature extraction . . . 41
1.5 Noise compensation . . . 41
1.6 Organization . . . 42

Chapitre 2
Bayesian Networks

2.1 Contributions . . . 43
2.2 Bayesian Networks . . . 43
2.2.1 Some notation . . . 44
2.2.2 Definition . . . 44
2.2.3 Structure Semantics . . . 44
2.2.4 Numerical Parameterization . . . 46
2.3 Inference . . . 48
2.3.1 Inference in probabilistic models . . . 48
2.3.2 Exploiting conditional independence for efficient inference . . . 49
2.3.3 Overview of exact inference in BNs . . . 49
2.4 Learning . . . 50

2.4.1 Parameter learning 50
2.4.2 Parameter learning with incomplete data 51
2.4.3 Structure learning 52
2.5 Dynamic Bayesian Networks 55
2.5.1 Representation 57
2.5.2 Inference in DBNs 60
2.5.3 Learning DBNs 63

Chapitre 3
Acoustic Modeling with DBNs

3.1 Introduction 65
3.2 Contributions 66
3.3 Overview of Acoustic modeling using DBNs 66
3.3.1 Factored state space models 67
3.3.2 Explicit observation dependence modeling 69
3.3.3 Multi-stream techniques 71
3.4 Structure learning of DBNs for acoustic modeling 71
3.5 A Class of DBNs for acoustic modelling 72
3.5.1 Numerical parameterization 73
3.5.2 State transition topology 75
3.5.3 Model complexity 75
3.6 Learning DBNs for speech recognition 76
3.6.1 Maximum likelihood parameter estimation 76
3.6.2 Learning the structure 77
3.6.3 Embedded training for Speech recognition 78
3.7 Inference for Speech Recognition 79
3.7.1 Isolated speech recognition 79
3.7.2 Continuous speech recognition 79
3.7.3 Brute force approach 79
3.7.4 A fast decoding algorithm 81
3.8 Experiments 85
3.8.1 Isolated digit recognition 85
3.8.2 Connected digit recognition 87
3.8.3 Continuous phoneme recognition 88

Chapitre 4
Language Modeling with DBNs

4.1 Introduction 91
4.2 Contributions 91
4.3 Statistical Language Modeling 92
4.4 n-gram models 93
4.5 class-based n-gram models 94
4.6 Language Modeling with DBNs 95
4.6.1 Model Specifications 96
4.7 Learning 97
4.7.1 Smoothing and Back-off 97
4.8 Language model evaluation 98
4.8.1 Entropy of natural language 99
4.8.2 Perplexity as a measure for language model evaluation 99
4.9 Experiments 100
4.9.1 Le Monde 100
4.9.2 MIAMM 101
4.9.3 SPORT 102

Chapitre 5
Speech Features Extraction

5.1 Introduction 103
5.2 Contributions 104
5.3 Overview of speech parameterization schemes in ASR 104
5.3.1 Common feature extraction schemes for ASR 104
5.3.2 Common processing blocks 106
5.4 Wavelets in speech signal parameterization 107
5.4.1 Wavelets in speech recognition 107
5.4.2 Experiments with wavelet coefficients 110
5.4.3 Multi-resolution cepstral features 117
5.5 Frequency Filtering 117
5.5.1 Frequency localized features 117
5.5.2 Experiments : Frequency Filtering in Real Noise Conditions 119
5.5.3 Experiments : Effect of number of filter-banks in FF 120
5.6 Wavelet Filtering 121
5.6.1 Frequency Filtering and Wavelet transform 121
5.6.2 Multi-resolution Frequency Filtering 123

5.7 Experiments on full Aurora 3 dataset . . . 125
5.8 Conclusion and perspectives . . . 126

Chapitre 6
Predictive Compensation for Noise Robustness

6.1 Introduction . . . 127
6.2 Contributions . . . 128
6.3 Techniques for noise robustness . . . 128
6.3.1 Adaptive model compensation . . . 129
6.3.2 Predictive model compensation . . . 130
6.4 Supervised-predictive compensation . . . 130
6.4.1 Simple linear regression modeling . . . 132
6.5 Model Parameter Selection . . . 132
6.5.1 Noise model . . . 133
6.5.2 Speech model . . . 135
6.6 Experiments . . . 137
6.6.1 Experimental setting . . . 137
6.7 Results and discussion . . . 138
6.8 Conclusion . . . 141

Chapitre 7
Conclusions

Annexe A
JLO algorithm

A.1 Constructing the Junction Tree . . . 145
A.2 Message passing . . . 147
A.3 Initializing the Junction Tree . . . 149
A.3.1 Incorporating evidence . . . 149
A.4 Generalized operations . . . 150
A.4.1 Most probable configuration : . . . 150
A.4.2 Random configuration : . . . 150

Annexe B
Complete data ML parameter learning

B.1 Likelihood Function . . . 151
B.2 ML parameter estimation for discrete BNs . . . 151
B.3 ML parameter estimation for continuous BNs . . . 152

B.4 ML parameter estimation for hybrid BNs . . . 152

Annexe C
Incomplete data ML parameter learning : EM algorithm

C.1 EM algorithm for parameter learning in discrete BNs . . . 155
C.2 EM algorithm for parameter learning in continuous BNs . . . 156
C.3 EM algorithm for parameter learning in hybrid BNs . . . 156

Annexe D
Maximum Likelihood Parameter Estimation of Acoustic DBN Models

D.1 Transition Probabilities . . . 158
D.2 Output Probabilities . . . 160
D.3 Mixture Output Densities . . . 161
D.4 Summary . . . 162

Annexe E
Hidden Markov Trees

E.1 Les modèles d'arbres de Markov cachés . . . 165
E.2 Evaluation de la Vraissemblance . . . 166
E.3 L'algorithme Upward-Downward . . . 167
E.3.1 L'étape Upward . . . 168
E.3.2 L'étape Downward . . . 169
E.4 Estimation des paramètres par l'algorithme EM . . . 170

Annexe F
t Tests

F.1 The Student-t distribution . . . 177
F.2 The Student-t Test for One Sample (paired t-test) . . . 178
F.3 The Student-t Test for Two Samples . . . 179

Bibliographie **181**

Table des figures

2.1 HMM représenté comme DBN . . . 11
2.2 Structure de DBN $(\kappa, \tau_p, \tau_f) = (2, 1, 1)$, $T = 4$ . . . 11

3.1 modèles trigramme et biclasse . . . 19
3.2 RBD où chaque mot (resp. classe) dépend de la classe antérieure, présente et futur (resp. antérieure). . . . 20
3.3 RBD donnant une perplexité plus faible que le bigramme. . . . 22

4.1 Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.3$. . . . 29
4.2 Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.4$. . . . 29
4.3 Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.5$. . . . 30

2.1 A Bayesian network representation for the joint probability distribution $P(X_1, X_2, X_3, X_4, X_5)$. . . . 45
2.2 Directed acyclic graph structures for $P(X, Y, Z)$ where $X \perp Y | Z$ w.r.t node orderings (a) X, Y, Z, (b) X, Z, Y and (c) Y, Z, X. . . . 46
2.3 SEM algorithm . . . 56
2.4 Illustration of DBN representation and the unrolling mechanism (modified from [52]). (a) Initial network. (b) Transition network. (c) Unrolled DBN for 4 time slices. . . . 58
2.5 Example DBN structure unrolled for 5 time slices, $T = 5$ . . . 59
2.6 (a), (b) Initial networks for $t = 1, 2$. (c) Transition network. (d) Final network for $t = T$ . . . 60

3.1 HMM represented as a DBN . . . 66
3.2 Factored state space models . . . 67
3.3 DBN models proposed to incorporate factorized speech dynamics [132]. . . . 68
3.4 DBN model for noisy speech recognition [94] with two auxiliary factors : noise level (S), noise type (N). . . . 69
3.5 Hidden feature model for speech recognition . . . 70
3.6 DBN models for explicit observation correlation modeling. . . . 70
3.7 Factored state multi-stream DBN model . . . 71
3.8 Graphical representation of CI assumptions in the proposed class of DBNs. . . . 73
3.9 The class of structures for $(1, 0, 0) \leq (\kappa, \tau_p, \tau_f) \leq (2, 1, 1)$. . . . 74
3.10 Left-to-right state transition HMM represented as a finite state automaton. . . . 75

3.11 The global maxima in the structure space of all the digits $(T = 4)$, $(\kappa, \tau_p, \tau_f) = (1, 0, 1)$. . . . 86
3.12 Recognition accuracy comparisons of DBN based systems with equivalent HMM based systems . . . 90

4.1 n-gram . . . 94
4.2 n-class . . . 95
4.3 Allowed dependencies due assumption 1 . . . 96
4.4 Allowed dependencies due assumption 2 . . . 97
4.5 DBN structure for $\prod_t P(w_t | w_{t-1}, c_{t-2}) P(c_t | w_t)$, that yields a lower perplexity than bi-gram. . . . 101

5.1 Source-filter model for speech production . . . 105
5.2 Frequency partitioning of [46, 69] . . . 108
5.3 Frequency partitioning of [81] . . . 108
5.4 Frequency partitioning of [47] . . . 108
5.5 Frequency partitioning of [110, 109] . . . 109
5.6 Frequency partitioning of [16] . . . 109
5.7 Frequency resolution of different wavelet decomposition trees : (a) dyadic tree, (b) a test tree, (c) mel-scale approximation tree [47], (d) full wavelet packet decomposition tree. . . . 115
5.8 Frequency filtering with different number of filterbanks on Spanish database WM conditions compared with W1007 standard parameterization. . . . 121
5.9 Frequency filtering with different number of filterbanks on Spanish database MM conditions compared with W1007 standard parameterization. . . . 121
5.10 Frequency filtering with different number of filterbanks on Spanish database HM conditions compared with W1007 standard parameterization. . . . 122

6.1 Model fit to the variation of (a) mean of c1 (b) variance of c1 . . . 133
6.2 Comparison of average KL distances between learned and estimated model parameters using noise model mean and variance vectors. . . . 135
6.3 Model fit to the variation of (a) mean (b) variance of c4 in model ONE, 1st state, 2nd mixture (c) the variation of mean c4 for all mixtures of noise model. . . . 136
6.4 Comparison between recognition accuracy of matched, multi-conditions, f-estimated and clean speech models for $\alpha = 0.05$. . . . 139
6.5 Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.3$. . . . 139
6.6 Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.4$. . . . 140
6.7 Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.5$. . . . 140

A.1 Illustration of the junction tree construction algorithm taken from [12] (pages 144-145). (a) A directed acyclic graph. (b) Corresponding moralized graph. (c) Triangulated graph with the indicated node ordering. (d) A junction tree associated with the graph. . . . 148

E.1 Modèle d'arbre de Markov caché : Chaque nœud noir représente l'observation et chaque nœud blanc représente l'état de l'observation. Pour modéliser les dépendances entre les cœfficients, on met des liens entre les états cachés 166

Liste des tableaux

2.1 Résultats de l'algorithme SEM. 14
2.2 Précision d'identification des chiffres (%). 15

3.1 Perplexité des réseaux Bayésiens dynamiques 22

4.1 Comparaison entre les taux de reconnaissance (%) du modèle matché, multi-conditions et f-estimé pour le bruit cible *machinegun*. 30

3.1 Digit recognition rates for different DBN model structures (%). 87
3.2 Learned digit DBN structures using SEM algorithm. 87
3.3 Digit recognition accuracies using DBN and HMM models (%). 88
3.4 Phoneme recognition accuracy of HMM and DBN based systems using different number of Gaussian mixture components (%). 89
3.5 Learned phone DBN structures using SEM algorithm. 89

4.1 Training is performed on 11 months of "Le Monde 87". Perplexity is computed using 1 month containing more than 1M words. 100
4.2 Some examples of queries in the MIAMM corpus. 101
4.3 Perplexity results on MIAMM corpus, $|W| = 160$, $|C| = 10$. 102
4.4 SPORT corpus perplexity results with class-based models computed with/without $< UNK >$ 102

5.1 Recognition rates for various wavelet functions using different level of decompositions (%). 112
5.2 Recognition rates for experiment 2 (%). 113
5.3 Recognition rates with and without scaling coefficients, using coiflet-5 wavelet(%). 113
5.4 Recognition rates using different parameterization schemes (%). 114
5.5 Recognition rates using different wavelet packet tree structures (%). 116
5.6 Recognition rates under white Gaussian noise (%). 117
5.7 Recognition rates using multi-resolution cepstral features. [154321] : denotes 1 cepstral coefficient for the 1st band, 5 cepstral coefficients for the second band, 4 coefficients for the third band etc. (%). 118
5.8 Word accuracies (%) using 12 Mel-scale filter-banks for FFs and 23 filter-banks for MFCCs. Total number of static features is 13. 120
5.9 Word accuracies (%) using FF of 12 Mel-scale filter-banks with Haar and Daub4 wavelets. Total number of static features is 13. 123
5.10 Word accuracies (%) using DWT on 23 Mel-scale filter-banks with Haar wavelet. The size of the feature vector for each case is indicated inside the parenthesis. . . 124

5.11 Word accuracies (%) using DWT on 23 Mel-scale filter-banks with Daub4 wavelet. The size of the feature vector for each case is indicated inside the parenthesis. . . 124
5.12 Spanish database word accuracies (%) using W1007 STQ standart frontend using 23 FB and several frequency filters on 12 logFBEs. 125
5.13 Danish database word accuracies (%) using W1007 STQ standart frontend using 23 FBs and several frequency filteres on 12 logFBEs. 125
5.14 Finnish database word accuracies (%) using W1007 STQ standart frontend using 23 FBs and several frequency filteres on 12 logFBEs. 125

6.1 TiDigits database digit recognition accuracy(%) in white noise where not all model parameters are adapted to the environment. 137
6.2 Comparison between matched, multi-condition and estimated models when the target noise (machinegun) is excluded from training data. 141

Première partie

Résumé étendu

Chapitre 1

Introduction

Sommaire

1.1 Reconnaissance de la parole 3
1.2 Modélisation acoustique 5
1.3 Modélisation du langage 6
1.4 Paramétrisation du signal acoustique 6
1.5 Compensation du bruit 7

1.1 Reconnaissance de la parole

Le but d'un système de reconnaissance automatique de la parole est de déduire des unités linguistiques (i.e. mots) à partir de signaux acoustiques. A cause de la nature aléatoire de la parole et aux interférences, on ne peut pas construire une formulation déterministe pour cette tâche. Le problème est en générale formulé dans un cadre probabiliste. Dans ce cadre, la reconnaissance de la parole est énoncé comme l'estimation de la représentation linguistique 'la plus probable' étant donné un signal acoustique. Le formulation mathématique de ce problème est :

$$\hat{W} = \arg\max_{W} P(W|O)$$

où O est l'ensemble des observations extraites d'un signal acoustique et W est une variable aléatoire qui prend ses valeurs dans l'espace des représentations linguistiques possibles pour le langage considéré.

$P(W|O)$ est la distribution conditionnelle des représentations linguistiques étant données les observations. Cette distribution conditionnelle constitue la base de connaissance d'un système RAP. Cette base est construite en s'appuyant sur des techniques d'apprentissage statistique et sur des connaissance a priori des mécanismes physiologiques pour la production de la parole. Le rôle de ces connaissances expertes est de fournir des hypothèses simplificatrice pour guider le moteur statistique afin de mieux exploiter les informations appropriées lors de la reconnaissance.

La production de la parole est extrêmement complexe au niveau physiologique et linguistique. Par conséquent il n'existe pas de formulation paramétrique simple pour la distribution $P(W|O)$ modélisant à la fois les informations acoustiques et linguistiques. L'approche communément employée consiste alors à séparer le problème en ses deux composantes, acoustique et linguistique, qui sont alors traitées indépendamment. Ceci est réalisé par une reformulation Bayesienne :

$$\hat{W} = \arg\max_{W} P(O|W)P(W).$$

Dans cette équation, le modèle acoustique $P(O|W)$ encode la distribution statistique de l'acoustique étant donnée l'étiquetage linguistique. $P(W)$ est la probabilité attribuée par le modèle de langage qui encode l'information a priori sur le langage. Cette approche est appelée le modèle source - chaîne. $P(W)$ constitue la source du langage avec toutes ses contraintes linguistiques. $P(O|W)$ est la chaîne acoustique qui produit le signal de parole à partir de l'unité linguistique W. La composante linguistique est appris sur des documents écrits, que l'on suppose en nombre suffisant pour représenter les propriétés du langage. Pour la composante acoustique, on utilise des enregistrements de phrases étiquetées avec les unités linguistique correspondantes, i.e. les mots, les phonèmes. L'apprentissage statistique de ces deux composantes consiste à extraire les informations pertinentes à partir de données et ensuite à décrire une formulation paramétrique capable de représenter ces informations.

Du point de vue de l'apprentissage, on peut identifier trois modules principaux dans un système de reconnaissance de la parole :

- Extraction d'information, acoustique et/ou linguistique
  L'extraction d'information pour la modélisation acoustique est réalisée en utilisant les propriétés physiques des mécanismes de la parole. Les signaux acoustiques sont projetés dans une espace de paramètres où le contenu linguistique peut être identifié malgré les interférences liées aux variations environnementales, comme le locuteur, le bruit etc. Pour la modélisation du langage, l'information qui nous intéresse sont les séquences de mots possédant un contenu sémantique. Les séquences de mots simples, leurs propriétés sémantiques, syntaxiques et morphologiques sont aussi utilisées pour améliorer la modélisation.
- Modélisation probabiliste
  Dès que les informations pertinentes sont extraites des données, l'étape suivante consiste à définir une représentation cohérente de ces informations par des outils probabilistes. Ceci est fait en spécifiant les statistiques suffisantes et leur représentation par des formulations paramétriques simples. Les statistiques suffisantes et la représentation paramétrique doivent être choisies pour les accorder avec les données et pour permettre une complexité raisonnable. En reconnaissance de la parole, les approches les plus utilisées sont les modèles de Markov cachés pour la modélisation acoustique et les n-grammes pour la modélisation du langage.
- Adaptation statistique des modèles aux conditions d'application
  En général, l'apprentissage statistique induit un biais en faveur des données d'apprentissage. Un des problèmes rencontré dans ce cadre est le sur-apprentissage. Ce problème peut survenir à cause d'un déséquilibre dans les données ou d'une trop grand complexité du modèle. A cause de ce problème les propriétés de généralisation du modèle peuvent s'affaiblir. Ceci est un des problèmes fréquemment rencontrés dans le cadre de la reconnaissance de la parole, quand les modèles sont appris dans des conditions spécifiques. Ces modèles sont en général moins performants dans d'autres conditions. Dans ces circonstances, les techniques d'adaptation sont utilisées pour compenser les différences entre les conditions d'application et d'apprentissage.

Dans cette thèse, nous proposons de nouvelles stratégies pour ces trois composantes principales d'un système de reconnaissance de la parole. Nous traitons le problème de modélisation probabiliste avec un outil statistique puissant, les modèles probabilistes graphique [104]. Ce formalisme généralise la plupart des techniques probabilistes utilisées dans le traitement de la parole [1]. Précisément, nous proposons de nouvelles approches de modélisation pour les composantes acoustiques et linguistiques, la modélisation par des réseaux Bayesiens dynamiques (RBD). Les RBD constituent un sous-ensemble des modèles probabilistes graphiques. Ils incluent les modèles de Markov cachés et les n-grammes comme des instances particulières. La reformulation des modules de modélisation dans le formalisme des RBD, nous ouvre vers de nouvelles perspectives inexploitées auparavant. En plus des nouvelles approches pour la modélisation, nous proposons également de nouvelles stratégies pour l'extraction des paramètres acoustiques. Notre motivation principale dans ce domaine est de chercher des paramètres robustes qui ne sont pas liés à la modélisation par des HMMs. Pour le problème d'adaptation aux conditions d'application, nous proposons une nouvelle méthode de compensation prédictive. Nous proposons une stratégie d'apprentissage supervisée pour apprendre la variation des modèles de parole dans différentes conditions bruitées. Dans les parties suivantes, nous résumons notre contribution dans chacun de ces domaines.

1.2 Modélisation acoustique

La grande majorité des systèmes actuels de reconnaissance automatique de la parole (RAP) utilise une modélisation statistique du signal acoustique par des modèles de Markov cachés (ou HMM pour Hidden Markov Models). En effet, ces modèles ont permis d'obtenir les meilleurs résultats en reconnaissance de mots isolés, mots enchaînés et parole continue dans des conditions de laboratoire ou pour des tâches simples. En revanche, dans des conditions réelles de traitement de la parole (milieu bruité, parole spontanée, prononciations diverses...), les performances de ces systèmes se dégradent considérablement et leur utilisation se trouve très limitée. Pour cette raison, la conception de systèmes robustes et fiables capables d'affronter des situations réelles est un grand défi scientifique et technologique qui se pose dans le domaine de la RAP depuis plusieurs années.

Notre objectif dans ce domaine est de relever ce défi en s'attaquant au coeur du problème : la robustesse de la modélisation de la parole. En d'autres termes, notre stratégie scientifique consiste à concevoir des systèmes de RAP dont la robustesse est liée à

- la fidélité de la modélisation de la parole plutôt qu'à des améliorations ad-hoc des systèmes à base de HMM,
- une meilleure exploitation d'information contenue dans les données disponibles.

Ceci est motivé par le fait que les contre-performances des HMM dans les situations réelles sont principalement dues à leur manque de prise en compte des phénomènes acoustiques spécifiques à la parole. Il est donc primordial de développer des modèles qui capturent toutes les caractéristiques de la parole pour espérer obtenir des systèmes de RAP robustes.

Une des principales faiblesses de la modélisation HMM est l'hypothèse de dépendances entre les variables des processus observé et caché. Nous avons ainsi développé un algorithme de modélisation de la parole qui ne fait aucune hypothèse de dépendance *a priori* entre ces variables, mais plutôt qui apprend les dépendances à partir des données d'apprentissage. Cet algorithme est obtenu en formalisant le problème de modélisation dans le cadre de la théorie de *l'apprentissage structurel* des Réseaux Bayésiens [34]. Cet algorithme a l'avantage d'offrir une meilleure modélisation et discrimination de la parole (que les HMM), en outre il garantit de meilleures performances de reconnaissance par rapport aux HMM tout en permettant un contrôle sur la complexité souhaité du moteur de reconnaissance [36].

Nous avons aussi appliqué cette méthodologie de modélisation en reconnaissance de la parole continue, qui est un problème beaucoup plus difficile que la reconnaissance de mots isolés. Un des problèmes principaux que nous avons rencontré est celui du décodage, c.-à-d., l'identification de la séquence d'unités acoustiques qui explique au mieux les observations. Dans le contexte des HMMs ce problème est aisément résolu par des algorithmes bien connus. Dans notre cas, la difficulté surgit en raison de la variabilité des structures graphiques des modèles acoustiques.

Dans une première approche pour résoudre ce problème, nous avons proposé une méthode de construction d'un méta-modèle tel que toutes les unités acoustiques peuvent être représentées avec une structure unique. Le décodage se fait ensuite par des algorithmes standards d'inférence de Réseaux Bayésiens [39, 37]. Cette approche a cependant une complexité relativement élevée. Nous avons ainsi développé un algorithme rapide de décodage, basé sur la programmation dynamique, qui exploite les propriétés structurelles des modèles acoustiques utilisés [38].

1.3 Modélisation du langage

L'approche statistique pour la modélisation du langage tente de construire une représentation du langage en utilisant la distribution probabiliste empirique des événements linguistiques. La spécification d'un modèle de langage comporte la définition des variables implicites et/ou explicites du langage. L'évolution dynamique du langage est géré par ces variables et leurs interactions dans le temps. Chaque variable est en relation avec un certain nombre de facteurs qui constituent son contexte. En termes probabilistes, le contexte d'une unité linguistique est défini par des propriétés d'indépendance conditionnelle. Etant donnée son contexte, chaque unité linguistique est supposée être indépendante des autres événements linguistiques.

D'autre part, l'indépendance conditionnelle est la propriété essentielle des modèles probabilistes graphiques. En fait, grâce à l'exploitation de cette propriété on peut développer des algorithmes d'inférence efficaces et génériques pour les modèles graphiques. De plus, les modèles n-grammes et n-classes (et d'autres modèles du langage) sont des instances particulières des réseaux Bayesiennes dynamiques (RBD). C'est donc une idée naturelle de reconsidérer les modèles du langage dans le cadre des RBDs et de rechercher des profits potentiels dans ce domaine.

Dans l'approche classique basée sur les n-grammes, la probabilité d'observer un mot est conditionnée par l'historique récent des n-1 mots. Ces modèles modélisent bien le contexte lorsque les données d'apprentissage sont suffisantes. Les approches basées sur les classes de mot représentent l'historique par des classes de mots plutôt que des mots. Ceci a l'avantage de diminuer la complexité du modèle et par conséquent possède une meilleure propriété de généralisation pour les mots inconnus. Afin d'exploiter les caractéristiques complémentaires de ces approches plusieurs techniques de combinaison sont proposées en utilisant des méthodes d'interpolation ou des méthodes de back-off de mot-à-classe. Nous proposons de définir un modèle de langage plus général où nous intégrons ces deux approches classiques dans un cadre formel : les réseaux Bayesiens. Nos expérimentations initiales indiquent des améliorations potentielles par rapport aux approches classiques [42, 44, 43]. Les modèles proposés comportent un meilleur équilibre entre les propriétés de généralisation et la modélisation du contexte.

1.4 Paramétrisation du signal acoustique

Une des faiblesses principales des systèmes basés sur les HMM dans des conditions réelles d'application est le fait que la dynamique fréquentielle (qui est phonétiquement très importante) est faiblement modélisée dans ces systèmes. Ceci nous a motivé pour définir des modèles statistiques de la parole qui peuvent capturer à la fois la dynamique temporelle *et* fréquentielle.

Pour résoudre ce problème, il est nécessaire de travailler avec des paramétrisations qui sont localisées dans le domaine des fréquences et, de préférence, qui ont une interprétation physique claire. La paramétrisation la plus largement répandue dans les systèmes de reconnaissance de la parole est l'analyse cepstrale (ou MFCC pour Mel Frequency Cepstral Coefficients) [29]. Les MFCC ne possèdent pas d'interprétation physique consistante, et ils vivent dans un domaine qui ne permet pas l'analyse de la dynamique fréquentielle de la parole. Néanmoins, les paramétrisations basées sur les MFCC donnent les meilleures performances et il est difficile de les concurrencer.

En général, quand une nouvelle paramétrisation est proposée, les taux de reconnaissance peuvent être améliorés par rapport aux MFCC, mais seulement dans des conditions de parole propre ou bruitée, rarement dans les deux. Motivé par le désir d'établir des modèles temps-fréquence de la parole, nous avons étudié s'il était possible de concevoir des paramétrisations de la parole qui ont les avantages des MFCC mais pas leurs inconvénients.

Une technique de paramétrisation récemment développée qui semble être une candidate appropriée pour réaliser cet objectif est l'approche de filtrage fréquentiel (FF) [97, 99] . Le principe de cette technique est de filtrer les énergies spectrales avec des filtres linéaires du premier ou second ordre afin de les décorreler. Il a été montré que le FF mène à des performances semblables voir meilleures que les MFCC, dans des conditions de parole propre et artificiellement bruitée.

Néanmoins, nos expérimentations ont prouvé que la technique de FF présente quelques anomalies dans des conditions de bruits réels. Ainsi nous avons proposé une approche de filtrage de fréquence en utilisant des filtres d'ondelettes. Cette formulation mathématique nous permet d'analyser les propriétés de localisation avec une meilleure précision. Les expérimentations ont prouvé qu'il est possible de concevoir une paramétrisation de la parole qui est localisée dans le domaine fréquentiel, et qui a des performances au moins aussi bonnes que les MFCC dans toutes les conditions [24].

1.5 Compensation du bruit

Nous avons conçu une nouvelle technique pour l'adaptation au bruit qui ne fait aucune hypothèse sur la façon dont les sources de bruit affectent le signal de la parole et qui ne font pas intervenir les modèles de la parole propre. Cette nouvelle méthode fait l'hypothèse (réaliste) que des bases de données de la parole enregistrée dans différentes conditions sont disponibles. La philosophie de cette approche est de traiter ces bases de données afin de construire un "outil" qui permettra de manipuler de nouvelles conditions de bruit d'une manière robuste. Un des avantages principaux de cette approche est qu'elle est indépendante de la paramétrisation et de la modélisation acoustique.

Le principe de cette nouvelle approche d'adaptation prédictive est le suivant : Pour chaque condition de bruit, d'abord le modèle de la parole est appris en utilisant les données disponibles dans cette condition. Ensuite en utilisant les observations de bruit seules (de la base d'apprentissage), un modèle probabiliste qui représente la distribution du bruit est construit. Après le traitement de toutes les conditions de bruit, un modèle paramétrique qui décrit la relation entre le modèle de bruit et les modèles de parole dans chaque environnement bruité est appris. Ce modèle paramétrique nous permet ensuite de construire automatiquement le modèle pour un nouvel environnement sachant le modèle du bruit dans cet environnement.

Les principaux avantages de cette approche sont les suivants :
- Aucune hypothèse n'est faite sur la façon dont la source de bruit affecte le signal acoustique.
- Le méthode est complètement indépendante du type des modèles acoustiques et des paramétrisations employées pour traiter et modéliser la parole.
- Elle ne présuppose pas la disponibilité d'un modèle canonique initial de la parole. Seul le modèle paramétrique de transformation est requis.
- Comme les approches prédictives classiques, elle est algorithmiquement rapide.
- Elle peut être facilement "incrémentée" ou "mise à jour" avec de nouveaux environnement.

Nous avons évalué cette technique avec un modèle paramétrique linéaire simple. Les expériences ont montré qu'on peut obtenir des performances meilleures qu'avec des modèles multi-conditions qui sont les modèles les plus largement répandus dans ce genre de scénarios [24]. Les expériences que nous avons effectuées ont montrés que cette nouvelle technique est très intéressante et prometteuse.

Chapitre 2

Modélisation acoustique par des réseaux Bayésiens dynamiques

Sommaire

2.1 Introduction 9
2.2 Réseaux Bayésiens dynamiques 10
2.3 Classe de recherche de structures 11
2.4 Apprentissage de structures 12
2.5 Algorithme de décodage 13
2.6 Expériences 14
2.7 Conclusion 15

2.1 Introduction

La grande majorité des systèmes actuels de reconnaissance automatique de la parole (RAP) utilise une modélisation probabiliste du signal de parole par des modèles de Markov cachés (ou HMM pour Hidden Markov Models). Dans un HMM, les observations sont supposées être gouvernés par un processus dynamique caché. Les hypothèses d'indépendance associées sont telles que le processus caché est markovien du premier ordre et chaque observation dépend seulement de la variable cachée actuelle. Il y a cependant une question fondamentale concernant ces hypothèses de dépendance : sont-elles consistantes avec les données et avec tout type d'application ?

Dans [36], nous avons proposé une méthode dans laquelle nous ne faisons aucune hypothèse de dépendance *a priori*. Plutôt, nous donnons aux données une liberté complète (mais contrôlée) pour dicter les dépendances appropriées. En d'autres termes, nous apprenons les dépendances entre les variables (cachées et observables) à partir des données. Le principe de cette méthode est de rechercher toutes les dépendances "réalistes" possibles, et à choisir celles qui expliquent au mieux les données. Cette approche a l'avantage de garantir que le modèle résultant représente la parole avec une plus grande fidélité que les HMM. En outre, un contrôle est donné à l'utilisateur pour faire un compromis entre la fidélité et la complexité du modèle. De plus, l'approche est techniquement très attrayante parce que tout l'effort de calcul est fait dans la phase d'apprentissage.

Notre approche est basée sur le formalisme des réseaux Bayésiens dynamiques (ou DBNs pour Dynamic Bayesian Networks). La théorie des DBNs est une généralisation des réseaux

Bayésiens (ou BNs Bayesian Networks) aux processus dynamiques. Brièvement, le formalisme des réseaux Bayésien consiste à associer un graphe acyclique orienté à une distribution jointe de probabilités (ou JPD pour Joint Probability Distribution) $P(X)$ d'un ensemble de variables aléatoires $X = \{X_1..., X_n\}$. Les noeuds de ce graphique représentent les variables aléatoires, et les flèches codent les indépendances conditionnelles (IC) qui sont supposées dans le JPD. L'ensemble de toutes les relations d'IC que les propriétés de séparation du graphe impliquent, se nomment les propriétés de Markov. Un BN est complètement défini par une structure de graphe S et un jeu de paramètres Θ de probabilités conditionnelles des variables étant donné leurs parents. En effet, le JPD peut être exprimée sous une forme factorisé qui est, $P(X) = \prod_{i=1}^{n} P(X_i|\Pi_i)$, où Π_i dénote les parents de X_i dans S.

L'utilisation de DBNs dans la reconnaissance de la parole a suscité beaucoup d'intérêt ces dernières années [6, 27, 132]. Dans la section 3.3, nous avons brièvement récapitulé les différentes approches pour la modélisation acoustique. Dans ce chapitre, nous utilisons la flexibilité de cette approche, et au lieu de fixer à priori la structure des modèles acoustiques (comme on le ferait avec les HMM), nous établissons un système " intelligent " qui fonctionne comme suit. Nous alimentons le système en utilisant les données observées. Puis, le système détermine la structure (càd les dépendances) et les paramètres Θ qui représentent au mieux les données. Cette stratégie est connue comme l'apprentissage de structures dans la littérature des BNs. Dans [36], nous avons utilisé cette méthodologie d'apprentissage pour une tâche de reconnaissance de mots isolés. Nous étendons l'application de la méthodologie à la reconnaissance de mots connectés dans [39]. Dans ce chapitre, nous présentons une vue d'ensemble de l'approche et nous l'évaluons à travers un problème de reconnaissance de chiffres connectés.

Dans la section suivante, nous présentons une brève introduction aux réseaux Bayésiens dynamiques. Dans la section 2.3, nous définissons la classe de DBNs que nous utilisons dans notre configuration. Ensuite, nous récapitulons brièvement l'algorithme d'apprentissage de structures. Dans la section 2.5, nous décrivons l'algorithme de décodage pour la reconnaissance de la parole en utilisant les DBNs. Enfin, nous évaluons le potentiel de notre approche sur une tâche de reconnaissance de chiffres connectés.

2.2 Réseaux Bayésiens dynamiques

Un DBN code la distribution jointe de probabilités d'un ensemble de variables $X[t] = \{X_1[t], \ldots, X_n[t]\}$ évoluant dans le temps. Si nous considérons T pas de temps, le DBN peut être considéré comme un BN (statique) avec $T \times n$ variables. En utilisant la propriété de factorisation des BNs, la densité jointe de probabilités de $\mathbf{X}_1^T = \{X[1], \ldots, X[T]\}$ est :

$$P(X[1], \ldots, X[T]) = \prod_{t=1}^{T}\prod_{i=1}^{n} P(X_i[t]|\Pi_{it}) \tag{2.1}$$

où Π_{it} dénote les parents de $X_i[t]$. Dans la littérature des BNs, les DBNs sont définis en faisant l'hypothèse que $X[t]$ est un processus markovien [52]. Dans cet article, nous affaiblissons cette hypothèse pour permettre des processus non-markovien et pour que le processus $X[t]$ satisfasse à :

$$P(X_i[t]|\mathbf{X}_1^{t+\tau_f}) = P(X_i[t]|X[t-\tau_p], \ldots, X[t+\tau_f]) \tag{2.2}$$

pour tous les nombres entiers positifs τ_p et τ_f. Graphiquement, l'hypothèse ci-dessus déclare qu'une variable au temps t peut avoir des parents dans l'intervalle $[t-\tau_p, t+\tau_f]$. Cependant, un soin particulier doit être pris avec les variables aux limites (voir Section 2.5 pour les détails). Dans cette perspective, il est facile de représenter un HMM comme un DBN. En effet, les propriétés de Markov (les hypothèses de dépendance) d'un HMM, sont encodées par la structure graphique représentée sur le Figure 2.1. Chaque noeud de cette structure représente une variable aléatoire $X_h[t]$ ou $X_o[t]$, dont la valeur indique l'état ou l'observation au temps t.

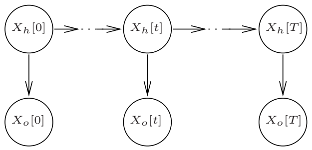


FIG. 2.1 – HMM représenté comme DBN

2.3 Classe de recherche de structures

L'apprentissage de structures de DBNs exige l'utilisation d'une classe de structures. La recherche à travers toutes les structures possibles de DBNs nécessite un temps de calcul exponentiel. Par conséquent, nous nous limitons à un petit mais riche ensemble de structures qui représente seulement des dépendances réalistes, d'un point de vue physique et algorithmique. Le lecteur peut se référer à la section 3.5 pour le raisonnement sur les dépendances autorisées qui sont définies comme suit. Soit $X[t] = \{X_h[t], X_o[t]\}$ l'ensemble de variables cachées et observées au temps t. Nous supposons que :

- la variable cachée au temps t est indépendante de $\mathbf{X}_1^{t-\kappa-1}$ sachant les κ dernières variables cachées, pour $t > \kappa$,

$$P(X_h[t]|\mathbf{X}_1^{t-1}) = P(X_h[t]|X_h[t-\kappa], ..., X_h[t-1]). \tag{2.3}$$

- la variable d'observation au temps t est indépendante de toutes les autres variables sachant les variables cachées dans l'intervalle $[t-\tau_p, t+\tau_f]$, pour tout nombres entiers positifs τ_p et τ_f,

$$\begin{aligned} P(X_o[t]|\mathbf{X}_1^T \setminus \{X_o[t]\}) = P(X_o[t]|X_h[t-\tau_p], \\ \ldots, X_h[t+\tau_f]). \end{aligned} \tag{2.4}$$

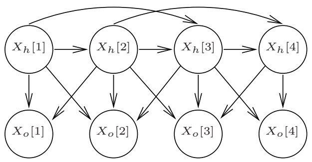


FIG. 2.2 – Structure de DBN $(\kappa, \tau_p, \tau_f) = (2, 1, 1)$, $T = 4$

Par conséquent, la classe de recherche des structures de DBN permises est définie par les triplets (κ, τ_p, τ_f) pour, $1 \leq \kappa \leq \kappa_{max}$, $0 \leq \tau_p \leq \tau_{p_{max}}$, $0 \leq \tau_f \leq \tau_{f_{max}}$, où $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$ est une limite supérieure à la taille de la classe de recherche. A la limite inférieure $(\kappa, \tau_p, \tau_f) = (1, 0, 0)$, la structure se réduit à un HMM standard d'ordre 1 (Figure 2.1) où, Eq.(2.3) définit les probabilités de transition d'état et l'Eq.(2.4) définit les probabilités d'observation. Chaque triplet (κ, τ_p, τ_f) spécifie une structure de DBN. La Figure 2.2 illustre le cas où $(\kappa, \tau_p, \tau_f) = (2, 1, 1)$.

Si chaque variable cachée discrète $X_h[t]$ prend ses valeurs dans l'ensemble ordonné d'étiquettes $I = \{1 \ldots N\}$, et chaque variable observable a une densité gaussienne conditionnelle, la paramétrisation numérique de notre DBNs est la suivante :

$$\begin{aligned} P_v(X_h[t] = j|\Pi_{ht} = \mathbf{i}) &= a_{\mathbf{i}j}[t], \ for\ j \in I \\ P_v(X_o[t]|\Pi_{ot} = \mathbf{i}) &\sim \mathcal{N}(\mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t]). \end{aligned} \tag{2.5}$$

L'index $\mathbf{i}$ prend ses valeurs dans l'ensemble des valeurs possibles des variables parents. Pour la classe spécifique des structures que nous considérons, les parents sont toujours les variables cachées, ainsi, $\mathbf{i}$ est un point dans l'espace cartésien I^m, où m est le nombre de parents de la variable considérée.

Afin de fournir des comparaisons justes dans les expériences, nous calculons le nombre de paramètres requis pour définir le DBN. Soit M le nombre de paramètres requis pour encoder la densité de probabilité d'observation. Pour une topologie de gauche à droite, le nombre exigé de paramètres pour encoder le réseau est donné par :

$$C = 2^\kappa N - \sum_{l=1}^{\kappa} l 2^{\kappa-l} + M \times [2^\beta N - \sum_{l=1}^{\beta} l 2^{\beta-l}]. \tag{2.6}$$

β est défini par $\beta = \tau_p + \tau_f$ (nous supposons que $\kappa, \beta \leq N$).

2.4 Apprentissage de structures

Dans le formalisme des BNs, le problème de l'apprentissage a deux aspects : l'apprentissage des paramètres sachant une structure et l'apprentissage de la structure. Dans notre situation nous utilisons l'approche du maximum de vraisemblance (ou ML pour Maximum Likelihood) pour le premier cas. Étant donné un DBN avec la structure S et un ensemble d'observations, les paramètres Θ sont choisis tels que la probabilité des observations soit maximisée. Ces paramètres peuvent être calculés en utilisant l'algorithme EM. Dans la section 3.6 nous dérivons les équations de mise à jour de EM pour la classe des DBNs que nous avons défini dans la section précédente.

L'apprentissage de la structure d'un BN est réalisé par une recherche dans un espace des structures possibles. La taille de cet espace de recherche est un problème critique. Dans la section précédente nous avons défini un ensemble de structures réalistes et plausibles pour modéliser la parole. Dans ce qui suit, nous récapitulons le principe général de l'algorithme d'apprentissage de structures. Les détails de l'algorithme et les références sont donnés dans la section 2.4.3. Étant donné un ensemble d'observations, notre but est de trouver la structure optimale définie par le triplet (κ, τ_p, τ_f), et les probabilités conditionnelles associées qui expliquent au mieux les données. L'optimalité est atteinte quand la probabilité des observations est maximale et la complexité de la structure est minimale. Il y a fondamentalement deux méthodes d'évaluations du degré auquel une structure s'adapte aux données : la métrique bayésienne de Dirichlet (BD), et le Minimum Description Length (MDL) (ou d'une manière équivalente le Bayesian Information Criterion (BIC)) [64]. Nous utilisons la métrique MDL, qui pénalise les structures complexes basées sur le nombre de paramètres employés pour coder le modèle. La métrique MDL est défini de la façon suivante :

$$Score_{MDL} = \log P(D|\Theta, S) - \frac{\log L}{2} \sum_{i=1}^{n} ||X_i, \Pi_i|| \tag{2.7}$$

où D est l'ensemble des observations, L est le nombre d'exemples (réalisations) dans D et $||X, Y||$ est défini comme le nombre de paramètres requis pour coder la probabilité conditionnelle, $P(X|Y)$. Le terme de vraisemblance peut être calculée en utilisant l'algorithme de JLO qui est un algorithme efficace d'inférence pour les réseaux Bayésiens [74]. Dans l'annexe A nous expliquons en détail l'algorithme de JLO pour l'inférence dans des BNs.

Afin de trouver le modèle optimal, nous employons l'algorithme EM structurel (SEM pour Structural Expectation Maximization) [49]. L'algorithme commence avec une structure quelconque et des paramètres initiaux. A chaque étape, les scores des structures candidates sont calculés avec la métrique MDL. La structure qui obtient le score maximal est choisie comme prochaine structure, et les paramètres de cette structure sont mis à jour avec une étape de EM paramétrique. En mettant à jour la structure itérativement à chaque étape, l'algorithme SEM garantit une augmentation des scores et une convergence vers un maximum local [49].

Dans notre application, nous initialisons l'algorithme avec $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$ comme la limite supérieure sur l'espace de recherche de structure et utilisons la structure de HMM pour la première itération. Cette initialisation garantit que le modèle résultant aura une fidélité plus élevée (ou égale), par rapport au HMM. Le compromis entre la complexité de l'algorithme d'apprentissage et la fidélité du modèle résultant est contrôlée par la limite supérieure sur l'espace de recherche.

2.5 Algorithme de décodage

Dans cette section, nous discutons brièvement l'algorithme de décodage dans des tâches de reconnaissance de mots isolés et connectés en utilisant la classe de DBNs que nous avons décrit. Soit un vocabulaire V de $|V|$ mots et un modèle de DBN pour chaque mot $v \in V$. L'algorithme d'apprentissage proposé nous permet de capturer les différentes structures de dépendance pour chaque mot. Par conséquent, les modèles appris ont (probablement) des structures $(\kappa^v, \tau_p^v, \tau_f^v)$ différentes pour chaque mot $v \in V$.

Considérons alors la tâche de reconnaissance de mots isolés. Sachant une séquence d'observation, le problème est de trouver le meilleur DBN tel que la probabilité des observations soit maximale. Dans les DBNs, la vraisemblance peut être calculée en utilisant l'algorithme JLO [74]. Cet algorithme est une version généralisée de l'algorithme Forward-Backward dans les HMMs [116]. Ainsi la classification est réalisée en opérant l'algorithme JLO sur chaque modèle de DBN et en choisissant celui qui donne le maximum de vraisemblance. Dans la section 3.8.1, nous présentons les résultats d'une tâche de reconnaissance de chiffres isolés en utilisant l'approche d'apprentissage décrite dans la section précédente.

La reconnaissance de mots connectés est une tâche plus compliquée. Le problème du décodage est d'identifier la séquence la plus vraisemblable de mots, étant donnée une phrase prononcée. A chaque instant, la probabilité de chaque mot doit être calculée et le meilleur ordre des mots doit être choisi pour maximiser la probabilité globale. Dans les HMMs, ceci est fait en utilisant l'algorithme de Viterbi. Dans notre configuration il y a deux étapes canoniques pour faire le décodage : 1) un algorithme pour trouver la meilleur séquence d'états dans un DBN sachant une séquence d'observation, 2) une méthodologie pour "augmenter" le modèle DBN avec le modèle de langage. La première étape est effectuée grâce à l'algorithme de Dawid [30] ce qui peut être

vu comme un version plus générale de l'algorithme de Viterbi. Pour la deuxième étape, nous proposons dans [39] une méthode pour construire un méta-modèle qui encapsule tous les DBNs de mots et le modèle de langage. Quand les DBNs des mots ont différentes structures, la construction du modèle augmenté a besoin d'une attention particulière. Nous construisons le modèle augmenté en utilisant une structure qui représente toutes les relations de dépendance dans tous les modèles de mots. Afin de respecter les différentes IC (Indépendances conditionnelles) de chaque DBN, nous employons une paramétrisation spéciale où nous codons les IC dans les paramètres du DBN plutôt que dans la structure. Une fois qu'un tel DBN est construit, l'algorithme de Dawid peut être employé sur ce DBN pour trouver la meilleure séquence de mots. Les détails de cette procédure peuvent être trouvés dans Section 3.7.3.

2.6 Expériences

Dans cette section, nous comparons [1] les exécutions des modèles à base de DBNs et à base de HMMs standards. Nos expériences sont effectuées sur la base de données *Tidigits*. Dans l'apprentissage, nous utilisons seulement la partie contenant les mots isolés de la base de données d'apprentissage où chaque locuteur prononce 11 chiffres ('o', $0, 1, \ldots 9$) deux fois. L'étape initiale a été de segmenter le corpus d'apprentissage aux régions de silence. Nous faisons un apprentissage isolé pour chaque modèle de HMM et de DBN utilisant cette première segmentation. Dans toutes les expériences, le silence est modélisé par un HMM à un état et une gaussienne. Dans les expérimentations, nous utilisons toute la base de données (de test) dans laquelle 8636 phrases sont prononcées, chaque phrase contenant entre 1 et 7 chiffres. La taille de chaque vecteur d'observation est 35. Ils se composent de 11 MFCCs (énergie enlevée), de 12 Δ, et de 12 $\Delta\Delta$. La matrice de covariance est supposée diagonale. Nous utilisons une topologie de transition de gauche à droite et un modèle uniforme de langage, càd, $P(v|v') = \frac{1}{12}$ ($|V| = 12$). Dans l'algorithme d'apprentissage de structures, pour chaque chiffre, la limite supérieur de l'espace de recherche est $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}}) = (2, 1, 1)$.

(κ, τ_p, τ_f)	N=4	N=5	N=6
(1,0,0)			'o',0,2,7
(1,0,1)	'o',2,4,7,8,9	'o',1,4,5,9	3,6,8
(1,1,0)	0,1,3,5,6	0,2,3,6,7,8	1,4,5,9

TAB. 2.1 – Résultats de l'algorithme SEM.

Nous effectuons des expériences en utilisant un nombre variable d'états, càd différentes valeurs pour N. Dans le Tableau 2.1, nous montrons les résultats de l'algorithme d'apprentissage de structures. Quand $N \geq 7$, notre système montre que le HMM standard est le meilleur modèle pour tous les chiffres. Ceci prouve que quand un HMM est suffisant pour modéliser des données, notre système le reconnaît. Les taux de reconnaissance sont donnés dans le Tableau 2.2, pour $N = 4, 5, 6$. Notre système surpasse en grande partie le système à base de HMMs, en particulier quand $N = 4$. Cette augmentation remarquable de l'exécution peut être expliquée comme suit. Puisque le nombre d'états cachés est petit, le système à base de HMMs conduit à beaucoup d'insertions. En utilisant notre système, tous les modèles appris impliquent des dépendances de contexte. Il est

[1] Des scores très bons peuvent être obtenus sur cette base de données en utilisant des mixtures de gaussiennes HMMs et des paramètres ajustés. Notre but ici n'est pas d'accorder les paramètres afin de réaliser les rendements les plus élevés. Nous voulons plutôt, fournir des comparaisons justes en utilisant des systèmes de base. Nous croyons que de cette façon, nous pouvons porter un jugement initial juste sur les capacités de chaque système.

bien connu que la modélisation du contexte améliore la modélisation de durée. Par conséquent, quoique le nombre d'états soit encore petit, notre système fournit une meilleure modélisation de la durée des chiffres, et le nombre d'insertions est ainsi énormément réduit. Cependant, pour faire des comparaisons justes, nous devrions comparer notre système à un HMM, avec un nombre équivalent de paramètres. Pour ce faire, nous comparons notre système à un HMM avec le même nombre d'états, mais nous utilisons un mélange de 2 gaussiennes comme densité de probabilité des observations. Le nombre moyen de paramètres C_{av} de chaque système est calculé en faisant la moyenne de la quantité C définie par l'Eq.(2.6) avec tous les modèles de chiffre [2]. Pour $N = 4, 5$, notre système ($C_{av} = 497,\ 639$) donne toujours de meilleurs résultats que le HMM_{2G} (qui utilise bien plus de paramètres que notre système $C_{av} = 567,\ 709$). Pour $N = 6$, bienque notre système utilise beaucoup moins de paramètres ($C_{av} = 653.6$) que HMM_{2G} ($C_{av} = 1551$), les deux systèmes donnent approximativement les mêmes résultats. Comme remarque finale sur ces expériences,

N	HMM(N)	DBN(N)	$\text{HMM}_{2G}(N)$
4	20.57	63.22	46.32
5	60.92	79.86	77.99
6	77.93	84.47	84.44

Tab. 2.2 – Précision d'identification des chiffres (%).

nous noterons que notre système permet l'apprentissage des processus non-Markoviens (le DBN (1,0,1) est non-Markovien, par exemple). Ceci est un avantage important par rapport aux HMMs. En effet, en faisant ainsi, notre système peut tenir compte du phénomène *d'anticipation* bien connu qui se produit dans le mécanisme de production de la parole. Ces expériences montrent la puissance de l'approche des DBNs pour modéliser les aspects acoustiques et phonétiques de la parole avec une fidélité plus élevée que les HMMs.

2.7 Conclusion

Nous avons utilisé le formalisme des DBNs pour construire des modèles acoustiques qui sont capables d'apprendre la structure de dépendance entre le processus caché et observé de la parole. Nous avons présenté une méthodologie pratique pour utiliser de tels modèles dans la reconnaissance de la parole continue. L'approche permet l'utilisation de modèles à différentes structures pour différents mots dans le vocabulaire. Nous avons montré qu'en utilisant nos modèles, nous pouvons obtenir de meilleurs résultats de reconnaissance qu'avec les HMMs équivalents.

[2] Pour 1 gaussienne $M = 70$, pour 2 gaussiennes $M = 140$.

Chapitre 3

Modélisation du langage par des réseaux Bayésiens dynamiques

Sommaire

3.1 Introduction **17**
3.2 Réseaux Bayésiens dynamiques **18**
3.3 RBDs pour la modélisation du langage **19**
3.3.1 Structures graphiques utilisées 20
3.3.2 Expérimentations et évaluation 21
3.4 Conclusion et perspectives **23**

3.1 Introduction

Le rôle d'un modèle de langage statistique est de modéliser les différents événements langagiers de la langue. Cette modélisation est assurée par l'affectation d'une probabilité à chaque séquence de mots hw_t (où h désigne l'historique du mot w_t). Cette probabilité est estimée à l'aide d'une chaîne de Markov d'un certain ordre (souvent 1 ou 2). Les n-grammes issus des chaînes de Markov sont les modèles les plus couramment utilisés. En partant du constat que certains mots ont un comportement similaire, leur regroupement en classes est envisageable et des modèles dits n-classes se substituent parfois aux n-grammes, et se combinent dans d'autres cas. L'objectif étant de réduire la complexité des modèles de base, de les généraliser, de passer outre les limites des données manquantes, et d'introduire des connaissances linguistiques plus importantes (morphologiques, syntaxiques, sémantiques et autres). L'exploitation de différentes connaissances linguistiques peut se faire par une simple combinaison linéaire de modèles de langage différents [73] ou par l'utilisation du maximum d'entropie [108].

Comme nous le verrons par la suite, les n-grammes et n-classes sont en fait des réseaux Bayésiens dynamiques (RBDs) très particuliers. Dans ce chapitre, nous proposons d'utiliser le formalisme des RBDs afin de mieux exploiter chacune des unités linguistiques considérées dans la modélisation. Nous développons une approche qui permet de traiter toutes ces unités dans une seule et même procédure et de fournir de nouveaux modèles de langage performants. Le principe de cette approche est de considérer des RBDs où une variable (qui peut être un mot ou une classe, ou tout autre unité linguistique) peut dépendre d'un sous-ensemble de variables. Ces liens de dépendance inter-unités linguistiques peuvent être déterminés automatiquement ou

manuellement. Notre objectif ultime est de proposer une méthode d'identification automatique à partir du corpus d'apprentissage du RBD optimal pour modéliser le langage. Pour atteindre cet objectif, nous étudions dans ce chapitre la faisabilité de cette approche en testant d'abord des modèles pour lesquels la topologie du RBD correspondant est fixé manuellement. Soulignons qu'un avantage de notre approche par rapport à l'interpolation linéaire est qu'elle ne fait pas de moyenne pondérée entre modèles fondés sur ces unités, mais intègre l'ensemble des unités linguistiques au sein d'une même structure qui ne fait pas la différence entre classes et mots. Comparé au principe du maximum d'entropie, on peut dire que notre approche conduit à des modèles qui ont une interprétation plus simple.

3.2 Réseaux Bayésiens dynamiques

Notre approche est fondée sur le formalisme des réseaux Bayésiens dynamiques (RBDs) qui est une généralisation des réseaux Bayésiens statiques aux processus dynamiques. Brièvement, un réseau Bayésien (statique) consiste à associer un graphe acyclique orienté à une distribution jointe de probabilités (DJP) $P(X)$ d'un ensemble de variables aléatoires $X = \{X_1 ..., X_n\}$. Les noeuds de ce graphe représentent les variables aléatoires, et les flèches codent les indépendances conditionnelles (IC) (supposées) existantes dans $P(X)$. Un réseau Bayésien ainsi est complètement défini par une structure graphique S et une paramétrisation numérique Θ de probabilités conditionnelles des variables sachant leurs parents. En effet, la DJP se factorise sous la forme :

$$P(X) = \prod_{i=1}^{n} P(X_i|\Pi_i),$$

où Π_i dénote les parents de X_i dans S.

Un RBD code la distribution jointe de probabilités d'un ensemble de variables $X[t] = \{X_1[t], \ldots, X_n[t]\}$ évoluant dans le temps. Si on se fixe T pas de temps, le RBD peut être considéré comme un réseau Bayésien (statique) avec $T \times n$ variables. En utilisant la propriété de factorisation, la densité jointe de probabilités de $\mathbf{X}_1^T = \{X[1], \ \ldots, X[T]\}$ est :

$$P(X[1], \ldots, X[T]) = \prod_{t=1}^{T} \prod_{i=1}^{n} P(X_i[t]|\Pi_{it}) \tag{3.1}$$

où Π_{it} dénote les parents de $X_i[t]$. Dans la littérature, les RBDs sont définis en faisant l'hypothèse que $X[t]$ est un processus markovien [52]. Dans cet notre approche, nous affaiblissons cette hypothèse pour permettre des processus non-markovien et pour que le processus $X[t]$ satisfasse :

$$P(X_i[t]|\mathbf{X}_1^{t+\tau_f}) = P(X_i[t]|X[t-\tau_p], \ldots, X[t+\tau_f]) \tag{3.2}$$

pour des entiers positifs τ_p et τ_f donnés. Graphiquement, l'hypothèse ci-dessus signifie qu'une variable au temps t peut avoir des parents dans l'intervalle $[t - \tau_p, t + \tau_f]$ (voir [36] pour les détails).

De ce point de vue, il est clair que les modèles de langage classiques peuvent être représentés comme des RBDs. Dans la section 4.5 et la section 4.4 nous avons analysé en détail ces modèles dans le cadre des RBD. En effet, les modèles n-gramme supposent que la probabilité d'une séquence de mots se factorise comme le produit des probabilités conditionnelles de chaque mot sachant son historique récent de $n-1$ mots. Précisément, si V est l'ensemble des mots et $w_1^T =$

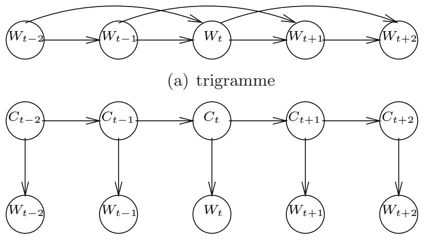
$\{w_1...w_T\} \in V^T$ est une séquence, on suppose que :

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t|w_{t-1},\ldots,w_{t-n+1}) \tag{3.3}$$

Ainsi, si W_t est une variable aléatoire discrète prenant ses valeurs dans V pour tout t, un n-grammes peut être représenté par le RBD de la figure 3.1-(a) (avec $n = 3$, i.e., un trigramme). Les modèles à base de classes représentent l'historique sur les classes de mots plutôt que sur les mots. Précisément, si $C = \{l_1, ..., l_m\}$ est l'ensemble des étiquettes de classe et $c_1^T = c_1...c_T \in C^T$ est une séquence de classes, on suppose que :

$$P(w_1^T, c_1^T) = \prod_{t=1}^{T} P(w_t|c_t)P(c_t|c_{t-1},\ldots,c_{t-n+1}). \tag{3.4}$$

Ainsi si C_t est une variable aléatoire discrète prenant ses valeurs dans C pour tout t, un n-classe peut être représenté par le RBD de la figure 3.1-(b) (avec $n = 2$, i.e., un biclasse).



(a) trigramme

(b) biclasse

FIG. 3.1 – modèles trigramme et biclasse

3.3 RBDs pour la modélisation du langage

Les n-grammes et n-classes sont les modèles les plus utilisés actuellement. Les n-grammes ont prouvé leur efficacité lorsqu'ils sont appris sur de gros corpus de données. Ces modèles ont comme inconvénient principal d'être algorithmiquement gourmand étant donné que l'on doit estimer $|V|^n$ probabilités où V est le vocabulaire et n l'ordre du modèle de langage. Malgré l'élagation opérée par les techniques de "cut-off", ces modèles demeurent complexes. Les modèles n-classes quant à eux sont moins complexes car le nombre de classes utilisées est souvent très inférieur à celui du vocabulaire. Ils permettent également une meilleure généralisation. Néanmoins, ils sont moins précis et leur perplexité est plus importante. C'est pourquoi ils sont souvent combinés avec les n-grammes. La procédure de mise en oeuvre d'un modèle de langage dans ce cas consiste à à déterminer *a priori* l'ordre du modèle(n). Ensuite, les deux modèles (classes et grammes) sont appris séparément en utilisant le maximum de vraisemblance comme critère d'apprentissage. L'étape suivante consiste à combiner linéairement les modèles ou d'intégrer leurs caractéristiques respectives dans une architecture du type maximum d'entropie.

Cette approche est intéressante, mais si nous voulons repousser les limites de ces modèles et mettre à profit leurs informations lexicales et syntaxiques, il est préférable de les considérer comme un tout indivisible devant être appris dans une même et unique procédure. Dans l'approche que nous proposons, nous souhaitons d'abord que "l'ordre" du modèle soit déterminé automatiquement à partir des données d'apprentissage. Ensuite, comme un mot est influencé par son contexte lexical et syntaxique, nous proposons de prendre en compte directement cette influence linguistique dans l'estimation du mot à prédire.

Le formalisme des RBDs nous offre le cadre théorique et algorithmique pour réaliser cet objectif. Notre idée principale est de ne faire aucune hypothèse *a priori* sur la façon de représenter le langage mais plutôt de considérer toutes les données disponibles (mots et classes) comme des observations du système dynamique $\{W_t, C_t\}$, notre but ensuite est de trouver le modèle qui décrit ces observations au mieux (en terme de perplexité). De cette façon, nous laissons les données dicter ce qui influence la prononciation d'un mot. Dans la terminologie des réseaux Bayésiens, ceci est le problème d'*apprentissage structurel* : trouver la structure graphique (et sa paramétrisation numérique) qui explique "au mieux" les données. Il existe des algorithmes [64] qui tente de résoudre ce problème (assez difficile) et qui fonctionnent plus au moins bien selon les applications. Notre but dans ce chapitre n'est pas d'utiliser de tels algorithmes (sachant que notre objectif ultime et de développer un algorithme d'apprentissage structurel qui soit bien adapté à la modélisation du langage). Notre but est plutôt de vérifier que notre approche peut effectivement concurrencer les n-grammes et n-classes classiques. Pour ce faire nous considérons un ensemble assez riche de structures graphiques que nous jugeons plausibles pour modéliser le langage. Ensuite, nous évaluons sur une application concrète les performances de plusieurs RBDs appartenant à cet ensemble en les comparant aux n-grammes et n-classes.

3.3.1 Structures graphiques utilisées

Pour définir un ensemble de structures graphiques plausibles pour la modélisation du langage, nous devons spécifier des hypothèses d'IC qui soient linguistiquement informatives et faciles à interpréter. Pour ce faire, nous commençons par assouplir l'hypothèse d'IC des n-classes en considérant qu'un mot peut dépendre non seulement de la classe courante mais aussi d'autres classes dans un contexte limité dans le passé et/ou le futur. Pour incorporer les propriétés des n-grammes, nous autorisons aussi la dépendance d'un mot de son historique (lexical). Enfin, nous considérons qu'une classe peut dépendre non seulement de son historique mais aussi du mot courant et/ou d'autres mots dans un contexte limité dans le passé et/ou le futur. La figure 3.2 montre un exemple de RBD possible dans notre modélisation.

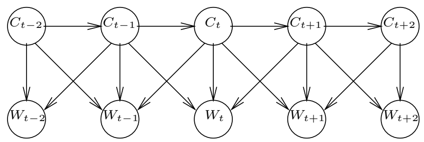


FIG. 3.2 – RBD où chaque mot (resp. classe) dépend de la classe antérieure, présente et futur (resp. antérieure).

La probabilité jointe associée à un modèle spécifique s'exprime comme :

$$P(w_1^T, c_1^T) = \prod_t P(w_t|\pi_{w_t})P(c_t|\pi_{c_t}) \tag{3.5}$$

où π_{w_t} (resp. π_{c_t}) est la réalisation des parents Π_{W_t} (resp. Π_{C_t}) de W_t (resp. C_t). Les paramètres du modèle sont donnés par les tables de probabilités conditionnelles (indépendantes du temps t) $P(w_t|\pi_{w_t})$ and $P(c_t|\pi_{c_t})$ que nous notons, pour $X_t \in \{W_t, C_t\}$, par :

$$\theta_{x,j,\mathbf{k}} = P(X_t = j|\Pi_{X_t} = \mathbf{k}). \tag{3.6}$$

Ces paramètres sont estimés en utilisant le critère du maximum de vraisemblance, ce qui donne :

$$\theta_{x,j,\mathbf{k}} = \frac{N_{x,j,\mathbf{k}}}{\sum_i N_{x,i,\mathbf{k}}} \tag{3.7}$$

où $N_{x,j,\mathbf{k}}$ est le nombre de réalisations $X_t = j, \Pi_{X_t} = \mathbf{k}$. Les schémas classiques de “discounting”, de “smoothing” et de “back-off” restent applicables à ce type de modèles. Voir Section 4.7.1 pour les detailles.

3.3.2 Expérimentations et évaluation

Les corpus d'apprentissage et de test sont extraits du journal *Le monde*. Nous avons utilisé 22M de mots pour l'apprentissage et 2M de mots pour le test. Le vocabulaire est constitué des 5000 mots les plus fréquents. Le corpus d'apprentissage a été étiqueté automatiquement par un ensemble de 200 classes syntaxiques définies manuellement [115]. Tous les modèles utilisés dans cette expérimentation ont été lissés par la méthode de “absolute discounting” [101].

La table 3.1 montre les perplexités que nous avons obtenues pour 14 réseaux Bayésiens dynamiques différents. Les modèles RBD1, RBD2 et RBD3 sont respectivement le bigramme, le biclasse et le triclasse. Afin d'atteindre l'objectif de trouver de meilleurs modèles de langage, nous avons pris le biclasse comme modèle noyau que nous avons enrichi incrémentalement par un contexte lexical et/ou syntaxique plus important. Nous avons introduit également la notion du contexte droit d'un mot. C'est le cas du modèle RBD6 qui intègre non seulement le contexte gauche de la classe d'un mot mais aussi son contexte syntaxique droit. On obtient ainsi une amélioration de $16,6\%$ par rapport à RBD4, ce qui prouve l'importance de la prise en compte du contexte droit. Il est vrai que linguistiquement parlant cela ne constitue nullement une surprise. En reconnaissance de la parole en revanche, il est difficile d'envisager la prise en compte du contexte droit mais cela peut être rendu possible grâce à un décodage multi-passes. D'autre part le modèle RBD5 montre que l'influence du contexte gauche est importante, c'est pourquoi son omission fait baisser les résultats de $7,8\%$. Une réduction considérable dans la perplexité est observée lorsque on fait dépendre un mot non seulement de son contexte syntaxique mais aussi lexical. Ainsi, RBD11 apporte une amélioration de $24,6\%$ par rapport à RBD4. Ceci confirme bien que l'historique lexical est indispensable et montre que l'historique syntaxique apporte une contribution indéniable.

En poussant cette stratégie encore plus, nous sommes arrivés à concevoir un modèle qui non seulement est largement meilleur que le biclasse mais qui est meilleur que le bigramme aussi. En effet, le modèle RBD14, dont la structure graphique est montrée dans la figure 3.3, améliore de $57,9\%$ par rapport à RDB2 et de $2,4\%$ par rapport à RBD1 (le bigramme). Même si l'amélioration par rapport au bigramme n'est pas considérable (dans cette application), ce résultat (en plus des précédents) montre surtout que notre approche peut effectivement conduire à de nouvelles catégories de modèles de langage qui peuvent concurrencer les catégories classiques.

	structure (ou DJP) du RBD	Perplexité
RBD1	$\prod_t P(w_t\|w_{t-1})$	65.24
RBD2	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1})$	151.31
RBD3	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1}c_{t-2})$	130.00
RBD4	$\prod_t P(w_t\|c_t, c_{t-1})P(c_t\|c_{t-1})$	113.13
RBD5	$\prod_t P(w_t\|c_t, c_{t+1})P(c_t\|c_{t-1})$	121.98
RBD6	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1})$	94.35
RBD7	$\prod_t P(w_t\|c_t, c_{t-1})P(c_t\|c_{t-1}, c_{t-2})$	97.19
RBD8	$\prod_t P(w_t\|c_t, c_{t+1})P(c_t\|c_{t-1}, c_{t-2})$	104.8
RBD9	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1}, c_{t-2})$	81.06
RBD10	$\prod_t P(w_t\|w_{t-1}, c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1})$	78.00
RBD11	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1})$	85.20
RBD12	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1}, c_{t-2})$	73.20
RBD13	$\prod_t P(w_t\|w_{t-1}, c_{t-1})P(c_t\|w_t)$	70.86
RBD14	$\prod_t P(w_t\|w_{t-1}, c_{t-1}, c_{t-2})P(c_t\|w_t)$	**63.67**

TAB. 3.1 – Perplexité des réseaux Bayésiens dynamiques

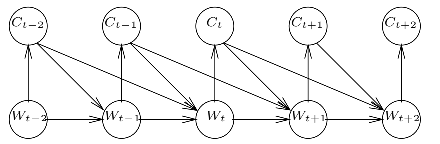


FIG. 3.3 – RBD donnant une perplexité plus faible que le bigramme.

3.4 Conclusion et perspectives

Nous avons présenté une nouvelle approche pour la construction de modèles de langage fondée sur le formalisme des réseaux Bayésiens dynamiques. Cette approche possède plusieurs avantages par rapport aux techniques classiques. Tout d'abord, elle permet d'inférer le meilleur modèle pour le langage traité à partir des corpus d'apprentissage, le modèle tente ainsi d'expliquer au mieux les données et n'est pas contraint par des hypothèses *a priori*. En outre, elle incorpore en une seule et même procédure toutes les unités linguistiques considérées dans la modélisation. Les modèles qui en résultent sont ainsi consistants et faciles à interpréter. Dans cet article, nous avons choisi de tester plusieurs type de RBDs pour évaluer le potentiel de notre approche. Les résultats montrent qu'elle est prometteuse et mérite d'être plus approfondie. D'autres expérimentations du même genre doivent évidemment être effectuées pour mieux analyser le comportement de cette technique. Mais notre objectif majeur est de développer un algorithme qui permet d'inférer *automatiquement* à partir des corpus d'apprentissage le meilleur RBD pour modéliser le langage traité. Ceci fera l'objet de nos futurs travaux.

Chapitre 4

Compensation prédictive pour la robustesse au bruit

Sommaire

4.1	**Introduction**	25
4.2	**Architecture de la compensation supervisée**	26
4.3	**Cadre expérimental**	28
4.4	**Résultats et discussion**	28
4.5	**Conclusion**	30

4.1 Introduction

La robustesse au bruit est un problème très difficile auquel sont confrontés les systèmes de reconnaissance de la parole dans les applications concrètes. Plusieurs techniques [60] ont été proposées pour améliorer les performances de la reconnaissance en présence de disparité entre les conditions d'apprentissage et celles de l'application. Ces techniques peuvent être classifiées en deux catégories : celles fondées sur le pré-traitement du signal de la parole (RASTA [67] ou amélioration de l'intelligibilité [45] par exemple) et les techniques de compensation. Dans ces dernières, des modèles acoustiques initiaux (généralement les modèles de parole propre) sont transformés pour représenter le nouvel environnement. Les techniques de compensation peuvent elles mêmes être classifiées en deux catégories : les schémas *adaptatifs* et *prédictifs*. La compensation adaptative, telle que MAP[55] et MLLR [85], consiste à utiliser la parole bruitée observée (au test) pour transformer les modèles acoustiques initiaux. Un inconvénient de cette approche est qu'elle nécessite au moins quelques dizaines de secondes de parole pour fournir de bonnes estimations des nouveaux modèles. Un autre inconvénient plus important est la nécessité d'une transcription lorsque le mode d'adaptation est non supervisé. Ceci est généralement fait à l'aide des modèles initiaux, ce qui limite les performances si la précision de la transcription est faible. La compensation prédictive (PMC [54] par exemple) ne repose pas sur les données de parole bruitée mais utilise plutôt les observations du bruit pour estimer les modèles dans le nouvel environnement. Les nouveaux modèles de la parole sont alors une combinaison entre les modèles initiaux et un modèle (paramétrique) du bruit estimé à partir d'observations de ce bruit. La combinaison est obtenue par une fonction qui modélise la contamination de la parole par les sources de bruit (bruit additif ou/et convolutif...). Un inconvénient est que cette modélisation peut être irréaliste et conduire ainsi à des estimations erronées des distributions de la parole bruitée. Un

autre inconvénient de la compensation prédictive est sa forte dépendance de la paramétrisation (MFCC en général) et du type de modèles probabilistes (HMMs en général) utilisés.

Il est important de souligner la complémentarité entre la compensation adaptative et prédictive surtout lorsque la parole est contaminée par un bruit de fond. En effet, dans ce cas les observations du bruit peuvent être utilisées dans un schéma prédictif pour fournir un "meilleur" modèle initial (que celui de la parole propre) pour la compensation adaptative.

Outre les inconvénients que nous avons mentionnés, une importante limitation de la compensation prédictive est la nécessité de disposer d'un modèle initial spécifique (généralement celui de la parole propre), supposant ainsi que les données d'apprentissage ont été collectées dans un environnement spécifique. Cependant, de nos jours il est plus courant de disposer de données enregistrées dans plusieurs environnements acoustiques. Par exemple, pour les systèmes de RAP en voiture, les données sont collectées en utilisant différentes voitures, avec fenêtres ouvertes et fermées, avec radio allumée et éteinte...etc. Dans ces scénarios, la compensation prédictive n'est pas le meilleur moyen pour traiter le problème de mismatch. La technique la plus utilisée dans ces cas (car elle donne en général les meilleurs résultats) est celle de la reconnaissance *multi-conditions*. Cette dernière consiste à utiliser lors du test les modèles appris sur toutes les données disponibles (dans toutes les conditions), ces modèles représentent alors un environnement acoustique "moyen" qu'on espère proche de celui du test.

Dans notre approche nous considérons que les données de parole ont été enregistrées dans différentes conditions de bruit. Nous développons ensuite une nouvelle architecture de compensation prédictive, que nous appelons compensation *supervisée* et qui est bien adaptée à ce type de scénarios. Le principe de cette nouvelle architecture est le suivant. En premier lieu, pour chaque condition de bruit, les modèles acoustiques sont appris en utilisant les données correspondantes à cette condition, nous les appelons modèles *matchés*. Ensuite, nous utilisons les observations du bruit correspondant (extraites de la base d'apprentissage) pour construire un modèle probabiliste qui tente de représenter la distribution du bruit, nous l'appelons modèle du *bruit*. Après avoir traité toutes les conditions, nous procédons à l'apprentissage supervisé d'un modèle paramétrique (prédéfini) qui tente de décrire la variation des modèles matchés par rapport à ceux du bruit. Lors du test, les observations du bruit de l'environnement de l'application sont utilisées pour estimer le modèle du bruit correspondant, ce dernier est ensuite donné comme argument au modèle paramétrique pour fournir des modèles estimés que nous espérons proches des modèles matchés (à l'environnement de l'application).

Ce chapitre est organisé de la façon suivante. Dans la prochaine section, nous commençons par développer l'architecture générale de la compensation supervisée. Nous définissons ensuite la procédure particulière que nous utilisons dans cet article. Dans la section 4.3, nous posons le cadre expérimental pour nos évaluations. La section 4.4 est consacrée aux résultats et leur analyse.

4.2 Architecture de la compensation supervisée

Nous avons décidé de considérer la compensation supervisée comme faisant partie de la classe des techniques de compensation prédictive en raison de deux propriétés fondamentales qu'elles partagent. Tout d'abord, les deux se basent sur l'hypothèse (très réaliste) que les meilleures performances de reconnaissance sont atteintes lorsqu'il n'y a pas de mismatch entre les conditions d'apprentissage et celles du test. Ensuite, les deux requièrent des modèles du bruit pour faire la compensation, *aucune* donnée de parole dans le nouvel environnement n'est nécessaire. Nous développons maintenant l'architecture générale de la compensation supervisée.

Nous supposons la donnée d'un ensemble $D_1, ..., D_K$ de bases d'apprentissage enregistrées respectivement sous les conditions de bruit $n_1, ..., n_K$. Soit W l'ensemble des unités acoustiques (mots, phonèmes...) que nous cherchons à modéliser. Pour chaque $w \in W$ et$k \in \{1, ..., K\}$, nous notons $\lambda_w(k)$ le modèle matché pour w appris en utilisant la base D_k, c.à.d. sous la condition du bruit n_k. Si N représente la variable bruit, l'idée principale derrière notre architecture est de supposer qu'il existe une fonction paramétrique f_w telle que :

$$||\lambda_w(N) - f_w(N)|| \cong 0, \quad \text{pour une certaine norme} \quad ||.||.$$

Les paramètres de f_w sont estimés par un apprentissage supervisé en utilisant l'ensemble des entrées-sorties $\{n_k, \lambda_w(k) : k = 1, ..., K\}$. Lors de la reconnaissance, un nouveau bruit n^* est observé et est donné comme entrée à chaque f_w pour calculer le modèle, que nous appelons *f-estimé*, $\tilde{\lambda}_w(n^*) = f_w(n^*)$. Notre but étant que $\tilde{\lambda}_w(n^*)$ soit "assez proche" du modèle $\lambda_w(n^*)$ matché au bruit n^*. Cette nouvelle architecture possède plusieurs avantages par rapport aux techniques de compensation usuelles :

- Elle ne fait aucune hypothèse sur la façon dont le bruit contamine la parole.
- Elle est complètement indépendante de la paramétrisation et du type de modèles probabilistes utilisés pour traiter et modéliser la parole.
- Elle ne requière pas la disponibilité d'un modèle initial de la parole. Seuls les paramètres de f_w sont requis pour estimer les modèles de la parole dans le nouvel environnement.
- Elle est algorithmiquement très rapide.
- Elle peut être facilement incrémentée dans le sens où si une nouvelle base D_{K+1} est disponible, seul $\{n_k, \lambda_w(k) : k = 1, ..., K\}$ doit être stocké en mémoire pour tenir en compte cette nouvelle information et mettre à jour l'estimation de f_w. Ceci est un avantage majeur par rapport à l'apprentissage multi-conditions. En effet, pour tenir en compte D_{K+1}, ce dernier nécessiterait de stocker tout $\{D_1, ..., D_K\}$ et réapprendre les modèles en utilisant $\{D_1, ..., D_K, D_{K+1}\}$. Ceci est évidemment très lourd en terme de complexité et de mémoire.

Le choix de la forme des f_w est défini par l'utilisateur selon ses connaissances a priori, les contraintes algorithmiques et les spécificités de l'application. Dans le reste de ce papier nous nous proposons de considérer que f_w est linéaire. Nous verrons que même avec un choix aussi simple, les résultats sont intéressants et prometteurs. Nous utilisons des HMM et des mono-Gaussiennes pour modéliser respectivement la parole et le bruit. Pour compléter la définition de notre architecture, nous devons spécifier des représentations pour $\lambda_w(k)$ et n_k. Pour ce dernier, nous choisissons simplement le vecteur moyenne ν_{n_k} de la Gaussienne qui modélise la distribution des MFCC statiques de n_k. Pour $\lambda_w(k)$, nous choisissons les moyennes du mélange de Gaussiennes comme étant les paramètres à transformer (comme c'est le cas généralement dans les techniques de compensation). Pour une discussion empirique sur le choix des paramètres voir Section 6.5. Formellement, pour chaque $w \in W$, si $\mu_{wls}(n_k)$ est le vecteur moyenne de la composante l du mélange à l'état s, nous définissons f_w comme étant l'ensemble $\{A_{wls}, B_{wls}\}$ de matrices de régression et de biais du modèle linéaire :

$$\mu_{wls}(n_k) = A_{wls}\nu_{n_k} + B_{wls}. \tag{4.1}$$

L'apprentissage de $\{A_{wls}, B_{wls}\}$ est fait de façon supervisée par le critère des moindres carrés. Lors de la reconnaissance, un nouveau ν_{n^*} est estimé à partir d'observations de n^*, le modèle (4.1) est ensuite appliqué pour calculer la f-estimation $\tilde{\mu}_{wls}(n^*)$. Pour les matrices de transitions, les poids des composantes du mélange et les covariances, nous gardons ceux du modèle de la parole propre. Nous soulignons ici que ceci n'implique en aucune façon que le modèle de la parole propre est un prérequis dans notre architecture. Ceci est juste un choix dans le cadre de ce travail préliminaire pour montrer le potentiel de notre approche.

4.3 Cadre expérimental

Nos expérimentations sont effectuées sur la partie "man" du corpus de chiffres connectés TIDIGITS. La base d'apprentissage (resp. test) est constituée de 4235 (resp. 428) séquences prononcées par 55 (resp. 56) locuteurs. Chaque séquence contient entre 1 et 7 chiffres. Pour créer la base de parole bruitée (à l'apprentissage et au test), nous utilisons 15 enregistrements différents de bruit de la base NOISEX (http ://spib.rice.edu/spib/select_noise.html). Les séquences bruitées sont obtenues en additionnant des segments de bruit à la parole propre. Pour chaque séquence, un segment est choisi aléatoirement à partir de l'enregistrement complet du bruit. Le niveau du bruit est ajusté par un facteur positif α. Précisément, si $s(t)$ est le signal de parole propre et $n(t)$ est le bruit sélectionné, le signal de parole bruitée est :

$$y(t) = s(t) + \alpha n(t). \tag{4.2}$$

Le choix de α détermine le rapport signal sur bruit (RSB) moyen de la base bruitée. Le RSB est défini comme étant 10 fois le rapport logarithmique entre la puissance du signal et celle du bruit. Quand cette définition est appliquée directement pour calculer le RSB d'une séquence de parole, les segments de silence faussent le calcul. En effet, comme le bruit est très dominant dans ces segments, le RSB calculé ainsi sera faible pour les séquences contenant beaucoup de silence. Pour remédier à ce problème, nous calculons le RSB sur les seuls segments de parole, c.à.d., en excluant les régions de silence. Ceci nécessite une segmentation du corpus en régions de parole et de silence. Cette segmentation est obtenue par alignement forcé en utilisant les HMMs appris sur le corpus de la parole propre.

Le front-end et le back-end sont basée sur les spécifications du consortium AURORA3 [68]. Les vecteurs acoustiques sont obtenus utilisant le front-end standard ETSI SQL W1007 [68] qui est une analyse cepstrale où 13 MFCC sont extraits à partir de 23 bancs de filtre Mel. Cette analyse est effectuée en appliquant une fenêtre de Hamming sur des trames de 25ms et un shift de 10ms. Un filtre de préaccentuation avec $a = 0.97$ est appliqué. Les vecteurs sont ainsi constitués de 12 MFCC (c_0 est exclu) plus la trame d'énergie logarithmique, les Δ et $\Delta\Delta$ sont aussi inclus. Les onze chiffres (de 0 à 9 plus le "oh") sont modélisés comme mots par des HMMs à 16 états émetteurs par mot avec une topologie gauche-droite sans saut. La densité de chaque état est modélisée par une mélange de 3 Gaussiennes à matrices de covariance diagonales. Deux modèles de silence sont utilisés : un HMM à 3 états pour les pauses avant et après chaque séquence, et à un seul état pour les pauses entre mots. Nous utilisons la même paramétrisation pour les signaux de bruit, et nous modélisons chaque source $\alpha n(t)$ par une mono-Gaussienne apprise sur l'enregistrement complet du bruit $n(t)$. Dans toutes nos expérimentations, le software HTK est utilisé pour la paramétrisation, l'apprentissage des modèles et la reconnaissance.

4.4 Résultats et discussion

Dans cette section nous évaluons les performances de notre architecture de compensation prédictive. Dans toutes les expérimentations nous avons choisi $\alpha = 0.05$ pour construire les bases d'apprentissage $D_1, ..., D_{K=15}$ (avec $\alpha = 1$ les bruits dominent la parole ce qui conduit à des performances très faibles des modèles matchés). Nous commençons l'évaluation par le cas où le mismatch est en terme du RSB, c.à.d., lorsque le type du bruit de testa été rencontré à l'apprentissage mais le RSB est différent (une situation souvent rencontrée dans les applications réelles). Le test est effectué en utilisant des facteurs de bruit plus élevés ($\alpha = 0.3, 0.4$ et 0.5) ce qui conduit à des niveaux de RSB plus faibles comparé aux conditions d'apprentissage. Les

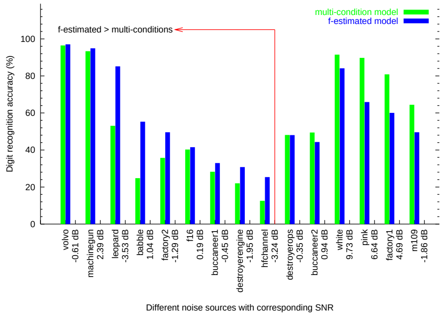


FIG. 4.1 – Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.3$.

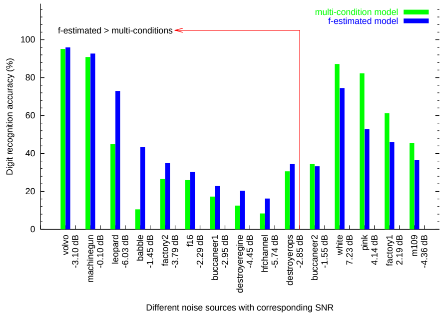


FIG. 4.2 – Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.4$.

figures 4.1,4.2 et 4.3 montrent les performances de la reconnaissance multi-conditions et celles de la compensation supervisée. Sur l'axe des abscisses, les sources de bruits sont groupées en deux ensembles : le premier (resp. second) est composé de bruits pour lesquels le modèle f-estimé (resp. multi-conditions) donne de meilleurs taux de reconnaissance que le modèle multi-conditions (resp. f-estimé). La première observation frappante est que notre modèle f-estimé donne de meilleurs résultats (parfois de façon considérable) que le multi-conditions dans la majorité des cas. Ceci montre que, dans cette application, notre architecture est "globalement" plus robuste au bruit que la reconnaissance multi-conditions. En outre, plus le RSB décroît plus le nombre de cas où le modèle f-estimé est meilleur augmente. En effet, parmi 15 bruit il y a 9, 10 puis 11 cas pour lesquels le f-estimé est meilleur que le multi-conditions (voir figures 4.1, 4.2 et 4.3, respectivement). Ceci suggère que la compensation supervisée est plus robuste à la décroissance du RSB que la reconnaissance multi-conditions. On peut aussi noter que plus le RSB décroît plus cette dernière "perd" un bruit en faveur de notre approche, ce qui suggère une certaine stabilité de notre procédure.

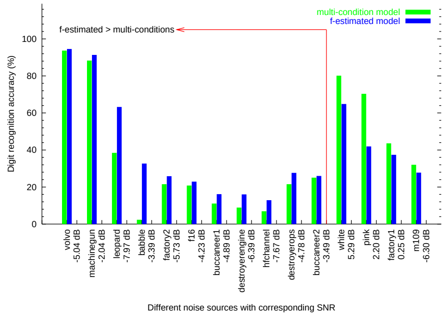


FIG. 4.3 – Comparaison entre les performances du modèle multi-conditions et celles du modèle f-estimé pour $\alpha = 0.5$.

Dans la dernière expérimentation, nous évaluons les propriétés de généralisation de notre architecture lorsque le mismatch est en termes de RSB **et** de type de bruit. Pour ce faire, nous sélectionnons un bruit cible sur lequel les tests seront effectués. Le modèle matché à ce bruit est alors exclu de l'ensemble des entrées-sorties pour l'apprentissage de la régression (4.1). Le corpus correspondant à ce bruit (c.à.d. les données contaminées par ce bruit) est aussi exclu lors l'apprentissage du modèle multi-conditions. La table 4.1 montre les taux de reconnaissance obtenus à différents niveaux de RSB pour le bruit cible *machinegun*. Les taux des modèles matchés (à la condition du bruit *machinegun*) correspondants sont donnés comme références. Ces résultats montrent que, même si aucune information sur le bruit cible n'a été utilisée lors del'apprentissage, les performances de la compensation supervisée sont nettement supérieure à celles de la reconnaissance multi-conditions. Ceci suggère que la compensation supervisée peut être plus robuste (que la reconnaissance multi-conditions) non seulement en terme de variation du RSB mais aussi en terme de nouveaux types de bruit non rencontrées lors de l'apprentissage.

RSB	matché	multi-conditions	f-estimé
$\alpha = 0.05$	98.33	95.54	**97.56**
$\alpha = 0.3$	98.05	82.38	**88.86**
$\alpha = 0.4$	97.84	78.41	**84.75**
$\alpha = 0.5$	97.49	75.49	**82.03**

TAB. 4.1 – Comparaison entre les taux de reconnaissance (%) du modèle matché, multi-conditions et f-estimé pour le bruit cible *machinegun*.

4.5 Conclusion

Nous avons développé une nouvelle architecture de compensation prédictive qui présente plusieurs avantages par rapport aux techniques classiques. Nous avons montré que malgré des choix très simples, cette architecture peut conduire à des performances supérieures à celles de la reconnaissance multi-conditions. Évidemment, en vue des expérimentations préliminaires que nous avons effectuées, nous ne pouvons pas faire de conclusions définitives. Nous pouvons cependant affirmer que cette nouvelle technique paraît prometteuse et mérite d'être plus approfondie. En effet, plusieurs pistes restent encore à explorer pour exploiter le potentiel de cette technique : choix de la forme de f_w et du critère d'estimation de ses paramètres, choix de la représentation pour $\lambda_w(k)$ et n_k, choix des caractéristiques à transformer, choix du modèle de bruit...etc. Ceci fera l'objet de nos travaux futurs.

Chapitre 5

Conclusion

Dans cette thèse nous avons élaboré quatre composantes fondamentales de tout système de reconnaissance automatique de la parole : la modélisation acoustique, la modélisation du langage, la paramétrisation du signal acoustique et la compensation du bruit. Nous avons utilisé des techniques nouvelles dans chacun de ces domaines, et apporté des perspectives novatrices.

Une des contributions fondamentales est l'utilisation des réseaux Bayésiens pour la modélisation probabiliste dans un système de RAP. En fait, la plupart des techniques probabilistes utilisées dans le traitement de la parole peut être unifiées dans ce formalisme. Ainsi, nos approches unifient et généralisent les approches classiques comme les HMMs et les n-grammes qui sont les modèles les plus utilisés dans les systèmes actuels.

Notre contribution dans le domaine de la modélisation acoustique consiste à utiliser le paradigme de l'apprentissage structurel pour extraire les relations de dépendance appropriées entre les processus cachés et observés de la parole. Cette approche nous permet de modéliser chaque unité acoustique avec ses propriétés d'indépendance conditionnelle, et ce de manière individuelle. Nous avons défini une classe de modèles regroupant diverses propriétés de dépendance. Puis nous avons appliqué l'algorithme de maximisation de l'espérance structurelle, dans le cadre des réseaux Bayésiens dynamiques. Comme chaque unité acoustique est apprise en utilisant un tel algorithme, un des avantages de notre approche est que les données contrôlent la complexité des modèles. Une des solutions les plus simple qui puisse être choisies par notre algorithme, si les données d'apprentissage le permettent, est la structure de HMM.

Afin de réaliser cette approche pour la RAP, nous avons développé les outils nécessaires dans le cadre des réseaux Bayésiens. Nous avons défini une nouvelle représentation des réseaux Bayésiens dynamiques pour les processus non Markovien afin d'identifier les relations de dépendance entre les observations de la parole et les futurs états du processus caché. Nous avons aussi développé un algorithme de décodage qui exploite les propriétés de notre classe de modèles. Les résultats des expérimentations en reconnaissance de la parole, isolée et continue, ont montré que notre approche est plus performante que l'approche HMM, à complexité équivalente. L'utilisation des réseaux Bayésiens dynamiques a attiré l'intérêt de la communauté de la RAP, et de nouvelles approches basées sur ce formalisme ont commencé à apparaître de plus en plus. De plus, notre approche se place comme une des rares approches initiatrices.

Une autre composante de la RAP nécessitant une modélisation probabiliste est le modèle de langage. Dans ce domaine, nous avons utilisé le formalisme de RBD afin de développer un cadre récapitulatif pour des modèles n-grammes et n-classes. En fait, les modèles n-grammes et n-classes ont des propriétés complémentaires. Notre approche est basée sur l'exploitation de chaque unité linguistique, i.e. mot, classe etc. . ., dans un seul modèle afin d'unifier les caractéristiques

complémentaires de ces modèles. Nous avons analysé les propriétés d'indépendance conditionnelle des approches classiques et nous avons développé un ensemble de modèles RBD qui les généralise. En faisant ainsi, nous pouvons construire un modèle unique qui a les avantages à la fois des n-grammes et des n-classes. Nous avons réalisé des expérimentations préliminaire, et montré les avantages potentiels de cette approche. Nos perspectives de travail concernant notamment le développement d'un algorithme d'apprentissage pour extraire la structure des RBD en utilisant des critères linguistiques.

Un autre axe de recherche de ma thèse porte sur les algorithmes d'extraction des paramètres acoustiques pour la RAP. Les paramètres actuels sont conceptuellement liés aux modèles de Markov cachés qui les utilisent. Ainsi, il est primordial de rechercher de nouvelles directions dans ce domaine afin de trouver des paramètres pertinents qui puissent être traités dans le cadre des RBD avec plus de souplesse. Nous avons ainsi élaboré des techniques de paramétrisations basées sur des transformées en ondelettes. Nous avons proposé une nouvelle technique de paramétrisation qui fourni des paramètres robustes, et localisés dans le domaine fréquentielle. La technique proposée est basée sur la projection des énergies d'un banc de filtre Mel sur une base d'ondelette. En fait, cette technique est une reformulation de l'approche "filtrage fréquentiel", dont le but est d'accentuer les fréquences des formants en utilisant un filtre linéaire. Cette approche nous permet d'obtenir des paramètres robustes et localisés dans le domaine fréquentiel. Nous avons remplacé le filtre linéaire par une base d'ondelette. Ainsi nous avons obtenu un meilleur contrôle sur la localisation dans le domaine fréquentiel. Notre perspective dans ce domaine est de capturer les dynamiques fréquentielles de la parole. Première étape dans ce but, le formalisme des RBD, et la technique de paramétrisation proposée, constituent une plate-forme potentielle.

Dans le domaine de l'extraction des paramètres acoustiques, nous avons analysé le problème de la robustesse au bruit. Bien que des paramètres robustes soient indispensables pour traiter des conditions bruitées, les méthodes de compensation des modèles acoustiques sont plus avantageuses pour s'adapter à des conditions variées. Nous avons ainsi proposé une nouvelle approche de compensation prédictive. Cette approche est basée sur l'apprentissage supervisé des effets du bruit sur les modèles de la parole. L'avantage de notre approche est qu'elle ne fait aucune hypothèse sur la façon dont le bruit affecte le signal acoustique. Notre approche est basée sur l'exploitation statistique des modèles de parole appris dans diverses conditions bruitées. Nous avons effectué une étude préliminaire basée sur un modèle de régression linéaire. Même avec de tels modèles simplistes, nous avons obtenu des propriétés de généralisation considérable. Nos perspectives concernent l'utilisation de modèles plus compliqué (i.e. déterministe ou probabiliste) pour mieux capturer les variations des modèles de parole bruitée.

minitoc

Deuxième partie

Revisiting speech recognition systems : dynamic Bayesian networks and new computational paradigms

Chapitre 1

Introduction

Sommaire

1.1	**Statistical speech recognition**	**37**
1.2	**Acoustic Modeling**	**39**
1.3	**Language Modeling**	**40**
1.4	**Speech feature extraction**	**41**
1.5	**Noise compensation**	**41**
1.6	**Organization**	**42**

1.1 Statistical speech recognition

Speech communication is one of the standard routines in our daily lives. However simple it might seem to us, it is a complicated process that involves auditory, linguistic and perceptual processing. The development of an automatic speech recognition system requires accurate representation of the actual physical processes that take place in this communication channel.

Simply put, the goal of an automatic speech recognition system is to deduce meaningful linguistic units (i.e., words) from acoustic waveforms. Due to the random nature of the process and interferences, it is not possible to derive a deterministic formulation that provides a mapping between acoustic signal and conceptual meanings. Instead the problem is generally formulated in a probabilistic framework. In probabilistic setting the speech recognition is stated as the estimation of 'most probable' linguistic representation of a 'given' acoustic waveform. The mathematical formulation of this problem is :

$$\hat{W} = \arg\max_{W} P(W|O)$$

where O is a set of observations from the acoustic waveform and W is a random variable that takes its values from the possible linguistic representations in the language under consideration. $P(W|O)$ is the conditional probability distribution of the linguistic representation given the observations. This conditional distribution constitutes the knowledge base of the recognizer. This knowledge base is constructed using statistical learning techniques and a priori expertise on speech production mechanisms. The role of a priori expertise on the domain is to provide a set of simplification assumptions that will guide the statistical machinery to extract relevant information for recognition.

Because of the complexity of the speech production mechanisms there is no simple parametric representation of $P(W|O)$ that involves both acoustic and linguistic information. The basic

approach is to first divide the problem into acoustic and linguistic components that can be handled separately. This is achieved using a Bayesian reformulation :

$$\hat{W} = \arg\max_{W} P(O|W)P(W).$$

In this formulation, the acoustic model, $P(O|W)$, encodes the statistical distribution of speech acoustics given the linguistic labeling. $P(W)$ is the probability assigned by the language model which encodes the a priori linguistic information. This approach is called the source-channel model. $P(W)$ constitute the language source with all the linguistic constraints of the underlying language and $P(O|W)$ is the acoustic channel that outputs the speech signal based on the linguistic unit W. With this formulation the two components of the system can be constructed separately later to be combined in the recognition phase. The information source for the linguistic component is written text documents that is supposed to be sufficient to represent the properties of language. For the acoustic component it is the speech utterances labeled with corresponding linguistic units, i.e. words, phonemes. For both of these components the statistical approach consists of extraction of relevant information content from data and formulation of a parametric representation that is capable to encode this content.

From a machine learning point of view one can identify three main modules of processing in a statistical speech recognition system :

- Information extraction, acoustic and/or linguistic.
  For acoustic modeling the information extraction stage involves the use of physical properties of speech production mechanisms. The acoustic speech waveforms are projected on to a feature space where speech content can be identified independent of interferences due to speaker and environment variations. In language modeling, the information content that should be extracted is consistent word sequences that imply a semantic representation. Besides simple word sequences their semantic, syntactic and morphologic properties are also used to achieve a finer representation of language.
- Probabilistic modeling of information content.
  Once useful information is extracted from available data, the next stage is to define a coherent representation of this data by probabilistic means. This consists of the specification of sufficient statistics and their representation in simple parametric formulations. The parametric representation and the sufficient statistics should be chosen to match the available data and also to allow affordable computational complexity. The widely accepted approach in speech recognition is to use HMMs for acoustic modeling and n-grams for language modeling.
- Statistical adaptation of models to application conditions.
  Generally speaking, statistical learning in probabilistic systems leads to a fit of the hypothesized model to available training data. One of the common problems encountered in this setting is over fitting. The over fitting problem may arise due unbalanced datasets or insufficient model complexity and causes weak generalization properties of the model. This is one of the common problems in automatic speech recognition that arises when models are trained in specific training conditions. These models generally fail to achieve good performances under mismatched application conditions. In such circumstances adaptation techniques are employed to compensate the effects of mismatch between training and application conditions.

In this thesis we address these three major components of a speech recognition system with novel strategies. We address the probabilistic modeling problem with a powerful statistical machinery, namely probabilistic graphical models (PGM) [104]. This formalism indeed encapsulates many probabilistic techniques used in the speech community [1]. Precisely, we propose novel modeling approaches for acoustic and linguistic components using dynamic Bayesian networks (DBN) which are directed probabilistic graphical models. Dynamic Bayesian networks include hidden Markov models (HMM) and n-gram models as a special case that are the most widely used probabilistic models for acoustic and linguistic information. Therefore rethinking the modeling problems in this formalism provides new perspectives that was not considered previously. Besides novel modeling approaches we also address new acoustic feature extraction schemes. Our main motivation in this direction is to seek for robust features that are not bound to be used in classical HMM setting. Finally we address the adaptation problem with a novel *supervised-predictive compensation* technique. We propose a supervised learning strategy to learn the variation of speech models in different noise conditions. In the sequel we briefly summarize the contributions in these domains.

1.2 Acoustic Modeling

State-of-the-art automatic speech recognition (ASR) systems are based on probabilistic modeling of the speech signal using Hidden Markov Models (HMM). In this thesis, we reformulate the acoustic modeling problem in speech recognition within the probabilistic graphical models (PGM) formalism. PGM is a unifying framework for statistical learning which provides an abstraction of quantitative and qualitative components of a statistical model. Dynamic Bayesian networks (DBN) are a subset of PGM which are defined on directed acyclic graph structures and which include HMM as a special case. These models are encoded with a graph structure that defines the probabilistic relations between its variables and a set of associated conditional probabilities. One of the main advantages of this representation is the graphical abstraction that provides a visual understanding of the modeled process. Moreover as a combination of probability theory and graph theory, this formalism covers several advantages from both domains.

One of the principle weakness of HMMs is the independence assumptions on correlations between the observed and hidden processes of speech. In HMMs, probabilistic relations between the hidden and the observed processes presume that the hidden process is first-order Markov, and that each observation depends only on the active state of the hidden process. These assumptions rely on the classical feature extraction schemes of the acoustic waveform. In feature extraction stage, the speech signal is considered as a quasi-stationary process consisting of consecutive frames that can be treated independently. However this independent processing does not yield independent feature sequences. Although HMMs encode frame dependencies through the first order Markov relation on the hidden process, they are not inherently appropriate to model complex dependency relations between hidden and observed processes [127, 5]. A common practical solution for frame dependency problem is the use of time derivative of feature sequences.

In this thesis we propose to use the DBN setting to extract the proper dependence structure for speech modeling rather than limiting ourselves with HMMs. The proposed approach is based on structure learning paradigm in DBN framework [51]. Briefly structure learning in Bayesian networks is defined as the extraction of causal relations between model variables with minimal complexity. Our strategy is to define a class of DBN model structures that can capture various dependence properties between hidden and observed processes of speech [36, 34]. HMM is also included in this class as a specific model. This class is later used as the search space for the learning algorithm. By defining the learning strategy in DBN framework we are able to extract individual dependence properties for each acoustic unit. That is, when HMM is found to be insufficient to model a particular acoustic unit, the algorithm seeks for more appropriate dependence structures

and fits to the one that is the best encoding for the unit. This approach has the advantage to guaranty that the resulting model represents speech with higher fidelity than HMM.

In the first place we perform preliminary experiments on isolated digit recognition task [36, 37, 35]. Next, we extend this approach for continuous speech recognition. In this task, one of the main problems was the decoding algorithm. Although Viterbi algorithm provides an efficient solution in HMM framework, in the proposed DBN setting each acoustic unit may have different conditional independence properties. As an initial approach, we develop a technique to construct a meta model that represents all acoustic units with the same graph structure such that standard Bayesian network inference algorithms can be applied directly [39]. Although all model structures are simplified to the same structure we use a special encoding technique to respect the learned conditional independence properties of each unit. This technique is known as asymmetric encoding and is the basis of Bayesian multinets in BN framework [66]. However this approach is computationally demanding. Therefore in [41, 38], we develop a fast decoding algorithm based on dynamic programming that exploits the structural properties of the acoustic models proposed. The experimental results on continuous digit and phoneme recognition shows relative improvements with respect to HMM. Recently, we proposed a new noise robust modeling technique in this framework in [79].

1.3 Language Modeling

Statistical language models attempt to construct a representation of language using the empirical probability distribution of linguistic events. Briefly the specification of a language model involves the definition of implicit and/or explicit variables of language. The dynamic evolution of language is controlled by these variables and their interactions through time. Each variable interacts with a certain number of factors that constitute its context. In probabilistic terms the context of a linguistic unit is defined with conditional independence properties. Given its context each linguistic unit is assumed to be independent of other linguistic events.

On the other hand, conditional independence is the core property of probabilistic graphical models, it is indeed the exploitation of this property that leads to efficient and generic inference algorithms of graphical models. Moreover, n-gram and n-class models (and other language modes) are very particular instances of *dynamic Bayesian networks* (DBNs) which are *oriented* probabilistic graphical models. Thus, it is a natural idea to rethink language models within the general framework of DBNs and seek potential benefits from this rethinking.

The widely used n-gram models encode the word sequences as a Markov chain. This approach yields very good results as long as the training data is sufficient to train the model for the specified history range n. Generally back-off schemes are employed to replace a high order model with simpler ones, i.e. smaller values of n, in case of data insufficiency. n-class models are defined on higher level information. Words are clustered according to semantic, syntactic and/or morphologic criteria. This abstraction defines a smaller state space, therefore data insufficiency is less significant for these models. The combination of higher level semantic/syntactic information and word level information is generally performed using linear interpolation techniques or maximum entropy framework. In this thesis we use the DBN framework to combine these two levels of information with appropriate dependence relations. We develop a unifying approach that processes each linguistic unit (i.e., word, class) in a unique model and construct new data-driven language models with improved performances. The principle of this approach is to construct DBNs in which a variable (word, class or any other linguistic unit) may depend on a set of context variables. The preliminary results obtained using several datasets show the potential improvements of this approach [42, 44, 43]. The proposed models provide a model complexity that is in between n-gram and n-class models. Therefore they possess a good balance for generalization and context modeling properties.

1.4 Speech feature extraction

Another direction of my thesis is on new acoustic feature extraction algorithms for ASR. Existing feature extraction algorithms are developed considering the properties of HMM formalism. Therefore there is a need for new directions in this field to be able to extract informative features that can later be processed within the general DBN setting without the constraints of HMM approach. With this motivation in mind we seek for potential novel approaches. Precisely, we review wavelet analysis based feature extraction schemes and propose a new technique for frequency localized robust feature extraction using wavelets [40].

State of the art speech feature extraction schemes (Mel frequecy cepstral coefficients (MFCC) and perceptual linear prediction (PLP)) are based on auditory processing on the spectrum of speech signal and cepstral representation of the resulting features [29]. The spectral and cepstral analysis is generally performed using Fourier transform. The advantage of Fourier transform is that it possesses very good frequency localization properties. Consequently due to time-frequency uncertainty principle the time resolution of Fourier transform is very poor. On the other hand wavelet transform is defined as a projection that provides both time and frequency localization. Moreover the projection basis defined for wavelet transformations have adjustable smoothness properties [93]. Therefore it is potentially more appropriate for the analysis of speech irregularities rather than the infinitely differentiable exponential basis of the Fourier transform.

In this thesis we surveyed feature extraction techniques that exploits these nice properties of wavelet transform. In the literature, the use of wavelet transform takes place both in spectral and cepstral analysis. Wavelet packets are used to approximate the mel-scale frequency resolution [46]. Dyadic wavelet transform is used for cepstrum computation [61]. We provide an empirical analysis of several techniques with different wavelet functions and feature computation schemes.

An important desirable property in speech signal representation is frequency localization. Frequency local features facilitates compenstation of band limited additive noise components and provides an auditory processing medium. Cepstral domain features however does not have this property. Frequency filtering is a successful approach that attains this property with noise robust features localized in frequency. The principle of this technique is to use a simple first order decorrelating filter to compute speech features from the mel-spectrum [97, 99] . Experimental evidence show that these features are as successful as cepstral features in clean and noisy conditions [97, 103, 99, 95]. We reformulate this technique in wavelet domain. Indeed the simple filters proposed in this setting correspond to Haar wavelet transformation. Using a wavelet basis with better smoothness properties improves decorrelation property and allows a controlled localization in frequency domain. We evaluate these techniques on Aurora-3 noisy digit recognition task under real noise conditions.

1.5 Noise compensation

In ASR systems one of the commonly encountered problems is the mismatch between training and application conditions. Solutions to this problem are provided as pre-processing of the speech signal for enhancement, noise resistant feature extraction schemes and statistical adaptation of models to accommodate application conditions. In order to achieve improved performances all

these techniques should be applied in a practical system. In compensation schemes the matching between training and application conditions is achieved either using a hypothesized mismatch function that describes the noise corruption or applying statistical adaptation of the model parameters using data recorded in new environment. In adaptive compensation such as MAP [55] and MLLR [85], the observed corrupted speech data is used to transform the initial models. One inconvenience of these schemes is that they require at least few dozens of seconds of speech to yield relatively good estimates of the new models. A more important inconvenience is the need of speech data transcription in an unsupervised mode, this is done generally using the initial models which limits the performances if the transcription accuracy is poor. Predictive compensation schemes (PMC [54] for instance) do not rely on observed speech data, rather they use noise observations and models to estimate the speech models in the new environment. Namely, the new speech model is a combination of the initial one and a (parametric) noise model for which the parameters are estimated from noise observations. The combination is obtained using a function (mismatch function) that hypothesizes the corruption of speech by noise sources [54]. An inconvenience of predictive schemes is the assumption made about the effects of the new acoustic environment on the clean speech, i.e., how the noise sources alter the clean speech signal (additive or/and convolutional...). This assumption may be unrealistic which consequently may lead to erroneous estimations of corrupted speech distributions. An other inconvenience of predictive schemes is their strong dependence on the front-end (MFCC in general) and the probabilistic models (HMMs in general) which are used.

We propose a new predictive compensation scheme which makes no assumption on how the noise sources alter the speech data and which do not rely on clean speech models. Rather, this new scheme makes the (realistic) assumption that speech databases recorded under different background noise conditions are available. The philosophy of this scheme is to process these databases in order to build a "tool" which will allow it to handle new noise conditions in a robust way. The advantages of this approach are :

- It makes no assumption on how the background noise source alters the speech signal, the approach is formulated in model parameter space.
- It is completely independent of the front-end and the probabilistic acoustic models used to process and model speech.
- It does not rely on the availability of an initial canonical speech model.
- It is computationally efficient.

The performance of this new compensation scheme is evaluated on a connected digits recognition task [24, 25]. The results show that this scheme can perform significantly better than multi-conditions training, which is the most widely used technique in these kind of scenarios.

1.6 Organization

The thesis is organized as follows. In chapter 2 we introduce the Bayesian networks (BN) formalism. We overview the inference and learning algorithms for static and dynamic networks. In chapter 3 we provide an overview of acoustic modeling approaches that can be considered as particular cases of BNs and propose a new modeling approach based on structure learning of DBNs. Chapter 4 focuses on language modeling. We provide a perspective on classical techniques in BN framework and develop a new set of class based language models using DBNs. Next in chapter 5 we investigate novel feature extraction schemes for frequency localized robust features based on wavelet analysis. Finally in chapter 6, we address noise robustness issue in ASR systems and propose a new supervised-predictive compensation scheme.

Chapitre 2

Bayesian Networks

Sommaire

2.1 Contributions . . . **43**
2.2 Bayesian Networks . . . **43**
2.2.1 Some notation . . . 44
2.2.2 Definition . . . 44
2.2.3 Structure Semantics . . . 44
2.2.4 Numerical Parameterization . . . 46
2.3 Inference . . . **48**
2.3.1 Inference in probabilistic models . . . 48
2.3.2 Exploiting conditional independence for efficient inference . . . 49
2.3.3 Overview of exact inference in BNs . . . 49
2.4 Learning . . . **50**
2.4.1 Parameter learning . . . 50
2.4.2 Parameter learning with incomplete data . . . 51
2.4.3 Structure learning . . . 52
2.5 Dynamic Bayesian Networks . . . **55**
2.5.1 Representation . . . 57
2.5.2 Inference in DBNs . . . 60
2.5.3 Learning DBNs . . . 63

2.1 Contributions

– A survey of learning and inference algorithms for static and dynamic Bayesian networks.
– A dynamic Bayesian network representation for non-Markov processes.

2.2 Bayesian Networks

Bayesian networks (BNs) (also known as belief networks, generative models, causal models, knowledge maps, etc.) are probabilistic graphical models that encode probabilistic relationships on directed acyclic graphs (DAG). The probabilistic relationships are defined as a set of Conditional Independence (CI) assertions. The graph structure encodes the conditional independence

relations. The numerical parameterization specifies the functional form of the representation based on these relations. The statistical methodologies devised in this formalism provides efficient computational techniques for both aspects of the representation, i.e. graph structure and numerical parametrization. The extension of this formalism to dynamic processes is termed as Dynamic Bayesian Networks (DBN). DBNs encode temporal relations among the system variables.

2.2.1 Some notation

In the following, X_i denotes a continuous or discrete random variable. Values of the random variable will be indicated by lower case letters as in x_i. For a discrete variable that takes r values, x_i^k denote a specific assignment for $1 \leq k \leq r$. A set of variables is denoted in boldface letters $\mathbf{X} = \{X_1, \ldots, X_n\}$.

2.2.2 Definition

Bayesian networks exploit the conditional independence properties between its variables to achieve a compact representation and efficient inference. Conditional independence is defined as follows.

Definition Let P be a probability distribution over three disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. $\mathbf{X}$ is said to be *conditionally independent* of $\mathbf{Y}$ given $\mathbf{Z}$ ($\mathbf{X} \perp \mathbf{Y} | \mathbf{Z}$) if

$$P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z}). \tag{2.1}$$

Conditional independence allows to handle subsets of variables independently conditioned of an initialization of another set. This is achieved via the factorization of the JPD.

A BN is constructed over a set of nodes, where each node is associated with a random variable X_i. The conditional independences among the variables are represented with a *directed acyclic graph (DAG)*. The semantics of the DAG structure of a BN implies that a variable is conditionally independent of its non-descendants given its parents. This implies a direct factorization of the joint probability distribution in terms of conditional probabilities of each variable given its parents. This factorization enables the use of simple local conditional distributions rather than using the full JPD for inferring probabilities. More formally, a BN is defined as :

Definition *Bayesian Network :* A Bayesian Network is a pair (S, Θ), where S is a DAG, Θ is a parameterization of a set $\{P(X_1|\Pi_1), \ldots, P(X_n|\Pi_n)\}$ of conditional probability distributions, one for each variable, and Π_i is the set of parents of node X_i in S. The set of CPDs defines the associated jpd as,

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\Pi_i). \tag{2.2}$$

2.2.3 Structure Semantics

The directed acyclic graph structure of BNs associate each node of the graph with a random variable. The arcs of the network encode the conditional independence assertions of the JPD. An arc between two nodes indicate a causal relation. The lack of an arc between two nodes imply a conditional independence. The set of CI assertions that are implied by the DAG of a BN are characterized with its *d-separation* properties [104].

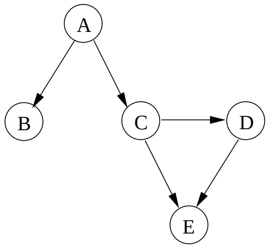


FIG. 2.1 – A Bayesian network representation for the joint probability distribution $P(X_1, X_2, X_3, X_4, X_5)$.

Definition The *underlying graph* of a Bayesian network is an undirected graph obtained from the network by replacing every link with an undirected edge.

Definition A *trail* in a Bayesian network is a sequence of links that form a cycle-free path on the underlying graph.

Definition A node X_i is called a *head-to-head* node with respect to a trail t if there are two consecutive links $X_j \rightarrow X_i$ and $X_i \leftarrow X_k$ on t.

Definition A trail t is *active* with respect to a set of nodes $\mathbf{Z}$ if
1. every head-to-head node w.r.t. t is either in $\mathbf{Z}$ or has a descendant in $\mathbf{Z}$, and
2. every other node along t is outside $\mathbf{Z}$.

Otherwise, the trail is said to be *d-separated* by $\mathbf{Z}$.

Theorem 2.2.1 *Let $B = (S, \Theta)$ be a Bayesian network of a probability distribution P over disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. If all trails between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ are d-separated by $\mathbf{Z}$, then $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$.*

This theorem implies that a variable is conditionally independent of its non-descendants given its parents because the parents of each node *d-separates* all trails between a node and its non-descendants [104]. Hence the JPD can be factored over conditional probabilities of each variable given its parents on the DAG. This is illustrated in the following example.

Example Suppose the CI assertions of the JPD $P(\mathbf{X})$ over a set of variables $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5\}$ is represented on the DAG structure shown in figure 2.1. Using the chain rule of probability and an ordering $\{X_1, X_2, X_3, X_4, X_5\}$ of variables, the JPD $P(\mathbf{X})$ can be factored as :

$$
\begin{aligned}
P(\mathbf{X}) &= \prod_{i=1}^{5} P(X_i | X_{i-1}, \ldots, X_1) && (2.3)\\
&= P(X_5|X_4, X_3, X_2, X_1)P(X_4|X_3, X_2, X_1)P(X_3|X_2, X_1)P(X_2|X_1)P(X_1) && (2.4)
\end{aligned}
$$

The CI assertion for each variable given its parents on the DAG in figure 2.1 can be read as :

$$X_5 \perp \{X_2, X_1\} | \{X_3, X_4\} \quad X_4 \perp \{X_1, X_2\} | X_3 \quad X_3 \perp X_2 | X_1 \quad X_2 \perp \{X_3, X_4, X_5\} | X_1.$$

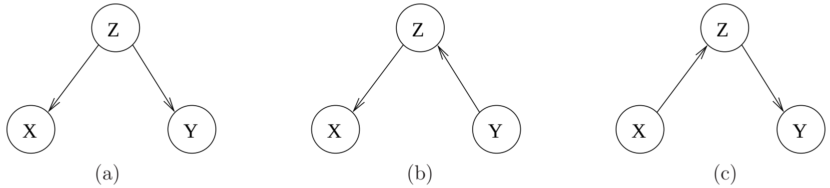


FIG. 2.2 – Directed acyclic graph structures for $P(X, Y, Z)$ where $X \perp Y | Z$ w.r.t node orderings (a) X, Y, Z, (b) X, Z, Y and (c) Y, Z, X.

Imposing these assertions on Eq.2.4 one obtains the factorization of the JPD in terms of conditional probabilities of each variable given its parents :

$$
\begin{aligned}
P(\mathbf{X}) &= P(X_5|X_4, X_3)P(X_4|X_3)P(X_3|X_1)P(X_2|X_1)P(X_1) && (2.5)\\
&= \prod_{i=1}^{5} P(X_i|\Pi_i) && (2.6)
\end{aligned}
$$

where Π_i denotes the parents of X_i in the DAG structure.

Each DAG implies a unique set of CI assertions. However the converse is not true, i.e. there exists more than one DAG that encodes the same set of CI assertions. This is illustrated in the following example.

Example Let P be a probability distribution over three variables X, Y and Z where X is conditionally independent of Y given Z, $X \perp Y | Z$. Using the CI assertion and the chain rule of probability the JPD $P(X, Y, Z)$ can be factored in three different ways as :

$$
\begin{aligned}
P(X, Y, Z) &= P(X, Y|Z)P(Z) = P(X|Z)P(Y|Z)P(Z) && (2.7)\\
&= P(X, Z|Y)P(Y) = P(X|Z)P(Z|Y)P(Y) && (2.8)\\
&= P(Y, Z|X)P(X) = P(X)P(Y|Z)P(Z|X) && (2.9)
\end{aligned}
$$

Each factorization is derived using a different ordering of the variables. The corresponding DAG structures are shown in figure 2.2

2.2.4 Numerical Parameterization

The numerical parameterization of the conditional probability distributions (CPD) constitute the quantitative component of a BN. The parameterization depends on the type of the variable (i.e. discrete or continuous) and the type of its parents in the DAG. The parametric form of the CPD plays a crucial role on the computational efficiency of inference and learning procedures of a BN. The exponential family of distributions allows one to perform such computations efficiently and in closed form. In this section we give a set of possible parameterizations in this family. For discrete variables we use multinomial distribution and for continuous variables we use Gaussian density. In the following we give respective parameterizations of CPDs, for each possible setting of a variable and its parents :

– discrete variable with discrete parents

– continuous variable with continuous parents
– continuous variable with discrete parents

We exclude the case where a discrete variable has a continuous parent. For this case there is no parametric form that allows exact inference.

Discrete variable with discrete parents

For a discrete variable X_i with discrete parents Π_i, the CPD is represented with a conditional probability table, Θ_i. For each distinct configuration π_i^j of Π_i, a probability θ_{ijk} is associated to each instance x_i^k of X_i. Hence the parameters of the CPD are defined as follows :

$$\theta_{ijk} = P(X_i = x_i^k | \Pi_i = \pi_i^j) \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, q_i \\ k = 1, \ldots, r_i. \end{array} \tag{2.10}$$

r_i is the number of possible values that X_i can attain and q_i is the number of distinct configurations of Π_i.

Continuous variable with continuous parents

For a continuous variable X_i, with continuous parents Π_i, the CPD is a conditional Gaussian where the mean is a linear function of the values of the parents.

$$P(X_i|\Pi_i) = \mathcal{N}(\beta_i^T \Pi_i, \sigma_i^2) \tag{2.11}$$

β_i is a measure of dependency of X_i on its parents. The size of the vector β_i is equal to the number of parents of X_i. Hence the parameters of the CPD are :

$$\beta_i, \quad \sigma_i^2, \qquad i = 1, \ldots, n. \tag{2.12}$$

Without loss of generality, we assume that the unconditional probability distribution has a zero mean. Extension to the more general case is trivial [3].

Continuous variable with discrete and continuous parents

For a continuous variable X_i, with discrete and continuous parents Π_i, the CPD is a table of conditional Gaussians. For each distinct configuration of the discrete parents, there is an associated conditional Gaussian where the mean of the Gaussian is a linear function of values of continuous parents.

$$P(X_i|\Pi_i) = \mathcal{N}(\beta_{ij}^T \Pi_i^{cont.}, \sigma_{ij}^2) \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, q_i. \end{array} \tag{2.13}$$

Hence the parameters of the CPD are :

$$\beta_{ij}, \quad \sigma_{ij}^2, \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, q_i. \end{array} \tag{2.14}$$

[3]Redefine $\beta_i = [1 \quad \beta_i]^T$ and $\Pi_i = [\mu_i \quad \Pi_i]^T$.

2.3 Inference

In this section we define the inference problem in Bayesian networks. We illustrate the use of conditional independence to achieve efficient inference algorithms. We overview some exact inference algorithms in BN framework. We give a detailed description of the most widely used one in the appendix, i.e. the junction tree (JLO) algorithm.

2.3.1 Inference in probabilistic models

A generic interpretation of probabilistic inference can be stated as the computation of posterior probabilities of a set of random variables given values of some other set. This is an essential step both for information extraction from a given probabilistic model and learning a probabilistic model from statistical data. For instance, in medical diagnosis, given some observed symptoms we would like to find out the underlying deceases. In a probabilistic approach, we require the conditional probability distributions of decease variables given observed symptoms to make decisions. This posterior distribution represents our belief on the possible causes of the symptoms. As new symptoms are observed the belief can be updated by successive computations of the posterior. Alternatively we may revise our belief to specify the most probable decease under the given set of symptoms. We can identify three principle inference procedures :

Let D denote a realization of the variables in the set $X = Y, Z$ where Y is the subset of observed variables and Z is the subset of missing variables.

Likelihood computation : Given a complete/incomplete set D of observations compute the likelihood $P(D)$ on the given BN.

Belief update : Given a partial set Y of observations, compute the *a posteriori* probability $P(Z|Y)$ of non-observed variables.

Most probable configuration : Given a partial set Y of observations estimate the most probable realization of missing variables : $\hat{Z} = \arg\max_Z P(Z|Y)$.

Let's illustrate the belief update process on a simple example using a naive approach. Suppose we have a joint distribution $P(A, B, C, D, E)$, over the set of discrete random variables variables $\{A, B, C, D, E\}$ Suppose also that we want to compute the posterior distribution $P(A|E)$. Using Bayes rule we have :

$$P(A|E) = \frac{P(A, E)}{\sum_A P(A, E)}. \tag{2.15}$$

Hence inference is reduced to a marginalization of the full joint distribution $P(A, B, C, D, E)$ to compute $P(A, E)$. The brute force approach is to compute the following sum directly.

$$P(A, E) = \sum_{B,C,D} P(A, B, C, D, E) \tag{2.16}$$

If each variable takes N possible values this computation requires $O(N^5)$ operations. For a small set of variables this naive approach is acceptable. The difficulty in inference arises when we are dealing with a lot of variables. The direct approach to compute and update the probabilities is not practical (and sometimes impossible) for a system with many variables. In that case, we need efficient techniques or approximations to avoid such computational bottlenecks.

2.3.2 Exploiting conditional independence for efficient inference

Now lets illustrate the notion of conditional independence to achieve efficiency in inference. Suppose that the joint distribution is represented by the Bayesian network given in Figure 2.1. The conditional independence assertions of this network allows the following factorization :

$$P(A,B,C,D,E) = P(A)P(B|A)P(C|A)P(D|C)P(E|C,D) \tag{2.17}$$

In this case, marginalization can be done efficiently by distributing sums over products. The idea is to move the sums toward right and the terms toward left so as to reduce the number of arguments of summation.

$$\begin{aligned} P(A,E) &= \sum_{B,C,D} P(A)P(B|A)P(C|A)P(D|C)P(E|C,D) && (2.18)\\ &= P(A)\sum_{B} P(B|A)\sum_{C} P(C|A)\sum_{D} P(D|C)P(E|C,D) && (2.19) \end{aligned}$$

For this specific example the maximum number of arguments in each marginalization is three. Hence the computational complexity of marginalization reduces to $O(N^3)$ operations. Distributing global computations to smaller sets of variables is the key idea to develop efficient inference algorithms.

2.3.3 Overview of exact inference in BNs

In Bayesian networks efficient inference is achieved by exploiting conditional independence. The direct approach to compute and update the probabilities using marginalization and multiplication operations is not practical for a system with many variables. However, these operations within small sets of variables are quite easy. Using the factorization property of Bayesian networks, the inference problem can be divided into operations within smaller sets of variables (i.e. cliques). There exists several algorithms that exploit the CI assertions of BNs for efficient inference. In all of them the aim is to find the best sequence of operations to infer the desired posterior with minimum computational cost. However, this problem is proved to be NP-hard [20]. In the sequel we briefly overview some exact inference algorithms. In Appendix ? we give the details of the clique tree propagation algorithm which is the most widely accepted exact inference algorithm. For a more general discussion the reader is referred to [62],[96].

Polytree message propagation : This algorithm is originally developed for BNs with a polytree structure [104]. It is a dynamic programming approach based on a message passing scheme between the nodes of the tree. Each message from children to parents propagates the evidence in the leaf nodes up to the roots. And redistribution of the information on the roots yields local marginal densities at each node.

Loop cutset conditioning : Loop cutset conditioning algorithm is an extension of polytree message propagation to multiply connected networks [104]. The basic principle is to instantiate a set of nodes (called cutset) such that the network becomes singly connected. Then the polytree message passing algorithm computes the inference results for each instantiation and the final result is obtained using a weighted average.

Clique-tree propagation : This is the most popular exact inference algorithm also known as JLO or clustering algorithm [74]. The network graph structure is first transformed into a special tree structure called the clique tree (or junction tree). The nodes of this tree are the cliques of the original network and they are obtained using a special clustering algorithm that gathers the subset of variables that enables a factorization of the joint probability density. The algorithm performs a message propagation on this clique tree. The algorithm is also applicable for BNs with continuous Gaussian variables [22].

Arc reversal/node reduction algorithm : The principle of this algorithm is to transform the network using a sequence of operators that reverses the links of the network using Bayes' rule [111, 112]. In the resulting network only the query nodes are left out with the evidence nodes as their parents.

Variable Elimination (VE) algorithm : Given an elimination ordering of variables VE algorithm eliminates the variables one by one by marginalization to compute the desired posterior density [129]. Some variations of VE algorithm are also known as symbolic probabilistic inference (SPI) [86], bucket elimination [32].

Differential approach : Differential approach compiles a BN into a multivariate polynomial [28]. The queries are obtained using the partial derivative of this polynomial with respect to each variable.

2.4 Learning

2.4.1 Parameter learning

In this section we handle the problem of learning parameters of a BN given a fixed structure and a set of observations. In the first section we describe the maximum likelihood (ML) estimation algorithm. Next we assume that there is some prior knowledge of the parameters and show how to incorporate this information in maximum a posteriori (MAP) estimation. Finally we show how to handle incomplete data sets in the case of ML estimation using expectation maximization (EM) algorithm and derive the update equations for different parameterizations.

Maximum Likelihood (ML) Estimation

In ML estimation, the parameters of the BN are chosen such that the likelihood of the observed data is maximum.

$$\hat{\Theta}_{ML} = \arg\max_{\Theta} p(D|S,\Theta) \tag{2.20}$$

For the exponential family of distributions the likelihood function is differentiable and an analytic solution for the maximization problem is possible. In Appendix B we give the solution for discrete, continuous and hybrid BNs.

ML estimation is a powerful method especially for large data sets. For small data sets, there are two basic problems about ML estimation, sparse data, and over-fitting [11], [7].

- *Sparse data :* Consider a discrete BN, and a dataset D. If there are no instances of a specific realization of a variable in D, then the sample likelihood does not exist. Hence, ML estimate is undefined. In order to define the ML estimate for n binary variables and a fully connected BN, one needs more than 2^{n-1} instances in the data set [11].

– *Over-fitting :* For small data sets, the maximum likelihood value may be much greater than the actual value. This over-estimation decreases as the data set size increases and the likelihood converges to the actual value. However, even there is over-estimation, the ML solution attempts to fit the data as well as possible. This results in a set of parameters that fit to an insufficient data. This is called over-fitting. (see [11] for an illustrated explanation)

Maximum a posteriori (MAP) estimation and Bayesian learning

The MAP estimate of a random variable Θ is defined as the value that maximizes the a posteriori pdf given observed data D. Using Bayes rule, one can show that maximizing the a posteriori pdf is equivalent to maximizing the joint pdf of D and Θ, which can be factorized as the product of the data likelihood and the a priori parameter pdf.

$$\hat{\Theta}_{MAP} = \arg\max_{\Theta} P(\Theta|D,S) \quad (2.21)$$

$$= \arg\max_{\Theta} P(D|\Theta,S)P(\Theta|S) \quad (2.22)$$

MAP estimation can be viewed as an approximation to Bayesian learning. In Bayesian approach, the data likelihood is computed by averaging over all possible parameter values (i.e. one takes the expectation using the prior parameter pdf).

$$P(D|S) = \int P(D|S,\Theta) \cdot P(\Theta|S) d\Theta \quad (2.23)$$

Hence, Bayesian parameter learning is the update of prior parameter pdf using the observed data.

$$P(\Theta|S,D) = \frac{P(\Theta|S)P(D|S,\Theta)}{P(D|S)} \quad (2.24)$$

MAP estimation approximates this integral with the maximum value of the integrand.

In order to use MAP estimation or Bayesian learning for the parameters of a BN, one needs to define the prior pdf for the parameters Θ. An important point in the selection of the prior is to have a conjugate family so that the functional form remains the same in the presence of data (i.e. the prior and posterior pdfs have the same functional form). Construction of parameter priors for DAG modes is discussed in [56].

The initialization of the priors is another issue, which requires prior knowledge over the domain. [66] defines a method to initialize parameter priors for BNs of discrete variables with multinomial distributions based on a set of assumptions. A similar method is extended to continuous case in [65].

2.4.2 Parameter learning with incomplete data

In the case of incomplete data, the factorization property of the BN cannot be used to compute the data likelihood. The likelihood of an incomplete sample D_l, where $\mathbf{X}_h$ is the set of unobserved (hidden) variables, turns out to be :

$$P(D_l|S,\Theta) = E\{P(D_l,\mathbf{X}_h|S,\Theta)\}. \quad (2.25)$$

The expectation is taken with respect to $P(\mathbf{X_h}|S,\Theta)$.

In this case there is no analytic solution for the maximization problem and iterative numerical algorithms are required. One solution is the expectation maximization algorithm [83]. The EM

algorithm is based on forming the conditional expectation of the log-likelihood function for complete data, given the observed data [33].

$$Q(\Theta^{'}|\Theta) = E_{\Theta}\{\log P(D_l,\mathbf{X}_h|S,\Theta^{'})\} \quad (2.26)$$

The algorithm starts with an initialization of the parameters Θ and alternates between E-step and M-step. At the E-step, Θ is fixed and Q is evaluated as a function of $\Theta^{'}$. At the M-step, Q is maximized in $\Theta^{'}$.

$$\hat{\Theta} = \arg\max_{\Theta^{'}} Q(\Theta^{'}|\Theta) \quad (2.27)$$

The details of the derivations and the update equations are given in Appendix C

2.4.3 Structure learning

The structure of a BN, defines the conditional independences among the variables. Learning structure means learning conditional independences from observations. The parameters of a BN with a given structure are estimated by ML or MAP estimation. One can use a similar method for choosing a structure for the BN that fits to the observed data. However, structural learning is slightly different than parameter learning. If we consider ML estimation or MAP estimation with uninformative priors then the structure that maximizes the likelihood will be the result. In this case complex structures (i.e. that has more dependencies between variables) will be more capable to increase the likelihood because they have more degrees of freedom. However, the aim is to find a structure that encodes the independences among the variables, that is we want our structure to be as simple as possible.

The general approach to structure learning is to introduce a scoring metric that evaluates each structure for the given data and then to search for the best structure according to this metric.

Scoring metrics

The criteria for selecting a structure has two terms. One for maximizing the likelihood and one for minimizing the complexity.

$$Score(S,D) = \log P(D|\hat{\Theta},S) - Pen(S) \quad (2.28)$$

An important property for a scoring metric is decomposability. If a metric is decomposable, it can be factorized as a multiplication of local metrics. The local metric is a function of a variable and its parents. Hence when comparing two structures, it is sufficient to compare the local metrics that differ.

Bayesian Information Criteria (BIC), Minimum Description Length (MDL) Score MDL says that, the best model of a collection of data items is the model that minimizes the sum of

– the length of the encoding of the model
– the length of the encoding of the data given the model

[82] uses the MDL principle to score a BN. The first term is related to the complexity of the model. The simpler the model the less the encoding length will be. Therefore the MDL scoring penalizes complex network structures. The second term is related to the likeliness of the data based on the model. In coding theory the codewords with higher probability are represented

with shorter codes. Therefore the length of the encoding is a measure of likelihood of data. The shorter the encoding the more likely the data. Hence learning a BN based on the MDL score is equivalent to

- minimizing the penalty score for the complexity of the structure
- maximizing the likelihood of data given the structure, based on ML parameter estimates

BIC score results from the approximation of the likelihood of the data given the structure around the MAP or ML estimates of the parameters. The approximation results in the same formulation of MDL [64].

$$Score_{MDL,BIC} = \log P(D|\hat{\Theta}, S) - Dim(S) \tag{2.29}$$

- For a discrete BN

$$Dim(S) = \begin{cases} \sum_{i=1}^{n} [k_i \log_2 n + d(r_i - 1)q_i] & \text{for MDL} \\ \frac{\log L}{2} \sum_{i=1}^{n} (r_i - 1)q_i & \text{for BIC} \end{cases} \tag{2.30}$$

Bayesian Dirichlet (BD, BDe) Score In Bayesian approach the data likelihood is computed by taking the expectation over all possible network structures.

$$P(D_l) = \sum_{S} P(D_l|\mathbf{S})P(\mathbf{S}) \tag{2.31}$$

Bayesian structural learning is updating the belief on each structure by computing the posterior $P(S|D)$. Hence, Bayesian scoring metric is the posterior structure pdf, or equivalently $P(S, D)$.

For discrete BNs, [21] derived a Bayesian scoring metric based on the following assumptions.

Assumption 1 Multinomial sample.
The network structure hypothesis is true if and only if the database D can be partitioned into a set of multinomial samples [4].

Assumption 2 Parameter Independence.
Given a network structure S, if $P(S) > 0$, then
- $P(\Theta_S|S) = \prod_{i=1}^{n} P(\Theta_i|S)$
- $P(\Theta_i|S) = \prod_{j=1}^{q_i} P(\Theta_{ij}|S)$ *for* $i = 1, \ldots, n$

With this assumption the likelihood $p(D|S)$ can be factorized. The assumption of parameter independence corresponds to the assumption that one's knowledge is equivalent to having seen only *complete cases*.

Assumption 3 Parameter modularity.
The parameters associated with a variable, only depends on its parents.

This assumption provides the use of prior parameter distributions defined on a complete network for all other networks with the same parental configuration.

[4]Consider a one-variable domain $U = y$ where y can take values $y^1, \ldots, y^r$. A finite sequence of observations $D = y_1, \ldots, y_N$ is an r-dimensional multinomial sample if
- D is exchangeable (interchanging any two observations in the sequence does not change the probability of D)
- $\theta_k > 0$ for $k = 1, \ldots, r$ and $\sum_{k=1}^{r} \theta_k = 1$ and $p(y_l = y^k | y_1, \ldots, y_{l-1}, \theta) = \theta_k$

Assumption 4 Dirichlet.
The parameters have a Dirichlet distribution. $P(\Theta_{ij}|S) = Dir(\alpha_{ijk})$

This assumption is eliminated when only positive distributions are considered. It is proved that Dirichlet distribution is the only distribution that is consistent with previous assumptions [66].

Assumption 5 Complete Data.
All variables are instantiated. In this case the likelihood can be computed in closed form.

$$Score_{BD} = P(S) \cdot P(D|S) = \prod_{i=1}^{n} \prod_{j=1}^{qi} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2.32}$$

In [66], the BDe metric is derived with the following additional assumptions.

Assumption 6 Hypotheses Equivalence.
S denotes the same hypotheses for S^1 and S^2, if they are isomorphic (i.e. they represent the same assertions of conditional independence).

With this property, all complete network structures can be considered under the same hypotheses.

Assumption 7 Structure possibility.
All complete network structures have probability greater than zero.

$$P(S_c) > 0 \tag{2.33}$$

The additional assumptions imposes the following constraints on the Dirichlet exponents :

$$\alpha_{ijk} = \alpha_{sc} P(X_i = x_i^k, \Pi_i = \pi^j | S_c) \tag{2.34}$$

α_{sc} is the equivalent sample size that is a measure of the prior domain knowledge. It can be considered as the number of observations on the domain to achieve current knowledge. This constraint enables the initialization of the prior parameter pdf of an arbitrary network structure given the pdf of a complete structure.

The BDe metric also has the property of likelihood equivalence. That is, the data likelihoods for two structures S^1 and S^2 are the same if they are isomorphic. Hence, assuming uniform structure priors, the scores of two structures, that have the same conditional independence assertions, are the same.

The Bayesian metric for continuous BNs is handled in [65].

Structure Priors

In Bayesian scoring metrics the preference for the structure (i.e. to choose simpler structure) is represented in the prior structure distribution. The structure priors are defined so as to penalize the structures that deviate from the initial structure. In [66] the deviation is represented as the number of different arcs between two networks. For a single variable this is denoted as δ_i.

$$P(S) = c \cdot \prod_{i=1}^{n} \kappa^{\delta_i} \qquad 0 < \kappa < 1 \tag{2.35}$$

In [10], Buntine considers different penalty factors for different arcs.

Searching structure space

Searching the structure space for high scored structures is the next issue in structure learning. It has been shown in [19] that finding the structure with maximum scoring is NP-hard. Therefore for arbitrary structures, heuristic search algorithms are used.

- *K2 search :* Initialize with an ordering of nodes such that the parents of a node are listed above the node. Starting with an empty structure and tracing the node list in order, one adds a parent to the node that increases the score. The number of parents to be added to a single variable may be limited to a predetermined constant for fast inference [21].
- *Local search :* Initialize with a network structure, possibly random. Evaluate the change in the score for all arc changes on this network and choose the one that has the maximum change. Continue this process until no more arc changes increases the score. This algorithm generally sticks into local maxima.
- *Iterated hill-climbing :* Apply local search until local maximum. Randomly perturb the structure and repeat the process for some manageable number of iterations.
- *Simulated annealing :* Initialize the system with some temperature T_0. Evaluate

$$p = \exp\{\frac{\Delta e}{T_0}\}. \tag{2.36}$$

If $p = 0$ make the change e ; otherwise make the change e with probability p. Repeat the evaluation and selection α times or until β changes. If there are no changes is α repetitions stop searching ; otherwise lower the temperature multiplying by $0 < \gamma < 1$, and continue searching.

There are also some polynomial algorithms for some special cases (i.e. for structures where each node can have only one parent, see [19] for a brief review).

Incomplete data structure learning

In the incomplete data case the parameters corresponding to a specific structure cannot be obtained without a parametric search algorithm. Hence, the scoring metrics described above, are not decomposable and cannot be evaluated directly. The structural EM algorithm evaluates the expected score of a network based on some initial network [49], [50].

$$Q(S', \Theta'|S, \Theta) = E_{S,\Theta}\{Score(S, D)\} \tag{2.37}$$

The expectation is taken with respect to $P(\mathbf{X_h}|D, S, \Theta)$. The computation of the expected score generally requires inference within the previous network (S, Θ).

The SEM algorithm starts with an initialization of the structure and corresponding parameters. At the E-step, the expected score is computed for the candidate structure. At the M-step, the search algorithm determines the next candidate structure to maximize the expected score. These two steps alternate until the maximizing structure is found.

An alternative SEM algorithm is also proposed that performs parametric maximization for some specified number of steps within structural maximization [49]. This leads to computational advantages, since parametric search is computationally cheaper than structure search.

2.5 Dynamic Bayesian Networks

Dynamic Bayesian networks (DBN) [31] extend the BN representation to dynamic processes. A DBN encodes the JPD of a time evolving set $\mathbf{X}[t] = \{X_1[t], \ldots, X_n[t]\}$ of variables. In other

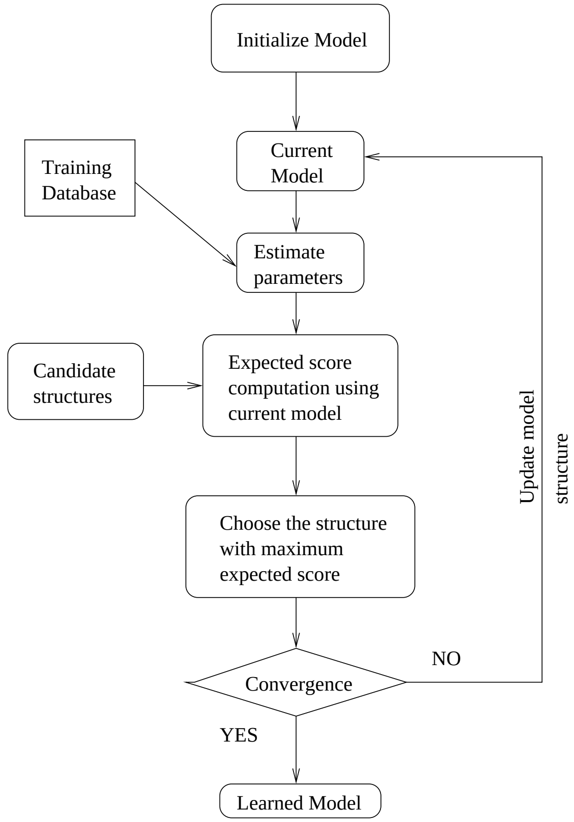


FIG. 2.3 – SEM algorithm

words, it represents the beliefs about possible trajectories of the dynamic process $\mathbf{X}[t]$. Following a similar notation with the static BN representation, the JPD for a finite time interval $[1, T]$ is factored as :

$$P(\mathbf{X}[1], \ldots, \mathbf{X}[T]) = \prod_{t=1}^{T} \prod_{i=1}^{n} P(X_i[t] | \mathbf{\Pi}_i[t]) \tag{2.38}$$

where $\mathbf{\Pi}_i[t]$ denotes the parents of $X_i[t]$ in the graphical structure of the DBN. The graphical structure of a DBN can be viewed as a concatenation of several static BNs linked with temporal arcs. We call each of these static networks a time slice[5] of the DBN. In the most general case, if no assumptions are imposed on the underlying dynamic process, the graph structure and the numerical parameterization of a DBN can be different for each time slice. In this case the DBN is considered as a (static) BN with $T \times n$ variables and the encoding of the JPD can be extremely complex. In general DBNs are defined for stationary processes where the network structure and parameters are time-invariant.

2.5.1 Representation

In the literature the DBN representation is most often used for stationary first order Markov processes. For this case, Friedman et al describes a simplified representation in terms of two static BNs defined over the variables of a single time slice [52]. The principle of the representation relies on the stationarity assumption which implies that the structure and the parameters of the DBN are repeating. The JPD is encoded using an initial network and an unrolled transition network. The initial network encodes the irregular structure in the boundary and specifies the distribution over initial states $\mathbf{X}[1]$. The transition network encodes the time-invariant transition probability $P(\mathbf{X}[t+1] | \mathbf{X}[t])$. The JPD for a finite time interval is obtained by *unrolling* the transition network for a sufficient number of time slices. The unrolling mechanism consists of introducing a set of variables for each time slice and replicating the structure and parameters of the transition network on these variables. Rearranging the terms in Eq.2.38 the JPD is factored over the initial and transition networks as :

$$P(\mathbf{X}[1], \ldots, \mathbf{X}[T]) = P_{B_I}(\mathbf{X}[1]) \prod_{t=2}^{T-1} P_{B_\rightarrow}(\mathbf{X}[t] | \mathbf{X}[t-1]) \ , \tag{2.39}$$

where $P_{B_I}(.)$ and $P_{B_\rightarrow}(.)$ are the probability densities encoded by the initial and transition networks, respectively.

Example Figure 2.4 illustrates this representation for a DBN defined over three variables : X_1, X_2 and X_3. The initial network that defines the prior JPD for time $t = 1$ is shown in Figure 2.4.a. In Figure 2.4.b, the stationary transition network structure is given. For an observation sequence of 4 time slices the unrolled network is obtained by concatenating the initial network with 3 slices of the transition network as shown in Figure 2.4.c.

In the following we extend the representation in [52] for a more general set of dynamic processes. Namely, we consider stationary processes $X[t]$ with the following conditional independence property :

$$P(X_i[t] | \mathbf{X}_1^{t+\tau_f}) = P(X_i[t] | \mathbf{X}[t-\tau_p], \ldots, \mathbf{X}[t+\tau_f]) \tag{2.40}$$

[5] A time slice is defined as the collection of the set of variables $\mathbf{X}[t]$ in a single time instant t and their associated parents $\mathbf{\Pi}[t]$ in the graph structure.

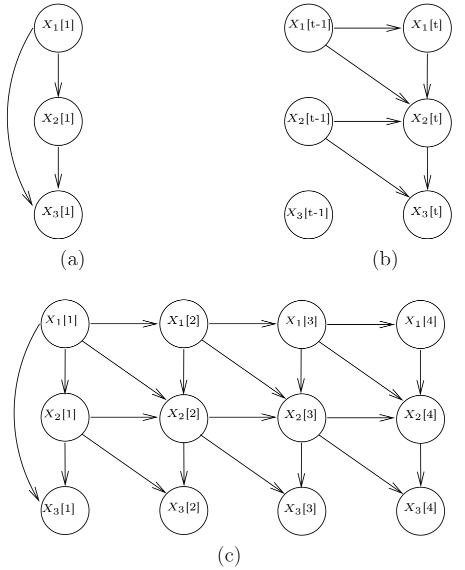


Fig. 2.4 – Illustration of DBN representation and the unrolling mechanism (modified from [52]). (a) Initial network. (b) Transition network. (c) Unrolled DBN for 4 time slices.

for some positive integers τ_p and τ_f. $\mathbf{X}_1^{t+\tau_f}$ denotes the set of variables in the range $[1, t+\tau_f]$. Graphically, the above assumption states that a variable at time t can have parents in the interval $[t-\tau_p, t+\tau_f]$. Using this general independence property rather than assuming Markovianity, we are able to take into account non-Markov processes.

Similar to [52] we use a transition network for the repeating structure of the DBN. The transition probability defined in Equation 2.40 is encoded using this network. However, the CI assumption in Equation 2.40 cannot be used for the first τ_p and the last τ_f time slices of the DBN. In these time slices, we do not have the repeating transition structure since the dependency time window vary with t. Assuming $T \geq \tau_p + \tau_f + 1$, the corresponding CI relations for these time slices can be stated as follows :

$$P(X_i[t] | \mathbf{X}_1^{t+\tau_f}) = P(X_i[t] | \mathbf{X}[t], \ldots, \mathbf{X}[t+\tau_f]) \ for \ 1 \leq t \leq \tau_p \tag{2.41}$$

$$P(X_i[t] | \mathbf{X}_1^{T}) = P(X_i[t] | \mathbf{X}[t-\tau_p], \ldots, \mathbf{X}[T]) \ for \ T-\tau_f+1 \leq t \leq T \tag{2.42}$$

The overall DBN structure is decomposed with respect to three time intervals : In $[1, \tau_p]$ and $[T-\tau_f+1, T]$ we have τ_p and τ_f different boundary structures, respectively. In $[\tau_p+1, T-\tau_f]$ we have the repeating transition structure. With this consideration, the DBN representation simplifies to $(\tau_p + \tau_f + 1)$ static BNs : τ_p initial networks, τ_f final networks and a transition network. The initial and final networks encode the non-stationary dynamics of the process on the boundaries. The transition network encode the stationary dynamics. By unrolling the transition network, one can construct the DBN for a required number of time slices. However care must be taken to properly concatenate the initial and final networks.

Using this representation the factorization of the JPD can be reinterpreted in terms of the probability densities of each static BN as :

$$P(\mathbf{X}[1],\ldots,\mathbf{X}[T]) = \prod_{t=1}^{\tau_p} P_{B_{I_t}}(\mathbf{X}[t]|\mathbf{\Pi}[t]) \prod_{t=\tau_p+1}^{T-\tau_f} P_{B_{\rightarrow}}(\mathbf{X}[t]|\mathbf{\Pi}[t]) \prod_{t=T-\tau_f+1}^{T} P_{B_{F_t}}(\mathbf{X}[t]|\mathbf{\Pi}[t]) \ , \tag{2.43}$$

where $\mathbf{\Pi}[t]$ denotes the set of parents for the variables $\mathbf{X}[t]$ in the corresponding initial, transition and final networks. The parameterization of the associated conditional probabilities for each of these networks, are defined individually.

In the following example we illustrate the construction of the initial, transition and final networks for the representation of a given dynamic process.

Example Consider a DBN for the variables $X_o[t], X_h[t]$, with the following CI assumptions :

$$P(X_h[t]|\mathbf{X}_1^{t+1}) = P(X_h[t]|X_h[t-2], X[t-1]) \tag{2.44}$$
$$P(X_o[t]|\mathbf{X}_1^{t+1}) = P(X_o[t]|X_h[t-1], X[t], X[t+1]) \tag{2.45}$$

The DBN structure unrolled for 5 time slices is given in Figure 2.5. The maximum past and future dependence parameters are, $\tau_p = 2$ and $\tau_f = 1$. Therefore, the DBN can be represented with 2 initial, 1 final and a transition static networks. The transition network can directly be constructed from the dependency assumptions as shown in Figure 2.6(c).

In constructing the initial and final networks we have to consider the dependency assumptions for each time slice in the boundaries. For $t = 1$, the assumptions for $X_o[1]$ and $X_h[1]$ can be restated as follows.

$$P(X_h[1]|\mathbf{X}_1^2) = P(X_h[1]) \tag{2.46}$$
$$P(X_o[1]|\mathbf{X}_1^2) = P(X_o[1]|X_h[1], X_h[2]) \tag{2.47}$$

These assumptions state that $X_h[1], X_h[2]$ are the only parents of $X_o[1]$ and that $X_h[1]$ has no parent. The implied structure is shown in Figure 2.6(a). Similarly considering the assumptions for $t = 2$ and $t = T$, one obtains the network structures given in Figure 2.6(b) and 2.6(d), respectively.

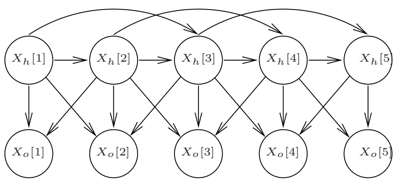


FIG. 2.5 – Example DBN structure unrolled for 5 time slices, $T = 5$

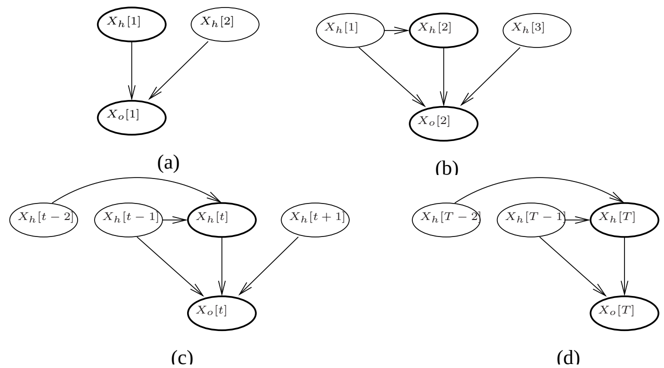


FIG. 2.6 – (a), (b) Initial networks for $t = 1, 2$. (c) Transition network. (d) Final network for $t = T$

2.5.2 Inference in DBNs

The inference problem in DBNs is analogous to BN inference problem such that the desired quantity is the posterior marginal distribution of a set of hidden variables given a sequence of observations (belief update) : $P(\mathbf{X}_h[t] \mid \mathbf{X}_o[1],\ldots,\mathbf{X}_o[\tau])$, where $\mathbf{X}[t] = \{\mathbf{X}_h[t], \mathbf{X}_o[t]\}$ is a set of time evolving variables in which $\mathbf{X}_o[t]$ and $\mathbf{X}_h[t]$ denote observed and hidden variables, respectively. Inference in time series is generally referred to as *filtering* ($\tau = t$), *smoothing* ($\tau > t$) and *prediction* ($\tau < t$) according to observation time window used in computations.

A direct approach to infer probabilities in a DBN, is to construct a huge static BN for the desired number of time slices and then use the general inference algorithms for static BNs. However, this requires that the end of the time sequence is known a priori. Furthermore, the computational complexity of this approach can be extremely demanding (especially in terms of memory). Therefore in general, DBN inference is performed using recursive operators that update the belief state of the DBN as new observations become available. The principle is similar to message passing algorithm for static BNs. The idea is to define messages on a Markov blanket of variables that d-separates the past from the future and use a forward-backward procedure to distribute all evidence along the DBN [132, 96, 78]. This technique requires only a time window of variables to be kept in memory. These algorithms are indeed generalization of the well known forward-backward (Baum-Welch) algorithm [106] in HMMs and special cases of the JLO algorithm [74].

Forward-Backward Algorithm

An *inefficient* way to perform inference in DBNs would be to convert them to HMMs with enlarging the state space and applying the forward-backward algorithm [106]. The forward-backward algorithm defines two recursive messages by exploiting the following CI assertions of HMMs :

– $X_h[t+1]$ is independent of past observations given $X_h[t]$

$$X_h[t+1] \perp \mathbf{X}_{o1}^{\ t} \mid X_h[t]$$

– $X_h[t-1]$ is independent of future observations given $X_h[t]$

$$X_h[t-1] \perp \mathbf{X}_{ot-1}^{T} \mid X_h[t]$$

These messages are defined on the hidden variable which is indeed sufficient to keep the information coming from the past or future. The recursive computation corresponds to expanding the message to the next slice by multiplications with corresponding conditional probabilities and then by marginalizing out the hidden variable in the previous slice. The forward message is defined as the joint probability of the hidden state variable and the observations upto and including time t :

$$\begin{aligned} \alpha_t(i) &= P(\mathbf{X}_{o1}^{t}, X_h[t]=i) && (2.48)\\ &= \sum_j \alpha_{t-1}(j) P(X_h[t]=i \mid X_h[t-1]=j) P(X_o[t] \mid X_h[t]=i) && (2.49) \end{aligned}$$

The backward message is defined as the conditional probability of observations from time $t+1$ up to T given the hidden state variable at time t :

$$\begin{aligned} \beta_t(i) &= P(\mathbf{X}_{ot+1}^{T} \mid X_h[t]=i) && (2.50)\\ &= \sum_j \beta_{t-1}(j) P(X_h[t+1]=j \mid X_h[t]=i) P(X_o[t+1] \mid X_h[t+1]=j) && (2.51) \end{aligned}$$

The posterior marginal is then computed as :

$$\begin{aligned} \gamma_t(i) &= P(X_h[t]=i \mid \mathbf{X}_{o1}^{T}) && (2.52)\\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)} && (2.53) \end{aligned}$$

Frontier algorithm

The recursive formulation of forward-backward message passing in HMMs is based on the CI assumption that the hidden variable at time t d-separates past and future. Similarly the frontier algorithm is defined for a DBN with first order Markov assumption where the hidden variables at time t constitute a Markov blanket that d-separates past and future [131]. In that, the forward and backward messages are constructed over the hidden variables in a time slice. Recursive computation of the messages is performed by including variables from the next slice by multiplications with corresponding conditional probabilities and marginalizing out variables in previous slice.

Let $\mathbf{X}_h[t] = \{X_h^1[t], \ldots, X_h^N[t]\}$ denote the hidden variables at time t. The frontier for time t is $F_t = \mathbf{X}_h[t]$ and the observed variables in the frontier is denoted as O_F.

In the forward pass the forward message is defined as :

$$\alpha_t = P(F_t, O_F, \mathbf{X_o}_1^{t-1})$$

where $\mathbf{X_o}_1^{t-1}$ denotes the observations upto time t. In the forward pass the frontier is expanded by adding variables from the next slice. In order to keep the Markov blanket property of the frontier set, a variable $X^i[t+1]$ is added to the frontier only if all its parents are already in the frontier. The new frontier is defined as $F = F_t \cup X^i[t+1]$ and the updated message is obtained by multiplying the message by the conditional probability of the new variable :

$$P(F, O_F, \mathbf{X_o}_1^{t-1}) = \alpha_t P(X^i[t+1] | F_t, O_F).$$

A variable $X^j[t]$ is removed from the frontier when all its children are in the frontier. The reduced frontier is defined as $F = F \backslash X^j[t]$ and the new message is obtained by marginalizing out $X^j[t]$:

$$P(F \backslash X^j[t], O_F, \mathbf{X_o}_1^{t-1}) = \sum_{X^j[t]} P(F, O_F, \mathbf{X_o}_1^{t-1})$$

When all variables in slice $t+1$ is added and all variables in slice t are removed the message in the frontier is obtained as :

$$\alpha_{t+1} = P(F_{t+1}, O_F, \mathbf{X_o}_1^{t})$$

In the backward pass the frontier is advanced from t to $t-1$ using the opposite variable order used in the forward phase. The backward message is defined as :

$$\beta_t = P(\mathbf{X_o}_{t+1}^{T} \mid F_t, O_F)$$

When a node $X^i[t-1]$ is added to the frontier the domain of the message is expanded to include the values of this variable. Since F_t d-separates $\mathbf{X_o}_{t+1}^{T}$ and $X^i[t-1]$ no multiplications are required.

$$P(\mathbf{X_o}_{t+1}^{T} \mid F_t, X^i[t-1], O_F) = P(\mathbf{X_o}_{t+1}^{T} \mid F, O_F)$$

When a node $X^j[t]$ is removed from the frontier the marginalization is performed as :

$$P(\mathbf{X_o}_{t+1}^{T} \mid F \backslash X^j[t], O_F) = \sum_{X^j[t]} P(\mathbf{X_o}_{t+1}^{T} \mid F, O_F) P(X^j[t] \mid O_F, F \backslash X^j[t])$$

When all variables in slice $t-1$ is added and all variables in slice t are removed from the frontier the message in the frontier is obtained as :

$$\beta_{t-1} = P(\mathbf{X_o}_{t}^{T} \mid F_{t-1}, O_F)$$

The posterior marginal of the frontier set is then computed using the forward and backward messages :

$$P(F_t | \mathbf{X_o}_1^{T}) = \frac{\alpha_t \beta_t}{\sum_{F_t} \alpha_t \beta_t}$$

Interface algorithm

The interface algorithm is a more efficient algorithm where only a subset of variables "forward interface" is used to define the Markov blanket that d-separates each time slice [96]. For each time slice with variables X_t, the set of "forward interface" nodes $I_t \subseteq X_t$ are defined as the nodes that have children in the next time slice X_{t+1}. The algorithm uses a junction tree structure in which a subtree of each time slice is connected with separators consisting of interface nodes. In order to have this specific junction tree structure the subtree of each time slice is constructed using the following procedure.

First a modified two-slice DBN is defined that consists of the interface nodes of the first slice and all nodes of the second, i.e. all non-interface nodes and their arcs are removed from the first slice. The junction tree of this modified DBN constitutes the subtree of each time slice. The junction tree construction procedure is the same as the JLO algorithm (see Appendix A) with the additional constrained that the interface nodes I_{t-1} and I_t each should form a clique, D_t and C_t respectively. The message passing algorithm then proceeds as follows.

In the forward phase the message that separates consecutive time slices is defined as :

$$P(I_{t-1}|\mathbf{X}_{o1}^{t-1})$$

This message is passed from C_{t-1} to D_t using the clique tree message passing algorithm. Next a collect-evidence phase on the subtree of time slice t updates the potential of the clique C_t with the information from the previous slices and current observations. Next the potential on C_t is marginalized onto I_t to compute $P(I_t|\mathbf{X}_{o1}^{t})$. In the backward phase first the potential on C_t is distributed to the subtree and then the potential on D_t is passed to C_{t-1}.

The advantage of this algorithm is that message passing can be performed independently in each subtree.

Inference with constrained elimination ordering

For more general cases, Kjaerulff describes a computational scheme for dynamically updating the inference engine with incoming data [78]. In this scheme, the interface nodes in each time slice are defined as the set of nodes and their children that has a parent in the previous time slices. Kjaerulff proves that these nodes d-separate two consecutive time slices. The proposed algorithm uses a junction tree representation for a window of variables that constitutes a Markov blanket over a fixed number of time slices. This junction tree representation for a fixed length time window is called the active model. In filtering (i.e. forward message passing) the active model window is expanded toward future and reduced from the past, i.e. new time slices are included and old slices are eliminated. When a new time slice is added to the active model, first the expanded junction tree is constructed to include the cliques with the additional variables. The construction of the junction tree is based on a constrained elimination ordering of the variables in each time slice such that all variables in time slice $t-1$ are eliminated before the variables in t. This procedure prevents the construction of very large cliques that span many time slices and guarantees a repeating structure of the junction tree. The message propagation algorithm on this junction tree distributes the new observations on the active representation. To remove a time slice $t-k$ from the active model it suffices to remove all cliques of the junction tree that consists of the variables at time $t-k$ and keep the marginal of the interface nodes between $t-k$ and $t-k+1$.

The on the fly construction of the junction tree could be computationally very demanding. For DBNs that have a stationary structure for each time slice, a practical trick is to first construct a junction tree for a fixed time window length using a constrained elimination ordering [132]. The resulting junction tree has a repetitive structure which can be unrolled for a given observation sequence length. Indeed one can recover interface cliques on this repetitive structure to perform the forward backward message passing procedure on interface cliques.

2.5.3 Learning DBNs

Learning algorithms for BNs can indeed be generalized for DBNs [52]. Learning in BNs is based on the maximization of a scoring of the network with respect to structure and/or parameters. In previous sections we described different scorings of a BN for parameter and structure learning. In parameter learning the score is either data likelihood or posterior distribution of the parameters given data. In structure learning the score is either BIC/MDL or BD/BDe. These scorings can be generalized for DBNs. The common property of these scores is that they are decomposable such that maximization can be performed for each variable independently. This is

achieved by exploiting the factorization of the JPD over the conditional probabilities of each variable. Using the factorization of DBNs over static BNs the DBN scoring is similarly decomposed over each static BN and consecutively over the conditional probabilities of each variable in each static BN. However, to learn the static networks the observation sequence should be properly feed to each network (i.e the transition network should be feed with the proper transitions in the data). Hence complete data learning is equivalent to partitioning the data sequence to each static BN and applying BN learning algorithms directly. For incomplete data case, the score cannot be decomposed due to marginalization over hidden variable values. The use of EM algorithm is such cases necessitates the computation of the expected score based on the posterior distribution of the hidden variables. The posterior is obtained by inference on the unrolled DBN.

Chapitre 3

Acoustic Modeling with DBNs

Sommaire

3.1 Introduction . . . 65
3.2 Contributions . . . 66
3.3 Overview of Acoustic modeling using DBNs . . . 66
3.3.1 Factored state space models . . . 67
3.3.2 Explicit observation dependence modeling . . . 69
3.3.3 Multi-stream techniques . . . 71
3.4 Structure learning of DBNs for acoustic modeling . . . 71
3.5 A Class of DBNs for acoustic modelling . . . 72
3.5.1 Numerical parameterization . . . 73
3.5.2 State transition topology . . . 75
3.5.3 Model complexity . . . 75
3.6 Learning DBNs for speech recognition . . . 76
3.6.1 Maximum likelihood parameter estimation . . . 76
3.6.2 Learning the structure . . . 77
3.6.3 Embedded training for Speech recognition . . . 78
3.7 Inference for Speech Recognition . . . 79
3.7.1 Isolated speech recognition . . . 79
3.7.2 Continuous speech recognition . . . 79
3.7.3 Brute force approach . . . 79
3.7.4 A fast decoding algorithm . . . 81
3.8 Experiments . . . 85
3.8.1 Isolated digit recognition . . . 85
3.8.2 Connected digit recognition . . . 87
3.8.3 Continuous phoneme recognition . . . 88

3.1 Introduction

Statistical speech recognition systems are based on the extraction and exploitation of statistical information from a set of labeled speech data. In this process, acoustic models play the role to encode the information related to speech production mechanisms. Statistically, an acoustic model defines a parameterized representation for the posterior probability distribution $P(\mathbf{O}|v)$

of a set of speech observations $\mathbf{O}$ for the given acoustic unit v. This distribution does not have a simple functional form. The attempts to obtain an efficient representation greatly relies on a set of simplifying assumptions for the underlying physical processes. These assumptions are merely concerned with the probabilistic and causal relationships among the variables that dominate the process. They should be based on prior knowledge on acoustic phenomena such that the exploitation of statistical data will yield consistent and reliable modelling. However, they should also be flexible enough to allow adaptation to different characteristics inherent in the observed data. That is, the simplification assumptions and the observations of the actual speech process should be in harmony. This requires a mechanism to manipulate and improve our prior assumptions with statistical methodologies. The Bayesian Networks formalism provides a natural framework for this task.

In the sequel we briefly describe some successful approaches to acoustic modeling in this setting.

3.2 Contributions

- A compilation of acoustic modeling approaches using dynamic Bayesian networks.
- A new class of acoustic models defined in dynamic Bayesian networks framework.
- A novel approach to learning variable dependencies in acoustic modeling.
- A decoding algorithm for continuous speech recognition using the proposed set of models.
- Evaluation of the proposed approach in isolated and continuous speech recognition.

3.3 Overview of Acoustic modeling using DBNs

The most commonly used DBNs in speech recognition is the first-order hidden Markov models (HMMs)[106]. In HMMs, the speech signal is assumed to be governed by a hidden (unobserved) dynamic process. The probabilistic relations among the hidden and the observed processes state that the hidden process is first-order Markov, and that each observation depends only on the current state of the hidden process. Indeed, the conditional independence assertions of an HMM are encoded by the graphical structure showed in Figure 3.1. Each node in this structure represents a random variable : $X_h[t]$ specifies the state of the hidden process and $X_o[t]$ denotes the observation at time t.

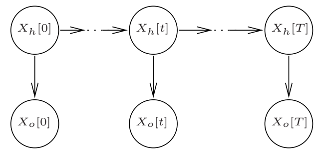


Fig. 3.1 – HMM represented as a DBN

The numerical parameterization of the conditional probabilities are, in general, specified by a multinomial distribution for the first order hidden process and a conditional Gaussian density for the observations.

The DBN framework is not limited to HMMs. HMM is only a specific instance among the set of models that can be defined in this setting. Indeed most probabilistic modeling techniques used in speech recognition can be defined as particular cases of BNs. In [3], Bilmes provides a detailed survey that compiles several modeling techniques in speech recognition by reformulating them as Bayesian networks. In this section we cite a non-exhaustive list of acoustic modeling approaches that have been recently proposed. Most of these approaches are proposed as novel modeling techniques whereas we classify them as particular cases of dynamic Bayesian networks. We present these approaches in three categories :

- Factored state space models
- Explicit observation dependence modeling
- Multi-stream observation modeling

3.3.1 Factored state space models

In this category of models the observations are assumed to be governed by several hidden processes. The state space is factored such that each controlling process is represented by individual state dynamics. The factorization of the state space is achieved using hypothetical conditional independence relations between these processes. In this category of models we include mixture of Gaussian HMMs, factorial HMMs [58] and coupled HMMs [102] (see Figure 3.2).

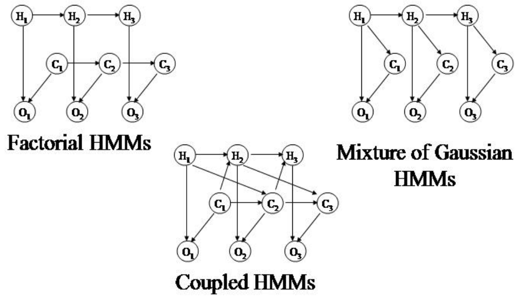


Fig. 3.2 – Factored state space models

In [132], Zweig introduces a general framework to manually design DBN structures for modeling various speech phenomena. The proposed DBN structures are aimed to model speech-articulators, accent, speaking rate and similar phenomena. The principles of the model design strategy are based on introducing additional variables in each time slice of the HMM structure and to impose dependencies according to expert knowledge. The proposed structures are given in Figure 3.3. The *correlation* network models the correlations between observations within a single time slice, *Pd-correlation* network models these correlations conditioned on the hidden state (phone) variable, *chain* network models state independent temporal continuity of the auxiliary variable and *articulator* network models the two dependent state streams. Once the structure of the model is designed manually the associated parameters are learned from a given training database. The main inconvenience in this strategy is that although the model is designed to model

factored dynamics, the applied training strategy does not guaranty that the additional factor models the desired quantity, i.e. accent, speaking rate etc. In [117], Stephenson uses the *articulator* network to train models with acoustic and articulatory variables using articulatory measurements such that the additional factor is trained with relevant information. In [118, 120, 119] the authors incorporates pitch frequencies as the additional factor using the *chain* and *articulator* networks. Even with appropriate training strategy these models do not yield significant improvements due

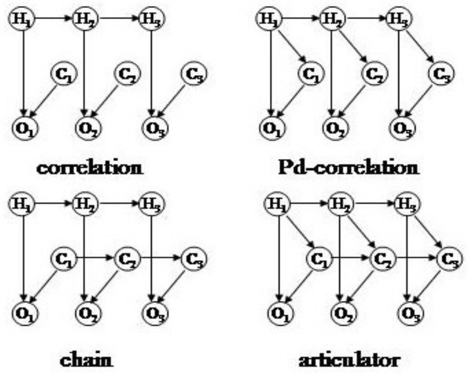


Fig. 3.3 – DBN models proposed to incorporate factorized speech dynamics [132].

to discrete parameterization of the networks. The observations are quantized to facilitate DBN inference problem which in turn yields erroneous modeling.

Recently, Shinozaki proposed an approach to model speaking rate using a hidden variable that represents the variation of the "mode" of the speaking rate [113]. The approach uses a structure similar to *correlation* network in Figure 3.3. The observation output density is defined as a mixture of Gaussians, hence significant improvements are attained with respect to classical HMMs.

In [94], Markov proposes a DBN structure for noisy speech recognition. Two auxiliary factors are integrated to the classical HMM model to represent noise type N and level S. Given a multi-condition database with different noise types and levels, individual HMM models are trained for each distinct combination of these variables. In recognition a DBN (see Figure 3.4 is constructed to combine all these models assuming uniform priors for the variables S and N [94]. This model yields relatively good performances in multi-condition training situations, however the performance in mismatched conditions is poor. This is due to the applied training strategy that overfits to individual noisy conditions in training data.

Korkmazsky recently proposed a novel training strategy with an auxiliary environment variable using the *correlation* network structure (see Figure 3.3) [79]. This approach (called *hidden factor dynamic Bayesian networks*) uses an initial clustering of speech data based on different environmental variations to train a model for each different condition. At the second stage the composite DBN model (with the auxiliary variable) is constructed and refined with further DBN training. The experimental results shows significant improvement with respect to equivalent

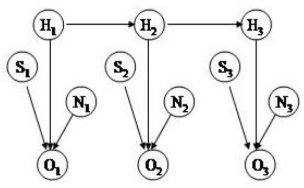


FIG. 3.4 – DBN model for noisy speech recognition [94] with two auxiliary factors : noise level (S), noise type (N).

HMM models.

In [88], Logan proposes the use of factorial HMMs (FHMM) [58] in speech recognition. The FHMM is used to model the dynamics of static, Δ and $\Delta\Delta$ features separately. The state space is factored into conditionally independent hidden streams. Each stream is first trained individually using respective observations, i.e. static, Δ and $\Delta\Delta$ features. Then all streams are combined in an FMM structure (see Figure 3.2) using two different parameterizations of the observation output density :

- Linear FHMM : Gaussian density where the mean is a linear combination of the means of each stream.
- Streamed FHMM : Product of Gaussian densities of each stream.

This technique does not improve with respect to classical HMMs due to the fact that the dynamics of Δ and $\Delta\Delta$ are indeed similar to static features. Hence the model introduces unnecessary parameters for modeling the same dynamics.

Motivated from the fact that speech is produced by a system of loosely coupled articulators Nock proposes the use of coupled HMMs in ASR [102]. Coupled HMMs (see Figure 3.2) are indeed FHMMs with additional dependencies between each stream. The results are comparable to classical HMMs and multiband approaches. Richardson applies coupled HMMs for articulatory speech recognition using only articulatory observations. The authors name the proposed model as *hidden articulator Markov models*, which is indeed a coupled HMM structure with 8 streams [107]. In [100], Nefian presents the application of such models for audio visual ASR.

In [87], Livescu proposes a novel network structure to explicitly represent models of hidden features, such as articulatory or other phonological features, for automatic speech recognition. The authors show that the proposed representation is more expressive and parsimonious than implicit representation in HMMs. This model can be considered as a FHMM where each factor is correlated through a hidden variable H. The model structure is shown in Figure 3.5.

3.3.2 Explicit observation dependence modeling

In this category of models the dependence of observations in successive time frames are modeled explicitly. Whereas in factored state space models this is achieved through the hidden factor variables. The main motivation to these approaches is to provide a solution to the frame dependency problem in acoustic modeling which arises mainly due to overlapping of analysis windows used in signal parameterization. The practical approach to this problem in classical HMMs is to augment the observation vectors with its time derivatives. This technique is widely

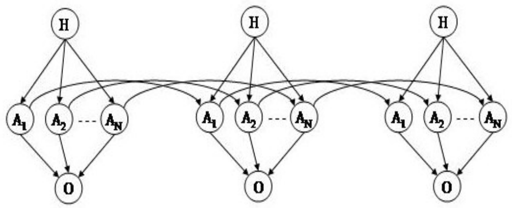


FIG. 3.5 – Hidden feature model for speech recognition

accepted.

One of the first attempts is proposed by Wellekens as *correlation HMMs* [127] where the correlation between successive observation vectors are modeled as a conditional linear Gaussian density. In [77], a linear predictive model is used for the observations of an HMM model. The graphical representation of these models is shown in Figure 3.6.

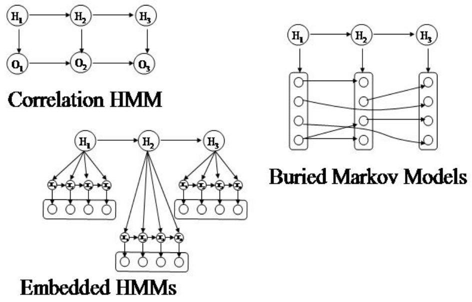


FIG. 3.6 – DBN models for explicit observation correlation modeling.

In [4, 5, 6], Bilmes proposes a modeling strategy using *Buried Markov models* (BMM) (see Figure 3.6). The approach is based on removing the CI properties of an HMM whenever the model is found to be poorly representing the given training data. Each element of the observation vector is considered as a separate variable and the dependence structure among these variables are learned using a discriminative mutual information metric. The resulting acoustic models represents a structural discrimination for each state of the underlying first order hidden Markov process [14].

Another approach that can be considered in this category is embedded HMMs (also called HMM2), where correlation between the observations in a single time slice are modeled with a secondary fixed length HMM. These models attempt to capture the dynamics of the feature vector for each frame [126].

3.3.3 Multi-stream techniques

In this category of models we cite the approaches that explicitly model more than one observation stream. The applications include multi-band and multi-modal speech recognition. A typical model structure is given in Figure 3.7.

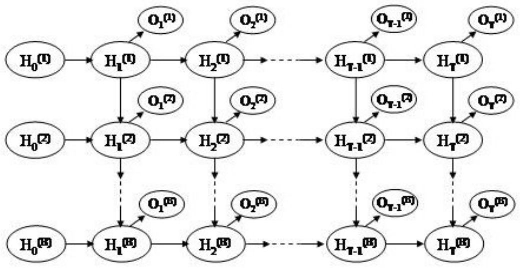


FIG. 3.7 – Factored state multi-stream DBN model

In [26], Daoudi uses this model for multi-band speech recognition. Each observation stream corresponds to speech features in different frequency bands. The DBN structure enables modeling of the speech dynamics in each band individually. This approach is particularly interesting for noisy speech recognition where only a part of the spectrum is contaminated by noise. The multi-stream modeling compensates the degradation arising due to corrupted band and the recognition performance is significantly higher than classical HMMs. Furthermore the coupling in the factored state space improves the modeling with respect to classical multi-band recognition techniques [13].

A similar strategy is used in [130] to model the dynamics of different speech parameterizations. Several parameterization techniques are combined in the model structure to complement respective robust characteristics of each technique. Another application of this model is presented in [15] where additional cross-stream dependencies are learned for different parameterizations.

3.4 Structure learning of DBNs for acoustic modeling

In [36] we proposed a methodology to learn the CI assertions of acoustic models based on structural learning of DBNs. As in HMMs, we consider that the observations are governed by a hidden process. On the other hand, we do not make any *a priori* dependency assumption between the hidden and observed processes. Rather, we give data a relative freedom to dictate the appropriate dependencies. In other words, we learn the dependencies between (hidden and observable) variables *from data*. The principle of this methodology is to search over all possible "realistic" dependencies, and to chose the ones which best explain the data. This approach has the advantage to *guaranty* that the resulting model represents speech with higher fidelity than HMMs. Moreover, a *control* is given to the user to specify the maximal dependency structure and hence to make a trade-off between modeling accuracy and model complexity.

In the following, we describe the class of DBN structures we use. We specify this class as the search space of our training algorithm and choose our acoustic models among this set. The training algorithm is described in Section 3.6. We first specify the set of structures (i.e. CI assertions) for our models. Next we define the numerical parameterization for the conditional probability densities to complete the definition of a DBN in this class.

3.5 A Class of DBNs for acoustic modelling

In acoustic modeling we consider that speech observations are governed by a discrete hidden process. Our goal is to learn the DBN structures (and their corresponding parameters) which "best" explain data, i.e., the "appropriate" dependencies between the hidden and observed variables of the speech process. This requires a search over a class of structures. Searching over all possible DBN structures would be computationally infeasible. Therefore, we restrict ourselves to a small but rich set of structures that represents only *realistic* dependencies, in a physical and computational sense.

In specifying the class of DBNs we first define the set of plausible dependencies among the hidden and observed processes. First, we do not consider direct dependencies between observable variables. We believe this is a reasonable assumption because, short-time correlations between cepstral coefficients are indirectly considered by Δ and $\Delta\Delta$ coefficients. Second, we do not allow future dependencies between the hidden variables. This is because past dependencies must be considered given causality of speech. Thus, given that the graph has to be acyclic, we cannot have both past and future dependencies between hidden variables. Finally, we do not allow a hidden variable to have an observable one as a parent. This is because there exists no exact algorithm to infer BNs where continuous variables have discrete children. Therefore, the remaining authorized dependencies in the class of structures we are considering are the following.

Let $\mathbf{X}[t] = \{X_h[t], X_o[t]\}$ be the set of hidden and observed variables at time t respectively. The notation $\mathbf{X}_1^T$ denotes the variables in the range $[1, T]$, i.e. $\{X[1], \ldots, X[T]\}$. The generalized conditional independence assumptions are stated as :

Assumption 1 *The hidden state variable, $X_h[t]$, is independent of previous observations and states given the values of last κ hidden states.*

$$X_h[t] \perp \{\mathbf{X}_{h1}^{\,t-\kappa-1}, \mathbf{X}_{o1}^{\,t-1}\} \mid \mathbf{X}_{h\,t-\kappa}^{\,t-1}$$

Assumption 2 *The observation variable $X_o[t]$ is independent of all other variables given the hidden variables in the time window $[t-\tau_p, t+\tau_f]$, for some positive integers τ_p and τ_f,*

$$X_o[t] \perp \{\mathbf{X}_{h1}^{\,t-\tau_p-1}, \mathbf{X}_{h\,t-\tau_f+1}^{\,T}, \mathbf{X}_{o1}^{\,t-1}, \mathbf{X}_{o\,t+1}^{\,T}\} \mid \mathbf{X}_{h\,t-\tau_p}^{\,t+\tau_f}$$

The graphical representation of these CI assumptions is given in Figure 3.8 for a time slice. Using these assumptions the conditional probability distribution of the hidden and observed variables are defined as :

$$P(X_h[t] \mid \mathbf{\Pi}_h[t]) = P(X_h[t] \mid X_h[t-\kappa], \ldots, X_h[t-1]) \;, \tag{3.1}$$

$$P(X_o[t] \mid \mathbf{\Pi}_o[t]) = P(X_o[t] \mid X_h[t-\tau_p], \ldots, X_h[t+\tau_f]) \;. \tag{3.2}$$

Equation (3.1) states that the hidden process is Markovian of order κ. Equation (3.2) states that the observation at time t depends on states of the hidden process in the interval $[t-\tau_p, t+$

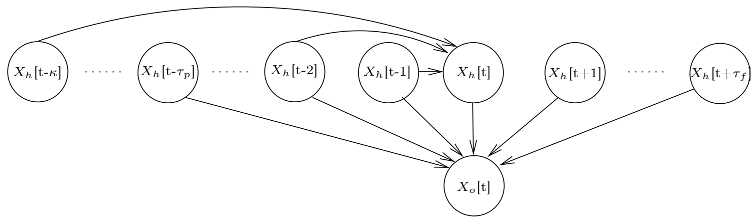


Fig. 3.8 – Graphical representation of CI assumptions in the proposed class of DBNs.

τ_f]. Notice that using these independence properties we are able to take into account various phenomena in the speech production process. For instance, the non-Markov property on the observations may take into account the well known anticipation phenomena.

Based on these CI assumptions and stationarity we define a unique DBN structure using the triple (κ, τ_p, τ_f). In that, κ defines the order of the hidden process, τ_p and τ_f define the dependence window of the observation on the hidden process. When $(\kappa, \tau_p, \tau_f) = (1, 0, 0)$, the model reduces to the standard first-order HMM (Figure 3.1) where Eq.(3.1) defines the state transition probabilities and Eq.(3.2) defines the observation probabilities.

Using this structure specification we define a class of structures by specifying a lower and upper limit on the parameters κ, τ_p and τ_f. We set the lower limit as the first order HMM structure, i.e. $(\kappa, \tau_p, \tau_f) = (1, 0, 0)$. The upper limit $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$ provides a control mechanism to restrict the size of the search class and therefore reduce the computational complexity of the modeling. The class of model structures for $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}}) = (2, 1, 1)$ is given in Figure 3.9.

3.5.1 Numerical parameterization

Consider a DBN with structure $S = (\kappa, \tau_p, \tau_f)$. This DBN is represented with $\max(\kappa, \tau_p)$ initial and τ_f final networks. If each discrete hidden variable $X_h[t]$ takes its values in the set of ordered labels $I = \{1 \ldots N\}$, and each observable variable $X_o[t]$ has a conditional Gaussian density, the numerical parameterization $\Theta = \{\Theta_h, \Theta_o\}$ of these networks is defined as follows.

The set of state transition probabilities, $\Theta_h = \{a_{\mathbf{i}j}[t]\}$, specifies the probability that the value of the hidden variable equals j, conditioned on its parents configuration $\mathbf{i}$.

$$a_{\mathbf{i}j}[t] \quad \triangleq \quad P(X_h[t] = j \mid \mathbf{\Pi}_h[t] = \mathbf{i}), \text{ for } j \in I, \ \mathbf{i} = [i_{t-\kappa}, \ldots, i_{t-1}] \in I^{\kappa} \tag{3.3}$$

The conditional probability densities $b_{\mathbf{i}t}(x_{ot})$ specify the probability of a particular observation $X_o[t] = x_{ot}$ in each network conditioned on the parent set configuration.

$$b_{\mathbf{i}t}(x_{ot}) \triangleq P(X_o[t] = x_{ot} \mid \mathbf{\Pi}_o[t] = \mathbf{i}), \quad \text{for} \quad \mathbf{i} = [i_{t-\tau_p}, \ldots, i_{t+\tau_p}] \in I^{\tau_p+\tau_f+1} \tag{3.4}$$

If we consider Gaussian densities,

$$b_{\mathbf{i}t}(x_{ot}) \quad \sim \quad \mathcal{N}(\mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t]) \tag{3.5}$$

the parameterization of the output probabilities are in terms of the means and covariances, $\Theta_o = (\mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t])$.

The (possibly) vector-index $\mathbf{i}$ is over all possible values of the variable's parents which depend on the structure of the corresponding time slice. For the specific structure space defined,

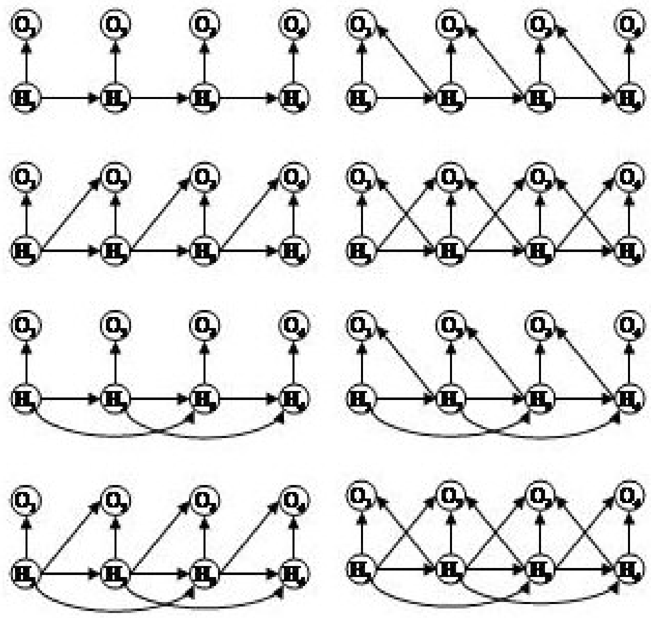


Fig. 3.9 – The class of structures for $(1, 0, 0) \leq (\kappa, \tau_p, \tau_f) \leq (2, 1, 1)$.

$\mathbf{i}$ is a point in the cartesian space I^m, where m is the number of parents of a variable. The dimensionality of $\mathbf{i}$ is different for each initial, transition and final network. Given the stationarity assumption, the parameters of the conditional probability distributions for the transition network are constant over time, i.e. the structure is self-repeating.

$$\begin{aligned} a_{\mathbf{i}j}[t] &= a_{\mathbf{i}j}\ , \quad for \quad t \in [\kappa+1, T] \\ \mu_{\mathbf{i}}[t] &= \mu_{\mathbf{i}}\ , \Sigma_{\mathbf{i}}[t] = \Sigma_{\mathbf{i}}\ , \quad for \quad t \in [\tau_p+1, T-\tau_f] \end{aligned}$$

Hence a DBN in the class of structures we defined is completely defined as : $S = (\kappa, \tau_p, \tau_f)$, $\Theta = (a_{\mathbf{i}j}, a_{\mathbf{i}j}[t], \mu_{\mathbf{i}}, \Sigma_{\mathbf{i}}, \mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t])$.

3.5.2 State transition topology

In general the graphical representation of HMMs used in the literature employs the finite state automaton representation where the nodes represent state values and the arcs represent allowed state transition (see Figure 3.10). This representation is useful when the state transition topology is to be emphasized. However the associated CI properties of the model is completely ignored. Whereas the BN graphical representation (see Figure 3.1)is based on CI properties and it neglects the underlying state transition topology. Therefore although these two graphical representations denote the same model they encode different aspects of the model.

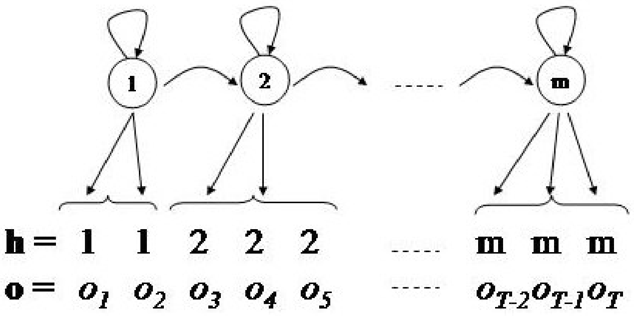


FIG. 3.10 – Left-to-right state transition HMM represented as a finite state automaton.

In the context of this paper we employ the BN representation and encode the associated state transition topology of the model in the state transition probability tables.

3.5.3 Model complexity

In order to provide a measure of complexity for different model structures we compute the number of parameters required to encode the transition network (for a single time slice) for a DBN with structure (κ, τ_p, τ_f). Let N be the number of states for the hidden process and M be the number of parameters required to encode the observation probability density for a given configuration of its parents. Then, for a left-to-right topology, the required number of parameters

to encode the network is given by :

$$C = 2^{\kappa} N - \sum_{l=1}^{\kappa} l 2^{\kappa - l} + M \times [2^{\beta} N - \sum_{l=1}^{\beta} l 2^{\beta - l}]. \tag{3.6}$$

β is defined as $\beta = \tau_p + \tau_f$ (we assume that $\kappa, \beta \leq N$). This quantity can be used to compare the complexities of different structures. For the baseline HMM, this quantity is computed for $(\kappa, \tau_p, \tau_f) = (1, 0, 0)$. Therefore in general, a DBN model requires a lot more parameters than a standard HMM using the same number of states and the same output density function. In order to balance this difference, we can construct an equivalent HMM for each DBN structure, by increasing the number of states and/or using mixtures for output probability densities. Hence, the effect of CI assertions imposed by the data can be analyzed by comparing the DBN model with an equivalent HMM.

3.6 Learning DBNs for speech recognition

In this section we present a data driven learning strategy to train acoustic models from the class of DBNs we described in the previous section. Our learning strategy is based on a search over this class to achieve the best representation of the underlying speech phenomena with minimum complexity. We aim to attain our goal by extracting the most effective dependencies imposed by the training data.

Referring to the definition of BNs, a graph structure and a parameterization of the associated conditional probabilities are required for a complete specification. Therefore, the learning problem in BNs formalism is two fold : 1) learning the parameters given a fixed structure and 2) learning the structure. We define the learning problem in our setting as follows :

Given a set of observations, find the "optimal" structure defined by the triple (κ, τ_p, τ_f), and the associated conditional probabilities which best explain the data. The optimality will be considered in the sense that the likelihood of observations is maximum and the structural complexity is minimum.

We first describe the parameter learning algorithm in which we use the Maximum Likelihood (ML) approach. Next we describe structure learning algorithm which is based on Structural Expectation Maximization (SEM) algorithm defined in [52].

3.6.1 Maximum likelihood parameter estimation

Given a fixed structure in our search class, we estimate the parameters relative to this structure that achieve the maximum likelihood of data. This is done using the classical expectation maximization (EM) algorithm [33, 83]. Based on the dependency assumptions, the factorization of the joint probability function is as follows.

$$P(\mathbf{X_h}_1^T, \mathbf{X_o}_1^T) = \prod_{t=1}^{T} P(X_h[t] \mid \Pi_h[t]) P(X_o[t] \mid \Pi_o[t]) \tag{3.7}$$

Using this factorization property one can derive the EM update equations for the parameters defined in Section 3.5.1 , namely the state transition probabilities and observation density means and variances. Here we summarize the update equations for a set of L observation sequences $O_l = \{\mathbf{o}_1^l, \ldots, \mathbf{o}_{T_l}^l\}$, each of length T_l. The derivations of the update equations and extension to mixture of Gaussians can be found in the Appendix D.

$$\begin{aligned} \xi(t,j,\mathbf{i},l) &= P(X_h[t]=j,\Pi_h[t]=\mathbf{i}|O_l) \\ \gamma(t,\mathbf{i},l) &= P(\Pi_o[t]=\mathbf{i}|O_l) \end{aligned} \tag{3.8}$$

$for \quad \kappa+1 \leq t \leq T_l$

$$a_{\mathbf{i}j} = \frac{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\xi(t,j,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\sum_{j=1}^{N}\xi(t,j,\mathbf{i},l)} \tag{3.9}$$

$for \quad 1 \leq t \leq \kappa$

$$a_{\mathbf{i}j}[t] = \frac{\sum_{l=1}^{L}\xi(t,j,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{j=1}^{N}\xi(t,j,\mathbf{i},l)} \tag{3.10}$$

$for \quad \tau_p+1 \leq t \leq T_l-\tau_f$

$$\begin{aligned} \mu_{\mathbf{i}} &= \frac{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\mathbf{o}_t^l\gamma(t,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\gamma(t,\mathbf{i},l)} \\ \Sigma_{\mathbf{i}} &= \frac{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\mathbf{o}_t^{l*}\mathbf{o}_t^l\gamma(t,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\gamma(t,\mathbf{i},l)} - \hat{\mu}_{\mathbf{i}}^*\hat{\mu}_{\mathbf{i}} \end{aligned} \tag{3.11}$$

$for \quad 1 \leq t \leq \tau_p \quad and \quad T_l-\tau_f+1 \leq t \leq T_l$

$$\begin{aligned} \mu_{\mathbf{i}}[t] &= \frac{\sum_{l=1}^{L}\mathbf{o}_t^l\gamma(t,\mathbf{i},l)}{\sum_{l=1}^{L}\gamma(t,\mathbf{i},l)} \\ \Sigma_{\mathbf{i}}[t] &= \frac{\sum_{l=1}^{L}\mathbf{o}_t^{l*}\mathbf{o}_t^l\gamma(t,\mathbf{i},l)}{\sum_{l=1}^{L}\gamma(t,\mathbf{i},l)} - \hat{\mu}_{\mathbf{i}}[t]^*\hat{\mu}_{\mathbf{i}}[t] \end{aligned} \tag{3.12}$$

Note that, the maximization of the auxiliary function should be handled for each time t in the initial and final networks. The computation of $\xi(t,j,\mathbf{i},l)$ and $\gamma(t,\mathbf{i},l)$ is an inference problem for DBNs, which can be handled easily using the JLO algorithm [74].

3.6.2 Learning the structure

In Section 3.5, we defined a set of plausible DBN structures for speech recognition. In this section we use this set as the structure search class for the learning algorithm. The learning problem for Bayesian networks is to find the optimal structure and parameters of the network given a set of training data. The general approach is to define a scoring metric that evaluates the optimality of a given BN and use a search algorithm that traces the space of allowed structures to maximize the scoring metric. In [52], the structure scoring rules for static BNs is extended to dynamic case. Using the representation of a DBN for a first order Markov process in terms of an initial and a transition network the authors show that the DBN learning problem can be reformulated as learning static BNs. The same reasoning can be applied for DBNs of non-Markov processes represented with several static BNs, i.e., initial, transition and final networks as described in Section 2.5.

Our goal to find the "optimal" DBN structure and the associated parameters for each acoustic unit v in the given set V. The optimality will be considered in the sense that the likelihood of

observations is maximum and the structural complexity is minimum. We use the minimum description length (MDL) metric (or equivalently Bayesian Information Criteria (BIC)) to evaluate the complexity of a given network. Thus, the scoring of a network is given by :

$$Score_{MDL} = \log P(D|\Theta,S) - \frac{\log L}{2}\sum_{i=1}^{n}||X_i,\Pi_i|| \tag{3.13}$$

where D is the observation set, L is the number of examples (realizations) in D and $||X,Y||$ is defined as the number of parameters required to encode the conditional probability, $P(X|Y)$. We use the set of structures defined in Section 3.5 as the search class for the learning algorithm

The principle of the learning algorithm is to start with some initial structure and parameters and to run a generalized version of the Expectation Maximization (EM) algorithm[49], called structural EM (SEM) which allows the update of structure and parameters iteratively. The block diagram of this algorithm is given in Fig. 2.3. At each step, first a parametric EM algorithm is performed to find the optimal parameters for the current structure. Second, a structural search is performed to find a structure that yields a higher score. The computation of the scoring metric for a candidate structure generally requires inference within the previous network. This algorithm guarantees an increase in the MDL score at each iteration, and converges to a local maximum.

Our learning algorithm can be stated as follows :

- Specify the triple $(\kappa_{max},\tau_{p_{max}},\tau_{f_{max}})$, in order to define the search class.
- Run the structural EM algorithm. However, instead of using a random structure in the first iteration, we use the HMM structure. This way, we guaranty that the learned model will have a likelihood greater (or equal) than the HMM likelihood. In other words, the learned model is guaranteed to model speech with higher fidelity than HMMs.

The triple $(\kappa_{max},\tau_{p_{max}},\tau_{f_{max}})$, controls the size of the set of allowed structures. Therefore, a trade off can be made at this level over the complexity of the learning algorithm and the fidelity of the resulting model.

3.6.3 Embedded training for Speech recognition

In continuous speech recognition the training database consists of continuous utterances that are labeled with the underlying phone sequence. In general the boundaries dividing the phone segments are either unavailable or unreliable. Therefore an embedded training strategy is necessary for reliable acoustic modeling. In HMM based systems embedded training is performed by constructing a composite model for the given training utterance based on the underlying phone sequence and reestimating the parameters of each phone model using the computed statistics (i.e. forward backward coefficients) on the composite model. The composite model is obtained by concatenating the phone models according to the transitions in the given sequence. The same methodology could be applied for a DBN based system. However the construction of the composite model requires that all phone models have the same graph structure. Therefore a preliminary step is to construct an equivalent DBN for each phone using the maximal structure. The maximal structure is defined as the structure that is able to encode all the CI relations encoded by the phone DBNs. Next the composite model is constructed using these equivalent DBNs. This procedure is explained in Section 3.7. After the model is constructed, the last step is to compute the sufficient statistics for each phone DBN using this composite model.

Embedded reestimation procedure is computationally highly demanding, both for HMMs and DBNs. In practice this approach is only used for fine tuning of some initial models. The practical approach is to start with an initial segmentation of the given utterance and perform an isolated training in order to obtain initial models. Embedded reestimation is then used for fine tuning of these initial models. Another practical approach is to use an approximation in the reestimation formulas. The exact Baum-Welch update equations requires the computation of the statistics based on all possible state sequences. However most of these paths are unrealistic segmentations of the utterance. A reasonable approximation is to compute the statistics only over the maximum likelihood state sequence.

3.7 Inference for Speech Recognition

In this section, we present the recognition algorithms in isolated and continuous speech recognition tasks using the class of DBNs we described.

3.7.1 Isolated speech recognition

Given an observation sequence the recognition problem is to find the best DBN model in the sense that the likelihood of the observations is maximum. In DBNs the likelihood of a set of observations can be computed using the JLO algorithm [74]. This algorithm is a generalized version of the forward-backward algorithm in HMMs [116]. The recognition is performed by running the JLO algorithm on each DBN model and than choosing the one that gives the maximum likelihood.

3.7.2 Continuous speech recognition

Continuous recognition is a relatively more complicated task. The decoding problem is to identify the most likely sequence of acoustic units, given a speaker utterance. However in addition to maximizing the acoustic model likelihoods one has to take into account the effect of the language model as well. For each time instant the probability of each acoustic unit should be computed and the most likely sequence should be selected to maximize the overall likelihood.

In HMMs, this is done using the Viterbi algorithm [124]. In our setting there are two basic requirements to perform decoding : 1) an algorithm to find the best state sequence in a DBN given the observation sequence, 2) a methodology to augment the DBN models (with different structures) with the language model. The first requirement is overcome by Dawid's algorithm which is a message propagation algorithm that allows the identification of the most likely sequence of hidden states in an arbitrary (discrete) Bayesian network given the observations [30]. The details of this algorithm is explained in Appendix A. For the second requirement we develop two algorithms. First we describe a brute force approach based on the construction of a state-augmented meta model and the direct use of Dawid's algorithm on this meta DBN. The main difficulty arises due to the different structures (i.e. conditional independence properties) of each acoustic unit DBN. The decoding should be performed without violating the learned CI properties of each unit. This is achieved by encoding asymmetric CI properties on the meta DBN. Next in Section 3.7.4, we propose a computationally efficient algorithm based on dynamic programming that exploits the properties of the class of DBN structures we propose.

3.7.3 Brute force approach

In the following we first describe the construction of the meta DBN assuming that all word DBNs have the same structure. This meta DBN is constructed such that it encodes all necessary

information required for decoding. Next we describe how different model structures can be integrated. This is indeed achieved by encoding asymmetric CI properties on the meta DBN. Once such a meta model is constructed, the decoding can be performed using the Dawid algorithm [30] which allows the identification (with the same time complexity as the JLO algorithm [74]) of the most likely sequence of hidden states, given observations. It is important to note that if the DBN structures reduce to HMM, then this technique is equivalent to Viterbi decoding [116].

Let us assume that we are given a vocabulary V of $|V|$ words [6] and a DBN model for each word $v \in V$, with structure $S = (\kappa^v, \tau_p^v, \tau_f^v)$ and parameters $\Theta = \{a_{\mathbf{i}_v j_v}[t], \mu_{\mathbf{i}_v}[t], \Sigma_{\mathbf{i}_v}[t]\}$. Assume also that we are given a bi-gram language model $P(v|v') \quad \forall v, v' \in V$.

State-augmented meta DBN

If all the DBN structures were the same, i.e. $(\kappa^v, \tau_p^v, \tau_f^v) = (\kappa, \tau_p, \tau_f), \ \forall\ v$, then an efficient solution to our decoding problem is the following. The state-augmented meta DBN is defined on the graph structure (κ, τ_p, τ_f) using the common CI properties of the word models. The state space of the meta DBN is enlarged to represent all internal states of word DBNs and the language model. In this meta model, each variable $X_h[t]$ takes now its values in the set $I = \bigcup_{v \in V} I_v$. This enlargement enables the internal word state transition probabilities and the word transition probabilities to be encoded simultaneously in the conditional probability distribution of the augmented state variable, $X_h[t]$. The conditional probabilities $P(X_i[t]|\Pi_{it} = \mathbf{i})$, for $i \in \{h, o\}$ for the hidden and observed variables are defined as follows.

First recall that in our class of structures the parents of a variable (hidden or observed) $\Pi_{it}, \quad i \in \{h, o\}$ consists of a subset of hidden variables from different time frames. Therefore the configuration $\Pi_{it} = \mathbf{i}$ may correspond to a different word at each time frame. When all state variables take a value corresponding to the same word v, i.e. $\mathbf{i}$ takes its values in I_v^m, then the conditional probabilities are given by the parameters of the specific word :

$$
\begin{aligned}
P(X_h[t] = j_v \mid \Pi_{ht} = \mathbf{i}_v) &= a_{\mathbf{i}_v j_v}[t], \quad \forall\ \mathbf{i}_v \in I^{\kappa^v}, j_v \in I. & (3.14)\\
P(X_o[t] \mid \Pi_{ot} = \mathbf{i}_v) &\sim \mathcal{N}(\mu_{\mathbf{i}_v}[t], \Sigma_{\mathbf{i}_v}[t]), \quad \forall\ \mathbf{i}_v \in I^{\tau_p^v + \tau_f^v + 1}. & (3.15)
\end{aligned}
$$

When state variables take values corresponding to different words in $\Pi_{it} = \mathbf{i}$ these parameters specify the allowed word transitions probabilities. Note that, we only allow transitions from the last state of a word to the first state of another, and assign zero probability for all other configurations. We use the language model for these transition probabilities of the hidden variable. For the observable variable, the emission probabilities in the word transitions are specified using the word model that is indicated by the configuration of $X_h[t]$. When $X_h[t] = N_v$ and $X_h[t+1] = 1_{v'}$, we assume that the emission is from word v such that the state sequence ends at the last state. Similarly, when $X_h[t-1] = N_{v'}$ and $X_h[t] = 1_v$, we assume that the emission is from word v such that the state sequence starts at the first state of word v. Hence, the conditional probabilities for the transitions between two (not necessarily different) words v and v' are given as :

$$
P(X_h[t] = 1_v \mid \Pi_{ht} = \mathbf{i}_{v'}) = P(v|v') \quad \forall\ \mathbf{i}_{v'} \in I^{\kappa^v} \text{ s.t. } \Pi_{ht} \ni X_h[t-1] = N_{v'}. \qquad (3.16)
$$

$$
P(X_o[t] \mid \mathbf{i}) = \begin{cases} P_v(X_o[t] \mid (1_v \ldots 1_v, i_{t+1} \ldots i_{t+\tau_f})) & \text{for } i_t = 1_v \text{ and } (i_{t+1} \ldots i_{t+\tau_f}) \in I_v^{\tau_f} \\ P_v(X_o[t] \mid (i_{t-\tau_p} \ldots i_{t-1}, N_v \ldots N_v)) & \text{for } i_t = N_v \text{ and } (i_{t-\tau_p} \ldots i_{t-1}) \in I_v^{\tau_p} \\ 0 \text{ otherwise} \end{cases} \qquad (3.17)
$$

[6] For sake of clarity we assume that the basic acoustic unit used in recognition is word. The formulation is indeed valid for any other acoustic unit.

where $P(v|v')$ defines the language model. Notice that, for a left to right topology, all probabilities are set to zero for $i_t \neq i_{t-1}, i_{t-1} - 1$.

Encoding asymmetric CI properties

In general, after performing a structural learning, the resulting DBN structures would not be the same for all the words. In this case, we cannot directly use the technique described above, because we do not have, at hand, a unique structure to represent all the words. Indeed, in the augmented model when $X_h[t]$ takes a value j_v corresponding to a word v, the independence assertions should be identical to those imposed by the structure $(\kappa^v, \tau_p^v, \tau_f^v)$. In other words, we have to encode *asymmetric* independence assertions in the sense that variables are independent for some but not necessarily for all of their values. We present, in the following, a technique to do so which relies on encoding the asymmetric assertions in the numerical parameterization. The basic idea is to build a "maximal" DBN model and use a special parameterization to encode the asymmetry.

We find the "maximal" DBN structure $(\kappa^m, \tau_p^m, \tau_f^m)$ that is capable to encode all the dependencies represented in the learned word models as follows :

$$\kappa^m = \max_v(\kappa^v), \ \tau_p^m = \max_v(\tau_p^v), \ \tau_f^m = \max_v(\tau_f^v) \ .$$

Then, for each word, we construct an equivalent DBN with this maximal structure that encodes the same JPD as the original word DBN. Although the Conditional Independence (CI) assertions of the maximal structure is different from the word DBN, we exploit the Markov properties of the word DBN in specifying the conditional probabilities of the equivalent maximal DBN. This comes back to setting several of them to be equal to those given by the original model. Precisely,

$$P_{mv}(X_i[t]|\Pi_{it}^{mv} = (\mathbf{i}_v, \mathbf{j})) = P_v(X_i[t]|\Pi_{it}^v = \mathbf{i}_v) \quad \text{for all} \ \ \mathbf{i}_v \in \Pi_{it}^v, \mathbf{j} \in \Pi_{it}^{mv} \setminus \Pi_{it}^v \tag{3.18}$$

where Π_{it}^m is the set of parents of $X_i[t]$ in the maximal DBN, and Π_{it} is the set of parents of $X_i[t]$ in the learned DBN.

Now that we have a set of DBNs with the same structure, we can construct a meta DBN as described above. Note that, since we do not violate the different CI assertions of individual word models in this procedure, the CI assertions encoded in the resulting meta DBN are asymmetric.

Obviously, the extra parameterization in the maximal model causes unnecessary computation while performing inference. Therefore in the following we provide a more efficient algorithm that exploits the CI properties of the class of structures used.

3.7.4 A fast decoding algorithm

As in the brute force approach, the first step is to construct a "maximal" DBN for each acoustic unit with structure $(\kappa^m, \tau_p^m, \tau_f^m)$. The goal is to represent all acoustic models with the same graphical structure without violating the learned Conditional Independence (CI) relations.

Once the maximal network of each acoustic unit is constructed, the second step (which is the main contribution of this section) consists, basically speaking, in operating a dynamical programming procedure to perform a "parallel" inference algorithm on "local clique potentials" (see [78] for details).

The principle of the algorithm is similar to Viterbi decoding algorithm that is widely used in HMM based speech recognition systems [124]. The main difference is that Viterbi algorithm

is used for first order hidden Markov models whereas the proposed algorithm is derived for the set of DBN structures we propose in [36]. Indeed, if the maximal structure is $(1, 0, 0)$, i.e. HMM structure, our algorithm is identical to Viterbi decoding algorithm. We proceed now to describe this algorithm.

In the experiments we carry out later, the maximal network structure we obtain is $(\kappa^m, \tau_p^m, \tau_f^m) = (2, 1, 1)$. Given the technicality of the algorithm and for the sake of clarity and simplicity of the paper, we describe our decoding algorithm for this particular case. We emphasize however that this algorithm can be readily generalized to any kind of maximal network.

Let us denote the hidden and observed variables $X_h[t], X_o[t]$ as H_t and O_t respectively. Let H_t take its values in $\{1_v, \ldots, N_v, \ \forall v \in V\}$ to denote the hidden states for all acoustic unit models. As described in Chapter 2.5, the conditional probabilities of a DBN are defined according to the initial, transition and final networks. For the maximal structure (2,1,1) there are 2 initial, 1 transition and 1 final networks. The conditional probabilities of these networks are denoted as follows (where $i, j, k \in \{1_v, \ldots, N_v, \ \forall v \in V\}$).

Initial networks :

$$\begin{aligned} a_j^v &= P(H_1 = j) \ , \\ a_{jk}^v &= P(H_2 = k|H_1 = j) \ , \\ b_{jk}^{Iv}(O_1) &= P(O_1|H_1 = j, H_2 = k) \ . \end{aligned}$$

Transition network :

$$\begin{aligned} a_{ijk}^v &= P(H_{t+1} = k|H_t = j, H_{t-1} = i) \ , \\ b_{ijk}^v(O_t) &= P(O_t|H_{t-1} = i, H_t = j, H_{t+1} = k) \ . \end{aligned}$$

Final network :

$$b_{ij}^{Fv}(O_t) = P(O_t|H_{t-1} = i, H_t = j) \ .$$

Our aim is to find the most likely sequence of hidden states $H_1^T = \{H_1, \ldots, H_T\}$ given the observation sequence $O_1^T = \{O_1, \ldots, O_T\}$, and consequently the likelihood of observations along this sequence :

$$P_{max}(O_1^T) = \max_{H_1 \ldots H_T} P(O_1^T, H_1^T). \tag{3.19}$$

First note that the state transitions are restricted to a left-to-right topology and a model transition from model r to v is only allowed from the last state of r (N_r) to the first state of v (1_v). In order to provide a recursive formulation for (3.19) we define two intermediate quantities :

$$\begin{aligned} \delta_t^v(i,j,k) &= \max_{H_1 \ldots H_{t-2}} P(O_1^t, H_1^{t-2}, H_{t-1} = i, H_t = j, H_{t+1} = k) \ , \\ \gamma_t^v(i,j) &= \max_{H_1 \ldots H_{t-2}} P(O_1^t, H_1^{t-2}, H_{t-1} = i, H_t = j) \ . \end{aligned}$$

$\delta_t^v(i,j,k)$ is the likelihood of the first t observations along the most likely state sequence up to H_{t-2} and for $H_{t-1} = i, H_t = j, H_{t+1} = k$. Similarly $\gamma_t^v(i,j)$ is the likelihood of the first t observations along the most likely state sequence up to H_{t-2} and for $H_{t-1} = i, H_t = j$ assuming that O_t is the last observation from model v. We need this second quantity to compute the likelihood when a model transition occurs at $t+1$.

For each model v we initialize the recursion with $\delta_1^v(i,j,k) = a_j^v a_{jk}^v b_{jk}^{Iv}(O_1)$ which is the likelihood of O_1 emitted from the initial network of v. Then for each t, the new evidence O_t can be emitted from either of the initial, transition and final networks of v. We know that $\delta_{t-1}^v(n,i,j)$ is the maximum likelihood for the observation sequence O_1^{t-1} for $H_{t-2}=n, H_{t-1}=i, H_t=j$.

If the emission is from the final network then O_t does not depend on H_{t+1} and the new likelihood is computed by maximizing $\delta_{t-1}^v(n,i,j)$ over n and incorporating the emission probability from the final network. We defined this term as $\gamma_t^v(i,j)$:

$$\gamma_t^v(i,j) = b_{ij}^{Fv}(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \; . \tag{3.20}$$

This term is the maximum likelihood of the observation sequence O_1^t for $H_{t-1}=i, H_t=j$ for $i,j \in \{1_v,\ldots,N_v\}$. Considering the fact that for the full observation sequence O_1^T the last observation is emitted from the final network of some $v \in V$, the maximum likelihood for the full observation sequence is obtained from the following maximization over v,i,j :

$$P_{max}(O_1^T) = \max_{i,j,v} [\gamma_T^v(i,j)] \; . \tag{3.21}$$

To complete our formulation we continue with the derivation of the recursion formula for $\delta_t^v(i,j,k)$. If the emission is from initial or transition networks of v the incorporation of the new evidence O_t depends on the value of H_{t-2} such that :

- if $H_{t-2} \in \{1_v,\ldots,N_v\}$ then O_{t-1} was emitted from v and O_t will be emitted from the transition network of v. The new likelihood is computed by maximizing $\delta_{t-1}^v(n,i,j)$ over n and incorporating the emission probability from the transition network of v.

$$\delta_t^v(i,j,k) = a_{ijk}^v b_{ijk}^v(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \tag{3.22}$$

- if $H_{t-2} \in \{N_r, \forall r \in V\}$ then O_{t-1} was emitted from the final network of r and a model transition occurs at t. Hence O_t will be emitted from the initial network of v with $H_t = 1_v$. In this case we need a maximization over n and r to specify the model that transits with the maximum likelihood. Therefore the likelihood term is computed as :

$$\delta_t^v(i,1_v,k) = a_{1_v k}^v b_{1_v k}^{Iv}(O_t) \max_{n,r} [\gamma_{t-1}^r(n,N_r) P(v|r)] \; . \tag{3.23}$$

$P(v|r)$ is the transition probability from model r to v and it is given by the language model obtained from frequency counts of acoustic unit pair occurrences in the training corpus.

Now in order to specify the value[7] of H_{t-2} that maximizes the overall likelihood for O_1^t we maximize among these two cases :

$$\begin{aligned} \delta_t^v(i,j,k) = \max \{ & a_{ijk}^v b_{ijk}^v(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \; , \\ & a_{jk}^v b_{jk}^{Iv}(O_t) \max_{n,r} [\gamma_{t-1}^r(n,N_r) P(v|r)] \} \; . \end{aligned} \tag{3.24}$$

The case that yields the maximum likelihood specifies the value of H_{t-2} in the state sequence.

The complete algorithm is given as follows where we introduce $\psi_t(i,j)$ for back tracking the maximization arguments. Once the maximum likelihood is computed, the most likely state sequence is obtained by back tracking the maximization arguments. The outline of the algorithm is given in Algorithm 3.7.1.

[7]Notice that, we allowed H_{t-2} to take values only in $\{1_v,\ldots,N_v,N_r,\forall r \in V\}$. For all other cases no state transitions are allowed and the likelihood is zero.

Algorithm 3.7.1 Fast Decoding

Initialization :

$$\begin{aligned} \delta_1^v(j,k) &= a_j^v a_{jk}^v b_{jk}^{Iv}(O_1) \; , \\ \gamma_1^v(j) &= 0 \; , \\ \psi_1(j) &= 0 \; . \end{aligned}$$

Recursion for $1 < t < T$ *:*
For $2_v \leq j \leq N_v$,

$$\begin{aligned} \delta_t^v(i,j,k) &= a_{ijk}^v b_{ijk}^v(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \; , \\ \psi_t(i,j) &= \arg\max_{n,v} [\delta_{t-1}^v(n,i,j)] \; . \end{aligned}$$

For $j = 1_v$,

$$\begin{aligned} \delta_t^v(i,j,k) = \max \{ & a_{ijk}^v b_{ijk}^v(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \; , \\ & a_{jk}^v b_{jk}^{Iv}(O_t) \max_{n,r} [\gamma_{t-1}^r(n,N_r) P(v|r)] \} \; , \end{aligned}$$

$$\psi_t(i,j) = \begin{cases} \arg\max_{n,v} [\delta_{t-1}^v(n,i,j)] \; . \\ \arg\max_{n,v} [\gamma_{t-1}^r(n,N_r) P(v|r)] \; . \end{cases}$$

For $j = N_v$,

$$\gamma_t^v(i,j) = b_{ij}^{Fv}(O_t) \max_n [\delta_{t-1}^v(n,i,j)] \; .$$

Termination :

$$\begin{aligned} \gamma_T^v(i,j) &= b_{ij}^{Fv}(O_T) \max_n [\delta_{T-1}^v(n,i,j)] \; , \\ \psi_T(i,j) &= \arg\max_{n,v} [\delta_{T-1}^v(n,i,j)] \; , \\ P_{max}(O_1^T) &= \max_{i,j,v} [\gamma_T^v(i,j)] \; , \\ (H_{T-1}, H_T) &= \arg\max_{i,j} [\delta_T^v(i,j)] \; . \end{aligned}$$

Back Tracking :

$$H_t = \psi_{t+2}(H_{t+1}, H_{t+2}), \; t = T-2, \ldots, 1 \; .$$

Complexity Analysis

Before describing the computational complexity of the proposed algorithm, we shall refer to our initial decoding algorithm. As described before, the decoding is performed using Dawid's algorithm on a state augmented model [39]. The complexity of Dawid's algorithm depends on the size of each clique on the junction tree and the number of values of each variable in the clique [30]. In our setting, the cliques[8] of the maximal network with structure $(\kappa^m, \tau_p^m, \tau_f^m) = (2,1,1)$

[8]A constrained triangulation scheme results in a repeating clique tree for the transition network. The cliques for the initial and final networks deviate from this repeating form. These are not considered for asymptotic complexity

contains three consecutive hidden state variables (H_{t-1}, H_t, H_{t+1}). If we assume that $N_v = N \; \forall v \in V$, then each augmented variable takes $N|V|$ values. The number of distinct realizations for each clique is therefore $N^3|V|^3$. The computational complexity for an observation sequence of length T is $O(N^3|V|^3T)$.

For the proposed fast decoding algorithm, there are three major items in the recursive computation. These are computed for all $t = 2, \dots, T-1$ and $v \in V$. We analyze each of them separately for each t, v and then derive the asymptotic complexity accordingly.

- $a^v_{ijk} b^v_{ijk}(O_t) \max_n[\delta^v_{t-1}(n,i,j)]$
  This term is computed for $1 \leq j \leq N$. The complexity depends on the number of distinct values of the (i,j,k) triple. This would be N^3 without any topological constraints yielding $2N^3$ multiplications and N^3 operations in maximization. Exploiting the left-to-right topology for the state sequence, i.e. if $H_{t-1} = i$, then $H_t = j \in \{i, i+1\}$, the number of distinct values for (i,j,k) reduces to $4N+4$. The total number of operations is therefore (4N+4 + 2(4N+4)). Hence the complexity introduced by this term is $O(N)$ rather than $O(N^3)$.
- $a^v_{jk} b^{Iv}_{jk}(o_t) \max_{n,r}[\gamma^r_{t-1}(n,N)P(v|r)]$
  This term is computed only for $j = 1$. The computation requires $N+1$ multiplications for each k and $N|V|$ multiplications for each n, r. The maximization also introduces $N|V|$ operations. The asymptotic complexity is $O(N|V|)$. The topological constraint does not reduce the complexity because it only affects the range of values for k to reduce the $N+1$ multiplications to 3.
- $b^{Fv}_{ij}(O_t) \max_n[\delta^v_{t-1}(n,i,j)]$
  This term is computed only for $j = N$ with N multiplications and N^2 operations in maximization. The left-to-right topology limits the values of i to $\{N, N-1\}$. So the complexity reduces to $O(N)$ instead of $O(N^2)$.

Now considering the computation for all t and v, the overall asymptotic complexity "without" the topological constraints is $O(N^3|V|T + N|V|^2T + N^2|V|T)$. The improvement is significant as compared to $O(N^3|V|^3T)$. This improvement is due to 1) the parallel nature of the proposed algorithm and 2) exploitation of the structural properties of the DBNs we use. Further improvement is achieved by imposing the a left-to-right transition topology for the hidden states. In this case the dominant term in the asymptotic complexity is $O(N|V|^2T)$.

3.8 Experiments

3.8.1 Isolated digit recognition

In isolated word recognition, we model each word in the database with a DBN and train the models from a given set of utterances. The structure and parameter learning of the model is performed using the techniques described in the previous sections. Once the models are trained for each word, in the test phase, we compute the likelihood of each test utterance on each model. The recognition is based on maximum likelihood principle. The model that gives the maximum likelihood is chosen as the recognized word.

The experiments are carried out on the isolated part of the Tidigits database in which 112 (resp. 113) speakers are used for training (resp. test). Each speaker uttered 11 digits twice. The size of each observation vector is 35, consisting of 11 MFCC (energy dropped), 12 Δ, and 12 $\Delta\Delta$. Each hidden variable takes 4 values, $M = 4$. Each observable variable has a single Gaussian

analysis.

distribution with a diagonal covariance matrix. We assume a left-to-right transition topology for all the structures considered.

For the purpose of this experiment the upper-bound on the dependence parameters are chosen to be $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}}) = (2,1,1)$. As a first step we evaluate all possible structures for all digits using the BIC score. This experiment gives us the global maximum among the structure space for each digit. We observe that for all the digits, the structure with the highest score is the same[9] with the dependency parameters as, $(\kappa, \tau_p, \tau_f) = (1,0,1)$. This shows that future dependencies can be very important in modelling. The recognition rate obtained using this structure is 96.43%, while the rate given by the HMM structure is 92.86%. The improvement is remarkable given that the inference complexity of both structures is asymptotically the same. Indeed, due to the left-to-right topology assumption, the increase in the decoding complexity is only by a factor of 2 with respect to HMM decoding.

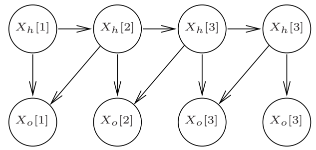


FIG. 3.11 – The global maxima in the structure space of all the digits $(T = 4)$, $(\kappa, \tau_p, \tau_f) = (1,0,1)$.

Next, we apply the structural learning algorithm using BIC scoring for each digit. The learned structure for 'o', '0', '2', '4', '5', '7', '9', is $(\kappa, \tau_p, \tau_f) = (1,0,1)$, and for '1', '3', '6', '8' it is $(\kappa, \tau_p, \tau_f) = (1,1,0)$. We observe a recognition rate of 96.63% with the learned models. For most of the digits, the SEM algorithm converges to the global maxima of the structure space. For the rest of the models, although the algorithm is struck in the local maxima the score is very close to the global maxima. Moreover, recognition results indicate that the modelling performance is acceptable.

It is also interesting to compare the recognition rates of all the structures within the search space. We performed separate parameter learning algorithms for each structure. In Table 3.1 we give a list of the recognition rates for each structure. The structures are listed in increasing structural complexity, where (1,0,1)-(1,1,0) and (2,0,1)-(2,1,0) have the same complexities respectively. One can see that, increasing model complexity generally increases the recognition performance. It should also be noted that none of the structures has a lower performance than the baseline HMM. MDL scoring leads to a good trade off between modeling fidelity and complexity. However we should observe that, although the structures (1,0,1) and (1,1,0) have the same complexity, and that (1,0,1) has a higher likelihood for the training set, the recognition performance is higher for (1,1,0). We relate this inconsistency to the imperfections in the data set.

[9]We should note that the convergence to the same structure for all data sets is a special case. In general, it is possible to have a different structure for each data set.

κ	τ_p	τ_f	Recognition Rate
1	0	0	92.86
2	0	0	92.86
1	**0**	**1**	**96.43**
1	1	0	97.44
2	0	1	96.51
2	1	0	97.56
1	1	1	97.05
2	1	1	98.42

TAB. 3.1 – Digit recognition rates for different DBN model structures (%).

3.8.2 Connected digit recognition

In this section, we compare[10] the performances of DBN models to standard HMMs in a connected digit recognition task. Our experiments are carried out on the Tidigits database. In learning, we only use the isolated part of the training database where each speaker utters 11 digits ('o',0, 1, . . . 9) twice. As an initial step, we segment the silence regions in the training corpus. We perform isolated training for all HMM and DBN models based on this initial segmentation. In all experiments, the silence is modelled with a single state, single Gaussian HMM. In tests, we use the full (test) database in which 8636 sentences are uttered, each sentence contains between 1 and 7 digits. The size of each observation vector is 35, consisting of 11 MFCCs (energy dropped), 12 Δ, and 12 $\Delta\Delta$. The covariance matrix is assumed diagonal. We use a left-to-right transition topology and a uniform language model, i.e., $P(v|v') = \frac{1}{12}$ ($|V| = 12$). In the structural learning algorithm, for all digits, the upper-bound on the search class is $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}}) = (2, 1, 1)$.

(κ, τ_p, τ_f)	N=4	N=5	N=6
(1,0,0)			'o',0,2,7
(1,0,1)	'o',2,4,7,8,9	'o',1,4,5,9	3,6,8
(1,1,0)	0,1,3,5,6	0,2,3,6,7,8	1,4,5,9

TAB. 3.2 – Learned digit DBN structures using SEM algorithm.

We show experiments using different numbers of states, i.e. different values of N where we set $N_v = N \;\forall\; v$. In Table 3.2, we show the results of the structural learning algorithm. When $N \geq 7$, our system yields a standard HMM as the best model for all digits. This shows that when an HMM is enough to model data, our system recognize it. The recognition scores are given in Table 3.3, for $N = 4, 5, 6$, the maximal model structure is $(\kappa^m, \tau_p^m, \tau_f^m) = (1, 1, 1)$. Our system largely outperforms the HMM system, particularly when $N = 4$. This remarkable increase in the performance can be explained as follows. Since the number of hidden states is small, the HMM system introduces a lot of insertions. Using our system, the learned models all involve context dependencies. It is well known that modelling context, improves duration modelling. Therefore, even though the number of states is still small, our system provides a better digit duration modelling, and thus the total number of insertions is drastically reduced. However, to

[10] Very good scores can be obtained on this database using Gaussian-mixture HMMs and adjusted parameters. Our goal here is not to tune the parameters in order to achieve the highest performances. Rather, we want to provide fair comparisons using baseline systems. We believe that this way we have a fair initial judgment on the capacities of each system.

make fair comparisons, we should compare our system to an HMM, with an equivalent number of parameters. To do so, we compare our system to an HMM with the same number of states, but using a mixture of 2 Gaussians as the observation probability density. The average number of parameters C_{av} of each system is computed by averaging the quantity C defined by Eq. (7) over all digit models [11]. For $N = 4, 5$, our system ($C_{av} = 497,\ 639$) performs still better than HMM$_{2G}$ (which actually uses even more parameters than our system, $C_{av} = 567,\ 709$). For $N = 6$, even though our system uses much less parameters ($C_{av} = 653.6$) than HMM$_{2G}$ ($C_{av} = 1551$), both systems yields the same scores approximately.

N	HMM(N)	DBN(N)	HMM$_{2G}$(N)
4	20.57	63.22	46.32
5	60.92	79.86	77.99
6	77.93	84.47	84.44

TAB. 3.3 – Digit recognition accuracies using DBN and HMM models (%).

As a final remark on these experiments, note that our system allows learning of non-Markovian processes (the DBN (1,0,1) is non-Markov, for instance). This is a major advantage with respect to HMMs. Indeed, by doing so, our system is able to take into account the well known *anticipation* phenomenon which occurs in the speech production mechanism. These experiments show the power of the DBN approach to model both acoustical and phonemical aspects of speech with higher fidelity than HMMs.

3.8.3 Continuous phoneme recognition

In this section, we evaluate our approach on a more complicated task, i.e, continuous phoneme recognition. Our experiments are carried out on the TIMIT database that has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems [84]. The training corpus consists of 3695 sentences. The tests are performed on 192 sentences in the core test set. We train 48 phoneme models and a silence model. Each phoneme and the silence are modeled with 3-states for HMMs and DBNs, i.e $H_t = \{1, 2, 3\}$. The output probability density of each observed variable O_t is represented with a mixture of Gaussians with diagonal covariance matrices. In tests, we use a phoneme based bi-gram language model learned from the training corpus. In accuracy computations we neglect the confusions within 7 phoneme groups to yield 39 effective phonemes as in [84]. The acoustic parameterization is based on standard Mel frequency cepstral coefficients (MFCC). These are computed from 26 Mel scaled logarithmic filter-bank energies of 25ms frames. The frame rate is 10ms. The resulting parameter vector consists of 12 static MFCC coefficients and energy concatenated with first and second order time derivatives. Front-end parameterization and HMM modeling are performed using HTK toolkit.

As a first step we construct an HMM based system. This system is used as a reference to compare DBN based systems. We initialize the models using the phonetic transcriptions provided in the database [84]. Given the segment boundaries of each phoneme, we consider isolated observations for each model. We use a uniform segmentation for the state sequence and obtain initial estimates for the means and variances of the Gaussians. Next, we apply the Viterbi alignment procedure to compute the maximum likelihood state sequence for each phoneme segment and reestimate the models' parameters iteratively. These initial models are then refined by increasing

[11] For 1 Gaussian $M = 70$, for 2 Gaussians $M = 140$.

nb mixtures	1	2	4	8	16	32	64
HMM	-	57.48	62.25	65.32	65.77	67.82	69.00
DBN-1	59.83	61.79	65.50	67.50	69.23	70.22	-
DBN-2	59.83	62.38	65.84	67.97	69.64	70.82	-

Tab. 3.4 – Phoneme recognition accuracy of HMM and DBN based systems using different number of Gaussian mixture components (%).

(κ, τ_p, τ_f)	phonemes
(1,0,1)	b, uh, th, epi, zh
(1,1,0)	g, en, oy, d
(1,1,1)	dh, ah, ng, el, ow, w, ch, dx, q, aa, y, k,
	f, m, t, ey, ay, ae, eh, v, hh, jh, aw
(2,0,1)	ax
(2,1,1)	vcl, z, p, l, ix, s, cl, ih, n, sh, ao,
	r, iy, er, uw, s#

Tab. 3.5 – Learned phone DBN structures using SEM algorithm.

the number of Gaussian mixtures and embedded parameter reestimation iterations. The phoneme recognition accuracies obtained by this system with different number of Gaussian mixtures are given in the first row of Table 3.4.

In the construction of DBN based systems we use an initialization step that relies on the HMM system defined above. Rather than using the phonetic segmentation of the database, we start with a more reliable segmentation obtained by a forced alignment procedure. We use HMMs with 8 mixtures of Gaussians for this task. The recognition accuracy obtained with these models on the core test set is **65.32** %. Using this segmentation we initialize single Gaussian DBN models by running the SEM algorithm. The upper bound on the structure search space is set to be $(2, 1, 1)$. The learned structure for each phoneme is given in Table 3.5. It is interesting to observe that for most of the phonemes the learned structure turns out to be non-Markov, i.e., with future dependencies. This result supports our specification of the class of allowed structures for acoustic modeling. Using this class of structures we are able to capture the non-Markov phenomena in speech.

We use two different schemes to further refine single Gaussian DBN models. In the first scheme (DBN-1), we fix the model structures obtained in the first run of the SEM algorithm and increase the number of mixtures of each model. This technique allows a finer tuning of model parameters with fixed dependency assumptions. In the second scheme (DBN-2), we reestimate the structures each time the number of mixtures is increased. In this case the dependency assumptions are updated as the model accuracy is increased which allows a better exploration of dependency relations in data.

Recognition is performed using the decoding algorithm presented in Sect. 3.7.4. The maximal structure for the set of learned structures is $(2, 1, 1)$. Recognition accuracies for different number

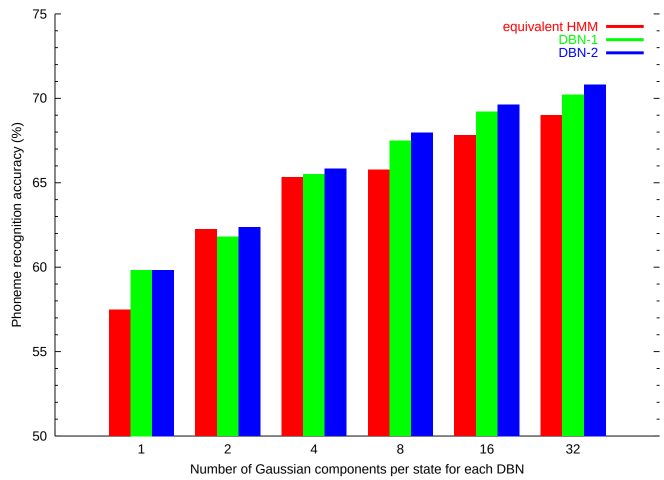


Fig. 3.12 – Recognition accuracy comparisons of DBN based systems with equivalent HMM based systems

of Gaussian mixtures are given in the second and third rows of Table 3.4 for DBN-1 and DBN-2, respectively. Our first observation from these tests is that DBN-2 systems perform better than DBN-1 systems which implies that updating model structure as the model is refined yields improved modeling.

From Table 3.4, it is clear that both DBN based systems perform better than HMM based systems with same number of Gaussian mixtures. However to make fair comparisons, we should compare a DBN system to an HMM, with an equivalent number of parameters. To do so, we compare our system to an HMM with the same number of states, but using an observation probability density with two times more mixture of Gaussians. These comparisons are provided as a bars-plot in Fig. 3.12. On the x-axis we indicate the number of mixture components for the DBN based systems. The equivalent HMM system at the same abscissa has two times more mixture components. At each abscissa we compare equivalent HMM, DBN-1 and DBN-2 systems presented in the same order from left to right. In all cases the DBN system yields a higher recognition accuracy than equivalent HMM.

These results show that substantial gain in recognition accuracy can be obtained when model dependencies are learned from data. This supports our previous results on isolated and connected digit recognition experiments [36, 39]. In addition, the computational complexity of decoding is greatly reduced using the proposed algorithm which enables the application of the methodology for complex speech recognition tasks.

Chapitre 4

Language Modeling with DBNs

Sommaire

4.1 Introduction **91**
4.2 Contributions **91**
4.3 Statistical Language Modeling **92**
4.4 n-gram models **93**
4.5 class-based n-gram models **94**
4.6 Language Modeling with DBNs **95**
4.6.1 Model Specifications 96
4.7 Learning **97**
4.7.1 Smoothing and Back-off 97
4.8 Language model evaluation **98**
4.8.1 Entropy of natural language 99
4.8.2 Perplexity as a measure for language model evaluation 99
4.9 Experiments **100**
4.9.1 Le Monde 100
4.9.2 MIAMM 101
4.9.3 SPORT 102

4.1 Introduction

The statistical speech recognition systems process basically two levels of information in speech : acoustic and linguistic. In previous chapters we deal with acoustic aspects, namely acoustic modeling, speech signal parameterization and noise robustness. However without linguistic information the performance of an ASR system is insufficient to be applied in a real world application. In this chapter we focus on the linguistic component of a statistical speech recognition system, namely language modeling.

4.2 Contributions

- A new perspective for n-gram and n-class language models in Bayesian networks formalism.
- A new set of language models defined in dynamic Bayesian networks framework.
- Evaluation of the proposed set of models on different corpora.

4.3 Statistical Language Modeling

A statistical speech recognition system estimates the most probable sequence of linguistic units (i.e. words, syllables, phonemes, etc.) given acoustic observations. The general probabilistic formulation is given as :

$$\hat{W} = \arg\max_{W} P(W|O)$$

where O denotes acoustic observations and W denotes the underlying sequence of linguistic units. The Bayesian reformulation of the problem yields a factorization over the acoustic and linguistic components :

$$\hat{W} = \arg\max_{W} P(O|W)P(W).$$

In this formulation, the acoustic model, $P(O|W)$, encodes the statistical distribution of speech acoustics given the linguistic labeling. P(W) is the probability assigned by the language model which encodes the a priori linguistic information. The main role of a language model in a speech recognition system is to propose at each stage of the decoding process the best and the most corresponding words. In fact, without a language model the system could propose any word which match acoustically the current frames. A good language model is the one which proposes at each step only the likelihood words. To achieve this objective, language models use not only simple words but also care about the construction of phrases. The phrases allow to capture syntactic information by introducing the left context which is called history. Statistical language models attempt to attain this goal using the empirical probability distribution of linguistic events.

One can identify two main steps in the definition of a statistical language model. The first one is the qualitative specification of the model, i.e. the type of linguistic events to be modeled. The second is the quantitative specification, i.e. the parametric form of the probability distributions.

The qualitative specification of a language model involves the definition of implicit and/or explicit variables of language. These variables represent the factors that drive the language. For example classical n-gram models use *word* as the only variable in language whereas syntactic n-class models use both *word* and *syntactic classes*. The dynamic evolution of language is controlled by these variables and their interactions through time. Each variable interacts with a certain number of factors that constitute its context. In general terms, context is defined as *the discourse that surrounds a language unit and helps to determine its interpretation*. In practice, in speech recognition, the surrounded context is reduced to its left context. The specification of the most informative context for each language unit is therefore a key issue on the accuracy of a language model. In probabilistic terms the context of a linguistic unit is defined with conditional independence properties. Given its context each linguistic unit is assumed to be independent of other linguistic events. For example classical n-gram models define the context of a word as the most recent $n-1$ words and make the assumption that a word is independent of all preceding words given the most recent $n-1$.

On the other hand, conditional independence is the core property of probabilistic graphical models, it is indeed the exploitation of this property that leads to efficient and generic inference algorithms of graphical models. Moreover, as it will become clear in the following sections, n-gram and n-class models (and other language modes) are very particular instances of *dynamic Bayesian networks* (DBNs) which are *oriented* probabilistic graphical models. Thus, it is a natural idea to rethink language models within the general framework of DBNs and seek potential benefits from this rethinking.

It is our purpose in this chapter to use the DBNs framework in order to achieve a better exploitation of each linguistic unit considered in modeling. We develop a unifying approach that processes each of these units in a unique model and construct new data-driven language models with improved performances. The principle of our approach is to construct DBNs in which a variable (word, class or any other linguistic unit) may depend on a set of context variables. These dependences between linguistic units can be determined automatically or manually. Of course our ultimate goal is to propose an automatic scheme to learn the optimal DBN structure from a training corpus. However, in order to investigate the feasibility of our approach, we start by analyzing DBN models for which the graphical structure is specified manually.

In the following we define language models putting an emphasis on their conditional independence properties. We use *word* (W_t) as the basic unit for the definition of a language model, keeping in mind that the formulation is valid for any other linguistic unit.

4.4 n-gram models

Considering the causality of natural language and using the chain rule of probability a factorization of $P(W)$ for a word sequence $W = \{W_1, W_2, \ldots, W_T\}$ can be obtained as :

$$P(W) = P(W_1)P(W_2|W_1)P(W_3|W_2, W_1)\ldots P(W_T|W_{T-1}, \ldots, W1) \tag{4.1}$$

$$= \prod_{t=1}^{T} P(W_t|W_{t-1}, \ldots, W_1) \tag{4.2}$$

In this representation the context of a word is defined as the set of all its predecessors. However due to computational limitations and data sparseness problems the incorporation of such a large context is generally not feasible. Therefore state-of-the-art language models use limited context definitions to approximate this exact formulation. These approximations are based on conditional independence hypothesis between the variables constituting the model.

Classical n-gram models make the assumption that distant history does not contain relevant information to estimate a word probability. It is assumed that contextual information is localized in the most recent n words, where n is generally chosen as 2, 3 and rarely 4. The probability of observing a word W_t at time t is conditioned on recent history $H_t = \{W_{t-1}, \ldots, W_{t-n+1}\}$ of only $n-1$ words. The assumed conditional independences of n-gram models is stated as follows :

Assumption 1 *Given the last $n-1$ words $\{W_{t-1}, \ldots, W_{t-n+1}\}$, a word W_t is independent of all preceding words $\{W_1, \ldots, W_{t-n}\}$.*

$$W_t \perp \{W_1, \ldots, W_{t-n}\} \mid \{W_{t-1}, \ldots, W_{t-n+1}\} \tag{4.3}$$

This assumption approximates the marginal probability of a given word sequence W as :

$$P(W) = \prod_{t=1}^{T} P(W_t|W_{t-1}, \ldots, W_{t-n+1}) \tag{4.4}$$

Schematically this model can be visualized as in Figure 4.1.

These models have shown reliable context modeling and have found wide use in state-of-the-art ASR systems. However even for small values of n, the estimation of the conditional probabilities is biased due to the high dimensionality of the problem. Therefore these models fail to generalize when the training data is insufficient. Several computational techniques have been developed to overcome this problem, i.e. back-off with discounting, interpolation with simpler models etc.

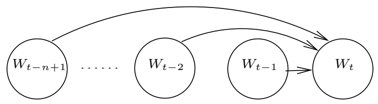


FIG. 4.1 – n-gram

4.5 class-based n-gram models

Class based n-gram (or n-class) models incorporates a higher level of information into classical word based n-grams. Each word is considered to belong to some word class that is specified using either statistical, syntactic and/or semantic constraints [72, 9]. These classes define a much smaller state space for the language. The principle of this approach is to model the dynamic evolution of language in this smaller state space and reduce the ambiguity of word observation probability using class context rather than word context. The word class is a hidden factor that represents the context of each word. The context for word is defined as its class and the context for class is defined as the classes of preceding $n-1$ words.

The model is defined on two variables : *word* (W_t) and *class* (C_t). It approximates the joint distribution $P(W, C)$ of word and class sequences $W = \{W_1, W_2, \ldots, W_T\}$ and $C = \{C_1, C_2, \ldots, C_T\}$. Since word class is a hidden factor, the word sequence probability is computed with a marginalization operation as :

$$P(W) = \sum_{C_1,\ldots,C_T} P(W, C) \tag{4.5}$$

The conditional independence assumptions of this model can be stated as follows :

Assumption 1 *Given its class C_t, a word W_t is independent of all previous words and their classes $\{W_1, \ldots, W_{t-1}, C_1, \ldots, C_{t-1}\}$.*

$$W_t \perp \{W_1, \ldots, W_{t-1}, C_1, \ldots, C_{t-1}\} \mid C_t \tag{4.6}$$

Assumption 2 *Given the class label of previous n-1 words $\{C_{t-1}, \ldots, C_{t-n+1}\}$, the class C_t is independent of previous words and distant class history $\{W_1, \ldots, W_{t-1}, C_1, \ldots, C_{t-n}\}$.*

$$C_t \perp \{W_1, \ldots, W_{t-1}, C_1, \ldots, C_{t-n}\} \mid \{C_{t-1}, \ldots, C_{t-n+1}\} \tag{4.7}$$

Given these assumptions the word sequence probability is factorized as :

$$P(W) = \sum_{C_1,\ldots,C_T} \prod_t P(W_t, C_{t-1}, \ldots, C_{t-n+1}|W_{t-1}, \ldots, W_{t-n+1}) \tag{4.8}$$

$$= \sum_{C_1,\ldots,C_T} \prod_t P(W_t|C_t)P(C_t|C_{t-1}, \ldots, C_{t-n+1})P(C_{t-1}, \ldots, C_{t-n+1}|W_{t-1}, \ldots, W_{t-n+1}) \tag{4.9}$$

In common use, n-class models make the assumption that given a word, its class can be determined deterministically : $C_t = cls(W_t)$. Hence,

$$P(C_t|W_t) = \begin{cases} 1 & if\ C_t = cls(W_t) \\ 0 & otherwise \end{cases} \tag{4.10}$$

Using a such a deterministic class definition, the word sequence probability is simplified to :

$$P(W) = \prod_t P(W_t|C_t)P(C_t|C_{t-1}, \dots, C_{t-n+1}) \tag{4.11}$$

Schematically this model can be visualized as in Figure 4.2.

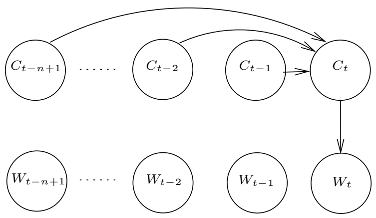


FIG. 4.2 – n-class

As opposed to n-gram models the word probability does not only depend on the lexical context but also syntactic/semantic context via the word class. Besides the additional information on the word classes another advantage of this approach is that it provides a solution to the generalization problem observed in word based n-gram models. Since the state space is reduced the number of parameters is much less and the estimation procedure is more reliable with less data. The disadvantage, however, is that the encoded lexical context information is much more simplified and generally insufficient to achieve an accurate language model. Because of this, the performance of n-class models is generally worse than n-grams, i.e. they yield higher perplexity (see Section 4.8.2 for a definition of perplexity).

4.6 Language Modeling with DBNs

n-gram and n-class models are the most commonly used language models in state-of-the-art speech recognition systems. It has been shown that n-gram models are very efficient when they are trained on a sufficiently informative corpus. The principle inconvenience of these models is that they are computationally very demanding given that they require $|W|^n$ probabilities to be estimated, where n is the model order. Despite the "cut-off" techniques that reduce the complexity, these models remain extremely complex. On the other hand n-class models are less complex since the number of classes is generally much smaller than the vocabulary size. They also have better generalization properties. Nevertheless, they are less accurate and their perplexity is higher. For this reason they are generally combined with n-grams. In such circumstances the realization of a language model is performed first by specifying the model order (n) and then the two models (n-gram and n-class) are trained individually using maximum likelihood criteria. The last step is either a linear combination of these models or an integration of their respective characteristics in a single architecture using maximum entropy techniques [9, 108]. This approach yields quite interesting results, however if we want to better exploit the lexical and syntactic information, a solution would be to consider them in a unique model that is trained within a single procedure.

The DBN formalism provides a theoretical and computational framework to achieve our goal. Our principle idea is to impose no *a priori* hypothesis on the way a language should be represented

but to consider all available data (words, classes, ...) as observations of the dynamic system $\{W_t, C_t\}$. Our goal then is to find the model that has the best description (in terms of perplexity) of these observations. In this way, we let data dictate what influences the pronunciation of a word. In Bayesian networks terminology this is the *structure learning* problem : find the graph structure (and its numerical parameterization) that explains the data at "best".

4.6.1 Model Specifications

In order to define a set of DBN structures plausible for language modeling, we need to specify conditional independence assertions that are linguistically informative and easy to interpret. We also want n-gram and n-class models to be included in this set in order to be able to exploit their linguistic properties. To do so we start by relaxing the CI assumptions of n-class models. We assume that a word does not only depend on its class but also on the classes of words in a limited past and/or future context. To incorporate the properties of n-grams we authorize the dependence of the word to its lexical history. And finally we consider that a class may not only depend on its history but also on the current word and/or other words in a limited past and/or future context. The generalized CI assumptions can be stated as :

Assumption 1 *Given the most recent $n-1$ words and the classes of $m-1$ previous and k future words, a word W_t is independent of all previous words and their classes $\{W_1, \dots, W_{t-n}, C_1, \dots, C_{t-m}\}$.*

$$W_t \perp \{W_1, \dots, W_{t-n}, C_1, \dots, C_{t-m}\} \mid \{C_{t-m+1}, \dots, C_{t+k}, W_{t-n+1}, \dots, W_{t-1}\} \tag{4.12}$$

This assumption specifies the context of a word from both word and class variables allowing also the incorporation of the classes of future words. Schematically the allowed dependences are shown in figure 4.3.

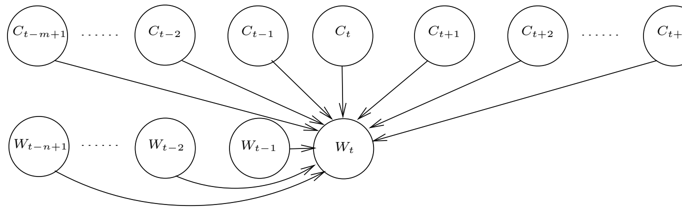


FIG. 4.3 – Allowed dependencies due assumption 1

The second assumption generalizes the class context to include word history. The schematical representation is shown in figure 4.4.

Assumption 2 *Given the most recent $n-1$ words and the class labels of previous $m-1$ words $\{C_{t-1}, \dots, C_{t-m+1}, W_{t-1}, \dots, W_{t-n+1}\}$, the class C_t is independent of previous words and distant class history $\{W_1, \dots, W_{t-n}, C_1, \dots, C_{t-m}\}$.*

$$C_t \perp \{W_1, \dots, W_{t-n}, C_1, \dots, C_{t-m}\} \mid \{C_{t-1}, \dots, C_{t-m+1}, W_{t-1}, \dots, W_{t-n+1}\} \tag{4.13}$$

The joint probability for a specific model is expressed as :

$$P(W, C) = \prod_t P(W_t|\Pi_{W_t})P(C_t|\Pi_{C_t}) \tag{4.14}$$

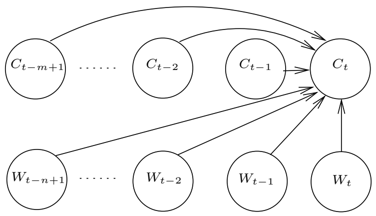


FIG. 4.4 – Allowed dependencies due assumption 2

where Π_{W_t} and Π_{C_t} are the set of parents of W_t and C_t respectively. The model reduces to classical n-class when the set of parents are specified as : $\Pi_{W_t} = \{C_t\}$ and $\Pi_{C_t} = \{C_{t-1}, \ldots, C_{t-n+1}\}$.

The numerical parameterization of the model is given by conditional probability tables (independent of time t) $P(W_t|\Pi_{W_t})$ and $P(C_t|\Pi_{C_t})$ that we note, for $X_t \in \{W_t, C_t\}$, by :

$$\theta_{x,j,\mathbf{k}} = P(X_t = j|\Pi_{X_t} = \mathbf{k}). \tag{4.15}$$

4.7 Learning

Learning a model in the above specified set involves first the specification of parent sets and then the estimation of the associated conditional probability tables. The specification of parents can be manual or automatic using the structure learning approaches in BN formalism as explained in Chapter ?. The estimation of the conditional probability tables is based on maximum likelihood principle as in classical n-gram and n-class models.

The maximum likelihood estimates of the parameters yields :

$$\theta_{x,j,\mathbf{k}} = \frac{N_{x,j,\mathbf{k}}}{\sum_j N_{x,j,\mathbf{k}}} \tag{4.16}$$

where $N_{x,j,\mathbf{k}}$ is the number of realizations of $X_t = j, \Pi_{X_t} = \mathbf{k}$. The classical "smoothing" techniques are still applicable to these models.

4.7.1 Smoothing and Back-off

The maximum likelihood estimates of the conditional densities assign zero probabilities to unseen word and/or class sequences.

$$P_{ML}(w|h) = \frac{N(h,w)}{N(h)} \tag{4.17}$$

Discounting is a method to spare a certain amount of probability density for unseen sequences in the training corpus. This is achieved by discounting the actual counts. For sake of simplicity we mention here *absolute discounting* among several techniques. For a general discussion on discounting schemes see [17].

In absolute discounting a constant value $m < 1$ is subtracted from each count. Then the conditional probabilities are estimated as :

$$P(w|h) = \frac{N(h,w) - m}{N(h)} \tag{4.18}$$

Hence for the context h the probability mass left out for unseen sequences is

$$\beta(h) = 1 - \sum_{w:N(h,w)\neq 0} \frac{N(h,w) - m}{N(h)} \tag{4.19}$$

$$= \frac{\gamma(h)}{N(h)} m \tag{4.20}$$

where $\gamma(h)$ is the number of different words that occur in context h.

$$\gamma(h) = \sum_{w:N(h,w)\neq 0} 1 \tag{4.21}$$

The back-off scheme consists of distribution of this spared mass over unseen events. Each unobserved w in context h, gets its portion in this mass in proportion to its conditional probability in a reduced context h'.

$$P(w|h) = \alpha(h)\beta(h)P(w|h'), \ for \ N(h,w) = 0 \tag{4.22}$$

where $\alpha(h)$ is the normalization factor.

$$\alpha(h) = \frac{1}{\sum_{w:N(h,w)=0} P(w|h')} \tag{4.23}$$

$$= \frac{1}{1 - \sum_{w:N(h,w)\neq 0} P(w|h')} \tag{4.24}$$

In classical n-grams the back-off scheme is used to replace a higher order model whenever there is not enough statistical information. This is achieved, for example, by backing-off a tri-gram to a bi-gram and a bi-gram to a uni-gram. The choice of the back-off model with reduced context is, in general, based on the principle assumption of n-grams, i.e. distant history contains the least amount of information on a word. So whenever there is not enough statistical data, the model is reduced by eliminating the most distant history until a sufficiently reliable model is achieved for the given linguistic event. At each step the model is backed-off to yield the minimal complexity reduction rather than directly backing-off to the simplest model, i.e. backing a tri-gram to uni-gram. The classical n-grams uses a back-off variable elimination order as : $\{W_{t-n+1}, \ldots, W_{t-1}, W_t\}$.

We use a similar back-off scheme for DBN based language models defined in this paper. For a given conditional probability the number of possible back-off models is equal to the number of parents of the word/class variable. Following the intuitive reasoning as classical n-grams at each step we choose the back-off model that yields the minimal complexity reduction. We first eliminate the distant variables based on the assumption that their information content is relatively smaller. Furthermore we first eliminate future context variables considering that historical context is more influential than future context. Hence the back-off variable elimination order we use is : $\{C_{t+k}, \ldots, C_{t+1}, C_{t-m+1}, \ldots, C_{t-1}, C_t, W_{t-n+1}, \ldots, W_{t-1}, W_t\}$.

4.8 Language model evaluation

The quality of a language model is measured by its influence on the performance of a speech recognition system. The best way to compare different language models is to evaluate the error rate reduction of the system when the models are used. However this could be quite demanding due to computational complexity. A language model can also be considered as an autonomous information source independent of the recognition system. In this case it can be evaluated based on its prediction capacity using information theoretic measures.

4.8.1 Entropy of natural language

A language L can be considered as a discrete information source generating a sequence of words $w_1^T = \{w_1, \dots, w_T\}$. The information quantity related to an observation w_t is defined as :

$$I(w_t) = -\log_2 P(w_t) \tag{4.25}$$

The average information quantity of an information source is defined as entropy. The entropy of an information source with independent observations is defined in bits as :

$$H(L) = -\sum_{w_t} P(w_t) \log_2 P(w_t) \tag{4.26}$$

In a natural language the observation w_t depends on previous observations $w_1, \dots, w_{t-1}$. In this case the entropy is defined as the average information quantity of all possible observation sequences :

$$H(L) = -\lim_{T\to\infty} \frac{1}{T} \sum_{w_1^T} P(w_1^T) \log_2 P(w_1^T) \tag{4.27}$$

For a natural language one can argue that it is not necessary to average over all possible word sequences. Indeed, we do not need to know all written and spoken word sequences in order to disambiguate words in a given text. A piece of text or recent parts of a conversation is generally sufficient. Therefore natural language can be considered as an ergodic information source. An information source is ergodic if any long enough observation sequence is sufficient for a statistical representation of the source. With ergodicity assumption the entropy can be computed as :

$$H(L) = -\lim_{T\to\infty} \frac{1}{T} \log_2 P(w_1^T) \tag{4.28}$$

And for a large enough value of T, this term can be approximated as :

$$\hat{H}(L) = -\frac{1}{T} \log_2 P(w_1^T) \tag{4.29}$$

4.8.2 Perplexity as a measure for language model evaluation

Considering a language model as an information source, it follows that an accurate model would achieve the entropy of language. Therefore it makes sense to define a measure related to entropy for language model evaluation. One such measure is perplexity (PP). It is defined as :

$$PP = 2^{\hat{H}} \tag{4.30}$$

$$= \hat{P}(w_1, w_2, \dots, w_T)^{-1/T} \tag{4.31}$$

where $\hat{P}(w_1, w_2, \dots, w_T)$ is the probability estimate assigned to the word sequence $w_1, w_2, \dots, w_T$ by the language model.

Perplexity can be considered to be a measure of on average how many different equally most probable words can follow any given word. Lower perplexities represent better language models, although this simply means that they 'model language better', rather than necessarily work better in speech recognition systems : perplexity is only loosely correlated with performance in a speech recognition system.

Model	$P(W)$	PP
DBN1	$\prod_t P(w_t\|w_{t-1})$	65.24
DBN2	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1})$	151.31
DBN3	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1}c_{t-2})$	130.00
DBN4	$\prod_t P(w_t\|c_t, c_{t-1})P(c_t\|c_{t-1})$	113.13
DBN5	$\prod_t P(w_t\|c_t, c_{t+1})P(c_t\|c_{t-1})$	121.98
DBN6	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1})$	94.35
DBN7	$\prod_t P(w_t\|c_t, c_{t-1})P(c_t\|c_{t-1}, c_{t-2})$	97.19
DBN8	$\prod_t P(w_t\|c_t, c_{t+1})P(c_t\|c_{t-1}, c_{t-2})$	104.8
DBN9	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1}, c_{t-2})$	81.06
DBN10	$\prod_t P(w_t\|w_{t-1}, c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1})$	78.00
DBN11	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1})$	85.20
DBN12	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1}, c_{t-2})$	73.20
DBN13	$\prod_t P(w_t\|w_{t-1}, c_{t-1})P(c_t\|w_t)$	70.86
DBN14	$\prod_t P(w_t\|w_{t-1}, c_{t-1}, c_{t-2})P(c_t\|w_t)$	**63.67**

TAB. 4.1 – Training is performed on 11 months of "Le Monde 87". Perplexity is computed using 1 month containing more than 1M words.

4.9 Experiments

We carried out several experiments using three different corpora. In these experiments we compare the performances of different DBN model structures with classical n-gram and n-class models in terms of model perplexity.

4.9.1 Le Monde

The training and test corpus are extracted from *Le Monde* newspaper. We use 22M words for training and a test corpus of 2M words. The vocabulary consists of the most frequent 5000 words. The training corpus has been labeled automatically by a set of 200 syntactic classes set defined by hand [115]. All models used in the experiments are smoothed using absolute discounting method [101]

Table 4.1 shows perplexity results of 14 different Bayesian network language models. The models DBN1, DBN2 and DBN3 correspond to bi-gram, bi-class and tri-class models respectively. In order to achieve our objective to find the best model we set the bi-class model as baseline and extend it incrementally by incorporating additional lexical and/or syntactic context. We also introduce the concept of right context of a word. DBN6 is a typical example of this case that integrates not only the left class context of a word but also its right syntactic context. We obtain a 16.6% improvement with respect to DBN4 that proves the importance of right context. It is true that linguistically this is not a surprising result. On the other hand, it is difficult to realize the use of right context in speech recognition, but this could be achieved with a multi-pass decoding scheme. DBN5, on the other hand, shows that left context is quite important. That is why its removal reduces the results by 7.8%. A significant perplexity reduction is observed if a word not only depends on its syntactic but also lexical context. Indeed, DBN11 yields an improvement of 24.6% with respect to DBN4. This results confirms that lexical history is indispensable and that syntactic history provides a significant improvement.

Pushing forwards this strategy, we achieve a model that is not only much better than the bi-class but also better than the bi-gram. Indeed, the model DBN14 that is shown in Fig. 4.5, reduces the perplexity by 57.9% with respect to DBN2 and 2.4% with respect to DBN1 (bi-gram). This new approach outperforms the classical bi-gram.

4.9.2 MIAMM

This corpus is extracted from the European project MIAMM, its aim is to build up a platform of an oral multi-modal dialog. The corpus contains 71000 different queries expressed in French. Some examples of these queries are given in the table 4.2. This corpus is used in an understanding speech process [70]. Tests have been performed on a closed vocabulary of 160 words. Clusters have been extracted automatically by using HTK tools and each word has been assigned to only one class.

Show me the contents of my bookmarks.
I would like to know if you can take the contents that I prefer.
Do you want to select the titles that I prefer.
Is it possible that you select the first of my bookmarks.
Is it possible to indicate me a similar thing.
Can you show me only December 2001.
It is necessary that you print the list that I used early this morning.

TAB. 4.2 – Some examples of queries in the MIAMM corpus.

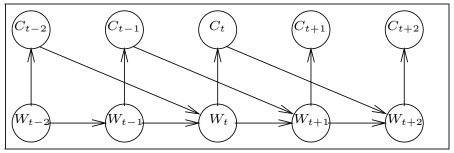


FIG. 4.5 – DBN structure for $\prod_t P(w_t|w_{t-1}, c_{t-2})P(c_t|w_t)$, that yields a lower perplexity than bi-gram.

Table 4.3 summarizes the results obtained on MIAMM corpus using baseline and different DBN language models. One can remark that the value of perplexity is very low, this is due to the the size of vocabulary (only 160 words) and also to the weakness of linguistic structure of this corpus. Indeed, the variability of this corpus is very reduced, so many sentences differ only by one or two words. One can observe that all DBNs (DBN1, DBN2, DBN3, DBN4) which do not take into account a word history fail in terms of perplexity performance. Nevertheless, the performances of these models are better than those obtained by n-class. This is due to the fact that they take into account a larger class history. On the other hand, DBN5 to DBN8 show better performances due to a richer linguistic left context composed of word and class histories. In particular DBN7, shown in Figure 4.5, yields an improvement of 13.1% with respect to bigram. More importantly DBN8, which has a complexity asymptotically lower than trigram, not only yields an improvement of 18.8% with respect to bigram but also is slightly better than a trigram. These results have been assessed by a t-test which shows that they are statistically significant.

Model	$P(W)$	PP
2-gram	$\prod_t P(w_t\|w_{t-1})$	2.81
3-gram	$\prod_t P(w_t\|w_{t-1}, w_{t-2})$	2.29
4-gram	$\prod_t P(w_t\|w_{t-1}, w_{t-2}, w_{t-3})$	2.24
2-class	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1})$	11.54
3-class	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1}, c_{t-2})$	10.42
4-class	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1}, c_{t-2}, c_{t-3})$	9.23
DBN1	$\prod_t P(w_t\|c_t, c_{t-1})P(c_t\|c_{t-1})$	8.14
DBN2	$\prod_t P(w_t\|c_t, c_{t+1})P(c_t\|c_{t-1})$	7.80
DBN3	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1})$	5.72
DBN4	$\prod_t P(w_t\|c_{t-1}, c_t, c_{t+1})P(c_t\|c_{t-1}, c_{t-2})$	5.16
DBN5	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1})$	4.16
DBN6	$\prod_t P(w_t\|w_{t-1}, c_t)P(c_t\|c_{t-1}, c_{t-2})$	3.75
DBN7	$\prod_t P(w_t\|w_{t-1}, c_{t-2})P(c_t\|w_t)$	2.44
DBN8	$\prod_t P(w_t\|w_{t-1}, c_{t-2}, c_{t-3})P(c_t\|w_t)$	2.28

TAB. 4.3 – Perplexity results on MIAMM corpus, $|W| = 160$, $|C| = 10$.

Model	$P(W)$	$\|C = 1\|$	$\|C\| = 15$	$\|C\| = 25$	$\|C\| = 45$
2-gram	$\prod_t P(w_t\|w_{t-1})$	38.5 / 50.7			
3-gram	$\prod_t P(w_t\|w_{t-1}, w_{t-2})$	30.9 / 39.3			
2-class	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1})$		94.1 / 139.2	84.2 / 122.7	77.5 / 111.9
3-class	$\prod_t P(w_t\|c_t)P(c_t\|c_{t-1}, c_{t-2})$		90.6 / 133.4	80.0 / 115.9	71.5 / 102.2
DBN1	$\prod_t P(w_t\|w_{t-1}, c_{t-2})P(c_t\|w_t)$		36.3 / 47.2	35.2 / 45.6	34.4 / 44.3

TAB. 4.4 – SPORT corpus perplexity results with class-based models computed with/without $< UNK >$

4.9.3 SPORT

The SPORT corpus consists of articles dedicated to sport news (SPORT) extracted from a French newspaper. The corpus consists of 8500 sentences, the tests are performed using an open vocabulary of 2000 words with different cluster sizes.

Table 4.4 shows the baseline and DBN models performance for SPORT corpus. Several set of classes extracted by HTK tool have been used. The first remark is that the use of a higher number of classes leads to a reduction of perplexity. As for MIAMM tests, the use of a history which combines classes and words is beneficial to language models and yields better results. The best performance (for both cases : with and without UNK) is obtained by DBN1 (also represented by Figure 4.5) which yields an improvement of 12,6% in comparison to bigram. The other important point is that, even if the trigram computational complexity ($O(|W|^3)$) is higher than the one of DBN1 ($O(|W|^2|C| + |W||C|)$), there is only (in average) 5 points between their perplexities, which is relatively small. Thus we can hope that, with a larger vocabulary and with a classification containing more classes, we can build DBN models similar to DBN1 with equivalent performances as a trigram.

Chapitre 5

Speech Features Extraction

Sommaire

5.1 Introduction . . . **103**
5.2 Contributions . . . **104**
5.3 Overview of speech parameterization schemes in ASR . . . **104**
5.3.1 Common feature extraction schemes for ASR . . . 104
5.3.2 Common processing blocks . . . 106
5.4 Wavelets in speech signal parameterization . . . **107**
5.4.1 Wavelets in speech recognition . . . 107
5.4.2 Experiments with wavelet coefficients . . . 110
5.4.3 Multi-resolution cepstral features . . . 117
5.5 Frequency Filtering . . . **117**
5.5.1 Frequency localized features . . . 117
5.5.2 Experiments : Frequency Filtering in Real Noise Conditions . . . 119
5.5.3 Experiments : Effect of number of filter-banks in FF . . . 120
5.6 Wavelet Filtering . . . **121**
5.6.1 Frequency Filtering and Wavelet transform . . . 121
5.6.2 Multi-resolution Frequency Filtering . . . 123
5.7 Experiments on full Aurora 3 dataset . . . **125**
5.8 Conclusion and perspectives . . . **126**

5.1 Introduction

State-of-the-art automatic speech recognition (ASR) systems are based on Mel-Frequency Cepstral Coefficients (MFCC) speech parameterization and on probabilistic modeling of the signal using Hidden Markov Models(HMM). These systems lead to the best recognition performances in ideal "lab" conditions or for easy tasks. However, in real word conditions of speech processing (noisy environment, spontaneous speech, non-native speakers...), the performances of MFCC-HMM based ASR systems can decrease drastically.

The discrepancies of MFCC-HMM based systems in real-world situations are mainly due to their weakness of capturing many acoustic phenomena that are specific to speech and which are very informative. Indeed, MFCC do not have a clear physical interpretation and are not localized in the frequency domain. Still, cepstral analysis is the most widely used parameterization even

if it is well known that a frequency-localized parameterization would increase noise robustness and would allow miming of the human auditory system [2, 48, 27]. Moreover, it would allow modeling of frequency dynamics, which are weakly modeled by classical systems and which can be phonetically very informative. Thus, an effort should be made to come up with new speech parameterizations that yield frequency-localized features. These features should be noise robust (at least as MFCC) and should allow for acoustic models which can capture *both* temporal and frequency dynamics. This is the goal of the work presented in this chapter.

The chapter is organized as follows. In the first section we briefly overview classical parameterization schemes for ASR. Next, we review wavelet based parameterization schemes and provide an empirical analysis. Based on these analysis we propose multi-resolution cepstral features. The following sections are focused on frequency localized features. We start by the frequency filtering approach and provide experimental results on Aurora 3 database [97, 99]. Considering the discrepancies in these experiments and the properties of wavelet analysis, we provide a new perspective for this approach, i.e. wavelet filtering.

5.2 Contributions

- A thorough analysis of wavelet based speech feature extraction algorithms with experiments.
- A novel multiresolution cepstral feature extraction technique.
- Analysis of frequency filtering approach.
- A new perspective for frequency filtering in wavelet domain.
- A novel technique for frequency localized cesptral features.

5.3 Overview of speech parameterization schemes in ASR

The goal of front-end speech processing in ASR is to attain a projection of the speech signal to a compact parameter space where the information related to speech content can be extracted easily. Most parameterization schemes are developed based on the source-filter model of speech production mechanism [105, 106]. In this model, speech signal is considered as the output of a filter (vocal tract) whose input source is either glottal air pulses or random noise. For voiced sounds the glottal excitation is considered as a slowly varying periodic signal. This signal can be considered as the output of a glottal pulse filter feed with a periodic impulse train. For unvoiced sounds the excitation signal is considered as random noise. A simplified block diagram of this model is given in Figure 5.1.

5.3.1 Common feature extraction schemes for ASR

Linear Prediction Coefficients Linear predictive coding (LPC) has been considered one of the most powerful techniques for speech analysis. LPC relies on the lossless tube model of the vocal tract. The lossless tube model approximates the instantaneous physiological shape of the the vocal tract as a concatenation of small cylindrical tubes. The model can be represented with an all pole (IIR) filter.

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

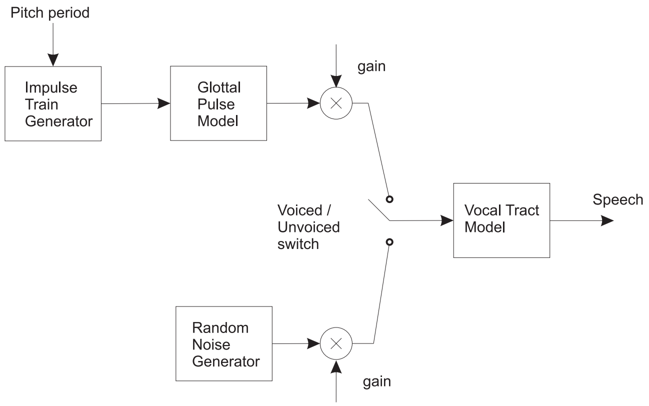


FIG. 5.1 – Source-filter model for speech production

The speech production model is then formulated as :

$$s(t) = \sum_{k=1}^{p} a_k s(t-k) + e(t).$$

LPC coefficients a_k can be estimated using autocorrelation or covariance methods [106].

MFCCs (Mel-Frequency Cepstral Coefficients) Cepstral [12] analysis denote the unusual treatment of frequency domain data as it were time domain data [8]. The cepstrum is a measure of the periodicity of a frequency response plot. The unit measure in cepstral domain is second but it indicates the variations in the frequency spectrum.

One of the powerful properties of cepstrum is the fact that any periodicities, or repeated patterns, in a spectrum will be mapped to one or two specific components in the cepstrum. If a spectrum contains several harmonic series, they will be separated in a way similar to the way the spectrum separates repetitive time patterns in the waveform. The mel-frequency cepstral coefficients proposed by Mermelstein [29] make use of this property to separate the excitation and vocal tract frequency components in cepstral domain. The spectrum of excitation signal is composed of several peaks at the harmonics of the pitch frequency. This constitutes the quickly varying component of the speech spectrum. On the other hand the vocal tract frequency response constitutes the slowly varying component of the speech spectrum. Hence a simple low pass liftering (i.e. filtering in cepstral domain) operation eliminates the excitation component. Briefly the source separation (cepstral smoothing) in cepstral domain is performed as follows :

In time domain speech signal is represented as the response of the vocal tract filter to the glottal excitation signal :

$$s(t) = e(t) * h(t)$$

[12]The terminology in cepstral domain is proposed as the anagrams of frequency domain nomenclature, i.e. cepstrum for spectrum, quefrency for frequency.

Using a Fourier transform, the convolution operation is transformed to multiplication in the magnitude spectrum :

$$|S(f)| = |E(f)|.|H(f)|$$

Using the logarithm operation the multiplication is transformed to addition :

$$\log |S(f)| = \log |E(f)| + \log |H(f)|$$

Now using an inverse Fourier transform (or equivalently discrete cosine transform (DCT)) we switch to cepstral domain where the signal is represented with two additive components :

$$s\hat{(t)} = e\hat{(t)} + h\hat{(t)}$$

where one can find a threshold T for which

$$\hat{s}(t) = \begin{cases} \hat{e}(t) for t > T, \\ \hat{h}(t) for t < T. \end{cases}$$

5.3.2 Common processing blocks

In the following we summarize common processing blocks in speech signal parameterization for ASR.

Framing One of the basic assumptions in speech signal processing is quasi-stationarity. The speech signal is considered to embody uniform characteristics for small time intervals (approximately 10ms). Therefore to extract stationary characteristics the speech signal is analyzed in sequential frames. The frames are overlapped such that the coefficients have a smooth transition between frames. The overlap ratio is generally chosen as 1/3 of window length. However, as a consequence of overlapping adjacent frames are highly correlated.

Pre-emphasis Due to physiological characteristics of the speech production system the voiced segments of the speech signal has a negative spectral slope [106]. In the source-filter model this characteristic can be associated to the glottal pulse model that can be represented as a simple one-pole filter with a pole near $z = 1$. The goal of pre-emphasis filtering is to boost high frequencies using a first order FIR filter that cancel out the glottal pulse model.

$$H(z) = 1 - az^{-1} \qquad 0.9 \leq a \leq 1.0 \tag{5.1}$$

The resulting signal has a flattened spectrum such that low and high frequency formants have similar amplitudes.

Windowing Frames are windowed in order to minimize the discontinuities at the frame boundaries. The window is selected so as to attenuate the signal around the frame boundaries (e.g. Hamming, Hanning, Gauss).

Filter bank analysis The human ear resolves frequencies non-linearly across the audio spectrum and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves recognition performance. Filter-bank analysis provides the desired non-linear frequency resolution. A common approach is to use a simple Fourier transform based filter-bank designed to give approximately equal resolution on a mel-scale which is defined by

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{5.2}$$

Normalization/Compression Generally a *log* transformation or root-compression. Alternatives are presented in [125].

Cepstrum computation Cepstral coefficients are computed from the log filter bank amplitudes m_j using the Discrete Cosine Transfom (DCT) as

$$c_j = \sqrt{\frac{2}{N}} \sum_{i=1}^{N} m_j \cos(\frac{\pi i}{N}(j - 0.5)) \tag{5.3}$$

where N is the number of filterbank channels.

Cepstral Liftering Higher order cepstra are numerically quite small and this results in a very wide range of variances when going from the low to high cepstral coefficients. Therefore it is convenient to re-scale the cepstral coefficients to have similar magnitudes. Liftering is performed using the following formula

$$c'_n = (1 + \frac{L}{2} \sin \frac{\pi n}{L}) c_n \tag{5.4}$$

Time-derivatives The performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters. This is due to the high correlation between each frame that is not handled with HMM modeling. Appending the time derivative coefficients to each frame the correlation between successive frames can be modeled with HMMs.

The time derivatives are computed using the following regression formula. Second derivatives are similarly computed from first order coefficients using the same formula.

$$d_n = \frac{\sum_{\tau=1}^{w} \tau (c_{n+\tau} - c_{n-\tau})}{2 \sum_{\tau=1}^{w} \tau^2}$$

5.4 Wavelets in speech signal parameterization

5.4.1 Wavelets in speech recognition

The main motivation to use wavelet analysis in speech recognition is its time-frequency localization properties [93]. A fundamental paper by Allen [2] on human speech recognition mentions that perceptual features are local in frequency and asynchronous in time. Fourier analysis has good frequency localization properties but no time localization. Therefore in ASR, short time Fourier transform is used where the analysis is performed on fixed length windows such that a partial time localization is achieved. On the other hand wavelet analysis provides a formal framework to decompose a signal into components that are well localized in time and in frequency.

One of the attempts to use this property in ASR is proposed by Janer et al. [71]. They propose a segment based speech recognition system where the speech signal is separated into non-uniform frames. The segmentation is performed using wavelet analysis. Frame lengths are adjusted according to their information content. The advantage of using wavelet analysis in this technique is that a simple thresholding is sufficient in wavelet domain to identify the time span of the frequency content.

Most techniques that use wavelet analysis in speech parameterization attempt to replace the classical Fourier analysis based processing blocks with wavelet analysis. These include the

computation of spectral representation, shaping of the spectra for auditory representation and cepstral analysis.

In the following we cite some approaches that perform the spectral analysis using wavelets. Next we summarize different techniques to compute features from the wavelet coefficients. Finally we cite cepstrum computation approaches using wavelets.

Spectral analysis using wavelets

The use of wavelets in spectral analysis for speech recognition formalizes the filter-bank analysis scheme in the common parameterization techniques, i.e. MFCC. The rescaling of the STFT spectrum in classical MFCC computation with a Mel-scaled filter bank actually adjusts the fixed frequency resolution of the STFT to the Mel scale such that high frequencies have less resolution. Wavelet transform on the other hand provides a signal decomposition that implicitly has this property. The following approaches replace the STFT based Mel-scale filter-bank analysis with a wavelet packet decomposition.

One of the first attempts was by Erzin [46] where a tree structured perfect reconstruction filter bank is used to approximate the Mel scale filter bank. The selected filter-bank corresponds to a bi-orthogonal wavelet transform. The low pass filter is selected as a 7-th order Lagrange filter. The resulting frequency partitioning for 22 sub-bands in the interval $[0, 4000]$ Hz is shown in Figure 5.2. The same partitioning is also used in [69]. In [81], Kryze uses a similar approach using wavelet packets to obtain the the partitioning in Figure 5.3.

Farooq in [47], uses an admissible wavelet packet tree structure using Daubechies-6 wavelets to yield 24 frequency bands (the same number of Mel filters). The corresponding frequency partitioning is shown in Figure 5.4.

In [110, 109] another wavelet packet tree structure is used with Daubechies-32 wavelets to yield the 24 subband partitioning in Figure 5.5. In [16] the mel-scaled packet tree with Daubechies-12 tap FIR filters are used to obtain 22 sub-bands as shown in Figure 5.6.

FIG. 5.2 – Frequency partitioning of [46, 69]

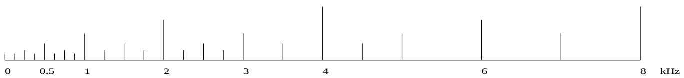


FIG. 5.3 – Frequency partitioning of [81]

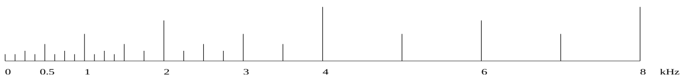


FIG. 5.4 – Frequency partitioning of [47]

Fig. 5.5 – Frequency partitioning of [110, 109]

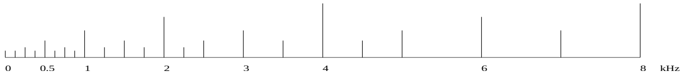


Fig. 5.6 – Frequency partitioning of [16]

Other approaches that use the same principle can be found in [121, 125, 89]. The main principle in all these approaches is to use a wavelet decomposition to approximate the Mel-scale frequency resolution. The differences are the wavelet functions used. The choice of the wavelet function is generally experimental. A thorough theoretical analysis for the selection of the best wavelet function with desired localization and smoothness properties for ASR does not exist. Some attempts have been made to make a best basis search in [91, 90].

Feature extraction from wavelet coefficients

In MFCCs the spectral features are computed as energy in each sub-band. In the following we summarize the features used in wavelet spectrum.

Notation : w_k^j denotes the wavelet coefficient at position k and scale j. N_j denotes the number of coefficients at scale j. f_j denotes the feature vector for scale j.

Average magnitudes In [46] the feature extraction from the sub-band signals is computed as average magnitudes. In [81] the averaging is performed over the same number of samples in each sub-band. This provides a hierarchical time resolution.

$$f_j = \frac{1}{N_j} \sum_{k=1}^{N_j} |w_k^j| \tag{5.5}$$

Energy Another common technique is to use the energy in each band [110, 109, 47]. Similar to [81] a hierarchical time resolution for energy computation is used in [57].

$$f_j = \frac{1}{N_j} \sum_{k=1}^{N_j} |w_k^j|^2 \tag{5.6}$$

Features with hierarchical time resolution Wavelet coefficients have finer time resolution for high frequency components. Lowest frequency coefficients span the full time scale whereas higher frequency coefficients span a smaller time window with increasing frequency. The hierarchical time resolution technique extracts features concentrated at the center of the analysis window. The features are computed as average magnitudes [81] or energies [57] where the averaging is performed for a fixed number of coefficients at each scale.

$$f_j = 20log_{10}(\frac{1}{N_J} \sum_{k=(N_j-N_J)/2}^{(N_j+N_J)/2} |w_k^j|) \qquad for\ j = 1 \ldots J \tag{5.7}$$

$$f_j = 20log_{10}(\frac{1}{N_J} \sum_{k=(N_j-N_J)/2}^{(N_j+N_J)/2} |w_k^j|^2) \qquad for\ j = 1 \ldots J \tag{5.8}$$

Teager Energy [69] proposes to use the discrete Teager energy operator (TEO) rather that the instantaneous energy. This technique allows the reduction of the noise bias using the modulation energy tracking capability of TEO. Magnitude of Teager energy is used to ensure non-negativity of f_j

$$f_j = \frac{1}{N_j} \sum_{k=1}^{N_j} |(w_k^j)^2 - w_{k-1}^j w_{k+1}^j| \tag{5.9}$$

Thresholded coefficients Another technique is to use the wavelet coefficients directly as features rather than computing the magnitude averages or energies [80]. The principle of this technique is to choose the coefficients with the highest magnitude. In [121] two maximum coefficients at each scale is used. In [63] 80-248 out of 256 coefficients are used using a hard thresholding.

$$f_j = \{w_k^j\} \text{ such that } |w_k^j| > \theta \tag{5.10}$$

Cepstral analysis using wavelets

Classical cepstral features are computed using a discrete cosine transform (DCT). DCT analysis is a signal decomposition technique where the basis vectors are simple sinusoids. Wavelet analysis can also be used to compute cepstrum to have a better frequency-quefrency resolution [61]. It has also been shown that wavelet transform is a better decorrelator for the subband energies [110].

In [110, 109] a three level wavelet transform with Daubechies 4 tap filters is used. In [61], 4 to 32 level of decompositions are evaluated using Cohen, Daubechies and Feauveau's compactly supported bi-orthogonal spline wavelet with 3-taps and 5-taps for the high pass and the low pass filter respectively.

In section 5.5, we provide a more detailed analysis of the use of wavelets for cepstral analysis.

5.4.2 Experiments with wavelet coefficients

In these experiments we evaluate the performance of different feature extraction algorithms based on wavelet analysis. We test several wavelet decomposition methods and feature computation schemes under clean and noisy conditions. Our purpose is not to achieve significant recognition improvements but to analyze the characteristics of wavelet based parameterization schemes. In this respect we setup a simple experimental setting. In the first set of experiments we use dyadic wavelet transform that provides higher resolution for low frequencies. Next we use wavelet packet decomposition for a more detailed analysis of high frequencies.

The first experiment focuses on the choice of wavelet function. We fix the coefficient combination scheme and use several wavelet functions for spectral decomposition at different resolutions. The recognition performances are compared at different decomposition levels. In the second experiment we analyze the effects of coefficients at different resolutions. Namely we perform recognition tests where only a subset of coefficients are used. The results of these two experiments are then used to choose an appropriate wavelet function and decomposition level. The third experiment compares different coefficient combination schemes proposed in the literature. The next set of experiments uses wavelet packet decomposition. We tested different frequency partitionings and compared the results with MFCCs. Finally, we illustrate a noise compensation scheme using wavelet domain features.

Experimental setup

The experiments are performed on a simple isolated digit recognition task. We use the TIDIGITS database. Training is performed on the isolated male part of the training set and the tests are performed on the isolated male part of the test set. The HTK toolkit is used to construct whole word HMMs. Each HMM has 6 states : 4 emitting, 2 non-emitting. The output density for each state is modeled as a mixture of 8 Gaussians.

Baseline system

The baseline parameterization is obtained using the ETSI SQL W1007 standard front-end [68]. This is a cepstral analysis scheme where 13 Mel-frequency cepstral coefficients are extracted from 23 Mel filter-bank energies. The analysis is performed on 25ms Hamming windowed frames with 10ms frame shift. A notch filtering is applied to compensate signal offset. Pre-emphasis filter with $a = 0.97$ is used. The Mel filter-bank has 23 frequency bands in the range from 64 Hz to half of the sampling frequency. The resulting feature vector consists of 12 cepstral coefficients (excluding c0) and logarithmic frame energy. The first and second time derivatives (Δ and $\Delta\Delta$) are also appended as dynamic features in all the experiments.

The recognition accuracy achieved with the baseline front-end is **96.40%**.

Choice of wavelet function

Aim : In this experiment we use dyadic wavelet decomposition for spectral feature extraction and analyze the effect of the wavelet function on the recognition results. Various wavelets (daubechies, coiflets and symlets) are used with different smoothness properties and decomposition levels from 3 to 8. We want to observe the effect of increasing the number of wavelet decomposition levels and smoothness of the wavelet basis.

Setup : The parameters are computed as average magnitude of wavelet coefficients at each scale [81]. The feature for each scale is computed as :

$$f_j = 20log_{10}(\frac{1}{N_j}\sum_{k=1}^{N_j}|w_k^j|) \qquad for\ j = 1 \dots J \tag{5.11}$$

where w_k^j denotes the wavelet coefficient at time k and scale j. N_j denotes the number of coefficients at scale j. We not only use the detail coefficients but also the scaling coefficients as well.

$$g_J = 20log_{10}(\frac{1}{N_J}\sum_{k=1}^{N_J}|s_k^J|) \tag{5.12}$$

where J denotes the number of decomposition levels.

#levels	3	4	5	6	7	8
db-4	81.10	91.00	**91.65**	90.75	88.30	87.15
db-8	81.42	93.94	**94.19**	93.29	91.49	89.85
db-12	82.49	94.11	**94.35**	93.62	92.23	91.41
db-16	82.49	**94.27**	93.21	91.73	89.77	89.12
db-20	82.32	**94.35**	93.21	91.41	89.93	89.28
db-24	81.34	**94.19**	93.54	92.47	90.59	89.36
db-28	81.67	**94.60**	93.62	92.06	90.26	88.87
db-32	81.91	86.66	**88.05**	86.66	85.19	83.55
coiflet-1	78.56	85.11	**87.32**	84.53	83.72	80.52
coiflet-3	82.16	93.21	**93.94**	92.96	91.24	89.44
coiflet-5	82.24	**94.11**	93.13	92.31	90.59	89.77
symlet-2	79.38	85.02	**86.50**	84.94	83.55	81.42
symlet-4	83.80	90.67	**91.41**	90.67	88.30	86.82
symlet-8	82.32	94.19	**95.01**	93.70	92.23	90.34

TAB. 5.1 – Recognition rates for various wavelet functions using different level of decompositions (%).

Results : The results summarized in Table 5.1 are organized as follows. In each column, for each wavelet type, the smoothness increases as we go down. In each row, the decomposition level is increased from right to left, yielding a higher resolution at low frequencies.

Peak recognition rate is achieved at 4 decomposition levels with Daubechies-28 wavelet.

When we read Table 5.1 column-wise, we observe that the regularity (i.e., smoothness) of the wavelet function has a globaly positive effect on the recognition score at all decomposition levels. For coiflets and symlets the recognition score increases as we increase the smoothness. For Daubechies wavelets, however, the score starts to decrease after db-12.

Now if we read the Table row-wise, we realize that best results are attained around 4-6 decomposition levels for all wavelets. The poor performance for small decomposition levels can be explained easily due to the limited frequency partitioning. The resolution for low frequencies is too coarse and this causes a loss of valuable formant information. When we increase the number of decomposition levels, the resolution at low frequencies is increased. However after 6 levels of decomposition one achieves a high resolution at very low frequencies which does not necessarily contain useful information for recognition. The degradation at these levels is probably due to high correlation between the low frequency coefficients.

Conclusions : The number of decomposition levels of wavelet analysis should be adjusted such that useful information is represented with proper resolution. 4-6 levels of decomposition seems to be appropriate.

The smoothness of the wavelet function also affects the recognition rates. Speech signal is quite smooth for vowels and and highly irregular for consonants. Therefore neither very smooth nor very irregular wavelets yields good recognition rates.

Effect of scaling and detail coefficients

Aim : The aim of this experiment is to identify the informative coefficients in the decomposition. We perform experiments that use only a subset of coefficients. In the first set of experiments we exclude detail coefficients at different levels. In the second set we exclude the scaling coefficients for different decomposition levels.

Experiment 1 : Excluding detail coefficients

Setup : For this experiment we use Daubechies-4 (d-4) and Coiflet-5 (coif-5) wavelets. We choose db-4 for its time localization properties and coif-5 for smoothness properties. The parameterization scheme is the same as the previous experiment, i.e. average magnitude of the coefficients at each scale. At each step the highest detail coefficients are excluded from the parameterization scheme up to three levels.

excluded coeffs	-	d1	d1,d2	d1,d2,d3
coif-5, 3 levels	82.24	77.74	-	-
coif-5, 4 levels	94.11	**94.52**	86.66	-
coif-5, 5 levels	93.13	93.13	84.12	67.28
coif-5, 6 levels	92.31	90.26	71.75	68.90
coif-5, 7 levels	90.59	88.63	80.44	68.09
coif-5, 8 levels	89.77	87.97	79.38	66.37

TAB. 5.2 – Recognition rates for experiment 2 (%).

Results : The results are summarized in Table 5.2. The recognition rate decreases if we remove detail coefficients. The only exception is observed for coiflets with 4 levels of decomposition : the score increases if the finest detail coefficients are removed.

The conclusion of this experiment is that the detail coefficients do contain class specific information. The frequency partitioning of wavelet analysis is too coarse for high frequency regions. This partitioning divides the frequency plane into two regions at each scale and only the low frequency region is processed further partitioned in the next level of decomposition. The detail coefficients span an important portion of the spectrum.

For further analysis of the effect of different frequency components wavelet packet decomposition should be employed which enables a finer partitioning in the high frequency regions.

Experiment 2 : Excluding scaling coefficients The scaling coefficient is the lowest frequency component in the decomposition. As the level of decomposition increases this coefficient corresponds to the DC component of the signal. In this experiment we evaluate the effect of this coefficient for different decomposition levels.

# levels	3	4	5	6	7	8
scores with scaling coeff	82.24	94.11	93.13	92.31	90.59	89.77
scores without scaling coeff	67.68	90.51	95.17	94.03	92.06	90.59

TAB. 5.3 – Recognition rates with and without scaling coefficients, using coiflet-5 wavelet(%).

Results : When scaling coefficient is removed from the parameterization the score decreases for small decomposition levels (see Table 5.3. However if the decomposition level is high, removing the scaling coefficient increase the recognition rate. At high levels the scaling coefficient has less

information. In these cases, removing the scaling coefficient is like removing the DC offset of the signal which indeed is a normalization of the database.

Comparing different parameterizations in wavelet domain

Aim : In this experiment we compare different parameterization techniques using wavelet coefficients as described in Section 5.4.1.

Setup : We use Daubechies-8 wavelet for decomposition. The compared parameterization schemes are :

1) MAG : Average magnitude at each scale [46].

$$f_j = 20log_{10}(\frac{1}{N_j}\sum_{k=1}^{N_j}|w_k^j|) \qquad for\ j = 1 \dots J \tag{5.13}$$

2) H_MAG : Average magnitude computed with hierarchical time resolution [81].

$$f_j = 20log_{10}(\frac{1}{N_J}\sum_{k=(N_j-N_J)/2}^{(N_j+N_J)/2}|w_k^j|) \qquad for\ j = 1 \dots J \tag{5.14}$$

3) ENG : Energy at each scale [110, 109, 47].

$$f_j = 20log_{10}(\frac{1}{N_j}\sum_{k=1}^{N_j}|w_k^j|^2) \qquad for\ j = 1 \dots J \tag{5.15}$$

4) H_ENG : Energy computed with a hierarchical time resolution [57].

$$f_j = 20log_{10}(\frac{1}{N_J}\sum_{k=(N_j-N_J)/2}^{(N_j+N_J)/2}|w_k^j|^2) \qquad for\ j = 1 \dots J \tag{5.16}$$

5) T_ENG : Teager energy [69].

$$f_j = 20log_{10}(\frac{1}{N_j}\sum_{k=1}^{N_j}|(w_k^j)^2 - w_{k-1}^j w_{k+1}^j|) \qquad for\ j = 1 \dots J \tag{5.17}$$

# levels	3	4	5	6	7	8
MAG	81.42	93.94	94.19	93.29	91.49	89.85
H_MAG	82.61	93.19	**94.35**	90.83	89.11	87.96
ENG	74.88	91.24	90.67	90.10	88.54	87.56
H_ENG	77.50	92.31	93.70	91.08	88.38	88.05
T_ENG	73.00	90.75	91.41	90.26	89.12	88.30

TAB. 5.4 – Recognition rates using different parameterization schemes (%).

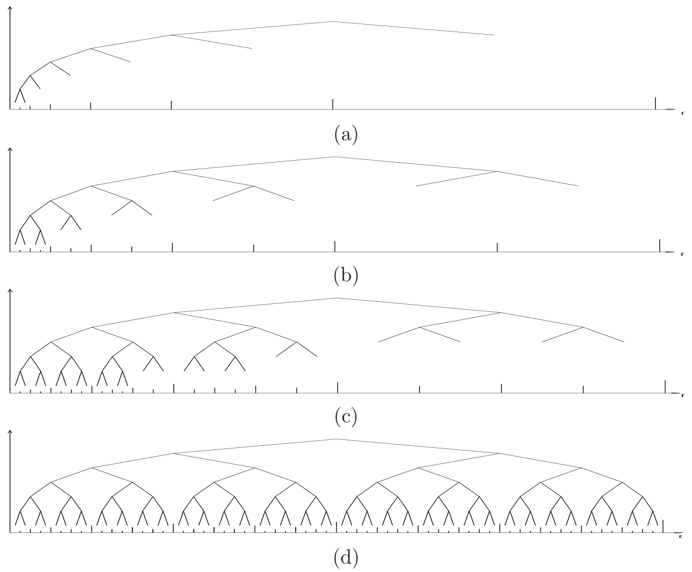


FIG. 5.7 – Frequency resolution of different wavelet decomposition trees : (a) dyadic tree, (b) a test tree, (c) mel-scale approximation tree [47], (d) full wavelet packet decomposition tree.

Results : The results are given in Table 5.4. For all parameterization techniques the best recognition rate is achieved at 5 decomposition levels. The best recognition score is achieved using average magnitudes computed with a hierarchical time resolution. Using a hierarchical time resolution has a positive effect on the recognition rates. This technique extracts the information localized around the center of analysis window. For high resolutions the time localization is more precise.

Frequency resolution tuning with wavelet packets

Aim : Dyadic wavelet transform yields a dyadic partitioning of the frequency such that low frequencies are represented with high resolution and high frequencies with low resolution (see Figure 5.7.(a)). On the other hand, frequency sensitivity of human auditory system follows a logarithmic frequency scale (i.e., Mel, Bark). Using wavelet packets one can achieve a decomposition with an adjustable frequency resolution both for low and high frequencies. It is therefore possible to approximate the Mel-scale using wavelet packets. In this experiment we employ wavelet packets to achieve different frequency resolutions.

Setup : Wavelet packet decomposition is performed with Daubechies-8 wavelets for 5 scales. The features are wavelet coefficient magnitude averages at each scale with hierarchical time resolution [81]. Four different tree structures are used for the decomposition as shown in Figure 5.7. The dyadic tree (Figure 5.7.(a)) corresponds to classical dyadic wavelet transform. Next we test increasing the resolution around formant frequencies using the tree in Figure 5.7.(b).

Figure 5.7.(c) is an approximation of the mel-scale [47]. Finally we also use a full wavelet packet decomposition that yields equal resolution for all frequencies (see Figure 5.7.(d)).

WP tree structure	recognition rate(%)	number of features
dyadic	91.90	6
test	83.06	12
apprx. Mel scale	86.17	24
full	86.99	32

TAB. 5.5 – Recognition rates using different wavelet packet tree structures (%).

Results : The recognition results are summarized in Table 5.5. For each tree structure the number of parameters used are also given in the last column. Although the dyadic decomposition has the least number of parameters it leads to the best results.

Further tests showed that by decorrelating the coefficients of the mel-scale approximating tree structure we can attain a recognition rate of 95.09 %. This indicates that there is a high correlation between the coefficients and the diagonal covariance Gaussian models is not sufficient to model these features.

Noise robustness

Aim : We illustrate how to perform additive noise compensation with features in wavelet domain. The experiments are performed under zero mean white Gaussian noise.

Noise compensation in wavelet domain : Additive noise compensation in wavelet domain has is based on the following principles [23] :
- wavelet coefficients are conditionally Gaussian,
- wavelet transform is a linear transformation,
- the sum of independent Gaussian random variables is also a Gaussian random variable with variance being equal to the sum of the variances,

Denote the wavelet coefficient mixture variance of clean signal at scale j and time k as σ_{jk}^2. If the signal is contaminated by zero mean white Gaussian noise with variance σ_n^2 then the mixture variance of the noisy signal γ_{jk}^2 would be :

$$\gamma_{jk}^2 = \sigma_{jk}^2 + \sigma_n^2 \tag{5.18}$$

Hence, given a wavelet domain HMM of clean speech, adaptation is performed by increasing the variance of each mixture component by the estimated noise variance. The above method is theoretically verified when the coefficients are directly used in parameterization. Further theoretical analysis is required for the effect of parameterization techniques on the wavelet coefficients, i.e., when energy is used as a feature, the mean of the Gaussian is effected rather than the variance.

Setup : Wavelet decomposition is performed with Daubechies-8 wavelets for 5 scales. The features are wavelet coefficient magnitude averages at each scale with hierarchical time resolution [81].

# features	MFCC	$\sum \lvert w_j^k \rvert$	compensated features
clean	96.40	91.90	-
$\sigma_n^2 = 20$	93.29	31.51	91.82

TAB. 5.6 – Recognition rates under white Gaussian noise (%).

Results : The results are given in Table 5.6. Wavelet coefficients are not inherently robust to noise. The recognition performance degrades significantly. But a simple adaptation technique can easily be applied to compensate the effect of noise. In fact, wavelets are widely used in signal de-noising because of their time-frequency localization.

5.4.3 Multi-resolution cepstral features

Cepstral analysis provides an efficient means for handling convolutive interferences (i.e., the separation of the excitation and the vocal tract filter). However for additive interferences the cepstral domain is not appropriate. The DCT operation (due to insufficient time localization properties of Fourier analysis) distributes a frequency local component to the full cepstrum. Hence cepstral features have poor frequency localization properties. In [123], Vaseghi proposes multi-resolution cepstral features to complement the classical cepstral features. The principle of this technique is to compute cepstral features at different sub-bands i.e. resolutions, and to combine these with classical cepstral features.

In this experiment we propose a modified version of this technique considering the following inconveniences :

- The superposition of full-band and sub-band cepstra causes a redundancy in the parameter space. Therefore the resulting parameters are highly correlated.
- For the purposes of speech recognition we do not need the same localization property at all frequency bands, i.e., first three formant frequencies are more critical for phoneme classification.

In this respect, we redefine multi-resolution cepstral features as follows. First of all instead of an equal partitioning of the sub-bands we divide the spectrum into dyadic intervals such that we have a better localization at low frequencies. Next in computing the cesptra of each sub-band we use different number of coefficients. Such that some bands are represented with more detail than others.

We performed several experiments with different number of coefficients in each band under various noisy conditions. In Table 5.7 we summarize some of the results obtained. The best results are obtained when the second and third sub-bands are represented with more detail. For clean speech the performances are slightly inferior to MFCCs, but in noisy conditions when MFCCs fail this technique can achieve significant recognition rates.

5.5 Frequency Filtering

5.5.1 Frequency localized features

To address the problem of modeling frequency dynamics, one needs to work with speech features that are localized in the frequency domain and, preferably, that have a meaningful physical interpretation. The most widely used features in speech recognition systems are Mel-frequency cepstral coefficients (MFCC). These features are calculated by a Mel-scale filtering of the power spectrum followed by discrete cosine transform (DCT) of the logarithmic filter-bank energies.

noise (c=0.05)	12 MFCC	[154321]	[244321]	[054321]
clean	**96.40**	93.94	94.93	94.52
babble	53.19	60.64	68.49	**74.55**
buccaneer2	**25.70**	14.98	13.50	13.50
destroyerengine	32.90	**70.70**	68.58	64.24
f16	32.16	**47.79**	41.98	44.35
factory1	48.77	**67.02**	60.39	58.02
factory2	68.00	**74.71**	70.95	70.54
hfchannel	12.93	25.37	**52.45**	42.72
leopard	47.55	**52.54**	34.45	32.00
m109	79.87	79.62	84.45	**87.23**
machinegun	**70.79**	66.94	62.27	64.73
pink	72.50	86.66	**91.41**	91.33
volvo	78.89	69.15	**90.18**	86.58
white	40.18	93.04	**93.94**	85.92

TAB. 5.7 – Recognition rates using multi-resolution cepstral features. [154321] : denotes 1 cepstral coefficient for the 1st band, 5 cepstral coefficients for the second band, 4 coefficients for the third band etc. (%).

The DCT decorrelates the filter-bank energies at the cost of loosing frequency localization. Therefore, MFCC features have a poor physical interpretation and they lie in an artificial domain which do not allow for the analysis of frequency dynamics of speech. Nevertheless, MFCC-based parameterizations lead to the best recognition performances and it is difficult to compete with them. In general, when a new parameterization is proposed, it may improve performances with respect to MFCC, but only in clean or noisy conditions, rarely in both. Motivated by our desire to build time-frequency speech models, we wanted to investigate whether it was possible to design speech parameterizations which have the advantages of MFCC but not their disadvantages. By looking at the literature addressing the same problem, we were attracted by the frequency filtering (FF) technique developed by Nadeu et al [97, 99] because of it's simplicity and the reported results. The principle of this technique is to filter the log filter-bank energies by first or second order linear filters in order to decorrelate them. It has been reported [97, 103, 99, 95] that FF leads to recognition performances similar to or better than MFCC, in clean and artificial noise conditions. However, when we tested FF in real noise conditions (using Aurora 3 database [68]), we found some discrepancies. The scope of this section is to show that, by looking at the frequency filtering approach [97, 99] from a wavelet filtering perspective, it is in fact possible to design speech features that are localized in the frequency domain, and that perform at least as well as MFCC in *all* conditions.

To overcome the limitations imposed by MFCCs, Nadeu et al [97, 99] proposed a simple logarithmic filter-bank energy (logFBE) based parameterization which consists of a linear filtering of logFBE features that quasi-decorrelates them. The basic principle of this technique, called *frequency filtering* (FF), is to design a filter that would equalize the variance of the cepstral coefficients. Nadeu et al [97, 99] report that a simple first or second order FIR filter can achieve this goal. The transfer functions of these two filters, that we note FF1 and FF2 as in [99, 95], are respectively given by : $H_{FF1}(z) = 1 - z^{-1}$; $H_{FF2}(z) = z - z^{-1}$.

That is, if the logFBE sequence of a given frame is : $S = \{S(1), S(2), ..., S(Q)\}$ then the resulting filtered features are

$$S_{FF1} = \{S(1) - 0, S(2) - S(1), ..., S(Q) - S(Q-1)\},$$
$$S_{FF2} = \{S(2) - 0, S(3) - S(1), ..., 0 - S(Q-1)\}.$$

Here Q denotes the number of filter-banks used in the spectral representation. In the same spirit, Paliwal [103] designed different filters and reported that the one that achieves the best results is (we call it FF3) : $H_{FF3}(z) = 1 - z^{-2}$.

$$S_{FF3} = \{S(3) - S(1), ..., 0 - S(Q-1), 0 - S(Q)\}$$

Note that FF3 is just a shifted version of FF2. Besides decorrelating logFBEs, frequency filtering also has the effect of weighting different regions in the cepstrum. This liftering effect is another advantage of this technique as compared to DCT. A discussion on this aspect can be found in [99].

FF-based parameterizations have been evaluated on different databases in clean and artificial noise conditions [97, 103, 99, 95]. They have been shown to give equivalent or better performances as compared to MFCCs. Thus, the FF technique seems to be a serious candidate to address frequency localization problem. Here we recall that our primary objective is to have a frequency localized parameterization that perform equivalent (not even better) to MFCC, but in *all* conditions. Frequency filtering yields well localized features because of the small length of the filters used. Thus, what remains is to evaluate this FF technique in real world conditions. If the (good) behavior observed in clean and artificial noise conditions is again observed in real world conditions, then frequency filtering provides a solution to our problem. From this point of view, the Aurora 3 Speechdat-Car database provides a very good experimental setting to evaluate the performances of the FF technique. Indeed, this database was recorded in real noise and therefore includes several degradations that occur in real world conditions. Moreover, this database was designed especially to give the speech recognition community a test bed to evaluate "seriously" alternative parameterizations to MFCC.

5.5.2 Experiments : Frequency Filtering in Real Noise Conditions

Experimental Setup

The experiments are performed on the Aurora-3 German database. The database is divided into 3 subsets according to the mismatch between training and test sets : well match (WM), medium mismatch (MM) and high mismatch (HM). We report results for each of these subsets using different parameterizations.

The baseline parameterization is obtained using the ETSI SQL W1007 standard front-end [68]. This is a cepstral analysis scheme where 13 Mel-frequency cepstral coefficients are extracted from 23 Mel filter-bank energies. The analysis is performed on 25ms Hamming windowed frames with 10ms frame shift. A notch filtering is applied to compensate signal offset. Pre-emphasis filter with $a = 0.97$ is used. The Mel filter-bank has 23 frequency bands in the range from 64 Hz to half of the sampling frequency. The resulting feature vector consists of 12 cepstral coefficients (excluding c0) and logarithmic frame energy. The first and second time derivatives (Δ and $\Delta\Delta$) are also appended as dynamic features in all the experiments.

The recognition is performed using the HTK software. The digits are modeled as whole word HMMs with 16 emitting states per word and left-to-right non-skipping transition topologies. The output density of each state is modeled with a mixture of 3 Gaussians using diagonal covariance matrices. Two silence models are used : 1) 3 state silence HMM to model pauses before and after each utterance, 2) single state short pause HMM for modeling pauses between words.

	MFCC	FF1	FF2	FF3
WM	90.58	90.80	90.10	89.68
MM	79.06	79.80	79.21	78.62
HM	74.28	73.17	**64.80**	**63.78**

TAB. 5.8 – Word accuracies (%) using 12 Mel-scale filter-banks for FFs and 23 filter-banks for MFCCs. Total number of static features is 13.

Evaluations

It was reported in [97, 99] that the number of Mel-scale filter-banks has to be carefully chosen because it effects the performances of the FF technique (a discussion is given in [99] and we will come back to this point in the next section). Thus, in the experiments presented in this section, we evaluate FF-based parameterizations obtained by filtering 12 Mel filter-bank energies as the original authors do in [97, 99, 95]. We also incorporate the same energy feature as in MFCC parameterization, i.e. log frame energy. Obviously, this also leads to a fair comparison between MFCC and FF given that the features dimension is the same in both parameterizations.

Previous results on FF-based parameterizations report that this technique performs at least as good as MFCCs in clean an artificial noise conditions. The filters that perform the best were found to be FF1, FF2 and FF3 [97, 103, 99] and that they generally lead to similar recognition scores. We now evaluate the behavior of FF1, FF2 and FF3 in real world conditions. The recognition scores of FF-based parameterizations and the baseline MFCC parameterization are shown in Table 1. We observe that the recognition accuracy of FF1, FF2 and FF3 features are almost equivalent to MFCCs on well matched and medium mismatched conditions. So far, this is consistent with the previous results on FF-based parameterizations [97, 103, 95]. However, in high mismatched conditions only FF1 still performs similar to MFCCs and there is a significant degradation in the performance of FF2 and FF3 features. One may conclude that under real world noise conditions FF1 is superior to the other two filters. But this would be a rather fast conclusion because a natural question is : what is the inherent reason that causes a huge discrepancy between these similar derivative type filtering techniques ? Indeed, all these filters have been designed using the same principle, so what is the reason for this discrimination between FF1 and {FF2, FF3} ? In the following sections, we argue that a possible explanation can be given by looking at these filters from another perspective : *the wavelet transform*.

5.5.3 Experiments : Effect of number of filter-banks in FF

In this experiment we would like to see the effect of the number of filter-banks used in FF technique that specifies the resolution of the spectral representation. The experiments are carried out on Spanish database. We performed several tests using different number of filter-banks (12 to 26) with FF1 and FF2 frequency filters. The modeling and baseline MFCC parameterization is the same as the previous experiment.

In Figure 5.8 the comparison of FF1, FF2 for WM conditions are given. The baseline result obtained with W1007 standard MFCC parametrization on 23 filter-banks is also indicated. Even though the performance of frequency filtering tends to increase with higher number of filter-banks, it is always inferior to MFCCs for this well-matched condition. Similar comparisons are given for MM and HM conditions in Figure 5.9 and Figure 5.10 respectively. We observe that as the number of filter-banks is increased the performance improves with respect to MFCCs.

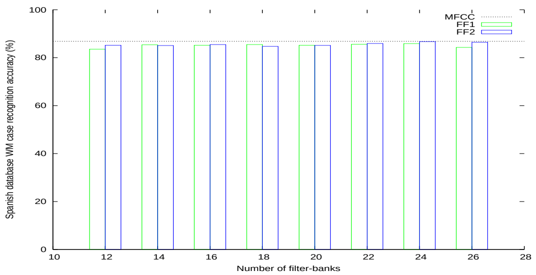


FIG. 5.8 – Frequency filtering with different number of filterbanks on Spanish database WM conditions compared with W1007 standard parameterization.

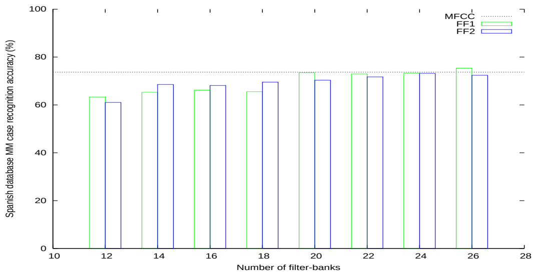


FIG. 5.9 – Frequency filtering with different number of filterbanks on Spanish database MM conditions compared with W1007 standard parameterization.

Increasing the number of filter-banks also increases the number of features used in modeling. Normally one would expect a corresponding increase in the recognition performance. Although this is observed in mismatched conditions, in matched conditions this is not the case. This can be related to frequency resolution of the spectral representation and the effect of noise on different frequency bands. Increasing the resolution allows modeling of smaller band intervals. Since FF technique provides frequency local features, the corrupted bands effect only some of the features. Increased resolution in mismatched cases separates the contaminated and clean bands which in turn reduce the effect of local noise on the global likelihood of a given frame.

5.6 Wavelet Filtering

5.6.1 Frequency Filtering and Wavelet transform

Our starting point is to notice that FF1 is nothing but applying the simplest orthogonal wavelet transform : the Haar wavelet transform [93]. Indeed, the transfer function of the high pass filter of the Haar wavelet is : $H_{Haar}(z) = \frac{1}{\sqrt{2}} H_{FF1}(z)$

Thus, given that wavelets are known for their decorrelation properties, a certain decorrelation is achieved when using FF1. Similarly, FF2 and FF3 are nothing but applying the Haar transform

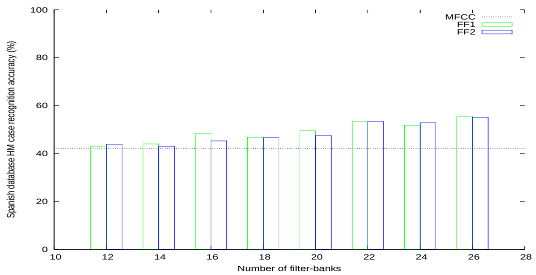


FIG. 5.10 – Frequency filtering with different number of filterbanks on Spanish database HM conditions compared with W1007 standard parameterization.

to the odd and even sequences of the input sequence (up to a shift factor for FF3). If the odd and even sequences are viewed as approximations of the input sequence, then FF2 and FF3 can be seen as an erroneous way to implement the dyadic Haar decomposition. Indeed, to implement the latter correctly, the approximations should be computed using the low pass filter of the Haar wavelet. Thus, from this point of view, the discrepancy of FF2 and FF3 is probably due to this erroneous implementation. We emphasize here that we are completely ignoring the motivations and the analysis given by the original authors and we are looking at these filters from a different angle. We now proceed to check whether our reasoning makes sense.

Let's assume that the fundamental reason behind the relatively good performances of FF1 is the decorrelation property of the corresponding wavelet (the Haar wavelet). It is well known that decorrelation generally increases with the number of vanishing moments of the analyzing wavelet. Therefore (if our reasoning is correct), if we replace the Haar filter (FF1) with a wavelet with a higher number of vanishing moments, we should get at least similar performances as FF1. We must however be careful on the choice of the degree of the corresponding filter (or equivalently the length of the corresponding FIR). Indeed, choosing a highly regular wavelet would not only lead to boundary artifacts, but also (and more importantly) it would loosen the frequency localization of the resulting features. The wavelet which fulfills the desired requirements is the Daubechies 4-tap filter (Daub4). This wavelet belongs to the same family as the Haar wavelet (actually, the Haar wavelet is Daubechies 2-tap filter), it has 2 vanishing moments while the Haar wavelet has only 1, and the length of it's FIR is 4. The transfer function of the high pass filter of Daub4 is : $H_{Daub4}(z) = 0.129z + 0.224 - 0.836z^{-1} + 0.483z^{-2}$.

The recognition scores obtained with the Daub4 filter are shown in Table 2. If we do not consider the medium mismatch case, our reasoning is verified since we get a similar performance in well match case and a significantly improved performance in high mismatch case.

However, we have a rather disturbing discrepancy in medium mismatch case because it is inconsistent with how a parameterization should behave. One would expect that the recognition performance degrades as the mismatch between training and test data increases. In our experiments the performance of the recognizer for the high mismatch case is higher than the performance in medium mismatch. After spending a lot of time to understand this behavior, we realized that it is mainly due to the Speechdat-Car-German database. Indeed, a lot of people working on this database finds the same discrepancy as reported by several papers in the special session "Aurora : Speech Recognition in Noise" in ICSLP'2002 (see [92] for instance). Therefore, the scores in MM should not be given a lot of credit. In summary, these experiments suggest that,

	MFCC	Haar (= FF1)	Daub4
WM	90.58	90.80	90.30
MM	79.06	79.80	76.94
HM	74.28	73.17	78.21

TAB. 5.9 – Word accuracies (%) using FF of 12 Mel-scale filter-banks with Haar and Daub4 wavelets. Total number of static features is 13.

looking at FF from a wavelet filtering perspective provides an explanation for the discrepancy of FF2 and FF3 and the good behavior of FF1. Moreover, this perspective can lead to significant improvement in the recognition accuracy in the HM case.

At this point we can conclude that, putting the frequency filtering in a wavelet filtering perspective *seems* to provide a solution to our problem. That is, depending on the desired frequency localization of the spectral features, one can choose the appropriate wavelet with the appropriate number of vanishing moments to decorrelate the logFBEs. Of course, more experiments and evaluations need to be done in order to have a precise idea on the behavior of wavelet filters.

5.6.2 Multi-resolution Frequency Filtering

In the previous section we proposed a formal perspective to tune the decorrelation and localization properties of frequency filtering. Another aspect that influences the localization property of the resulting parameterization is the resolution of the initial spectral representation. As reported in [99], the choice of number of filter-banks (Q) that is used to compute logFBEs is a critical aspect in frequency filtering. This parameter specifies the resolution of the initial spectral representation. For high Q, logFBEs will include many details that may not carry useful information for discrimination. On the other hand if Q is too small there may not be sufficient resolution in the spectral representation [99]. In [98, 95], Q is tuned to achieve the spectral resolution that yields the best recognition performance. In this section we would like to investigate if it is possible to avoid this tuning. Since we have paraphrased frequency filtering in a wavelet filtering perspective, a way to do so is to apply a multi-resolution wavelet decomposition and evaluate the performance of frequency filtering at multiple resolutions. We note here that multi-resolution wavelet analysis is the usual way wavelets are used in speech parameterization as an alternative to the DCT operation [61, 110, 109].

The wavelet decomposition of a signal $s_0 = \{S_0[k]\}$ at resolution j ($j > 0$) is recursively obtained by :

$$D_{j+1}[n] = \sum_k h[k-2n]S_j[k], \;\; S_{j+1}[n] = \sum_k l[k-2n]S_j[k]$$

where $h[n]$ and $l[n]$ are the high-pass and low-pass decomposition filters. $d_j = \{D_j[k]\}$ and $s_j = \{S_j[k]\}$ are the detail and scaling coefficients respectively. The properties of the wavelet basis is specified by the decomposition filters $h[n]$ and $l[n]$. The reader is referred to [93] for further details on wavelet analysis. In terms of frequency filtering, d_1 is the decimated frequency filtered sequence where $h[n]$ is the frequency filter.

In our experiments we use 23 filter-banks as specified by the ETSI SQL W1007 front-end standardization for the Aurora database. This selection specifies an initial resolution that is found to be optimal for this database [68]. Moreover, we would like to start with an initial resolution

	(d_1) (13)	(d_1+d_2) (19)	$(d_1+d_2+d_3)$ (21)	(d_2+d_3) (10)
WM	83.01	88.14	91.08	89.04
MM	64.42	76.57	78.77	71.89
HM	71.88	77.75	78.82	74.65

TAB. 5.10 – Word accuracies (%) using DWT on 23 Mel-scale filter-banks with Haar wavelet. The size of the feature vector for each case is indicated inside the parenthesis.

	(d_1) (13)	(d_1+d_2) (19)	$(d_1+d_2+d_3)$ (21)	(d_2+d_3) (10)
WM	74.19	85.25	89.40	88.56
MM	58.49	71.96	77.23	75.26
HM	62.86	78.26	81.17	76.83

TAB. 5.11 – Word accuracies (%) using DWT on 23 Mel-scale filter-banks with Daub4 wavelet. The size of the feature vector for each case is indicated inside the parenthesis.

that is detailed enough to allow a range of resolutions to analyze. We conduct experiments using 3-level ($j = 1, 2, 3$) wavelet decompositions using the Haar and Daub4 wavelets. The results are shown in Tables 3 and 4, respectively. In the first columns we report the results when only d_1 (which denote the finest level detail coefficients) is used as static features. In this case the number of static parameters is 13 (the same as MFCCs). We observe that the recognition performance at the first decomposition level is inferior to MFCC and FF techniques previously reported in Tables 1 and 2. The main reason for this degradation is the initial resolution selection which is optimal for MFCCs but not for frequency filtering. In the second columns we compute the detail coefficients at the second level and construct an appended feature vector to see the contribution of the coefficients at the second level. Although the number of parameters is only slightly increased (13 to 19) there is a considerable improvement in the recognition performance. Going one step further, we append the coefficients at the third decomposition level. This case is reported in the third columns of Table 3 and 4. We observe that the performance is comparable to and even (especially in high mismatch conditions) better than MFCC and fixed resolution FF parameterizations. From these results we can conclude that frequency filtering at different resolutions have complementary discriminative information. With appropriate integration of information from different resolutions we can achieve performance improvements. However we should note that this improvement comes with the cost of increasing the number of parameters, and also with a loss in frequency localization as compared to fixed resolution FF.

Finally in the last columns we discard the first level detail coefficients (d_1), i.e. the highest resolution, and use the combination of lower resolution coefficients ($d_2 + d_3$). By doing so we reduce the number of coefficients to 10 and also eliminate the highest resolution coefficients which may include non-discriminative information as reported in [99]. With the decreased number of parameters, the recognition performance is still remarkable. These results confirm that the highest resolution coefficients are not as discriminative as lower resolutions. We note that a similar approach has been proposed in [61] where a spline wavelet is used for decomposition.

In summary, these experiments suggest that a way to avoid tuning Q is to use a full wavelet decomposition of logFBEs. Then, if there is a desire to reduce the computational complexity, fine resolution detail coefficients could be removed.

5.7 Experiments on full Aurora 3 dataset

In previous sections we used a subset of the Aurora database. In this section we present the results of frequency and wavelet filtering techniques on Spanish, Danish and Finnish datasets. One of our motivations to use wavelets in frequency filtering was the degraded performance of FF2 and FF3 under high mismatch conditions.

For Spanish database none of the filtering techniques improve the performance as compared to MFCCs except for FF2 filter in HM conditions as shown in Table 5.12. Among the frequency filters the best one appears to be FF2 for WM and HM cases. The best filter for MM case is FF3.

The results for Danish database are given in Table 5.13. All filtering techniques gives similar scores to MFCCs. Among the filtering techniques the best one appears to be FF3 for WM and MM and FF1 for HM.

The most reliable database is the Finnish database that is the biggest. The results are shown in Table 5.14. FF2 appears to be the best filter for MM and HM cases with a performance far ahead of the others. For WM case however, the best filter appears to be Daub4. The most significant improvement is for HM case with FF2.

These results does not approve a global improvement with respect to MFCCs. Some significant improvements are observed in certain cases but these cannot be generalized.

	MFCC	FF1	FF2	FF3	Daub4
WM	**86.87**	83.59	85.23	83.53	85.02
MM	**73.74**	63.35	61.06	66.61	63.26
HM	42.23	43.10	**43.94**	37.65	43.49

Tab. 5.12 – Spanish database word accuracies (%) using W1007 STQ standart frontend using 23 FB and several frequency filters on 12 logFBEs.

	MFCC	FF1	FF2	FF3	Daub4
WM	**79.61**	78.26	78.89	79.03	79.01
MM	49.44	44.63	47.74	**50.28**	49.58
HM	**33.15**	32.01	30.05	28.02	28.02

Tab. 5.13 – Danish database word accuracies (%) using W1007 STQ standart frontend using 23 FBs and several frequency filteres on 12 logFBEs.

	MFCC	FF1	FF2	FF3	Daub4
WM	90.39	88.96	89.48	89.70	**92.14**
MM	72.37	67.24	**74.42**	60.88	61.42
HM	31.06	35.80	**55.30**	44.13	39.72

Tab. 5.14 – Finnish database word accuracies (%) using W1007 STQ standart frontend using 23 FBs and several frequency filteres on 12 logFBEs.

5.8 Conclusion and perspectives

In this chapter, we addressed the problem of designing frequency-localized speech features that can perform at least as well as MFCC in all conditions. The recently introduced frequency filtering technique has been shown to solve this problem in clean and artificial noise conditions. But when we evaluated it in real-noise conditions using Aurora 3 German database, we found some discrepancies. By looking at frequency filtering from a wavelet filtering perspective, we were able to explain the FF discrepancies and to achieve significant improvements in recognition accuracy in the highly mismatch case. However, at this point we can not draw any decisive conclusions because more experiments need to be done. But we can say that the framework of the frequency and wavelet filtering approaches has definitely a huge potential to provide a solution to this problem, and that more investigations should be conducted in this framework.

Although the use of wavelet analysis in ASR has found an interest in the community, significant improvement achievements has not been attained. One of the main reasons for this is the processing blocks and modeling techniques used. First of all frame by frame analysis is not appropriate for wavelet based techniques since wavelets already provide an adjustable time localization. Another inconvenience arises in the modeling stage. HMMs are developed for conditionally independent features at each time frame, however the statistical properties of wavelet coefficients does not fit to this hypothesis. The coefficients are correlated in time and in frequency. Statistical techniques for modeling wavelet coefficients has been proposed in various domains [23, 76]. An interesting model for wavelet coefficients called hidden Markov wavelet trees (HMT) has been proposed in [23]. We have initiated a DEA project to use these models with appropriate wavelet based parameterizations [59]. The learning and inference machinery to use such models for speech recognition is given in Appendix E

Chapitre 6

Predictive Compensation for Noise Robustness

Sommaire

6.1 Introduction **127**
6.2 Contributions **128**
6.3 Techniques for noise robustness **128**
6.3.1 Adaptive model compensation 129
6.3.2 Predictive model compensation 130
6.4 Supervised-predictive compensation **130**
6.4.1 Simple linear regression modeling 132
6.5 Model Parameter Selection **132**
6.5.1 Noise model 133
6.5.2 Speech model 135
6.6 Experiments **137**
6.6.1 Experimental setting 137
6.7 Results and discussion **138**
6.8 Conclusion **141**

6.1 Introduction

In real world applications, noise robustness of speech recognition systems is an important issue. Generally, models that are well matched to initial training conditions perform poorly in a new application environment. This is due to several noise sources that affect the speech signal in communication channel. In a general framework these distortions can be represented with the following formulation :

$$y(t) = \left[\left\{\left(\left[s(t)|^{Stress}_{Lombard}\right]_{n_1(t)} + n_1(t)\right) * h_{mike}(t) + n_2(t)\right\} * h_{chan}(t)\right] + n_3(t)$$

where $n_1(t)$ is the background noise which also induces a nonlinear distortion on the signal due speaker stress and Lombard effect [75]. $h_{mike}(t)$ is the impulse response of the recording microphone. $n_2(t)$ and $h_{chan}(t)$ are the additive and convolutive noise distortions of the communication channel, and $n_3(t)$ is the noise at the receiver. All these distortions alter the statistical properties of the speech signal. Furthermore, combined with the pre-processing in feature extraction stage

the overall affect on the recognition system is highly non-linear. This causes a mismatch between training condition models and speech observations in a new environment.

The goal of noise compensation in ASR is to eliminate the mismatch between training and application conditions. This problem can be addressed in three basic ways [60] :

- Pre-processing of speech signal to filter out the effects of environment conditions.
- Using speech features that are implicitly noise resistant.
- Model compensation, i.e., transformation of training condition models to accommodate the new environment.

These techniques are complimentary to each other, and in a practical system all these techniques are applied sequentially to achieve good performances. In this chapter we focus on model compensation and develop a new supervised predictive compensation scheme. The principle of this new scheme is to learn a mapping between noise and matched speech model parameters using a dataset of matched models for various noisy conditions. The noise and matched models are used respectively as inputs and outputs in a supervised learning procedure to estimate the parameters of a mapping that attempts to describe how matched models vary w.r.t. noise models. In recognition, noise observations in the application environment are used to estimate the corresponding noise model. The learned mapping is then used to estimate speech models based on the noise model.

The chapter is organized as follows. In Section 6.3 we briefly overview model based compensation techniques for noise robustness. Next in Section 6.4 we describe the principles of a new supervised-predictive compensation technique. In Section 6.5 we present preliminary experimentations that guide us to specify the parameter space that should be used for this technique. Finally in Section 6.6 we report the experimental results on a connected digits recognition task under various noise conditions using the NOISE-EX database and follow-up with discussion and conclusions.

6.2 Contributions

- A new noise compensation scheme in model parameter space : *supervised-predictive compensation.*
- An analysis of noise model parameter selection in supervised-predictive compensation.
- Evaluation of the proposed compensation scheme under various noise conditions using NOISE-EX database.

6.3 Techniques for noise robustness

Many techniques [60] have been proposed to handle the difficult problem of mismatch between training and application conditions of a speech recognition system. These can be classified into two categories : signal processing techniques and model compensation schemes. In the former, the input speech signal is processed in such a way to eventually achieve noise robustness (RASTA [67] or CMN for instance) or to denoise the signal (speech enhancement [45] for instance). In the latter the processing is performed on the statistical model of speech. Initial canonical models (generally clean speech models) are transformed to match the new environment. These techniques can be categorized into two classes : *adaptive* and *predictive*. In adaptive techniques initial models are refined using some adaptation data recorded in the new environment. In predictive techniques, first a model for environment/noise is estimated and then this model is combined with clean speech models based on a hypothesized mismatch function that describes how noise alters speech.

6.3.1 Adaptive model compensation

The principle of these techniques is to use new speech data recorded in noisy conditions to transform initial models to match the noisy environment. Adaptation is performed either by directly re-estimating the parameters using adaptation data or indirectly estimating an adaptation function that will later be used to transform the model parameters.

Maximum Likelihood Linear Regression (MLLR) : MLLR is one of the most widely used model compensation schemes [85]. It is an indirect approach such that adaptation is performed using a linear transformation of initial model parameters.

$$\hat{\lambda} = f_\theta(\lambda) \quad (6.1)$$
$$= \mathbf{A}\lambda + B \quad (6.2)$$

where λ denotes the initial speech model parameters and $\hat{\lambda}$ denotes the adapted model parameters. Traditionally, the transformation parameters $\theta = \{\mathbf{A}, B\}$ are learned such that the likelihood of the adaptation data is maximized :

$$\hat{\theta} = \arg\max_\theta P(D_{adapt}|f_\theta(\lambda)).$$

It is however reasonable to assume that some values of the transformation parameters are more likely that others. Additional information about the transformation parameters can be introduced in the form of a prior distribution, which is then used to constraint the estimation via the use of a MAP estimation criterion, rather than maximum likelihood :

$$\hat{\theta} = \arg\max_\theta P(D_{adapt}|f_\theta(\lambda))P(\theta).$$

This technique is called MAPLR (maximum a posteriori linear regression) [18].

Further improvements on the parameter estimation can be achieved using a Bayesian approach, i.e. (BPA) Bayesian predictive adaptation, such that rather than using fixed ML or MAP estimates the distribution of the parameters $P(\theta|D_{adapt})$ can be learned [114]. Next the speech model for the new environment is obtained as :

$$P(D|\lambda, D_{adapt}) = \int_\theta P(D|\lambda,\theta)P(\theta|D_{adapt}).$$

Obviously this scheme is computationally much more demanding and the choice of the form of the prior densities is a critical problem to be addressed.

Maximum A Posteriori (MAP) adaptation : In MAP adaptation it is assumed that there is some prior knowledge on speech model parameters in the form of a prior distribution [55]. The adaptation is performed by directly estimating the speech model parameters that maximize the a posteriori distribution of the adaptation data :

$$\hat{\lambda} = \arg\max_\lambda \int_\lambda P(D_{adapt}|\lambda)P(\lambda)$$

A fundamental property of MAP adaptation is its nice asymptotic convergence to maximum likelihood estimation when the amount of adaptation data increases. MAP estimation is favorable when adaptation data is insufficient.

One inconvenience of adaptive schemes is that they require at least few dozens of seconds of speech to yield relatively good estimates of the new models. A more important inconvenience is the need of speech data transcription in an unsupervised mode, this is done generally using the initial models which limits the performances if the transcription accuracy is poor.

6.3.2 Predictive model compensation

Predictive model compensation techniques estimate matched speech models for a new environment without using any data in the new environment. This is achieved by using models of noise sources in the environment and a mismatch function that describes how these noises affect the speech observations [54]. The basic formulation is :

$$\hat{\lambda} = f_{mismatch}(\lambda, \lambda_n).$$

where λ and λ_n denote clean speech and noise model parameters respectively. The form of the mismatch function depends on the noise and speech model feature space.

Parallel Model Combination (PMC) : In its basic form PMC approximates the mismatch function in log spectral domain for additive noise as :

$$\hat{\lambda} = E\{log\big(exp(s) + exp(n)\big)\}$$

where s and n are the corresponding spectral components of speech and noise. In [53], Gales considers several approximations of this expression to simplify the mismatch function in spectral and cepstral domains for means and variances of the model.

An inconvenience of predictive schemes is the assumption made about the effects of the new acoustic environment on the clean speech, i.e., how the noise sources alter the clean speech signal (additive or/and convolutional...). This assumption may be unrealistic which consequently may lead to erroneous estimations of corrupted speech distributions. An other inconvenience of predictive schemes is their strong dependence on the front-end (MFCC in general) and the probabilistic models (HMMs in general) which are used.

It is important to emphasize the complementarity of adaptive and predictive compensation schemes particularly when the observed speech is corrupted by background noise. Indeed, in this case noise observations can be used in a predictive scheme to provide a "relatively good" initial model for an adaptive scheme that will use corrupted speech data to build the final compensated model. However, besides the inconveniences we mentioned above, an important limitation of predictive compensation schemes is the fact that they rely on the availability of a given canonical model (generally the clean speech model), thus implicitly assuming that training data has been recorded in one specific acoustic environment. Nowadays however, it is more common to have at hand speech databases that are recorded in many different acoustic environments. For instance, for car-embedded speech recognition systems, training data are recorded using different cars, with windows open or closed, with radio switched on or off...etc. In these scenarios, predictive compensation is not the best way to handle the mismatch between training and application conditions. Instead, the most widely used technique in these scenarios and which has been shown to give best performances is *multi-conditions* recognition. That is, the probabilistic models are trained using all the available speech data, i.e., in all the recording conditions, then the resulting models (eventually modified in an adaptive scheme) are used in the new environment.

6.4 Supervised-predictive compensation

In this section we consider that we are given databases of speech recorded under different background noise conditions. We then put the foundations of a new predictive compensation scheme, that we call *supervised-predictive* compensation, capable of handling these scenarios. The principle of this new scheme is the following. For each noise condition, we first build the speech models set using the available training speech data in this condition. These models will represent speech in this condition, we thus call them *matched* models. Then we use corresponding noise observations (from the training database) to build a probabilistic model that attempts to represent the noise distribution, we call it *noise* model. After processing all noise conditions, the resulting noise and matched models are used respectively as inputs and outputs in a supervised learning procedure to estimate the parameters of an hypothesized (parametric) model that attempts to describe how matched models vary w.r.t. noise models. In recognition, noise observations in the application environment are used to estimate the corresponding noise model, the latter will be then given as input to the hypothesized model to provide speech models that we hope will be close to the matched models in this environment.

We consider our new compensation scheme as belonging to the class of predictive techniques because both share two fundamental characteristics. First, for both, the basic (realistic) assumption is that the best recognition performances are achieved when there is no mismatch between training and application conditions. Second, both rely on models describing the noise sources to perform compensation, and *no* speech data from the new acoustic environment is used. We now proceed to develop our supervised-predictive compensation scheme. We assume that we are given a set $D_1, ..., D_K$ of speech databases recorded under background noise condition $n_1, ..., n_K$ respectively. Let W be the set of acoustic units (words, phones,...) we try to model. For each $w \in W$ and $k \in \{1, ..., K\}$, we denote by $\lambda_w(k)$ the (matched) acoustic probabilistic model of w trained using database D_k, i.e., under noise condition n_k. If N denotes a noise variable, the main principle of our scheme is to assume that, for each $w \in W$, there exists a parametric function f_w such that :

$$||\lambda_w(N) - f_w(N)|| \cong 0, \quad \text{for some given norm} \quad ||.||.$$

The set of input-output observations $\{n_k, \lambda_w(k) : k = 1, ..., K\}$ will be then used in a supervised learning procedure to estimate the parameters of f_w. In recognition, a new noise condition n^* is observed and is given as input to all f_w hoping that the resulting models $\tilde{\lambda}_w(n^*) = f_w(n^*)$, that we call *f-estimated* models, will be "close enough" to the true matched models $\lambda_w(n^*)$ in noise condition n^*. The parametric form of the functionals f_w, which can be random or deterministic, is the central hypothesis of our scheme and is defined by the user depending on his prior knowledge, belief, computational constraints and the application specificities.

We now stretch the numerous advantages of our supervised-predictive compensation scheme :

- It makes no assumption on how the background noise source alters the speech signal.
- It is completely independent of the front-end and the probabilistic acoustic models used to process and model speech.
- It does not rely on the availability of an initial canonical speech model. Only the parameters of f_w are required to estimate the corrupted speech models in the new environment.
- As classical predictive schemes, it is computationally efficient.
- It can be easily "incremented" or "updated". That is, if a new training database D_{K+1} (in a new environment n_{K+1}) is available, one has only to keep $\{n_k, \lambda_w(k) : k = 1, ..., K\}$ in memory to take into account this new information (or observation) and update the estimation of f_w. This is a major advantage w.r.t. multi-conditions training for instance. Indeed, in order to take into account D_{K+1}, the latter requires to keep in memory all $\{D_1, ..., D_K\}$ and retrain using all $\{D_1, ..., D_K, D_{K+1}\}$. This is obviously very demanding in terms of memory requirements and computational processing.

6.4.1 Simple linear regression modeling

Probably the simplest parametric form of f_w one can think of is the linear one. In this case we consider that the speech parameter $\lambda_w(k)$ in environment k can be represented as a linear function of noise parameter vector $\mathbf{n}_k$ such that :

$$\lambda_w(k) = \mathbf{n}_k'\mathbf{a} + b$$

where $\mathbf{a} = [a_1, a_2, \ldots, a_L]'$ is a vector of dimension L and b is the bias term.

Now to get rid of the bias term and simplify the notation, define $\hat{\mathbf{a}} = [\hat{\mathbf{a}}'\ b]'$ and $\hat{\mathbf{n}}_k = [\mathbf{n}_k'\ 1]'$. Then the model is reformulated as :

$$\lambda_w(k) \quad = \quad \hat{\mathbf{n}}_k'\hat{\mathbf{a}} \tag{6.3}$$

For training this model we use K pairs of $(\lambda_w(k), \hat{\mathbf{n}}_k)$ from K different environments. So we have K linear equations with $L+1$ variables to be solved simultaneously.

$$\begin{bmatrix} \lambda_w(1) \\ \lambda_w(2) \\ \vdots \\ \lambda_w(K) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{n}}_1' \\ \hat{\mathbf{n}}_2' \\ \vdots \\ \hat{\mathbf{n}}_K' \end{bmatrix} \hat{\mathbf{a}} \tag{6.4}$$

$$\Lambda = \mathbf{M}\hat{\mathbf{a}} \tag{6.5}$$

We are looking for $\hat{\mathbf{a}}$ that satisfies all K equations simultaneously. If $K < L+1$ there are infinitely many solutions. If $K = L+1$ and $\mathbf{M}$ is reversible there is a unique solution. If $K > L+1$ there are no solutions. In this case an approximate solution can be obtained using least squares fitting. For $K = L+1$, the least squares solution is the unique solution of the linear system :

$$\hat{\mathbf{a}} = \mathbf{M}^{-1}\Lambda.$$

Least squares fitting : Find $\tilde{\mathbf{a}}$ that minimizes the mean squared error :

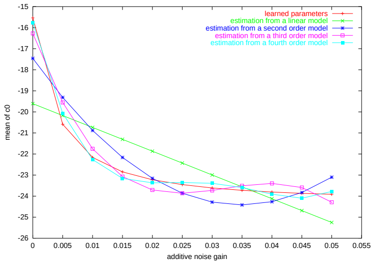
$$\tilde{\mathbf{a}} = \arg\min_{\hat{\mathbf{a}}} \sum_{k=1}^{K} |\lambda_w(k) - \hat{\mathbf{n}}_k'\hat{\mathbf{a}}|^2 \tag{6.6}$$

The minimization yields :

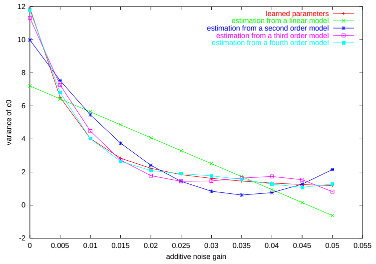
$$\tilde{\mathbf{a}} = \Big[\sum_{k=1}^{K} \lambda_w(k)\hat{\mathbf{n}}_k'\Big]\Big[\sum_{k=1}^{K} \hat{\mathbf{n}}_k\hat{\mathbf{n}}_k'\Big]^{-1} \tag{6.7}$$

6.5 Model Parameter Selection

The question that we address in this section is : what are the parameters of speech and noise that should be used in the compensation scheme ? In an HMM based system the speech parameters are the means, variances and the mixture weights of the HMMs. These constitute the target parameters to be estimated. In order to specify the parameters for the noise we first present a series of experiments. Next we analyze the sensitivity of the estimated parameters and decide which parameters of speech should be estimated and which chould be retained from clean conditions.



(a)



(b)

FIG. 6.1 – Model fit to the variation of (a) mean of c1 (b) variance of c1

6.5.1 Noise model

Gain (SNR)

As a first step we consider the case where the variation in the environment is limited to noise amplitude. The corrupted speech signal is given as :

$$y(t) = s(t) + \alpha n(t)$$

We want to observe the variation of speech parameters (i.e., means and variances) with respect to noise and try to fit a model in which noise is represented with α.

To model the variation of the mean and the variance of the noisy cepstra we setup a simple experiment. We use isolated utterances of digit "ONE" in the TiDigits database. 12 MFCCs are used for the parameterization of speech waveforms. 4 state single Gaussian HMMs are trained for different levels of additive white noise. This noise is added to each test signal with a multiplicative coefficient $0 \leq \alpha \leq 0.05$. Speech models are then trained for each value of α. These models constitute the training database for the environment model.

The cepstrum of corrupted speech take the following parametric form :

$$Ceps_y = DCT\ log\big(|S(f)|^2 + \alpha^2|N(f)|^2 + 2\alpha|S(f)||N(f)|\cos(\theta)\big) \tag{6.8}$$

where $S(f)$ and $N(f)$ are the speech and noise spectra. If we assume that speech and noise spectra are constant within each state mixture of the HMM, then noisy speech cesptra can be considered as a logarithmic function of a second order polynomial of α.

We trained individual models for the mean and variance of each speech model Gaussian of the form :

$$\lambda_w(k) = \mathbf{n}'_k\mathbf{a} + b$$

where $\mathbf{n}'_k = [\alpha^p \alpha^{p-1} \ldots \alpha]$. This corresponds to fitting a polynomial to the change of cepstra. We tested polynomial orders of p up to 4.

In Figure 6.1 we plot the variation of the mean and variance of the 1st cepstral coefficient as a function of α obtained from different model orders. The variation of the mean and the variance have a exponential decay as α is increased. The fourth order model yields a relatively good fit this decay. Similar results are observed for other cepstral coefficients.

Gaussian density

In the previous scenario the environmental variation was controlled only by noise level, therefore using the noise gain as the model parameter is reasonable. If the environment variation is also due to type of noise, a more detailed specification for the environmental change is needed. A simple solution is to train a statistical model for noise. Then the compensation model can be constructed by conditioning the speech HMM parameters on the noise model parameters. In this case the question to address is : which parameters of the noise model should be used, means, variances or both ?

We use 13 different noise types from NOISE-X database to simulate different environments. Matched speech models are trained for each case. For the noise model we train a single Gaussian on the silence segments of each utterance. The training is performed by embedded reestimation assuming that each utterance starts and ends with silence. Let μ and σ denote the mean and the diagonal variance vectors of the noise model.

We train individual models for the mean and variance of each Gaussian density in the speech model as :

$$\lambda_w(k) = \mathbf{n}'_k\mathbf{a} + b$$

where $\mathbf{n}'_k$ is either μ or σ. To compare the two cases we compute the average KL divergence between the estimated models and the matched models. [13]

As a measure of distance we compute the divergence of for each state and take the average. The comparison of these two cases are given in Figure 6.2. The average KL distance is found to be greater for most noise types when the variance vector is used for estimation. Conditioning on means yields better estimations as compared to variances.

Mixture of Gaussian density

In this case we refine the noise model by using a mixture of 6 Gaussian densities. In this case, the noise parameter vector is chosen to be the mean in each mixture component, i.e each mean and variance of the ith cepstral coefficient for all models, states and mixtures are conditioned on the means of mixtures of the ith cepstral coefficient. Hence the environment model is trained as :

$$\lambda^i_w(k) = \mathbf{n}'_k\mathbf{a} + b$$

where $\mathbf{n}'_k = [\mu^i_1, \mu^i_2, \ldots, \mu^i_6]$ where μ^i_j is the mean value for the ith cepstral coefficient in the jth mixture of the noise model.

An example of the estimation results is given in Figure 6.3 (a) and (b). Neither the mean nor the variance could be estimated correctly. Moreover variance estimates takes negative values in some cases. This model is insufficient for accurate modeling. One reason for this is that the variation of the elements of the noise vector takes similar values for each mixture component. These are shown in Figure 6.3 (c). There is a lot of redundant information in the noise vector. The same information could be represented with a 1 dimensional vector rather than 6.

[13] The Kullback-Leibler divergence (KL) is regarded as a measure of the extent to which two probability density functions agree and is expressed as :

$$KL = \int p(x) ln \frac{p(x)}{q(x)}$$

For two Gaussian densities $p(x) \approx \mathcal{N}(\mu_1, \sigma_1^2)$ and $q(x) \approx \mathcal{N}(\mu_2, \sigma_2^2)$ divergence is computed as :

$$KL = \frac{1}{2}\Big[ln(\frac{\sigma_2^2}{\sigma_1^2}) + \frac{(\mu_1 - \mu_2)^2}{\sigma_2^2} + (\frac{\sigma_1^2}{\sigma_2^2} - 1)\Big]$$

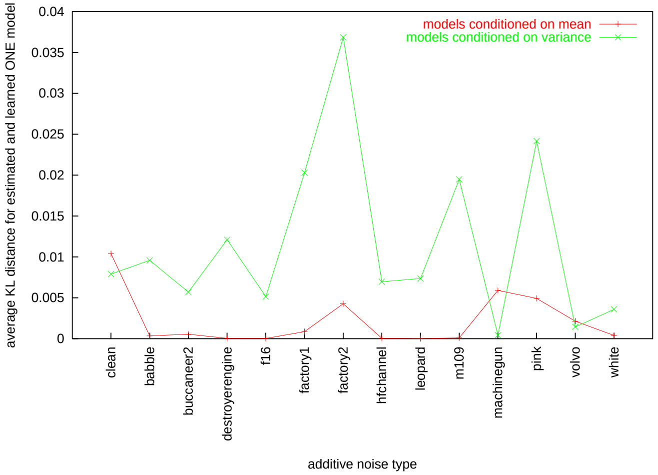


FIG. 6.2 – Comparison of average KL distances between learned and estimated model parameters using noise model mean and variance vectors.

Conclusion

As a result of these observations it seems that in our experimental setting, cepstral mean vector of a single Gaussian noise model is an appropriate representation to model the environment variation. In the following we use this noise representation. We note however that further analysis is required for a better representation in a noise space where different noise types are well separated.

6.5.2 Speech model

The speech model is parameterized with means, variances and mixture coefficient weights. In a general scenario the compensation should be applied to all these parameters. However, for the simple linear regression model there are some special considerations that should be taken into account. Namely the non-negativity constraints on the variances and mixture coefficient weights.

- mixture coefficients normalization : This problem is due to the fact that each parameter is estimated separately. The sum-to-one constraint of the mixture coefficients is not imposed in the estimation algorithm. Either an additional constraint to guaranty this property should be added, or a post-processing should be performed for normalization.
- negative variance values : Since the estimation is based on a linear model the estimates can take negative values. To prevent this the estimation can be formulated for standard deviation.

In this section we perform a simple experiment on Tidigits connected recognition task to analyze which of these parameters are more affective on the recognition performance of the estimated model. We setup an idealized experiment by assuming a perfect estimator that yields the matched model parameters for a given condition. We perform recognition tests under additive

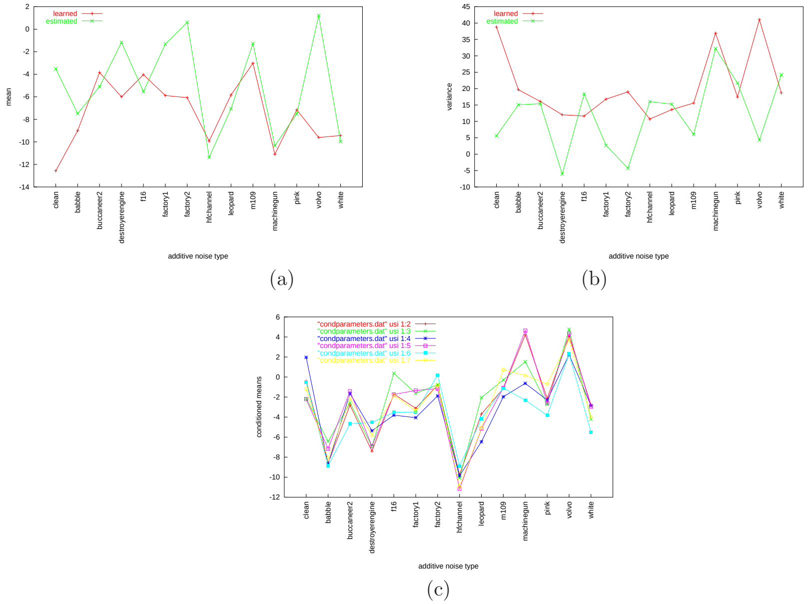


FIG. 6.3 – Model fit to the variation of (a) mean (b) variance of c4 in model ONE, 1st state, 2nd mixture (c) the variation of mean c4 for all mixtures of noise model.

white noise using only a subset of the estimated parameters and retaining clean speech model parameters for the rest. The results of these tests are given in Table 6.1. The first row is the worst case condition where only clean speech parameters are used. The last row is the ideal case where the model is perfectly matched to the test environment. The results in Table 6.1 indicate

means	variances	mixture coeffs	accuracy (%) in noise
clean	clean	clean	89.03
matched	clean	clean	98.10
matched	clean	matched	98.15
clean	matched	matched	79.20
matched	matched	matched	98.73

TAB. 6.1 – TiDigits database digit recognition accuracy(%) in white noise where not all model parameters are adapted to the environment.

that as long as the means are well estimated, the improvement provided by the variances and mixture coefficients adaptation is not significant. On the other hand if the means are mismatched there is a significant performance degradation. Based on this observation, and to simplify the compensation model, in the following experiments we estimate only the mean vectors in the new environment and use clean speech model parameters for the variances and mixture coefficients.

6.6 Experiments

In the following, all the results we present are obtained using linear functionals f_w and, as we will see later, even with such simple choice the results are very interesting and promising. The models we use for speech and noise are HMMs and single Gaussian densities, respectively. To complete the definition of our scheme, we need to define representations of the matched models $\lambda_w(k)$ and the noise models n_k. For the latter, considering the analysis in Section 6.5.1 we choose the mean vector ν_{n_k} of the Gaussian density that models the distribution of the static MFCC of n_k. For the former, we choose the set of mean vectors of the corresponding Gaussian-mixtures as the parameters that has to be transformed. This choice is based on the preliminary experiment in Section 6.5.2 and it is the common approach in compensation schemes. Precisely, for each $w \in W$, if $\mu_{wls}(n_k)$ denotes the mean vector of the mixture component l of state s, we define f_w as the set $\{A_{wls}, B_{wls}\}$ of regression matrices and bias vectors of the linear model :

$$\mu_{wls}(n_k) = A_{wls}\nu_{n_k} + B_{wls}. \tag{6.9}$$

The training of $\{A_{wls}, B_{wls}\}$ is performed in a supervised manner using minimum mean squared error (MMSE) minimization. In recognition, a new ν_{n^*} is estimated from noise n^* observations, then the model (6.9) is applied to compute the f-estimated $\tilde{\mu}_{wls}(n^*)$. For transition matrices, mixture component weights and covariances, we keep those of the clean speech model. We emphasize here that this does not mean that our compensation scheme requires a clean speech model, it is just a choice we made for the purpose of this preliminary work.

6.6.1 Experimental setting

Our experiments are carried out on Tidigits database. The corpus contains read utterances of digit sequences made up of 11 digits. Each utterance contains 1 to 7 digits. We use the man subset of the database in our evaluations. Training subset consists of 4235 utterances from 55 speakers

and test subset contains 428 utterances from 56 speakers. To create the noisy databases (in training and test), we use 15 different noise recordings from the NOISEX database which is available on the Rice University DSP group home page (http ://spib.rice.edu/spib/select_noise.html). Noisy utterances are obtained by adding a segment of the noise recording to the clean speech signal. For each utterance a random noise segment is selected from the full noise recording. Noise level is adjusted with a positive scaling factor α. That is, if s is the clean speech and n is the selected noise, the corrupted speech is :

$$y(t) = s(t) + \alpha n(t). \tag{6.10}$$

The selection of the scaling factor α specifies the average signal to noise ratio (SNR) of the noisy database. SNR is defined as 10 times the logarithmic ratio of signal power to noise power. When this definition is directly applied to compute the SNR of a speech utterance, the silence segments in speech cause misleading measurements. Since noise power is highly dominant in these segments the computed SNR value would be lower for utterances including more silence. We compute SNR using only speech segments, i.e. excluding the silence regions. This requires the segmentation of the corpus for speech and silence intervals. This segmentation is obtained by applying a forced alignment procedure on the clean corpus using HMMs trained on clean speech. We use this initial segmentation to compute the SNR of each noisy corpus.

The front-end parameterization of speech waveforms and back-end modeling of the parameterizations are based on the specifications provided by the Aurora-3 consortium [68]. Acoustic vectors are obtained using the ETSI SQL W1007 standard front-end [68]. This is a cepstral analysis scheme where 13 Mel-frequency cepstral coefficients are extracted from 23 Mel filter-bank energies. The analysis is performed on 25ms Hamming windowed frames with 10ms frame shift. Pre-emphasis filter with $a = 0.97$ is used. The resulting feature vector consists of 12 cepstral coefficients (excluding c0) and logarithmic frame energy. The first and second time derivatives (Δ end $\Delta\Delta$) are also appended as dynamic features. The digits are modeled as whole word HMMs with 16 emitting states per word and left-to-right non-skipping transition topologies. The output density of each state is modeled with a mixture of 3 Gaussians using diagonal covariance matrices. Two silence models are used : 3-states silence model for pauses before and after each utterance, and a single state short pause model for pauses between each word. We use the same parameterization scheme described above for the noise waveforms. Given a value of α, each noise source ($\alpha n(t)$) is modeled using a single Gaussian density trained from the full recording. In all experiments, the HTK software is used for parameterization, model training and recognition.

6.7 Results and discussion

In this section we evaluate the performances of our supervised-predictive compensation scheme. We start by showing an experiment where the test conditions have been seen in the training phase (in a way or another depending on the model considered). We then present experiments in a mismatched setting, where first the mismatch is in terms of SNR level, then in terms of noise type at different SNR levels. In all experiments, we have selected the noise factor to be $\alpha = 0.05$ to construct the training databases $D_1, ..., D_K (K = 15)$. This is because with $\alpha = 1$ most of the noises highly dominate speech resulting in poor performances of the matched models.

In the first experiment, the test databases are also constructed using $\alpha = 0.05$. We compare the performances of the matched models, multi-condition model, f-estimated model and the clean speech model presented (in this order from left to right) as bars-plot in Figure 6.4. For each noise

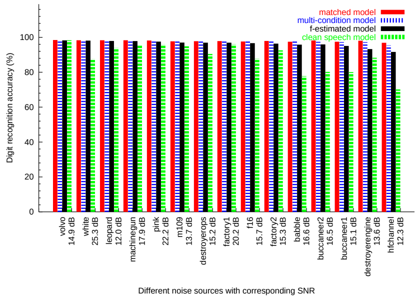


FIG. 6.4 – Comparison between recognition accuracy of matched, multi-conditions, f-estimated and clean speech models for $\alpha = 0.05$.

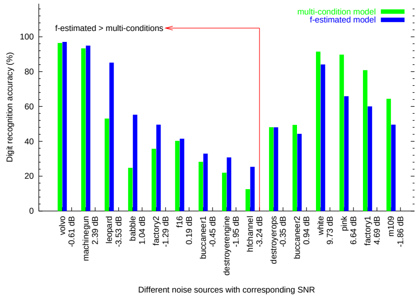


FIG. 6.5 – Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.3$.

source, displayed in the x-axis along with the average SNR computed on the corresponding test database, the 4-bars group shows the digit recognition accuracy of the four models. Noise sources are displayed from left to right in an increasing order of the difference between matched and f-estimated models' accuracies. As expected, the matched and clean speech models lead to the best and worst performances respectively. The multi-conditions model performs very well and is very close to the matched models for all noise conditions. What is interesting is that our f-estimated models are also very close the matched ones for most of the noise sources. This shows that, in this application, even with a very simple choice of the functional f the performances of the matched models are almost recovered. This also shows that there is a kind of stability in the variation of matched models w.r.t. noise models, suggesting that philosophy of our technique makes big sense.

In the second set of experiments, training is done as in the previous one ($\alpha = 0.05$), test however is carried out on databases using higher noise factors ($\alpha = 0.3, 0.4$ and 0.5) leading to lower SNR levels as compared to training conditions. Given that matched and clean speech models always lead respectively to the best and worst performances, for sake of visual clarity we show comparisons only between the performances of the multi-conditions and f-estimated models (in fact, it is the comparison of major interest in this paper). Figures 6.5, 6.6 and 6.7 present this comparison for $\alpha = 0.3, 0.4$ and 0.5 respectively. This time the noise sources are grouped into two

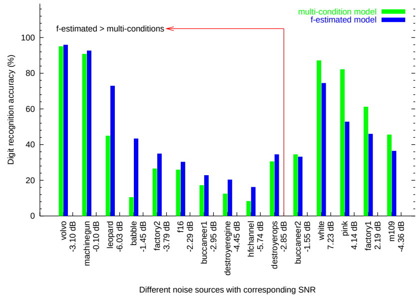


FIG. 6.6 – Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.4$.

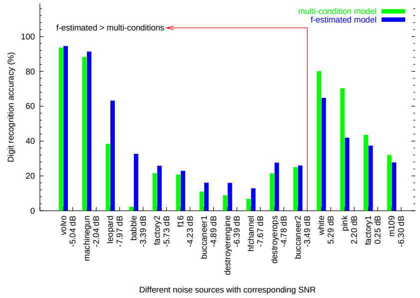


FIG. 6.7 – Comparison between recognition accuracy of multi-conditions and f-estimated models for $\alpha = 0.5$.

sets on the x-axis : the first set is composed of noise conditions where f-estimated models perform better and the second one of the remaining noise conditions (i.e., where multi-conditions model perform better). The first striking observation from these results is the fact that our f-estimated models lead to (sometimes drastically) better performances in a majority of cases. This shows that, in this application, our scheme is "globally" more robust to noise than the multi-conditions technique. Moreover, as SNR decreases the number of cases in which f-estimated models perform better than multi-condition models increases. Indeed, out of 15 noise types there are 9, 10 then 11 cases where they perform better (see Figures 6.5, 6.6 and 6.7, respectively). This suggests that the f-estimated models are more robust to SNR decrease than multi-condition models. One can also note that as the SNR decreases we "win" an additional noise, thus showing a stability in the behavior of our procedure.

Let's now analyze the cases in which the f-estimated models perform better and the recognition accuracy is significant, i.e. the first three noise sources in Figures 6.5, 6.6 and 6.7 (*volvo*, *machinegun* and *leopard*). It is interesting to observe that, besides *white* which is a synthetic noise and can be considered as an outlier, these noises have the best accuracies in no-mismatch conditions (see the ordering in Figure 6.4). For these noises, the estimated models perform consistently better than multi-condition models. Indeed, these are the noises that has the most accurate estimates in training conditions (see the ordering in Figure 6.4). Now, consider the *hfchannel* noise for which the corresponding f-estimated model has the worst recognition performance in no-mismatch conditions. As the SNR decreases the performance of the f-estimated model degrades very fast (see column 9 in Figures 6.5, 6.6 and 6.7). These observations suggest that the generalization properties of f-estimated models are related to the estimation accuracy in no-mismatch conditions, that is, if the estimation error (w.r.t. matched model) is small in no-mismatch conditions then the estimation in mismatched test conditions is also relatively accurate.

SNR	matched	multi-conditions	f-estimated
$\alpha = 0.05$	98.33	95.54	**97.56**
$\alpha = 0.3$	98.05	82.38	**88.86**
$\alpha = 0.4$	97.84	78.41	**84.75**
$\alpha = 0.5$	97.49	75.49	**82.03**

Tab. 6.2 – Comparison between matched, multi-condition and estimated models when the target noise (machinegun) is excluded from training data.

In the last experiment, we evaluate the generalization property of our scheme when the mismatch is in terms of noise type and SNR level. To do this we select a target noise that will be used in tests. We exclude the matched model of this target noise from the training set of the regression model (6.9), and we exclude the corresponding training corpus when learning the multi-conditions model. Given the lack of space, we show results for only one target noise : *machinegun*. Table 6.2 presents the recognition scores we obtain using the multi-conditions and f-estimated models at different SNR levels as in the previous experiments. The scores of the corresponding matched models are also given as a reference. These results show that, even though we did not use any information about target noise in training, the performances of the f-estimated models are significantly higher than those of the multi-conditions model. This suggests that our compensation scheme is not only more robust (than multi-condition training) to SNR change but also to new noise types not seen in training.

6.8 Conclusion

We presented a new predictive compensation scheme which not only has several advantages w.r.t. classical ones but also is capable to perform better than multi-conditions training. The experiments we carried out show that this new technique is definitely very interesting, promising and should be further investigated.

One of the problems to be addressed is the choice of the functional form for the variation of speech model parameters with respect to noise. In this chapter we have used a simple deterministic linear function. Perspectives in this direction include the use of probabilistic formulations such that causal relations can be extracted between noise and speech parameters.

Another problem is the noise representation. In our formulation the domain of noise model parameters is not constrained with speech feature space. Although we have used such a representation for our preliminary experiments, the choice of noise space should be addressed carefully in order to yield an efficient mapping.

Chapitre 7

Conclusions

In this thesis, we analyzed the four main modules of automatic speech recognition : acoustic modeling, language modeling, speech feature extraction and noise compensation. We employed novel techniques in all these problems and provided new perspectives.

One of the main contributions is the use of Bayesian networks for probabilistic modeling in ASR. Most probabilistic modeling techniques used in speech community can indeed be unified within this formalism. Our approach unifies and generalizes classical techniques used in the state of the art ASR systems, namely HMMs and n-grams.

Our contribution in acoustic modeling consists of the use of structure learning paradigm to extract appropriate dependence relations between hidden and observed processes of speech. This data-driven approach allows modeling of each acoustic unit with distinctive conditional independence assertions. We defined a class of models that encapsulates various dependence relations and applied the structural expectation maximization algorithm in dynamic Bayesian networks setting. Since the dependence properties are specified using a data-driven learning algorithm, one of the advantages of our approach is that model complexity is increased only if it is dictated by data. Hence HMM based acoustic modeling is only one of the possible solutions that would be automatically provided by this approach if it is the best choice to fit training data. In order to apply this approach for ASR we have developed the required machinery in BN framework. We defined a DBN representation for non-Markov processes that is required to identify dependence relations of speech observations to future states of the hidden process. We also developed a speech decoding algorithm that exploits properties of the class of models we proposed. The experimental results on isolated and continuous speech recognition indicate that the proposed modeling scheme achieves better performances as compared to HMMs with equivalent complexity. The use of DBNs in acoustic modeling has gained a lot of interest in the community and there are more and more attempts to exploit the enriching properties of this framework. Our approach takes its place as one of the pioneering works.

Another probabilistic modeling module of ASR that we have addressed is language modeling. In this domain, we have used the DBN setting to propose a unified setting for the classical n-gram and n-class models. In general n-gram and n-class models have complementary characteristics. Our approach is based on the exploitation of each linguistic unit, i.e. word, class, etc., within a single model. We have analyzed the conditional independence properties of classical approaches and proposed a new set of models in DBN setting that covers classical ones. By doing so we are able to learn a unique model that have the properties of both n-gram and n-class models. We performed preliminary evaluations of this technique and showed potential advantages. Our perspectives include the development of automatized learning algorithms that extract the ap-

propriate model structure using linguistic criteria. The structure learning algorithms proposed in BN framework does not seem to be appropriate for language modeling because they do not take into account linguistic constraints.

Another direction of my thesis is on new acoustic feature extraction algorithms for ASR. Existing feature extraction algorithms are developed considering the properties of HMM formalism. Therefore there is a need for new directions in this field to be able to extract informative features that can later be processed within the DBN setting without the limiting constraints of HMMs. With this motivation in mind we reviewed wavelet analysis based feature extraction schemes for speech recognition and proposed a new technique for frequency localized robust feature extraction using wavelets. The proposed scheme is based on the projection of mel-scale filter-bank energies on a wavelet basis. This technique is indeed a reformulation of the frequency filtering approach that emphasizes formant frequencies with a simple linear filter and achieves robust frequency localized features. Using a wavelet basis improves decorrelation property and allows a controlled localization in frequency domain. Our perspective in this direction is to devise features that can be used to capture frequency dynamics of speech using appropriate statistical models. The proposed set of features and the DBN framework constitute a potential candidate for this goal.

In speech feature extraction we partially addressed the problem of noise robustness. Although robust features are indispensable for handling noisy conditions, compensation schemes in model parameter space are more advantageous especially if application conditions are highly varying. In this respect we proposed a new predictive compensation scheme which is based on a supervised learning of the effect of noise on speech model parameters. The advantage of our scheme is that it does not make any assumptions on how the noise sources alter the speech data. Our approach is based on the statistical exploitation of speech models' parameter variations under noisy conditions. We presented the results of a preliminary attempt based on a linear regression model of speech and noise model parameters. Even with such a simple model we achieved good generalization properties. Perspectives in this direction include the use of more complicated models (i.e. deterministic or probabilistic) to capture the variation of speech models with noise.

Annexe A

JLO algorithm

The JLO algorithm is a recursive message passing algorithm that works on the junction tree of a Bayesian network (BN). The junction tree of a BN is constructed from its directed acyclic graph (DAG) using some graph-theoretic tools. Each node of the tree, called clique, consists of a subset of variables. The joint probability distribution (JPD) of the BN is factorized over these clique potentials. Using a local message passing scheme between cliques the clique potentials and evidences are distributed to the complete set of variables.

In the sequel we describe an algorithm for constructing the junction tree and show how the message passing procedure is used for exact inference in discrete BNs.

A.1 Constructing the Junction Tree

In order to define the junction tree we need to define the term clique. A clique is a complete set of nodes which is not a proper subset of another complete set. A set of nodes is said to be complete if every pair of nodes in the set is linked. A junction tree is a special tree of cliques that satisfies the running intersection property (RIP). RIP implies that if a node is contained in any 2 cliques, then it is contained in all the cliques in the unique path between them.

The first step in the construction of the junction tree is to obtain an equivalent undirected graph satisfying the associated Markov property of the BN. This is performed by the process of moralization as explained in AlgorithmA.1.1. The moral graph of a DAG is obtained by introducing additional undirected edges between any two nodes with a common child and subsequently replacing all directed edges with undirected ones. The moralization process guarantees that a family of nodes (a node with all of its parents) will occur together in a clique.

Algorithm A.1.1 Moralization
- *Input : A directed acyclic graph* $G = (X, E)$ *with nodes* $X = \{X_1, \ldots, X_N\}$ *and directed edges* E.
- *Outputs : A set of undirected links* L *such that* $G' = (X, L)$ *is the moral graph of* G.
  1. *Set* $L = \emptyset$.
  2. *For* $i = 1, \ldots, N$,
     - *Add to* L *the link* (X_i, X_j) *for each* $X_j \in Pa(X_i)$, *where* $Pa(X_i)$ *are the parents of* X_i *in* G.
     - *Add to* L *the link* (X_j, X_k) *for each* $X_j, X_k \in Pa(X_i), j \neq k$.

The second step is to add sufficient edges to the moral graph to obtain a triangulated (chordal) graph. The aim of triangulation is to obtain a decomposable model such that the joint probability distribution can be factorized over the clique potential functions. An undirected graph is triangulated if every loop of length four or more has at least one chord. There are several ways to add a chord in a loop with length greater than four. Hence, the triangulation process is not unique. In general it is desired to obtain a triangulation with a minimum number of additional edges. However, this problem is NP-complete [128].

Here we present the Maximum Cardinality Search Fill-In algorithm in AlgorithmA.1.2 to obtain a triangulation of a given undirected graph [12]. This algorithm can be implemented in linear time $O(N + l)$, where N is the number of nodes and l is the number of links in the graph [122] . Given an initial node the algorithm constructs a triangulated graph and returns an ordering of the nodes.

Algorithm A.1.2 Triangulate
- *Input : An undirected graph* $G = (X, L)$ *with* N *nodes and an initial node* X_i.
- *Outputs : A set of links* $L^{'}$, *such that* $G^{'} = (X, L \bigcup L)^{'}$ *is a triangulated graph. An ordering (numbering)* $\alpha(j)$ *of nodes*
  1. *Set* $L^{'} = \emptyset$.
  2. *Set* $j = 1$, *and* $\alpha(j) = X_i$.
  3. *An unnumbered node* X_k *is with a maximum number of numbered neighbors is assigned label* j, $\alpha(j) = X_k$.
  4. *If* $\{X_k \bigcup Neighbors(X_k)\} \bigcap \{\alpha(1) \ldots \alpha(j-1)\}$ *is not complete, add necessary links to make this set complete and go to Step 2 ; otherwise, go to Step 5.*
  5. *If* $j = N$, *then stop ; otherwise let* $j = j + 1$ *and go to Step 3.*

Once the decomposable structure is obtained, the last step consists of the specification of the cliques and the construction of the junction tree. For constructing the junction tree we present the linear time algorithm defined in [132]. The algorithm proceeds in two steps. The first step is tree formation (see AlgorithmA.1.3), where a clique is defined for each node with its lower numbered neighbors. Then, the cliques are linked to satisfy the RIP property.

Algorithm A.1.3 Tree Formation
- *Input : A triangulated graph* $G = (X, L)$ *with a numbering* $\alpha(j)$ *of nodes.*
- *Outputs : The junction tree with non-maximal cliques.*
  1. *Form a clique for each variable and its lower numbered neighbors.*
  2. *Order the cliques in increasing order according to the highest numbered node in each clique.*
  3. *For each clique* C_i *in increasing order*
     - *Identify the subset of its constituent variables that have occurred in the lower numbered cliques.*
     - *Find a lower numbered clique* C_j *that contains this subset.*
     - *Make* C_i *the parent of* C_j.

The resulting cliques are not necessarily maximal. In the next step a tree reduction algorithm is used to obtain the maximal cliques and replace the smaller cliques with the maximal ones (see AlgorithmA.1.4).

Algorithm A.1.4 Tree Reduction
- *Input : A junction tree of cliques created by Tree Formation algorithm.*
- *Outputs : The junction tree with maximal cliques.*
  - *repeat*
    1. *Identify a child that is a superset of its parent.*
    2. *Contract the edge between the two and replace the parent. by the child*
  - *until no child is a superset of its parent.*

Example Consider the DAG given in A.1.a associated to a BN for the variables $X = \{A, B, C, D, E, F, G, H, I, J\}$. The moral graph of this BN is obtained by marrying the nodes that have the same child, i.e. introducing the links (A-B), (C-D) and (G-H) as shown in figure A.1.b. The maximum cardinality search fill-in algorithm yields the triangulated graph in figure A.1.c. The algorithm also outputs an ordering of nodes as : $\alpha(j) = \{A, B, D, C, E, F, G, H, J, I\}$. Using this ordering and applying the Tree Formation and Tree Reduction algorithms one achieves the junction tree shown in figure A.1.d.

A.2 Message passing

The joint probability density of a set of variables can be represented as a product of clique potential functions on the associated junction tree.

$$P(\mathbf{X}) = \prod_{C \in \mathcal{C}} a_C(\mathbf{X}_C) \tag{A.1}$$

This factorization can be generalized using the separators in the junction tree. For each edge joining two cliques C_i, C_j in the junction tree a separator set S is defined as the intersection of the nodes in C_i and C_j. Associating a separator potential function to each separator set, we have the following factorization of the joint probability density.

$$P(\mathbf{X}) = \frac{\prod_{C \in \mathcal{C}} a_C(\mathbf{X}_C)}{\prod_{S \in \mathcal{S}} b_S(\mathbf{X}_S)} \tag{A.2}$$

In the above equation, whenever the denominator is zero, the division is defined to be zero. With any valid initialization of clique and separator potential functions, at the end of the message passing algorithm every potential function becomes the marginal density for the relative set of variables. And at any step of the algorithm the potential functions constitute a valid factorization.

After the initialization, the local message passing scheme proceeds as follows. A *flow* from clique C_i to its neighbor C_j is defined through their separator S_k. First the separator potential $b_{S_k}(\mathbf{X}_{S_k})$ is updated by marginalizing the clique potential over the variables that are in C_i but not in S_k.

$$b^*_{S_k}(\mathbf{X}_{S_k}) = \sum_{C_i \setminus S_k} a_{C_i}(\mathbf{X}_{C_i}) \tag{A.3}$$

Then the update factor of the separator is used to update the potential of the destination clique C_j.

$$\lambda_{S_k}(\mathbf{X}_{S_k}) = \frac{b^*_{S_k}(\mathbf{X}_{S_k})}{b_{S_k}(\mathbf{X}_{S_k})} \tag{A.4}$$

$$a^*_{C_j}(\mathbf{X}_{C_j}) = a_{C_j}(\mathbf{X}_{C_j})\lambda_{S_k}(\mathbf{X}_{S_k}) \tag{A.5}$$

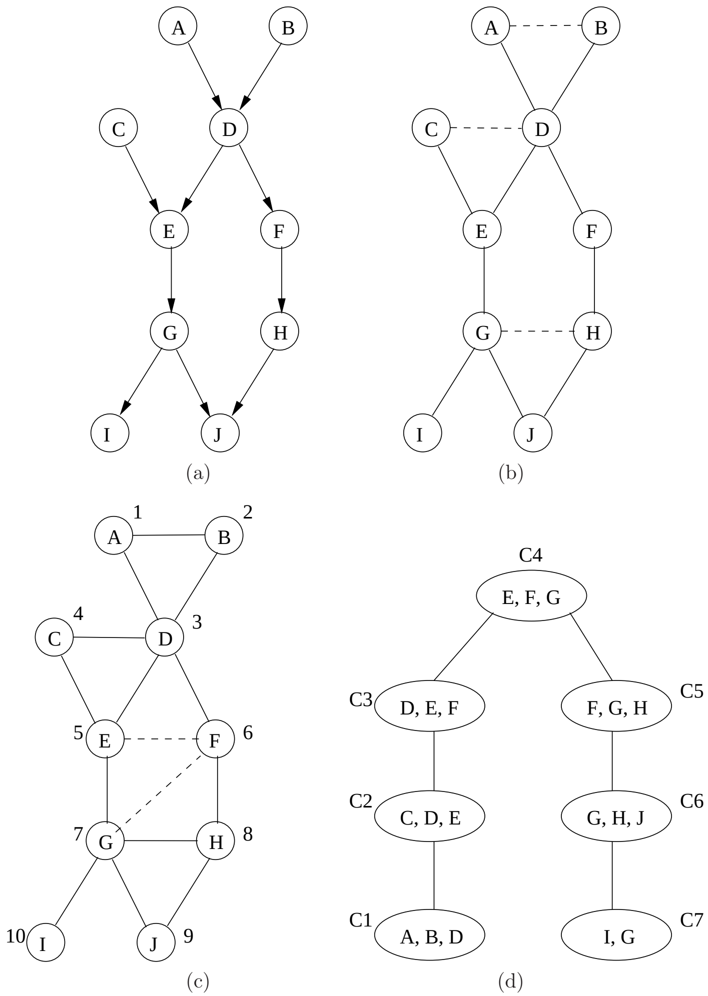


FIG. A.1 – Illustration of the junction tree construction algorithm taken from [12] (pages 144-145). (a) A directed acyclic graph. (b) Corresponding moralized graph. (c) Triangulated graph with the indicated node ordering. (d) A junction tree associated with the graph.

Note that the factorized representation of the joint distribution is still valid with the locally updated potentials. In order to distribute the information in each clique to the whole tree a two-phase propagation algorithm is used in general. Given a root clique in the tree, the *collection-phase* absorbs the flows starting from the leafs towards the root. Once all the flows are collected in the root, messages are send towards the leafs in the *distribution-phase*. At the end of the algorithm at most two flows must have occurred along each edge.

A.3 Initializing the Junction Tree

Given the conditional probability distributions $p(X_i|\Pi_i)$ of the variables X_i, the initialization of the junction tree can be performed as follows.

1. Assign each X_i to just one clique.
2. For each clique C that is assigned with at least one variable define the potential function as the product of $p(X_i|\Pi_i)$ over all X_i assigned to C.
3. For all separators and the remaining cliques define the potential function to be 1.

With the above initialization of the junction tree, the message passing algorithm yields a prior probability distribution in each clique and separator.

$$P(\mathbf{X}) = \frac{\prod_{C\in\mathcal{C}} P_C(\mathbf{X}_C)}{\prod_{S\in\mathcal{S}} P_S(\mathbf{X}_S)} \tag{A.6}$$

The same algorithm can also be used to compute posterior probability distributions given an observation of a subset of variables. Once the clique tree is properly initialized with the observation, the two-phase propagation algorithm diffuses the observation and the resultant potentials are the posterior marginal distributions in each clique and separator. The observations are used in the algorithm as follows.

A.3.1 Incorporating evidence

Let $\mathbf{X_h}$ be the subset of variables that are hidden or unobserved and let $D_l = \{X_i = x_i^*, X_j = x_j^*, \ldots\}$ be an observation of the set of variables $\mathbf{X_o} = \mathbf{X}\backslash\mathbf{X_h}$. In order to compute $p(\mathbf{X_h}|D_l)$ we first define an evidence function such that

$$g(x_i) = \begin{cases} 1 & \text{if } x_i = x_i^* \\ 0 & \text{otherwise.} \end{cases} \tag{A.7}$$

After initializing the junction tree with the conditional probability distributions, we multiply each clique potential with the evidence function according to the variable assignment in the initialization step.

$$a_C(\mathbf{X_C}) = a_C(\mathbf{X_C}) * \prod_{i:X_i\in\mathbf{X_C}} g(x_i) \tag{A.8}$$

With the incorporation of evidence, when the propagation algorithm reaches equilibrium, the clique potentials are the joint probability of the local hidden variables and the observed evidence.

$$a_C(\mathbf{X_C}) = P(\mathbf{X_C^h}, D_l) \tag{A.9}$$

If the potential function at a clique is normalized to sum to 1, one gets the conditional probability of the local hidden variables given the observation,

$$P(\mathbf{X_C^h}|D_l). \tag{A.10}$$

If the clique potential is marginalized over the hidden variables the probability of the observed evidence is obtained.

$$P(D_l) = \sum_{\mathbf{X_C^h}} P(\mathbf{X_C^h}, D_l) \tag{A.11}$$

A.4 Generalized operations

Other than probability computations, the inference engine can also be used for finding the most probable configuration of the variables or selecting a random sample from the distribution.

A.4.1 Most probable configuration :

In order to find the most probable configuration of the variables with or without evidence, it is sufficient to replace the marginalization operation in the flows with max-marginalization. The max-margin of a potential $a(X)$ on $Y \subseteq X$ is defined as follows :

$$b(Y) = \max_{Y\backslash X} a(X) \tag{A.12}$$

When the propagation algorithm is run with max-marginalization incorporating some evidence, the best explanation of the evidence is obtained. This procedure is known as Dawid's algorithm [30].

A.4.2 Random configuration :

Another useful operation could be generation of random samples from the joint probability distribution. This is performed with a slight modification in the propagation algorithm. After the regular collection phase, the distribution phase is replaced by a sample random-configuration phase. Starting from the root clique, a random-configuration is picked from the distribution on the each clique. Whenever a random-configuration is picked in a clique, this is entered as evidence by setting all other entries to zero in the clique potential.

Annexe B

Complete data ML parameter learning

B.1 Likelihood Function

Let C_l denote a complete (i.e. C_l consists of the realization of all the variables) random sample from the jpd of $\mathbf{X}$. Using the factorization property of the jpd, the likelihood of C_l can be expressed as a product of local likelihoods.

$$P(C_l|S,\Theta) = \prod_{i=1}^{n} P(x_i|\pi_i, S, \Theta) \tag{B.1}$$

x_i^k and π_i^j denote the respective realizations of X_i and Π_i in C_l. For a data set D consisting of L i.i.d. samples, the likelihood can be computed as :

$$P(D|S,\Theta) = \prod_{l=1}^{L} P(C_l|S,\Theta) \tag{B.2}$$

The maximum likelihood estimates are obtained as :

$$\hat{\Theta}_{ML} = \arg\max_{\Theta} p(D|S,\Theta) \tag{B.3}$$

B.2 ML parameter estimation for discrete BNs

For a discrete BN, the likelihood of a complete sample C_l is written as :

$$\begin{aligned} P(C_l|S,\Theta) &= \prod_{i=1}^{n} P(x_i^k|\pi_i^j, S, \Theta) && \text{(B.4)} \\ &= \prod_{i=1}^{n} \theta_{ijk} && \text{(B.5)} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} \theta_{ijk}^{1_{lijk}} && \text{(B.6)} \\ & && \text{(B.7)} \end{aligned}$$

$$1_{lijk} = \begin{cases} 1 & \text{if } X_i = x_i^k \text{ and } \Pi_i = \pi_i^j \text{ in } C_l \\ 0 & \text{otherwise} \end{cases} \tag{B.8}$$

The likelihood of a complete dataset D, where each random sample C_l is independent and identically distributed (i.i.d.), can be written as :

$$\begin{aligned} P(D|S,\Theta) &= \prod_{l=1}^{L} P(C_l|S,\Theta) && \text{(B.9)} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} && \text{(B.10)} \end{aligned}$$

where N_{ijk} is the total number of observations satisfying $X_i = x_i^k$ and $\Pi_i = \pi_i^j$ in the data set D.

$$N_{ijk} = \sum_{l=1}^{L} 1_{lijk} \tag{B.11}$$

Maximizing the likelihood term is equivalent to maximizing the log likelihood.

$$\log P(D|S,\Theta) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\theta_{ijk}$$

Hence, we obtain the ML parameter estimates as follows where N_{ijk} are computed by counts in the observed data.

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k=1}^{r_i} N_{ijk}} \qquad \begin{array}{l} i = 1,\ldots,n \\ j = 1,\ldots,q_i \\ k = 1,\ldots,r_i. \end{array} \tag{B.12}$$

B.3 ML parameter estimation for continuous BNs

For a continuous BN the likelihood of a complete dataset D is written as :

$$\begin{aligned} P(D|S,\Theta) &= \prod_{l=1}^{L} P(C_l|S,\Theta) && \text{(B.13)} \\ &= \prod_{l=1}^{L}\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\{-\frac{(x_i^l - \beta_i^T\pi_i^l)^2}{2\sigma_i^2}\} && \text{(B.14)} \end{aligned}$$

Maximizing the log-likelihood , one obtains the following ML parameter estimates :

$$\begin{aligned} \hat{\beta}_i^T &= \Big[\sum_{l=1}^{L} \pi_i^l \pi_i^{lT}\Big]^{-1}\Big[\sum_{l=1}^{L} x_i^l \pi_i^{lT}\Big] && \text{(B.15)} \\ \hat{\sigma}_i &= \frac{\sum_{l=1}^{L}(x_i^l - \hat{\beta}_i^T\pi_i^l)^2}{L} \qquad for \quad i = 1,\ldots,n && \text{(B.16)} \end{aligned}$$

B.4 ML parameter estimation for hybrid BNs

For hybrid BNs the likelihood can be decomposed according to the type (discrete or continuous) of the variable and its parents. Then, each factor in the decomposition can be maximized separately. Therefore, the ML parameters for discrete variables with discrete parents and continuous variables with continuous parents are as given above. For the parameters of continuous variables with discrete and continuous parents, maximization should be performed separately for each distinct configuration of discrete parents. Hence the estimates are obtained as :

$$\hat{\beta}_{ij}^T = \Big[\sum_{l=1}^{L} \pi_i^{j_l} \pi_i^{j_l T}\Big]^{-1} \Big[\sum_{l=1}^{L} x_i^l \pi_i^{j_l T}\Big] \tag{B.17}$$

$$\hat{\sigma}_{ij} = \frac{\sum_{l=1}^{L}(x_i^l - \hat{\beta}_{ij}^T \pi_i^{j_l})^2}{L} \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, q_i \end{array} \tag{B.18}$$

Annexe C

Incomplete data ML parameter learning : EM algorithm

C.1 EM algorithm for parameter learning in discrete BNs

Let C_l denote a completion of the incomplete sample D_l with some random realizations of $\mathbf{X}_h$. Then the auxiliary function is written as :

$$
\begin{aligned}
Q(\Theta^{'}|\Theta) &= \sum_{x_h} P(\mathbf{X}_h|D_l, S, \Theta) \log P(C_l|S, \Theta^{'}) && \text{(C.1)}\\
&= \frac{1}{P(D_l|S,\Theta)} \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \log \theta^{'}_{ijk} \sum_{x_h} P(\mathbf{X}_h, D_l|S,\Theta) 1_{lijk} && \text{(C.2)}\\
&= \frac{1}{P(D_l|S,\Theta)} \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \log \theta^{'}_{ijk} P(x_i^k, \pi_i^j|D_l, S, \Theta) && \text{(C.3)}
\end{aligned}
$$

The M-step gives the parameters for the next iteration where we need to perform inference to compute the probabilities.

$$
\hat{\theta}_{ijk} = \frac{P(x_i^k, \pi_i^j|D_l, S, \Theta)}{\sum_{k=1}^{r_i} P(x_i^k, \pi_i^j|D_l, S, \Theta)} \tag{C.4}
$$

For a set of L observations, we have :

$$
\hat{\theta}_{ijk} = \frac{\sum_{l=1}^{L} P(x_i^k, \pi_i^j|D_l, S, \Theta)}{\sum_{k=1}^{r_i}\sum_{l=1}^{L} P(x_i^k, \pi_i^j|D_l, S, \Theta)} \tag{C.5}
$$

C.2 EM algorithm for parameter learning in continuous BNs

$$
\begin{aligned}
Q(\Theta^{'}|\Theta) &= \int P(\mathbf{X}_h|D_l, S, \Theta) \log P(C_l|S, \Theta^{'}) dx_h && \text{(C.6)}\\
&= \int P(\mathbf{X}_h|D_l, S, \Theta) \sum_{i=1}^{n} \Big[\log \frac{1}{\sqrt{2\pi\sigma_i^{2'}}} - \frac{(x_i^l - {\beta_i^{'}}^T \pi_i^l)^2}{2\sigma_i^{2'}}\Big] dx_h && \text{(C.7)}\\
&= \sum_{i=1}^{n} \Bigg[\log \frac{1}{\sqrt{2\pi\sigma_i^{2'}}} \\
&\quad -\frac{1}{2\sigma_i^{2'}} \Big[E\{x_i^l\} - 2{\beta_i^{'}}^T E\{x_i^l \pi_i^{lT}\} + E\{\pi_i^l \pi_i^{lT}\}\Big]\Bigg] && \text{(C.8)}
\end{aligned}
$$

In the M-step the maximization over $\beta_i^{'}$ and $\sigma_i^{2'}$ is performed for each i separately.

$$
\begin{aligned}
\hat{\beta}_i^T &= \Big[E\{\pi_i^l \pi_i^{lT}\}\Big]^{-1} \Big[E\{x_i^l \pi_i^{lT}\}\Big] && \text{(C.9)}\\
\hat{\sigma}_i &= E\{(x_i^l - \hat{\beta}_i^T \pi_i^l)^2\} \qquad for \quad i = 1, \ldots, n && \text{(C.10)}
\end{aligned}
$$

For a set of L observations, we have :

$$
\begin{aligned}
\hat{\beta}_i^T &= \Big[\sum_{l=1}^{L} E\{\pi_i^l \pi_i^{lT}\}\Big]^{-1} \Big[\sum_{l=1}^{L} E\{x_i^l \pi_i^{lT}\}\Big] && \text{(C.11)}\\
\hat{\sigma}_i &= \frac{\sum_{l=1}^{L} E\{(x_i^l - \hat{\beta}_i^T \pi_i^l)^2\}}{L} \qquad for \quad i = 1, \ldots, n && \text{(C.12)}
\end{aligned}
$$

The expectations are computed over $P(\mathbf{X}_h|D_l, S, \Theta)$ which can be computed by inference using the initial parameter set.

C.3 EM algorithm for parameter learning in hybrid BNs

As in ML estimation, the parameter estimates of hybrid BNs are considered separately for different types of parents and children. The estimates for a hybrid configuration where a discrete node has both discrete and continuous parents, are obtained as follows :

$$
\begin{aligned}
\hat{\beta}_{ij}^T &= \Big[\sum_{l=1}^{L} E\{\pi_i^{j_l} \pi_i^{j_l T}\}\Big]^{-1} \Big[\sum_{l=1}^{L} E\{x_i^l \pi_i^{j_l T}\}\Big] && \text{(C.13)}\\
\hat{\sigma}_{ij} &= \frac{\sum_{l=1}^{L} E\{(x_i^l - \hat{\beta}_{ij}^T \pi_i^{j_l})^2\}}{L} \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, q_i \end{matrix} && \text{(C.14)}
\end{aligned}
$$

Annexe D

Maximum Likelihood Parameter Estimation of Acoustic DBN Models

Consider a DBN with structure $S = (\kappa, \tau_p, \tau_f)$ and parameters $\Theta = (\Theta_h, \Theta_o)$. $\Theta_h = (a_{\mathbf{i}j}, a_{\mathbf{i}j}[t])$ denotes the set of transition probabilities for the discrete hidden variable $X_h[t]$ that takes its values x_{ht} in the set of ordered labels $I = \{1, \ldots, N\}$. The parameters of a single Gaussian output density for the continuous observable variable $X_o[t]$ are denoted as $\Theta_o = (\mu_{\mathbf{i}}, \Sigma_{\mathbf{i}}, \mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t])$.

The maximum likelihood parameter estimation problem is to find the set of model parameters $\hat{\Theta}$ such that the emission probability of a given observation sequence $\mathbf{x_o}_1^T$ is maximized.

$$\hat{\Theta} = \arg\max_{\Theta} P(\mathbf{x_o}_1^T|\Theta, S)$$

There is no analytical closed form solution to this problem. However, we can choose Θ such that the likelihood is locally maximized using an iterative procedure. We use Baum-Welch method which is also known as the EM (Expectation-Maximization) algorithm. Starting with some initial parameters Θ, we estimate a new set of parameters Θ' such that the likelihood is maximized with respect to Θ'. The re-estimation formulas are derived by maximizing Baum's auxiliary function :

$$Q(\Theta', \Theta) = \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta) \log P(\mathbf{x_o}_1^T, \mathbf{x_h}_1^T|\Theta').$$

over Θ'. Because,

$$Q(\Theta', \Theta) \geq Q(\Theta', \Theta') \Rightarrow P(\mathbf{x_o}_1^T|\Theta, S) \geq P(\mathbf{x_o}_1^T|\Theta', S)$$

we can maximize the function $Q(\Theta', \Theta)$ over Θ' to improve Θ in the sense of increasing the likelihood $P(\mathbf{x_o}_1^T|\Theta', S)$.

The factorization property of DBNs allows us to factorize the data likelihood in terms of the conditional probabilities of observable and hidden variables.

$$P(\mathbf{x_h}_1^T, \mathbf{x_o}_1^T|\Theta) = \prod_{t=1}^{T} P(x_{ht}|\pi_{ht}) \prod_{t=1}^{T} P(x_{ot}|\pi_{ot})$$

Therefore, we can handle the auxiliary function in two terms :

$$Q(\Theta', \Theta) = Q_h(\Theta', \Theta_h) + Q_o(\Theta', \Theta_o)$$

where,

$$\begin{aligned} Q_h(\Theta, \Theta'_h) &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ht}|\pi_{ht}) \\ Q_o(\Theta, \Theta'_o) &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ot}|\pi_{ot}). \end{aligned}$$

The maximization can be performed on individual terms separately.

D.1 Transition Probabilities

Estimation of transition probabilities requires maximization of $Q_h(\Theta, \Theta'_h)$ based on the following constraints :

$$\begin{aligned} a'_{\mathbf{i}j}[t] &\geq 0 \; \forall\, j \in I,\; \mathbf{i} \in I^t \\ \sum_j a'_{\mathbf{i}j}[t] &= 1 \; \forall\, \mathbf{i} \in I^t \\ a'_{\mathbf{i}j} &\geq 0 \; \forall\, j \in I,\; \mathbf{i} \in I^\kappa \\ \sum_j a'_{\mathbf{i}j} &= 1 \; \forall\, \mathbf{i} \in I^\kappa \end{aligned}$$

$Q_h(\Theta, \Theta'_h)$ can further be decomposed into a sum of functions each of which depends only on $\mathbf{i}$ and t :

$$\begin{aligned} Q_h(\Theta, \Theta'_h) &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ht}|\pi_{ht}) \\ &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ht}|\pi_{ht}) \sum_{\mathbf{i},j} \delta(x_{ht}, j)\delta(\pi_{ht}, \mathbf{i}) \\ &= \sum_{\mathbf{i},j} \sum_{t=1}^{T} \log a'_{\mathbf{i}j}[t] \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta)\delta(x_{ht}, j)\delta(\pi_{ht}, \mathbf{i}) \\ &= \sum_{\mathbf{i}} \left(F_{h\mathbf{i}}(a'_{\mathbf{i}j}) + \sum_{t=1}^{\kappa} F_{h\mathbf{i}t}(a'_{\mathbf{i}j}[t]) \right) \end{aligned}$$

where :

$$\begin{aligned} F_{h\mathbf{i}}(a'_{\mathbf{i}j}) &\triangleq \sum_j \sum_{t=\kappa+1}^{T} \xi(t, j, \mathbf{i}) \log a'_{\mathbf{i}j} \\ F_{h\mathbf{i}t}(a'_{\mathbf{i}j}[t]) &\triangleq \sum_j \xi(t, j, \mathbf{i}) \log a'_{\mathbf{i}j}[t] \\ \xi(t, j, \mathbf{i}) &\triangleq P(\mathbf{x_h}_1^T|\mathbf{x_o}_1^T, \Theta)\delta(x_{ht}, j)\delta(\pi_{ht}, \mathbf{i}) \\ &= P(x_{ht} = j, \pi_{ht} = \mathbf{i}|\mathbf{x_o}_1^T, \Theta) \end{aligned}$$

Each $F_{h\mathbf{i}t}(a'_{\mathbf{i}j}[t])$ and $F_{h\mathbf{i}}(a'_{\mathbf{i}j})$ have the following form,

$$F(y_j) = \sum_{j=1}^{N} w_j \log y_j$$

Maximization of this function with respect to y_j, subject to the constraints $\sum_{j=1}^{N} y_j = 1$, $y_j \geq 0$ can easily be solved analytically, by applying the method of Lagrange multipliers for constraint optimization. This requires finding (λ, y_j) such that :

$$\begin{cases} \frac{\partial}{\partial y_j} F(y_j) + \lambda = 0, \ \forall\ j = 1, \ldots, N \\ \sum_{j=1}^{N} y_j = 1 \end{cases}$$

Given that :

$$\frac{\partial}{\partial y_j} \sum_{j=1}^{N} w_j \log y_j = \frac{w_j}{y_j}$$

the system of equations becomes

$$\begin{cases} w_j + \lambda y_j = 0, \ \forall\ j = 1, \ldots, N \\ \sum_{j=1}^{N} y_j = 1 \end{cases}.$$

Summing the first set of equations and applying the constraint one gets :

$$\lambda = -\sum_{j=1}^{N} w_j$$

which leads to the solution as :

$$y_j = \frac{w_j}{\sum_{j=1}^{N} w_j}$$

When similar arguments are applied to $F_{h\mathbf{i}t}(a'_{\mathbf{i}j}[t])$ and $F_{h\mathbf{i}}(a'_{\mathbf{i}j})$ we obtain the transition probabilities as :

$$\begin{aligned} a_{\mathbf{i}j} &= \frac{\sum_{t=\kappa+1}^{T} \xi(t, j, \mathbf{i})}{\sum_{t=\kappa+1}^{T} \sum_{j=1}^{N} \xi(t, j, \mathbf{i})} \\ a_{\mathbf{i}j}[t] &= \frac{\xi(t, j, \mathbf{i})}{\sum_{j=1}^{N} \xi(t, j, \mathbf{i})}, \text{ for } 1 \leq t \leq \kappa \end{aligned}$$

D.2 Output Probabilities

In analogy to the previous section, $Q_o(\Theta, \Theta'_o)$ can be expressed as a sum of independent terms, each of which is related to a specific distribution.

$$\begin{aligned} Q_o(\Theta, \Theta'_o) &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T | \mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ot}|\pi_{ot}) \\ &= \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T | \mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ot}|\pi_{ot}) \sum_{\mathbf{i}} \delta(\pi_{ot}, \mathbf{i}) \\ &= \sum_{\mathbf{i},j} \sum_{t=1}^{T} \log b'_{\mathbf{i}t}(x_{ot}) \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T | \mathbf{x_o}_1^T, \Theta) \delta(\pi_{ot}, \mathbf{i}) \\ &= \sum_{\mathbf{i}} \left(F_{o\mathbf{i}}(b'_{\mathbf{i}}(x_{ot})) + \sum_{t=1}^{\tau_p} F_{o\mathbf{i}t}(b'_{\mathbf{i}t}(x_{ot})) + \sum_{t=T-\tau_f+1}^{T} F_{o\mathbf{i}t}(b'_{\mathbf{i}t}(x_{ot})) \right) \end{aligned}$$

where :

$$\begin{aligned} F_{o\mathbf{i}}(b'_{\mathbf{i}}(x_{ot})) &\triangleq \sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t, \mathbf{i}) \log b'_{\mathbf{i}}(x_{ot}) \\ F_{o\mathbf{i}t}(b'_{\mathbf{i}t}(x_{ot})) &\triangleq \gamma(t, \mathbf{i}) \log b'_{\mathbf{i}t}(x_{ot}) \\ \gamma(t, \mathbf{i}) &\triangleq \sum_{\mathbf{x_h}_1^T} P(\mathbf{x_h}_1^T | \mathbf{x_o}_1^T, \Theta) \delta(\pi_{ot}, \mathbf{i}) \\ &= P(\pi_{ot} = \mathbf{i} | \mathbf{x_o}_1^T, \Theta) \end{aligned}$$

We consider Gaussian output densities, therefore the unknown parameters of the the densities $b'_{\mathbf{i}}(x_{ot})$ and $b'_{\mathbf{i}t}(x_{ot})$ are the means $\mu'_{\mathbf{i}}$, $\mu'_{\mathbf{i}}[t]$ and covariances $\Sigma'_{\mathbf{i}}$, $\Sigma'_{\mathbf{i}}[t]$. The maximizing parameters are obtained by setting the relative derivatives to zero. The Gaussian density $b'_{\mathbf{i}}(x_{ot})$ is expressed as :

$$\frac{1}{\sqrt{(2\pi)^D |\Sigma_{\mathbf{i}}|}} \exp\left(-\frac{1}{2} (x_{ot} - \mu_{\mathbf{i}})^* \Sigma_{\mathbf{i}}^{-1} (x_{ot} - \mu_{\mathbf{i}}) \right)$$

where * denotes matrix transposition. The derivation is more convenient if the derivatives are expressed with respect to $\Sigma_{\mathbf{i}}^{\prime -1}$. Using the relations :

$$\begin{aligned} \nabla_x (x^* \Sigma^{-1} x) &= 2\Sigma^{-1} x \\ \nabla_\Sigma \det(\Sigma) &= \det(\Sigma) \Sigma^{-1} \\ \nabla_{\Sigma^{-1}} (x^* \Sigma^{-1} x) &= x x^* \end{aligned}$$

one gets :

$$\begin{aligned} \nabla_{\mu'_{\mathbf{i}}} b'_{\mathbf{i}}(x_{ot}) &= b'_{\mathbf{i}}(x_{ot}) \Sigma_{\mathbf{i}}^{\prime -1} (x_{ot} - \mu'_{\mathbf{i}}) \\ \nabla_{\Sigma_{\mathbf{i}}^{\prime -1}} b'_{\mathbf{i}}(x_{ot}) &= \frac{1}{2} b'_{\mathbf{i}}(x_{ot}) \left(\Sigma'_{\mathbf{i}} - (x_{ot} - \mu'_{\mathbf{i}})(x_{ot} - \mu'_{\mathbf{i}})^* \right) \end{aligned}$$

and hence,

$$\begin{aligned}
\nabla_{\mu'_{\mathbf{i}}} F_{o\mathbf{i}}(b'_{\mathbf{i}}(x_{ot})) &= \sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i}) \Sigma'^{-1}_{\mathbf{i}} (x_{ot} - \mu'_{\mathbf{i}}) \\
&= \Sigma'^{-1}_{\mathbf{i}} \left(\sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i}) x_{ot} - \gamma(t,\mathbf{i}) \mu'_{\mathbf{i}} \right) \\
\nabla_{\Sigma'^{-1}_{\mathbf{i}}} F_{o\mathbf{i}}(b'_{\mathbf{i}}(x_{ot})) &= \frac{1}{2} \sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i}) \left(\Sigma'_{\mathbf{i}} - (x_{ot} - \mu'_{\mathbf{i}})(x_{ot} - \mu'_{\mathbf{i}})^* \right)
\end{aligned}$$

Setting the derivatives to zero we have

$$\begin{aligned}
\mu'_{\mathbf{i}} &= \frac{\sum_{t=\tau_p+1}^{T-\tau_f} x_{ot} \gamma(t,\mathbf{i})}{\sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i})} \\
\Sigma'_{\mathbf{i}} &= \frac{\sum_{t=\tau_p+1}^{T-\tau_f} x_{ot}{}^* x_{ot} \gamma(t,\mathbf{i})}{\sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i})} - \mu'^*_{\mathbf{i}} \mu'_{\mathbf{i}}
\end{aligned}$$

In the case of a single observation sequence the output density parameters of the initial and final networks $\mu'_{\mathbf{i}}[t]$ and $\Sigma'_{\mathbf{i}}[t]$ are estimated from a single observation, therefore

$$\mu'_{\mathbf{i}}[t] = x_{ot}, \;\; \Sigma'_{\mathbf{i}}[t] = 0, \;\; \text{for } 1 \le t \le \tau_p \text{ and } T - \tau_f + 1 \le t \le T$$

D.3 Mixture Output Densities

When the emission probabilities are represented with mixtures of densities, a new set of parameters Θ_c is introduced for mixture probabilities. The output densities now takes the following from :

$$\begin{aligned}
b_{\mathbf{i}}(x_{ot}) &\triangleq \sum_k c_{\mathbf{i}k} b_{\mathbf{i}kt}(x_{ot}), \quad \text{for} \quad \tau_p < t \le T - \tau_f \\
b_{\mathbf{i}t}(x_{ot}) &\triangleq \sum_k c_{\mathbf{i}k}[t] b_{\mathbf{i}kt}(x_{ot}), \quad \text{for} \quad 1 \le t \le \tau_p, \text{ and} \\
& \qquad\qquad\qquad\qquad\qquad\quad T - \tau_f < t \le T
\end{aligned}$$

We define a discrete mixture variable $X_c[t]$ that takes its values in the set of ordered labels $K = \{1, \ldots, M\}$. The set of parameters associated to mixtures $\Theta_c = (c_{\mathbf{i}k}, c_{\mathbf{i}k}[t])$ is defined as follows :

$$\begin{aligned}
c_{\mathbf{i}k} &\triangleq P(x_{ct} = k | \pi_{ot} = \mathbf{i}), \quad \text{for} \quad \tau_p < t \le T - \tau_f \\
c_{\mathbf{i}k}[t] &\triangleq P(x_{ct} = j | \pi_{ot} = \mathbf{i}), \quad \text{for} \quad 1 \le t \le \tau_p, \text{ and} \\
& \qquad\qquad\qquad\qquad\qquad\quad T - \tau_f < t \le T
\end{aligned}$$

In this case, the auxiliary function is written as :

$$Q(\Theta', \Theta) = \sum_{\mathbf{x_h}_1^T \mathbf{x_c}_1^T} P(\mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \mathbf{x_o}_1^T, \Theta) \log P(\mathbf{x_o}_1^T, \mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \Theta').$$

Analogous to single density case, the auxiliary function is separated into three terms for observable, hidden state and mixture variables.

$$Q(\Theta', \Theta) = Q_h(\Theta', \Theta_h) + Q_o(\Theta', \Theta_o) + Q_c(\Theta', \Theta_c)$$

The maximization of the first two terms is handled for each value of the mixture variable separately, as in the single density case. The last term is further separated into independent terms as follows :

$$\begin{aligned}
Q_c(\Theta, \Theta'_c) &= \sum_{\mathbf{x_h}_1^T \mathbf{x_c}_1^T} P(\mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ct} | \pi_{ot}) \\
&= \sum_{\mathbf{x_h}_1^T \mathbf{x_c}_1^T} P(\mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \mathbf{x_o}_1^T, \Theta) \sum_{t=1}^{T} \log P'(x_{ct} | \pi_{ot}) \sum_{\mathbf{i},k} \delta(x_{ct}, k) \delta(\pi_{ot}, \mathbf{i}) \\
&= \sum_{\mathbf{i},k} \sum_{t=1}^{T} \log c'_{\mathbf{i}k}[t] \sum_{\mathbf{x_h}_1^T \mathbf{x_c}_1^T} P(\mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \mathbf{x_o}_1^T, \Theta) \delta(x_{ct}, k) \delta(\pi_{ot}, \mathbf{i}) \\
&= \sum_{\mathbf{i}} \left(F_{c\mathbf{i}}(c'_{\mathbf{i}k}) + \sum_{t=1}^{\tau_p} F_{c\mathbf{i}t}(c'_{\mathbf{i}k}[t]) + \sum_{t=T-\tau_f+1}^{T} F_{c\mathbf{i}t}(c'_{\mathbf{i}k}[t]) \right)
\end{aligned}$$

where :

$$\begin{aligned}
F_{c\mathbf{i}}(c'_{\mathbf{i}k}) &\triangleq \sum_k \sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i},k) \log c'_{\mathbf{i}k} \\
F_{c\mathbf{i}t}(c'_{\mathbf{i}k}[t]) &\triangleq \sum_k \gamma(t,\mathbf{i},k) \log c'_{\mathbf{i}k}[t] \\
\gamma(t,\mathbf{i},k) &\triangleq \sum_{\mathbf{x_h}_1^T \mathbf{x_c}_1^T} P(\mathbf{x_h}_1^T, \mathbf{x_c}_1^T | \mathbf{x_o}_1^T, \Theta) \delta(x_{ct}, k) \delta(\pi_{ot}, \mathbf{i}) \\
&= P(\pi_{ot} = \mathbf{i}, x_{ct} = k | \mathbf{x_o}_1^T, \Theta)
\end{aligned}$$

Similar to state transition probabilities, the maximization of each $F_{c\mathbf{i}t}(c'_{\mathbf{i}k}[t])$ and $F_{c\mathbf{i}}(c'_{\mathbf{i}k})$ is carried out by means of Lagrange method to lead :

$$\begin{aligned}
c_{\mathbf{i}k} &= \frac{\sum_{t=\tau_p+1}^{T-\tau_f} \gamma(t,\mathbf{i},k)}{\sum_{t=\tau_p+1}^{T-\tau_f} \sum_{k=1}^{M} \gamma(t,\mathbf{i},k)} \\
c_{\mathbf{i}k}[t] &= \frac{\gamma(t,\mathbf{i},k)}{\sum_{k=1}^{M} \gamma(t,\mathbf{i},k)}
\end{aligned}$$

The re-estimation formulae for the parameters of the mixture components are analogous to those of single Gaussians, with $\gamma(t,\mathbf{i})$ replaced by $\gamma(t,\mathbf{i},k)$.

D.4 Summary

The formulation in the previous sections assumes a single observation sequence $O = \mathbf{x_o}_1^T$. In order to generalize to L observation sequences $\mathbf{O} = \{O_1, \ldots, O_L\}$, we assume that each sequence is independent. In this case, the auxiliary function can be written as a sum of the individual auxiliary functions of the sequences. Applying the previous derivations we have the following set of re-estimation formulae :

$$\begin{aligned} \xi(t,j,\mathbf{i},l) &= P(X_h[t]=j,\Pi_h[t]=\mathbf{i}|O_l) \\ \gamma(t,\mathbf{i},k,l) &= P(\Pi_o[t]=\mathbf{i},X_c[t]=k|O_l) \end{aligned} \tag{D.1}$$

$$a_{\mathbf{i}j} = \frac{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\xi(t,j,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\sum_{j=1}^{N}\xi(t,j,\mathbf{i},l)} \tag{D.2}$$

$$c_{\mathbf{i}k} = \frac{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\sum_{t=\kappa+1}^{T_l}\sum_{k=1}^{M}\gamma(t,\mathbf{i},k,l)} \tag{D.3}$$

$$\begin{aligned} \mu_{\mathbf{i}k} &= \frac{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}x_{ot}^{l}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\gamma(t,\mathbf{i},k,l)} \\ \Sigma_{\mathbf{i}k} &= \frac{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}{x_{ot}^{l}}^{*}x_{ot}^{l}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\sum_{t=\tau_p+1}^{T_l-\tau_f}\gamma(t,\mathbf{i},k,l)} - \hat{\mu}_{\mathbf{i}k}^{*}\hat{\mu}_{\mathbf{i}k} \end{aligned} \tag{D.4}$$

$$a_{\mathbf{i}j}[t] = \frac{\sum_{l=1}^{L}\xi(t,j,\mathbf{i},l)}{\sum_{l=1}^{L}\sum_{j=1}^{N}\xi(t,j,\mathbf{i},l)} \tag{D.5}$$

$$c_{\mathbf{i}k}[t] = \frac{\sum_{l=1}^{L}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\sum_{k=1}^{M}\gamma(t,\mathbf{i},k,l)} \tag{D.6}$$

$$\begin{aligned} \mu_{\mathbf{i}k}[t] &= \frac{\sum_{l=1}^{L}x_{ot}^{l}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\gamma(t,\mathbf{i},k,l)} \\ \Sigma_{\mathbf{i}k}[t] &= \frac{\sum_{l=1}^{L}{x_{ot}^{l}}^{*}x_{ot}^{l}\gamma(t,\mathbf{i},k,l)}{\sum_{l=1}^{L}\gamma(t,\mathbf{i},k,l)} - \hat{\mu}_{\mathbf{i}k}[t]^{*}\hat{\mu}_{\mathbf{i}k}[t] \end{aligned} \tag{D.7}$$

Annexe E

Hidden Markov Trees

Le modèle d'arbre de Markov caché (en anglais Hidden Markov Tree model ou HMT) a été développé par Crouse, Nowak et Baraniuk (1998)[23], pour la modélisation des dépendances statistiques entre les coefficients d'une transformée en ondelettes, pour laquelle des variables sont organisées dans une structure arborescente normale.

E.1 Les modèles d'arbres de Markov cachés

Soit $H = (H_1, \ldots, H_N)$ une séquence de données observées. Elle est supposée être indexée comme un arbre de N noeuds enraciné en H_1. On suppose que chaque noeud non terminal a au moins deux fils et que la longueur du chemin joignant n'importe quel noeud terminal (feuille) à la racine est une constante, appelée la profondeur d'arbre et notée J_0. On dit que ces variables définissent un arbre de Markov caché si et seulement si elles accomplissent les cinq conditions :s

- $\forall i \in \{1, \ldots, N\}$, H_i résulte d'une distribution de mélange avec une densité :

$$P(H_i = h) = \sum_{m=1}^{M} P(S_i = m) P(h | S_i = m)$$

  tel que S est une variable discrète à M états, notés $\{1, \ldots, M\}$.
- $(S_1, \ldots, S_N)$ a la même structure d'indexation que H. Ainsi on peut l'indexer comme un arbre enraciné en S_1. Pour chaque $i \in \{2, \ldots, N\}$ on dénote le parent de S_i par $S_{p(i)}$. Si S_i n'est pas une feuille, on dénote ses fils par $(S^i_{c_1}, \ldots, S^i_{c_{n_i}})$ tel que n_i représente le nombre de fils du noeud S_i. On dénote aussi l'ensemble des indices des fils par $c(i)$.
- $P(S_i | S_{i'} / i' \neq i) = P(S_i | S_{p(i)}, S^i_{c_1}, \ldots, S^i_{c_{n_i}})$ (propriété d'arbre de Markov),
- $P(H_1, \ldots, H_N | S_1, \ldots, S_N) = \prod_{i=1}^{i} P(H_i | S_1, \ldots, S_N)$
- $\forall i \in \{1, \ldots, N\} \qquad P(H_i | S_1, \ldots, S_N) = P(H_i | S_i)$

Nous nous référons aux deux dernières propriétés en tant que les propriétés conditionnelles d'independance. On choisit de représenter les relations d'indépendance entre les variables par un graphe acyclique dirigé pour modéliser la paramétrisation originale de Crouse, Nowak et Baraniuk(1998)[23] (figure E.1).

Un modèle d'arbre de Markov caché $(H_1, \ldots, H_N, S_1, \ldots, S_N)$ est caractérisé par les paramètres suivants :

1. La distribution du noeud racine S_1, $\pi = (\pi_m)_{m \in \{1, \ldots, M\}}$ où

$$\pi_m = P(S_1 = m)$$

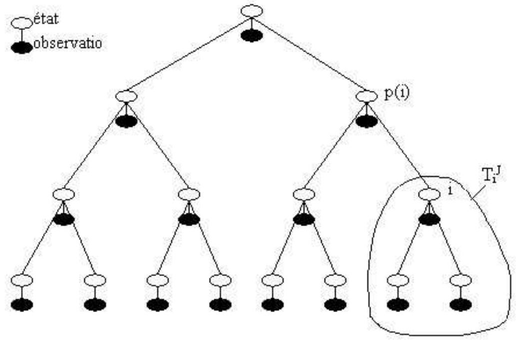


FIG. E.1 – Modèle d'arbre de Markov caché : Chaque nœud noir représente l'observation et chaque nœud blanc représente l'état de l'observation. Pour modéliser les dépendances entre les cœfficients, on met des liens entre les états cachés

2. Les probabilités de transition $A = (a^j_i(n, m))$, où $i \in \{1, \ldots, N/2^j\}$, $j \in \{1, \ldots, J_0\}$, $n \in \{1, \ldots, M\}$, $m \in \{1, \ldots, M\}$, définies par :

$$a^j_i(n, m) = P(S^j_i = m | S^{j+1}_{p(i)} = n)$$

3. La probabilité conditionnelle d'émettre h^j_i étant donné m l'état occupé,

$$b^j_i(m, \cdot) = P(h^j_i = \cdot | S^j_i = m)$$

   tel que b^j_i appartient à une famille de distribution paramétrique. Si on suppose que b^j_i est une densité de distribution gaussienne, alors $b = (\mu, \sigma)$ dénote la moyenne et la matrice de covariance de la distribution gaussienne.

L'ensemble de ces paramètres sera regroupé dans le vecteur $\Theta = (\pi, A, \mu, \sigma)$.

E.2 Evaluation de la Vraissemblance

On considère la séquence des cœfficients de détail observés $H = [H^1 \ldots H^{J_0}]$ avec $H^j = [h^j_1 h^j_2 \ldots h^j_{N/2^j}]$ la séquence des coefficients de détails observés à l'échelle j.

On associe à chaque cœfficient une variable d'état cachée $S = [S^1 \ldots S^{J_0}]$ avec $S^j = [S^j_1 S^j_2 \ldots S^j_{N/2^j}]$ À l'échelle j on a $\frac{N}{2^j}$ nœuds $S^j_i \in \{1, \ldots, M\}$ pour $i \in \{1, \ldots, \frac{N}{2^j}\}$, chaque nœud représente l'état caché derrière le cœfficient du détail h^j_i.

Calculer la vraisemblance de la séquence observée H par rapport au modèle Θ revient à sommer toute les probabilités d'émission de cette séquence le long de tous les chemins possibles. En effet le nombre de chemins possibles étant fini, la probabilité de génération d'une séquence d'observation $H^1, \ldots, H^{J_0}$ par le modèle Θ vaut donc : $P_\Theta(H) = \sum_S P(H, S|\Theta)$ où Θ est le modèle associé à un mot ou une phrase, H l'observation acoustique et S un chemin possible.

$$P_\Theta(H) = \sum_S P(H, S|\Theta) = \sum_S P(S|\Theta)P(H|S, \Theta) \tag{E.1}$$

Par application de la règle de Bayes[14] et des propriétés conditionnelles d'indépendance du HMM et après réarrangement, l'équation E.1 devient :

$$P_\Theta(H) = \sum_S \pi_{S_1^{J_0}} P(h_1^{J_0}|S_1^{J_0}) \prod_{j=1}^{J_0-1} \prod_{i=1}^{\frac{N}{2^j}} a_i^j(S_{p(i)}^{j+1}, S_i^j) b_i^j(S_i^j, h_i^j). \tag{E.2}$$

Ainsi, pour obtenir la vraisemblance on doit développer la somme sur tous les chemins possibles des états cachés S, ce qui n'est pas toujours évident si le nombre des états est grand. Toutefois, cette vraisemblance peut être calculée à l'aide de l'algorithme *Upward-Downward*.

E.3 L'algorithme Upward-Downward

On introduit les notations suivantes :

- T_i^j est le sous arbre enraciné au nœud i de l'échelle j. T_1 est l'arbre entier.
- $\hat{T}_i^j$ est le complèment de T_i^j dans T_1, i.e. l'ensemble des noeuds de T_1 qui n'appartiennent pas à T_i^j.

L'étape Upward consiste à calculer la probabilité conditionnelle de chaque sous arbre T_i^j de l'arbre entier T_1, en commençant par les noeuds feuilles. L'étape Downward consiste à calculer la probabilité jointe de T_1 tel que chaque sous arbre a été enlevé alternativement, en commençant par le noeud racine. À chaque noeud de l'arbre, on définit les variables suivantes [23] :

$$\begin{aligned}
\alpha_i^j(m) &= P(S_i^j = m, \hat{T}_i^j) \\
\beta_i^j(m) &= P(T_i^j | S_i^j = m) \\
\beta_{i,p(i)}^j(n) &= P(T_i^j | S_{p(i)}^{j+1} = n) \\
\beta_{p(i)\backslash i}^{j+1}(n) &= \frac{\beta_{p(i)}^{j+1}(n)}{\beta_{i,p(i)}^j(n)}
\end{aligned}$$

pour chaque $m \in \{1, \ldots, M\}$. La vraissemblance des coefficients d'ondelettes observés est alors :

$$\begin{aligned}
P_\Theta(H) &= \sum_{m=1}^M P(S_i^j = m, H) \\
&= \sum_{m=1}^M P(S_i^j = m, T_i^j, \hat{T}_i^j) \\
&= \sum_{m=1}^M P(T_i^j | S_i^j = m, \hat{T}_i^j) P(S_i^j = m, \hat{T}_i^j) \\
&= \sum_{m=1}^M P(T_i^j | S_i^j = m) P(S_i^j = m, \hat{T}_i^j)
\end{aligned}$$

[14] $P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$

À partir des définitions des probabilités Upward et Downward, on peut écrire :

$$P_\Theta(H) = \sum_{m=1}^M \beta_i^j(m) \alpha_i^j(m) \tag{E.3}$$

Enfin, pour obtenir la probabilité d'émission d'une observation, nous sommons le produit de α et β à un noeud donné i de l'échelle j. On remarque ici que la partie gauche de cette équation ne dépend ni du noeud i ni de son échelle j, ainsi, la vraisemblance peut être obtenue à partir de n'importe quel noeud de l'arbre. Pour faciliter les calculs on peut penser à utiliser le noeud racine ou les noeuds feuilles, c'est à dire :

$$P_\Theta(H) = \sum_{m=1}^M \alpha_i^1(m) b_i^1(m, h_i^1)$$

ou

$$P_\Theta(H) = \sum_{m=1}^M \pi_m \beta_1^{J_0}(m)$$

E.3.1 L'étape Upward

La variable Upward d'un couple nœud/état donné désigne la probabilité conditionnelle d'émettre les données du sous arbre attaché à ce nœud étant donné son état :

$$\begin{aligned}
\beta_i^j(n) &= P(T_i^j | S_i^j = n) \\
&= \{ \prod_{k \in c(i)} P(T_k^{j-1} | S_i^j = n) \} P(h_i^j | S_i^j = n) \\
&= \{ \prod_{k \in c(i)} \beta_{k,i}^{j-1}(n) \} b_i^j(n, h_i^j);
\end{aligned} \tag{E.4}$$

$$\begin{aligned}
\beta_{i,p(i)}^j(n) &= P(T_i^j | S_{p(i)}^{j+1} = n) \\
&= \sum_{m=1}^M P(T_i^j, S_i^j = m | S_{p(i)}^{j+1} = n) \\
&= \sum_{m=1}^M P(T_i^j | S_i^j = m) P(S_i^j = m | S_{p(i)}^{j+1} = n) \\
&= \sum_{m=1}^M \beta_i^j(m) a_i^j(n, m)
\end{aligned} \tag{E.5}$$

Notons ici que la probabilité Upward peut être calculée, efficacement et récursivement, en commençant par l'échelle la plus fine.
On note par :

$$\beta_{p(i)\backslash i}^{j+1}(n) = \frac{\beta_{p(i)}^{j+1}(n)}{\beta_{i,p(i)}^j(n)} \tag{E.6}$$

L'étape Upward sera donc décrite par l'algorithme suivant :

Algorithme UP step

1. Initialisation

   À l'échelle la plus fine $j = 1$, on calcule :

   $$\beta_i^1(m) = b_i^1(m, h_i^1); \qquad \forall m \in [1, M] \qquad et \qquad \forall i \in [1, N/2]$$

2. Induction
   tant que $j < J_0$
   pour $i = 1$ à $\frac{N}{2^j}$
   pour $m = 1$ à M

   $$\begin{aligned} \beta_{i,p(i)}^j(m) &= \sum_{n=1}^{M} a_i^j(m,n)\beta_i^j(n); \\ \beta_{p(i)}^{j+1}(m) &= b_{p(i)}^{j+1}(m, h_{p(i)}^{j+1}) \prod_{k \in c(p(i))} \beta_{k,p(i)}^j(m); \\ \beta_{p(i)\backslash i}^{j+1}(m) &= \frac{\beta_{p(i)}^{j+1}(m)}{\beta_{i,p(i)}^j(m)}; \end{aligned}$$

   fin pour
   fin pour
   $j = j + 1;$
   fin tant que.

E.3.2 L'étape Downward

La variable Downward, pour un nœud dans l'arbre et un état de ce nœud, représente la probabilité jointe d'observer l'ensemble des données depuis la racine à l'exception de la partie des données relative au sous arbre enraciné à ce nœud.

$$\begin{aligned} \alpha_i^j(m) &= P(S_i^j = m, \hat{T}_i^j) \\ &= \sum_{n=1}^{M} P(S_i^j = m, S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, H(T_k^j), h_{p(i)}^{j+1}) \\ &= \sum_{n=1}^{M} P(S_i^j = m | S_{p(i)}^{j+1} = n) \frac{P(T_{p(i)}^{j+1} | S_{p(i)}^{j+1} = n)}{P(T_i^j | S_{p(i)}^{j+1} = n)} \times P(S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}) \\ &= \sum_{n=1}^{M} \frac{a_i^j(n,m)\beta_{p(i)}^{j+1}(n)\alpha_{p(i)}^{j+1}(n)}{\beta_{i,p(i)}^j(n)} \end{aligned}$$

Et en utilisant l'equation (E.6) la variable forward sera donnée par :

$$\alpha_i^j(m) = \sum_{n=1}^{M} a_i^j(n,m)\alpha_{p(i)}^{j+1}(n)\beta_{p(i)\backslash i}^{j+1}(n)$$

Ainsi, $\alpha_i^j(m)$ peut se déduire récursivement à partir des probabilités du noeud $p(i)$ à l'échelle $j+1$ de tous les états $S_{p(i)}^{j+1}$ arrivant à l'état S_i^j. L'étape Downward est en effet décrite par l'algorithme

ci-dessous :

Algorithme Down step

1. Initialisation
   À l'échelle $j = J_0$, on pose :

   $$\alpha_1^{J_0}(m) = \pi_m; \qquad \forall m \in [1, M]$$

2. Induction
   soit $j = j - 1$;
   tant que $j >= 1$
   pour $i = 1$ à $\frac{N}{2^j}$
   pour $m = 1$ à M

   $$\alpha_i^j(m) = \sum_{n=1}^{M} a_i^j(n,m)\alpha_{p(i)}^{j+1}(n)\beta_{p(i)\backslash i}^{j+1}(n);$$

   fin pour
   fin pour
   $j = j - 1;$
   fin tant que.

E.4 Estimation des paramètres par l'algorithme EM

L'algorithme EM maximise le logarithme de la fonction de vraisemblance de données observables $P(H|\Theta)$ par maximisation itérative de l'espérance de la fonction de vraisemblance des données complètes $P(H, S|\Theta)$, exprimée par une fonction auxiliaire $Q(\Theta, \Theta^l)$, où Θ désigne l'ensemble des paramètres à estimer à l'itération courante et Θ^l désigne l'ensemble des paramètres estimés à l'itération précédente l :

$$\hat{\Theta} = arg \max_{\Theta} Q(\Theta, \Theta^l) = arg \max_{\Theta} E_S[\log P(H, S|\Theta)|H, \Theta^l].$$

La fonction auxiliaire $Q(\Theta, \Theta^l)$ peut être exprimée de la façon suivante :

$$\begin{aligned} Q(\Theta, \Theta^l) &= E_S[\ln P(H, S|\Theta)|H, \Theta^l] \\ &= \sum_S P(S|H, \Theta^l) \ln P(H, S|\Theta) \end{aligned}$$

En introduisant la formule de $P(H, S|\Theta)$ touvée dans l'équation E.2, on écrit donc :

$$Q(\Theta, \Theta^l) = \sum_S P(S|H, \Theta^l) \left[\ln \pi_{S_1^{J_0}} + \sum_{j=1}^{J_0 - 1} \sum_{i=1}^{N/2^j} \ln a_i^j(S_{p(i)}^{j+1}, S_i^j) + \sum_{j=1}^{J_0} \sum_{i=1}^{N/2^j} \ln b_i^j(S_i^j, h_i^j) \right]. \tag{E.7}$$

Le nom de l'algorithme EM vient du fait que chaque itération l est divisée en deux étapes :

E-step calcul de l'espérance $Q(\Theta, \Theta^l)$;

M-step maximisation de l'espérance : $\Theta^{l+1} = arg \max_{\Theta} Q(\Theta, \Theta^l)$

M-step

À l'étape de maximisation, ou 'M-step', la fonction Q est maximisée, ainsi, les paramétres Θ^l sont réestimés pour obtenir Θ^{l+1}. En augmentant le nombre d'itérations, on atteint des éstimations plus fiables des paramètres[93]. D'après l'équation (E.7), on remarque que les paramètres qu'on veut estimer sont séparés en trois termes de la somme, on peut donc optimiser chaque terme indépendamment des autres.
Dans le cadre de notre approche, pour la maximisation de ces termes, on considère les hypothéses suivantes :

1. On suppose que J_0 est la profondeur de l'arbre T_1 où la racine de l'arbre est à l'échelle 1 et les feuilles sont à l'échelle J_0 ;
2. On suppose que les probabilités de transition sont indépendantes de la position des nœuds dans l'arbre.
$$a_i^j(k,l) = a(k,l) \quad \forall i,j$$
3. Chaque état est modélisé par une gaussienne de moyenne $\mu_{j,n}$ et de variance $\sigma^2_{j,n}$. On suppose que les probabilités d'émission sont égales pour tous les nœuds i à l'échelle j, pour un état donné n ;
$$\begin{aligned} b_i^j(n,.) &= b^j(n,.) \\ &= g(H^j;\mu_{j,n},\sigma^2_{j,n}) \\ &= \frac{1}{\sqrt{2\pi}\sigma_{j,n}} \exp\Big\{-\frac{(H^j-\mu_{j,n})^2}{2\sigma^2_{j,n}}\Big\} \end{aligned}$$

Ces hypothèses ne changent en rien l'algorithme générale.

Probabilités du noeud racine

Le premier terme de l'équation (E.7) devient :

$$\sum_{S\in\{1,\ldots,M\}^{N-1}} P(S|H,\Theta^l) \ln \pi^{J_0}_{S_1} = \sum_{m=1}^{M} P(S_1^{J_0}=m|H,\Theta^l)\ln \pi_m$$

Pour trouver l'estimation de π_m, on utilise le multiplicateur de Lagrange λ avec la contrainte $\sum_m \pi_m = 1$, et résoudre l'équation suivante :

$$\frac{\partial}{\partial \pi_m}\left[\sum_{m=1}^{M} P(S_1^{J_0}=m|H,\Theta^l)\ln\pi_m + \lambda\left(\sum_{m=1}^{M}\pi_m - 1\right)\right] = 0$$

$$P(S_1^{J_0}=m|H,\Theta^l) + \lambda\pi_m = 0$$

$$\sum_{m=1}^{M} P(S_1^{J_0}=m|H,\Theta^l) + \lambda\sum_{m=1}^{M}\pi_m = 0$$

on trouve ainsi :

$$\pi_m = \frac{P(S_1^{J_0}=m,H|\Theta^l)}{P(H|\Theta^l)} \tag{E.8}$$

Probabilités de transition

Le second terme de l'équation (E.7) devient :

$$\sum_S P(S|H,\Theta^l)\sum_{j=1}^{J_0-1}\sum_{i=1}^{N/2^j}\ln a(S_i^j, S_{p(i)}^{j+1})$$

$$= \sum_{n=1}^{M}\sum_{m=1}^{M}\sum_{j=1}^{J_0-1}\sum_{i=1}^{N/2^j} P(S_i^j=m, S_{p(i)}^{j+1}=n|H,\Theta^l)\ln a(n,m)$$

De la même manière, on peut utiliser le multiplicateur de Lagrange λ avec la contrainte $\sum_{m=1}^{M} a(n,m) = 1$ pour trouver :

$$\frac{\partial}{\partial a(n,m)}\left[\sum_{n,m}\sum_{i,j} P(S_i^j=m,S_{p(i)}^{j+1}=n|H,\Theta^l)\ln a(n,m) + \lambda\left(\sum_{m=1}^{M} a(n,m)-1\right)\right] = 0$$

$$\sum_{j=1}^{J_0-1}\sum_{i=1}^{N/2^j} P(S_i^j=m,S_{p(i)}^{j+1}=n|H,\Theta^l) + \lambda a(n,m) = 0$$

$$\sum_{m=1}^{M}\sum_{j=1}^{J_0-1}\sum_{i=1}^{N/2^j} P(S_i^j=m,S_{p(i)}^{j+1}=n|H,\Theta^l) + \lambda\sum_{m=1}^{M} a(n,m) = 0$$

$$\lambda = -\sum_{j=1}^{J_0-1}\sum_{i=1}^{N/2^j} P(S_{p(i)}^{j+1}=n|H,\Theta^l)$$

$$a(n,m) = \frac{\sum_{j=1}^{J_0-1}\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j=m,S_{p(i)}^{j+1}=n|H,\Theta^l)}{\sum_{j=1}^{J_0-1}\sum_{i=1}^{\frac{N}{2^j}} P(S_{p(i)}^{j+1}=n|H,\Theta^l)} \tag{E.9}$$

Probabilités d'émission

Le troisième terme de l'équation (E.7) devient :

$$\sum_S P(S|H,\Theta^l)\sum_{j=1}^{J_0}\sum_{i=1}^{\frac{N}{2^j}}\ln b^j(S_i^j,h_i^j)$$

$$= \sum_{n=1}^{M}\sum_{j=1}^{J_0}\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j=n|H,\Theta^l)\ln b^j(n,h_i^j) \tag{E.10}$$

Dans le cas d'une distribution gaussienne, on a :

$$b^j(n,h_i^j) = \frac{1}{\sqrt{2\pi}\sigma_{j,n}}\exp\Big\{-\frac{(h_i^j-\mu_{j,n})^2}{2\sigma^2_{j,n}}\Big\}$$

L'estimation pour chaque paramètre $(\mu_{j,n},\sigma^2_{j,n})$, qui maximise $Q(\Theta,\Theta^l)$, est obtenue en dérivant la fonction (E.10) sur chacun de ces paramètres et en mettant la dérivée égale à zéro, on obtient :

$$\frac{\partial}{\partial \mu_{j,n}}\left[\sum_{n=1}^{M}\sum_{j=1}^{J_0}\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j = n|H,\Theta^l)\Big\{\ln\frac{1}{\sqrt{2\pi}\sigma_{j,n}} - \frac{(h_i^j-\mu_{j,n})^2}{2\sigma_{j,n}^2}\Big\}\right] = 0$$

$$\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j = n|H,\Theta^l)(h_i^j - \mu_{j,n}) = 0$$

$$\mu_{j,n} = \frac{\sum_{i=1}^{N/2^j} h_i^j P(S_i^j = n|H,\Theta^l)}{\sum_{i=1}^{N/2^j} P(S_i^j = n|H,\Theta^l)} \tag{E.11}$$

et

$$\frac{\partial}{\partial \sigma^2_{j,n}}\left[\sum_{n=1}^{M}\sum_{j=1}^{J_0}\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j = n|H,\Theta^l)\Big\{\ln\frac{1}{\sqrt{2\pi}\sigma_{j,n}} - \frac{(h_i^j-\mu_{j,n})^2}{2\sigma_{j,n}^2}\Big\}\right] = 0$$

$$\sum_{i=1}^{\frac{N}{2^j}} P(S_i^j = n|H,\Theta^l)\Big\{-\frac{1}{2\sigma_{j,n}^2} + \frac{(h_i^j-\mu_{j,n})^2}{2\sigma_{j,n}^4}\Big\} = 0$$

$$\sigma^2_{j,n} = \frac{\sum_{i=1}^{N/2^j} P(S_i^j = n|H,\Theta^l)\left(h_i^j - \mu_{j,n}\right)^2}{\sum_{i=1}^{N/2^j} P(S_i^j = n|H,\Theta^l)}$$

$$\sigma^2_{j,n} = \frac{\sum_{i=1}^{N/2^j} P(S_i^j = n|H,\Theta^l)\left(h_i^j\right)^2}{\sum_{i=1}^{N/2^j} P(S_i^j = n|H,\Theta^l)} - \mu^2_{j,n} \tag{E.12}$$

E-step pour une seule observation

Pour calculer ces Maximisations on a besoin de connaître les probabilités des états cachés $P(S|H,\Theta^l)$. Ces probabilités sont calculées à l'aide des variables Upward et Downward.
En combinant l'information des α et β, on obtient les pmfs conditionnelles de l'état de chaque cœfficient de détail dans l'arbre, par l'intermédiaire des équations suivantes :

$$P(S_i^j = m|H,\Theta) = \frac{P(S_i^j = m, H|\Theta)}{P(H|\Theta)}$$

$$\begin{aligned}
P(S_i^j = m, H|\Theta) &= P(S_i^j = m, T_i^j, \hat{T}_i^j|\Theta)\\
&= P(T_i^j|S_i^j = m, \hat{T}_i^j, \Theta)P(S_i^j = m, \hat{T}_i^j|\Theta)\\
&= P(T_i^j|S_i^j = m, \Theta)P(S_i^j = m, \hat{T}_i^j|\Theta)\\
&= \alpha_i^j(m)\beta_i^j(m)
\end{aligned}$$

D'après l'équation E.3, on peut écrire :

$$P(S_i^j = m|H,\Theta) = \frac{\alpha_i^j(m)\beta_i^j(m)}{\sum_{n=1}^{M}\alpha_i^j(n)\beta_i^j(n)} \tag{E.13}$$

De même, on a :

$$P(S_i^j = m, S_{p(i)}^{j+1} = n|H,\Theta) = \frac{P(S_i^j = m, S_{p(i)}^{j+1} = n, H|\Theta)}{P(H|\Theta)}$$

$$\begin{aligned}
&P(S_i^j = m, S_{p(i)}^{j+1} = n, H)\\
&= P(S_i^j = m, S_{p(i)}^{j+1} = n, T_i^j, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1}, T_k^j)\\
&= P(T_i^j|S_i^j = m, S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1}, T_k^j)P(S_i^j = m, S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1}, T_k^j)\\
&= \beta_i^j(m)P(S_i^j = m|S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1}, T_k^j)P(S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1}, T_k^j)\\
&= \beta_i^j(m)a_i^j(n,m)P(T_k^j|S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1})P(S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1}, h_{p(i)}^{j+1})\\
&= \beta_i^j(m)a_i^j(n,m)\beta_{k,p(i)}^j(n)P(h_{p(i)}^{j+1}|S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1})P(S_{p(i)}^{j+1} = n, \hat{T}_{p(i)}^{j+1})\\
&= \beta_i^j(m)a_i^j(n,m)\beta_{k,p(i)}^j(n)b_{p(i)}^{j+1}(n, h_{p(i)}^{j+1})\alpha_{p(i)}^{j+1}(n)\\
&= \beta_i^j(m)a_i^j(n,m)\alpha_{p(i)}^{j+1}(n)\beta_{p(i)\setminus i}^{j+1}(n).
\end{aligned}$$

Ainsi, on trouve :

$$P(S_i^j = m, S_{p(i)}^{j+1} = n|H,\Theta) = \frac{\beta_i^j(m)a_i^j(n,m)\alpha_{p(i)}^{j+1}(n)\beta_{p(i)\setminus i}^{j+1}(n)}{\sum_{n=1}^{M}\alpha_i^j(n)\beta_i^j(n)} \tag{E.14}$$

Résumé

Dans le cas le plus génèral, on manipule $K > 1$ arbres d'ondelettes $H^{(1)},\ldots,H^{(K)}$ avec $H^{(k)} = [H^{1(k)},\ldots,H^{J_0(k)}]$. Afin de construire le modèle HMT qui caractérise mieux les observations $H^{(1)},\ldots,H^{(K)}$, qu'on suppose génèrées indépendamment, on doit trouver le vecteur Θ qui maximise la fonction auxiliaire qu'on suppose dans ce cas donnée par la somme des fonctions auxiliaires de chacun des observations.
Les équations E.8, E.9, E.11 et E.12 deviennent :

$$\begin{aligned}
\pi_m &= \frac{1}{K}\sum_{k=1}^{K} P(S_1^{J_0(k)} = m|H^{(k)},\Theta^l)\\
a(n,m) &= \frac{\sum_{k=1}^{K}\sum_{j=1}^{J_0-1}\sum_{i=1}^{\frac{N}{2^j}} P(S_i^{j(k)} = m, S_{p(i)}^{j+1(k)} = n|H^{(k)},\Theta^l)}{\sum_{k=1}^{K}\sum_{j=1}^{J_0-1}\sum_{i=1}^{\frac{N}{2^j}} P(S_{p(i)}^{j+1(k)} = n|H^{(k)},\Theta^l)}\\
\mu_{j,n} &= \frac{\sum_{k=1}^{K}\sum_{i=1}^{N/2^j} h_i^{j(k)} P(S_i^{j(k)} = n|H^{(k)},\Theta^l)}{\sum_{k=1}^{K}\sum_{i=1}^{N/2^j} P(S_i^{j(k)} = n|H^{(k)},\Theta^l)}\\
\sigma^2_{j,n} &= \frac{\sum_{k=1}^{K}\sum_{i=1}^{N/2^j} P(S_i^{j(k)} = n|H^{(k)},\Theta^l)\left(h_i^{j(k)}\right)^2}{\sum_{k=1}^{K}\sum_{i=1}^{N/2^j} P(S_i^{j(k)} = n|H^{(k)},\Theta^l)} - \mu^2_{j,n}
\end{aligned}$$

Pour implèmenter le E-step à l'itèration l de l'algorithme EM, on applique l'algorithme upward-downward indépendamment à chacun des K arbres d'ondelettes. Dans ce cas, on peut écrire :

$$P(H^{(1)},\ldots,H^{(k)}|\Theta) = \prod_{k=1}^{K} P(H^{(k)}|\Theta)$$

En utilisant les paramètres estimés $\Theta = \Theta^l$, on calcule les probabilités $P(S_i^{j(k)} = m | H^{(k)}, \Theta^l)$ et $P(S_i^{j(k)} = m, S_{p(i)}^{j+1(k)} | H^{(k)}, \Theta^l)$ dont les expressions sont données par (E.13) et (E.14) pour chaque arbre.

Annexe F

t Tests

Statistical t tests are appropriate if one wants to test the significance of the mean (average) of a set (or sets) of numerical data. They are adapated to small sample sizes ($N < 30$). The basic assumption required for these tests is that the data comes from an underlying Normal distribution. There are various types of Student's t Tests, as outlined below.

Single sample t test Given a set of numerical data. The experimental hypothesis is of the form "the mean of this variable is different from (greater than, less than) a specified numerical value". If the test statistic exceeds the critical value, the conclusion is that the mean of the variable is significantly different from the specified value.

Two sample t test Given a sample of numerical data from population A, and a sample of similar data from population B. The two sample t test can establish whether one set has a significantly different (higher, lower) mean from the other set. If the test statistic exceeds the critical value, the conclusion is that the mean of the variable from A is significantly different from the mean of the variable from B.

In the case of the two sample t test, one needs to be clear whether the variances of set A and set B are known, or whether the pooled variance of both sets are estimated together. This influences the form of the actual test statistic used.

Paired t test As with the two sample t test, this addresses the question of whether the mean of one set is significantly different from the mean of another set. However, in this case the data values in sample A are take taken from the same respondents as in sample B. The data values are thus naturally paired. This makes the calculations easier, since they are based on the differences of each pair of data, which reduces the two data sets to one.

The paired t test can establish whether one data set has a significantly different (higher, lower) mean from the other set. If the test statistic exceeds the critical value, the conclusion is that the mean of the variable from A is significantly different from the mean of the variable from B.

F.1 The Student-t distribution

The Student-t distribution, is derived from the Normal distribution. It is the distribution of the (mean/standard deviation) of a sample of Normal distributed values with unknown variance.

This means that this distribution should be used when the test parameter has a Normal distribution and the variance is estimated from the same sample as the mean value is. The Student-t distribution varies with the number of items in the sample, more specific, with the number of independent values from which the variance is calculated. This number is called the Degrees of Freedom (DoF) of the distribution.

$$t = \frac{\mu}{\hat{\sigma}}\sqrt{N}$$

where μ : mean, $\hat{\sigma}$: estimated standard deviation, N : sample size.

$DegreesofFreedom$ = Number of independent values from which the variance is calculated.

Remarks : If the $DegreesofFreedom > 30$, the distribution of t can be approximated by a Standard Normal Distribution. Because of their popularity, tests based on this distribution are very often applied indiscriminately, when the underlying assumptions are invalid, i.e., when the observations are not Normal distributed. This can lead to illusory high sensitivities.

F.2 The Student-t Test for One Sample (paired t-test)

Characteristics : This is the standard test for matched-pairs. It is also the yard-stick for calculating the relative efficiency of other tests. The Student-t test is the most sensitive test for interval data, but it also requires the most stringent assumptions.

H0 : The mean value of the difference is zero.

Assumptions : The difference is Normal distributed. If there is any reason to doubt this assumption, use another, distribution-free, test (e.g., Wilcoxon Matched-Pairs Signed-Ranks Test).

Scale : Interval

Procedure : Calculate the Mean value (μ) and standard deviation (σ) of the differences, determine the number of sample pairs, N, and $DegreesofFreedom$ ($= N - 1$, in this case). The test parameter is

$$t = \frac{\mu}{\sigma}\sqrt{N}$$

If necessary, mean (μ) can be replaced by (Mean - expected value).

Level of Significance : The significance levels of t for different $DegreesofFreedom$ are tabulated.

Approximation : If the $DegreesofFreedom > 30$, the distribution of t can be approximated by a Standard Normal Distribution.

Remarks : Because of its popularity, this test is very often applied indiscriminately, when the underlying assumptions are invalid, i.e., when the observations are not Normal distributed. This can lead to illusory high sensitivities. Although it is the most powerfull test when the experimental data are indeed Normal distributed, do not hesitate to use a distribution-free test instead whenever there is some doubt about Normality (the Wilcoxon Matched-Pairs Signed-Ranks Test is almost as sensitive).

F.3 The Student-t Test for Two Samples

Characteristics : This is the standard test for two samples. It is also the yard-stick for calculating the relative efficiency of other tests. The Student-t test is the most sensitive test for interval data, but it also requires the most stringent assumptions.

H0 : Both populations have identical mean values. That is, the difference between the means is zero.

Assumptions : Both distributions are Normal distributed with identical variances. If there is any reason to doubt these assumptions, use another, distribution-free, test (e.g. the Wilcoxon Test).

Scale : Interval

Procedure : Calculate the Mean values (μ_1, μ_2) and standard deviations (σ_1 and σ_2) of both samples. Determine

$$\sigma_g = \sqrt{\frac{(N1-1)\sigma_1^2 + (N2-1)\sigma_2^2}{N1+N2-2}}$$

The test parameter is

$$t = \frac{\mu_1 - \mu_2}{\sigma_g\sqrt{\frac{1}{N1} + \frac{1}{N2}}}$$

The number of $DegreesofFreedom = N1 + N2 - 2$.

Level of Significance : The significance levels of t for different Degrees of Freedom are tabulated.

Approximation : If the $DegreesofFreedom > 30$, the distribution of t can be approximated by a Standard Normal Distribution.

Remarks : Because of its popularity, this test is very often applied indiscriminately, when the underlying assumptions are invalid, i.e., when the observations are not Normal distributed with equal variance. This can lead to illusory high sensitivities. Although it is the most powerful test when the experimental data are indeed Normal distributed with equal variance, do not hesitate to use a distribution-free test instead whenever there is some doubt about Normality (the Wilcoxon Test is almost as sensitive).

Bibliographie

[1] *Mathematical Foundations of Speech and Language Processing*, volume 138 of *Institute of Mathematical Analysis Volumes in Mathematics Series*, chapter Graphical Models and Automatic Speech Recognition. Springer-Verlag, 2003.

[2] J.B. Allen. How do humans process and recognize speech. *IEEE Trans. on Audio and Speech Processing*, 2(4) :567–577, 1994.

[3] J. Bilmes. Graphical models and automatic speech recognition. Technical Report UWEETR-2001-0005, Department of Electrical Engineering, University of Washington, November 2001.

[4] Jeff A. Bilmes. Buried markov models for speech recognition. In *ICASSP'99*, 1999.

[5] Jeff A. Bilmes. *Natural Statistical Models for Automatic Speech Recognition*. PhD thesis, International Compute Science Institute, Berkeley, California, October 1999.

[6] Jeff A. Bilmes. Dynamic bayesian multinets. In *UAI'2000*, 2000.

[7] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29 :213–244, 1997.

[8] B.P. Bogert, M.J. R. Healy, and J.W. Tukey. The quefrency alanysis of time series for echoes : Cepstrum, psuedo-autocovariance, cross-cepstrum and saphe cracking. In *Proceedings of the Symposium on Time Series Analysis*, 1963.

[9] P.F. Brown, V.J. DellaPietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. Class based n-gram models of natural language. *Computational Linguistics*, 18(4) :467–478, 1992.

[10] Wray Buntine. Theory refinement on bayesian networks. In *UAI'91*, pages 52–60, Los Angeles, 1991.

[11] Wray Buntine. A guide to literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering*, 8 :195–210, 1996.

[12] Enrique Castillo, José Manuel Gutiérrez", and Ali S. Hadi. *Expert Systems And Probabilistic Network Models*. Springer-Verlag, New York, 1997.

[13] C. Cerisara and D. Fohr. Multi-band automatic speech recognition. *Computer Speech Language*, 15 :151–174, 2001.

[14] O. Cetin, H. Nock, K. Kirchhoff, J. Bilmes, and M. Ostendorf. The 2001 gmtk-based spine asr system. In *International Conference on Spoken Language Processing*, 2002.

[15] Ozgur Cetin and Mari Ostendorf. Cross-stream observation dependencies for multi-stream speech recognition. In *Eurospeech*, 2003.

[16] C. P. Chan, P. C. Ching, and Tan Lee. Noisy speech recognition using de-noised multiresolution analysis acoustic features. *Journal of the Acoustical Society of America*, 110(5) :2567–2574, Nov. 2001.

[17] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 14 :359–394, 1999.

[18] C. Chesta, O. Siohan, and C.-H. Lee. Maximum a posteriori linear regression for hidden markov model adaptation. In *In Proceedings of European Conference on Speech Communication and Technology*, volume 1, pages 211–214, Budapest, Hungary, 1999.

[19] David M. Chickering, Dan Geiger, and David Heckerman. Learning bayesian networks : Search methods and experimental results. In *Proc. AI and Statistics*, pages 112–128, 1995.

[20] C.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42 :393–405, 1990.

[21] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9 :309–347, 1992.

[22] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[23] Matthew Crouse, Robert Nowak, and Richard Baraniuk. Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on Signal Processing*, 1997.

[24] K. Daoudi and M. Deviren. A new supervised-predictive compensation scheme for noisy speech recognition. In *Eurospeech*, 2003.

[25] K. Daoudi and M. Deviren. Une nouvelle architecture de compensation du bruit pour la reconnaissance robuste de la parole. In *XXVèmes journées études sur la parole (JEP)*, 2004.

[26] K. Daoudi, D.Fohr, and C. Antoine. Dynamic bayesian networks for multi-band automatic speech recognition. *Computer speech and Language (CSL)*, 2001.

[27] Khalid Daoudi, Dominique Fohr, and Christophe Antoine. A new approach for multi-band speech recognition based on probabilistic graphical models. In *ICSLP'2000*, 2000.

[28] A. Darwiche. A differential approach to inference in bayesian networks. In *UAI*, 2000.

[29] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, (4) :357–366, 1980. Mel-frequency cepstral coefficients (MFCC).

[30] A.P. Dawid. Application of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2 :25–36, 1992.

[31] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5 :142–150, 1989.

[32] R. Dechter. Bucket elimination : A unifying framework for probabilistic inference. In *UAI*, pages 211–219, 1996.

[33] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Roy. Stat. Soc. Ser.*, 39 :1–38, 1977.

[34] M. Deviren. Structural learning of dynamic Bayesian networks for speech recognition. Technical report, LORIA-Speech Group, 2001.

[35] M. Deviren. Dynamic bayesian networks for speech recogniton. In *Proceedings of AAAI 2002, SIGART/AAAI Doctoral Consortium*, Edmonton, Canada, 2002.

[36] M. Deviren and K. Daoudi. Structural learning of dynamic Bayesian networks in speech recognition. In *Eurospeech 2001*, Aalborg, Denmark, 2001.

[37] M. Deviren and K. Daoudi. Apprentissage de structure de réseaux bayésiens dynamiques pour la reconnaissance de la parole. In *XXIVèmes journées études sur la parole (JEP)*, 2002.

[38] M. Deviren and K. Daoudi. Continuous speech recognition using dynamic bayesian networks : A fast decoding algorithm. In *First European Workshop on Probabilistic Graphical Models (PGM'02)*, 2002.

[39] M. Deviren and K. Daoudi. Continuous speech recognition using structural learning of dynamic Bayesian networks. In *EUSIPCO*, Toulouse, France, 2002.

[40] M. Deviren and K. Daoudi. Frequency filtering or wavelet filtering ? In *Joint Intl. Conf. on Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*, 2003.

[41] M. Deviren and K. Daoudi. *Advances in Bayesian networks*, chapter Continuous speech recognition using dynamic Bayesian networks : A fast decoding algorithm, pages 289–307. Phsica Verlag (Springer), 2004.

[42] M. Deviren, K. Daoudi, and K. Smaïli. Language modeling using dynamic bayesian networks. In *Language resources and evaluation conference (LREC)*, 2004.

[43] M. Deviren, K. Daoudi, and K. Smaïli. Rethinking language models within the framework of dynamic bayesian networks. Preprint, 2004.

[44] M. Deviren, K. Daoudi, and K. Smaïli. Une nouvelle approche de modélisation du langage par des réseaux bayésiens dynamiques. In *XXVèmes journées études sur la parole (JEP)*, 2004.

[45] Ephraim92. Gain-adapted hidden markov models for recognition of clean and noisy speech. *IEEE Trans. Signal Processing*, 40 :1303–1316, 1992.

[46] E. Erzin, A.E. Cetin, and Y. Yardimci. Subband analysis for robust speech recognition in the presence of car noise. In *ICASSP*, 1995.

[47] O. Farooq and S. Datta. Mel filter-like admissible wavelet packet structure for speech recognition. *IEEE Signal Processing Letters*, 8(7) :196–198, July 2001.

[48] H. Fletcher. *Speech and Hearing in Communication*. Krieger, 1953.

[49] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Int. Conf. Machine Learning*, 1997.

[50] Nir Friedman. The bayesian structural em algorithm. In *UAI-98*, 1998. ../friedman/Fr2.ps.

[51] Nir Friedman, Moises Goldszmidt, and Thomas J. Lee. Bayesian network classification with continuous attributes : Getting the best of both discretization and parametric fitting. In *Fifteenth Int. Conf. on Machine Learning, ICML-98*, 1998.

[52] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *UAI'98*, Madison, Wisconsin, 1998.

[53] M.J.F. Gales. *Model based techniques for noise robust speech recognition.* PhD thesis, Cambridge university, September 1995.

[54] M.J.F Gales. Predictive model-based compensation schemes for robust speech recognition. *Speech Communication*, 25(1-3) :49–74, 1998.

[55] J.L. Gauvain and C.H. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Trans. Speech and Audio Processing*, 2 :291–298, 1996.

[56] Dan Geiger and David Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. Technical Report MSR-TR-98-67, Microsoft Research, Advanced Technology Division, 1999. ../geiger/tr-98-67.ps.

[57] R. Gemello, D. Albesano, L. Moisa, and R. De Mori. Integration of fixed and multiple resolution analysis in a speech recognition system. In *ICASSP*, 2001.

[58] Zoubin Ghahramani and Michael Jordan. Factorial hidden markov models. *Machine Learning*, 29 :245–273, 1997. ../ghahramani/fhmmML.ps.

[59] Sana Ghouzali. Reconnaissance automatique de la parole a l'aide de models markov caches sur les arbes d'ondolettes. Master's thesis, INRIA, 2003.

[60] Yifan Gong. Speech recognition in noisy environments : a survey. *Speech communication*, 16 :261–291, 1995.

[61] J.N. Gowdy and Z. Tufekci. Mel-scaled discrete wavelet coefficients for speech recognition. In *ICASSP*, 2000.

[62] H. Guo and William H. Hsu. A survey of algorithms for real-time bayesian network inference. In *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, 2002.

[63] Maya Gupta and Anna Gilbert. Robust speech recognition using wavelet coefficient features. In *ASRU*, 2001.

[64] David Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, March 1995. ../heckerman/tr-95-06.ps.

[65] David Heckerman and Dan Geiger. Learning bayesian networks. Technical Report MSR-TR-95-02, Microsoft Research, Advanced Technology Division, 1995. ../heckerman/tr-95-02.ps.

[66] David Heckerman, Dan Geiger, and David. M. Chickering. Learning bayesian networks : The combination of knowledge and statistical data. *Machine Learning*, 20 :197–243, 1995. ../heckerman/tr-94-09.ps.

[67] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Trans. Speech and Audio Processing*, 2 :578–589, 1994.

[68] H.G. Hirsch and D. Pearce. The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ISCA ITRW ASR2000*, Paris, France, September 2000.

[69] Firas Jabloun and A. Enis Cetin. The Teager energy based feature parameters for robust speech recognition in car noise. In *ICASSP*, 1999.

[70] Salma Jamoussi, Kamel Smaili, and Jean-Paul Haton. Understanding process for speech recognition. In *Eighth European Conference on Speech Communication and Technology - EuroSpeech 2003, Geneva, Switzerland*, Sep 2003.

[71] Léonard Janer, Josep Martí, Climent Nadeu, and Eduardo Lleida-Solano. Wavelet transforms for non-uniform speech recogntion systems. In *ICSLP*, 1996.

[72] F. Jelinek. *Self-organized language modeling for speech recognition*, pages 450–506. Morgan Kaufmann, 1989.

[73] F. Jelinek and R.L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Pattern Recognition in Practice*, pages 381–397, Amsterdam, Holland, 1980.

[74] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in recursive graphical models by local computations. *Computational Statistics and Data Analysis*, 4 :269–282, 1990.

[75] J.C. Junqua. The lombard reflex and its role on human listeners and automatic speech recognizers. *Journal of Acoustics Society of America*, 1993.

[76] Katrin Keller, Souheil Ben-Yacoub, and Chafic Mokbel. Combining wavelet-domain hidden markov trees with hidden markov models. IDIAP Technical Report, August 1999.

[77] P. Kenny, M. Lennig, and P. Mermelstein. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 38(2) :220–225, February 1990.

[78] Uffe Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 121–129, San Mateo, 1992. Morgan Kaufmann.

[79] Filipp Korkmazsky, Murat Deviren, Dominique Fohr, and Irina Illina. Hidden factor dynamic bayesian networks for speech recognition. In *Intl. Conf. on Spoken Language Processing (ICSLP)*, 2004.

[80] M. Krishnan, C. Neophytou, and G. Prescott. Wavelet transform speech recognition using vector quantization, 1994.

[81] David Kryze, Luca Rigazio, Ted Applebaum, and Jean-Claude Junqua. A new noise-robust subband front-end and its comparison to plp. International Workshop on Automatic Speech Recognition and Understanding (ASRU'99), 1999.

[82] Wai Lam and Fahiem Bacchus. Learning bayesian belief networks an approach based on the mdl principle. *Computational Intelligence*, 10(4) :269–293, 1994. ../lam/LBCL94.ps.

[83] Steffen L. Lauritzen. The em algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19 :191–201, 1995.

[84] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(11) :1641–1648, November 1989.

[85] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hmms. *Computer Speech and Language*, 9 :171–186, 1995.

[86] Z. Li and B. D'Ambrosio. Efficient inference in bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1) :55–81, 1994.

[87] Karen Livescu, James Glass, and Jeff Bilmes. Hidden feature models for speech recognition using dynamic bayesian networks. In *Eurospeech*, 2003.

[88] B. Logan and P. Moreno. Factorial HMMs for acoustic modeling. In *ICASSP*, 1998.

[89] C. Long and S. Datta. Wavelet based feature extraction for phoneme recognition. In *Proc. ICSLP '96*, volume 1, pages 264–267, Philadelphia, PA, 1996.

[90] Christopher John Long and Sekharajit Datta. Discriminant wavelet basis construction for speech recognition. In *ICSLP*, 1998.

[91] E. Lukasik. Wavelet packets based features selection for voiceless plosives classification. In *ICASSP*, 2000.

[92] D. Macho, L. Mauuary, B. Noe, Y.M.Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, and F. Saadoun. Evalutaion of noise-robust DSR front-end on AURORA databases. In *ICSLP*, 2002.

[93] Stephane Mallat. *A wavelet tour of signal processing*. Academic Press, 1999.

[94] K. Markov and S. Nakamura. Modeling HMM state distributions with Bayesian networks. In *ICSLP*, 2002.

[95] P.P. Marsal, S.P. Font, A. Hagen, H. Bourlard, and C. Nadeu. Comparison and combination of RASTA-PLP and FF features in hybrid HMM/MLP speech recognition system. In *ICSLP*, 2002.

[96] Kevin P. Murphy. *Dynamic Bayesian Networks : Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

[97] C. Nadeu. On the decorrelation of filter-bank energies in speech recognition. In *Eurospeech*, pages 1381–1384, 1995.

[98] C. Nadeu. On the filter-bank parameterization front-end for robust HMM speech recognition. In *Robust Methods for Speech Recognition in Adverse Conditions*, pages 235–238, Tampere, Finland, May 1999.

[99] C. Nadeu, D. Macho, and J. Hernando. Time and frequency filtering of filter-bank energies for robust HMM speech recognition. *Speech Communication*, 34 :93–114, April 2001.

[100] A.V. Nefian, L. Liang, X. Pi, L. Xiaoxiang, C. Mao, and K. Murphy. A coupled HMM for audio-visual speech recognition. In *ICASSP*, 2002.

[101] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8 :1–38, 1994.

[102] H.J. Nock and S.J. Young. Loosely coupled HMMs for ASR. In *ICSLP*, 2000.

[103] K.K. Paliwal. Decorrelated and liftered filter-bank energies for robust speech recognition. In *Eurospeech*, volume 1, pages 85–88, 1999.

[104] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[105] F. Rabiner and B.H. Juang. *Fundamentals of speech recognition*. Pearson Education POD, 1993.

[106] Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, Englewood Cliffs, NJ, USA, 1978.

[107] M. Richardson, J. Bilmes, and C. Diorio. Hidden-articulator Markov models for speech recognition. In *ISCA ITRW Conf. on Automatic Speech Recognition*, 2000.

[108] R. Rosenfeld. *Adaptive Statistical Language Modeling : A Maximum Entropy Approach*. PhD thesis, School of Computer Science Carnegie Mellon University, Pittsburgh, PA 15213, April 1994.

[109] R. Sarikaya and J. H. L. Hansen. High resolution speech feature extraction parameterization for monophone-based stressed speech recognition. *IEEE Signal Processing Letters*, 7(7) :182–185, July 2000.

[110] R. Sarikaya, B. L. Pellom, and J. H. L. Hansen. Wavelet packet transform features with application to speaker identification. In *IEEE Nordic Signal Proc. Symp.*, 1998.

[111] R. Shachter. Intelligent probabilistic inference. In *UAI*, pages 371–382, 1986.

[112] R. Shachter. Evidence absorbtion and propagation through evidence reversals. In *UAI*, pages 173–190, 1990.

[113] Takahiro Shinozaki and Sadaoki Furui. Time adjustable mixture weights for speaking rate fluctuation. In *Eurospeech*, 2003.

[114] O. Siohan and A. C. Surendran. Structural bayesian predictive adaptation of hidden markov models. In *In Workshop on Adaptation Methods for Speech Recognition, ISCA*, Sophia-Antipolis, France, 2001.

[115] K. Smaïli, A. Brun, I. Zitouni, and J.P. Haton. Automatic and manual clustering for large vocabulary speech re cognition : A comparative study. In *European Conference on Speech Communication and Technology*, volume 4, pages 1795–1798, Budapest, Hungary, September 1999.

[116] Padraic Smyth, David Heckerman, and Michael I. Jordan. Probabilistic independence networks for hidden markov probability models. *Neural Computation*, 9(2) :227–269, 1997.

[117] T. Stephenson, H. Bourlard, S. Bengio, and A. Morris. Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables, 2000.

[118] T.A. Stephenson, M. Mathew, and H. Bourlard. Modeling auxiliary information in Bayesian network based ASR. In *Proc. Eurospeech*, Scandinavia, 2001.

[119] Todd A. Stephenson, Jaume Escofet, Mathew Magimai-Doss, and Hervé Bourlard. Dynamic Bayesian network based speech recognition with pitch and energy as auxiliary variables. In *2002 IEEE International Workshop on Neural Networks for for Signal Processing (NNSP 2002)*, Martigny, Switzerland, September 2002.

[120] Todd A. Stephenson, Mathew Magimai-Doss, and Hervé Bourlard. Mixed Bayesian networks with auxiliary variables for automatic speech recognition. In *International Conference on Pattern Recognition (ICPR 2002)*, Quebec City, PQ, Canada, September 2002.

[121] Beng T. Tan, Minyue Fu, Andrew Spray, and Phillip Dermody. The use of wavelet transforms in phoneme recognition. In *ICSLP*, 1996.

[122] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclity of hypergraphs and selectively reduce acyclic hyypergraphs. *SIAM Journal of Computing*, 13 :566–579, 1984.

[123] S. Vaseghi, N. Harte, and B. Milner. Multi resolution phonetic/segmental features and models for hmm based speech recognition. In *ICASSP*, 1997.

[124] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, 13 :260–269, 1967.

[125] H. Wassner and G. Chollet. New cepstral representation using wavelet analysis and spectral transformation for robust speech recognition. In *Proc. ICSLP '96*, volume 1, pages 260–263, Philadelphia, PA, 1996.

[126] K. Weber, S. Bengio, and H. Bourlard. Increasing speech recognition robustness with HMM2. In *ICASSP*, 2002.

[127] C.J. Wellekens. Explicit time correlation in hidden Markov models for speech recognition. In *ICASSP*, 1987.

[128] M. Yannakakis. Computing the minimal fill-in is np-complete. *SIAM Journal of Algebraic Discrete Methods*, 2 :77–79, 1981.

[129] N. L. Zhang and D. Poole. A simple approach to bayesian network computations. In *In Proc. of the Tenth Canadian Conf. on AI*, pages 171–178, 1994.

[130] Yimin Zhang, Qian Diao, Shan Huang, Wei Hu, Chris Bartels, and Jeff Bilmes. Dbn based multi-stream models for speech. In *ICASSP*, 2003.

[131] G. Zweig. A forward-backward algorithm for inference in bayesian networks and an empirical comparison with hmms. Master's thesis, Dept. Comp. Science, UC Berkeley, 1996.

[132] Geoffrey G. Zweig. *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, University of California, Berkeley, Spring 1998.