Contextual Markovian Models

Mathieu Radenen

Thèse de doctorat

Modèles Markoviens Contextuels

Auteur :
Mathieu Radenen

Directeur de thèse :
Pr. Thierry Artieres

École doctorale Informatique, Télécommunications et Électronique (Paris)
25 octobre 2014

Catherine Achard	Maître de Conférences (HDR) à l'Université Paris 6	Examinatrice
Thierry Artières	Professeur de l'Université Paris 6	Directeur de thèse
Frédéric Béchet	Professeur de l'Université d'Aix-Marseille	Examinateur
Laurent Heutte	Professeur de l'Université de Rouen	Rapporteur
Laurence Likforman-Sulem	Maître de Conférences (HDR) à Télécom-ParisTech	Examinatrice
Georges Linarès	Professeur de l'Université d'Avignon	Rapporteur
Gérard Chollet	Directeur de Recherche CNRS à Télécom-ParisTech	Invité
Olivier Grisvard	Directeur d'équipe HCI Télécom Bretagne	Invité
Catherine Pelachaud	Directeur de Recherche CNRS à Télécom-ParisTech	Invitée

UPMC SORBONNE UNIVERSITÉS

LIP6

Remerciements

Je voudrais en premier lieu remercier toutes les personnes qui m'ont permis de mener cette aventure jusqu'au bout. Je remercie ma famille et particulièrement mes parents pour leur soutient inconditionnel. Merci tout simplement de m'aimer et de m'avoir encouragé à poursuivre mes rêves, merci de m'avoir aidé de toutes les manières qu'il vous étaient possible. Je ne me serais pas permis le luxe d'entreprendre une thèse alors que j'étais déjà dans la vie profesionnelle sans la force de vous savoir derrière moi. Merci Ariane, toi qui m'a accompagnée partout dans les plus belles aventures de ma vie. Merci de m'avoir tant donné, pour tous ces rires, cette complicité, ce que tu es, c'est avec toi que j'ai partagé et appris le plus. Tu es dans mon coeur.

Après cette séquence émotions, je voudrais remercier les personnes de mon bureau dont j'ai partagé le quotidien si spécial de "thésard" pendant plusieurs années, forcément ça crée des liens ! Alors merci Antoine Vinel bravo pour tes filles, chapeau pour ton embauche chez Apple. J'espère que tu vas gagner ton recent pari, on se refait une session running quand tu veux ! Merci Yann Soullard, bravo pour ton postdoc, tu as cette qualité de toujours te plier en 4 pour les autres fais gaffe ! ;) On se revoit à l'ICBS (International Conference on Beers and Sequences). Merci à "sa majesté" Moustapha Mouhamadou Cissé, bravo pour les résultats de ta thèse, n'oublie pas qu'on a rendez vous à Dakar pour manger des langoustes grillées sur la plage ! Merci à vous 3 pour toutes ces discussions qui m'ont permis d'avancer dans ma compréhension du monde, de ma thèse et des fois pas ;) !

Merci à mon directeur de thèse Thierry Artières pour m'avoir accueillli au LIP6 et permis de nourrir mes passions pendant ces quelques années dans une grande confiance. Je remercie aussi Olivier Grisvard qui a grandement contribué à trouver un financement pour ma thèse. Merci de m'avoir accueilli très chaleuresement à Brest et de nous avoir donné une grande liberté dans notre recherche au sein du projet Usixml.

Merci aux membres du Jury d'avoir accepté de m'accorder leur temps et leur expertise dans la relecture de mon manuscrit. Je remercie Georges Linarès directeur du LIA d'Avigion, Laurent Heutte du LITIS à Rouen, Catherine Achard de l'ISIR à Paris 6 ainsi que

Laurence Likforman-Sulem à Telecom ParisTech. Merci aussi à Catherine Pélachaud, Gérard Chollet et Yu Ding à Telecom ParisTech avec qui j'ai pu avoir des collaborations pendant ma thèse.

Je fais enfin un remerciement plus global à toute l'équipe MLIA du LIP6 avec qui j'ai pu échanger principalement le midi dans une grande convivialité. Merci aux anciens, Alex Spengler, Bruno Pradel, Cédric Le Barz, Gabriel Dulac Arnold. Merci aux jeunots, Emilie Au, Simon Bourigault, Gabriella Contardo, Ludovic Dos Santos, Luc-Aurélien Gauthier, Elie Guardia Sebaoun, Mickael Poussevin, Raphael Puget, Georgios Balikas. Je n'oublie pas non plus Ghislaine Mary et Jacqueline Le Baquer qui s'occupent de nous avec bienveillance pour les missions et pour toutes nos tracasseries administratives ! Et merci à tous ceux que je n'ai pas cité et avec j'ai partagé des moments sympas !

Table of Contents

1 Introduction **11**

2 Statistical models for time series modeling **17**
2.1 Statistical models . . . 18
2.1.1 Notations . . . 18
2.1.2 Supervised learning . . . 19
2.1.3 Model types . . . 20
2.2 Tasks and evaluation measures . . . 22
2.2.1 Isolated classification . . . 22
2.2.2 Recognition . . . 24
2.2.3 Synthesis . . . 25
2.3 Generative Markov models . . . 26
2.3.1 Hidden Markov Models . . . 26
2.3.2 Handling variability with HMMs . . . 31
2.4 Discriminative Markov models . . . 38
2.4.1 Conditional Random Fields . . . 38
2.4.2 Hidden Conditional Random Fields . . . 40
2.5 Conclusion . . . 42

3 Contextual Hidden Markov Models **45**
3.1 Introduction . . . 46
3.2 Single Gaussian Contextual HMM . . . 47
3.2.1 Mean parameterization . . . 47
3.2.2 Covariance parameterization . . . 48
3.2.3 Transitions parameterization . . . 49
3.2.4 Bayesian perspective . . . 50
3.3 Training . . . 50
3.3.1 With covariances parameterized . . . 51
3.3.2 With transitions parameterized . . . 52
3.3.3 Dynamic context . . . 53
3.3.4 Gaussian mixtures . . . 53
3.3.5 Tuning the gradient step size . . . 53
3.4 CHMM relative to similar approaches . . . 56
3.4.1 Variable Parametric HMMs . . . 56
3.4.2 Maximum Likelihood Linear Regression . . . 57
3.4.3 Context dependent modeling . . . 58
3.5 Application to the classification of handwritten characters . . . 59

3.5.1 Dataset . . . 59
3.5.2 Preliminary results . . . 60
3.5.3 Extended results . . . 63
3.6 Conclusion . . . 65

4 Contextual Hidden Conditional Random Fields 67
4.1 Introduction . . . 67
4.2 Discriminative training of Hidden Markov Models . . . 68
4.2.1 MMI . . . 69
4.2.2 MCE . . . 70
4.2.3 MWE/MPE . . . 71
4.2.4 Discussion . . . 72
4.3 Exploiting contextual information in Hidden Conditional Random Fields . 73
4.3.1 HCRF as a generalization of HMM . . . 73
4.3.2 Contextual HCRFs . . . 75
4.3.3 Training Contextual HCRFs . . . 77
4.3.4 Experiments . . . 77
4.4 Conclusion . . . 79

5 Exploiting Contextual Markov Models for synthesis 81
5.1 Motivation . . . 81
5.2 Using HMMs for synthesis . . . 82
5.2.1 Improved synthesis using non stationary HMMs . . . 82
5.2.2 Synthesis with constraints . . . 83
5.3 Speech to motion synthesis, an application . . . 87
5.3.1 Related work . . . 88
5.4 Speech to motion synthesis using Contextual Markovian models . . . 89
5.4.1 Parameterizations . . . 89
5.4.2 Training . . . 89
5.4.3 Synthesis . . . 90
5.4.4 Experiments . . . 90
5.5 Conclusion . . . 93

6 Combining contextual variables 95
6.1 Introduction . . . 95
6.2 Dropout regularization . . . 97
6.2.1 Dropout in CHMMs . . . 99
6.3 Multistream combination of variables . . . 100
6.3.1 Experimental setup . . . 101
6.3.2 Contextual variables . . . 101
6.3.3 CHMMs . . . 103
6.3.4 Multistream CHMMs . . . 104
6.4 Conclusion . . . 106

7 Toward Transfer Learning 107
7.1 Design of a global model . . . 108
7.1.1 Using a class code as contextual variables . . . 110
7.1.2 Task & dataset . . . 111

7.1.3 Preliminary results with one-hot class coding 111
7.1.4 Using a distributed representation of class as contextual variables . 113
7.1.5 Retraining discriminatively 116
7.2 Dynamic Factor Graphs 116
7.2.1 Continuous state space models 116
7.2.2 Analogy with Dynamic Factor Graphs 119
7.3 Conclusion 121

8 Conclusion & Perspectives **123**

Appendices **127**

Abstract **145**

Résumé français **147**

Résumé français long **149**

List of Figures

1.1 Various types of time series . . . 12

2.1 Bias variance dilemna . . . 20
2.2 HMM as a Dynamic Bayesian Network (DBN) . . . 26
2.3 Composite HMM . . . 31
2.4 Parametric HMM as a Dynamic Bayesian Network . . . 35
2.5 Input Output HMM as a Dynamic Bayesian Network . . . 36
2.6 Simple example of a CRF with a chain structure . . . 38
2.7 HCRF with a chain structure . . . 41

3.1 Effects of covariance parameterization . . . 49
3.2 Contextual HMM as a Dynamic Bayesian Network . . . 50
3.3 Examples of handwritten characters extracted from IAM dataset . . . 59
3.4 HMM character classification accuracy as a function of the mixture size . . . 62
3.5 CHMM character classification accuracy as a function of the window's length with $\boldsymbol{\theta} = \sigma^2$ or $\boldsymbol{\theta}_t = \sigma^2(t)$ . . . 62
3.6 CHMM character classification accuracy as a function of the window's length with $\boldsymbol{\theta} = \mu$ or $\boldsymbol{\theta}_t = \mu(t)$ . . . 63

5.1 Example of a possible Tokuda trajectory along a 7 states sequence of a HMM . . . 83
5.2 W transform matrix . . . 86
5.3 HMM as a Dynamic Bayesian Network for speech to motion synthesis . . . 88
5.4 CHMM as a Dynamic Bayesian Network for speech to motion synthesis . . . 89
5.5 Illustration of the extracted facial animation parameters . . . 91
5.6 Comparison of normalized motion sequence synthesized by HMM, μCHMM and μtrCHMM . . . 93

6.1 Train vs Test classification accuracy of mean+covariance parameterized CHMMs for different sizes of contextual variables . . . 96
6.2 Dropout in neural networks . . . 98
6.3 Means of a 4 states HMM as a function of SNR . . . 102
6.4 Word recognition accuracy of CHMMs using different contextual information . . . 104

7.1 Two examples of similar gesture classes extracted from HDM05 database 108
7.2 A global CHMM learnt on 3 gesture classes . . . 109
7.3 Global CHMM as a Dynamic Bayesian Network . . . 110
7.4 Global CHMM inference . . . 111

7.5 Motion capture data stream of "cartwheel" gesture represented as a sequence of poses . . . 112
7.6 HMM confusion matrix on the 37 gesture classes . . . 115
7.7 Helix dataset . . . 117
7.8 2D projections of helix data points with GPDM . . . 117
7.9 2D projections helix data points with Autoencoder RBM . . . 117
7.10 2D projections of 10 Walks gestures with PCA . . . 118
7.11 2D projections of 10 walks gestures with Autoencoder RBM . . . 118

List of Tables

3.1 Accuracy of HMM baselines as a function of the number of states per class model and Gaussians per state 64
3.2 Accuracy of Contextual HMM baselines for various topologies and definitions of contextual vector 65

4.1 Classification accuracy of discriminative models 78

5.1 Performance of various models with respect to the synthesis quality (MSE) and the labelling accuracy (Edit and Hamming distance) on Biwi 3D dataset 92
5.2 Performance of various models with respect to the synthesis quality (MSE) and the labelling accuracy (Hamming distance) on Biwi 3D dataset 92

6.1 Dropout experiments on Contextual HMMs 100
6.2 CHMM letters+digit word recognition accuracy on CHIME dataset using a combination of Speaker ID, SNR, SFM, and Rate as contextual vector . 103
6.3 CHMM letter+digits classification accuracy on CHIME dataset 105

7.1 F1 Accuracy of standard HMMs vs global CHMMs parameterized by a one hot code class contextual variable 112
7.2 Similarity matrix on 3 classes 114
7.3 F1 accuracy of global CHMMs parameterized by a one hot code or class similarity type contextual variable 115

Glossary

AugHCRF Augmented Hidden Conditional Random Fields. 76

AugHMM Augmented Hidden Markov Model. 64

CHCRF Contextual Hidden Conditional Random Fields. 14, 68

CHMM Contextual Hidden Markov Model. 14

CRF Conditional Random Fields. 38

DBN Dynamic Bayesian Network. 5

DFG Dynamic Factor Graph. 116

E Edit Distance. 92

EDA Edit Distance Accuracy. 25

FAP Facial Animation Point. 91

GPDM Gaussian Process Dynamical Model. 21

H Hamming Distance. 92

HCRF Hidden Conditional Random Fields. 14, 40

HMM Hidden Markov Model. 13

HTK Hidden Markov Model Toolkit. 101

IOHMM Input Output Hidden Markov Model. 36

ITEA Information Technology for European Advancement 2. 15

kNN k-Nearest-Neighbor. 21

LSTM Long Short Term Memory. 22

MCE Minimum Classification Error. 70, 71

MFCC Mel Frequency Cepstral Coefficients. 101

MLE Maximum Likelihood Estimation. 29

MLLR Maximum Likelihood Linear Regression. 58

MMI Maximum Mutual Information. 69

MPE Minimum Phone Error. 71

MWE Minimum Word Error. 71

PCA Principal Component Analysis. 118

PHMM Parametric Hidden Markov Model. 48

RBM Restricted Boltzmann Machine. 22

RNN Recurrent Neural Network. 22

SFM Spectral Flatness Measure. 103

SNR Signal Noise Ratio. 101

Usixml User Interface eXtended Markup Language. 15

VPHMM Variable Parametric Hidden Markov Model. 56

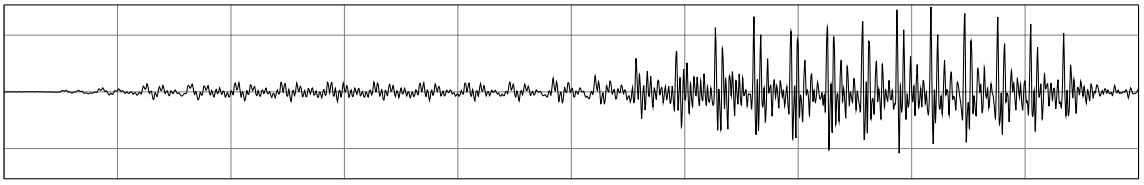
Chapter 1

Introduction

Time series or sequential data are defined as a sequence of data points typically measured at uniform time interval. Their modeling has practical applications like classification of isolated sequences, recognition which stands for labeling an input sequence into successive segments labels corresponding to particular units (e.g. phonemes in a speech signal), and synthesis new sequence data alike a training set. We will mainly focus in this thesis on pattern recognition tasks namely sequence classification and labeling.

Such tasks may concern a variety of domains. Speech, gesture, and handwriting recognition are among them. We can also mention other similar tasks like Named Entity Recognition in Natural Language Processing, the prediction of gene expression from DNA in biology, the recognition of patterns in financial analysis ... Figure 1.1 illustrates few typical examples of time series.

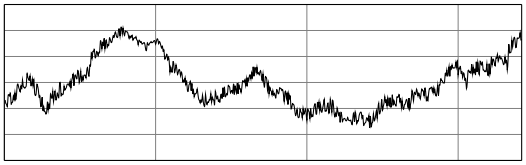
The modeling of time series with the goal of classification and labeling has been studied for many years. One of the most popular method until now has been the Hidden Markov Models. These models being generative they have been trained first with a non discriminative criterion (Maximum Likelihood) which is actually not particularly suited for discrimination tasks such as classification and sequence labeling. To overcome this drawback discriminative learning approaches have been proposed for Hidden Markov Models based on the optimization of discriminative criteria like Minimum Classification Error, Minimum Phone Error or Maximum Mutual Information. More recently new approaches have been proposed to tackle what is named the structured output prediction problem (i.e. tasks where the output to be predicted is structured, like sequences, graphs or trees). Among these works the Conditional Random Field has quickly been adapted to deal with complex sequences like speech, handwriting and videos.



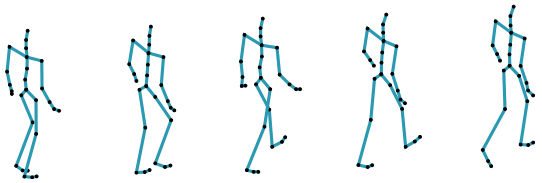
(A) Speech signal

(B) DNA sequences

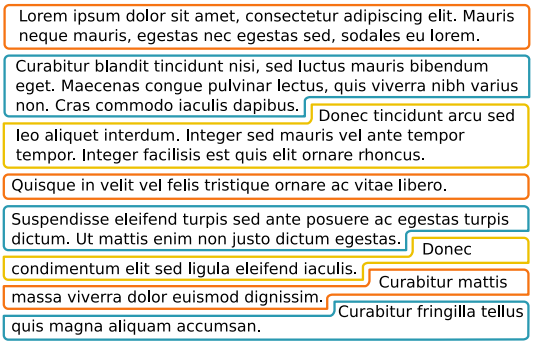
(C) Handwriting



(D) Financial data

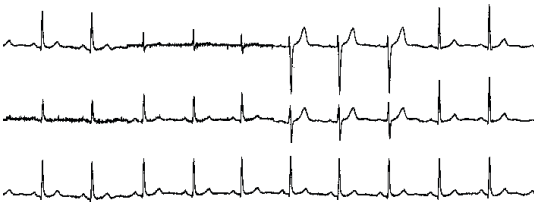


(E) Gestures

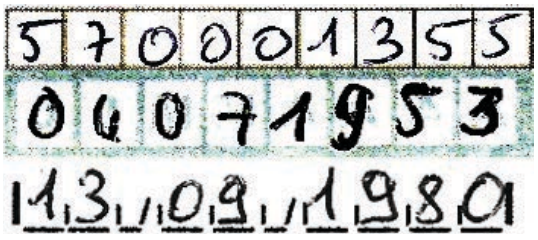


(F) Text

(G) Videos



(H) Medical data (electrocardiogram, electroencephalogram,...)



(I) Isolated characters

FIGURE 1.1: Various types of time series

Yet whatever the family of these models they are not well suited for handling the inherent variability of the signals. This variability may come from static features like the gender of the speaker in speech recognition or the morphology of the person whose gesture is captured in gesture recognition. A sentence may be uttered quite differently according to the speaker gender, a gesture may have more amplitude if it is performed slower, and its overall shape may depends on the weight or the height of the performer. Alternatively the variability may also come from time varying (or dynamic) features like the level of noise in speech recognition or the emotional state of the speaker in speech recognition.

The usual way for handling variability consists in narrowing it as much as possible in a preprocessing step, and then to model the remaining variability using standard strategies. In Markovian systems built on Hidden Markov Models this is done through multiplying states and through increasing Gaussian mixture size within states. Unfortunately this is a poor way to handle variability. Assuming for instance that one wants to build a recognition system for signals which have a particular bimodal structure (e.g. one wants to build a speech recognition system for male and female speakers). Then a simple solution would be to use an increased number of Gaussian distributions in Gaussian mixtures associated to states, part of these Gaussian distributions are dedicated to the first mode (e.g. male speakers) and the remaining are for the second mode (e.g. female speakers). However such modeling would then not be very accurate, for instance it would model well any signal that changes of mode at every time step although none sample of this kind was in the training set.

We are concerned in this thesis with the problem of handling variability in a sound way. We propose strategies for tackling such a variability coming from static as well as from dynamic features. We investigated these strategies for generative models (Hidden Markov models) as well as for discriminative models (Hidden Conditional Random Fields).

The starting point of our proposal is that an important part of the variability between observation sequences may be the consequence of a few contextual variables (which may be hidden or observed) that remain fixed all along a sequence or that vary slowly with time. Such a global variability cannot always be removed through preprocessing or normalization and would benefit from a specific handling in HMM. We propose the framework of Contextual Hidden Markov Models, whose starting point are Parametric Hidden Markov Models that have been proposed for gesture recognition, to model directly the influence of contextual information on observation sequences. We introduce several declinations of our framework to incorporate the influence of contextual variables at various levels into the models.

The thesis is organized in six main chapters. Chapter 2 presents useful background on sequence classification and labeling using statistical modeling and provides necessary details about Hidden Markov Models and Conditional Random Fields. Remaining chapters include the contributions of this thesis.

In Chapter 3 we propose the framework of Contextual Hidden Markov Models (CHMM) to model directly the influence of contextual information on observation sequences. We detail ways to implement this idea into Hidden Markov Models at various levels, by parameterizing (i.e. making dependent on contextual variables) either the emission probability density function within states (mean and covariance of Gaussian distribution) or the transition distributions from states to states. This modeling scheme may be used either with static or dynamic contextual variables.

Chapter 4 goes one step further and introduces a natural and efficient way to exploit contextual information into discriminative models for sequences like Hidden Conditional Random Fields (HCRF), yielding what we name Contextual Hidden state Conditional Random Fields (CHCRF). Such models are the discriminative counterpart of CHMMs. Although the idea of including contextual variables in HRCFs seems simple, its realization is problematic. Indeed HCRFs are highly subject to overfitting and are difficult to train accurately. Our proposal consists in learning Contextual HRCFs based on first, learning CHMMs which are less subject to overfitting, and secondly on using an initialization scheme that allows building initial CHCRFs estimates from learned CHMMs. Our proposal for CHCRF may then be viewed as an efficient way to learn a HCRF that exploit contextual information.

Chapter 5 shows a very different application of our work to the animation of an avatar. CHMMs are employed to synthesize facial moves of the avatar from the speech signal. The idea is to learn a CHMM system for modeling facial moves using speech features as contextual variables. Having learned such a system allows then animating an avatar (synthesizing its moves) based on the speech signal it is supposed to utter. Since CHMMs can be more accurate models than HMMs, performing synthesis based on CHMMs may be more accurate than HMMs based synthesis. This work has been performed in cooperation with Yu Ding, Ph.D. student at Telecom ParisTech under the supervision of C. Pélachaud and T. Artières.

Chapter 6 is related to an optimization problem we sometimes encountered when exploiting multiple contextual variables, eventually carrying very different information. Actually we observed that the optimization did not always succeeded in exploiting the discriminative power of all contextual variables yielding better results with only few contextual variables than with all, even when every single contextual variable was shown to carry discriminative information. We investigated two particular ways for dealing

with this problem. The first one is based on a regularization technique recently used in the Neural Networks literature, the second one relies on multistream modeling which has been popularized in the speech recognition literature for combining audio and video streams for improved speech recognition.

Finally we evoke in chapter 7 preliminary works to explore transfer learning for sequence data, i.e. learning to recognize a class leveraging information from other classes. These works have been motivated by the idea that contextual models (CHMMs, CHCRFs) might be good candidates for learning the models of many classes with only few examples per class. We explored a kind of Transfer Learning approach which relies on learning a global generative CHMM on the data of all classes and on using a special type of "class code" contextual variables. Basically the contextual variables for a given observation sequence may be or include an encoding of its class label. This strategy effectively allows sharing information between classes during learning. Our results on a gesture classification task show that a very simple implementation of this scheme helps in achieving a better generalization compared to classical independent class training of standard HMMs.

This thesis has been funded by the European Grant UsiXml (ITEA, Eureka #3674). Usixml is a project aimed at providing natural and multi-modal human interaction with computers where modalities might be speech, gesture or any sequential data provided by a human machine interface. A general view of this project which involves several industrial and academic partners is detailed in Appendices.

Chapter 2

Statistical models for time series modeling

Contents

2.1 Statistical models 18
2.1.1 Notations 18
2.1.2 Supervised learning 19
2.1.3 Model types 20
2.2 Tasks and evaluation measures 22
2.2.1 Isolated classification 22
2.2.2 Recognition 24
2.2.3 Synthesis 25
2.3 Generative Markov models 26
2.3.1 Hidden Markov Models 26
2.3.2 Handling variability with HMMs 31
2.4 Discriminative Markov models 38
2.4.1 Conditional Random Fields 38
2.4.2 Hidden Conditional Random Fields 40
2.5 Conclusion 42

We first present here generalities on modeling time series using statistical models. Then, we focus on presenting the generative and discriminative families of Markovian approaches which sets the basis of our work.

2.1 Statistical models

2.1.1 Notations

First we introduce a number of notations that will be useful for presenting statistical models for time series.

A database $\mathcal{D}$ is defined as N pairs $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ or examples with $k \in 1 \dots N$
where $\boldsymbol{x}^k \in \mathcal{X}$ is a sequence of observation vectors for example k
and $\boldsymbol{y}^k \in \mathcal{Y}$ its corresponding sequence of labels

More precisely $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$ are defined as two time series :

$$\boldsymbol{x}^k = (\boldsymbol{x}_1^k, \dots, \boldsymbol{x}_T^k)$$
$$\boldsymbol{y}^k = (y_1^k, \dots, y_C^k)$$

- where $\boldsymbol{x}_t^k \in \mathbb{R}^d$ is an observation vector at time t
- and $y_c^k \in Y$ is a label $(\forall c \in 1 \dots C)$
- label space Y is assumed here a finite alphabet

When dealing with a single example $(\boldsymbol{x}^k, \boldsymbol{y}^k)$, the index k is often dropped for clarity. Lowercase letters are used for scalars, vectors or sequences, while matrices are uppercased. Vectors or sequences are noted in boldface but scalars are not.

When annotation is available at the frame level, there is one label $y_t \in Y$ for each observation $\boldsymbol{x}_t$, thus, $\boldsymbol{y} = (y_1, \dots, y_T)$. Alternatively, the dataset $\mathcal{D}$ can be annotated at the level of symbols (e.g $\boldsymbol{y}$ can be a sequence of letters). In this case, the observation sequence $\boldsymbol{x}$ and its corresponding sequence of labels $\boldsymbol{y}$ have not necessarily the same length.

We also assume the examples $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ of $\mathcal{D}$ are iid: Independent (2.1) and Identically Distributed (2.2):

$$p((\boldsymbol{x}^1, \boldsymbol{y}^1), \dots, (\boldsymbol{x}^k, \boldsymbol{y}^k)) = \prod_k p((\boldsymbol{x}^k, \boldsymbol{y}^k)) \tag{2.1}$$

$$(\boldsymbol{x}^k, \boldsymbol{y}^k) \sim P(\mathcal{X}, \mathcal{Y}) \tag{2.2}$$

- where $\mathcal{X}$ is the space of all possible observations sequences $\boldsymbol{x}$
- and $\mathcal{Y}$ is the space of all possible label sequences $\boldsymbol{y}$

Finally, we will note $\mathcal{X}_y$ the space of observation sequences whose label is y.

2.1.2 Supervised learning

The tasks of classification, recognition and synthesis can carried out by first training a model through supervised learning. It consists of finding the parameters Λ of a function or model $f_\Lambda : \mathcal{X} \to \mathcal{Y}$ which minimizes the risk

$$\mathcal{R}(f_\Lambda) = \int \mathcal{L}(f_\Lambda(\boldsymbol{x}), \boldsymbol{y}) dP(\boldsymbol{x}, \boldsymbol{y})$$

$\mathcal{R}(f_\Lambda)$ is the expectation of a Loss functional $\mathcal{L}$ under the joint distribution of the observations $\boldsymbol{x}$ and its corresponding labels $\boldsymbol{y}$.

Various losses can be employed, section 2.2 will present some of which are suited for our tasks.

2.1.2.1 Empirical risk minimization

The true distribution $P(\mathcal{X}, \mathcal{Y})$ is unknown, so instead, we approximate $\mathcal{R}(f_\Lambda)$ with the empirical expectation over a training set $\mathcal{D}_{train}$.

$$\mathcal{R}_{emp}(f_\Lambda) = \frac{1}{N} \sum_{\mathcal{D}_{train}} \mathcal{L}(f_\Lambda(\boldsymbol{x}^k), \boldsymbol{y}^k) \tag{2.3}$$

2.1.2.2 Structural risk minimization

Because we do not exactly minimize the true risk, there is a famous problem arising which is known as the bias variance dilemna.

Generally, the more the model has capacity (for example higher degree polynomial), the lower its error $\mathcal{R}_{emp}(f_\Lambda)$ on the training set. Unfortunately, because $\mathcal{R}_{emp}(f_\Lambda)$ is not defined by the true data distribution, using models of increasing capacity can overfit the training set, and, consequently, the true risk error $\mathcal{R}(f_\Lambda)$ is growing (the model error has low bias on the training set but high variance on new data).

On the other hand, we can choose a simpler model, its error will have low variance on new data but if the model is too simple, its error will be high on the training set (high bias). Figure 2.1 is a simple illustration of this tradeoff.

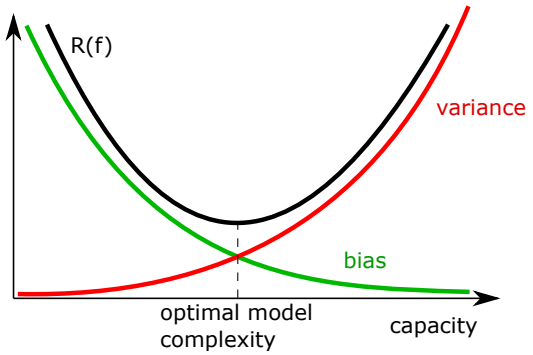


FIGURE 2.1: Bias variance dilemna

One of the popular strategy is that instead of choosing the right model capacity, we can train a high capacity model but regularize it to prevent overfitting.

Regularization is a form of constraint imposed on the parameters of the model Λ. This can be easily added by minimizing the new function

$$J(f_\Lambda) = \mathcal{R}_{emp}(f_\Lambda) + \lambda C(f_\Lambda)$$

Typically $C(f_\Lambda)$ can impose a sparsity constraint (L1 norm) on the weights Λ. This forces the model to drive many parameters towards 0, resulting in a simpler model less prone to overfitting. Another form of constraint is to add L2 norm on Λ, which limits the magnitude of the parameters. Large weights tends to favor sharp decision boundaries very sensitive to the input, typically what we want to avoid for good generalization.

The hyper-parameter λ controls the amount of regularization. If we choose $\lambda = 0$, the model is not regularized at all. If the model has enough capacity, its training error ($\mathcal{R}_{emp}(f_\Lambda)$) will be small (low bias) but the model errors will be high on a separated test set (high variance). On the contrary, if λ is large the model will be strongly regularized, resulting in many error on the training set (high bias) but low variance on a separated test set. Many other regularization schemes can be devised or even combined.

2.1.3 Model types

2.1.3.1 Parametric and non parametric

Parametric models are defined as a function $f_\Lambda : \mathcal{X} \to \mathcal{Y}$ which is differentiable with respect to its parameters Λ. We can then train them by directly minimizing the empirical risk (2.3) with respect to Λ. This can be done with various forms of gradients descent or even in closed form depending on the form of f_Λ.

On the other hand, non parametric models do not perform training on a parameter set but build an estimator $f_{\mathcal{D}_{train}}$ which depends on all the training examples. The computational complexity is almost completely postponed at test time. Generally, f is expressed in terms of similarity.

A famous example of such models is k-Nearest-Neighbor (kNN) although not really suited for sequential data. For a classification task, a 1-NN will give to a new input $\boldsymbol{x}$ the label y of the most similar example $\boldsymbol{x}_{nearest}$ in $\mathcal{D}_{train}$. If the input examples $\boldsymbol{x}$ are fixed sized vectors, a candidate similarity measure can be the Euclidean distance. For sequential data, $\boldsymbol{x}$ and $\boldsymbol{x}_{nearest}$ might not be the same length, and a Dynamic Time Warping like algorithm [70] must be employed to compute their similarity.

Gaussian Process Dynamical Model (GPDM) [82] are another form of non parametric models. They define a probability over observation sequences by the use of a kernel similarity measure in a latent space. They do not have real parameters as they are marginalized over, instead they directly optimize over the latent representation of the data.

Nowadays, a vast majority of approaches for modeling sequential data uses parametric models. They are more compact, (ie do not need to remember the training set), and their inference time is generally quicker. Nevertheless non parametric methods can be of interest when there are few examples as their complexity is reduced. Because they do not estimate parameters, they are also less prone to overfitting.

2.1.3.2 Generative and discriminative

Generative models specify a joint probability distribution between observations and labels. They are trained to maximize the joint probability of observation sequences and its corresponding sequence of labels $p(\boldsymbol{x}, \boldsymbol{y})$.

These models are called generative because it is possible to sample from their distribution to synthesize new observation sequences (see 2.2.3). Yet they are not restricted to synthesize data, using Bayes' rule, one can build a conditional probability function of the class labels suitable for the tasks of classification and recognition. In inference, the candidate labeling $\boldsymbol{y}$ for observation sequence $\boldsymbol{x}$ is then selected as

$$\text{argmax}_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda) = \text{argmax}_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}; \Lambda) p(\boldsymbol{y}; \Lambda)$$

Discriminative models directly optimize the conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$. They do not attempt to model the distribution of the data $p(\boldsymbol{x})$ which can be seen as an

unnecessary intermediate step for sequence labeling. The immediate drawback is that we can not sample from their distribution to generate new observation data $\boldsymbol{x}$. This makes them unsuitable for synthesis.

2.1.3.3 Discrete and continuous state space

Except for a few non parametric approaches (like GPDM), models of sequential data are characterized by an internal structure. They are composed of unobserved (hidden) states taking their value in a discrete or continuous space.

For instance, Hidden Markov Models (HMM) and Hidden Conditional Random Fields (HCRF) use a set of hidden states. Each of the hidden states assign either a probability (HMM) or a score (HCRF) to the observations at each time step. Their states are discrete and mutually exclusive. It means that the hidden state variable is a K-multinomial: out of K possible states, only one is active at each time step.

On the other side, the internal structure of Recurrent Neural Networks (RNN [42], LSTM [39], ...) can be viewed as hidden states in a continuous space. The state (or activation) of each hidden unit is real valued and influences the final output.

Some successful approaches also combine discrete state and continuous state space models such as RNN-RBM [9], DBN-HMM [36], LSTM-HMM [33], ... In RNN-RBM for example, the Restricted Boltzmann Machines (RBM) can be viewed as a discrete state space model because their hidden units are binary-valued. Yet RBM do not explicitly model the temporal nature of the data which amounts to the RNN part of the model.

The focus of our work mainly concerns the widely popular family of discrete state space Markovian approaches for modeling time series.

2.2 Tasks and evaluation measures

2.2.1 Isolated classification

We assume here the dataset $\mathcal{D}$ is now defined as N pairs $(\boldsymbol{x}^k, y^k)$ $(k \in 1 \dots N)$ of examples with $\boldsymbol{x}^k$ an observation sequence and y^k a single corresponding label.

Classification of isolated sequences is defined as the task of assigning a unique label y^k to an observation sequence $\boldsymbol{x}^k = (\boldsymbol{x}_1^k, \dots, \boldsymbol{x}_t^k)$.

This can be done with a function (a classifier) f which is a mapping from the space of observation sequences $\mathcal{X}$ to the space of individual labels Y.

$$f(\mathcal{X}) \rightarrow Y$$

We note $\hat{y} = f(\boldsymbol{x})$ the predictive label assigned by a classifier f to observation sequence $\boldsymbol{x}$ while y is its true label. As opposed to the task of recognition (described in section 2.2.2), isolated classification refers to the fact that we know the time boundaries of each observation sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$. Theses boundaries are assumed to be known at training and test time.

One can measure the quality of a classifier f in term of its accuracy :

$$Acc(f) = \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_{0,1}(f(\boldsymbol{x}^k), y^k) \times 100$$

$$\mathcal{L}_{0,1}(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

- $\mathcal{L}_{0,1}$ is the zero-one loss
- and Acc counts the proportion of right labels given by f on N examples.

When the dataset is unbalanced (i.e. few classes contain much more instances than other ones in $\mathcal{D}_{train}$, a classifier with high accuracy is not always good. Consider the simple example of a binary classification task where a classifier has to decide whether a data belongs to its class (positive example) or not (negative one). If there are 99% negative examples and 1% positive ones, the accuracy of a classifier answering always 'negative' will be 99% whereas it has clearly not learned anything.

For these kind of situations, the precision and recall give a better understanding of the classifier behavior.

Precision is defined as the number of true positive examples (tp) over all examples classified positively. (true positive (tp) and false positive (fp) examples)

$$Precision = \frac{tp}{tp + fp}$$

If precision is high, the number of false positive will be low. It means the classifier makes a few mistakes for relevant examples.

On the other side, recall is the number of true positive examples over true positive and false negative examples.

$$Recall = \frac{tp}{tp + fn}$$

For high recall, false negative must be low which means the classifier do retrieve a large proportion of relevant examples.

Precision and recall convey two different types of information which can be combined through the F-measure :

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

Note that in a multi-class setting (more than 2 classes), precision and recall are generally averaged over all classes (Macro-Precision/Recall).

2.2.2 Recognition

In a recognition task (or sequence labeling), an observation sequence $\boldsymbol{x}$ must be assigned a sequence of labels $\boldsymbol{y}$. We are then looking for a model f mapping the space of observation sequences to the space of label sequences.

$$f(\mathcal{X}) \to \mathcal{Y}$$

This is a more realistic setting than classification. For example, in speech recognition, a speaker generally utters a whole sentence of words that we need to recognize. In this setting, the labels are words or phonemes and observations sequences are feature vectors characterizing the speech signal at each time step. At test time, there is no information regarding the number of words uttered nor their time boundaries, our goal is precisely to find them.

To assess the quality of the match between a predicted sequence of labels $\hat{\boldsymbol{y}}$ and the true sequence of labels $\boldsymbol{y}$, one can measure the edit distance between them $ed(\hat{\boldsymbol{y}}, \boldsymbol{y})$. It is the minimum number of elementary operations (insertions (I), deletions (D) or substitutions (S)) that transforms $\boldsymbol{y}$ into $\hat{\boldsymbol{y}}$ or vice versa. Note that operations can have different weights but most often, they are equal.

Computing $ed(\hat{\boldsymbol{y}}, \boldsymbol{y})$ involves running a dynamic programming procedure whose complexity is $O(mn)$ where m and n are the length of the two strings of labels $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$.

A normalized version is given by

$$ed(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{S + D + I}{L}$$

where $L = (S + D + H)$ is the length of the ground truth sequence of labels $\boldsymbol{y}$ and $H = (L - S - D)$ is the number matched symbols (hits).

We define the edit distance accuracy loss (EDA) $\mathcal{L}_{EDA}$ as the opposite

$$\mathcal{L}_{EDA}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - ed(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{L - S - D - I}{L}$$

which can be simplified to

$$\mathcal{L}_{EDA}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{H - I}{L}$$

Finally, the accuracy evaluation measure for a recognition task is given by the average $\mathcal{L}_{EDA}$ for all predicted sequences of labels.

$$Acc(f) = \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_{EDA}(f(\boldsymbol{x^k}), \boldsymbol{y^k})$$

2.2.3 Synthesis

The particularity of generative models is that they can be used to synthesize new observation data. Given a trained model of a specific class y, we can sample directly from its distribution $P(\mathcal{X}, y)$ to generate unseen realistic observations sequences. This property has applications in many domains. For example, in text to speech systems, one trains generative models on words or phonemes and sample from these individual models to synthesize realistic utterances. In the field of character animation, one uses generative models trained on gestures to animate an avatar realistically. Other approaches can also merge speech and motion features to create even more realistic animations (see chapter 5).

There are two ways of estimating the performance of a synthesis system. The subjective way is to ask a group of people to evaluate the realism of synthesized observations (utterances, animations etc ...) on a scale and average their marks. More objectively, if there is a reference on which we can compare, we can use the mean square error (MSE) between the synthesized $\hat{\boldsymbol{x}}$ observations and the ground truth $\boldsymbol{x}$. On N sequences the average MSE is given by

$$MSE = \frac{1}{N} \sum_{k=1}^{N} ||\hat{\boldsymbol{x}}_{\boldsymbol{t}}^{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{t}}^{\boldsymbol{k}}||_2^2$$

It measures the squared Euclidean distance at every time steps between the reference $\boldsymbol{x_t^k}$ and the prediction $\hat{\boldsymbol{x}}_t^{\boldsymbol{k}}$, hence lower MSE means better synthesis. Note that $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ must be the same length.

2.3 Generative Markov models

2.3.1 Hidden Markov Models

Hidden Markov Models [67] are among the most popular approaches for time series modeling. As it is the basis of many models presented in this thesis, we will present it in more details compared to its other variants.

Formally, it can be described by

- $\mathcal{H}$ a set of S discrete hidden states
- $h_t \in \mathcal{H}$ the value of current hidden state at time t
- $A_{i,j} = p(h_t = i \mid h_{t-1} = j)$ the transitions probabilities from hidden state j at time $t-1$ to hidden state i at time t with the constraint $\sum_j A_{i,j} = 1$
- $p(\boldsymbol{x}_t \mid h_t)$ a probability distribution function responsible for emitting an observation $\boldsymbol{x}_t$ in a state h_t
- π_i an initial state distribution corresponding to the probability of being in hidden state i at the beginning of the sequence at time $t = 1$

A HMM actually hypothesizes that a latent cause (unobservable) is responsible for generating the observation sequence $\boldsymbol{x}$. In the following graphic, we show an HMM sketched as a Directed Acyclic Graph (also known as Bayesian network). Nodes in the graph represent random variables and a links between two nodes stands for a direct conditional dependency between two corresponding random variables.

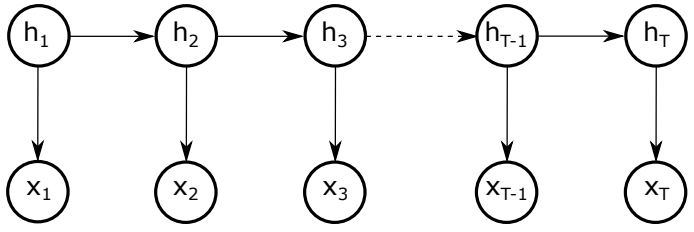


FIGURE 2.2: HMM with a chain structure, each state h_t is responsible for emitting an observation $\boldsymbol{x}_t$ while the model switches from state to state according to a transition probability matrix.

In the HMM depicted above, observations ($\boldsymbol{x}_t$) are conditionally independent from each others given the their hidden state (h_t). Note that we represented a HMM with a chain structure, that is, the emission and transition probabilities only depend on the current hidden state. This HMM has a 1 order Markov chain structure. One can add longer time dependencies between hidden states by defining order two transition probabilities as $p(h_t \mid h_{t-1}, h_{t-2})$ for example. However, the number of parameters of the model and the complexity of its training and inference algorithms grows critically. Hence, from now on, we will consider the most popular form of HMM with a Markov chain structure of order 1.

Given an hidden state sequence $\boldsymbol{h}$ which has the same length T as the observation sequence $\boldsymbol{x}$, we can compute the joint probability of the hidden state and observation sequence as :

$$p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) = \pi_{h_1} p(x_1 \mid h_1) p(h_2 \mid h_1) p(x_2 \mid h_2) \dots p(h_T \mid h_{T-1}) p(x_T \mid h_T) \tag{2.4}$$

where Λ are the parameters of the model.

Or, equivalently :

$$p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) = \pi_{h_1} p(x_1 \mid h_1) \prod_{t=2}^{T} p(h_t \mid h_{t-1}) p(\boldsymbol{x}_t \mid h_t)$$

The probability distribution function $p(\boldsymbol{x}_t \mid h_t)$ is often defined as a Gaussian

$$p(\boldsymbol{x}_t \mid h_t) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_{h_t}, \Sigma_{h_t})$$

or a mixture of M Gaussians in each state

$$\begin{aligned} p(\boldsymbol{x}_t \mid h_t) &= \sum_{m=1}^{M} p(m \mid h_t) p(\boldsymbol{x} \mid \boldsymbol{h}_t, m) \\ p(\boldsymbol{x}_t \mid h_t) &= \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_{c_m}, \Sigma_{c_m}) \end{aligned} \tag{2.5}$$

- where c_m is the Gaussian mixture weight (scalar)
- $\boldsymbol{\mu}_{c_m}$ is its mean (d vector, same size as an observation $\boldsymbol{x}_t$)
- Σ_{c_m} its covariance ($d \times d$ matrix)

Although it is possible to share the parameters of the Gaussian distributions between states, we consider the general case of states with independent parameter sets.

Finally, the parameters Λ of a Gaussian mixture HMM are composed of

- the initial state distribution π (vector of dimension S)
- a $S \times S$ transition probability matrix A
- $S \times M$ Gaussians with parameters $\boldsymbol{\mu}_{h,m}$, $\Sigma_{h,m}$, $c_{h,m}$

2.3.1.1 Inference

So far, we have seen how to compute the joint probability of state and observation sequence $p(\boldsymbol{x}, \boldsymbol{h}; \Lambda)$. However, it is rarely the case that we can know in advance the hidden causes of the observations. One has two choices, whether to guess the hidden state sequence, or, sum over all possible state sequences (i.e. marginalize).

Infering the best hidden state sequence is described by the following equation :

$$\text{argmax}_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) \tag{2.6}$$

which can be handled by a dynamic programming procedure known as the Viterbi algorithm (see [67]).

If one marginalize out (or sum over all) the possible hidden states sequences, it can be written

$$p(\boldsymbol{x}; \Lambda) = \sum_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) \tag{2.7}$$

However, the summation over all possible hidden state sequences contains an exponential number of terms with respect to the length of the observation sequence $\boldsymbol{x}$. Fortunately, one can compute equation 2.7 efficiently thanks to a dynamic programming procedure known as the Forward-Backward algorithm (cf [67]).

Both Viterbi and Forward Backward algorithms have a complexity in $O(S^2T)$ where S is the number of states and T the length of observation sequence $\boldsymbol{x}$.

2.3.1.2 Training HMMs by Maximum Likelihood Estimation

Learning a HMM "generatively" consists in finding the parameters Λ that maximize the joint probability of all the observation sequences in the training set. Using the iid assumption from 2.1.1 one can write :

$$\hat{\Lambda} = \text{argmax}_{\Lambda} p(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N; \Lambda) = \prod_{k=1}^{N} p(\boldsymbol{x}^k; \Lambda) \tag{2.8}$$

which is equivalent to

$$\text{argmax}_{\Lambda} \log p(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N; \Lambda) = \sum_{k=1}^{N} \log p(\boldsymbol{x}^k; \Lambda) \tag{2.9}$$

This type of estimation is known as Maximum Likelihood Estimation (MLE). However, due to the presence of constraints in HMMs ($\sum_j A_{i,j} = 1$), it turns out to be easier to maximize an auxiliary function than 2.9 directly.

This can be done by the Expectation Maximization (EM) algorithm [19][7] which maximizes the following auxiliary function Q with respect to the parameters Λ for each observation sequence $\boldsymbol{x}$

$$Q(\Lambda, \Lambda') = \mathbb{E}_{\boldsymbol{h}|\mathbf{x};\Lambda'} \left[\log P(\mathbf{x}, \mathbf{h}; \Lambda)\right] \tag{2.10}$$

where Λ' is the current value of the model parameters and Λ is their updated value

Actually, EM alternates between two steps in an iterative procedure :

- E (Expectation) step : compute $p(\boldsymbol{h} \mid \boldsymbol{x}; \Lambda)$ the posterior probability of hidden variables given the observations sequence under the model with current parameter values Λ
- M (Maximization) step : Maximize Q with respect to Λ. That is, compute the closed form reestimation of the model parameters Λ which maximizes Q using $p(\boldsymbol{h} \mid \boldsymbol{x}; \Lambda)$ and update the current parameters values $\Lambda' \leftarrow \Lambda$

Each iteration of EM increases the value of Q which, in turn, guarantees an increase of the likelihood function 2.9. When the algorithms stops, or Q no more improves, we are only guaranteed to reach a local maximum of the likelihood because it is not concave.

Note that a modified form of the M-step is required for certain complex HMM variants where there is no closed form reestimation of their parameters. Instead of maximizing $Q_k(\Lambda, \Lambda')$ with respect to Λ at iteration k, we find a Λ such that $Q_k(\Lambda, \Lambda') > Q_{k-1}(\Lambda, \Lambda')$. This form of the algorithm is called Generalized Expectation Maximization (GEM) and is also guaranteed to converge.

2.3.1.3 Synthesis, classification and recognition with HMMs

In all the subsequent tasks, a training set is composed of different labels, or classes. The first step is to assign each label y a set of hidden states S_y modeled by a separate HMM with parameter Λ_y. Each HMM is then trained on every observation sequence of its

class $\boldsymbol{x} \in \mathcal{X}_y$. As a result, a HMM with parameters Λ_y then model $p(\mathcal{X}_y, y; \Lambda_y)$ the joint probability of observation sequences and its label y (or class).

Synthesis : Once such a HMM is trained, it can be used directly for a basic form of synthesis. The principle is to sample from its distribution $p(\mathcal{X}_y, y; \Lambda_y)$ to synthesize a new observation sequence. One begins by choosing an initial hidden state h_1 by sampling from the initial distribution π. Then, a first observation $\boldsymbol{x}_1$ is sampled from the emitting distribution $p(\boldsymbol{x}_1 \mid h_1; \Lambda)$. Next one samples from p$(h_2 \mid h_1; \Lambda)$ to choose a second hidden state h_2, sample from $p(\boldsymbol{x}_2 \mid h_2; \Lambda)$ and so forth until are reached an ending state or the number of observation samples desired.

Classification : In this setting, the most likely label y of an observation sequence $\boldsymbol{x}$ is infered as :

$$\text{argmax}_y p(\boldsymbol{x}, y; \Lambda_y) \tag{2.11}$$

which encompass computing equation 2.7 via the Forward-Backward algorithm for every class represent HMM having parameters Λ_y

Recognition : For this task, it is a bit more complex because an observation sequence $\boldsymbol{x}$ need to be assigned a *sequence* of labels $\boldsymbol{y}$. To infer the most likely sequence of labels $\boldsymbol{y}$, a composite HMM must be built from the concatenation of every class HMM (as depicted in the figure below).

In figure 2.3 we illustrate the topology of a composite HMM built from 3 class HMMs. Here, the class HMM have a "left right" topology only allowing transitions from the current state to itself or the following state. The first class is modeled by a 4 states HMM, the second one a 3 state HMM, and the third one a two state HMM. Counting an initial and a final state, the composite model has 11 states.

The most likely sequence of labels for an observation sequence $\boldsymbol{x}$ is then given as :

$$\text{argmax}_{\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y}; \Lambda_s) \tag{2.12}$$

where Λ_s are the parameters of the composite HMM.

The most likely label sequence may be computed from a simple Viterbi application. The most likely state sequence from the composite HMM translates naturally in a label sequence.

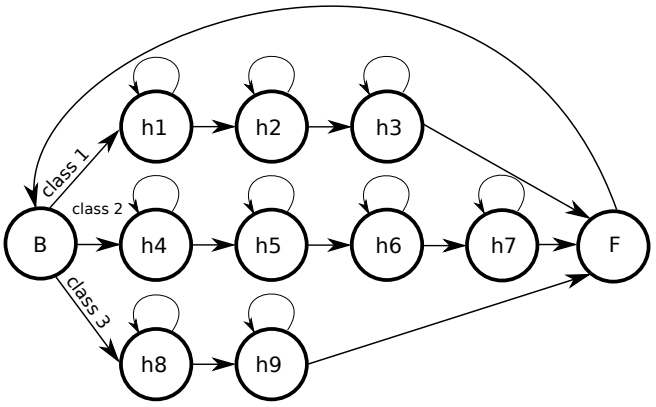


FIGURE 2.3: Topology of a HMM built from the HMMs of 3 classes. Non emitting initial and final state (B and F) are added to allow transitions between the different classes models.

Sometimes, the architecture of the composite model venture off this simple example. This is the case when the labeling is constrained by a grammar which specify the transitions between labels.

A last problem we didn't talk about concerns the frequent absence of frame-wise labeling of the training set (label boundaries are missing). For each observation sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, we don't necessarily have a corresponding sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_T)$ at the frame level. Instead it can be given at the level of symbols $\boldsymbol{y} = (y_1, \ldots, y_C)$ where $C < T$. Yet, if the boundaries of the labels are unknown in training, it is still possible to train individual classes HMMs. For each observation sequence $\boldsymbol{x}^k$ with labels $\boldsymbol{y}^k = (y_1, \ldots, y_C)$, one builds a "sentence HMM" as a simple left right concatenation of HMMs modeling each class symbol y_c of the labels sequence $\boldsymbol{y}$. Then, posterior probabilities over the hidden states sequences $p(\boldsymbol{h}^k \mid \boldsymbol{x}^k)$ are accumulated in a E step. After all observations sequences have been presented, one can compute individual M-steps for each class HMM.

To conclude on Standard HMMs, we would like to stress a few important points. Although simple, they are still very popular in the domain of time series. They can perform various tasks (like classification, recognition or synthesis) in an efficient way.

2.3.2 Handling variability with HMMs

However, HMMs have several limitations and many variants have been proposed to improve upon them. One particular shortcoming is that HMM probability distributions are stationary in a given state. Concretely it means that a HMM models time series with piecewise constant distribution functions. This is a grossly way of modeling the variability of observation sequences. In the following, we will expose several approaches

which introduce non stationary state distributions in Hidden Markov Models. Some of them especially rely on conditioning HMM distributions with external variables which will set the basis of our work.

2.3.2.1 Trended HMM

A first attempt at modeling non stationary state distributions in HMM seems to be the work of Deng et al. [20][21] on Trended HMMs.

In their work, the observation is defined as a polynomial function of the entry time τ_i in hidden state i

$$\boldsymbol{x}_t = \sum_{m=0}^{M} B_i(m) f_m(t - \tau_i) + R_t(\Sigma_i) \tag{2.13}$$

- with f_m a polynomial of order m
- $B_i(m)$ is the learnable coefficient for f_m in state i
- M is the order of the polynomial function

Assuming R_t is a residual of zero mean and covariance Σ_i, the emission probability of HMMs is defined as the following function of the state sojourn time $d_t = t - \tau_i$.

$$p(\boldsymbol{x}_t \mid h_t = i, d_t) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu_i}(d_t), \Sigma_j)$$

where the mean $\boldsymbol{\mu}_i$ in state i is modeled as a polynomial $\boldsymbol{\mu_i}(d_t) = \sum_{m=0}^{M} B_i(m) f_m(d_t)$

Trended HMMs belong to the class of non stationary state models because the states' emission distributions (Gaussian means) is allowed to vary along time. This is a desirable property to capture more variability in the observation sequence. In [20], Deng et al. shown how this modeling can better fit test data while being more economical in the number of parameters.

Inference with this model is slightly modified compared to standard HMMs. Classically, a Viterbi inference must compute the optimal state sequence, but here, it must be done for each possible setting of the state sojourn time d_t. This new Viterbi then associates each observation $\boldsymbol{x}_t$ to a state h_t with a duration d_t.

During training, estimation of the polynomial coefficients can be done in closed form by solving a linear system of $(M+1)$ unknows for each dimension of the observations vector $\boldsymbol{x}_t$ and each hidden state h_t attributed to the observations sequence $\boldsymbol{x}$. As usual when estimating HMM parameters, reestimation is performed in an iterative EM procedure.

The main problem with Trended HMMs is that they become inefficient for long sequences. Inference complexity is now $O(S^2T^2)$ instead of $O(S^2T)$ (for standard HMMs) because of the new dimension of maximization. Hence, to reduce computational time, Deng et al. use an approximation to the exact Viterbi inference to extract their segmentation. Moreover, they introduce a second approximation. The Expectation Maximization based learning algorithm uses only the best hidden state sequence to reestimate the polynomial parameters instead of a sum over all possible hidden state sequences. The true posterior distribution over hidden states sequence is in fact approximated by its mode (maximum).

2.3.2.2 Trajectory HMM

Trajectory HMMs were originally proposed as another form of non stationary state model for the classification and synthesis of speech utterances. In their work [90], Zen et al. define an observation vector $\boldsymbol{x}_t$ as a vector of static features $\boldsymbol{c}_t$ ($M \times 1$ vector) and its first and second order derivatives :

$$\boldsymbol{x}_t = [\boldsymbol{c_t^T}, \boldsymbol{\Delta c_t^T}, \boldsymbol{\Delta^2 c_t^T}]^T$$

Then, they reformulate the training of HMMs with constraints between static and dynamic features.

$$\boldsymbol{x} = W\boldsymbol{c} \tag{2.14}$$

where W is a $3MT \times MT$ known matrix transforming a static observation sequence $\boldsymbol{c}$ ($MT \times 1$) to a full observation sequence $\boldsymbol{x}$ ($3MT \times 1$).

The idea is that the full observation sequence $\boldsymbol{x}$ must remain consistent with the definition of static features $\boldsymbol{c}$ ($\boldsymbol{x}_t$ shall be computable from $\boldsymbol{c}_t$ at all time). This is not the case when using standard HMMs to synthesize new observation sequences. Sampling from the model distribution $p(\mathcal{X})$ (explained in section 2.3.1.3) can lead to incoherences between static features $\boldsymbol{c}_t$ and their dynamic features $\boldsymbol{\Delta c_t}$ and $\boldsymbol{\Delta^2 c_t}$.

To get rid of this problem, they define the statistical model only with respect to $W\boldsymbol{c}$:

$$p(\boldsymbol{x} \mid \boldsymbol{h}) = p(W\boldsymbol{c} \mid \boldsymbol{h}) = \mathcal{N}(W\boldsymbol{c}; \boldsymbol{\mu_h}, \boldsymbol{\Sigma_h}) \tag{2.15}$$

where $\boldsymbol{\mu_h}$ ($3MT \times 1$) and $\boldsymbol{\Sigma_h}$ ($3MT \times 3MT$) are the mean vectors and covariance matrix corresponding to an entire state sequence $\boldsymbol{h}$.

Zen et al. then show how Eq 2.15 give rise to a new distribution over the static features:

$$p(\boldsymbol{c} \mid \boldsymbol{h}) = \mathcal{N}(\boldsymbol{c}; \hat{\boldsymbol{\mu}}_{\boldsymbol{h}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{h}})$$

where $\hat{\boldsymbol{\mu}}_{\boldsymbol{h}}$ is the mean vector $(MT \times 1)$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{h}}$ the covariance matrix $(MT \times MT)$ of static features along the state sequence. Note that for a same state h, $\hat{\mu}_h$ and $\hat{\Sigma}_h$ can vary along the state sequence $\boldsymbol{h}$: it is a non stationary state model.

For training, an EM type algorithm is then derived to reestimate $\boldsymbol{\mu}_{\boldsymbol{h}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{h}}$ for an observations sequence $\boldsymbol{x}$. It iterates between the two following steps :

$$\begin{aligned} \tilde{\boldsymbol{h}} &= \text{argmax}_{\boldsymbol{h}} p(\boldsymbol{c}, \boldsymbol{h}, \lambda) && \text{(E-step)} \\ \hat{\lambda} &= \max_{\lambda=(\boldsymbol{\mu}_{\boldsymbol{h}}, \boldsymbol{\Sigma}_{\boldsymbol{h}})} p(\boldsymbol{c}, \tilde{\boldsymbol{h}}, \lambda) && \text{(M-step)} \end{aligned}$$

One can see that it is not $(\hat{\boldsymbol{\mu}}_{\boldsymbol{h}})$ and covariances $(\hat{\boldsymbol{\Sigma}}_{\boldsymbol{h}})$ which are reestimated during learning but the standard (stationary state) HMM means $(\boldsymbol{\mu}_{\boldsymbol{h}})$ and covariances $(\boldsymbol{\Sigma}_{\boldsymbol{h}})$. Actually, $\hat{\boldsymbol{\mu}}_{\boldsymbol{h}}$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{h}}$ are not real parameters, they can be computed from closed formed formulas involving W and the standard HMM parameters. However $\hat{\boldsymbol{\mu}}_{\boldsymbol{h}}$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{h}}$ are used during inference (E-step and decoding).

Unfortunately, imposing consistency constraints (eq 2.14) between static and dynamic features *during* training is unacceptably costly ($O(M^3G^3)$ where G is the total number of Gaussian components (all models)).

In chapter 5, we will experiment and explain in more details a similar method (proposed by [76]) to improve the synthesis quality of HMMs. Because the method from [76] do not impose constraints (eq 2.14) *during* training but at synthesis time only, it does not suffer the computational overhead of Trajectory HMM training.

2.3.2.3 Parametric HMM

PHMM is another special class of non stationary state HMM where the Gaussian emission distributions are allowed to vary as a function of external (or contextual) variables. In speech recognition, these contextual variables may represent information regarding the speaker (gender, mother tongue, ...). In gesture recognition, it may be the height or corpulence of the actor etc...

A first attempt at conditioning HMM emitting distributions on contextual variables seems to be the work from [84] who proposed Parametric HMMs for gesture recognition.

In PHMM, Bobick et al. expressed the Gaussian means as a linear function of a contextual variable $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}_t \mid h_t = j) = \mathcal{N}(\boldsymbol{x}_t, \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta}), \Sigma_j)$$

with:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta}) = V^j \boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$$

where V^j is a matrix of $d \times c$ coefficients for state j (d being the dimension of the observations and c the dimension of the external or contextual variable $\boldsymbol{\theta}$) and $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ an offset vector.

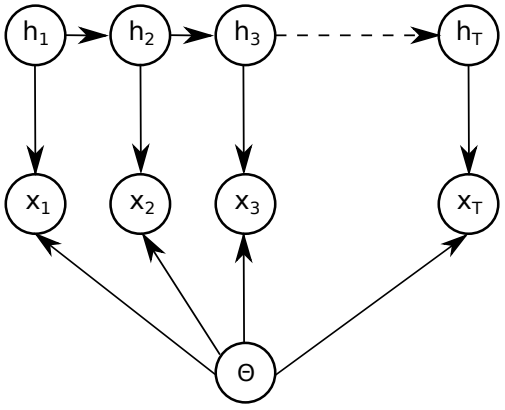


FIGURE 2.4: Parametric HMM as a Dynamic Bayesian Network

Reestimation of the mean parameterization can be done in a classic EM fashion simply by maximizing of the auxiliary function Q (defined in 2.3.1.2)) with respect to the parameters $Y^j = [V^j \ \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}]$. The linear form of the dependency between the contextual variable and the Gaussian mean induces closed form reestimation formulas which can be written as :

$$Y^j = \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{x}_t^k \boldsymbol{\theta}^{k^T}\right] \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{\theta}^k \boldsymbol{\theta}^{k^T}\right]^{-1} \tag{2.16}$$

where $\gamma_{k,t,j} = p(\boldsymbol{x}^k \mid h_t = j)$

Computing eq 2.16 requires inverting a $d \times (c+1)$ matrix. Yet if $c = (d-1)$ (the dimension of the contextual vector is the same as the dimension of the observation vector minus one), this is not more computationally demanding than training a full Gaussian HMM.

The contextual variable $\boldsymbol{\theta}$ is always observable (or known) at training time. At test time, this may not always be the case. To this regard, Bobick et al showed in [84] how it can be inferred. Once again, it involves maximizing the Q auxiliary function of HMMs with respect to $\boldsymbol{\theta}$. This procedure also yields closed form reestimation formulas because of

the simple linear dependency between the Gaussian mean and the contextual variable:

$$\boldsymbol{\theta} = \left[\sum_{k,t,j} \gamma_{k,t,j} {V^j}^T \Sigma_j^{-1} V^j\right]^{-1} \left[\sum_{k,t,j} \gamma_{k,t,j} {V^j}^T \Sigma_j^{-1} (\boldsymbol{x}_t^k - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})\right] \tag{2.17}$$

A very similar approach (Multiple Regression HMM, or MR-HMM) has been proposed in ([29]) for speech recognition, using fundamental frequency as external variable. Basically MR-HMM may be viewed as PHMM with time dependent external variables $\boldsymbol{\theta}_t$.

A second class of models called Variable Parameter HMMs (VPHMMs) have been introduced in ([17], [18]). In this approach, the means as well as the (diagonal) covariance matrices are expressed as a polynomial function of a static scalar environment variable.

2.3.2.4 Input Output HMM

Input Output HMM (IOHMM) [5] and PHMMs share similar mechanism in the way they parameterize HMM distributions using conditioning variables.

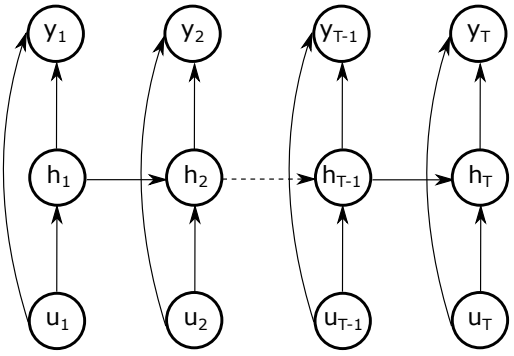


FIGURE 2.5: Input Output HMM as a Dynamic Bayesian Network

The main difference lie in that IOHMM use time dependent conditioning variables ($\boldsymbol{u}$) as well as different kinds of parameterization. In their work, Bengio et al. proposed to condition both the state transitions and labels distributions of HMMs using Neural Networks.

Their architecture is composed of several output networks, and state networks which are uniquely associated with a hidden state j.

First, the output network O_j predicts the labels distribution at each time step for a specific hidden state $h_t = j$ given an input $\boldsymbol{u}_t$.

$$p(\boldsymbol{y}_t \mid h_t = i, \boldsymbol{u}_t)$$

Secondly, the state network N_j predict the distribution of the current hidden state h_t for a previous hidden state $h_{t-1} = j$ given $\boldsymbol{u}_t$.

$$p(h_t \mid h_{t-1} = j, \boldsymbol{u}_t)$$

Finally, a softmax layer is used on top of the last layer of the state network to convert its outputs units activations a_j to state probabilities.

$$p(h_t \mid h_{t-1} = j, \boldsymbol{u}_t) = \frac{exp(a_j)}{\sum_k exp(a_k)}$$

This ensures that the output probabilities sum to 1. Note that there are no weights to learn in the softmax layer.

The whole system is jointly trained to maximize $p(\boldsymbol{y} \mid \boldsymbol{u})$ in a EM fashion just like a HMM. However, there are no closed form reestimation when the conditioning functions have non linearities like Neural Networks. In that case, training is performed via a gradient based Generalized Expectation Maximization algorithm ([7]).

The flexibility of IOHMM make them suitable for tasks like classification, recognition, or synthesis. In a classification setting, $\boldsymbol{y}_t$ might be the class, $\boldsymbol{u}_t$ the observation ($\boldsymbol{x}_t$) at time t.

They have several advantages over simple HMMs. They can introduce non linear conditioning on the state and output variables breaking the state stationarity. They can also be considered to be discriminatively trained because we optimize over the conditional likelihood of the class labels $p(\boldsymbol{y} \mid \boldsymbol{u})$ into a single model. Yet, in order to condition the state distributions with *external* variables (not observations) alike PHMMs, IOHMM need to define $\boldsymbol{u}_t$ as the contextual variable ($\boldsymbol{\theta}$) and $\boldsymbol{y}_t$ as the observation vector ($\boldsymbol{x}_t$). This choice of modeling then result in a non discriminative training criterion.

Finally, this architecture can have a lot of parameters and may require a lot of examples to be trained accurately. In [45], results indicate that HMMs outperforms IOHMMs on a gesture recognition experiment. Training this architecture requires a lot of data if we use complex functions such as Neural Networks to condition the transition and label distributions. Using two Neural Networks per hidden state make this architecture prone to overfitting and difficult to tune.

2.4 Discriminative Markov models

2.4.1 Conditional Random Fields

These different models partially alleviates some limitations of HMMs by relaxing simplifying assumptions. However the framework of Hidden Markov Model is not particularly suited for a pure classification task. Indeed, as a generative model, it maximizes $p(\boldsymbol{x}, y)\ \forall\ \boldsymbol{x} \in X_y$ independently for each class y. Yet, the only real task in classification is to discriminate between the classes. That is, for a class y, we should maximize for $p(y/\boldsymbol{x})\ \forall x \in X_y$ and consequently minimize over $p(y'/\boldsymbol{x})$ for every other competitive class $y' \neq y$, explicitly creating a gap between the probability of the true class and the others. This is what Conditional Random Fields (CRF) exactly does.

Basically, a Conditional Random Field ([46]) is an undirected graphical model expressing the joint probability of random variables ($\boldsymbol{y}$) when globally conditioned on an other set of random variables ($\boldsymbol{x}$).

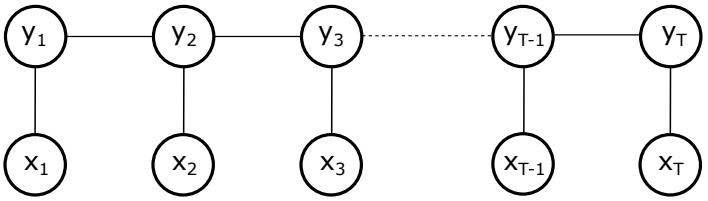


FIGURE 2.6: Simple example of a CRF with a chain structure

A node in the graph represents a variable which is dependent on its direct neighborhood (the variables that are linked to it), but conditionally independent from all the others variables conditioned on its neighborhood (Markov Blanket property in undirected graphs).

From the Hammersley-Clifford Theorem [35], the set of distributions consistent with these conditional independence rules is the same as the set of distributions that can be expressed as a factorization with respect to the maximal cliques of the graph (For recall, a clique is subgraph defined as a set of strongly connected nodes such that there is a link between every pair of two nodes).

We can then express the joint distribution of all variables $\boldsymbol{y}$ as a product of potential functions Ψ_C over the maximal cliques (C) of the graph

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z} \prod_C \Psi_C(y_C, \boldsymbol{x}) \tag{2.18}$$

where Z is a normalization constant (partition function) which ensures that the distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ remains a valid distribution. It is given by :

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \prod_{C} \Psi_C(y_C, \boldsymbol{x})$$

Potential functions Ψ_C are not constrained to be true probabilities, but they need to satisfy $\Psi_C(\boldsymbol{x}) \geq 0$ to ensure $p(\boldsymbol{y} \mid \boldsymbol{x}) \geq 0$. They can be defined as a dot product between a parameter vector λ_C and a feature map ϕ_C corresponding to a particular clique C :

$$\Psi_C(y_C, \boldsymbol{x}) = exp(\langle \lambda_C, \phi_C(y_C, \boldsymbol{x}) \rangle)$$

where

$$\langle \lambda_C, \phi_C(y_C, \boldsymbol{x}) \rangle = \sum_{k=1}^{K} \lambda_k f_k(y_C, \boldsymbol{x})$$

- with $\lambda_C = [\lambda_1, \ldots, \lambda_K]^T$
- and $\phi_C(y_C, \boldsymbol{x}) = [f_1(y_C, \boldsymbol{x}), \ldots, f_K(y_C, \boldsymbol{x})]^T$

Finally, we can rewrite equation 2.18 as :

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda) = \frac{1}{Z} exp \left[\sum_{C} \langle \lambda_C, \phi_C(y_C, \boldsymbol{x}) \rangle \right] \tag{2.19}$$

where Λ is the full parameters set of the model (concatenation of the C cliques parameter vectors λ_C)

The shape of the feature functions f_k essentially depends on the problem. They can be boolean or real valued. In Part Of Speech TAGging (i.e. the task of labeling each word of an input sequence with a tag), common features can be the presence or absence of an uppercase letter, the presence of particular tags in a clique etc ...

One can see in equation 2.19 that the conditional likelihood of $p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda)$ involves the summation of scores. However unlike in HMMs, these scores are not constrained to be true probabilities, instead, the normalization occurs at the global level of the computation by a rescaling factor $1/Z$. In fact, in [48], LeCun et al. advocate that for pure classification tasks, this is an unnecessary burden to constraint the distributions to be true probabilities.

When the graph is a chain or a tree, exact inference (i.e. finding the most probable sequence of labels) :

$$\text{argmax}_{\boldsymbol{y}} \; p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda) \tag{2.20}$$

can be handled via Viterbi or Forward Backward algorithms analogous to the one used in HMMs ([75]). In general graphs however, exact inference is not possible, but loopy belief propagation can be employed to give an approximation ([60]). This is why, in sequence labeling problems, one often assumes a linear chain structure (as shown in Figure 2.6). This simpler topology involves two types of cliques :

- local cliques relating an observation $\boldsymbol{x}_t$ to its current label y_t. They are described by a vector of parameters $\boldsymbol{\lambda}^{loc}$ and a feature map $\boldsymbol{\phi}(\boldsymbol{x}_t, y_t)$.
- transitions cliques relating two successive labels y_{t-1} and y_t. We note $\boldsymbol{\lambda}^{trans}$ their parameter vector and $\boldsymbol{\phi}(y_{t-1}, y_t)$ their feature map.

If the observation sequence has length T, there are T local cliques and T transition cliques, however, it is a common practice to share their parameters along time to limit overfitting problems. Hence, the conditional probability 2.19 can be written as :

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda) = \frac{1}{Z} exp \left[\sum_t \langle \boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y_t) \rangle + \sum_{t>1} \langle \boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(y_{t-1}, y_t) \rangle \right] \tag{2.21}$$

Training can be done by minimizing the expected negative log conditional likelihood on the training set, which is convex :

$$\mathcal{L}(\Lambda) = -\mathbb{E}_{\mathcal{D}_{train}} [\log p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda)] \tag{2.22}$$

2.4.2 Hidden Conditional Random Fields

Initially, Hidden CRF (HCRF) have been proposed as an extension of CRFs for dealing with more complex and structured data [34]. Indeed in CRF-based systems, there is usually one state per class (e.g. a POS tag) while there are several states corresponding to a given class in HRCF, alike in HMMs. The presence of several hidden states per label gives HCRF a clear advantage over CRF to model complex distributions.

Hence HCRF have been applied to signals such as gestures and images [65], handwriting [80] [23] , speech [74] [34] [68] or eye's movements [22] whether for signal labeling or classification tasks. Figure 2.7 gives an example of such a network.

Alike HMMs when used in sequence labeling problems, a label y is assigned a set of hidden states S_y. As a result, to a sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_T)$ corresponds a state sequences $\boldsymbol{h} = (h_1, \ldots, h_T) \in S^T$ (where S is the union of S_y for all classes). We will note $s(\boldsymbol{y})$ the set of all possible state sequences that correspond to a particular sequence of labels $\boldsymbol{y}$.

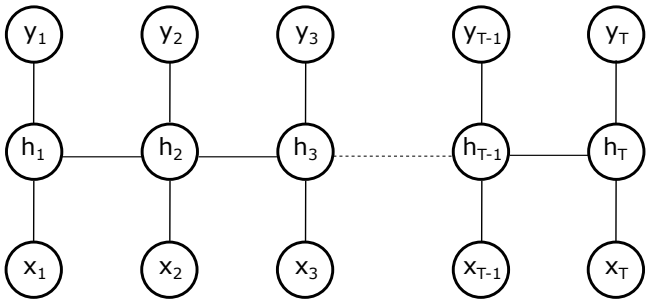


FIGURE 2.7: HCRF with a chain structure

The conditional probability of HCRFs can then be written as :

$$p(\boldsymbol{y}|\boldsymbol{x};\Lambda) = \sum_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x};\Lambda) \tag{2.23}$$

Compared to the eq 2.18 of CRFs, we now sum over all possible states sequences $\boldsymbol{h}$ which implies that the loss $\mathcal{L}(\Lambda)$ (eq 2.22) is no more convex.

To make the model tractable, HCRFs often consider a Markov network with transition cliques involving two successive hidden states and local cliques relating an observation and a hidden state at time t as discussed before.

In that case, the joint probability of a hidden state sequence and its corresponding sequence of labels is given by :

$$p(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x};\Lambda) = \frac{1}{Z} exp\left[\sum_t \langle \boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, h_t) + \sum_{t>1} \langle \boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(h_t, h_{t-1})\rangle\right] \tag{2.24}$$

Where $Z = \sum_{\boldsymbol{y}} \sum_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x};\Lambda)$ is a normalization term.

One can see the similarity between the expression 2.24 and the conditional probability of standard CRFs exposed in equation 2.21. In fact the hidden state sequence in HCRFs plays the role of the label sequence in CRFs.

Hence, finding the best hidden state sequence $\text{argmax}_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{h} \mid \boldsymbol{y},\boldsymbol{x};\Lambda)$ or marginalizing out the hidden states sequence $\sum_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x};\Lambda)$ can also be done with similar Viterbi and Forward Backward algorithms as employed in linear chain CRFs.

Note that because we have chosen to encode a direct correspondence between a hidden state and a label, and because we did not defined any cliques between the labels, inferring the most likely sequence of labels :

$$\text{argmax}_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x};\Lambda) \tag{2.25}$$

can be efficiently found by simply computing the best hidden state sequences via Viterbi

$$\text{argmax}_{\boldsymbol{h}\in S^T} p(\boldsymbol{h} \mid \boldsymbol{x}; \Lambda) \tag{2.26}$$

2.4.2.1 Training HCRF by maximizing the conditional likelihood

Training is performed through minimization of the expected negative log conditional likelihood on the training set which is non convex :

$$\begin{aligned}\mathcal{L}(\Lambda) &= -\,\mathbb{E}_{\mathcal{D}_{train}}\left[\log p(\boldsymbol{y} \mid \boldsymbol{x}; \Lambda)\right] \\ &= -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{train}} \log \sum_{h\in s(\boldsymbol{y})} p(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x}; \Lambda)\end{aligned} \tag{2.27}$$

There is no closed form reestimation of the parameters, so gradient descent must be employed. For ease of reading, we assume a single sequence in the dataset. Then the gradient of $\mathcal{L}(\Lambda)$ with respect to the parameters of the local cliques $\boldsymbol{\lambda}^{loc}$ and the parameters of the transition cliques $\boldsymbol{\lambda}^{trans}$ are given by :

$$\begin{aligned}\frac{\partial \mathcal{L}(\Lambda)}{\partial \lambda_i^{loc}} &= -\sum_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{h} \mid \boldsymbol{y}, \boldsymbol{x}) \sum_{t=1}^{T} \phi(\boldsymbol{x}_t, h_t = i) \\ &\quad + \sum_{\boldsymbol{h}'\in S^T} p(\boldsymbol{h}' \mid \boldsymbol{x}) \sum_{t=1}^{T} \phi(\boldsymbol{x}_t, h'_t = i) \\ \frac{\partial \mathcal{L}(\Lambda)}{\partial \lambda_{i\to j}^{trans}} &= -\sum_{\boldsymbol{h}\in s(\boldsymbol{y})} p(\boldsymbol{h} \mid \boldsymbol{y}, \boldsymbol{x}) \sum_{t=2}^{T} \phi(h_{t-1} = j, h_t = i) \\ &\quad + \sum_{\boldsymbol{h}'\in S^T} p(\boldsymbol{h}' \mid \boldsymbol{x}) \sum_{t=2}^{T} \phi(h'_{t-1} = j, h'_t = i)\end{aligned}$$

When optimizing $\mathcal{L}(\Lambda)$, the gradient with respect to the parameters Λ actually contains two terms. The first term pushes down the energy (or negative likelihood) of the good labeling $\boldsymbol{y}$ while the second term, pulls up the energy of all labelings. By doing so, the optimization creates an explicit gap between the likelihood of the correct labeling and incorrect ones.

2.5 Conclusion

We have exposed here two families of Markovian model for time series modeling.

First, generative HMMs provide several interesting possibilities to capture the variability of observation sequences :

- On the one hand, Trended and Trajectory HMMs use non stationary state distributions while they rely on two different approaches. The Trended HMM models observations as a polynomial function of the state sojourn time whereas the Trajectory HMM introduces constraints between static and dynamic features during training.
- On the other hand, InputOutput and Parametric HMMs condition the state distributions with external variables which also results in non stationary state distributions.

Yet, compared to other approaches the formulation of Parametric HMMs has several advantages. The simplicity of its parameterizations make them easier to train and less prone to overfitting than InputOutput HMMs while they do not suffer major complexity problems alike Trended or Trajectory HMMs. Moreover, it seems reasonable to think that an important part of the variability in observation sequences is indeed the consequence of a few external variables.

Secondly, discriminative models like HCRFs seems more suited to pure classification tasks. By optimizing the conditional likelihood $p(\boldsymbol{y} \mid \boldsymbol{x})$ they explicitly create a gap between the probability of the correct labeling and incorrect ones. Unfortunately, because they do not model the data distribution, they lack the modeling ability of previous methods and their capacity for synthesis.

In the following, we will present the framework of Contextual Hidden Markov Models (CHMM). Starting from the formulation of Parametric HMM, CHMM will propose new ways to influence the HMM distributions with contextual variables that may remain fixed or change along the observation sequence. Subsequently we will show how the similarity between HCRFs and CHMMs offers a simple and efficient way to incorporate contextual modeling into a pure discriminative model, the Contextual Hidden Conditional Random Field. Finally, our experiments will show how this better modeling capacity can translate into performance for various tasks.

Chapter 3

Contextual Hidden Markov Models

Contents

3.1 Introduction 46
3.2 Single Gaussian Contextual HMM 47
3.2.1 Mean parameterization 47
3.2.2 Covariance parameterization 48
3.2.3 Transitions parameterization 49
3.2.4 Bayesian perspective 50
3.3 Training 50
3.3.1 With covariances parameterized 51
3.3.2 With transitions parameterized 52
3.3.3 Dynamic context 53
3.3.4 Gaussian mixtures 53
3.3.5 Tuning the gradient step size 53
3.4 CHMM relative to similar approaches 56
3.4.1 Variable Parametric HMMs 56
3.4.2 Maximum Likelihood Linear Regression 57
3.4.3 Context dependent modeling 58
3.5 Application to the classification of handwritten characters 59
3.5.1 Dataset 59
3.5.2 Preliminary results 60
3.5.3 Extended results 63
3.6 Conclusion 65

3.1 Introduction

One topic we are concerned with in this study is how to handle variabilty. In HMMs (as well as in HCRFs) states are mutually exclusive so that it requires K states to get K different output distributions. The most popular approach to handle variability consists in increasing the number of states, in increasing the size of Gaussian mixtures in HMMs, in using context dependent unit (e.g. phone) models. These ideas are easy to implement but this quickly leads to too numerous parameters yielding over-fitting. To overcome this difficulty the speech recognition community has focused on different ways to tie parameters. Parameters can be shared between states which are acoustically indistinguishable ([41], [86], [66]). Another strategy is to tie parameters at the distribution level ([3], [63]). A pool of Gaussian is shared inside a model (partially tied), or across all models (fully tied). Yet these strategies allow capturing a local variability only, while keeping the number of parameters limited.

Our starting point is an alternative approach for handling variability. We assume that an important part of the variability between observation sequences may be modeled by a few contextual variables (which may be hidden or observed) that remain fixed all along a sequence or that vary slowly with time. For instance a sentence may be uttered quite differently according to the speaker emotion. A gesture may have more amplitude if it is performed slower, and its overall shape depends on the weight and on the height of the performer. Such a variability cannot always be removed through preprocessing or normalization and would not be captured accurately by the classical approaches above. Yet such a variability would benefit from a specific handling in HMMs.

Few researchers have tackled this problem by designing a HMM whose probability distribution depends on contextual variables (i.e. the context, that we note $\boldsymbol{\theta}$). [84] proposed Parametric Hidden Markov Models where the means of Gaussian distribution vary linearly as a function of the context. As the output distribution depends not only on the state but also on the context, a model may express many distribution with a limited number of additional parameters. [89], [18] and [29] investigated rather similar approaches.

All these approaches differ by the nature of the dependency of HMM parameters to context variables, the ability to deal with dynamic context variables, i.e. evolving with time, the ability to infer context variables at test time.

We build here upon these pioneer works and propose contextual extension of HMMs. We first extend parametric HMMs of [84] and we propose Contextual Hidden Markov models (CHMMs) that rely on the parameterization of the probability distribution of a HMM (i.e. means and covariances matrices instead of means only in [84]) by a set of contextual

variables that may vary in time. Than, we show how the transition probability between states can also be parameterized.

In the following sections, we first motivate and introduce our modeling framework for generative models and detail the definition and the learning of our Contextual HMMs. We focus first in section 3.2 on the case of single Gaussian CHMM when $\boldsymbol{\theta}$ is static and remain fixed all along a sequence. Then we discuss in sections 3.3.3-3.3.4, two variants, dealing with dynamic $\boldsymbol{\theta}$ and using Gaussian mixtures.

Next, we discuss the fundamental differences between CHMMs and similar approaches. Finally, we present experimental results showing the benefits of such modeling.

3.2 Single Gaussian Contextual HMM

We expose here the parameterizations that can be employed in the Contextual HMMs. Such a modeling allows the HMM distributions to vary according to a contextual variable. We first begin by introducing the parameterization of Gaussian means, which has already been proposed by Bobick et al. in [84] under the name of Parametric HMMs. Next we move on to show how the Gaussian covariances, and transition probabilities can also be parameterized.

3.2.1 Mean parameterization

Assume that for any observation sequence $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_T)$, where $\boldsymbol{x}_t$'s are d-dimensional feature vectors we are given a set of contextual variables $\boldsymbol{\theta}$ which is a vector of dimension c. $\boldsymbol{\theta}$ might be the age and gender of a speaker for speech signals, or a set of physiological features such as height and weight for gestures, or some quantities that are computed from the input sequence $\boldsymbol{x}$ such as its length.

We first define the mean $\hat{\boldsymbol{\mu}}_{\boldsymbol{j}}$ (d-dimensional vector) of the Gaussian distribution in state j to be a linear function of $\boldsymbol{\theta}$. In order to keep notations compact we consider an augmented $\boldsymbol{\theta}$ vector with all contextual variables plus a last additional component equal to 1. Hence, from now on, $\boldsymbol{\theta}$ is a c-dimensional vector $\boldsymbol{\theta} = [\text{Contextual variables}, 1]^T$ with $(c-1)$ contextual variables and a c^{th} component equal to 1. We consider that the mean in state j is defined as:

$$\hat{\boldsymbol{\mu}}^{\boldsymbol{j}}(\boldsymbol{\theta}) = Y^j \boldsymbol{\theta} \tag{3.1}$$

where Y^j is a $d \times c$ matrix.

The above formulation is equivalent to writing

$$\hat{\mu}^j(\boldsymbol{\theta}) = V^j\boldsymbol{\theta} + \bar{\boldsymbol{\mu}}^{\boldsymbol{j}} \tag{3.2}$$

with V^j, $\bar{\boldsymbol{\mu}}^{\boldsymbol{j}}$ and Y^j being related by: $Y^j = \left[V^j \quad \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}\right]$.
The vector $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ is $d \times 1$ offset vector which may be viewed as an average mean vector (eventually obtained from a traditionally learned HMM) that is modified by the linear transform part.

As we already pointed, mean parameterization of Single Gaussian HMMs has already been proposed by Bobick et al. in [84] as Parametric HMMs (PHMM). For clarity however, we will note μCHMM contextual HMMs when only mean vectors depend on $\boldsymbol{\theta}$ or $\mu\Sigma$CHMM when mean vectors and covariance matrices depend on $\boldsymbol{\theta}$.

3.2.2 Covariance parameterization

We go further by parameterizing covariance matrices as well. While some authors have proposed to define similarly diagonal covariance matrix that depends on external variables ([89]) we propose a full covariance parameterization scheme. Actually we want the covariance matrix $\bar{\Sigma}$ to be modified in such a way that each of its component $\bar{\Sigma}_{u,v}$ is transformed into $\bar{\Sigma}_{u,v} \times \alpha_u \times \alpha_v$ where α values depend on contextual variables $\boldsymbol{\theta}$. This allows providing an additional but limited degree of freedom to the model, allowing more expressive power while limiting over-training risk. This may be done according to:

$$\begin{aligned} \hat{\Sigma}^j(\boldsymbol{\theta}) &= D^j(\boldsymbol{\theta}) \times \bar{\Sigma}^j \times D^j(\boldsymbol{\theta}) \\ \text{with } D^j(\boldsymbol{\theta}) &= diag(exp(Z^j\boldsymbol{\theta})) \end{aligned} \tag{3.3}$$

where $\hat{\Sigma}^j$ is the $d \times d$ covariance matrix in state j, $\bar{\Sigma}^j$ is a $\boldsymbol{\theta}$ independent covariance matrix that is transformed by the above operation (it may be for instance initialized as a matrix learned in a standard HMM), Z^j is a $d \times c$ matrix with the same shape as for the mean parameterization. Actually one may see Z^j as $Z^j = \left[U^j \quad \widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}}\right]$ where U^j is a $d \times (c-1)$ matrix, $\widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}}$ is a $d \times 1$ offset vector and $\boldsymbol{\theta}$ is the same vector as before with a last component which is systematically equal to 1. Here we note the exponential of a matrix A, $exp(A)$, to be the matrix of the exponential function applied component-wise to all elements of A, and we note $diag$ the function transforming a vector to a diagonal matrix. The use of the exponential function ensures elements of $D^j(\boldsymbol{\theta})$ to be strictly positive, which makes $\hat{\Sigma}^j(\boldsymbol{\theta})$ a valid covariance matrix provided $\bar{\Sigma}^j$ is one.

At the end, as expected, the shape of the covariance matrix makes the term at u^{th} row and v^{th} column equal to:

$$\hat{\Sigma}^j_{u,v}(\boldsymbol{\theta}) = D^j_{u,u}(\boldsymbol{\theta}) \times D^j_{v,v}(\boldsymbol{\theta}) \times \bar{\Sigma}^j_{u,v}(\boldsymbol{\theta})$$

Figure 3.1 shows the effect of such a parameterization on the shape of a covariance matrix in a two dimensional data space. An original covariance matrix (upper left figure) is modified by various D matrices yielding three new covariance matrices (three other plots). Here each of the four covariance matrices is illustrated by few curves of isoprobability (ellipses).

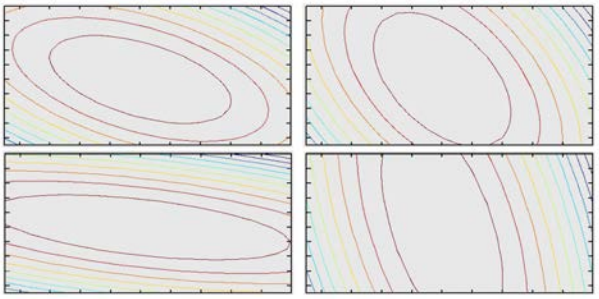


FIGURE 3.1: Examples of the parameterization of a covariance matrix in Eq. (3.3) on 2-dimensional data. An original covariance matrix (Top left) is transformed with various D matrices: $D = diag([1\ 2])$ (Top right), $D = diag([2\ 0.9])$ (Bottom left), $D = diag([0.8\ 3])$ (bottom right). Each covariance matrix shape is illustrated by ellipses corresponding to isoprobability curves.

3.2.3 Transitions parameterization

Finally, transition probabilities also play a role in modeling the data in HMM. We might also want to parameterize the state transition probabilities by a contextual variable so that it may increase the fitness of the model to a particular observation sequence. Indeed, because each observation sequence can be defined by a different context, it may be better to use different state probabilities as opposed to using shared transition probabilities for all observation sequences in a specific class. In this case, we define the state transition distribution $a_{i,j}$ from i^{th} state to j^{th} state at time t by :

$$\hat{a}_{i,j}(\boldsymbol{\theta}) = \frac{exp(\log \bar{a}_{ij} + \boldsymbol{w}^T_{ij}\boldsymbol{\theta})}{\sum_k exp(\log \bar{a}_{ik} + \boldsymbol{w}^T_{ik}\boldsymbol{\theta})} \tag{3.4}$$

with $\bar{a}_{ij}$ the original transition probabilities from state i to state j in a HMM or a CHMM without transition parameterization

and $\boldsymbol{w}_{ij}$ a weight vector the same dimension as $\boldsymbol{\theta}$.

It is interesting to note that CHMM subsumes standard HMM (with means $\bar{\boldsymbol{\mu}}^{\boldsymbol{j}}$, covariance matrices $\bar{\Sigma}^j$ and transition probabilities $\bar{a}_{ij}$) by setting V^j and U^j to null matrices and by setting $\widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}}$ and $\boldsymbol{w}_{ij}$ to null vectors.

3.2.4 Bayesian perspective

In CHMMs, the influence of the contextual variables over of emitting and transition distributions can be represented as the following graphical model

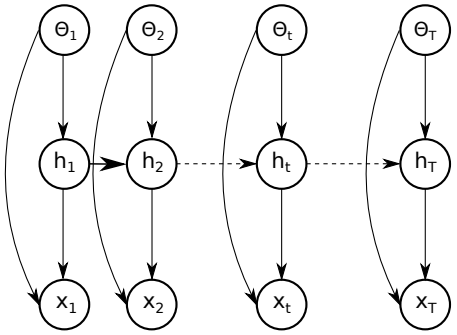


FIGURE 3.2: Bayesian representation of Contextual HMM when emitting and transition probabilities are parameterized by a dynamic contextual variable θ_t

This is to oppose to simple Parametric HMMs where the contextual variable is time independent and can only influence the Gaussian mean of hidden states (graphical model shown in figure 2.4).

At training time, the contextual variable $\boldsymbol{\theta}$ is given (or observable) but this may not always be the case at test time. To this regard, Bobick et al. showed in [84] how the contextual variable $\boldsymbol{\theta}$ can be inferred (see 2.3.2.3).

In CHMMs, it is also possible to infer the contextual variable at test time. However, there are no closed form solutions owing to the new kinds of parameterizations (for transitions probabilities, and Gaussian covariances). Additionally, the use of dynamic contextual variables would require inferring $\boldsymbol{\theta}_t$ for each time step. Hence, CHMM will be employed with a *given* context at test time.

3.3 Training

We consider we get a set of training sequences along with their labels (i.e. classes) and their context variables $\{(\boldsymbol{x}^k, y^k, \boldsymbol{\theta}^k | k = 1..N)\}$.

Training consists in modifying the matrices Y^j, Z^j and vectors W_{ij} so as to maximize the likelihood of the training sequences. Optimization is carried in EM style as for standard HMMs. To learn μCHMM, we use the closed form re-estimation formula detailed in previous sections in a standard EM setting. Yet when learning a $\mu\Sigma$CHMM there does not exist closed form re-estimation formula for parameters on covariance matrices so that we resort to use gradient ascent in the M step of every EM iteration.

3.3.1 With covariances parameterized

Learning a $\mu\Sigma$CHMM is performed in few successive steps that we describe now.

- First, we learn a μCHMM with parameterized means only, which is equivalent to learning a Parametric HMM as proposed by Bobick et al. in [84]. This may be done by using the following closed form re-estimation formulas (proofs can be found in [84]) :

$$Y^j = \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{x}_t^k \boldsymbol{\theta}^{k^T}\right] \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{\theta}^k \boldsymbol{\theta}^{k^T}\right]^{-1} \tag{3.5}$$

$$\Sigma^j = \frac{\sum_{k,t} \gamma_{k,t,j} (\boldsymbol{x}_t^k - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta^k}))(\boldsymbol{x}_t^k - \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta^k}))^T}{\sum_{k,t} \gamma_{k,t,j}} \tag{3.6}$$

  where we use the usual HMM notation for $\gamma_{k,t,j}$, that stand for $\gamma_{k,t,j} = p(h_t = j|\boldsymbol{x}^k, y^k)$.

- Then, for every state j, we set $\bar{\Sigma}^j = \Sigma^j$

- Second, we fix all model parameters and we re-estimate Z^j only.

  We initialize $Z^j = 0$ which allows starting from the covariance matrix obtained in first step : $\hat{\Sigma}^j(\boldsymbol{\theta}) = \Sigma^j$

  Re-estimation of Z^j is performed via the Generalized Expectation Maximization (GEM) algorithm, by computing the derivative of the auxiliary function Q of the HMM with respect to Z^j and doing a gradient ascent. We recall that $Q(\Lambda, \Lambda') = \mathbb{E}_{\boldsymbol{h}|\mathbf{x},\Lambda'}[\log P(\mathbf{x}, \mathbf{h} \mid \Lambda)]$ where Λ' stands for the current set of the CHMM parameters while Λ stands for the updated values of the CHMM parameters.

  Omitting details one can show without difficulty that :

$$\frac{\partial Q}{\partial Z^j} = \sum_{k,t,i} M_{i,i}^{k,t,j} \times \frac{\partial D_{i,i}^j(\boldsymbol{\theta}^k)^{-1}}{\partial Z^j} \tag{3.7}$$

with $M_{k,t,j} =$

$$\gamma_{k,t,j}\left[D^j(\boldsymbol{\theta}^k) - \bar{\Sigma^j}^{-1} D^j(\boldsymbol{\theta}^k)^{-1}(\boldsymbol{x}_t^k - \hat{\boldsymbol{\mu}}^{\boldsymbol{j}}(\boldsymbol{\theta}^{\boldsymbol{k}}))(\boldsymbol{x}_t^k - \hat{\boldsymbol{\mu}}^{\boldsymbol{j}}(\boldsymbol{\theta}^{\boldsymbol{k}}))^T\right]$$

where

$$\frac{\partial D^j_{i,i}(\boldsymbol{\theta}^k)^{-1}}{\partial Z^j_{m,n}} = \begin{cases} \frac{-\boldsymbol{\theta}^k_n}{D^j_{i,i}(\boldsymbol{\theta}^k)} & \text{if } i = m \\ 0 & \text{otherwise} \end{cases}$$

We do not perform simultaneous optimization of means and covariance matrices parameterization (Y^j and Z^j) since it appears to bear some difficulties during optimization. Instead we investigated here a sequential optimization of these two sets of parameters. Yet one could imagine to iterate these two steps leading to a kind of coordinate ascent optimization routine but we did not investigate this up to now.

3.3.2 With transitions parameterized

The scheme to train a transition parameterized CHMM is similar to the training of covariance parameterization. As there is no closed form solution for reestimating w_{ij}, we proceed in two steps.

- First we learn a HMM or a CHMM
- Then for every transition ij between state i and j

  set $\bar{a}_{ij}$ with the transition probability learned in previous step.

  set $\boldsymbol{w}_{ij} = 0$ which allows starting from the transition probabilities obtained in first step.

  reestimate $\boldsymbol{w}_{ij}$ with GEM by computing the derivative of Q with respect to $\boldsymbol{w}_{ij}$ and perform gradient ascent.

It can be shown that the gradient of Q with respect to $\boldsymbol{w}^{ij}$ is given by :

$$\frac{Q}{\boldsymbol{w}_{ij}} = \sum_{k,t}\left[\gamma_{i,j,t,k} - \sum_{i\bar{j},t,k}\hat{a}_{i,\bar{j}}(\boldsymbol{\theta}^k)\right]\boldsymbol{\theta}^k \tag{3.8}$$

3.3.3 Dynamic context

Now suppose $\boldsymbol{\theta}$ depends on time, for instance it may be an estimation of the instantaneous speed of a gesture, an estimation of fundamental frequency in a speech signal, etc. We then use the following definitions :

$$\begin{aligned}
\hat{\mu}^j(\boldsymbol{\theta}_t) &= Y^j \boldsymbol{\theta}_t \\
\hat{\Sigma}^j(\boldsymbol{\theta}_t) &= D^j(\boldsymbol{\theta}_t) \times \bar{\Sigma}^j \times D^j(\boldsymbol{\theta}_t) \\
\hat{a}_{i,j}(\boldsymbol{\theta}_t) &= \frac{e^{\log \bar{a}_{ij} + \boldsymbol{w}_{ij}^T \boldsymbol{\theta}_t}}{\sum_k e^{\log \bar{a}_{ik} + \boldsymbol{w}_{ik}^T \boldsymbol{\theta}_t}}
\end{aligned}$$

It is straightforward to show that the re-estimation formulas Eq.(3.5)(3.6)(3.7)(3.8) apply if one changes systematically $\boldsymbol{\theta}$ to $\boldsymbol{\theta}_t$. New re-estimation formulas are then simple extensions of Eq.(3.5)(3.6)(3.7)(3.8). For instance, the closed form solution for Y^j becomes:

$$Y^j = \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{x}_t^k \boldsymbol{\theta}_t^{k^T} \right] \left[\sum_{k,t} \gamma_{k,t,j} \boldsymbol{\theta}_t^k \boldsymbol{\theta}_t^{k^T} \right]^{-1}$$

3.3.4 Gaussian mixtures

Extending single Gaussian models to Gaussian mixture modeling may be done easily. The new pdf of l^{th} Gaussian in state j is then defined as :

$$\begin{aligned}
\hat{\mu}^{j,l}(\boldsymbol{\theta}_t) &= Y^{j,l} \boldsymbol{\theta}_t \\
\hat{\Sigma}^{j,l}(\boldsymbol{\theta}_t) &= D^{j,l}(\boldsymbol{\theta}_t) \times \bar{\Sigma}^{j,l} \times D^{j,l}(\boldsymbol{\theta}_t)
\end{aligned}$$

There is no difficulty to derive new re-estimation formulas similar to (3.5), (3.6), (3.7) by adding a component index l to all necessary quantities.

3.3.5 Tuning the gradient step size

For covariance or transitions parameterizations, we use a gradient ascent procedure.

Hence training CHMMs with theses types of parameterizations require setting an appropriate gradient step size. It is a difficult procedure and generally the step size is chosen after several trials and errors on validation data. On the other hand, one can use an

easier approach known as linesearch. Basically linesearch test several gradient step sizes and choose the one that improves the best the likelihood of the data under the model.

In our implementation given below, linesearch is a recursive process which cuts the step size space into pieces and recursively do so for a maximum depth. At each recursion level, we evaluate all the step sizes.

Of course, linesearch is much more computationally expensive than choosing the gradient step size at hand. First because it requires computing the likelihood of training sequences (or a bunch of them) for each gradient step, and secondly because it evaluates several step sizes recursively. Hence, we only used this procedure (on smaller validation data) as a tool to find an acceptable value for the gradient step size.

Algorithm 1 step size linesearch

current_energy ← compute_energy(model) // *average negative loglikelihood of the training data*
eta_initial ← 0.001
depth ← 1
etas ← [0 1 2 3 4] × eta_initial
energies ← [current_energy nan nan nan nan]
function [ETA, ENERGY]=LINESEARCH(model, gradient, etas, energies, depth)
    // *save the initial unupdated model*
    old_model ← model
    for k ← 1 to 5 **do**
        // *retrieve the initial model*
        model ← old_model
        if isnan(energies(k)) **then**
            // *make gradient update with etas(k) on model parameters*
            model ← updateParameters(model, etas(k))
            // *compute the new energy of this update*
            energies(k) ← compute_energy(model)
        end if
    end for
    // *select the minimum energy update*
    $k \leftarrow min(energies)$
    if $depth > 0$ **then**
        if $k = 1$ **then**
            // *create 4 new evenly spaced etas between two values*
            etas ← linspace(etas(1), etas(2))
            energies ← [energies(1) nan nan nan energies(2)]
        else if $k = 5$ **then**
            // *create 4 new evenly spaced etas between two values*
            etas ← linspace(etas(1), 4×(etas(5)-etas(1)))
            energies ← [energies(1) energies(5) nan nan nan]
        else
            // *fit a two degree polynomial on etas/energies values*
            // *compute the value at the minimum for eta*
            eta_min ← quadraticLineSearch(etas, energies)
            etas ← [etas eta_min]
            energies ← [energies nan]
            // *sort energies and etas by ascending eta values*
            [energies, etas] ← sort(energies, etas)
        end if
    else
        eta ← etas(k)
        energy ← energies(k)
    end if
    // *recursive call to the next level of linesearch*
    [eta, energy] ← LINESEARCH(model, gradient, etas, energies, depth -1)
end function

3.4 CHMM relative to similar approaches

Handling variability is a major focus when dealing with sequences and signals. Variability may be the consequence of various effects that may be eventually combined. As a consequence, one may distinguish between different kinds of variability.

For instance a speech signal is fundamentally different if the speaker is a male or a female, and two speakers utter differently a same word. This variability is usually modeled by multiplying models, e.g. by exploiting one model for male speakers and one model for female speakers.

There is a more fine-grained variability in that a single speaker never utters exactly the same way a single word. Also a human will never perform the same gesture exactly the same way. Such a variability depends on many factors that are usually unknown, like the emotion, the physical state, etc. This variability may be handled by increasing the number of Gaussian in Gaussian mixtures. Going further, there is another variability which is related to noise, to the recording material etc, this is usually handled through a preprocessing step which aims at removing this variability.

While there are historically standard ways to handle such kinds of variability, a number of other approaches have considered the benefit of explicitly including their modeling in the framework of markovian models. We introduce them here and discuss their difference compared to CHMMs.

3.4.1 Variable Parametric HMMs

A first attempt for conditioning HMM parameters on environment variables seems to be the work from [84] who proposed Parametric HMMs (PHMMs) for gesture recognition, context variables were related to the amplitude of the gestures. As we already said our modeling framework includes PHMM as a special case when ignoring parameterization of covariance matrices and transitions. A very similar approach (Multiple Regression HMM, or MR-HMM) has been proposed in [29] for speech recognition, using fundamental frequency as context variable. Basically MR-HMM may be viewed as PHMM with time dependent context variables $\boldsymbol{\theta}$. These models are again embedded in our framework.

A second class of models called Variable Parameter HMMs (VPHMM) are closely related to our approach. This type of model has been introduced in [18], [17]. It was proposed in the context of speech recognition to improve robustness to noisy conditions. In this approach, the means as well as the (diagonal) covariance matrices are expressed as a polynomial function of a static scalar environment variable v:

$$\hat{\mu}_s(v) = \sum_j \boldsymbol{w}_{s,j} v^j$$
$$\hat{\Sigma}_s(v) = \Lambda(v)\hat{\Sigma}_s$$

where $\boldsymbol{w}_{s,j} = [w_{s,j}(0) \dots w_{s,j}(D)]^T$ is a D dimensional parameter vector modifying the Gaussian mean in state s
v^j is the scalar environment variable raised at the j^{th} power exponent

and the scaling matrix $\Lambda(v)$ is expressed as :

$$\Lambda(v) = \begin{pmatrix} e^{\sum_j z_{s,j}(0)v^j} & & 0 \\ & \ddots & \\ 0 & & e^{\sum_j z_{s,j}(D)v^j} \end{pmatrix}$$

where $\boldsymbol{z}_{s,j} = [z_{s,j}(0) \dots z_{s,j}(D)]^T$ is a D dimensional parameter vector modifying the Gaussian covariance in state s

In [89], Deng et al refines VPHMM using piecewise spline interpolation instead of polynomial regression and handle time dependent environment multi dimensional variable $\boldsymbol{v}_t$.

It must be clear that that our approach already handle dynamical multi dimensional contextual variables. Additionally, CHMMs can easily use a P order polynomial regression simply by augmenting the contextual variable $\boldsymbol{\theta}$ with its power exponents : $\boldsymbol{\theta} = \left[\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots, \boldsymbol{\theta}^P\right]^T$.

Also compared to these works, our approach has several advantages. First we propose a parameterization of the transitions which is not the case of VPHMM. Secondly, we devise a full covariance matrix parameterization where they provide a diagonal one. Lastly, in VPHMM the parameterization of the emission distribution make each dimension of the observation to depend only on a *single* dimension of the contextual variable. This makes VPHMM not suited to exploit vector typed variables such as emotion, gender etc...typically encoded as a variable of several dimensions. In this case, each dimension of the observation should depend on "every" dimension of the contextual variable.

3.4.2 Maximum Likelihood Linear Regression

There is another well known approach that resemble our framework. It allows the adaptation of a standard HMM means and covariances. Maximum Likelihood Linear

Regression (MLLR) has been made popular as a speaker adaptation technique in speech recognition. Generally, there is not enough utterances of a single speaker to estimate the parameters of a speech recognition system, so, a speaker independent model is trained on the utterances of many speakers. However, it is well known that speaker dependent models are more accurate if one have enough data for each speaker. Hence, once a speaker independent HMM is trained, one can reestimate its means and covariances for a specific speaker.

To do so, one has to maximize the likelihood of the speaker adaptation data with respect to the following transforms parameters.

$$\hat{\mu} = W_s \boldsymbol{\xi_s}$$
$$\hat{\Sigma}_s = H_s \bar{\Sigma}_s H_s^T$$

where W_s is linear transform ($d \times d + 1$ matrix) for the hidden state s of the original HMM. $\boldsymbol{\xi}_s = [1, \mu_1, \dots \mu_n]^T$ is the extended mean vector of the original HMM in state s.

One can see that the mean transform of MLLR is a special case of CHMM where the contextual variable θ would be a static vector equal to the mean of the original HMM. The variance transform has also a similar shape compared to eq (3.3) however, it is not directly comparable as the matrix H can be full and does not depend on any variable.

The paradigm of MLLR is however much more restricted than CHMM. In fact, it is restricted to posterior means and variances adaptation of the original HMM means and variances. On the contrary, CHMMs can learn means, transitions and variances transforms that use any kind contextual information θ. Furthermore, CHMMs do not require retraining on separate a dataset to learn its transforms.

3.4.3 Context dependent modeling

For most of signal labeling tasks such as the recognition of speech, gesture or handwriting, there is another well known variability which comes from some transitional effect. This is the usual coarticulation effect in speech where the realization of a phone depends on the previous and of the next phone. A similar phenomenon arises in handwriting too, it is called ligature, when the beginning of the writing of a letter depends on the previous letter and the ending of its writing depends on the letter to come. Handling such a variability has particularly been investigated in the speech recognition literature first, e.g. [49] used right context dependent phone HMMs while [53] investigated the use of triphone models (one phone model for every context of a previous and a next phone).

To overcome the problem of learning many such triphone models, i.e. to improve generalization, various strategies have been proposed to cluster the possible contexts using expert knowledge. Such a strategy has more recently been exploited in the handwriting recognition field when [6] used a trigraph model for Arabic handwriting recognition, i.e. one model for every character and for every context of the previous and the next character. Our approach and the related works that we have described in this section probably cannot be straightforwardly used to handle such a variability. We rather view the two approaches, using models exploiting contextual information and using models for every context, as probably complementary, i.e. meaning that they could and should be combined easily and with benefit.

3.5 Application to the classification of handwritten characters

We now investigated the behavior of standard HMMs and CHMMs on an isolated off-line handwritten character classification, using a part of the IAM dataset [77]. This application only deal with mean and covariance parameterized Contextual Hidden Markov Models, parameterization of the transitions is discussed in chapter 5.

3.5.1 Dataset

Every sequence is an image of an isolated handwritten character which is pre-processed and represented at the end as a sequence of 9-dimensional observation vectors.

Precisely, the observation vector computes geometric features extracted on a sliding window of 1 pixel width over the character image. Theses geometric features include :

- The proportion of black pixels in the sliding window.
- The first and second order moments of black pixels
- The lower and upper contour positions with their respective derivatives
- The number of transitions from black to white pixels.
- The proportion of black pixels separating upper and lower contours.

FIGURE 3.3: Examples of characters 'm' and 'e' extracted from IAM dataset

The average length of all the sequences in the training set is approximately 42 time-steps with a min of 8 and a max of 155. The data are normalized so that every feature has mean 0 and variance 1 on the training set.

3.5.2 Preliminary results

We first report preliminary results on CHMMs gained on one fold while we report later more significant results gained through 12 folds for CHMMs. In every fold, there are 200 sequences for training, 50 for validation and 50 for testing, for each of the 23 classes (i.e. lowercase), 3 classes have been removed because they were under represented.

In the following HMMs and CHMMs are left right models without skip, all models exploit full covariance matrices. μCHMMs and HMMs were trained up to convergence with a maximum of 150 EM iterations. The training of $\mu\Sigma$CHMM with covariance parameterization had an additional 150 GEM iterations with one gradient iteration every M step. In both cases, model selection is performed as the set of character models, at a given iteration, that performs best on the validation set. Note that the complexity of learning $\mu\Sigma$CHMM is slightly increased, it should be about twice the cost of learning μCHMM but we observed it was slightly less in practice.

Initialization of HMMs and of CHMMs is performed according to a linear alignment of training sequences on the left-right models: every training sequence is divided into a number of consecutive segments of equal length, one segment per state. Re-estimation formulas are then used with this linear alignment. In case we use Gaussian mixtures, means and covariance matrices of a mixture are initialized by Kmeans on the set of all observations aligned with the state.

First of all we report in figure 3.4 the performance of standard HMMs wrt the size of Gaussian mixtures. The accuracy in test increases up to a plateau while accuracy still increases on training set, showing the difficulty of learning more complex models.

Next we investigate the use of a contextual information. We explore few definitions of contextual variables $\boldsymbol{\theta}$ and we focused in our experiments on information that may be computed directly from the observation sequence while other type of information (gender, age, etc) could be used as well (but are not always available). We first investigate the mean $\sum_t \boldsymbol{x}_t$ (d dimensional vector noted 'μ') and the variance of features vector (d

dimensional vector noted 'σ^2') computed on the full sequence.[1] We have actually observed that such an information may be used with great benefit in CHMMs. Although it looks like it could be taken into account in a standard preprocessing (normalization) step such a strategy do not work (see below table 3.1). We believe the reason is that in CHMM, one learns simultaneously (optimally) the HMM parameters and the way it uses such a global information.

To explore the possibility to extend the approach to sequence labeling (when one wants to predict a sequence of labels for a sequence of observations, like e.g. in speech recognition), we investigated the use of *dynamic and local* contextual variables. A number in parenthesis suffixing a context variable name, like '$\mu(31)$', means that $\boldsymbol{\theta}$ is a function of time and is averaged over a window of 31 frames centered at current time (e.g. $\boldsymbol{\theta}_t = mean(\boldsymbol{x_{t-15}}, ..., \boldsymbol{x_{t+15}})$). The idea is that in a sequence labeling task, such a dynamic and local mean may convey the same kind of information that the global information conveys for sequence classification.

We report in figure 3.5 results using contextual parameterization of the means only (μCHMM), and of means and covariance matrices ($\mu\Sigma$CHMM), with static and dynamic $\boldsymbol{\theta}$'s. In this figure, CHMMs are single Gaussian models (with 8 states) and $\boldsymbol{\theta}$ is defined as the vector of variances of frame features, computed on the whole sequence or locally on a sliding window of increasing size (abscissa). As may be seen, all CHMMs with either static $\boldsymbol{\theta}$ or dynamic $\boldsymbol{\theta}_t$ improve over standard single Gaussian HMMs (60.5% accuracy). Although static $\boldsymbol{\theta}$ work well, finding a good set up of dynamic $\boldsymbol{\theta}$ (e.g. window size) is harder. Yet equivalent or slightly better results may be obtained with dynamic variables, meaning that one can expect the extension of this framework to signal labeling (where a static $\boldsymbol{\theta}$ is less relevant) to work well. Finally, note that the covariance parameterization gives an additional improvement over mean only μCHMMs.

Figure 3.6 reports similar results but this time $\boldsymbol{\theta} = \boldsymbol{\mu}$ or $\boldsymbol{\theta} = \boldsymbol{\mu}(t)$. We can observe similar trends. As before, CHMMs outperform single Gaussian HMM but, more interestingly, single Gaussian CHMMs outperform the best HMM models whatever the size of Gaussian mixtures.

[1] More intuitively relevant contextual information could be used. For instance one can easily imagine that the mean of the absolute derivative of feature vectors should bring some useful information about the speed of a gesture, the speech rate... it should be high if a gesture is performed quickly meaning it will probably have less amplitude, or if a word is spoken quickly meaning few phones will be shortened... The reason why we report results gained with the mean and the variance of features in this section is that these contextual variables have been shown to consistently improve over HMM performances in our experiments.

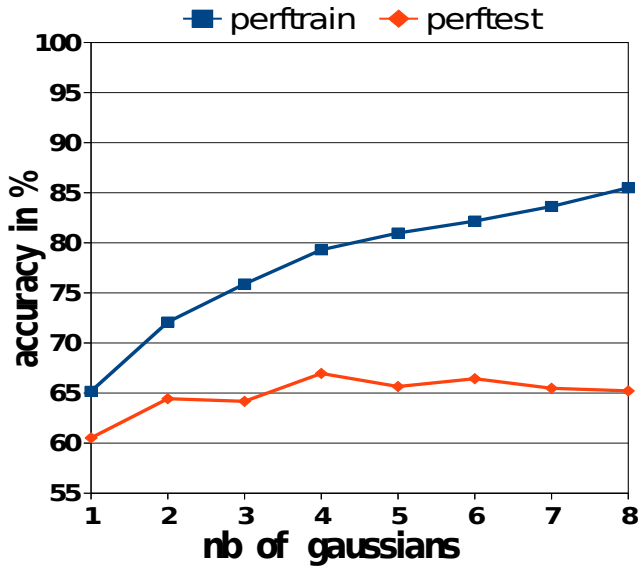


FIGURE 3.4: Performance of 8 states Gaussian HMMs as a function of Gaussian mixture size.

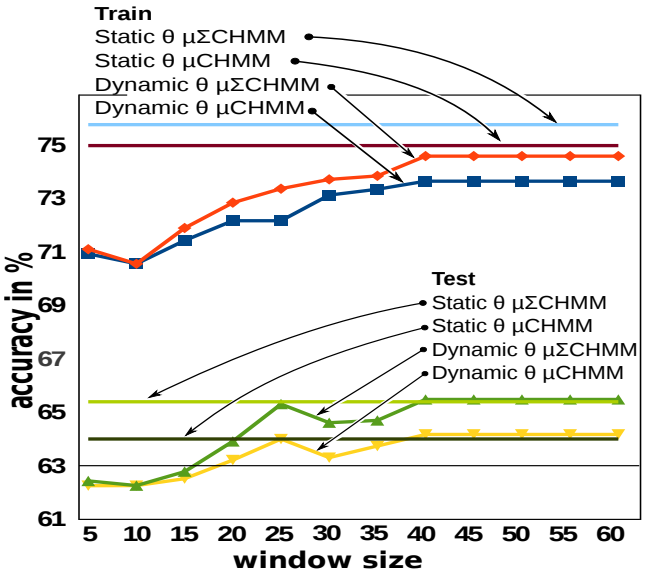


FIGURE 3.5: Performance of 8 states CHMM with $\boldsymbol{\theta} = \sigma^2$ or $\boldsymbol{\theta}_t = \sigma^2(t)$ as a function of the window's length used to compute $\boldsymbol{\theta}_t$.

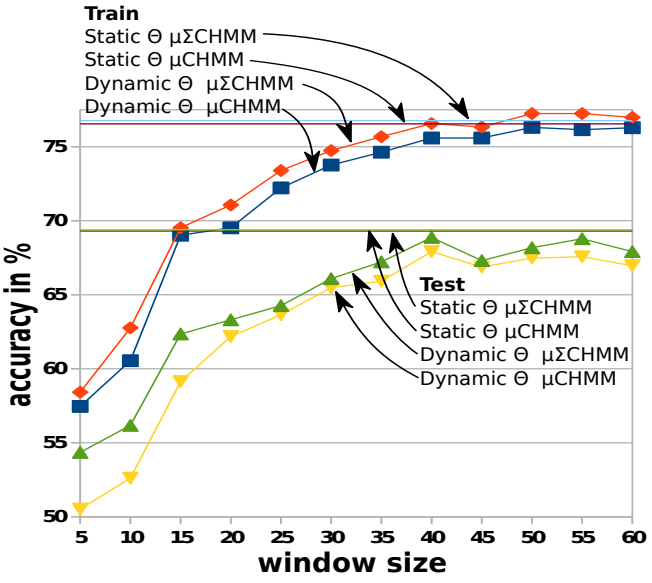


FIGURE 3.6: Performance of 8 states CHMM with $\boldsymbol{\theta} = \mu$ or $\boldsymbol{\theta}_t = \mu(t)$ as a function of the window's length used to compute $\boldsymbol{\theta}_t$.

3.5.3 Extended results

We now report results that have been obtained through 12 folds. We built a dataset of 12 folds with the same training/validation/test proportions (200/50/50). Note that the dataset now contains 20 classes (3 classes have been removed to maintain data balance between them) so the results are not directly comparable with above results.

This protocol allows us deriving more reliable averaged results, computing their standard deviation and investigating the performance improvement of one modeling over another through statistical testing. We used two popular such tests, a 2 tailed paired sample t-test as well as a Wilcoxon signed rank test (with $pvalue < 5\%$, i.e. testing significance at a 95% confidence interval). Note that all experiments follow the same procedure as above with a maximum of 150 EM and 150 GEM iterations.

First of all, Table 3.1 reports extensive results on standard HMMs for various settings (number of states, size of Gaussian mixtures). The columns with *sphering* in parentheses stand for HMMs working on sphered data: it means that the data have been preprocessed to have a global mean of zero and a global covariance equals to identity on the training set.

Next, we investigate the usefulness of various contextual information $\boldsymbol{\theta}$. We made trials with the instantaneous derivative of the sequence averaged on the whole sequence (noted 'Δ'), and the instantaneous acceleration, also averaged on the whole sequence (noted 'Δ^2'). In addition to these static context vectors, we investigated dynamic ones, where these quantities are computed on a sliding window rather than on the whole sequence.

We tried a few combination of these contextual variables by concatenating these contextual features in a single $\boldsymbol{\theta}$ vector. We also compared our approach to standard HMMs that are fed with the same information (we call these AugHMM), by simply adding the contextual variables as new features in the observation vectors. This is maybe not the ideal way for a HMM to make use of such features but this modeling allows in some way to capture correlations between feature vectors and contextual variables (e.g. as is used in [40]), it must be seen as a naive baseline here.

TABLE 3.1: Accuracy of HMM baselines as a function of the number of states per class model and as a function of the number of Gaussians per state. Results are averaged over 12 folds (Standard deviations are given in parentheses).

nb states	nb gauss	Train	Train (sphering)	Test	Test (sphering)
3	1	57.8 (1)	57.6 (0.8)	55.6 (2)	55.5 (2.2)
3	2	59.7 (1.2)	60.1 (1.3)	55.9 (1.6)	56.2 (1.5)
3	3	64.1 (1.3)	64.3 (1.3)	59.6 (1.8)	59.9 (1.7)
5	1	61.8 (0.8)	61.9 (0.7)	58.6 (2)	58.7 (1.6)
5	2	67.7 (1.7)	67.9 (1.2)	62.5 (2.8)	62.1 (2)
5	3	71.9 (1.1)	72.3 (1.2)	64.5 (1.2)	64.5 (0.9)
8	1	67.7 (0,8)	67.8 (1.1)	63.6 (1,7)	63.2 (1.8)
8	2	74.1 (0.8)	74.4 (0.7)	67.3 (1.6)	67.2 (1.7)
8	3	78.1 (0.7)	78.2 (0.8)	68.7 (1.3)	68.6 (1.2)

One sees that μCHMMs systematically outperform corresponding HMMs and that $\mu\Sigma$CHMMs systematically outperform corresponding μCHMMs (i.e. same line in Table 3.2). We ran the statistical tests and found that the improvement of all μCHMMs over HMMs are statistically significant, and also that whatever the setting (i.e. whatever the line in Table 3.2) the improvement of $\mu\Sigma$CHMMs over corresponding μCHMMs are all statistically significant under the two statistical tests we used, a 2 tailed paired sample t-test and a Wilcoxon signed rank test ($pvalue < 5\%$). Note finally that improvement of $\mu\Sigma$CHMMs over corresponding μCHMMs are consistent across different topologies and for various types of contextual variables.

One can finally observe that adding contextual variables to observation vector of standard HMMs can help a little (compare results of AugHMMs in table 3.2 to results of HMMs in table 3.1) but it may also degrade performance (e.g. for 8 state 1 Gaussian HMMs). On the other side, the gap is huge between AugHMMs and CHMMs. CHMMs manage to exploit the additional information in a more efficient way. Note also that decorrelating the data (sphering in table 3.1) does not raise significantly the performance in the context of our full covariance HMMs. To this respect, using contextual

TABLE 3.2: Accuracy of Contextual HMM baselines for various settings (number of states per class model, number of Gaussian distributions in a Gaussian mixture) and for various definition of the contextual information $\boldsymbol{\theta}_t$ computed on several windows sizes (found by trials and errors). All these results are averaged on 12 folds as previous table (Standard deviations are given in parentheses). Note that for every setting (every line), the improvement of $\mu\Sigma$CHMMs over μCHMMs and over HMMs, as well as the improvement of μCHMMs over HMMs are statistically significant under a 2 tailed paired sample t-test as well as a Wilcoxon signed rank test ($pvalue < 5\%$).

nb states	nb gauss	Context variable $\boldsymbol{\theta}_t$	Train augHMM	Train μCHMM	Train $\mu\Sigma$CHMM	Test augHMM	Test μCHMM	Test $\mu\Sigma$CHMM
3	1	$\mu(41)$	60.5 (0.6)	66.1 (0.7)	**67.2 (0.6)**	56.3 (1.4)	61.5 (1)	**62.2 (1.3)**
3	1	$\mu(61)$ $\sigma^2(55)$	64.5 (1)	76.9 (0.7)	**79 (1.1)**	55.6 (1.8)	67 (1.3)	**67.6 (1.7)**
3	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	64.6 (0.7)	73.6 (0.7)	**77.4 (1.6)**	55.8 (1.6)	65.1 (1.2)	**66.2 (1.5)**
3	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	64.5 (0.6)	74.8 (0.8)	**77.2 (1)**	55.3 (1.7)	64.6 (1)	**65.5 (1.2)**
3	2	$\mu(41)$	67.7 (1.1)	70.3 (0.9)	**73.7 (1.3)**	59.8 (1.3)	62.8 (1)	**65.5 (1.7)**
3	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	75 (0.9)	80.4 (1.1)	**83.2 (1.7)**	59.7 (1.5)	65.6 (1.2)	**67.1 (0.9)**
5	1	$\mu(41)$	66.1 (0.6)	72.1 (0.7)	**73.4 (1.1)**	60.1 (1.6)	65.5 (1.2)	**66.3 (1.2)**
5	1	$\mu(61)$ $\sigma^2(55)$	70.3 (0.7)	76.9 (0.7)	**79 (1.1)**	59.3 (1.3)	67 (1.3)	**67.6 (1.7)**
5	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	70.2 (0.7)	79.2 (0.5)	**81.2 (0.9)**	59.1 (1.2)	66.9 (1.1)	**68.1 (1.5)**
5	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	70.5 (0.9)	80.4 (0.8)	**81.4 (0.8)**	59.5 (1.1)	67 (0.8)	**67.4 (0.9)**
5	2	$\mu(41)$	75.7 (0.8)	79.7 (1.2)	**81.4 (1.2)**	64 (1.2)	69.5 (1.8)	**70.2 (1.1)**
5	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	80.7 (0.6)	88.1 (0.6)	**88.3 (0.7)**	62.6 (2)	69.4 (0.8)	**69.8 (0.9)**
8	1	$\mu(41)$	70.9 (0.7)	78 (0.9)	**79 (1)**	63.5 (1.3)	70.3 (1.1)	**70.9 (1.1)**
8	1	$\mu(61)$ $\sigma^2(55)$	74.2 (0.9)	82.6 (0.8)	**84.2 (0.7)**	61.7 (1.3)	70.8 (1.2)	**71.7 (1)**
8	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	73.7 (0.6)	84.6 (0.7)	**85.4 (0.7)**	61.2 (1.5)	71.4 (1.2)	**72.1 (0.9)**
8	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	74.3 (0.8)	85.6 (0.6)	**86.3 (0.8)**	61.6 (1.6)	70.7 (1.2)	**71.4 (1.3)**
8	2	$\mu(41)$	80.3 (1)	85.6 (0.7)	**86.5 (0.6)**	59.8 (1.3)	73.3 (1.5)	**74.2 (1.3)**
8	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	84.3 (0.7)	92.4 (1.1)	**92.5 (1.2)**	64.5 (2.2)	72.3 (1.2)	**72.6 (1.2)**

variables in CHMMs is a more accurate way to model the variability of the signal which can not always be removed through pre-processing or normalization steps.

3.6 Conclusion

In this chapter, we introduced the Contextual Hidden Markov Models. It is a framework allowing the parameterization of a HMM Gaussian distributions (means & covariances), and transition probabilities between hidden states with static or dynamic contextual variables. Compared to other approaches discussed in section 3.4, it is the most complete proposal belonging to the family of Parametric HMMs.

Experiments conducted on a handwriting classification task proved the effectiveness of parameterizing the means as well as the covariances of Gaussians with simple static or dynamic contextual variables that can be computed directly on the observations sequence. We measured for example the efficiency of using the mean or short term mean of the observations sequence as a contextual variable whereas augmenting the observation vector of simple HMMs with the same information was far less effective. Results revealed the important gains achieved by mean parameterization. On the other hand, the gains achieved by the addition of covariances parameterization appear less important, yet, they are significant and consistent across several model topologies. An application of CHMMs using transitions parameterization will be detailed in chapter 5.

Chapter 4

Contextual Hidden Conditional Random Fields

Contents

4.1 Introduction 67
4.2 Discriminative training of Hidden Markov Models 68
4.2.1 MMI 69
4.2.2 MCE 70
4.2.3 MWE/MPE 71
4.2.4 Discussion 72
4.3 Exploiting contextual information in Hidden Conditional Random Fields 73
4.3.1 HCRF as a generalization of HMM 73
4.3.2 Contextual HCRFs 75
4.3.3 Training Contextual HCRFs 77
4.3.4 Experiments 77
4.4 Conclusion 79

4.1 Introduction

HMMs are a famous class of probabilistic generative models that are well known for their efficient algorithms, their simplicity and their robustness for classifying and labeling sequences. As exposed in section 2.3.1.2, HMMs can be trained through Maximum Likelihood Estimation (MLE). Yet, this non discriminative training criterion fits well modeling applications where one wants to learn accurate models of production, e.g. for synthesis [40, 76], but it is not precisely tuned to classification or recognition tasks.

This is why, in the last years, researchers have proposed various methods for training HMMs discriminatively. However, in chapter 3 we have shown that Contextual HMMs seems more effective than HMMs at using contextual information. To this respect, our goal is here to design a similar and more efficient way to use contextual information into a discriminative model.

In a first step, we will review a bunch of methods that have been proposed to train HMM discriminatively. Then, we will show how another type of purely discriminative model, the HCRF, can easily simulate the decision function of HMMs. This finally opens the door to presenting the Contextual Hidden Conditional Random Field (CHCRF), the discriminative counter part of Contextual Hidden Markov Models. Experiments carried on a handwriting classification task demonstrate that CHCRF is an efficient way to train HCRF.

4.2 Discriminative training of Hidden Markov Models

In section 2.1.3.2 we have seen the difference between generative models and discriminative ones. The former model the joint distribution of observations and their label $p(\boldsymbol{x}, y)$, while the latter directly model $p(y \mid \boldsymbol{x})$, the conditional distribution of the label given the observation sequence.

Hidden Markov Models are easy to train generatively with Maximum Likelihood Estimation, but, this form of training only increases the probability of the correct labeling while ignoring what happens for incorrect labelings. Some attempts have been made however to estimate HMMs with discriminative criteria, more adapted to classification tasks. Discriminative training of HMMs aims at not only increasing the probability of correct labelings, but also at decreasing the probability of incorrect ones.

To build more accurate sequence recognition and labeling HMM systems, a first bunch of methods have been proposed to train HMMs in a discriminative way. The most famous approaches are Maximum Mutual Information ([1]), Minimum Classification Error ([44]) and Minimum Phone Error named after its initial application to speech recognition ([64]). More recently, few works have applied the large margin principle to HMM learning, most of these works have concerned speech recognition [72], [13], [56] and handwriting recognition [24, 80].

Before explaining the CHCRF, we will first present some of the most popular approaches for training HMMs discriminatively.

4.2.1 MMI

Maximum Mutual Information training (MMI) is based on an information theoretic point of view. Maximizing the Mutual Information between observations and their labels can be shown to reduce the uncertainty in the labelings $\boldsymbol{y}$ knowing the observations $\boldsymbol{x}$.

The mutual information between observation sequences $\boldsymbol{x}_{1\ldots T} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathcal{X}$ and their labeling $\boldsymbol{y}_{1\ldots T} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathcal{Y}$ is given by

$$\mathcal{L}_{MMI}(\Lambda) = \sum_{\mathcal{Y}} \int_{\mathcal{X}} p(\boldsymbol{x}_{1\ldots T}, \boldsymbol{y}_{1\ldots T}) \log \frac{p(\boldsymbol{x}_{1\ldots T}, \boldsymbol{y}_{1\ldots T})}{p(\boldsymbol{x}_{1\ldots T}) p(\boldsymbol{y}_{1\ldots T})} d\boldsymbol{x}_{1\ldots T} \tag{4.1}$$

$$= \sum_{\mathcal{Y}} \int_{\mathcal{X}} p(\boldsymbol{x}_{1\ldots T}, \boldsymbol{y}_{1\ldots T}) \log \frac{p(\boldsymbol{y}_{1\ldots T} \mid \boldsymbol{x}_{1\ldots T})}{p(\boldsymbol{y}_{1\ldots T})} d\boldsymbol{x}_{1\ldots T} \tag{4.2}$$

$$= H(Y) - H(Y \mid X) \tag{4.3}$$

where $H(Y) = -\sum_Y p(Y) \log p(Y)$ is the entropy of random variable Y (i.e. the labels with realizations $y_1, \ldots, y_T$) and $H(Y \mid X) = \sum_Y p(X, Y) \log p(Y \mid X)$ is the conditional entropy of the labels given the observation random variable (with realizations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$).

In practice $p(\boldsymbol{y}_{1\ldots T})$ (denominator of Eq 4.2) represents the probability of the label sequence and can be estimated with a n-gram language model on a large corpus. $p(\boldsymbol{y}_{1\ldots T}) = p(y_1)p(y_2 \mid y_1)p(y_3 \mid y_2, y_1) \ldots p(y_T \mid y_{T-1}, y_{T-2}, \ldots, y_{T-N})$.[1]

From 4.3 we can then see that optimizing $\mathcal{L}_{MMI}(\Lambda)$ is equivalent to minimize the conditional entropy $H(Y \mid X)$.

$$\mathcal{L}_{MMI}(\Lambda) \propto H(Y \mid X) = -\sum_{\mathcal{Y}} \int_{\mathcal{X}} p(\boldsymbol{x}_{1\ldots T}, \boldsymbol{y}_{1\ldots T}) \log p(\boldsymbol{y}_{1\ldots T} \mid \boldsymbol{x}_{1\ldots T}) d\boldsymbol{x}_{1\ldots T} \tag{4.4}$$

Finally, to make computation tractable, 4.4 is approximated using the empirical expectation of $p(Y \mid X)$ over the training set.

$$\mathcal{L}_{MMI}(\Lambda) = -\mathbb{E}_{\mathcal{D}_{train}} \left[\log p(\boldsymbol{y}_{1\ldots T} \mid \boldsymbol{x}_{1\ldots T})\right]$$

[1] The conditional probabilities can be computed by frequency counts $p(y_n \mid y_{n-N+1}, \ldots, y_{n-1}) = \frac{count(y_{n-N+1}, \ldots, y_{n-1}, y_n)}{count(y_{n-N+1}, \ldots, y_{n-1})}$ but smarter methods are generally used (Bellegarda [4]) as this would result in a lot of 0 or $\frac{0}{0}$ probabilities.

One can see that this loss is very similar to the one used in Conditional Random Fields (2.22). However in MMI training, the distribution $p(\boldsymbol{y}_{1...T} \mid \boldsymbol{x}_{1...T})$ is modeled by a Gaussian mixture HMM whereas CRFs do not impose such a constraint.

4.2.2 MCE

Minimum Classification Error (MCE) training proposes to minimize the expected recognition rate. It is more directly related to the performance of the classifier as we can measure it in sequence labeling problems.

$$L_{MCE(\Lambda)} = \sum_{y \in \mathcal{Y}} \int_{\boldsymbol{x} \in \mathcal{X}_y} l_y(\boldsymbol{x}_{1...T}; \Lambda) p(\boldsymbol{x}_{1...T}) d\boldsymbol{x}_{1...T} \tag{4.5}$$

where l_y is a smooth functional form which allows computing the derivatives of the loss 4.5 with respect to the parameters Λ.

$$l_i(\boldsymbol{x}_{1...T}; \Lambda) = \frac{1}{1 + exp(-\gamma d_y(\boldsymbol{x}_{1...T;\Lambda}) + \theta)}$$

with $\gamma \geq 1$, θ typically set to 0 and d_y represents the misclassification measure.

For isolated classification, i.e. one label y for each observation sequence $\boldsymbol{x}$ in the training set, the misclassification measure d_y is given by:

$$d_y(\boldsymbol{x}_{1...T}; \Lambda) = -g_y(\boldsymbol{x}_{1...T}; \Lambda) + \log \left[\frac{1}{|Y| - 1} \sum_{y' \neq y} exp(g_{y'}(\boldsymbol{x}_{1...T;\Lambda}))^{\eta} \right]^{\frac{1}{\eta}}$$

where $|Y|$ is the number of labels and g_y is the classifier score given to correct label y for the observation sequence $\boldsymbol{x}_{1...T}$. For a HMM modeling class y, this score can be computed as

$$g_y(\boldsymbol{x}_{1...T}; \Lambda) = \max_{h_1,...,h_T} p(\boldsymbol{x}_{1...T}, h_{1...T}, \Lambda)$$

As a result, when the classifier answers the correct label, $d_y > 0$ otherwise $d_y < 0$ and decreases proportionally to the sum of scores attributed to all incorrect labels y'. Also when η approaches ∞ the term in brackets becomes $\max_{y' \neq y} exp(g_{y'}(\boldsymbol{x}_{1...T;\Lambda}))$. Hence by varying η, d_y can take whether all competitive labels into considerations or only the most offending ones (most likely).

However, it is often the case that the label are not available at the frame level. In this case, $\boldsymbol{y}$ is a sequence of labels and $d_{\boldsymbol{y}}$ penalizes the N most incorrect labelings $\bar{\boldsymbol{y}}^{\boldsymbol{1\cdots N}}$

$$d_{\boldsymbol{y}}(\boldsymbol{x}_{1\ldots T};\Lambda) = -g_{\boldsymbol{y}}(\boldsymbol{x}_{1\ldots T};\Lambda) + \log\left[\frac{1}{N}\sum_{k=1}^{N} exp(g(\boldsymbol{x}_{1\ldots T,\bar{\boldsymbol{y}}^{\boldsymbol{k}};\Lambda}))^{\eta}\right]^{\frac{1}{\eta}} \tag{4.6}$$

In practice, the score $g_{\boldsymbol{y}}$ (Eq 4.6) is here assigned by a HMM model representing the correct labeling. In speech or handwriting recognition, this is called a "sentence HMM" which is created from the left to right concatenation of HMMs of all symbols in the sequence of labels $\boldsymbol{y}$.

Minimizing 4.5 with respect to Λ can be seen as increasing the likelihood of the correct sequence of labels $\boldsymbol{y}$ while decreasing the likelihood of N incorrect ones $\bar{\boldsymbol{y}}^{\boldsymbol{1\cdots k}}$. Actually, summing over all labelings is intractable as there are $|Y|^T$ possibles ones. Instead, eq 4.6 sums over a limited number of labelings (N-best) which represents the biggest probability mass. These N-best incorrect label sequences $\bar{\boldsymbol{y}}^{\boldsymbol{1\cdots N}}$, called contrastive examples, can be found using the N-Best algorithm ([83]) in the composite model g. [2]

4.2.3 MWE/MPE

MCE minimizes the classification errors at the sentence level. However, in speech and handwriting, error is measured more precisely at the substring level. Minimum Word Error (MWE) and Minimum Phone Error (MPE) criteria are respectively defined to minimize the number of word and phone errors.

$$\mathcal{L}(\Lambda) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{train}} \sum_{\boldsymbol{y'}} p(\boldsymbol{y'} \mid \boldsymbol{x};\Lambda) Acc(\boldsymbol{y'},\boldsymbol{y}) \tag{4.7}$$

In MWE the true labels $\boldsymbol{y}$ are words, while in MPE $\boldsymbol{y}$ represent a sequence of phones. Except that, both approaches share the same formulation.

Acc measures the number of words (MWE) or phones (MPE) which are correct in the label sequence $\boldsymbol{y'}$ with respect to the true segmentation $\boldsymbol{y}$ of the observations sequence $\boldsymbol{x}$. In sequence labeling problems, $Acc(\boldsymbol{y},\boldsymbol{y'})$ is defined as the edit distance accuracy. As we explained in 2.2.2, it is the minimum of character edits (insertions, deletions or substitutions) to match $\boldsymbol{y'}$ into $\boldsymbol{y}$.

Loss 4.7 can then be seen as penalizing the probability of a particular labeling $\boldsymbol{y'}$ proportionally to its edit distance with the true labeling $\boldsymbol{y}$. As with MCE, the summation

[2] the composite model is a HMM composed of the HMMs of each word (ie class) in the vocabulary, with transitions allowed from word to word or constrained by a grammar.

over all hypothesis $\boldsymbol{y'}$ (eq 4.7) could be approximated using a N-Best sequence of labels in the composite model.

As it is expressed loss 4.7 is however computationally expensive. For each hypothesis $\boldsymbol{y'}$, it requires computing the edit distance accuracy $Acc(\boldsymbol{y'}, \boldsymbol{y})$ using a costly dynamic programming algorithm. Nonetheless, one can use the following approximations:

Accuracy of sentence $\boldsymbol{y'}$ with respect to the true sentence $\boldsymbol{y}$ is defined as the sum over all its y' words accuracies.

$$Acc(\boldsymbol{y}, \boldsymbol{y'}) = \sum_{y' \in \boldsymbol{y'}} Acc(\boldsymbol{y}, y')$$

with

$$Acc(\boldsymbol{y}, y') = \begin{cases} -1 + 2e & \text{if y = y'} \\ -1 + e & \text{otherwise} \end{cases}$$

and e is the proportion that the word y' overlap with the word y in the correct transcription $\boldsymbol{y}$.

For the interested reader, we refer to the work of [43], [61] and [88] for additional information on MMI/MCE/MWE/MPE and related discriminative training methods for HMMs.

4.2.4 Discussion

In the last five years, researchers have also investigated the use of purely discriminative models for sequence labeling. Conditional Random fields and their extension for dealing with hidden states, namely Hidden CRFs [46, 65] (see 2.4.1 and 2.4.2).

A first reason to consider this approach is the fact that HCRF are not constrained to use Gaussian probability distributions as other related discriminatively trained HMMs. As we pointed in 2.4.1 it may be an unecessary burden for the task of classification.

Secondly, all above discriminative approaches, discriminatively trained HMMs and HCRFs, have been shown to significantly outperform non discriminatively trained HMMs [13, 28, 34, 54, 56, 71, 74, 80]. Also, there seems to be a slight advantage to MPE and MCE among discriminative learning criterion for HMMs. Indeed, [72, 74] find that MCE and MPE slightly outperform MMI (or Conditional Maximum Likelihood, a close variant) while [13, 56] report similar results for MCE, MPE and MMI. Next, the large margin approach is most often reported as outperforming MMI and MPE [13, 24, 71, 80]. HCRFs are also reported as outperforming other discriminative criterion for learning HMMs

(MMI, MPE, MCE) [28, 34, 54, 74, 80]. Finally HCRF and large margin learning of HMMs seem to yield similar results [80]. Both methods are state of the art methods today.

4.3 Exploiting contextual information in Hidden Conditional Random Fields

Motivated by these results, we present here a first approach for exploiting contextual information in a HCRF system. We first recall basics of HCRF modeling, then we explain how HMMs may be viewed as special cases of HCRFs, which lead to an efficient initializing scheme for learning HCRFs proposed in [34]. Finally we build over these works to propose an efficient way for parameterizing and for learning Contextual HCRFs.

4.3.1 HCRF as a generalization of HMM

First HCRF has a non convex loss, secondly because it computes scores instead of Gaussian probabilities, this model has a high degree of freedom. Consequently HCRF is particularly subject to overfitting. Yet, one efficient way to learn HCRFs has been proposed in recent years [34]. It consists in learning first a HMM system, then to initialize the HCRF parameters so that it reproduces the same classification as the HMMs.

This strategy also makes sense because HCRF is trained discriminatively which seems more adapted to pure classification tasks (see 2.4.1). However maximum likelihood training of HMMs is easily parallelizable because class models are independent of each others which is not the case in HCRFs. It gives HMM a clear advantage in training speed. Consequently, casting a HMM into a HCRF and retrain it for a few iterations can give a little boost in performance for a limited cost.

Now we explain how this initialization can be done. The key point is that the joint log likelihood of an input sequence and of a sequence of states may be written as a dot product between a particular parameter vector and a joint feature map depending on the class, the sequence of hidden states and the input sequence. Indeed, in HMMs for any state sequence $\boldsymbol{h}$ we have :

$$\begin{aligned}\log p(\mathbf{x}, y, \boldsymbol{h}; \Lambda) = &\log(\pi_{h_1}) + \log(p(\boldsymbol{x}_1|h_1)) \\ &+ \sum_{t=2}^{T} (\log p(h_t|h_{t-1}) + \log p(\boldsymbol{x}_t|h_t, \mu_{h_t}, \Sigma_{h_t})\end{aligned}$$

with :

$$\begin{aligned}\log p(\boldsymbol{x}_t|h_t;\Lambda) &= -\frac{1}{2}\Big(\boldsymbol{x}_t^T\Sigma_{h_t}^{-1}\boldsymbol{x}_t - \boldsymbol{x}_t^T\Sigma_{h_t}^{-1}\boldsymbol{\mu}_{h_t} - \boldsymbol{\mu}_{h_t}^T\Sigma_{h_t}^{-1}\boldsymbol{x}_t \\ &\qquad + \boldsymbol{\mu}_{h_t}^T\Sigma_{h_t}^{-1}\boldsymbol{\mu}_{h_t} + \log((2\pi)^d|\Sigma_{h_t}|)\Big) \\ &= \langle\boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t)\rangle\end{aligned}$$

and :

$$\log p(h_t|h_{t-1};\Lambda) = \langle\boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1})\rangle \tag{4.8}$$

Hence, we get:

$$\log p(\boldsymbol{x}, y, \boldsymbol{h};\Lambda) = \sum_t \langle\boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1})\rangle + \langle\boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t)\rangle$$

This result can be verified with the definitions below :

$$\begin{aligned}&\boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1}) = \left(\delta_{h_t=1\wedge h_{t-1}=1}, \ldots, \delta_{h_t=S\wedge h_{t-1}=S}\right)^T \\ &\boldsymbol{\lambda}^{trans} = (\log a_{1,1}, \ldots, \log a_{S,S})^T \\ &\boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t) = \left(\boldsymbol{\phi}_1^{loc}(\boldsymbol{x}_t, y, h_t)^T, \boldsymbol{\phi}_2^{loc}(\boldsymbol{x}_t, y, h_t)^T, \ldots, \boldsymbol{\phi}_S^{loc}(\boldsymbol{x}_t, y, h_t)^T\right)^T \\ &\boldsymbol{\lambda}^{loc} = \left((\boldsymbol{\lambda}_1^{loc})^T, (\boldsymbol{\lambda}_2^{loc})^T, \ldots, (\boldsymbol{\lambda}_S^{loc})^T\right)^T\end{aligned}$$

where

$$\delta_{h_t=i\wedge h_{t-1}=j} = \begin{cases}1, & \text{if } h_t = i \text{ and } h_{t-1} = j \\ 0 & \text{otherwise}\end{cases}$$

In the above equations:

$$\boldsymbol{\phi}_i^{loc}(\boldsymbol{x}, y, h_t) = \left(1, (x_u)_u, (x_u x_v)_{u,v}\right)^T \times \delta_{h_t=i} \;\forall i \in 1..S$$

$$\boldsymbol{\lambda}_i^{loc} = \begin{pmatrix} -\frac{1}{2}\left[\log((2\pi)^d|\Sigma_i| + \bar{\mu}^i\left(\Sigma^i\right)^{-1}\bar{\mu}^i\right] \\ \left(\left(\Sigma^i\right)^{-1}\bar{\mu}^i\right) \\ \left(-\frac{1}{2}\left(\Sigma^i\right)_{u,v}^{-1}\right)_{u,v}\end{pmatrix} \quad \forall i \in 1..S \tag{4.9}$$

with

$$\delta_{h_t=i} = \begin{cases} 1, & \text{if } h_t = i \\ 0 & \text{otherwise} \end{cases}$$

and

$$(x_u)_u = (x_1, ..., x_d)$$
$$(x_u x_v)_{u,v} = \left(x_1^2, x_1 x_2, \ldots, x_1 x_d, x_2 x_1, \ldots, x_d x_1, \ldots, x_d^2\right)$$

The above complex definition for local cliques can be understood as a feature map $\boldsymbol{\phi}^{loc}$ with degree two Cartesian products between observation features while parameter vector $\boldsymbol{\lambda}^{loc}$ only involves means and covariances terms of HMMs.

More formally:

- $(x_u)_u$ is a vector composed of all dimensions of $\boldsymbol{x}$. On the same principle $(x_u x_v)_{u,v}$ is a vector composed of the Cartesian product of the dimensions of $\boldsymbol{x}$. Time is removed from the indices for clarity.
- $a_{i,j}$ stands for the probability of the HMM to make a transition from state i to j. (states indices $\in 1..S$).
- $\Sigma^{-1}_{u,v}$ stands for the element line u and column v of the matrix Σ^{-1}. Also, $((\Sigma^{-1}_{u,v}))_{u,v} = \left(\Sigma^{-1}_{1,1}, \Sigma^{-1}_{1,2}, \ldots, \Sigma^{-1}_{1,S}, \Sigma^{-1}_{2,1}, \ldots, \Sigma^{-1}_{S,S}\right)^T$ is a vector of the elements of Σ^{-1} unfolded in column first order.

The above results yield an efficient learning procedure for learning HCRFs for sequence classification. First one learns a HMM system, with one (left-right) HMM per class, using either a maximum likelihood criterion or a discriminative criterion such as Maximum Mutual Information as in [74] [34]. Then one initializes a HCRF system with the same topology (a left right model per class) with the above formulas. This HCRF system outputs exactly the same decision as the HMM system. Finally one uses the standard discriminative conditional likelihood criterion of HCRFs for retraining the HCRF system. At the end the initialization by the HMM system allows starting the HCRF optimization process in an interesting area so as to reach a relevant local minima of the non convex HCRF optimization criterion.

4.3.2 Contextual HCRFs

In order to define contextual HCRFs (CHCRFs) one has to define new feature maps $\boldsymbol{\phi}^{trans}$ and $\boldsymbol{\phi}^{loc}$ and then to learn a linear HCRF on such representations. A simple

choice would be to concatenate $\boldsymbol{x}$ and the contextual variables $\boldsymbol{\theta}$ in $\boldsymbol{\phi}$ so that the HCRF would compute a linear function of these inputs. This is what we call Augmented HCRF (AugHCRF). A more interesting choice consists in defining a feature map $\boldsymbol{\phi}$ involving degree two Cartesian products of $\boldsymbol{x}_t$ and $\boldsymbol{\theta}_t$ components to make it possible for the HCRF to simulate the CHMM decision boundary. This strategy has the key advantage of allowing an initialization scheme by a learned Contextual HMM, overcoming the main drawback of HCRFs (and consequently of CHCRFs) which is the sensitivity of their optimization to initialization. Although MLE learning criterion for HMMs is non convex and HMM learning is sensitive to initialization as well, HCRFs have more capacity than HMMs (previous section actually shows that HCRFs include HMMs) and may quickly overfit or fall into a local minima. Actually in our experiments, randomly initialized HCRF hardly reach 40% accuracy on the IAM dataset using degree two observation features (Cartesian product of observation vector components). On the other hand HMMs are simpler models hence less sensitive to over-fitting and, more importantly, one has prior knowledge on how to initialize HMMs to optimize likelihood, e.g. left-right HMMs are usually initialized though linear alignment of sequences. At the end, getting an initial solution from MLE trained HMMs is a natural and practical idea. By the way discriminative training of HMMs (through MMI, MCE, LargeMargin) usually starts from MLE trained HMMs.

This led us to the idea of building a feature map $\boldsymbol{\phi}$ depending on $\mathbf{x}$ and $\boldsymbol{\theta}$ such that we could initialize a Contextual HCRF to perform exactly as a Contextual HMM. From this starting point a standard optimization process for HCRF leads to an, eventually more accurate HCRF system exploiting contextual variables.

To implement Contextual HCRF we therefore define the feature functions $\boldsymbol{\phi}^{loc}$ and $\boldsymbol{\phi}^{trans}$ and we initialize parameter vectors $\boldsymbol{\lambda}^{loc}$ and $\boldsymbol{\lambda}^{trans}$ from a Contextual HMM as follows, we detail the case of μCHCRFs derived from μCHMMs (it is straightforward to follow the same principle and write an initialization scheme starting from a fully parameterized CHMM):

$$
\begin{aligned}
&\boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1}) = \left(\delta_{h_t=1 \wedge h_{t-1}=1}, \ldots, \delta_{h_t=S \wedge h_{t-1}=S}\right)^T \\
&\boldsymbol{\lambda}^{trans} = (\log a_{1,1}, \ldots, \log a_{S,S})^T \\
&\boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t, \boldsymbol{\theta}_t) = \left(\boldsymbol{\phi}_1^{loc}(\boldsymbol{x}_t, y, h_t, \boldsymbol{\theta}_t)^T, \boldsymbol{\phi}_2^{loc}(\boldsymbol{x}_t, y, h_t, \boldsymbol{\theta}_t)^T, \ldots, \boldsymbol{\phi}_S^{loc}(\boldsymbol{x}_t, y, h_t, \boldsymbol{\theta}_t)^T\right)^T \\
&\boldsymbol{\lambda}^{loc} = \left((\boldsymbol{\lambda}_1^{loc})^T, (\boldsymbol{\lambda}_2^{loc})^T, \ldots, (\boldsymbol{\lambda}_S^{loc})^T\right)^T
\end{aligned}
\tag{4.10}
$$

with:

$$\boldsymbol{\phi}_i^{loc}(\boldsymbol{x}, y, h_t, \boldsymbol{\theta}) = \left(1, (x_u)_u, (x_u x_v)_{u,v}, (\theta_u)_u, (\theta_u \theta_v)_{u,v}, (\theta_u x_v)_{u,v}\right)^T \times \delta_{h_t=i} \; \forall i \in 1..S$$

$$\boldsymbol{\lambda}_i^{loc} = \begin{pmatrix} -\frac{1}{2}\left[\log((2\pi)^d|\Sigma_i| + \bar{\mu}^i \left(\Sigma^i\right)^{-1} \bar{\mu}^i\right] \\ \left(\left(\Sigma^i\right)^{-1} \bar{\mu}^i\right) \\ \left(-\frac{1}{2}\left(\Sigma^i\right)^{-1}_{u,v}\right)_{u,v} \\ \left(-\bar{\mu}_i^T \Sigma^{i^{-1}} V^i\right) \\ \left(\left(\frac{-1}{2}\left((V^i)^T \Sigma^{i^{-1}} V^i\right)\right)_{u,v}\right)_{u,v} \\ \left((V^i)^T \left(\Sigma^{i^{-1}}\right)_{u,v}\right)_{u,v} \end{pmatrix} \; \forall i \in 1..S$$

where V^i in state i is defined (Eq 3.2) as the coefficients matrix of Gaussian mean parameterization in CHMMs.

4.3.3 Training Contextual HCRFs

As is usually done when learning discriminative models such as HCRF, we used a L2 regularization term to penalize complexity. The actual optimization criterion is then $\sum_k \log p(y^k|\boldsymbol{x}^k; \boldsymbol{\theta}^k, \Lambda) - C \|\Lambda\|^2$ where C denotes the weight of the regularization term. The optimal value C is selected on the validation dataset. Optimization is performed through batch gradient learning.

4.3.4 Experiments

We investigate now the benefit one can get from exploiting contextual variables in HCRFs. Basically we are interested in comparing contextual HCRFs to standard HCRFs which are state of the art models for sequence labeling and sequence classification tasks [28, 34, 54, 74, 80].

We perform isolated characters classification experiments on a 6 folds version of IAM handwriting dataset (3.5.1). We report little less exhaustive results here since HCRFs are longer to train than HMMs. Our results are averaged over 6 folds only, for timing reasons, hence they cannot strictly be compared to those of previous sections. For the same reasons, we focused on few definitions of $\boldsymbol{\theta}$ and investigated 8 states models only, that were best performer models among the CHMMs we tested.

We compare HCRFs, AugHCRFs and μCHCRFs. Both AugHCRFs and μCHCRFs use the contextual information $\boldsymbol{\theta}$ in different ways while HMMs and HCRFs are not. The contextual variable $\boldsymbol{\theta}$ is here defined as the static or dynamic mean of the sequence alike in section 3.5.2. Initialization of all HCRF systems (standard, Augmented, Contextual) is performed from the best corresponding HMM system (standard, Augmented or Contextual) on validation data. We recall that Augmented HMM (defined in 3.5.3) is simply a standard HMM with $\boldsymbol{\theta}_t$ appended to the observation vector $\boldsymbol{x}_t$. In any case HCRF training was run for a maximum of 50 iterations of batch gradient descent learning. Selection of the best HCRF system was then done on the validation data again. The optimal value C of the regularization term is set on the validation dataset using a coarse grid search. Note that we provide HMMs results for information.

TABLE 4.1: Classification accuracy of discriminative models : HCRFs, augHCRFs (with context appended in the observation vector) and contextual HCRFs. Results are averaged on a 6 folds version of IAM dataset (3.5.1). All models have 8 states per class and are initialized from corresponding single Gaussian HMMs and CHMMs. All improvements of CHCRFs over AugHCRFs (using the same context) and over HCRF are statistically significant.

Model	Train	Test
HMM	65.6 (0.2)	60.6 (1.5)
HCRF	**67.8 (0.2)**	**63.7 (1.8)**
AugHCRF static $\boldsymbol{\theta} = \mu$	70.9 (0.6)	62.6 (1.1)
μCHCRF static $\boldsymbol{\theta} = \mu$	**78.1 (0.8)**	**70.7 (1.6)**
AugHCRF dynamic $\boldsymbol{\theta} = \mu(41)$	68.1 (0.7)	58.9 (2.2)
μCHCRF dynamic $\boldsymbol{\theta} = \mu(41)$	**77 (0.8)**	**68.4 (1.9)**
AugHCRF dynamic $\boldsymbol{\theta} = \mu(61)$	66.6 (1.2)	57.8 (2)
μCHCRF dynamic $\boldsymbol{\theta} = \mu(61)$	**77 (0.7)**	**68.2 (1.6)**

We can draw a few conclusion from Table 4.1. Of course, HCRFs strongly outperform non discriminatively trained HMMs which is quite normal. Next, Augmented HCRFs fail to exploit the additional contextual information and perform even lower than HMMs. This is probably partially due to over-fitting since augmented Augmented HMMs (whose number of parameters is approximately that of AugHCRFs) have much more parameters than HMMs. Finally, CHCRFs albeit having also significantly more parameters than HMMs allow drastic reduction of error rates. Using CHCRF with static (μ) or dynamic ($\mu(41)$) contextual variables, we get respectively 19.2% and 12.9% relative reduction of the test error compared to the best standard HCRFs. According to our statistical tests (two tailed t-test and Wilcoxon sign rank test) the improvements of CHCRFs over AugHCRFs (using the same context) and over HCRF are significant ($pvalue < 1\%$).

4.4 Conclusion

There are not much specific works on using contextual information in hidden CRFs for enhancing robustness to variability although the implementation is rather simple. Most works, being based on the initialization of a HCRF from a HMM indirectly rely on standard strategies used for capturing variability in HMMs, i.e. increasing the number of states or increasing the Gaussian mixture size. An attempt has been made to extend such methods in [22] for eye movements modeling, where a number of alternative weight vectors are used within each state to account for various styles. Our work is rather a way to start from meaningful estimates of the weight vectors while keeping their degree of freedom during the course of learning. Actually our modeling is rather simple, our contribution is then more an efficient way to learn discriminative models that exploit contextual information.

Chapter 5

Exploiting Contextual Markov Models for synthesis

Contents

5.1 Motivation 81
5.2 Using HMMs for synthesis 82
5.2.1 Improved synthesis using non stationary HMMs 82
5.2.2 Synthesis with constraints 83
5.3 Speech to motion synthesis, an application 87
5.3.1 Related work 88
5.4 Speech to motion synthesis using Contextual Markovian models 89
5.4.1 Parameterizations 89
5.4.2 Training 89
5.4.3 Synthesis 90
5.4.4 Experiments 90
5.5 Conclusion 93

5.1 Motivation

The framework of contextual Hidden Markov Models being generative, it is enclined to data synthesis which is particularly useful in certain domains like text to speech, melody generation, character animation etc .... We present here an approach exploiting CHMM modeling in conversational agents which is the result of a collaboration with Yu Ding et al. from Greta team at Telecom-Paristech university. Before moving on to this specific

application using CHMM as a synthesis system, we will first review major works on synthesizing smooth sequences from a HMM.

5.2 Using HMMs for synthesis

Basically we are interested in the techniques that allow generating an output information stream from an input information stream using HMMs. Synthesizing a realistic sequence of observations (called a trajectory hereafter) from a HMM is a key issue. Of course, synthesizing the most likely observation sequence given a particular state sequence yields a very unlikely piecewise constant trajectory.

5.2.1 Improved synthesis using non stationary HMMs

On the contrary, non stationary models are particularly suited to synthesizing smooth observation sequences because the distribution of their hidden states are allowed to change along time.

One such model is the Trended HMM discussed in section 2.3.2.1 where the probability of observing $\boldsymbol{x}_t$ in state j depends on the state sojourn time d_t.

$$p(\boldsymbol{x}_t \mid h_t = j, d_t) = \mathcal{N}(\boldsymbol{\mu_j}(d_t), \Sigma_j)$$

However, there are two problems with this non stationary model. First, its requires a modified Viterbi algorithm for inference which is quadratic in the length of the sequence, thus precluding the use of long sequences. Secondly, the model is not parameterized by an input variable. In the application we want to address, driving animation from speech, we need to condition the animation (output) stream from an audio (input) stream.

Another candidate model is the Input Output HMM where the HMM emission and transition distributions can be conditioned on a specific input $\boldsymbol{u}_t$ which varies in time

$$p(\boldsymbol{x}_t \mid h_t, \boldsymbol{u}_t) \quad \text{and} \quad p(h_t \mid h_{t-1}, \boldsymbol{u}_t)$$

In [52], Yan Li et al learned audio visual mapping using an InputOutput HMM where the transition probabilities between S hidden states are modeled by $S \times S$ Neural Networks. They use 3D facial animation points as observation features $\boldsymbol{x}_t$ and audio features such as MFCC and energy as inputs $\boldsymbol{u}_t$ given as entry to Neural Networks. At synthesis time, they use only audio input to define a most likely hidden state sequence. Given this state sequence, they generate a new likely facial animation by maximizing the likelihood of

the generated data in a EM style procedure. Yet, their synthesized sequence tends to converge to the means of the states, this is a solution that we want to avoid with the help of the technique explained in 5.2.2. Furthermore learning so many Neural Networks is a cumbersome task.

Finally, the Contextual HMMs are also a candidate as a non stationary HMM for synthesis. Its Gaussian emission probabilities and transitions can depend on dynamic input variables $\boldsymbol{\theta}_t$. In addition, the simplicity of the parameterization we proposed make them notably easier to train than e.g. Input Output HMMs.

5.2.2 Synthesis with constraints

Apart from using a non stationary HMM for synthesis, one can use any type of HMM and apply some kind of post-processing to smoothen the generated sequence. In this regard, a key technique has been proposed by [76] to synthesize more realistic smooth trajectories from a standard Gaussian mixture HMM. In standard HMMs, the most likely observation sequence for a state sequence is a piecewise constant function. It follows the means of the Gaussians in each state. This trajectory is not realistic and do not look like the training data at all. On the contrary, the Tokuda algorithm uses the states distributions of a standard HMM to generate non stationnary state and smooth trajectories which are much more realistic.

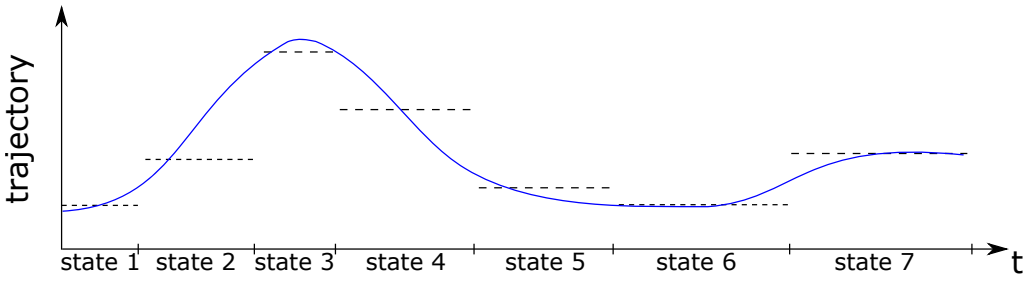


FIGURE 5.1: Example of a possible Tokuda trajectory along a 7 states sequence of a HMM. Means of Gaussians in each state are printed as dotted lines.

The idea of Tokuda algorithm is to augment the observation vector of an HMM with first and second order derivatives as is usually done in speech recognition for a better modeling. But, additionally, we impose consistency constraints between static and dynamic features during synthesis.

We will assume an observation vector $\boldsymbol{x_t}$ consisting of static features $\boldsymbol{c_t}$ as well as dynamic feature vectors $\Delta \boldsymbol{c_t}$, $\Delta^2 \boldsymbol{c_t}$, that is, $\boldsymbol{x_t} = \left[\boldsymbol{c}_t^T, \Delta \boldsymbol{c_t^T}, \Delta^2 \boldsymbol{c_t^T}\right]^T$. Theses dynamic features vectors are computed with a linear combination of their neighboring static features

$$\Delta \boldsymbol{c_t} = \sum_{\tau=-L_-^{(1)}}^{L_+^{(1)}} w^{(1)}(\tau)\boldsymbol{c_{t+\tau}} \tag{5.1}$$

$$\Delta^2 \boldsymbol{c_t} = \sum_{\tau=-L_-^{(2)}}^{L_+^{(2)}} w^{(2)}(\tau)\boldsymbol{c_{t+\tau}} \tag{5.2}$$

where $\{w^{(d)}(\tau)\}_{\tau=-L_-^{(n)}\dots L_+^{(n)}}$ are given coefficients to calculate the n^{th} order dynamic features. The widows length $L = \max_{n\in 1,2, s\in(-,+)} L_s^{(n)}$ to compute static these dynamic features is generally set from 1 to 4 [30].

Training with full frames $\boldsymbol{x_t} = \left[\boldsymbol{c}_t^T, \Delta \boldsymbol{c_t^T}, \Delta^2 \boldsymbol{c_t^T}\right]^T$ and synthesizing the most likely observation sequence may lead to inconsistency, that is Eqs (5.1) and (5.2) are not guaranteed to be verified.

However, there are two methods for synthesis under the constraints 5.1 and 5.2 that we will detail here after. First, we present the single method which maximizes $p(\boldsymbol{x} \mid \boldsymbol{h}; \Lambda)$ for a given hidden sequence $\boldsymbol{h}$. Then we explain the integrated method which takes all hidden states paths into account and maximizes $p(\boldsymbol{x}; \Lambda)$ independently of $\boldsymbol{h}$. The key idea is to directly optimize for $\boldsymbol{c}$ with constraints instead of $\boldsymbol{x}$ in $p(\boldsymbol{x} \mid \boldsymbol{h}; \Lambda)$ or $p(\boldsymbol{x}; \Lambda)$.

5.2.2.1 Single method

Assume $\boldsymbol{h}$ and its associated mixture sequence $\boldsymbol{i}$ is known. We first define $\boldsymbol{c}_t$ as a $M \times 1$ dimensional vector of static features for time t. Then, $\boldsymbol{x}$ is a $3MT \times 1$ vector representing the observations sequence so that we can write :

$$p(\boldsymbol{x} \mid \boldsymbol{h}; \Lambda) = -\frac{1}{2}\boldsymbol{x^T} U \boldsymbol{x} + \boldsymbol{x^T} U V + K$$

where K is simply a constant (independent of $\boldsymbol{x}$) linked to Gaussian terms and

$$U = diag\left[\Sigma_{h_1,i_1}^{-1}, \Sigma_{h_2,i_2}^{-1}, \dots, \Sigma_{h_T,i_T}^{-1}\right]$$
$$V = \left[\mu_{h_1,i_1}^T, \mu_{h_2,i_2}^T, \dots, \mu_{h_T,i_T}^T\right]^T$$

μ_{h_t,i_t}^T and Σ_{h_t,i_t}^{-1} are respectively the $3M \times 1$ mean vector and the $3M \times 3M$ inverse covariance matrix associated with mixture i_t in state h_t.

It is clear that without constraints 5.1 and 5.2, $p(\boldsymbol{x} \mid \boldsymbol{h}; \Lambda)$ is maximized when $\boldsymbol{x} = V$. That is, the best observations sequence $\boldsymbol{x}$ is the sequence of the mean vectors in the state path $\boldsymbol{h}$ having mixtures $\boldsymbol{i}$.

Now we can explicitly impose constraints 5.1 and 5.2 by arranging them into the matrix form:

$$\boldsymbol{x} = W\boldsymbol{c}$$

where

$$\begin{aligned}
\boldsymbol{c} &= [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_T]^T \\
W &= [w_1, w_2, \ldots, w_T]^T \\
w_t &= \left[w_t^{(0)}, w_t^{(1)}, w_t^{(2)}\right]
\end{aligned}$$

$$\begin{aligned}
w_t^{(n)} = \big[&0_{M\times M}, \ldots, 0_{M\times M}, \underbrace{w^{(n)}(-L_-^{(n)})I_{M\times M}}_{(t-L_-^{(n)})\text{-th}}, \\
&\ldots, \underbrace{w^{(n)}(0)I_{M\times M}}_{\text{t-th}}, \ldots, \underbrace{w^{(n)}(L_+^{(n)})I_{M\times M}}_{(t+L_+^{(n)})\text{-th}}, \\
&0_{M\times M}, \ldots, 0_{M\times M}\big]^T, n = 0, 1, 2
\end{aligned}$$

The transform matrix is illustrated in the following figure

Consequently maximizing $p(\boldsymbol{x} \mid \boldsymbol{h}; \Lambda)$ with respect to $\boldsymbol{x}$ is equivalent to maximizing $p(W\boldsymbol{c} \mid \boldsymbol{h}; \Lambda)$ with respect to $\boldsymbol{c}$. Thus, optimizing the emission probability $p(W\boldsymbol{c} \mid \boldsymbol{h}; \Lambda)$ with respect to $\boldsymbol{c}$ at synthesis do not require reestimating parameters for the dynamic features. Then, by setting

$$\frac{\partial \log p(W\boldsymbol{c} \mid \boldsymbol{h}; \Lambda)}{\partial \boldsymbol{c}} = 0$$

we obtain the equations

$$W^T U W \boldsymbol{c} = W^T U V \tag{5.3}$$

A direct solution of 5.3 for $\boldsymbol{c}$ needs $O(T^3M^3)$ because W^TUW is a $TM \times TM$ matrix. However, when $U_{h,i}$ is diagonal and by using the special structure of W^TUW, 5.3 can be solved for $\boldsymbol{c}$ by Cholesky decomposition in $O(TML^2)$ with $L = \max_{n\in 1,2, s\in(-,+)} L_s^{(n)}$ (cf [76]).

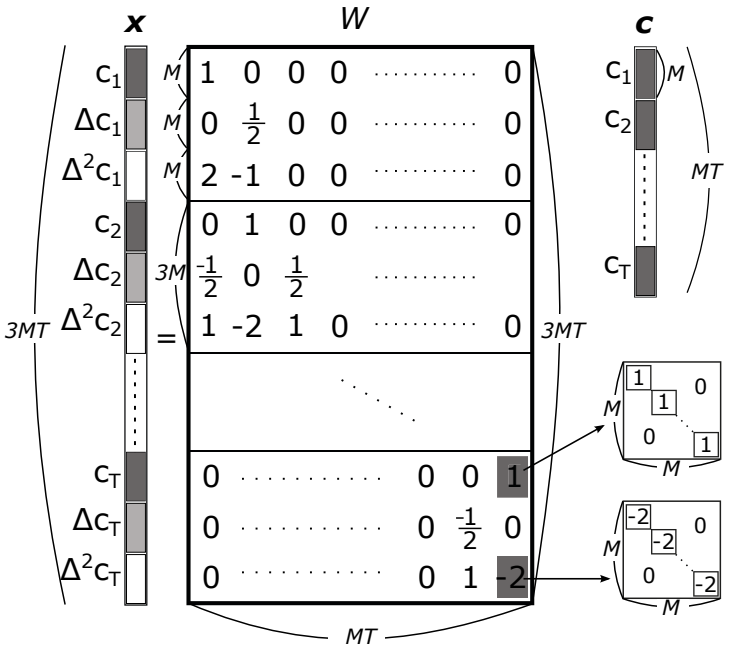


FIGURE 5.2: W transform matrix structured as a block diagonal matrix. Dynamic features are computed using 5.1 and 5.2 with $L_{-}^{(1)} = L_{+}^{(1)} = L_{-}^{(2)} = L_{+}^{(2)} = 1$, $w^{(1)}(-1) = -0.5$, $w^{(1)}(0) = 0$, $w^{(1)}(1) = 0.5$, $w^{(2)}(-1) = 1$, $w^{(2)}(0) = -2$, $w^{(2)}(1) = 1$

5.2.2.2 Integrated Method

Assume the state sequence $\boldsymbol{h}$ is unknown, then we are looking to maximize directly for $p(\boldsymbol{x}; \Lambda)$ independently of $\boldsymbol{h}$. This can be done by maximizing an auxiliary function Q with respect to $\boldsymbol{x}$ in an Expectation Maximization procedure.

The auxiliary function is defined as :

$$Q(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h}} p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) \log p(\boldsymbol{x}', \boldsymbol{h}; \Lambda) \tag{5.4}$$

where $\boldsymbol{x}$ is the current observation sequence and $\boldsymbol{x}'$ is the updated one we need to find.

One can show that maximizing $Q(\boldsymbol{x}, \boldsymbol{x}')$ with respect to $\boldsymbol{x}'$ and setting $\boldsymbol{x} \leftarrow \boldsymbol{x}'$ guarantees an increase in the likelihood $p(\boldsymbol{x}; \Lambda)$ unless $\boldsymbol{x}$ is already a local maximum of the likelihood.

(5.4) can be rewritten as (cf [76]) :

$$Q(\boldsymbol{x}, \boldsymbol{x}') = p(\boldsymbol{x}; \Lambda) \left(-\frac{1}{2} \boldsymbol{x}'^T \overline{U} \boldsymbol{x}' + \boldsymbol{x}'^T \overline{UV} + \overline{K} \right)$$

with:

$$\begin{aligned}
\overline{U} &= diag\left[\overline{U_1}, \overline{U_2}, \ldots, \overline{U_T}\right] \\
\overline{U_t} &= \sum_{h,i} \gamma_t(h,i)\Sigma_{h,i}^{-1} \\
\overline{UV} &= \left[\overline{U_1\mu_1}^T, \overline{U_2\mu_2}^T, \ldots, \overline{U_T\mu_T}^T\right]^T \\
\overline{U_t\mu_t} &= \sum_{h,i} \gamma_t(h,i)\Sigma_{h,i}^{-1}\mu_{h,i}
\end{aligned} \tag{5.5}$$

Maximizing $Q(\boldsymbol{x}, \boldsymbol{x}')$ with respect to $\boldsymbol{x}' = W\boldsymbol{c}'$ imply

$$W^T\overline{U}W\boldsymbol{c}' = W^T\overline{UV} \tag{5.6}$$

which is similar to eq 5.3, hence we can solve eq 5.6 for $\boldsymbol{c}'$ in the same way.

Finally, the algorithm can be summarized as this:

Algorithm 2 integrated method

1: Choose an initial vector sequence $\boldsymbol{c}$
2: Compute $\gamma_t(h,i) = p(h_t = h, i_t = i \mid \boldsymbol{x} = W\boldsymbol{c}; \Lambda)$ with forward-backward
3: Compute $\overline{U}$ and $\overline{UV}$ and solve for $\boldsymbol{c}'$ in eq 5.6
4: $\boldsymbol{c} \leftarrow \boldsymbol{c}'$
5: $\boldsymbol{x} \leftarrow W\boldsymbol{c}$
6: loop to 2 until $p(\boldsymbol{x}; \Lambda)$ no more improve

5.3 Speech to motion synthesis, an application

Synthesis is particularly useful in the domain of character animation. For instance, nonverbal communicative behaviors during speech are important to model a virtual agent able to sustain a natural and lively conversation with humans. In this context, we investigated the framework of Contextual HMMs. Such non stationary models may be used to synthesize automatically realistic character animations from synchronized speech. Furthermore, they can be used in conjunction with the methods of [40] that we have presented in section 5.2.2. In the following sections we will combine CHMMs modeling with both methods from section 5.2.2.1 and 5.2.2.2 to synthesize accurate eyebrow motions from speech.

5.3.1 Related work

Few researchers have presented statistical models using visual *and* speech streams to synthesize realistic animations (including body or face). [50, 51] and [14] generate body motions from spoken speech. Given the tight relationship between acoustic phonemes and visual visemes, speech is also used to drive lip motion in [10, 85]. While these works mainly focused on verbal content, other works have tackled the problem of synthesizing nonverbal communicative behaviors during speech, such as head and eyebrow motion. A key idea that was followed by a number of researchers has been to use Gaussian distribution on feature vectors including speech and motion features to capture the correlation between these two types of features. [16, 47] used Gaussian Mixture Model (GMM) while [69] and [11, 12, 40, 55] used HMMs. This latter approach is probably the most popular one for synthesizing behaviors from speech (we will use this as a baseline in our experiments). It consists in designing a Gaussian joint HMM, named λ hereafter, working on concatenated observation vectors for the two streams (i.e. a frame at time t is $\boldsymbol{x}_t = \left[\boldsymbol{x}_t^{1T} \boldsymbol{x}_t^{2T}\right]^T$ where $\boldsymbol{x}_t^{iT}$ stands for the transposed feature vector at time t for stream i). The application here is to predict the second stream $\boldsymbol{x}^2$ (namely the movements features) based on the first stream $\boldsymbol{x}^1$ (the speech). A key point is that one can build from the joint HMM a Gaussian HMM for every stream, named λ_1 and λ_2 by keeping only parameters related to the stream. Note that these models λ_i, have the same architecture and share transition probabilities. Based on this, once a joint HMM is trained, one can synthesize a trajectory for the second stream from the first stream as follows. Using λ_1 one determines the most likely state sequence. Then using λ_2, one can determine a synthesized trajectory for the second stream using the single method of [76]. Alternatively, one may use the most likely state sequence determined by λ_1 to compute an initial static feature sequence for λ_2. Concretely, $\boldsymbol{c}$ in algorithm 2 is initialized by the Gaussian means of static features in λ_2 which follow the most likely state sequence in λ_1. Then using λ_2 one can determine a synthesized trajectory for the second stream using the integrated method.

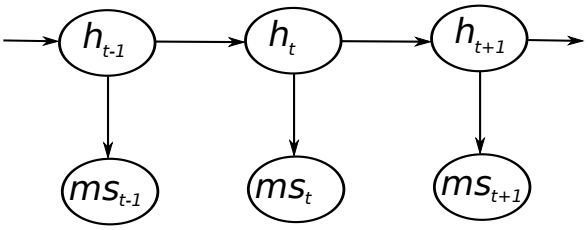


FIGURE 5.3: Representation of a HMM used for speech to motion synthesis in [40] as a Dynamic Bayesian Network (DBN). Motion and speech features ($\boldsymbol{m}_t$ and $\boldsymbol{s}_t$) are coupled in observation frames ($\boldsymbol{x}_t = [\boldsymbol{m}_t, \boldsymbol{s}_t]$) and their interdependency is modeled through covariance matrices.

5.4 Speech to motion synthesis using Contextual Markovian models

We present below an approach that is based on the framework of Contextual HMMs. We will now show how Contextual Markov Models can be used to infer motion from speech. This reference method generalizes in particular the method in [40] presented in section 5.2.2.

The proposed modeling is illustrated in Figure 5.4 as a dynamic Bayesian network. In the following we consider a training dataset where every observation sequence is a sequence of frames $\boldsymbol{x}_t$'s that are composed of motion features $\boldsymbol{m}_t$ and of speech feature $\boldsymbol{s}_t$.

5.4.1 Parameterizations

For synthesis, we tested two different types of CHMMs. $\mu CHMM$ where only the Gaussians means are parameterized and $\mu TrCHMM$ with a parameterization of both Gaussian means and transitions probabilities. Both models use a time dependent speech contextual variables $\boldsymbol{\theta}_t$.

To reduce complexity both at training and synthesis time, we employ diagonal covariance matrices.

The state transition distribution $a_{i,j}$ from i^{th} state to j^{th} state at time t is defined by eq 3.4 while the mean parameterization is given by eq 3.1.

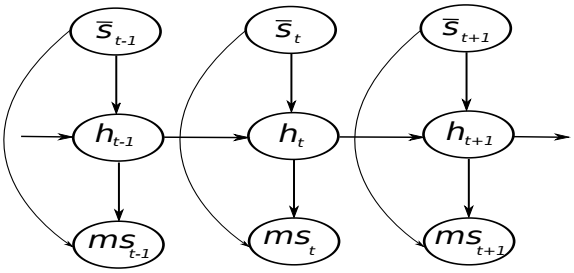


FIGURE 5.4: Representation of CHMM as a DBN. State at time t is noted h_t and short term mean of speech feature vectors (when speech is used as contextual variable) is noted $\bar{s}_t$.

5.4.2 Training

To design a speech-to-motion system we learn one CHMM with speech features as (dynamic) contextual variables and with both motion and speech features as observations

as in [40]. Note that when used as contextual variables we use short term means of the speech frames computed on a sliding window of length 10 (we note these contextual variables $\bar{\boldsymbol{s}}$). Once such a model is trained, one can determine a CHMM on speech λ_s only by ignoring probability distribution function parameters related to motion features. Also, one can define a CHMM on motion by ignoring the probability distribution parameters related to speech features, we note this model $\lambda_{m|s}$. Actually, λ_s and $\lambda_{m|s}$ are CHMMs which still depend on speech contextual variables $\bar{\boldsymbol{s}}_{\boldsymbol{t}}$. Depending on the experimental setting, only their Gaussian means ($\mu CHMM$) or their means and their state transition matrix ($\mu trCHMM$) will change with $\bar{\boldsymbol{s}}_{\boldsymbol{t}}$.

5.4.3 Synthesis

At the synthesis step, speech features are first processed with λ_s to find the most likely state sequence, then we use the single method from 5.2.2.1 (or respectively the integrated method from 5.2.2.2) to synthesize a trajectory with the motion model $\lambda_{m|s}$. While this approach is close to [40], however, we use contextual HMMs instead of HMMs, which allows capturing complex dependencies between speech and motion, yielding improved synthesis as we will demonstrate.

5.4.4 Experiments

5.4.4.1 Dataset

Experiments have been performed on the Biwi 3D Audio-visual Corpus of Affective Communication database (B3D / AC) [27]. 14 subjects were invited to speak 80 short English sentences. In total, this corpus includes 1109 sequences, each lasting 4.67s long on average. We used a part of this corpus corresponding to 240 sentences from three subjects. We manually annotated the data with respect to five labels $Y = \{c_1, \ldots, c_5\}$ that consist in combination of Action Units [1] (including a no move label). A sequence of observations is then annotated by a sequence of labels (a specific combination of action units) together with their boundaries, just like a speech signal is annotated in phones. Every training sequence consists then in a triple $(\boldsymbol{s}, \boldsymbol{m}, \boldsymbol{y})$ of a sequence of speech feature vectors (of length T), a sequence of motion feature vectors (of length T) and a sequence of labels $\boldsymbol{y}$ (of length T , with $\forall t, y_t \in Y$). We preprocessed each sequence to get a speech stream and an eyebrow motion stream at the same rate of 25 frames (i.e. feature vectors) per second (fps). For the motion stream, we gathered four features for each

[1] An Action Unit AU as defined by [25] is a minimal visible muscular contraction (e.g. raise eyebrow). Facial expressions are described as a combination of AUs and express emotional state (anger, fear, sadness, surprise...)

eyebrow corresponding to four facial animation points (FAP) as defined by the MPEG-4 standard [62] (see Figure 5.5); these features move with respect to a neutral pose according to FAPs values. We computed average values for the 4 FAPs between the two brows. Concerning speech, we used prosodic features (short term pitch and RMS energy) which we extracted with PRAAT [8]. We used augmented feature vectors both for motion and for speech streams by adding first and second order derivatives of static features (i.e. velocity and acceleration). Hence we get 6 dimensional frames for speech and 12 dimensional frames for motion. In contextual models, the speech feature $\bar{\boldsymbol{s}}$ used as contextual variables are short term means of the speech frames computed on a sliding window of length 10 (found by trials and errors to give the best results).

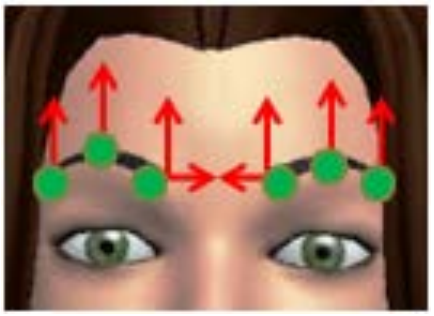


FIGURE 5.5: Illustration of the extracted facial animation parameters (arrows illustrate displacements).

5.4.4.2 Results

We performed experiments with our approaches and with the method in [40]. We considered as many models as there are eyebrow motion classes (5). We used an ergodic model for the no motion class and left-to-right models for the other classes. We trained the models with a dataset including speech and motion features for each sentence. We first trained independently class models (whatever the models used, HMM, μCHMM, μtrCHMM) using corresponding segments of training sequences. Then we combined these submodels into a composite model which is re-estimated on whole sentences. For the test, we use the sequence of speech features only. We primarily evaluated our methods with respect to a reconstruction error, i.e. the mean squared error between the synthesized motion signal (from the speech signal) and the real motion signal (MSE criterion). To gain more insight on the behavior of the methods, we also evaluated the methods with respect to their labeling quality, i.e. the recognition of the sequence of labels. We computed the recognition accuracy with respect to the Hamming distance (H criterion) and to the edit distance (E criterion) between recognized and manually

annotated sequences of labels. Reported results are averaged over 20 random splits of the dataset with 80% for training and 20% for testing (standard deviation in brackets).

TABLE 5.1: Performance of the models with respect to the synthesis quality (MSE) and the labelling accuracy where accuracy is computed by evaluating Hamming distance (H) and edit distance (E). Performance are averaged results gained on 20 experiments (standard deviation are given in brackets)

Model	#state	MSE	Acc (H)	Acc (E)
HMM [40]	3	0.67(0.052)	37%(4.7)	45%(4.2)
	5	0.59(0.042)	43%(4.7)	49%(4.4)
	7	0.56(0.056)	53%(5.7)	51%(4.3)
μCHMM	3	0.51(0.055)	55%(4.8)	49%(4.4)
	5	0.49(0.064)	58%(5.7)	50%(4.9)
	7	0.47(0.056)	59%(4.5)	50%(3.4)
μtrCHMM	3	0.55(0.042)	60%(5.3)	57%(4.7)
	5	0.46(0.051)	61%(5.1)	61%(3.8)
	7	0.45(0.037)	63%(3.0)	62%(3.7)

TABLE 5.2: Similar results as in Table 5.1 but where we assume the sequence of labels of each test observation sequence is known (but not the time boundaries). Here Acc(E) would be 100% in every entry

Model	#state	MSE	Acc (H)
HMM [40]	3	0.43(0.055)	73%(4.7)
	5	0.39(0.051)	75%(4.4)
	7	0.36(0.063)	78%(4.7)
μCHMM	3	0.37(0.057)	77%(5.0)
	5	0.31(0.061)	81%(4.7)
	7	0.30(0.061)	82%(5.0)
μtrCHMM	3	0.33(0.043)	80%(4.1)
	5	0.28(0.048)	83%(5.3)
	7	0.25(0.052)	84%(4.9)

Table 5.1 reports the performance, on the test set, of the 3 methods with respect to the three evaluation criteria and for a number of states per class model ranging from 3 to 7. For HMM and $\mu CHMM$ we employ the single synthesis method (from section 5.2.2.1), while for the $\mu trCHMM$ we employ the integrated method (5.2.2.2). Note that only the MSE criterion is affected by the synthesis method, recognition accuracy is still comparable between HMM, $\mu CHMM$, and $\mu TrCHMM$.

As can be seen in Table 5.1, our two novel approaches (μCHMM, μtrCHMM) perform better than conventional HMMs used by [40] and the performance with μtrCHMM is the best both in synthesis and recognition. Table 5.2 reports similar results in a slightly different setting. We computed the same performance criterion as in Table 5.1 but in that case, the sequence of labels was assumed known for every test sequence (but not the time boundaries between labels). Of course the H and MSE obtained here show significant improvements compared to Table 5.1, but the gap is not so big. This means

that even if the system does not always recognize labels, it does not affect too much the synthesized motion stream.

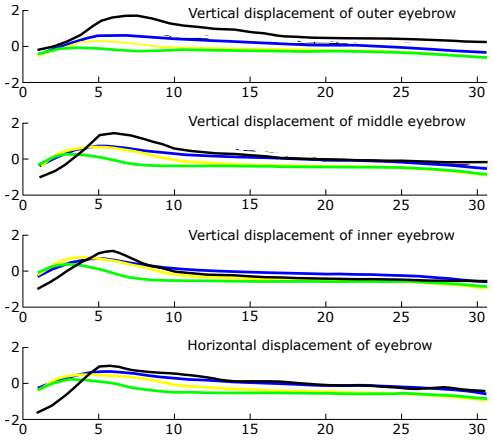


FIGURE 5.6: Comparison of normalized motion sequence synthesized by HMM (green), μCHMM (yellow) and μtrCHMM (blue). The original normalized motion sequence is in black. The four boxes correspond to the four motion features. In every box, the curves show the evolution of motion features values (y-axis) along time (x-axis) when action unit (AU1+AU2) is performed.

Figure 5.6 shows an example (from the test set) of real trajectories of four motion features along with their synthesized trajectories (HMM, μCHMM and μtrCHMM). The trajectories have been synthesized based on speech features only. We can note that μCHMM and μtrCHMM provide results that are closer to real eyebrow motion.

5.5 Conclusion

In this chapter, we presented a method exploiting contextual hidden Markov models to synthesize realistic eyebrow motions from speech. We made use of two important assets: using a non stationary state model which captures correlations between speech and motion, and, enforcing consistency constraints between the generated features and their derivatives at synthesis time.

The whole method can be summarized as follows. The first step is to train a contextual model using speech as contextual variables and with both speech and motion streams as observations. This gives a joint model of speech and motion that captures the correlation between both streams. The second step is to break up the joint model into two different models for the purpose of synthesis. One model of the speech alone λ_s by ignoring probability distribution function parameters related to motion features, and, one model of motion given the speech $\lambda_{m|s}$ by ignoring probability distribution function parameters related to speech. Though, λ_s and $\lambda_{m|s}$ are CHMMs still conditioned on

speech contextual variables $\bar{\boldsymbol{s}}_{\boldsymbol{t}}$. Then, given the speech stream and its model λ_s we can find the most likely hidden state sequence. Lastly, using this hidden state sequence, the motion model $\lambda_{m|s}$ together with the method of [76] can be used to generate a smooth motion sequence from the speech stream.

Our results show that contextual models combined with the method of [76] are significantly better than a benchmark method in the field [40]. Indeed, Tables 5.1 and 5.2 show that using contextual modeling improves both the recognition ability of the system, and the quality of generated sequences (MSE) compared to a HMM using the same single synthesis method exposed in section 5.2.2.1. Also, we can note that parameterizing the transition probabilities improves the recognition accuracy over a mean only parameterized CHMM. Lastly, combining a $\mu TrCHMM$ and the integrated method (5.2.2.2) in place of the single method (5.2.2.1) allows generating the best motion sequences. This simple application demonstrated that not only the added expressivity of our contextual models can improve the quality of synthesis, but that they effectively combine with the reference method of [76].

Chapter 6

Combining contextual variables

Contents

6.1 Introduction 95

6.2 Dropout regularization 97

6.2.1 Dropout in CHMMs 99

6.3 Multistream combination of variables 100

6.3.1 Experimental setup 101

6.3.2 Contextual variables 101

6.3.3 CHMMs 103

6.3.4 Multistream CHMMs 104

6.4 Conclusion 106

6.1 Introduction

In previous chapters, we have seen that contextual variables can improve the accuracy of the model in a classification setting.

However, a combination of contextual variables does not always generalize better than only few. Inspecting the results from chapter 3 table 3.2, we can note that the training accuracy improves relatively well by adding contextual variables, yet the test performance doesn't always follow up. Figure 6.1 illustrates this tendency.

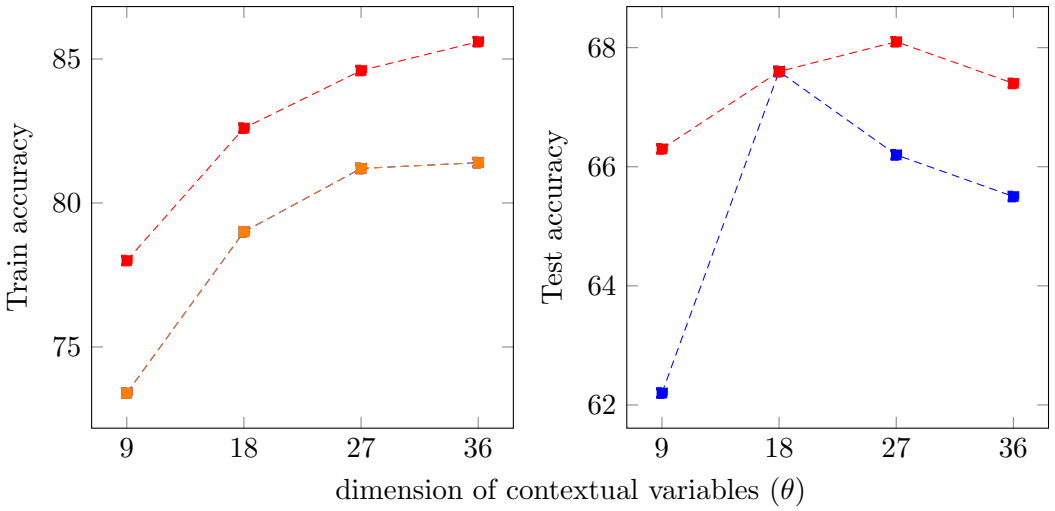


FIGURE 6.1: Train vs Test classification accuracy of mean+covariance parameterized CHMMs (5 states (red), 8 states (blue)) for different sizes of contextual variables $\boldsymbol{\theta}$. Results extracted from Table 3.2 (chapter 3) on the 12-folds IAM handwriting classification dataset (presented in 3.5.1). Each point of the curve uses a different setting of the contextual variables vector. For 9 dimensions $\theta = \mu(41)$, 18 dims $\theta = \mu(61)\ \sigma(55)$, 27 dims $\theta = \mu(61)\ \sigma(55)\ \Delta(15)$, 36 dims $\theta = \mu(61)\ \sigma(55)\ \Delta(15)\ \Delta^2(15)$

The kind of observation from figure 6.1 typically look like an overfitting problem that could be handled through regularization. As shown in chapter 4, it is straightforward to regularize the discriminative counterpart of a Contextual Hidden Markov Model. When transformed into a CHCRF, we can add a L1 or L2 penalty term to the training loss easily.

Noting $\mathcal{L} = \sum_k \log p(y^k|\boldsymbol{x}^k, \boldsymbol{\theta}^k, \Lambda)$ the unregularized loss of Contextual Hidden Conditional Random Field with parameters Λ, we can define a L2 regularized loss $\mathcal{L}_{Reg}$ as

$$\mathcal{L}_{Reg} = \mathcal{L} - C||\Lambda||^2$$

with C a scalar weighting the importance of the regularization.

However standard L1/L2 regularization is not easily achievable in generative HMMs or CHMMs. The maximum likelihood training algorithm involved in HMMs relies on optimizing an auxiliary function Q which guarantees likelihood improvement of the data under the model. We can be tempted to add a penalty term to this loss Q like is it usually done with L1/L2 regularization,

$$Q_{Reg} = Q - C||\Lambda||^2$$

However, by doing this, we loose the guaranty to improve the likelihood of the data under our model. Safe checks can be made to be sure that an update does not decrease the likelihood but, this is not particularly elegant.

In this chapter, our goal is to investigate ways to combine contextual variables more effectively. Our first proposal is a drop out regularization technique inspired by recent developments in learning Deep Neural Networks. This simple scheme is shown to improve the results from chapter 3 in the same handwriting classification task. Secondly, we devise a multistream combination of contextual variables that reveals more suited in a spoken words classification task.

6.2 Dropout regularization

Recently, the deep learning community came up with an interesting idea to limit overfitting. Unlike L1/L2 regularization which takes the form of a penalty term appended to the loss function, dropout [38] behaves differently.

The idea applied to Neural Networks is to cancel hidden layer activations stochastically during training. While forwarding an example through the network, each hidden unit i of Layer l (noted h_i^l) is reset (drop-out) with probability 1-p (and thus retained with probability p).

Formally the feed forward operation of a neural network having L layers with weights W^l and bias b^l in layer l can be described by iterating (for $l = 1, \ldots, L$) the subsequent operations :

$$h_i^{l-1} = \begin{cases} 0 & \text{with probability} \quad (1-p) \\ 1 & \text{otherwise} \end{cases}$$

$$h^l = f(W^l h^{l-1} + b^l)$$

- f any activation function such as sigmoid or tanh
- h^l the hidden layer l, and h_i^l its i^{th} unit
- h^0 being the input $\boldsymbol{x}$ to the network and h^L the output

At test time however, there is no dropout but one has to rescale the weights $W = pW$ to act as in the training regime.

Intuitively, each time an example is presented, it is as if we had a different network (depicted in figure 6.2).

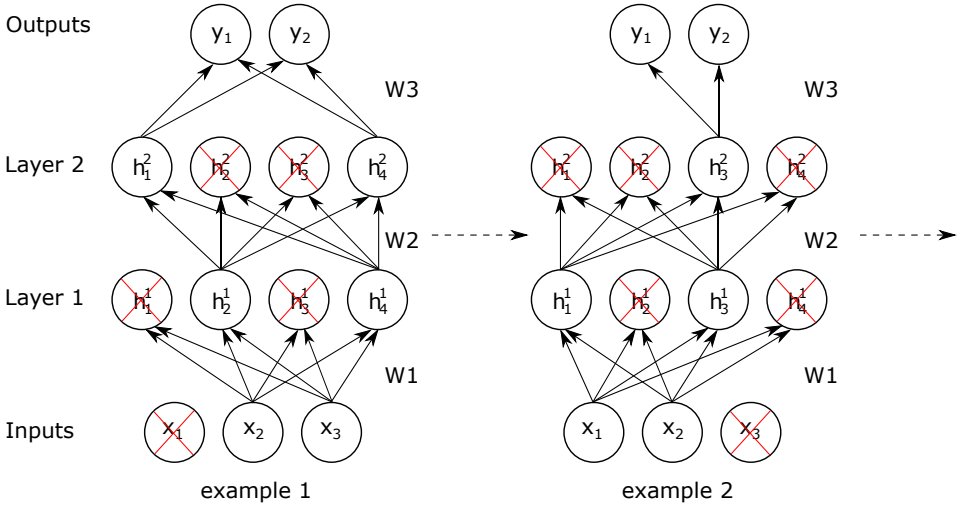


FIGURE 6.2: During training, for each example presented to the network, a new architecture is sampled. Dropped out hidden units are marked by red crossed lines, their outgoing connections then become inactive (light gray)

An interpretation given in [38] is that dropout training can be viewed as a kind of model averaging with a huge number of models. Indeed, for each example, a new network or model is sampled, and, at test time all are combined to produce the final output. To do model averaging the right way, one would normally need to train many separate models and apply each of them to test data, but, this would be a lot more expensive than the dropout strategy in training, as well as in inference.

Another form of dropout known as dropconnect [81] do not cancel hidden unit activations but a proportion of the layer weights. Hidden units in layer l are then computed as :

$$h^l = f((\mathcal{M} * W^l)h^{l-1} + b^l)$$

where $\mathcal{M}$ is a binary matrix encoding the connection information, and * denotes the element-wise product

A simpler interpretation of dropout (more evident in dropconnect) is that, because weights can become inactive, it essentially forces the weights of the model not to rely on each other to improve robustness.

Finally, dropout and dropconnect have been shown to improve generalization of Neural Networks over standard training in various tasks, such as image classification [81] or even phone recognition (using a HMM-NN Tandem architecture) [38]. Also, this new idea not only restricts to neural network training. In [73], Srivastava et al showed how

dropout can be applied to linear regression and how it induces a nice interpretable form of regularization close to ridge regression.

6.2.1 Dropout in CHMMs

As pointed earlier, dropout can be applied to neural networks or linear regression, but this simple idea can straightly be applied to CHMM training.

The idea is to zero out contextual variables during training, consequently, the model parameterization coefficients can no longer rely on each others to maximize the log-likelihood of the data. Dropout training should lead to a certain amount of redundancy, and, in the context of high capacity models, this kind of model averaging may increase generalization as we will see next.

During training, the probability of retaining a dimension of the contextual variable vector is fixed by a Bernoulli distribution with parameter p. Inversely, the dropout factor or the probability of canceling a dimension is then $1 - p$.

Algorithm 3 CHMM Dropout training

let X be the set of training sequences
let p be the proportion of retained variables
$\Theta_{\text{sav}} \leftarrow \Theta$ // *saving untouched contextual variables*
function [CHMM]=CHMM_DROPOUT_TRAINING(CHMM, X, Θ, p)
    for iter $\leftarrow$ 1 to max_em_iter **do**
        $\Theta \leftarrow \Theta_{\text{sav}}$
        for k $\leftarrow$ 1 to $card(X)$ **do**
            for i $\leftarrow$ 1 to size(θ) **do**
                rand $\leftarrow$ uniform random value between $[0, 1]$
                if rand $< p$ **then**
                    $\theta_i^k \leftarrow 0$ // *reset* i^{th} *dimension of* θ^k
                end if
            end for
        end for
        CHMM $\leftarrow$ EM_Training(CHMM, X, Θ, p) // *1 iteration of EM*
    end for
end function

In inference, the coefficients of the mean and covariance parameterization need to be rescaled by p

6.2.1.1 Results

To validate this simple regularization technique, we performed a few experiments on a handwriting classification task, similar to the ones from table 3.2. We use the same

dataset already presented in section 3.5.1.

nb states	nb gauss	Context variable $\theta(t)$	Test $\mu\Sigma$CHMM	Test $\mu\Sigma$CHMM dropout
3	2	$\theta = \mu(61)\ \sigma(55)\ \Delta(15)\ \Delta^2(15)$	67.1 (1)	67.6 (1.6)
5	2	$\theta = \mu(61)\ \sigma(55)\ \Delta(15)\ \Delta^2(15)$	69.8 (0.9)	71.6 (1.6)
8	2	$\theta = \mu(61)\ \sigma(55)\ \Delta(15)\ \Delta^2(15)$	72.6 (1.2)	74.5 (1.4)
8	4	$\theta = \mu(61)\ \sigma(55)\ \Delta(15)\ \Delta^2(15)$	71.9 (1.5)	74.5 (1.4)

TABLE 6.1: Experiments with dropout vs without dropout training using a combination of 36 contextual variables (θ is a 36 dimensional vector)

The results from Table 6.1 shows that overfitting can be limited by the use of this simple drop out technique. 74.5% test performance is our best result on IAM 12 fold dataset and this is achieved by CHMM using covariance parameterization, dropout regularization and a mix of contextual variables (resulting in a 36 dimensional contextual vector). We can notice that drop out training can yield up to 2.5% improvement.

For smaller dimensional contextual vectors, we did not achieve such systematic gains. This is not surprising because if regularization can have a positive impact, it is more likely to be seen in higher capacity models.

6.3 Multistream combination of variables

In the following, we conduct a study experiment with CHMMs on a noisy speech recognition task where we will use domain specific contextual variables, characterizing both intra and extra signal variability. As we observed in section 6.2, dropout regularization can help when the dimension of the contextual vector increases. Unfortunately, our initial tests do not seem to give improvements here. This lead us to consider a different approach to combine contextual variables more effectively.

The results of chapter 3 suggests that covariance parameterization bring less significant improvements than mean only parameterization, thus, as a good trade-off between computation time and accuracy, we perform our investigations on mean only parameterized CHMMs.

6.3.1 Experimental setup

Data come from CHiME corpus [79] [2]. It contains recordings in a domestic environment from 34 speakers (18 men and 16 women). Each speaker reads 6 words sentences following the grammar : <color:4> <preposition:4> <letter:25> <number:10> <adverb:4> where the numbers in brackets indicate the number of choices at each point. Each utterance is then mixed with 6 different level of noise (SNR = -6, -3, 0, 3, 6, 9 dB). More details about the background noise and recording process can be found in [15]. All data are provided as 16 bit WAV files sampled at 16 kHz.

The task is to recognize the letter and digit of each utterance at every SNR. Performance is measured in term of word accuracy. The training set contains 500 utterances from each of 34 speakers. The test sets contains each 3600 utterances (600 utterances $\times$ 6 SNR). These sets are annotated at the word level by the baseline system (HMM using forced alignment). The validation set contains also 3600 unsegmented utterance which we annotated the same way.

The baseline system is a left right HMM with 7 Gaussians using diagonal covariances. It is trained with HTK [87]. A model is learnt for every (word, speaker) pair using all available data whatever the SNR. Each of the 51 words of the vocabulary is modeled by a HMM using the rule of two states per phoneme in the word. These 51 $\times$ 34 (nbwords $\times$ nbspeakers) speaker dependent models are first initialized by a speaker independent HMM with the same topology (7 diagonal Gaussians, best choice between 3-5-7 diagonal Gaussians) trained on all speaker's utterances.

The models are learnt using a classical speech representation given by 39 MFCC coefficients (12 cepstral coefficients and 1 energy term augmented by their first and second temporal derivatives). Speech data is first reduced to one channel by taking the mean of the left and right channels. Then, the coefficients are extracted on 25 ms window with 10ms shift.

6.3.2 Contextual variables

We employed both intra and extra signal contextual variables. The intra signal variables characterize measures derived or estimated from the signal. The extra signal variables represent quantities that we can not estimate from the signal alone.

Signal to noise ratio Few efforts have been employed to make systems more robust to the variability of noise conditions at test time. Robustness to noise can be improved for instance directly by training with noisy sequences. However, first it can decrease

the performance under low noise conditions. Secondly, it has been shown in [31] that the distribution of HMM parameters greatly varies depending on the level of noise in the training data. This difference can cause a severe decrease in performance during recognition. Hence it is not a particular good idea to train a system with mixed levels of noise.

Based on the experimental setup of Xiaodong Cui and Yifan [18] we trained an acoustic model on CHiME with a controlled level of noise (SNR). In figure 6.3 we illustrate the variation of the mean as a function of the SNR in a 4 states HMM model trained on the phonetic sequence /se/. This figure clearly shows that each dimension of the mean vary along the SNR. Thus, it seems a good idea to model the speech signal as a function of the SNR.

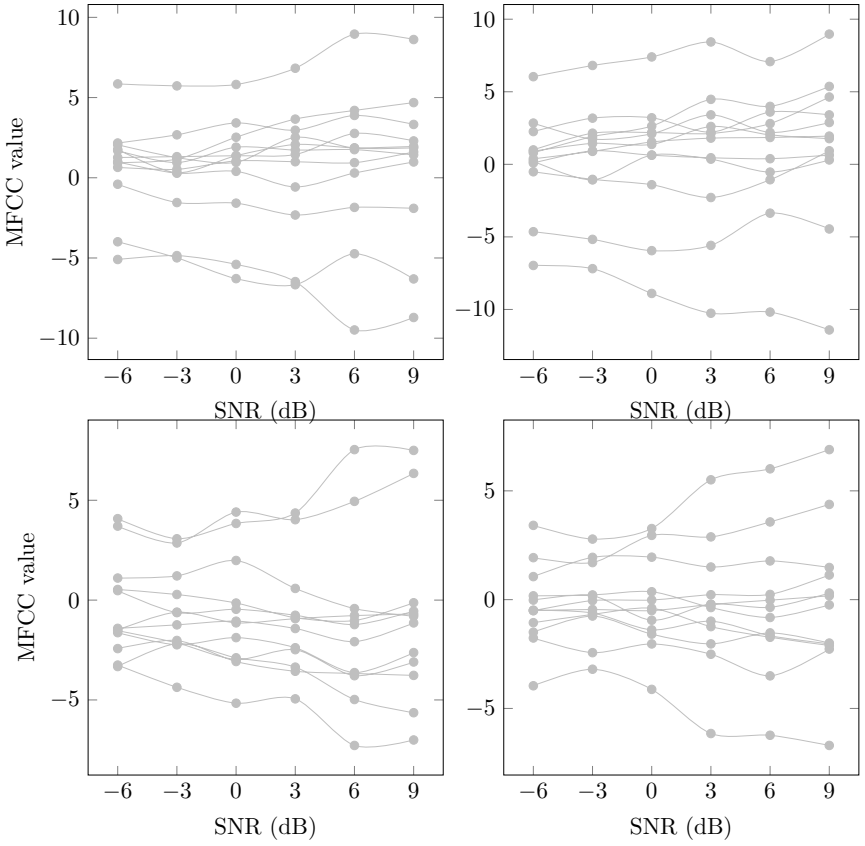


FIGURE 6.3: Means of a 4 states HMM (as a function of SNR) trained on CHiME corpus for the character 'c' (phonetic sequence /se/)

Other extra signal variables The SNR is an important part of signal variability which can not be perfectly estimated from the signal alone. Yet, an other extra signal variability is mainly due to the speaker itself. A person can not utter without variation caused by stress, fatigue, desease, environment etc ... The speech signal also greatly vary in function of the gender, the age, the origin or morphology. All these factors can

be used for a better modeling of the speech signal. In our work, we will experiment two other types of extra-signal parameters. First is the rate of the speech, which can be measured by the number of syllable per second. Second is the speaker identity (that we encode as a 34 dimensional contextual variable).

Spectral flatness We also explored the interest of Spectral Flatness (SFM) which represents the rate of sinusoidal components in each frequency band. It is estimated directly on the signal (intra signal variable) by the ratio of the geometric and the arithmetic mean of the spectrum.

$$SFM(band) = \frac{\left(\prod_{k\in\text{band}} a(k)\right)^{1/k}}{\sum_{k\in\text{band}} a(k)}$$

where $a(k)$ is the spectrum amplitude at frequency k and K the number of frequency k in the band. For a tone like signal, the SFM is close to 0. For a noisy one, SFM is close to 1. It is computed for the whole utterance.

6.3.3 CHMMs

In these preliminary experiments we used all above contextual variables : SNR (scalar), SFM (scalar), Rate (scalar), and ID (encoded as a 34 dimensional vector with 1 in speaker ID^{th} position, 0 everywhere else). Hence, here CHMMs are trained with a 37 dimensional contextual vector. We first searched the topology that worked best between 4,8,16,32 (full or diagonal covariances) Gaussian distributions per state and using 3 states per word model to reduce computational costs. 8 full covariances Gaussians per state was found performing the best. Note that using full covariances was found better than using diagonal ones even for MFCC type features (see table 6.2).

SNR (dB)	Baseline HMM	CHMM 8 Gaussians diag	CHMM 8 Gaussians full
-6	49.33	55.6	58.1
-3	58.67	64.8	67.6
0	67.5	76.2	76.2
3	75.08	81.3	82.6
6	78.83	85.5	86.4
9	82.92	87.6	89.1

TABLE 6.2: letters+digit word recognition accuracy using a combination of Speaker ID, SNR, SFM, and Rate

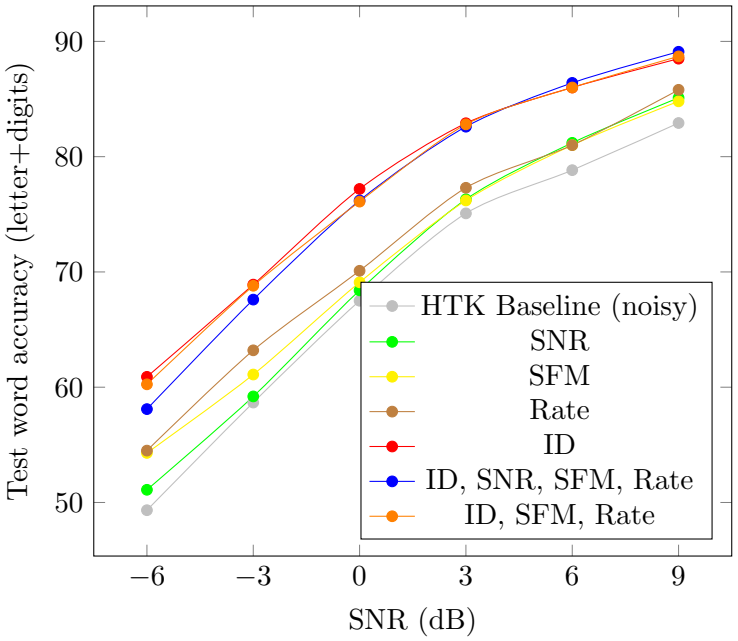


FIGURE 6.4: word recognition accuracy of 8 Gaussians full covariances CHMMs vs Baseline using different parameterization variables (SNR, SFM, Rate, speaker ID, or everything combined)

As we can see here, CHMMs improve significantly over the baseline system (a HMM with 7 diagonal covariance Gaussians). This is noticeable for any variable of parameterization employed (SNR, SFM, Rate, ID). However, combining contextual variable is surprisingly less accurate than using only the speaker ID (except for higher SNRs, 6 and 9dB).

Compared to speaker dependent HMMs (i.e. the baseline with 51×34 models), there are 51 CHMMs, one per word, for any speaker. CHMMs become speaker dependent in inference only, when a HMM is instantiated with the speaker ID as contextual variable. The parameterized models hence *jointly* use all speakers utterances to estimate their parameters contrary to HMMs. This type of estimation, which seems more robust, probably explains the important gap between HMM and CHMMs specially on low SNRs.

6.3.4 Multistream CHMMs

Using contextual variables improved performance systematically, but it is surprising that their combination is sometimes less accurate than using the speaker ID (except for high SNRs 6 and 9dB). Indeed, it seems natural to think that the speaker ID and the SNR for example, being two totally uncorrelated informations and each one giving improvements when used as a single contextual variable should be combined with success, which is not the case.

As an alternative to dropout regularization, we investigated if it was efficient to train different CHMM models, one with each contextual variable independently, and combine them in a more successful way. We refer to this method as a multistream combination of contextual variables.

For this particular application, we measure the classification accuracy on isolated words extracted by the baseline HMM system. For each isolated word of the vocabulary we train 4 CHMMs with a different contextual vector variable. The first CHMM is trained with the SNR as contextual variable, the second with the SFM, the third with the Rate of speech and the last one with the speaker ID. At inference time, the probability of an isolated word w is then expressed as the weighted average of the 4 different CHMMs log probabilities for the word w.

$$\log p(w) = \alpha \ \log p(w \mid \Lambda_{SNR}) + \beta \ \log p(w \mid \Lambda_{SFM}) + \gamma \ \log p(w \mid \Lambda_{Rate}) + \zeta \ \log p(w \mid \Lambda_{ID})$$

where α, β, γ, ζ are scalars (stream weights) controlling the influence of each stream. $p(w \mid \Lambda_\theta)$ is the probability of word w given by the CHMM using θ as contextual variable.

Model selection is performed on validation set, and we report here test classification accuracy of HMM (baseline), CHMM individual streams, and the best combination of streams found selected on validation set. Please note that to validate this type of regularization, we moved from a recognition task (in 6.3.3) to a pure classification task in order to perform fast experiments. Indeed, in isolated classification we can compute $p(w \mid \Lambda_\theta)$ once for all and efficiently search for a good combination of stream weights on the validation set.

SNR (dB)	CHMM 8 Gauss full SNR	CHMM 8 Gauss full SFM	CHMM 8 Gauss full Rate	CHMM 8 Gauss full ID	CHMM 8 Gauss full joint	CHMM 8 Gauss full multistream uniform combination	CHMM 8 Gauss full best combination
-6	55.6	56.9	58.2	63.7	64.4	62.9	**65.1**
-3	62.7	63.1	63.6	69.3	71.5	70.8	**72**
0	70.8	71.1	71.3	77.6	78.1	77.5	**79**
3	77	76.9	77.3	82.2	83.4	83.6	**83.7**
6	82.9	82.9	83.2	87.7	87.8	87.7	**88.7**
9	86.6	86	86.8	90.2	90.4	90.5	**91.3**

TABLE 6.3: Test letter+digits classification accuracy of CHMM individual streams, CHMM combination of individual streams, and normal (joint) CHMM trained on all contextual variables

This table shows that using naive uniform combination of all streams (α, β, γ, ζ set to 1) is less interesting than using a single CHMM to learn all the contextual variables at once. However, if we search for a good combination of the stream weights on validation data, multistream give the best results and so, with more streams (more contextual variables), we get more accuracy contrary to previous results.

6.4 Conclusion

In this chapter we noticed that naively combining contextual variables can decrease performance. This phenomenon occurs both using simple and generic contextual variables that we can compute directly from the observation sequence, and also using domain specific variables like the intra and extra signal variables that we employed in a speech recognition task. As combining contextual variables always increases the training performance but not the test one, this suggests that it can be the result of an overfitting problem. In this respect, we proposed a regularization scheme that indeed helped achieving a better test performance when there are many contextual variables. The drop out style regularization did help in a handwriting classification task using simple generic contextual variables. However, it did not help in a speech classification task using domain specific contextual variables. We believe this may be due to many dimensions of the speech specific contextual variables being zero. Hence, the effect of canceling dimensions of the contextual variables is very limited. We then proposed a multistream approach to combine contextual variables more effectively. This can be viewed as a weighted average of simpler CHMMs each using different contextual variables.

Chapter 7

Toward Transfer Learning

Contents

7.1 Design of a global model 108

7.1.1 Using a class code as contextual variables 110

7.1.2 Task & dataset 111

7.1.3 Preliminary results with one-hot class coding 111

7.1.4 Using a distributed representation of class as contextual variables 113

7.1.5 Retraining discriminatively 116

7.2 Dynamic Factor Graphs 116

7.2.1 Continuous state space models 116

7.2.2 Analogy with Dynamic Factor Graphs 119

7.3 Conclusion 121

Using more expressive models (e.g. augmenting the number of states, the size of the Gaussian mixtures, using contextual HMMs) usually comes with a learning problem, how to reliably estimate the increased number of parameters. We encountered this problem with Contextual HMMs and we presented in the previous chapter two solutions that aim at introducing some regularization in the learning (dropout training) or at taking into account some basic assumptions on the signals to limit the number of parameters to estimate (Multi-Stream combination of contextual variables).

Yet one may tackle the problem with a different viewpoint. One way to avoid overfitting and to overcome the lack of data might be to rely on transfer learning strategies. The idea would be to enable learning complex models of all classes by exploiting training data from all classes. Actually it seems a feasible and good idea when thinking at activity recognition for instance. It is true that a number of different activities (sitting on a chair, sitting on the ground, walking, running, etc) will exhibit some similar subparts in

the corresponding motion capture sequences. Our proposal in this chapter is to design a transfer learning strategy to learn classes with a limited number of training data.

The works we present here are only preliminary and we do not have strong experimental evidence to demonstrate their relevance, yet they include some interesting ideas that are worth presenting in our opinion.

7.1 Design of a global model

Starting from the viewpoint described above a simple idea would be to consider that by using a contextual model for all the classes, with the contextual variable being (or including) a class indicator, would allow to share some information between all the classes. The starting point of our proposal is observing that many classes are similar and would benefit from using the data of other classes to learn their representation. Consider the example of 2 gesture classes below.

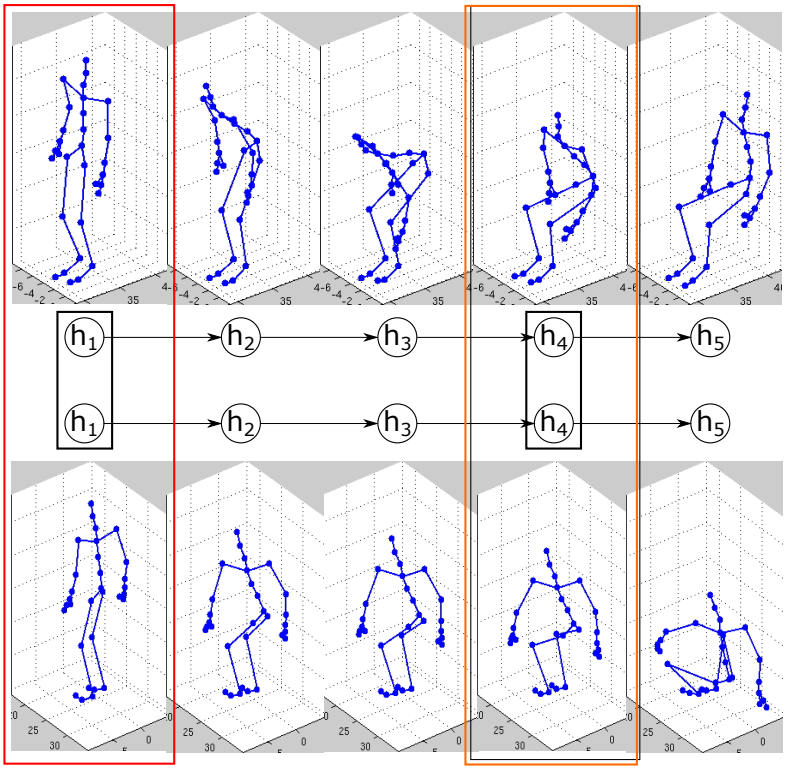


FIGURE 7.1: Two examples of similar gesture classes extracted from HDM05 database

In the top gesture, a person is sitting on a chair, while in the bottom gesture, a person is sitting on the floor. Those two gesture classes can be modeled for instance by a 5 states HMM. Each state in the HMM represents a probability distribution over the possible poses during the course of the gesture. Now in this example, the first and the fourth

poses of the two gestures are very similar and we would like to share at least a part of their representation. The idea is to learn the representation of states 1 and 4 in the two classes with the data of both classes.

A very simple way to do that it to consider one single model, that we name global, equipped with enough capacity (i.e. number of states, size of Gaussian mixtures) and to use as contextual variables $\boldsymbol{\theta}$ for an observation sequence $\boldsymbol{x}$ some coding that include its label information.

In the example below, we draw a global CHMM trained on three gesture classes. Suppose

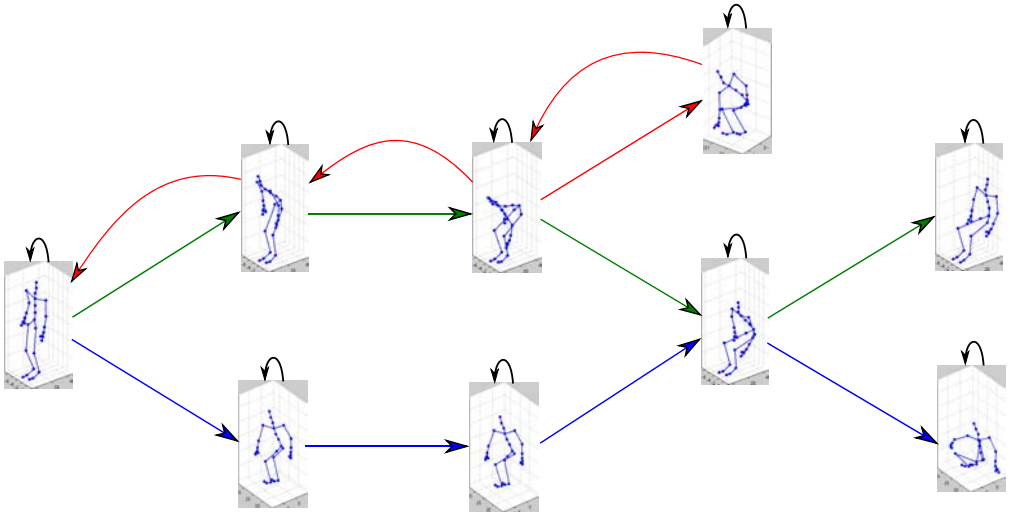


FIGURE 7.2: A global CHMM learnt on 3 gesture classes

that only the transition probabilites in the global CHMM are parameterized by a contextual vector $\boldsymbol{\theta}$ containing the class label. For each value of $\boldsymbol{\theta}$, the global CHMM can be instanciated into a different gesture class. The three paths in the global CHMM, which are printed in different colors (blue, green and red) represent the three possible gesture classes.

The idea is that the parameters of the emission probability distributions of similar poses may be estimated on all classes data. Of course one is not restricted to parameterize only transition probabilities, means and covariances of the Gaussians may also be parameterized. In that case, shared states may not be exactly the same between all classes, they will keep a limited degree of freedom in each class because of the parameterization of the Gaussian distribution functions.

All along this section we will consider the case of sequence classification only so that the label information of an observation sequence is a class. If $\boldsymbol{\theta}$ includes the class information only then the vector $\boldsymbol{\theta}$ does not vary along the sequence. This is the case we consider in the following for the sake of simplicity. Of course $\boldsymbol{\theta}$ might include a static part including

the class code and a dynamic part including other kind of external variables as those considered in previous chapters.

7.1.1 Using a class code as contextual variables

The simplest coding scheme one can imagine is a one-hot code of the class. The vector $\boldsymbol{\theta}$ has a dimension equal to the number of classes (37 in our experiments). For a sequence corresponding to the class y there is a 1 in the y^{th} position and 0 everywhere else. Using such a definition of the contextual variables it is clear that the global model may then be instanced into the model of any of the considered class by using the appropriate setting of $\boldsymbol{\theta}$. It is also quite natural that doing this way will allow learning offsets of parameterized Gaussian means and covariance matrices using all the data from all the classes thus enabling information sharing and transfer from classes to classes.

To jointly train a global CHMM with all classes data, we then define a particular setting of the contextual variable $\boldsymbol{\theta}_y$ which is used for all sequences whose class is y (e.g. one-hot class code). Setting $\boldsymbol{\theta}$ to $\boldsymbol{\theta}_y$ one instantiates the global contextual model Λ into the HMM of class y with parameters Λ_y. The global CHMM may then be learned with all observation sequences from all classes provided all sequences $\boldsymbol{x} \in \mathcal{X}_y$ are assigned contextual variables $\boldsymbol{\theta}_y$.

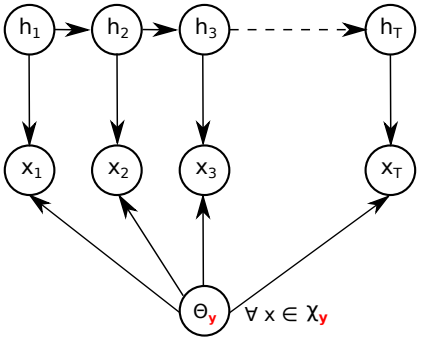


FIGURE 7.3: global CHMM sketched as a Bayesian network. An observation sequence $\boldsymbol{x} \in \mathcal{X}_y$ is assigned a class contextual variable vector $\boldsymbol{\theta}_y$.

If $|Y|$ is the number of classes, instead of training $|Y|$ models each with its own training data, we train here only one global CHMM learned with the data from all the classes.

Performing inference for an input observation sequence $\boldsymbol{x} \in \mathcal{X}$ with such a global CHMM is a two step process. First, we instantiate individual class HMMs from the global CHMM by setting $\boldsymbol{\theta}$ to the $|Y|$ possible class settings $\boldsymbol{\theta}_y$. Then we use the $|Y|$ models to determine the most likely label $y \in Y$ for this observation sequence $\boldsymbol{x} \in \mathcal{X}$.

Below we sketch the first step :

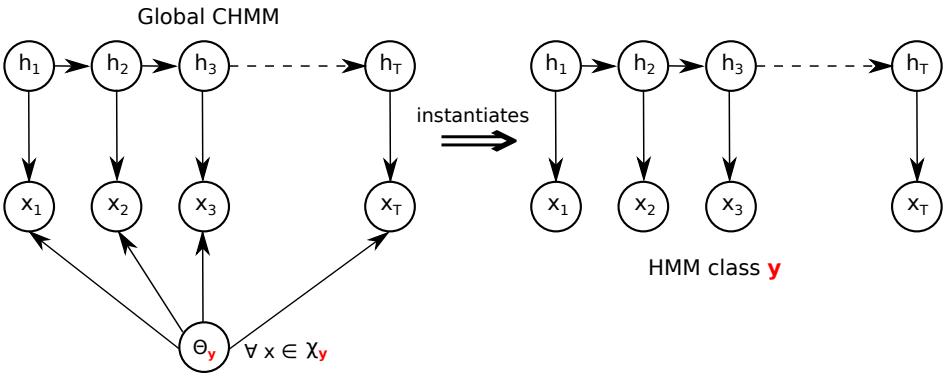


FIGURE 7.4: a global CHMM conditioned on a class contextual variable $\boldsymbol{\theta}_y$ induces a corresponding HMM for the class y.

From a global CHMM parameterized by a class contextual vector $\boldsymbol{\theta}_y$ one first instantiates a HMM with parameters $\Lambda_y = [\mu_1, \ldots, \mu_S, \Sigma_1, \ldots, \Sigma_S, A]$ modeling class y, where $\mu_s = \hat{\mu}_s(\boldsymbol{\theta}_y)$ is the mean for hidden state s of the HMM modeling class y (Cf. Eq. 3.2), and $\Sigma_s = \hat{\Sigma}_s(\boldsymbol{\theta}_y)$ denotes the covariance matrix in state s (Cf. Eq. 3.3). Finally $A = \hat{A}(\boldsymbol{\theta}_y)$ stands for the transition matrix.

Basically, when conditioned on the class contextual variable $\boldsymbol{\theta}_y$, the parameterized means $\hat{\mu}_s(\boldsymbol{\theta}_y)$, covariances $\hat{\Sigma}_s(\boldsymbol{\theta}_y)$, and transition matrix $\hat{A}(\boldsymbol{\theta}_y)$ of the global CHMM define a HMM with means μ_s, covariances Σ_s and transition matrix A. The parameters Λ_y of a HMM modeling class y no more depends on the class contextual variable $\boldsymbol{\theta}_y$.

The second step is performed as a classic inference in Hidden Markov Models:

$$\text{argmax}_y p(\boldsymbol{x}|y; \Lambda_y) \tag{7.1}$$

7.1.2 Task & dataset

As a preliminary experiment we performed a simple classification experiment on a motion capture dataset. It consists of isolated gestures from 37 classes extracted from the HDM05 dataset [59], the gestures are performed by 5 different actors. The observation frames are composed of 62 features which are joint angles representing the body pose of the actor at each time step. From this database, we built 5 folds each with 702 examples for the training set, 225 for validation, and 203 for test.

7.1.3 Preliminary results with one-hot class coding

We assume here that $\boldsymbol{\theta}_y$ is a $|Y|$-dimensional vector ($|Y|$=37 here) where there is 1 in y^{th} position and 0 everywhere else. We call this type of contextual variable the one-hot

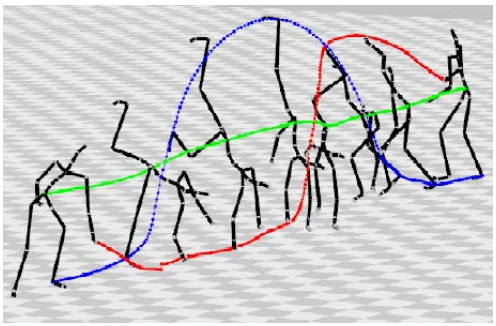


FIGURE 7.5: Motion capture data stream of "cartwheel" gesture represented as a sequence of poses. The figure shows the 3D trajectory of 3 joints (root, right finger, left ankle).

class code representation of class y.

For the experiments, we trained 37 left-right full covariances HMMs that we compared to a global left right CHMM with a one-hot code class representation as contextual variables. As the data is unbalanced, we output the macro average F1 to measure the classification accuracy (standard deviation is given in brackets). Here, the global CHMM has mean and full covariance parameterizations.

	model	covariance	nb states	nb gauss	Train average F1	Test average F1	number of parameters
independent class models	HMM	full	8	1	88.2(0.7)	62.2(1.9)	1158544
	HMM	full	8	5	100(0)	53.1(6.1)	5784728
	HMM	diag	8	1	87.7(0.7)	66.3(3.1)	39072
	HMM	diag	8	2	89.4(1.1)	66.4(3.1)	76368
	HMM	diag	8	5	91.8(2.5)	65.7(3.4)	185368
	HMM	diag	8	7	92.4(2)	63.5(2.5)	261368
	HMM	diag	8	9	94.2(1.2)	63.1(2.9)	335368
global model	$\mu\Sigma$CHMM	full	8	1	91.4(1.2)	67.4(2.9)	68512
	$\mu\Sigma$CHMM	full	8	5	97.1(0.6)	70.9(3.3)	342344

TABLE 7.1: F1 Accuracy of standard full Gaussians HMMs vs global CHMMs parameterized by a one hot code class contextual variable

In table 7.1, we can clearly notice that standard full Gaussians HMMs quickly overfit the data. With 8 states and 5 Gaussians per gesture class, its test accuracy decreases (from 62.3% to 53.1%) compared to using only 1 Gaussian whereas its training accuracy jumps to 100%. Moving to diagonal Gaussian significantly reduces the number of parameters and clearly improves generalization, but the HMMs accuracy quickly stabilizes at out around 66%.

The behaviour of the global contextual HMM is different. Using a bigger model improves both training and test accuracy. Remarkably, although we did not took much time to tune the global model learning as much as we did for HMMs, its accuracy is already

noticeably higher than that of all the HMMs we tested. And of course, the global CHMM with full Gaussians has less parameters than the many HMMs with full Gaussians, yet with 5 full covariance matrices Gaussian mixtures per state it has already more parameters than all the diagonal HMMs investigated in these experiments. Hence it seems that a globally trained CHMM effectively makes a better use of its parameter set and of all the training data from all classes.

We were at first quite surprised by such gains because in the global CHMM architecture we investigated here, the class codes are fully orthogonal resulting in a somehow limited sharing of the parameters between the classes. Only biases (offsets) are shared as the parameterizations for a CHMM modeling class y for a hidden state s yield:

$$\begin{aligned}\hat{\mu}_s(\boldsymbol{\theta}_y) =& V_s\boldsymbol{\theta}_y + \bar{\boldsymbol{\mu}}_{\boldsymbol{s}} \\ \hat{\Sigma}_s(\boldsymbol{\theta}_y) =& D_s(\boldsymbol{\theta}_y)\bar{\Sigma}_s D_s(\boldsymbol{\theta}_y) \quad \text{with } D_s(\boldsymbol{\theta}_y) = diag(exp(U_s\boldsymbol{\theta}_y + \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}))\end{aligned}$$

where $\bar{\boldsymbol{\mu}}_{\boldsymbol{s}}$ and $\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}$ are the biases of respectively the mean and covariance parameterizations.

Because the biases $\bar{\boldsymbol{\mu}}_{\boldsymbol{s}}$ and $\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}$ are common to all classes, the matrices V_s and U_s can be seen as only modeling the remaining differences between the classes with respect to a tied state.

Actually the biases are learnt with the data of all classes while the coefficients of parameterizations for the i^{th} class (the i^{th} row of V_s and U_s) are only learnt with the data of class i. Although it looks like a limited sharing of parameters it still explains that we get better estimates than learning isolated HMM per class, which is a promising and encouraging result.

7.1.4 Using a distributed representation of class as contextual variables

As we just said above the use of a one-hot class code, although it seems effective, does allow a limited sharing of parameters between classes. A more interesting idea is to design distributed class codes making the sharing of parameters much more important.

We investigated this idea by designing class codes $\boldsymbol{\theta}_y$ that are again $|Y|$-dimensional vectors but that reflects similarities between classes. Our implementation consists in designing codes such that the component at position y' of the class code of class y, $\boldsymbol{\theta}_y$, is proportional to the similarity between class y and y'.

Actually, using such a similarity based coding of the classes as a contextual variables defines a special of parameterization of the isolated class models. The more the class y is similar to the class y', the more the models of class y and y' will share parameters (i.e. and will be learned using data from the other class). Assume for instance a given similarity matrix on 3 classes as below.

class	1	2	3
1	1	0.8	0.2
2	0.8	1	0.1
3	0.2	0.1	1

TABLE 7.2: Similarity matrix on 3 classes, class 1 and 2 are close two each others and more distant to class 3

Assuming the observations are reals (and not real-valued vectors as usual), we illustrate the effect of using a denser code on the mean parameterization. The mean of a Gaussian distribution in a particular state can be written for the three class HMM models according to:

$$\hat{\mu}(\boldsymbol{\theta}_1) = w_1 + 0.8 \times w_2 + 0.2 \times w_3 + \bar{\mu}$$
$$\hat{\mu}(\boldsymbol{\theta}_2) = 0.8 \times w_1 + w_2 + 0.1 \times w_3 + \bar{\mu}$$
$$\hat{\mu}(\boldsymbol{\theta}_3) = 0.2 \times w_1 + 0.1 \times w_2 + w_3 + \bar{\mu}$$

where w_1, w_2 and w_3 are the mean parameterization weights. One can see that the mean of all 3 classes share the same parameters, hence all parameters here are learnt with the data of all classes. It is only the contextual variable $\boldsymbol{\theta}$ that will tend to define closer Gaussian means for similar classes.

At the end, if two classes are similar, their class codes will be close and will induce similar effects on the global CHMM. This can be seen as imposing a prior on the estimation of the parameters of each class model. Here this prior may be interesting because there is a lack of examples at training time, however when there are sufficient examples for each class, it may decrease performance.

We investigated this coding scheme by defining a class similarity from the confusion matrix of a standard HMM system (left-right, 8 states, 1 full Gaussian). The idea is that the more two classes confuses, the more similar they are. Here is the HMM confusion matrix.

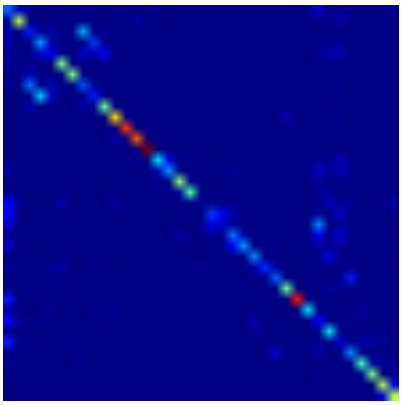


FIGURE 7.6: Confusion matrix of 8 states 1 Gaussian HMM on the 37 gesture classes. Each line and row of the matrix represent the 37 gesture classes in the same order. Brighter color off the diagonal means higher confusion

We define the class similarity matrix ("classSim") as the symmetrized confusion matrix of a standard HMM trained on the same task.

$$classSim = confusMat + confusMat^T$$

We also row-normalize classSim so that coefficients are between 0 and 1. At the end, the i^{th} row of classSim matrix is the contextual variable for class i and expresses its proximity to every other class.

model	nb states	nb gauss	Train average F1	Test average F1
$\mu\Sigma$CHMM (hcode)	8	1	91.4(1.2)	67.4(2.9)
$\mu\Sigma$CHMM (hcode)	8	5	97.1(0.6)	70.9(3.3)
$\mu\Sigma$CHMM (classSim)	8	1	89.5(0.9)	71.8(2.7)
$\mu\Sigma$CHMM (classSim)	8	5	96.5(0.7)	72.6(4.2)

TABLE 7.3: F1 accuracy of global CHMMs parameterized by a one hot code (hcode) or class similarity (classSim) type contextual variable

Results of table 7.3 show notable differences between a class hot code and class similarity based contextual variables. Using classSim type contextual variable produces better generalization whereas the training accuracy is a little bit inferior. These results are encouraging, as it shows that a change in the class code representation can encode useful information during learning for a better generalization. Although we did not investigate this up to now, it would be interesting to use directly the confusion matrix from the Contextual Model (with hcode contextual variable) instead of the confusion matrix from the HMM. This would maybe help define a better classSim contextual variable.

7.1.5 Retraining discriminatively

The global model we just detailed in previous sections seems better at classifying gestures than separate HMMs on a small dataset. However, it is a generative model and we could consider (but did not experimented here) retraining it in a discriminative way.

As explained in chapter 4, we can cast CHMMs into their discriminative counter part the Contextual HCRFs. Then, one can retrain all the class CHCRF models using the discriminative criterion exposed in 4.3.3.

In this method, $|Y|$ CHCRFs models are carefully initialized to reproduce the decision functions of the $|Y|$ individual classes CHMMs. Even if here we have trained only one global CHMM on all data, we can easily rebuild individual CHMMs for each class. The $|Y|$ classes CHMMs will in fact have the same parameters as the global CHMM, yet, they have a different contextual variable $\boldsymbol{\theta}_y$ assigned to their class y. Hence, when these individual CHMMs are cast into their discriminative counter part, the resulting $|Y|$ CHCRFs actually model different decision functions for each class.

Altogether, the whole learning process can be interpreted as a transfer Learning approach. In a first step, reliable probability estimates are extracted from much more data (all classes) using a generative global model. Then, using this knowledge, one can restore individual class models for recognition. Eventually, one could perform a final step of discriminative training as we described by starting the optimization from meaningful estimates.

7.2 Dynamic Factor Graphs

A continuous state space model might be more appropriate than a discrete state space model for the Transfer Learning approach we described. This motivated us to investigate the use of dynamical Factor Graphs (DFG). We first discuss why a continuous state space would be interesting here then we detail why and how DFG could be a relevant solution to the problem we consider here.

7.2.1 Continuous state space models

Very often the data that we model belong to a low dimensional manifold so that these data, and their dynamics could be well explained in a smooth continuous lower embedding. We first illustrate this concept on the "helix" artificial dataset [78]. In the "helix"

dataset, points are defined by the following 3 dimensional time series (see Figure 7.5):

$$x_t(1) = (2 + 2cos(8 \times 2\pi \times t)) \times cos(2\pi \times t)$$
$$x_t(2) = (2 + 2sin(8 \times 2\pi \times t)) \times sin(2\pi \times t)$$
$$x_t(3) = sin(8 \times 2\pi \times t)$$

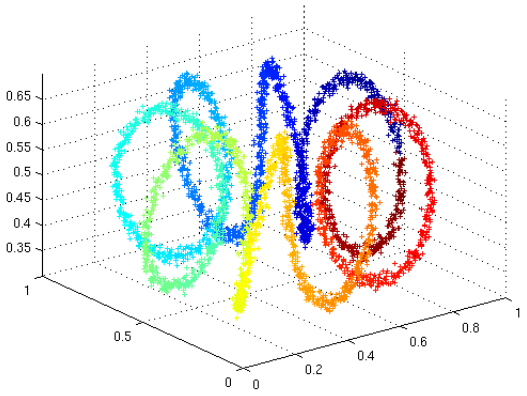


FIGURE 7.7: Helix dataset (1000 points, t evenly spaced in $[0, 1]$). Each point $\boldsymbol{x}_t$ is further noised (Gaussian with variance 0.05) and plotted in a different color for clarity

It is clear that the structure of these 3D data points lie on an helical manifold. In figure 7.8 we performed a dimensionality reduction on 2D space with Gaussian Process Dynamical Models (GPDM). The second figure 7.9 is a 2D projection using an Autoencoder, a special type of Neural Network architecture aimed at reducing efficiently the dimensionality of the data [37] (For this architecture we used 3 stacked layers of RBMs with 80 neurons for the two first hidden layers and a final linear layer with two neurons).

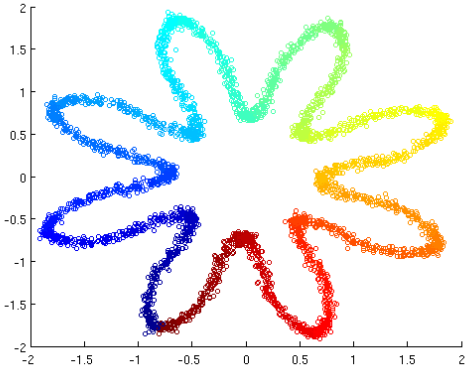


FIGURE 7.8: 2D projections of helix data points with GPDM

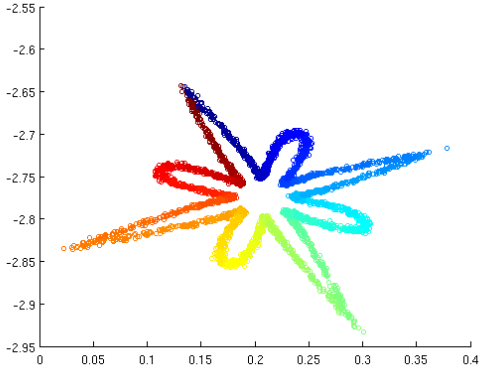


FIGURE 7.9: 2D projections helix data points with Autoencoder RBM.

As one can expect, both projection techniques (GPDM or Autoencoders) manage to unfold the 3D structure of the data into a smooth 2D latent space. The neighborhood of data points seems particularly well preserved in both cases.

In figure 7.10 and 7.11, we use a more realistic dataset. We computed two dimensionality reductions of 10 walk gestures from CMU motion capture database ([32]) performed by a single actor. Original data is composed as before of 62 joints angles. We abruptly project these training observations sequences in 2D for ease of visualization.

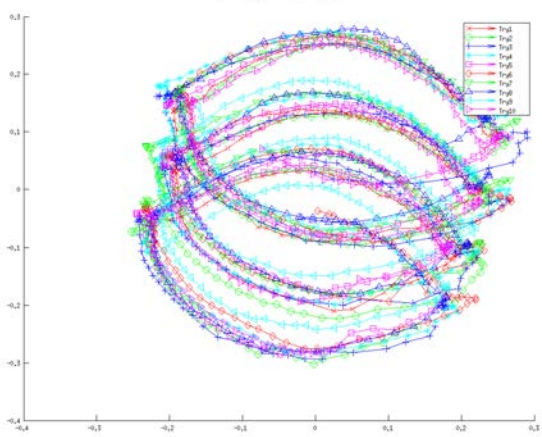


FIGURE 7.10: 2D projections of 10 Walks gestures with PCA

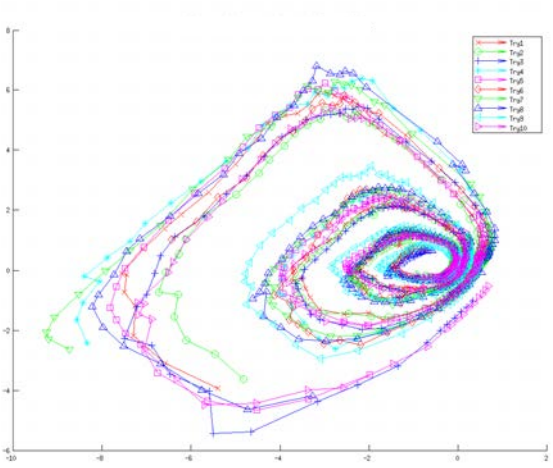


FIGURE 7.11: 2D projections of 10 walks gestures with Autoencoder RBM (same architecture as in helix dataset)

First image is the projection of the 10 walk gestures by principal component analysis (PCA). Mean square error between the original data and its reconstruction from the 2D space is 0.042. Second image is the 2D projection of the same gesture by means of an Autoencoder. Final mean square reconstruction error after finetuning the network to reproduce original data is 0.0081, hence far lower than for PCA.

Of course the variability is much contained because there are few gestures performed by a single actor. Anyhow, in such an extreme example, we can reconstruct the original data pretty well from as low as 2 dimensions. Furthermore, the trajectory of the latent space is relatively smooth and predictable, thus giving credits to using a model of the dynamics in a low dimensional embedding.

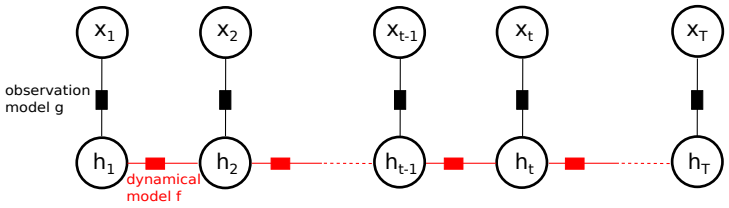
This discussion shows that most often the data we consider may lie in a low dimensional space where modeling their dynamics could be done with much less parameters, hence much less training data. For instance it is obvious that although we gather and model 62 dimensional data in motion capture applications we don't have actually 62 degrees of freedom. The modeling of such data could then be divided in first learning the low dimensional embedding which probably shares similarities between classes and then,

learning the dynamics in this low dimensional space where there is less (but still some) to share between classes.

7.2.2 Analogy with Dynamic Factor Graphs

The dynamical Factor graph [58] is an ideal candidate to implement our Transfer Learning approach.

In its simplest form, it look alike a Hidden Markov Model with a chain structure :



Yet DFGs can model the data from a continuous latent embedding by the use of two functions :

One function g responsible for emitting the observations from the latent space

$$\boldsymbol{x}_t = g(W_o, \boldsymbol{h}_t)$$

And one function f predicting the dynamics in latent space

$$\boldsymbol{h}_t = f(W_d, \boldsymbol{h}_{t-p}^{t-1}, \boldsymbol{x}_{t-p}^{t-1})$$

where W_o, and W_d are respectively the parameters of the observation and dynamical models.

A complete description of this model goes beyond the scope of this thesis. However training Dynamical Factor graphs consists in optimizing over the parameters $W = [W_o, W_d]$ *and* over the latent representation $\boldsymbol{h}$ such as to minimize :

$$\mathcal{L}_{DFG} = \sum_t \alpha ||\boldsymbol{x}_t - g(W_o, \boldsymbol{h}_t)||_2^2 + \beta ||\boldsymbol{h}_t - f(W_d, \boldsymbol{h}_{t-p}^{t-1}, \boldsymbol{x}_{t-p}^{t-1})||_2^2 \tag{7.2}$$

where α and β are two scalars controlling the trade-off between the observation and dynamical models. More details about the model and its EM-like learning and inference algorithms can be found in [57].

Following the discussion above, one could think that maybe the function g could be learned from all classes' data while the function f would more be class specific. This

indicates some possibilities for transfer learning and learning new classes from few samples.

7.2.2.1 A global Dynamical Factor Graph

As f and g can be any complex function, we could train a DFG with a high capacity function g such as an Autoencoder and eventually a less complex function f as a linear model for the dynamics. In such a setting, we think DFG could be employed as an efficient global model as we have done with Contextual models.

As before, this global model would be trained on all class data, but note here, we don't need to define a class-code contextual variable, hence we can potentially use unsupervised data from unknown classes to improve the global model.

We used the framework of Contextual models with class contextual variables as a way to efficiently share information between classes while maintaining a low number of discrete states in the global model. This limited the problem associated with HMM with a high number of states, such as overfitting and complexity (quadratic in the number of states).

Overfitting is less a risk with DFGs. First because DFGs can potentially use a low embedding, secondly because they share their parameters natively between hidden states, and finally because they can use unsupervised data. Lastly, the complexity is linear with respect to the number of parameters of the functions f and g which can be controlled at will.

7.2.2.2 Retraining DFG discriminatively

To be able to perform a classification task with these kind of models has not been explained. In the work of [58], they use DFGs for time series forecasting and synthesis but not for classification purpose. Yet, there is a simple way to train Dynamical Factor Graph discriminatively for classification tasks, which is analogous to what is done in HCRFs. We explain it now.

Let there be $|Y|$ Dynamical Factor Graphs, one per class y with its own parameters set $W^y = \left[W_o^y, W_d^y\right]$ for observation and dynamic factors. One DFG can be trained independently to maximize the likelihood of all examples of its class y (or equivalently minimize the loss as in Eq 7.2) with respect to its parameters W^y. Meanwhile, the $(|Y| - 1)$ DFGs modeling the remaining classes could be trained on the same data but to minimize its likelihood.

As a result, for an observation sequence $\boldsymbol{x}_y^k$ belonging to class y, the discriminative loss $\mathcal{L}_{DDFG}$ we propose to minimize becomes :

$$\mathcal{L}_{DDFG}(W, \boldsymbol{x}_y^k, \boldsymbol{h}^k) = \mathcal{L}_{DFG}(W_y, \boldsymbol{x}_y^k, \boldsymbol{h}_y^k) - \sum_{y' \neq y} \mathcal{L}_{DFG}(W_{y'}, \boldsymbol{x}_y^k, \boldsymbol{h}_{y'}^k) \tag{7.3}$$

with $\mathcal{L}_{DFG}$ the generative loss for training a DFG exposed in Eq 7.2
$\boldsymbol{h}_y^k = \text{argmin}_{\boldsymbol{h}_y^k}(\mathcal{L}_{DFG}(W_y, \boldsymbol{x}^{\boldsymbol{k}}, \boldsymbol{h}_y^k))$
and W the parameters set composed of all the $|Y|$ DFG parameters.

Ultimately, the whole transfer learning algorithm for training Dynamical Factor Graphs on few examples would be composed of the following steps:

- Train a single global DFG with parameters W_g on every class data (and unsupervised data if any)
- Build $|Y|$ class DFGs and initialize their parameters W_y with the parameters of the global model W_g
- Retrain the individual DFGs discriminatively using the loss 7.3

We can not show proper results concerning this method as we began to investigate it at the very end of this Thesis. Setting appropriately the model functions topology (f and g) and gradient steps has revealed critical to reduce both training and inference time. Starting from a generative implementation of DFGs gently provided by Piotr Mirowski, we implemented the discriminative training described above but without sufficient time to finetune hyper-parameters, we did not achieved the convergence speed required to run full experiments.

This is only a prospective work that could have been presented as a perspective. However, exposed here, one may understand a clear connection with the Transfer Learning approach described on contextual models and be motivated to go a step further.

7.3 Conclusion

In this chapter we have devised a transfer learning approach to learn contextual models from few examples. The idea is based on first learning a global Contextual model jointly on the data from all classes.

In this respect we propose to use class-code type contextual variables to share information between the classes. In a second step, one can transfer the knowledge of the global model to individual classes models, to do decoding.

This strategy revealed effective in our classification experiments on motion capture data. We achieved noticeable gains compared to training separate HMMs for each class with various topologies. Our experiments show that a global CHMM make a better use of its parameter set.

To go a step further toward efficient transfer learning, we advocate the use of continuous instead of discrete state space models. The analogy between the structure of Hidden Markov Models and Dynamical Factor Graphs make them ideal candidates to implement a similar approach.

Chapter 8

Conclusion & Perspectives

Through this thesis, we devised the framework of Contextual Hidden Markov Models for a better modeling of time series. The starting point of our proposal is that an important part of variability between observation sequences may be the consequence of a few contextual variables (which may be hidden or observed), that remain fixed all along a sequence or that vary slowly with time. We have shown for example that the rate of speech or the signal noise ratio can explain an important part of the variation in speaker utterances. Nonetheless, we have also seen that simple variables like the short term mean of the observations sequence can also be useful.

Various declinations around this framework were proposed to incorporate the influence of contextual variables into our modeling. We have shown several ways to implement this idea into Hidden Markov Models by extending the mean parameterized HMMs to other types of parameterization. We proposed a full covariance parameterization of the state's distributions with dynamic contextual variables, and, subsequently proposed a parameterization of their transition probabilities.

We then introduced a natural and efficient way to exploit contextual information into a discriminative model. First generative CHMMs are trained via a maximum likelihood, which can be done in parallel for each class model. Then, one can retrain the models for a few iterations by casting a CHMM into its discriminative counter part, the Contextual HCRF. At the end, CHCRF can be viewed as an efficient way to learn a HCRF that exploit contextual information compared to more standard ways of learning such models.

We applied several types of contextual models for the classification of handwritten characters, speech recognition, or synthesis of realistic eyebrow motions. We investigated

different types of contextual variables and parameterizations for each one of these domains. For all theses tasks, we have demonstrated that the better modeling ability of CHMMs can translate into performance gains.

Though, we revealed that the performance gap between mean and covariance parameterization may be limited, and that contextual variables do not always combine well. Part of our work, was to investigate strategies to lessen this effect with regularization or a multistream combination of contextual variables.

Developing on the idea to learn more expressive models with fewer examples, we finally envisioned a Transfer Learning approach using Contextual HMMs. The method relies on learning a global Contextual model which effectively share information between the classes by using a special type of contextual variable. This "class-code" contextual variable effectively encodes useful information during learning. Our results on a gesture classification task show that this simple scheme helps in achieving a better generalization compared to classic HMMs tested on various topologies, and, more importantly, even when the CHMMs have more parameters than HMMs.

A future research would be to continue working in that direction, and leverage possible transfer learning approaches like the one we sketched in the last chapter. A continuous lower embedding appear to be a relevant hypothesis in the task we addressed. Hence it seems a good idea to use the capability of DFGs to model the dynamics of the data in such a space.

Another interesting direction may focus on designing better contextual variables as it is an important part in the sucess of CHMMs. We have seen how some simple variables like the short term mean of the signal can help in a handwriting classification task. We have also seen how more domain specific variables like measures of the signal to noise ratio or the rate of speech can help in speech recognition. Yet, it would be valuable to investigate new types of contextual variables more dedicated to their domain. We think for example to the field of gesture recognition, where one can imagine using contextual variables such as gender, corpulence or any other characteristics ... The main problem until recently was the lack of clean and richly annotated gesture data. However, multi-modal gesture data now becomes increasingly available through open challenges in gesture recognition (e.g "ChaLearn" [26]).

A last idea would be to combine CHMM with context dependent modeling, a strategy better suited at handling the variability of transitional effects. In speech, it is known as the coarticulation effect, the realization of a phone depends on the previous and the next phone. In handwriting, it is called ligature, the shape of a letter is influenced by the previous and the next one. For a better handling of transitional effects, triphone

and trigraph models are employed instead of simple phones or characters. However, an important part of intra signal variability still remains inside triphones and trigraphs. Such a variability might be captured more efficiently by contextual models.

Appendices

A Usixml Project

The project Usixml is the effort of several academic and industrial partners (full list available at http://usixml.eu/view-all-partners) in order to define, validate and standardize an Open User Interface Description Language (UIDL).

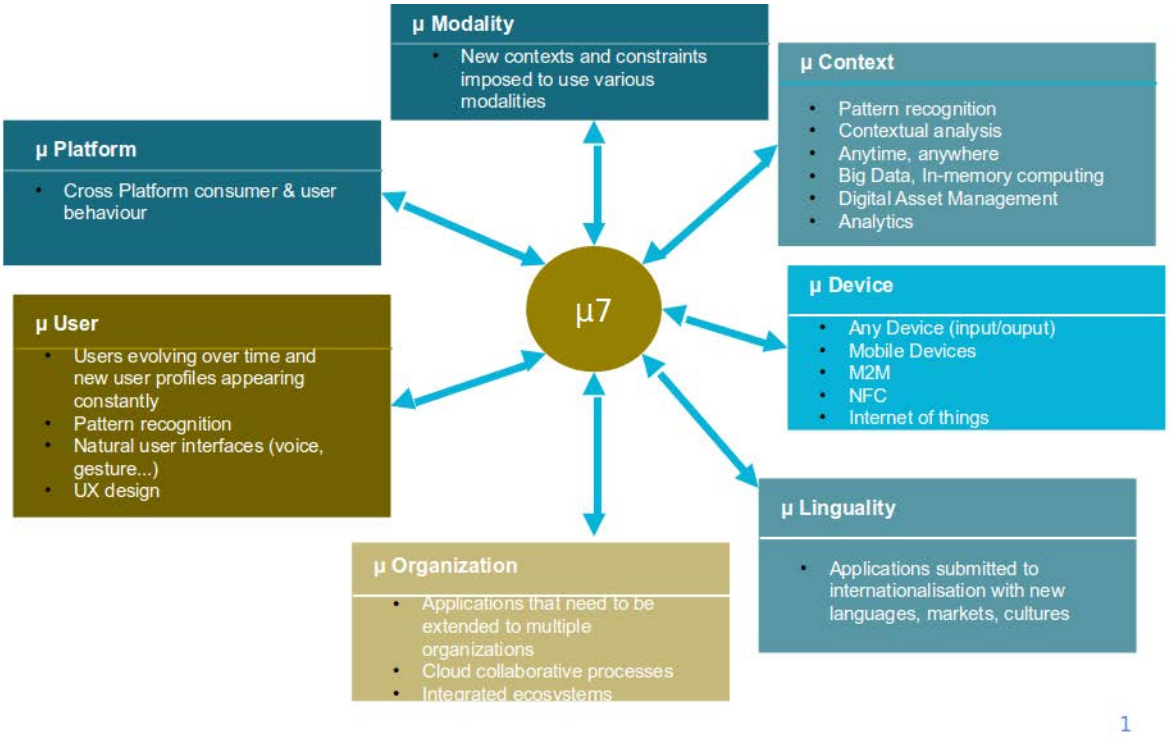
Currently, the development of the User Interface (UI) of interactive applications is very difficult because of the complexity and the diversity of existing development environments and the high amount of programming skills required by the developer to reach a usable UI: markup languages (e.g., HTML), programming languages (e.g., C++ or Java), development skills for communication, skills for usability engineering.

These difficulties are exacerbated when the same UI should be developed for multiple contexts of use such as multiple categories of users (e.g., having different preferences, speaking different native languages, potentially suffering from disabilities), different computing platforms (e.g., a mobile phone, a Pocket PC, an interactive kiosk, a laptop, a wall screen), andvarious working environments (e.g., stationary, mobile).

Although designers and programmers are involved in these types of project, the available tools are mainly target at the developer. Therefore, it is rather difficult for a designer to design a UI for multiple contexts of use while avoiding to reproduce multiple UIs for multiple contexts of use. This work proposes a way to separate responsabilities in these types of projects.

UsiXML (which stands for USer Interface eXtensible Markup Language) is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. In other words, interactive applications with

different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform.



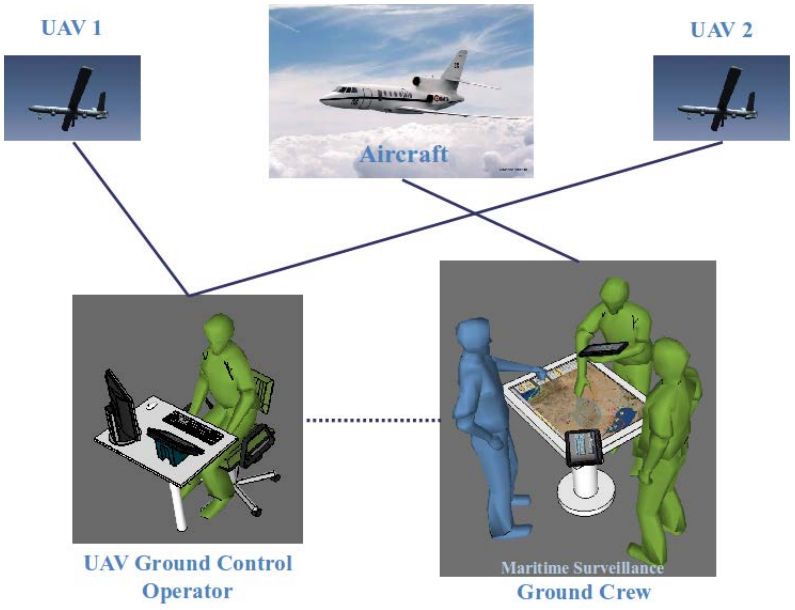
- UsiXML consists of a User Interface Description Language (UIDL), that is a declarative language capturing the essence of what a UI is or should be independently of physical characteristics.
- UsiXML supports device independance: a UI can be described in a way that remains autonomous with respect to the devices used in the interactions such as mouse, screen, keyboard, voice recognition system,... In case of need, a reference to a particular device can be incorporated.
- UsiXML supports platform independance: a UI can be described in a way that remains autonomous with respect to the various computing platforms, such as mobile phone, Pocket PC, Tablet PC, laptop, desktop,... In case of need, a reference to a particular computing platform can be incorporated.
- UsiXML supports modality independance: a UI can be described in a way that remains independent of any interaction modality such as graphical interaction, vocal interaction, 3D interaction, or haptics. In case of need, a reference to a particular modality can be incorporated.

- UsiXML supports other sets of abstraction layers to facilitate the integration of user profiling, extensions of an UI to multiple organisations or its portability to other languages.

B Pattern Recognition in Usixml

The UsiXML project contours is far beyond the scope of this thesis. We discuss here one of the goal of UsiXML which is to provide natural and multi-modal human interaction with computers where modalities might be speech, gesture or any sequential data provided by a human machine interface. In this context, the algorithms and statistical models we presented in this thesis have been employed in a first demonstrator of a rich 3D User Interface following the standards of Usixml.

The User Interface we describe now has been developped with the effort of several laboratories (Ecole Navale, Télécom Bretagne, Thales Airbone Systems and Thales Underwater Systems). This interface helps to manage and coordinate maritime surveillance operations involving several actors which might be for instance Unmanned Aerial Vehicules (UAV), Aircrafts, or ground crew operators.



B.1 SurMap Prototype

The demonstrator called SurMap is composed of an haptic tablet which displays a map and all necessary informations to conduct and coordinate the surveillance operations. Several ground crew operators can cooperate around this interface to follow the course of operations and add relevant informations in real time.

Our role in this demonstrator has been to integrate gesture recognition capability to this interface to enable richer human machine interaction. For the specific need of gesture recognition, the User Interface system includes a depth sensor (i.e. a kinect camera) which records the crew operations on the tablet from an upper position as illustrated in following picture.

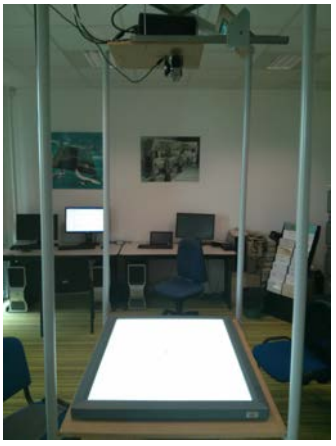


FIGURE B.1: SurMap prototype with depth sensor for 3D gesture recognition

From the information acquired by the depth sensor, a tracking algorithm extracts the positions of hand fingers required to recognize several gestures. The Figure B.2 shows the tracking algorithm at work.

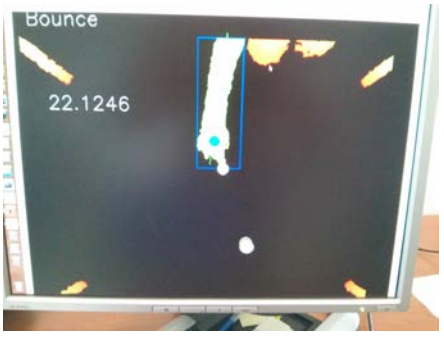


FIGURE B.2: Tracking the position of fingers with the depth sensor

B.2 Gesture definition

Several gestures have been proposed to enable a rich and natural human machine interaction. We describe them below.

- Enter draw mode : triggered by drawing a polygon in the air simultaneously with both hands. Each hand draw one half of the polygon.
- Stop mode : triggered by positionning the left hand at vertical and the right hand perpendicularly. The right hand fingers must point at the left hand palm.
- Non validation : Cross both hands with an X shape.
- Closing : Shake both hands.
- Rotation mode : The index finger of the left hand is straight while the right hand index finger turns around it. The position of the left finger indicates the axis of rotation. Different angles are allowed (quarter, half or three quarter rotations)
- Zoom mode : Both hands are flat next to each other, than the right hand goes up propotionnally to the zoom desired. Several zoom amplitudes are allowed.
- Center (on selected object) : Draw a double circle with the right hand index finger around the left hand.

FIGURE B.3: Example of rotation gesture around the Z axis

B.3 Gesture recognition system

For 3D gesture recognition, the demonstrator employs the Contextual HMM (CHMM) described in this thesis. The UsiXML user interface is connected to the core recognizer (CHMM) in a very simple way illustrated in figure B.4.

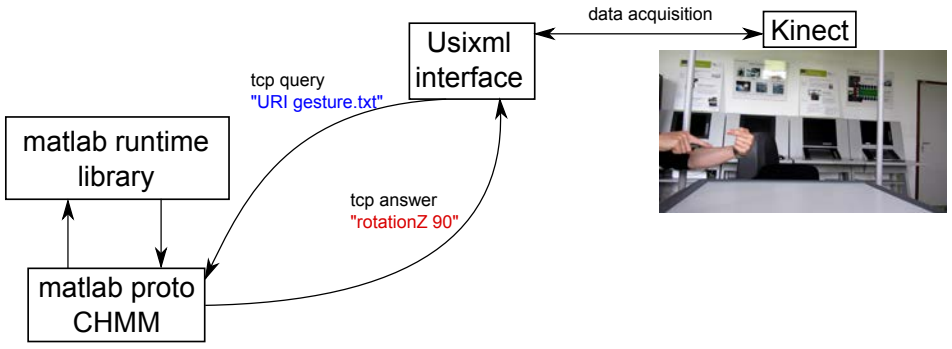


FIGURE B.4: Recognizer integration

The user interface is plugged to the depth sensors and provides finger positions to the core recognizer for recognition. The beginning and end of gestures are triggered by the user himself pushing a pedal placed on the ground which is connected to the computer hosting the interface system.

When a gesture is finished, the interface system queries the recognizer for an answer. The recognizer which is here a matlab standalone application than process this request and answers which gesture has been recognized to the interface.

More specifically, the matlab application actually hosts a TCP (Transmission Control Protocol) server and listens for a query coming from the user interface. When receiving a gesture, the user interface initiates a query which is a TCP message containing the Uniform Ressource Identifier (URI) of a gesture file. This gesture file contains the finger positions extracted by the tracking algorithm. In response to this message, the recognizer answers a TCP message to the interface containing which gesture has been recognized.

The Contextual HMM which defines the core gesture recognizer has been employed with very simple intra signal contextual variables for characterizing gestures (i.e. short term mean of the observation sequence or total length of the gesture). We do not provide experiments on the database used for the prototype as the lack of gestures examples and finger tracking problems would introduce an important bias in comparing different algorithms. This is why we performed experiments on a cleaner gesture dataset in chapter 7 and also a reason why we subsequently proposed a training strategy to learn contextual models from few examples.

Bibliography

[1] Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 49 – 52.

[2] Barker, J., Vincent, E., Ma, N., Christensen, H., and Green, P. (2013). The {PASCAL} {CHiME} speech separation and recognition challenge. *Computer Speech & Language*, 27(3):621 – 633. Special Issue on Speech Separation and Recognition in Multisource Environments.

[3] Bellegarda, J. and Nahamoo, D. (1990). Tied mixture continuous parameter modeling for speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(12):2033 –2045.

[4] Bellegarda, J. R. (2004). Statistical language model adaptation: review and perspectives. *Speech Communication*, 42:93–108.

[5] Bengio, Y. and Frasconi, P. (1996). Input/output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7:1231–1249.

[6] Bianne-Bernard, A.-L., Fares, M., Likforman-Sulem, L., Mokbel, C., and Kermorvant, C. (2012). Variable length and context-dependent hmm letter form models for arabic handwritten word recognition. pages 829708–829708–8.

[7] Bilmes, J. (1998). A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report.

[8] Boersma, P. and Weenink, D. (2001). Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345.

[9] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription.

[10] Brand, M. (1999). Voice puppetry. pages 21–28.

[11] Busso, C., Deng, Z., Grimm, M., Neumann, U., and Narayanan, S. (2007). Rigid head motion in expressive speech animation: Analysis and synthesis. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1075–1086.

[12] Busso, C., Deng, Z., Neumann, U., and Narayanan, S. (2005). Natural head motion synthesis driven by acoustic prosodic features: Virtual humans and social agents. *Comput. Animat. Virtual Worlds*, 16(3-4):283–290.

[13] Cheng, C.-C., Sha, F., and Saul, L. K. (2009). Matrix updates for perceptron training of continuous density hidden markov models. In *ICML*, page 20.

[14] Chiu, C.-C. and Marsella, S. C. (2011). How to train your avatar: A data driven approach to gesture generation. In *The 11th International Conference on Intelligent Virtual Agents (IVA 2011)*, ReykjavAk, Iceland.

[15] Christensen, H., Barker, J., Ma, N., and Green, P. (2010). The chime corpus: a resource and a challenge for computational hearing in multisource environments. In *in Proc. Interspeech́10 Makuhari*.

[16] Costa, M., Chen, T., and Lavagetto, F. (2001). Visual prosody analysis for realistic motion synthesis of 3d head models. In *In: Proc. of ICAV3D01 - International Conference on Augmented, Virtual Environments and 3D Imaging*, pages 343–346.

[17] Cui, X. and Gong, Y. Variable parameter gaussian mixture hidden markov modeling for speech recognition. In *ICASSP '03*.

[18] Cui, X. and Gong, Y. (2007). A study of variable-parameter gaussian mixture hidden markov modeling for noisy speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4):1366–1376.

[19] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[20] Deng, L. (1992). A generalized hidden markov model with state-conditioned trend functions of time for the speech signal. *Signal Process.*, 27(1):65–78.

[21] Deng, L., Aksmanovic, M., Sun, X., and Wu, C. (1994). Speech recognition using hidden markov models with polynomial regression functions as nonstationary states. *Speech and Audio Processing, IEEE Transactions on*, 2(4):507–520.

[22] Do, T. and Artières, T. (2005). Conditional random field for tracking user behavior based on his eye's movements. In *Workshop at NIPS 2005, in Whistler, BC, Canada, on December 10, 2005.*, page 19. Citeseer.

[23] Do, T.-M.-T. and Artières, T. (2006). Conditional random fields for online handwriting recognition. *Adv in Neur Inf Proc Sys*, 17:1097–1104.

[24] Do, T.-M.-T. and Artières, T. (2009). Large margin training for hidden Markov models with partially observed states. In Bottou, L. and Littman, M., editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 265–272, Montreal. Omnipress.

[25] Ekman, P. and Friesen, W. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto.

[26] Escalera, S., Gonzalez, J., Baro, X., Reyes, M., Guyon, I., Athitsos, V., Escalante, H. J., Sigal, L., Argyros, A., Sminchisescu, C., Bowden, R., and Sclaroff, S. (2013). Chalearn multi-modal gesture recognition 2013: grand challenge and workshop summary. In Epps, J., Chen, F., Oviatt, S., Mase, K., Sears, A., Jokinen, K., and Schuller, B., editors, *ICMI*, pages 365–368. ACM.

[27] Fanelli, G., Gall, J., Romsdorfer, H., Weise, T., and Gool, L. V. (2010). A 3-d audio-visual corpus of affective communication. *IEEE Transactions on Multimedia*, 12(6):591 – 598.

[28] Fujii, Y., Yamamoto, K., and Nakagawa, S. (2011). Automatic speech recognition using hidden conditional neural fields. In *ICASSP*, pages 5036–5039.

[29] Fujinaga, K., Nakai, M., Shimodaira, H., and Sagayama, S. (2001). Multiple-regression hidden markov model. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 1, pages 513 –516 vol.1.

[30] Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(1):52–59.

[31] Gong, Y. (1995). Speech recognition in noisy environments: A survey. *Speech Communication*, 16(3):261–291.

[32] Graphics Lab, C.-M. U. Carnegie-mellon university motion capture database (http://mocap.cs.cmu.edu/).

[33] Graves, A., Jaitly, N., and Mohamed, A.-R. (2013). Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278.

[34] Gunawardana, A., Mahajan, M., Acero, A., and Platt, J. C. (2005). Hidden conditional random fields for phone classification. In *in Interspeech*, pages 1117–1120.

[35] Hammersley, J. M. and Clifford, P. (1971). Markov field on finite graphs and lattices.

[36] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012a). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

[37] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507.

[38] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

[39] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[40] Hofer, G. (2007). Yamagishi j.: Speech driven head motion synthesis based on a trajectory model. In *in Proc. SIGGRAPH*.

[41] Hwang, M.-Y. and Huang, X. (1993). Shared-distribution hidden markov models for speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 1(4):414 –420.

[42] Jaeger, H. (2002). Tutorial on training recurrent neural networks. page 48 pp.

[43] Jiang, H. (2010). Discriminative training of {HMMs} for automatic speech recognition: A survey. *Computer Speech & Language*, 24(4):589–608.

[44] Juang, B.-H. and Katagiri, S. (1992). Discriminative learning for minimum error classification [pattern recognition]. *Signal Processing, IEEE Transactions on*, 40(12):3043 –3054.

[45] Just, A. and Marcel, S. (2009). A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. *Comput. Vis. Image Underst.*, 113(4):532–543.

[46] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[47] Le, B. H., Ma, X., and Deng, Z. (2012). Live speech driven head-and-eye motion generators. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1902–1914.

[48] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. In Bakir, G., Hofman, T., Schölkopf, B., Smola, A., and Taskar, B., editors, *Predicting Structured Data*. MIT Press.

[49] Lee, K.-F. and Hon, H.-W. (1989). Speaker-independent phone recognition using hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(11):1641 –1648.

[50] Levine, S., Krähenbühl, P., Thrun, S., and Koltun, V. (2010). Gesture controllers. *ACM Trans. Graph.*, 29(4).

[51] Levine, S., Theobalt, C., and Koltun, V. (2009). Real-time prosody-driven synthesis of body language. *ACM Trans. Graph.*, 28(5):172:1–172:10.

[52] Li, Y. and Shum, H.-Y. (2006). Learning dynamic audio-visual mapping with input-output hidden markov models. *Multimedia, IEEE Transactions on*, 8(3):542–549.

[53] Ljolje, A. (1994). High accuracy phone recognition using context clustering and quasi-triphonic models. *Computer Speech & Language*, 8(2):129–151.

[54] Mahajan, M., Gunawardana, A., and Acero, A. (2006). Training Algorithms for Hidden Conditional Random Fields. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1.

[55] Mariooryad, S. and Busso, C. (2012). Generating human-like behaviors using joint, speech-driven models for conversational agents. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(8):2329–2340.

[56] McDermott, E., Watanabe, S., and Nakamura, A. (2009). Margin-space integration of mpe loss via differencing of mmi functionals for generalized error-weighted discriminative training. In *INTERSPEECH*, pages 224–227.

[57] Mirowski, P. (2011). *Time Series Modeling with Hidden Variables and Gradient-based Algorithms*. PhD thesis, New York, NY, USA. AAI3445313.

[58] Mirowski, P. and Lecun, Y. (2009). Dynamic factor graphs for time series modeling. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 128–143, Berlin, Heidelberg. Springer-Verlag.

[59] Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., and Weber, A. (2007). Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn.

[60] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of Uncertainty in AI*, pages 467–475.

[61] Nopsuwanchai, R. (2005). Discriminative training methods and their applications to handwriting recognition. Technical report, University of Cambridge, computer laboratory.

[62] Pandzic, I. S. and Forchheimer, R., editors (2003). *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, NY, USA.

[63] Paul, D. (1991). The lincoln tied-mixture hmm continuous speech recognizer. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 329 –332 vol.1.

[64] Povey, D. and Woodland, P. (2002). Minimum phone error and i-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–105 –I–108.

[65] Quattoni, A., Wang, S., p Morency, L., Collins, M., Darrell, T., and Csail, M. (2007). Hidden-state conditional random fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[66] Rabiner, L. and Juang, B.-H. (Prentice Hall, 1993). *Fundamentals of speech recognition*. Prentice Hall.

[67] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286.

[68] Reiter, S., Schuller, B., and Rigoll, G. (2007). Hidden conditional random fields for meeting segmentation. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 639–642. IEEE.

[69] Sargin, M., Yemez, Y., Erzin, E., and Tekalp, A. (2008). Analysis of head gesture and prosody patterns for prosody-driven head-gesture animation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1330–1345.

[70] Senin, P. (2008). Dynamic Time Warping Algorithm Review. Technical Report CSDL-08-04, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822.

[71] Sha, F. (2006). Large margin training of acoustic models for speech recognition. Doctoral dissertation.

[72] Sha, F. and Saul, L. K. (2007). Large margin hidden markov models for automatic speech recognition. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *NIPS 19*, pages 1249–1256. MIT Press.

[73] Srivastava, N. (2013). Improving neural networks with dropout (doctoral dissertation, university of toronto).

[74] Sung, Y.-H. and Jurafsky, D. (2009). Hidden conditional random fields for phone recognition. In *ASRU*, pages 107–112.

[75] Sutton, C. and Mccallum, A. (2007). An introduction to conditional random fields for relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to Statistical Relational Learning*.

[76] Tokuda, K., Kobayashi, T., Masuko, T., Kobayashi, T., and Kitamura, T. (2000). Speech parameter generation algorithms for hmm-based speech synthesis. In *Proc. ICASSP*, pages 1315–1318.

[77] U.-V., M. and H., B. (1999). A full english sentence database for off-line handwriting recognition. In *International Conference on Document Analysis and Recognition, 1999*.

[78] Van der Maaten, L., Postma, E., and Van den Herik, H. (2009). Dimensionality reduction: A comparative review. *Technical Report TiCC TR 2009-005*.

[79] Vincent, E., Barker, J., Watanabe, S., Le Roux, J., Nesta, F., and Matassoni, M. (2013). The second 'CHiME' Speech Separation and Recognition Challenge: Datasets, tasks and baselines. In *ICASSP - 38th International Conference on Acoustics, Speech, and Signal Processing - 2013*, pages 126–130, Vancouver, Canada.

[80] Vinel, A., Do, T. M. T., and Artières, T. (2011). Joint optimization of hidden conditional random fields and non linear feature extraction. In *ICDAR*, pages 513–517. IEEE.

[81] Wan, L., Zeiler, M. D., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1058–1066. JMLR.org.

[82] Wang, J. M., Fleet, D. J., Member, S., and Hertzmann, A. (2007). Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Machine Intell.*

[83] Wessel, F., Macherey, K., and Ney, H. (1999). A comparison of word graph and n-best list based confidence measures. In *in Proc. EUROSPEECH*, pages 315–318.

[84] Wilson, A. and Bobick, A. (1999). Parametric hidden markov models for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):884 –900.

[85] Xue, J. (2008). *Acoustically-driven Talking Face Animations Using Dynamic Bayesian Networks*. PhD thesis, Los Angeles, CA, USA. AAI3351613.

[86] Young, S. and Woodland, P. (1994). State clustering in hidden markov model-based continuous speech recognition. *Computer Speech & Language*, 8(4):369 – 383.

[87] Young, S. J., Evermann, G., Gales, M. J. F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. C. (2006). *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK.

[88] Yu, D. and Deng, L. (2007). Large-margin discriminative training of hidden markov models for speech recognition. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 429–438, Washington, DC, USA. IEEE Computer Society.

[89] Yu, D., Deng, L., Gong, Y., and Acero, A. (2009). A novel framework and training algorithm for variable-parameter hmms. *IEEE Trans. on Audio, Speech, and Language Processing*, 17(7):1348 –1360.

[90] Zen, H., Tokuda, K., and Kitamura, T. (2007). Reformulating the hmm as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech & Language*, 21(1):153–173.

Abstract

Modeling time series has practical applications in many domains : speech, gesture and handwriting recognition, synthesis of realistic character animations etc ... The starting point of our modeling is that an important part of the variability between observation sequences may be the consequence of a few contextual variables that remain fixed all along a sequence or that vary slowly with time. For instance a sentence may be uttered quite differently according to the speaker emotion, a gesture may have more amplitude depending on the height of the performer etc ... Such a variability cannot always be removed through preprocessing.

We first propose the generative framework of Contextual Hidden Markov Models (CHMM) to model directly the influence of contextual information on observation sequences by parameterizing the probability distributions of HMMs with static or dynamic contextual variables. We test various instances of this framework on classification of handwritten characters, speech recognition and synthesis of eyebrow motion from speech for a virtual avatar. For each of these tasks, we investigate in what extent such modeling can translate into performance gains.

We then introduce a natural and efficient way to exploit contextual information into Contextual Hidden Conditional Random Fields (CHCRF), the discriminative counter part of CHMMs. CHCRF may be viewed as an efficient way to learn a HCRF that exploit contextual information.

Finally, we propose a Transfer Learning approach to learn Contextual HMMs from fewer examples. This method relies on sharing information between classes where in generative models classes are normally considered independent.

Résumé

La modélisation de données séquentielles est utile à de nombreux domaines : reconnaissance de parole, de gestes, d'écriture, ou encore la synthèse d'animations pour des avatars virtuels. Notre modélisation part du constat qu'une part importante de la variabilité entre les séquences d'observations peut être la conséquence de quelques variables contextuelles fixes le long de la séquence ou qui varient en fonction du temps. Une phrase peut être exprimée différemment en fonction de l'humeur du locuteur, un geste peut être plus ample en fonction de la taille de l'acteur etc ... Ce type de variabilité ne peut pas toujours être supprimée par des pré-traitements.

Dans un premier temps, nous proposons les modèles Markoviens Contextuels (CHMM), afin de modéliser directement l'influence du contexte sur les séquences d'observation en paramétrisant les distributions de probabilités des HMMs par des variables contextuelles statiques ou dynamiques.

Puis, nous décrivons une approche afin d'exploiter efficacement l'information contextuelle dans un modèle discriminant, les Champs de Markov Conditionnels et Contextuels (CHCRF).

Nous testons plusieurs variantes des CHMMs et investiguons dans quelle mesure cette modélisation est pertinente pour la classification de caractères manuscrits, la reconnaissance de parole ou pour synthétiser les mouvements de sourcils d'un avatar à partir du signal de parole.

Enfin, nous proposons une stratégie d'apprentissage afin d'apprendre des HMM Contextuels plus performants à partir de moins d'exemples. Cette méthode réalise du partage d'information entre les classes la ou les approches génératives classiques considèrent des modèles de classes indépendants.

Résumé français long

9.1 Introduction

La modélisation de données séquentielles est utile à de nombreux domaines : reconnaissance de parole, de gestes, d'écriture, ou encore la synthèse de comportements réalistes pour l'animation d'avatars virtuels. Cependant la nature temporelle des données séquentielles nécessite une modélisation adéquate. Deux familles de modèles Markoviens sont particulièrement adaptées aux données de séquence.

Une famille générative avec les modèles de Markov à états cachés (HMMs), notamment adaptés à la synthèse, et une famille discriminative incluant les Champ de Markov Conditionnels à états cachés (HCRF), plus adaptés aux tâches de classification et de reconnaissance.

Notre modélisation part du constat qu'une part importante de la variabilité entre les séquences d'observations peut être la conséquence de quelques variables contextuelles fixes le long de la séquence ou qui varient en fonction du temps. Une phrase peut être exprimée différemment en fonction de l'humeur du locuteur, un geste peut être plus ample en fonction de la taille de l'acteur etc ... Ce genre de variabilité ne peut pas toujours être supprimée par des prétraitements. Ainsi, il serait judicieux de modéliser explicitement l'influence du contexte sur la variabilité des séquences d'observations.

Dans un premier temps, nous proposons les modèles Markoviens Contextuels (CHMM), afin de modéliser directement l'influence du contexte sur les séquences d'observations.

Puis, nous décrivons une approche efficace afin d'exploiter l'information contextuelle dans un modèle purement discriminant, les Champs Markoviens Conditionnels et Contextuels (CHCRF). Cette approche qui est le pendant discriminatif des CHMMs génératifs peut être vue comme une façon efficace d'apprendre des HCRFs utilisant le contexte.

Nous testons plusieurs configurations des CHMMs en classification de caractères manuscrits, en reconnaissance de parole ou pour la synthèse de mouvements de sourcils à

partir du signal de parole pour animer un avatar virtuel. Une part de nos travaux s'attèle aussi à limiter les effets de sur apprentissage et à proposer des stratégies afin de mieux combiner les variables contextuelles.

Enfin, nous développons l'idée d'apprendre des modèles contextuels à partir de moins d'exemples. A cet effet, nous proposons une stratégie d'apprentissage générative qui réalise du partage d'information entre les classes la ou les approches classiques apprennent des modèles de classes indépendants.

9.2 Modéliser la variabilité dans les HMM

Les modèles Markoviens génératifs à états cachés (HMM) font partie des approches les plus populaires pour la modélisation des données séquentielles. Ils peuvent être utilisés dans des tâches de classification, de reconnaissance ou de synthèse. La simplicité de leur procédure d'apprentissage ou d'inférence rend ces modèles particulièrement attractifs. Pour modéliser les données séquentielles, les HMM partent de l'hypothèse que la séquence d'observations $\boldsymbol{x}$ est générée par des états latent (non observable).

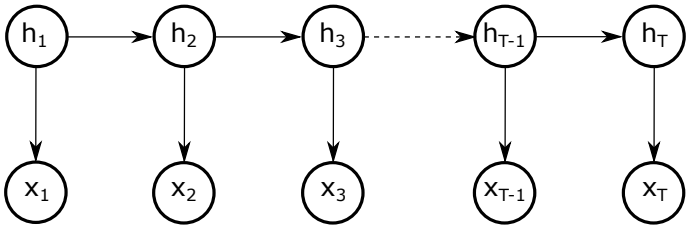


FIGURE 9.1: HMM avec une structure de type chaîne, chaque état h_t émet l'observation $\boldsymbol{x}_t$. Le modèle peut transiter d'un état à l'autre en fonction de probabilités de transitions

Etant donné une séquence d'états $\boldsymbol{h}$ de longueur T la probabilité jointe de la séquence d'états et de la séquence d'observations $\boldsymbol{x}$ se calcule comme suit :

$$p(\boldsymbol{x}, \boldsymbol{h}; \Lambda) = p(h_1)p(x_1 \mid h_1) \prod_{t=2}^{T} p(h_t \mid h_{t-1})p(\boldsymbol{x}_t \mid h_t) \tag{9.1}$$

où Λ sont les paramètres du modèle

La distribution de probabilités $p(\boldsymbol{x}_t \mid h_t)$ est définie comme une Gaussienne

$$p(\boldsymbol{x}_t \mid h_t) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_{h_t}, \Sigma_{h_t})$$

ou comme un mélange de M Gaussiennes dans chaque état

$$p(\boldsymbol{x}_t \mid h_t) = \sum_{m=1}^{M} p(m \mid h_t) p(\boldsymbol{x} \mid \boldsymbol{h}_t, m)$$
$$p(\boldsymbol{x}_t \mid h_t) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_{c_m}, \Sigma_{c_m}) \tag{9.2}$$

- où c_m est le coefficients (scalaire) de la Gaussienne m dans le mélange
- $\boldsymbol{\mu}_{c_m}$ est la moyenne de la Gaussienne (vecteur de dimension d)
- Σ_{c_m} est la covariance de la Gaussienne (matrice $d \times d$)

Les HMMs se basent cependant sur de nombreuses hypothèses simplificatrices et plusieurs variantes ont été proposées pour y remédier. Une hypothèse particulièrement limitante est notamment que les distributions de probabilités sont stationnaires dans les états. Concrètement, cela implique qu'un HMM modélise les séries temporelles avec des distributions constantes par morceaux. C'est une manière très simplicite de modéliser la variabilité des séquences d'observations.

De nombreuses approches ont introduit des distributions de probabilités non stationnaires dans les HMMs, on peut citer notamment les HMM polynomiaux Deng et al [20][21], les HMMs dit de trajectoire Zen et al. [90]. Par ailleurs, d'autres HMM non stationnaires conditionnent les distributions de probabilités par des variables externes telles que les Input Output HMMs (IOHMM [5]), ou les HMM paramétriques (PHMM [84]). C'est sur ce dernier type d'approche que nous élaborons nos propositions.

9.3 Les modèles Markoviens Contextuels (CHMM)

Le point de départ des CHMMs ou HMM Contextuels se base sur le formalisme des HMM paramétriques (PHMM). Partant des PHMM, les CHMM proposent d'augmenter l'expressivité de la modélisation en introduisent de nouveaux types de conditionnements possibles sur les distributions de probabilités d'un HMM (e.g. conditionnement des covariances pleines et des probabilités transitions par des variables de contexte dynamiques).

9.3.1 Paramétrisation de la moyenne des Gaussiennes

Dans les HMM paramétriques, les distributions Gaussiennes dans les états sont conditionnées par des variables externes (ou contextuelles). En reconnaissance de parole, ces variables peuvent par exemple représenter une information concernant le locuteur (genre,

langue maternelle, ...). En reconnaissance de geste, il peut s'agir de la taille ou de la corpulence d'un acteur.

Dans les PHMMs [84], Bobick et al, expriment la moyenne des distributions de probabilités Gaussiennes comme une combinaison linéaire des variables contextuelles notées $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}_t \mid h_t = j) = \mathcal{N}(\boldsymbol{x}_t, \hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta}), \Sigma_j)$$

avec :

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{j}}(\boldsymbol{\theta}) = V^j \boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$$

où V^j est une matrice $d \times c$ de coefficients dans l'état j (d la dimension des observations et c la dimension des variables contextuelles $\boldsymbol{\theta}$) et $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ un biais (vecteur $d \times 1$).

Une approche très similaire est également connue sous le nom de HMM à régression multiple (MRHMM [29]). Elle peut être vue comme des PHMM où la variable contextuelle dépend du temps $\boldsymbol{\theta}_t$. Enfin, une dernière classe de modèles connue sous le nom de VPHMM conditionnent également les covariances (diagonales) dans les Gaussiennes ([17], [18]).

9.3.2 Paramétrisation des covariances pleines des Gaussiennes

Nous proposons ici un autre type de conditionnement possible des distributions de probabilités en paramétrisant les matrices de covariances pleines des Gaussiennes d'un HMM. Une matrice de covariance $\bar{\Sigma}$ est ici modifiée de façon à ce que chacune de ses composantes $\bar{\Sigma}_{u,v}$ soit pondérée par deux facteurs d'échelle i.e. $\bar{\Sigma}_{u,v} = \bar{\Sigma}_{u,v} \times \alpha_u \times \alpha_v$ où α_u et α_v sont des termes dépendants des variables contextuelles $\boldsymbol{\theta}$. Cela permet d'obtenir un modèle plus expressif, en augmentant ses degrés de liberté.

La paramétrisation proposée s'écrit sous la forme matricielle suivante :

$$\begin{aligned} \hat{\Sigma}^j(\boldsymbol{\theta}) &= D^j(\boldsymbol{\theta}) \times \bar{\Sigma}^j \times D^j(\boldsymbol{\theta}) \\ \text{avec } D^j(\boldsymbol{\theta}) &= diag(exp(Z^j \boldsymbol{\theta})) \end{aligned} \tag{9.3}$$

avec $\hat{\Sigma}^j$ une matrice de covariance $d \times d$ pour l'état j, $\bar{\Sigma}^j$ une matrice de covariance indépendante de $\boldsymbol{\theta}$ (par exemple initialisée par celle d'un HMM) et Z^j une matrice $d \times c$ de la même forme que pour la paramétrisation de la moyenne. $Z^j = \left[U^j \quad \widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}} \right]$ où U^j est une matrice $d \times (c-1)$, $\widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}}$ un vecteur de biais $d \times 1$ et $\boldsymbol{\theta}$ un vecteur de variables contextuelle de dimension c où la dernière composante est systématiquement fixée à 1.

Nous avons noté ici $exp(A)$ une matrice ou l'exponentielle est appliquée sur chacun de ses termes, et $diag$ une fonction transformant un vecteur en matrice diagonale. Au final, comme nous l'avons évoqué, la forme de la matrice de covariance paramétrisée implique que chacun de ses termes ligne u et colonne v peut s'écrire :

$$\hat{\Sigma}^j_{u,v}(\boldsymbol{\theta}) = D^j_{u,u}(\boldsymbol{\theta}) \times D^j_{v,v}(\boldsymbol{\theta}) \times \bar{\Sigma}^j_{u,v}(\boldsymbol{\theta})$$

La Figure 9.2 illustre l'effet d'une telle paramétrisation sur une matrice de covariance en 2 dimensions. Une matrice de covariance originale (figure haut gauche) est ici modifiée par diverses matrices D. Chacune des 4 matrices de covariance est représentée par des courbes elliptiques d'isoprobabilités.

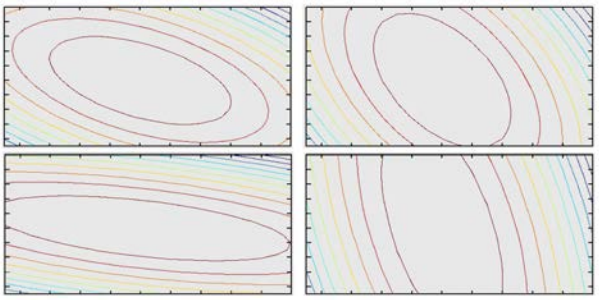


FIGURE 9.2: Exemples de matrices de covariances paramétrisées Eq. (9.3) en 2 dimensions. Une matrice de covariance initiale (Haut gauche) est transformée par plusieurs matrices D : $D = diag([1\ 2])$ (Haut droite), $D = diag([2\ 0.9])$ (Bas gauche), $D = diag([0.8\ 3])$ (Bas droite). Chaque matrice de covariance est illustrée par des ellipses correspondant aux courbes d'isoprobabilités.

9.3.3 Paramétrisation des probabilités de transitions

Enfin nous proposons une paramétrisation des probabilités de transitions qui jouent aussi un role dans la modélisation des données d'un HMM. Dans certains cas que nous envisagerons par la suite, il est intéressant de paramétriser les probabilités de transitions afin de définir des séquences d'états plus ou moins probables en fonction du contexte. En particulier, puisque chaque séquence d'observation évolue dans un contexte qui lui est propre, il paraît plus adapté d'utiliser des probabilités de transitions dépendantes du contexte plutôt que des transitions identiques pour chaque séquence.

Dans ce cas, nous re-définissons la probabilité de transition $a_{i,j}$ de l'état i vers l'état j au temps t par :

$$\hat{a}_{i,j}(\boldsymbol{\theta}) = \frac{\exp\left[\log \bar{a}_{ij} + \boldsymbol{w}_{ij}^T \boldsymbol{\theta}\right]}{\sum_k \exp\left[\log \bar{a}_{ik} + \boldsymbol{w}_{ik}^T \boldsymbol{\theta}\right]} \tag{9.4}$$

où $\bar{a}_{ij}$ est la probabilité de transition initiale de l'état i à l'état j telle qu'utilisée dans un HMM ou un CHMM sans paramétrisation des transitions et $\boldsymbol{w}_{ij}$ un vecteur de poids de la même dimension que $\boldsymbol{\theta}$.

Il est intéressant de constater que les CHMMs ainsi définis généralisent les HMMs. En effet, un CHMM devient un HMM (doté des moyennes $\bar{\boldsymbol{\mu}}^{\boldsymbol{j}}$, des covariances $\bar{\Sigma}^j$ et des probabilités de transitions $\bar{a}_{ij}$) en fixant simplement les matrices V^j et U^j et les vecteurs $\widetilde{\boldsymbol{\Sigma}^{\boldsymbol{j}}}$ et $\boldsymbol{w}^{ij}$ à 0.

Notons finalement que l'extension de ces paramétrisation à plusieurs Gaussiennes par états ou à l'utilisation de variables contextuelles dépendant du temps $\boldsymbol{\theta}_t$ ne pose pas de problèmes.

9.3.4 Vue Bayésienne

Dans les CHMMs, l'influence des variables contextuelles sur les distributions d'emisssion et de transitions peut être représentée par le modèle graphique ci-dessous :

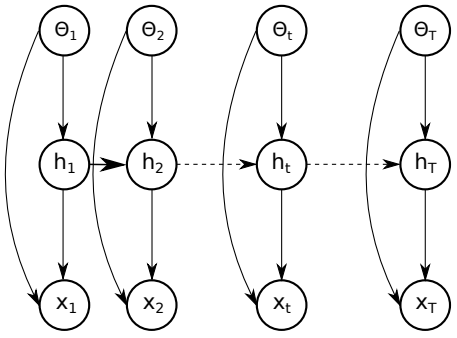


FIGURE 9.3: Représentation Bayésienne des HMM Contextuels lorsque les distributions de probabilités d'emission et de transitions du HMM sont paramétrisées par des variables contextuelles dynamiques θ_t

En apprentissage on considère la variable contextuelle $\boldsymbol{\theta}$ connue (ou observable), cependant, ça n'est pas toujours le cas en test. A cet effet, Bobick et al. ont montré [84] comment $\boldsymbol{\theta}$ peut être inferée dans les PHMM.

Dans les CHMMs, il est également possible d'inferer la variable contextuelle en test. Cependant, il n'y a pas de formule analytique pour reestimer $\boldsymbol{\theta}$ lorsqu'on conditionne les distributions de probabilités avec les nouvelles paramétrisations (pour les transitions, et covariances). D'autre part, utiliser des variables contextuelles dynamiques nécessiterait d'inférer un $\boldsymbol{\theta}_t$ à chaque instant ce qui est nettement plus couteux. De ce fait, on considère l'utilisation des CHMM avec l'emploi de variables contextuelles *observables* en test.

9.3.5 Résultats en classification de caractères

Nous rapportons ici des résultats obtenus sur une base de données de classification de caractères (IAM) comprenant 12 folds. Chaque fold est composé de 200 exemples d'apprentissage, 50 exemples en validation et 50 exemples en test. Les données sont préalablement normalisées de façon à avoir une moyenne nulle et un écart type de 1. Chaque séquence d'observations $\boldsymbol{x}$ représente un caractère manuscrit. Les observations $\boldsymbol{x}_t$ à chaque instant sont des vecteurs à 9 dimensions qui caractérisent une fenêtre glissante de 1 pixel sur l'image du caractère.

FIGURE 9.4: Exemples de caractères manuscrits 'm' et 'e' extraits de la base IAM

Nous utilisons ici plusieurs définitions du vecteur de variables contextuelles $\boldsymbol{\theta}$. Nous avons utilisé les dérivées des observations moyennées sur toute la longueur de la séquence (notées 'Δ'), les accélérations moyennées de la même manières (notées 'Δ^2'). Nous avons aussi utilisé des variables contextuelles dépendant du temps telles que la dérivée, l'accélération et la variance des observations à chaque instant. Leur valeur est moyennée sur une fenêtre glissante dont la taille est indiquée entre parenthèses. Nous comparons les CHMMs à moyennes paramétrisées (μCHMM) et à moyennes et covariances paramétrisées ($\mu\Sigma$CHMM) avec des HMMs (que l'on nomme AugHMMs) dont le vecteur d'observations à chaque instant est simplement augmenté ou concaténé avec l'information contextuelle $\boldsymbol{\theta}_t$.

On peut constater que les CHMMs dont la moyenne est paramétrisée (μCHMM) sont systématiquement plus performants que les HMMs auxquels l'information contextuelle est rajoutée naivement dans le vecteur de caractéristiques (AugHMM). D'autre part, la paramétrisation des covariances ($\mu\Sigma$CHMM) apporte un gain par rapport à la seule paramétrisation des moyennes. Par ailleurs la performance des HMMs standard sans information contextuelle n'excède pas 67% avec 8 états et 2 Gaussiennes par état.

TABLE 9.1: Performance des HMM Contextuels (CHMM) pour différentes topologies (nb d'états par classe, nb de Gaussiennes par états et différentes définitions du vecteur de variables contextuelles $\boldsymbol{\theta}_t$ calculé sur une fenêtre de taille variable (trouvée par essais erreurs sur la base de validation). Les résultats sont moyennés sur les 12 folds (écart type indiquée entre parenthèses). Notons que les gains des $\mu\Sigma$CHMMs comparés aux μCHMMs et aux HMMs sont statistiquement significatifs, de même que les gains des μCHMMs par rapport HMMs ($pvaleur < 5\%$).

nb states	nb gauss	Context variable $\boldsymbol{\theta}_t$	Train augHMM	Train μCHMM	Train $\mu\Sigma$CHMM	Test augHMM	Test μCHMM	Test $\mu\Sigma$CHMM
3	1	$\mu(41)$	60.5 (0.6)	66.1 (0.7)	**67.2 (0.6)**	56.3 (1.4)	61.5 (1)	**62.2 (1.3)**
3	1	$\mu(61)$ $\sigma^2(55)$	64.5 (1)	76.9 (0.7)	**79 (1.1)**	55.6 (1.8)	67 (1.3)	**67.6 (1.7)**
3	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	64.6 (0.7)	73.6 (0.7)	**77.4 (1.6)**	55.8 (1.6)	65.1 (1.2)	**66.2 (1.5)**
3	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	64.5 (0.6)	74.8 (0.8)	**77.2 (1)**	55.3 (1.7)	64.6 (1)	**65.5 (1.2)**
3	2	$\mu(41)$	67.7 (1.1)	70.3 (0.9)	**73.7 (1.3)**	59.8 (1.3)	62.8 (1)	**65.5 (1.7)**
3	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	75 (0.9)	80.4 (1.1)	**83.2 (1.7)**	59.7 (1.5)	65.6 (1.2)	**67.1 (0.9)**
5	1	$\mu(41)$	66.1 (0.6)	72.1 (0.7)	**73.4 (1.1)**	60.1 (1.6)	65.5 (1.2)	**66.3 (1.2)**
5	1	$\mu(61)$ $\sigma^2(55)$	70.3 (0.7)	76.9 (0.7)	**79 (1.1)**	59.3 (1.3)	67 (1.3)	**67.6 (1.7)**
5	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	70.2 (0.7)	79.2 (0.5)	**81.2 (0.9)**	59.1 (1.2)	66.9 (1.1)	**68.1 (1.5)**
5	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	70.5 (0.9)	80.4 (0.8)	**81.4 (0.8)**	59.5 (1.1)	67 (0.8)	**67.4 (0.9)**
5	2	$\mu(41)$	75.7 (0.8)	79.7 (1.2)	**81.4 (1.2)**	64 (1.2)	69.5 (1.8)	**70.2 (1.1)**
5	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	80.7 (0.6)	88.1 (0.6)	**88.3 (0.7)**	62.6 (2)	69.4 (0.8)	**69.8 (0.9)**
8	1	$\mu(41)$	70.9 (0.7)	78 (0.9)	**79 (1)**	63.5 (1.3)	70.3 (1.1)	**70.9 (1.1)**
8	1	$\mu(61)$ $\sigma^2(55)$	74.2 (0.9)	82.6 (0.8)	**84.2 (0.7)**	61.7 (1.3)	70.8 (1.2)	**71.7 (1)**
8	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$	73.7 (0.6)	84.6 (0.7)	**85.4 (0.7)**	61.2 (1.5)	71.4 (1.2)	**72.1 (0.9)**
8	1	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	74.3 (0.8)	85.6 (0.6)	**86.3 (0.8)**	61.6 (1.6)	70.7 (1.2)	**71.4 (1.3)**
8	2	$\mu(41)$	80.3 (1)	85.6 (0.7)	**86.5 (0.6)**	59.8 (1.3)	73.3 (1.5)	**74.2 (1.3)**
8	2	$\mu(61)$ $\sigma^2(55)$ $\Delta(15)$ $\Delta^2(15)$	84.3 (0.7)	92.4 (1.1)	**92.5 (1.2)**	64.5 (2.2)	72.3 (1.2)	**72.6 (1.2)**

9.3.6 Résultats en synthèse

Les CHMM étant des modèles génératifs, ils sont capable de synthétiser des données, ce qui est particulièrement utile dans certains domaines comme la synthèse de parole à partir de texte, ou l'animation d'avatars virtuels par exemple. Nous présentons ici une approche exploitant la capacité de modélisation des CHMMs pour modéliser les mouvements de sourcils d'un avatar à partir du signal de parole. La base de données utilisée contient 240 phrases étiquetées par 5 classes d'expression faciales types. 80% des données sont réservées à l'apprentissage, 20% à la validation et 20% au test.

Pour cette application, nous paramétrisons les moyennes et les transitions des CHMMs (μTrCHMM) avec des variables contextuelles dynamiques liées au signal de parole (pitch

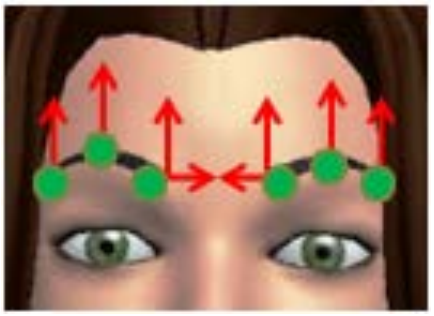


FIGURE 9.5: Illustration de l'animation des sourcils d'un avatar virtuel (les flèches indiques le déplacement).

et énergie calculés sur 10 trames). Les CHMMs sont appris sur des observations composées de caractéristiques de mouvement et de caractéristiques audio. Au moment de la synthèse, le signal de parole est utilisé pour calculer les variables contextuelles qui conditionnent le modèle. Uniquement les séquences d'observations liées au mouvement sont générées. Le processus de synthèse à partir des modèles fait également appel à la méthode décrite en [76] qui permet de générer des séquences d'observations ou trajectoires plus réalistes.

TABLE 9.2: Performance des modèles (HMM, μCHMM et μTrCHMM) par rapport à la qualité de synthèse (critère MSE i.e. erreur aux moindres carrés entre la séquence réelle et la séquence synthétisée). On rapporte aussi la performance des modèles en étiquetage de séquences sur la même base (distance de hamming (H) et distance d'édition (E) par rapport à la séquence d'étiquettes réelles). Les résultats sont moyénnés sur 20 expériences (écart type entre parenthèses).

Modèle	#états	MSE	Acc (H)	Acc (E)
HMM [40]	3	0.67(0.052)	37%(4.7)	45%(4.2)
	5	0.59(0.042)	43%(4.7)	49%(4.4)
	7	0.56(0.056)	53%(5.7)	51%(4.3)
μCHMM	3	0.51(0.055)	55%(4.8)	49%(4.4)
	5	0.49(0.064)	58%(5.7)	50%(4.9)
	7	0.47(0.056)	59%(4.5)	50%(3.4)
μtrCHMM	3	0.55(0.042)	60%(5.3)	57%(4.7)
	5	0.46(0.051)	61%(5.1)	61%(3.8)
	7	0.45(0.037)	63%(3.0)	62%(3.7)

Comme nous pouvons le voir, les 2 approches basées sur les CHMM (μCHMM, μtrCHMM) ont de meilleures performances que les HMM et la performance des CHMMs à moyennes et transitions paramétrisées est à la fois la meilleure en synthèse et en reconnaissance.

9.4 les Champs de Markov Conditionnels et Contextuels (CHCRF)

Dans les dernières années, un certain nombre de chercheurs ont investigué l'utilisation de modèles purement discriminants pour l'étiquetage de séquences. Les champs de Markov Conditionnels (CRF) et leur extension à l'utilisation d'états cachés (HCRF) en font notamment partie. Une première raison de considérer les HCRF par rapport aux autres approches discriminantes est qu'ils n'ont pas la contrainte d'utiliser des distributions de probabilités Gaussiennes pour modéliser les données. Comme évoqué par leCun et al. [48] l'utilisation de distributions de probabilités n'est pas nécessairement utile pour des tâches de classification.

D'autre part, la performance des modèles discriminants est souvent supérieur à celle des modèles génératifs en classification ou reconnaissance [13, 28, 34, 54, 56, 71, 74, 80]. Par ailleurs, les HCRFs figurent parmi les méthodes les plus performances [34, 54, 74, 80].

Motivés par ces résultats, nous présentons ici une approche afin d'exploiter l'information contextuelle dans les HCRFs. Nous rappelons brièvement les fondamentaux de la modélisation HCRF puis nous expliquons comment les HMMs peuvent être vus comme des cas particuliers des HCRFs, ce qui conduit à un schéma efficace d'initialisation pour apprendre les HCRFs [34]. Partant de cette idée, nous élaborons finalement une méthode efficace pour apprendre un HCRF tirant partie de l'information contextuelle.

Les HCRFs sont particulièrement sensibles au sur-apprentissage. D'une part parce que leur apprentissage utilise un critère d'optimisation non convexe et d'autre part car ils calculent des scores moins contraints que les probabilités Gaussiennes des HMMs. Cependant, il existe une méthode efficace pour apprendre les HCRFs qui a été proposée récemment [34]. Elle consiste à apprendre tout d'abord un HMM, puis initialiser les paramètres d'un HCRF afin qu'il reproduise la même fonction de décision que les HMMs.

Cette stratégie est aussi intéressante car les HCRFs sont entrainés avec un critère discriminant plus adapté à la classification. Cependant, l'apprentissage par maximum de vraissemblance dans les HMMs est facilement parallélisable car chacun des modèles de classe est indépendant ce qui n'est pas le cas avec les HCRFs. De ce fait, il est plus rapide d'apprendre des HMMs que des HCRFs et initaliser un HCRF avec un HMM peut améliorer les performances pour un coût limité.

Nous expliquons ici comment le schéma d'initialisation est réalisé. Le point crucial est d'exprimer la probabilité jointe des séquences d'observations et des états comme un produit scalaire entre un vecteur de paramètres et un vecteur de caractéristiques dépendant de la classe, de la séquence d'observations et des états.

Pour une séquence d'états $\boldsymbol{h}$ d'un HMM on peut en effet écrire :

$$\begin{aligned}\log p(\mathbf{x}, y, \boldsymbol{h}; \Lambda) &= \log(\pi_{h_1}) + \log(p(\boldsymbol{x}_1|h_1)) \\ &\quad + \sum_{t=2}^{T} \left(\log p(h_t|h_{t-1}) + \log p(\boldsymbol{x}_t|h_t, \mu_{h_t}, \Sigma_{h_t}\right)\end{aligned}$$

avec :

$$\begin{aligned}\log p(\boldsymbol{x}_t|h_t; \Lambda) &= -\frac{1}{2}\Big(\boldsymbol{x}_t^T \Sigma_{h_t}^{-1} \boldsymbol{x}_t - \boldsymbol{x}_t^T \Sigma_{h_t}^{-1} \boldsymbol{\mu}_{h_t} - \boldsymbol{\mu}_{h_t}^T \Sigma_{h_t}^{-1} \boldsymbol{x}_t \\ &\quad + \boldsymbol{\mu}_{h_t}^T \Sigma_{h_t}^{-1} \boldsymbol{\mu}_{h_t} + \log((2\pi)^d |\Sigma_{h_t}|)\Big) \\ &= \langle \boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t)\rangle\end{aligned}$$

et :

$$\log p(h_t|h_{t-1}; \Lambda) = \langle \boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1})\rangle \tag{9.5}$$

De ce fait, on a :

$$\log p(\boldsymbol{x}, y, \boldsymbol{h}; \Lambda) = \sum_t \langle \boldsymbol{\lambda}^{trans}, \boldsymbol{\phi}^{trans}(\boldsymbol{x}_t, y, h_t, h_{t-1})\rangle + \langle \boldsymbol{\lambda}^{loc}, \boldsymbol{\phi}^{loc}(\boldsymbol{x}_t, y, h_t)\rangle$$

où $\boldsymbol{\lambda}^{loc}$ et $\boldsymbol{\lambda}^{trans}$ sont les vecteurs de paramètres du HCRF et $\boldsymbol{\phi}^{loc}$ et $\boldsymbol{\phi}^{trans}$ sont les vecteurs de caractéristiques associés.

Afin de définir les HCRFs contextuels (CHCRFs), on doit préalablement définir les vecteurs de caractéristiques $\boldsymbol{\phi}^{trans}$ et $\boldsymbol{\phi}^{loc}$ puis apprendre un HCRF linéaire sur cette représentation. Un choix simple serait de concaténer les observations $\boldsymbol{x}$ et les variables contextuelles $\boldsymbol{\theta}$ dans le vecteur de caractéristiques $\boldsymbol{\phi}$ afin que le HCRF calcule une fonction linéaire de ces entrées. Nous nommons cette technique HCRF augmenté (AugH-CRF). Un choix plus intéressant consiste à définir un vecteur de caractéristiques $\boldsymbol{\phi}$ incluant le produit Cartésien des composantes de $\boldsymbol{x}_t$ et de $\boldsymbol{\theta}_t$ de façon à ce que le HCRF simule la fonction du décision du CHMM. Cette stratégie a l'avantage de pouvoir être initialisée par un HMM Contextuel (CHMM), surmontant l'obstacle principal des HCRFs qui est la sensibilité leur performance par rapport à l'initialisation. Dans nos expériences, l'initialisation aléatoire des HCRFs atteint péniblement une performance de 40% sur la base de classification IAM utilisée (en utilisant un produit Cartésien entre les composantes des observations comme vecteur de caractéristique). D'un autre côté, les HMMs sont des modèles plus simples et moins sensibles au sur-apprentissage, de plus il existe des méthodes standard pour les initialiser, (les paramètres des distributions Gaussiennes

peuvent notamment être initialisés par l'algorithme des K-moyennes). Au final, l'obtention d'une solution initiale par des HMMs entrainés par maximum de vraissemblance est une idée classique, notamment utilisée pour d'autres techniques d'apprentissage discriminant (MMI, MCE, MPE ... [43], [61] and [88]).

Nous proposons donc l'idée de construire un vecteur de caractéristiques $\boldsymbol{\phi}$ qui dépend de $\boldsymbol{x}$ et de $\boldsymbol{\theta}$ de manière à simuler la fonction de décision du CHMM par un HCRF contextuel. De ce point de départ, l'optimisation d'un HCRF conduit éventuellement à un HCRF plus performant exploitant efficacement l'information des variables contextuelles.

Afin d'implémenter les HCRFs Contextuels (CHCRF) nous devons définir des vecteurs de caractéristiques $\boldsymbol{\phi}^{loc}$ et $\boldsymbol{\phi}^{trans}$ et initialiser les vecteurs de paramètres $\boldsymbol{\lambda}^{loc}$ et $\boldsymbol{\lambda}^{trans}$ à partir des HMM Contextuels (CHMM) suivant le schéma suivant :

Pour une séquence d'états $\boldsymbol{h}$, la probabilité d'émission dans les CHMM se formule de la manière suivante :

$$\begin{aligned}\log p(\boldsymbol{x}_t|h_t,\boldsymbol{\theta}_t;\Lambda) &= -\frac{1}{2}\Big(\boldsymbol{x}_t^T\Sigma_{h_t}^{-1}(\boldsymbol{\theta}_t)\boldsymbol{x}_t - \boldsymbol{x}_t^T\Sigma_{h_t}^{-1}(\boldsymbol{\theta}_t)\boldsymbol{\mu}_{h_t}(\boldsymbol{\theta}_t) - \boldsymbol{\mu}_{h_t}(\boldsymbol{\theta}_t)^T\Sigma_{h_t}^{-1}(\boldsymbol{\theta}_t)\boldsymbol{x}_t \\ &\quad + \boldsymbol{\mu}_{h_t}(\boldsymbol{\theta}_t)^T\Sigma_{h_t}^{-1}(\boldsymbol{\theta}_t)\boldsymbol{\mu}_{h_t}(\boldsymbol{\theta}_t) + \log((2\pi)^d|\Sigma_{h_t}(\boldsymbol{\theta}_t)|)\Big) \\ &= \langle\boldsymbol{\lambda}^{loc},\boldsymbol{\phi}^{loc}(\boldsymbol{x}_t,y,h_t,\boldsymbol{\theta}_t)\rangle\end{aligned}$$

pour les transitions, on a :

$$\log p(h_t|h_{t-1},\boldsymbol{\theta}_t;\Lambda) = \langle\boldsymbol{\lambda}^{trans},\boldsymbol{\phi}^{trans}(\boldsymbol{x}_t,y,h_t,h_{t-1},\boldsymbol{\theta}_t)\rangle \tag{9.6}$$

On initialisera $\boldsymbol{\lambda}^{loc}$ et $\boldsymbol{\lambda}^{trans}$ avec les paramètres du HMM, tandis que les vecteurs de caractéristiques $\boldsymbol{\phi}^{loc}$ et $\boldsymbol{\phi}^{trans}$ contiendront essentiellement des termes dépendants de $\boldsymbol{\theta}$ et des observations $\boldsymbol{x}$. Ce choix permet de définir un CHCRF avec le même nombre de paramètres que le CHMM.

9.4.1 Résultats en classification de caractères

Nous investigons ici le gain apporté par l'exploitation des variables contextuelles dans les HCRFs. Nous nous intéressons à comparer les CHCRFs aux HCRFs standards qui sont à l'état de l'art des modèles de classification et d'étiquetage de séquences. Nous conduisons des expériences sur la classification de caractères isolés sur une partie de la base de données IAM d'écriture manuscrite (6 folds avec chacuns 200 examples en apprentissage, 50 en validation et 50 en test). Nous comparons les HCRFs, AugHCRFs et μCHCRFs

(μCHCRFs est initialisé par un CHMM dont la moyenne seule est paramétrisée). Les AugHCRFs et μCHCRFs utilisent l'information contextuelle $\boldsymbol{\theta}$ de différente manière tandis que les HMMs et les HCRFs ne l'utilisent pas. La variable contextuelle $\boldsymbol{\theta}$ est ici définie comme la moyenne statique ou dynamique de la séquence d'observations comme dans les expériences sur les CHMMs. L'initialisation du HCRF (Standard, Augmenté et Contextuel) est faite à partir du meilleur HMM correspondant (standard, Augmenté, Contextuel) sur la base de validation. Nous rappelons par ailleurs que les HMMs augmentés sont simplement les HMMs standards où $\boldsymbol{\theta}$ est concaténé au vecteur d'observations $\boldsymbol{x}_t$ à chaque instant.

TABLE 9.3: Performance en classification des modèles discriminants : HCRFs, augHCRFs et CHCRFs Les résultats sont moyennés sur les 6 folds de la base IAM. Tous les modèles ont 8 états par classe.

Modèle	Train	Test
HMM	65.6 (0.2)	60.6 (1.5)
HCRF	**67.8 (0.2)**	**63.7 (1.8)**
AugHCRF static $\boldsymbol{\theta} = \mu$	70.9 (0.6)	62.6 (1.1)
μCHCRF static $\boldsymbol{\theta} = \mu$	**78.1 (0.8)**	**70.7 (1.6)**
AugHCRF dynamic $\boldsymbol{\theta} = \mu(41)$	68.1 (0.7)	58.9 (2.2)
μCHCRF dynamic $\boldsymbol{\theta} = \mu(41)$	**77 (0.8)**	**68.4 (1.9)**
AugHCRF dynamic $\boldsymbol{\theta} = \mu(61)$	66.6 (1.2)	57.8 (2)
μCHCRF dynamic $\boldsymbol{\theta} = \mu(61)$	**77 (0.7)**	**68.2 (1.6)**

On peut tirer quelques conclusions des résultats du tableau 9.3. D'une part, les HCRFs sont nettement plus performants que les HMMs non discriminants, ce qui semble naturel étant donné que le critère d'apprentissage discriminant des HCRFs est plus adapté aux tâches de classification. D'autre part, les HCRFs augmentés n'arrivent pas à exploiter correctement l'information contextuelle car ils sont ici moins performants que les HCRFs ou les HMMs n'utilisant pas d'information contextuelle. C'est probablement du au sur-apprentissage puisque les HMMs augmentés (dont le nombre de paramètres est du même ordre que pour les AugHCRFs) ont beaucoup plus de paramètres que les HMMs. Cependant, bien qu'ils aient également plus de paramètres que les HMMs, les CHCRFs sont ici nettement plus performants. Dans cette tâche, le schéma d'initialisation des CHCRFs par CHMMs permet d'exploiter plus efficacement l'information contextuelle par rapport à une approche plus simple qui consiste à rajouter le contexte dans le vecteur de caractéristiques.

9.5 Approche de type Transfer avec les CHMM

L'utilisation de modèles plus expressifs comme les CHMMs est cependant plus sujet aux risques de sur-apprentissage que les HMMs lorsque l'on dispose de très peu de données d'apprentissage. Nous proposons à ce titre une stratégie d'apprentissage de type Transfer Learning qui permet d'améliorer la performance des CHMMs appris sur peu d'exemples.

Afin d'apprendre à partir de peu d'exemples, nous proposons d'apprendre un CHMM conjointement sur les données de l'ensemble des classes. Ce modèle unique, que nous appelons CHMM global, peut permettre de trouver des régularités dans les données qui sont partagées par les différentes classes. L'idée est qu'en utilisant davantage de données, on espère que l'estimation des paramètres du modèle sera plus fiable. Une fois appris, il s'agit de transférer la connaissance de ce modèle global à des modèles de classes individuels pour le décodage.

Nous définissons d'abord une variable contextuelle de classe $\boldsymbol{\theta}_y$ représentant la classe y qui est normalement modélisée par un HMM avec son jeu de paramètres Λ_y. Dans le modèle contextuel global, on attribue une variable contextuelle de classe $\boldsymbol{\theta}_y$ à chaque séquence d'observations $\boldsymbol{x} \in \mathcal{X}_y$ comme illustré dans la figure 9.6.

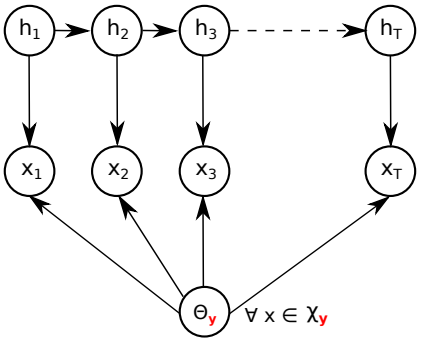


FIGURE 9.6: CHMM global représenté tel un réseau Bayésien dynamique. On attribue un vecteur contextuel de classe $\boldsymbol{\theta}_y$ à chaque séquence d'observation $\boldsymbol{x} \in \mathcal{X}_y$.

Ce mécanisme nous permet d'utiliser les données de toutes les classes conjointement dans un seul modèle. En d'autres termes, si $|Y|$ est le nombre de classes, au lieu d'entrainer $|Y|$ modèles avec leur propres données d'apprentissage, un seul CHMM global est entrainé sur l'ensemble des données de toutes les classes.

L'inférence avec un CHMM global comprend 2 étapes : Premièrement on instancie des HMMs pour chaque classe à partir du modèle global. Deuxièmement, on réalise une inférence classique de type Forward Backward sur les HMMs de chaque classe afin de sélectionner l'étiquette la plus probable pour la séquence d'observations $\boldsymbol{x} \in \mathcal{X}$.

Lorsqu'on conditionne le CHMM global par cette variable contextuelle de classe $\boldsymbol{\theta}_y$, les moyennes $\hat{\mu}_s(\boldsymbol{\theta}_y)$, les covariances $\hat{\Sigma}_s(\boldsymbol{\theta}_y)$ et les transitions $\hat{A}(\boldsymbol{\theta}_y)$ du CHMM définissent un HMM avec les moyennes μ_s, les covariances Σ_s et la matrice de transition A. Les paramètres Λ_y du HMM qui modélise la classe y ne dépendent donc plus de la variable contextuelle $\boldsymbol{\theta}_y$.

9.5.1 Résultats en classification de gestes

Nous avons effectué ici des expériences en classification de gestes sur des données de motion capture (5 folds chacun constitués de 702 examples en apprentissage, 225 en validation, et 203 en test). La base contient 37 classes de gestes (extraits dela base HDM05 [59]) effectués par 5 acteurs différents. Les observations sont composées de 62 angles d'articulations qui représentent le squelette de l'acteur à chaque instant. Dans ces expériences nous utilisons une réprésentation de type 1 parmi N pour le vecteur de variabes contextuelles $\boldsymbol{\theta}_y$ (soit un vecteur de dimensions 37 avec 1 à la position y et 0 partout ailleurs).

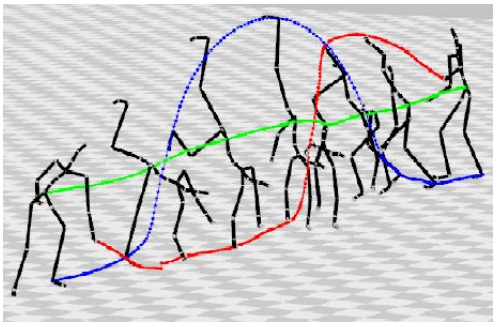


FIGURE 9.7: Données de motion capture du geste "cartwheel" représenté comme une séquence de postures. La figure indique les trajectoires 3D de 3 articulations (centre, doigt droit, cheville gauche).

	modèle	covariance	nb d'états	nb gauss	Train moyenne F1	Test moyenne F1	nombre de paramètres
modèles de classes indépendants	HMM	pleine	8	1	88.2(0.7)	62.2(1.9)	1158544
	HMM	pleine	8	5	100(0)	53.1(6.1)	5784728
	HMM	diag	8	1	87.7(0.7)	66.3(3.1)	39072
	HMM	diag	8	2	89.4(1.1)	66.4(3.1)	76368
	HMM	diag	8	5	91.8(2.5)	65.7(3.4)	185368
	HMM	diag	8	7	92.4(2)	63.5(2.5)	261368
	HMM	diag	8	9	94.2(1.2)	63.1(2.9)	335368
modèle global	$\mu\Sigma$CHMM	full	8	1	91.4(1.2)	67.4(2.9)	68512
	$\mu\Sigma$CHMM	full	8	5	97.1(0.6)	70.9(3.3)	342344

TABLE 9.4: Mesure F1 de performance des HMMs standards vs CHMM global dont les moyennes et covariances sont paramétrisées par une variable contextuelle de classe type 'hot code' (hcode) soit 1 parmi N

Dans le tableau 9.4 on peut constater que les HMMs à covariances pleines sur-apprennent les données rapidement. En utilisant des modèles de classe avec 8 états et 5 Gaussiennes par état, la performance en test des HMMs diminue (de 62.3% to 53.1%) comparé aux HMMs n'ayant qu'une Gaussienne pleine par état. En utilisant des Gaussiennes diagonales, le nombre de paramètres est nettement réduit et cela aide naturellement la généralisation, mais, la performance des HMMs stagne rapidement aux alentours de 66%.

A contrario, avec un CHMM global, ça n'est pas le cas : Utiliser un modèle à plus forte capacité améliore à la fois la performance en apprentissage et en test. Sans pour autant avoir fait une recherche aussi exhaustive que pour les HMMs, la performance du CHMM global est déjà supérieure à toutes les topologies de HMMs testées. Bien sur, le CHMM global utilisant des Gaussiennes pleines a moins de paramètres que les HMMs à Gaussiennes pleines, cependant, avec 5 Gaussiennes, il a déjà plus de paramètres que les HMMs à Gaussiennes diagonales ici présentés.

Il est relativement surprenant de constater ce genre de gains, car les code de classes sont totalement orthogonaux impliquant des modèles de classes presque indépendants. Toutefois, les classes partagent un biais commun dans chaque état. Afin de l'illustrer nous détaillons les paramétrisations d'un état s pour un CHMM modélisant la classe y :

$$\begin{aligned}\hat{\mu}_s(\boldsymbol{\theta}_y) =& V_s\boldsymbol{\theta}_y + \bar{\boldsymbol{\mu}}_{\boldsymbol{s}} \\ \hat{\Sigma}_s(\boldsymbol{\theta}_y) =& D_s(\boldsymbol{\theta}_y)\bar{\Sigma}_s D_s(\boldsymbol{\theta}_y) \quad \text{avec } D_s(\boldsymbol{\theta}_y) = diag(exp(U_s\boldsymbol{\theta}_y + \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}))\end{aligned}$$

où $\bar{\boldsymbol{\mu}}_{\boldsymbol{s}}$ et $\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}$ sont les biais respectivement associés à la paramétrisation de la moyenne et de la covariance.

Puisque $\bar{\boldsymbol{\mu}}_{\boldsymbol{s}}$ et $\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{s}}$ sont communs à toutes les classes, on peut voir les matrices V_s et U_s comme des paramètres modélisant seulement les différences restantes entre les classes vis à vis d'un état partagé. Ceci explique probablement pourquoi nous obtenons des performances bien supérieures aux HMMs sur peu d'exemples.

Supposons maintenant que $\boldsymbol{\theta}_y$ est encore un vecteur de dimension 37 (le nombre de classes) mais que chacun de ses coefficients à la position y' soit proportionnel à la similarité entre les classes y et y'. $\boldsymbol{\theta}_y$ est donc maintenant beaucoup plus dense que la représentation précédente (hot code). En réalité, utiliser un vecteur contextuel $\boldsymbol{\theta}_y$ avec une similarité entre les classes définit une pondération sur les coefficients de paramétrisation. Plus une classe y est similaire à une classe y', plus la pondération sur les coefficients de paramétrisation devient forte. Nous montrons ci-dessous une matrice de similarité sur 3 classes afin d'illustrer ce phénomène.

classe	1	2	3
1	1	0.8	0.2
2	0.8	1	0.1
3	0.2	0.1	1

TABLE 9.5: Matrice de similarité sur 3 classes, les classes 1 et 2 sont proches et plus éloignées de la classe 3

Supposons que les observations sont des scalaires, la moyenne d'une Gaussienne dans les HMMs des 3 classes peut être exprimée par :

$$\hat{\mu}(\boldsymbol{\theta}_1) = w_1 + 0.8 \times w_2 + 0.2 \times w_3 + \bar{\mu}$$
$$\hat{\mu}(\boldsymbol{\theta}_2) = 0.8 \times w_1 + w_2 + 0.1 \times w_3 + \bar{\mu}$$
$$\hat{\mu}(\boldsymbol{\theta}_3) = 0.2 \times w_1 + 0.1 \times w_2 + w_3 + \bar{\mu}$$

où w_1, w_2 et w_3 sont les moyennes des coefficients de paramétrisations

De ce fait, 2 classes similaires (1 et 2) auront tendance à définir des moyennes plus proches alors que la moyenne pour la classe 3 sera plus différente. Au final, si deux classes sont similaires, les moyennes et les covariances des HMMs représentant ces classes (qui sont issues du CHMM global) seront proches. En présence de peu de données d'apprentissage on impose ici un a priori fort sur l'estimation des paramètres des modèles de classes qui pourrait probablement nuire à la discrimination si l'on disposait de davantage de données.

Nous définissons maintenant la matrice de similarité "classSim" comme la matrice de confusion symmétrisée d'un HMM entrainé sur la même tâche. L'idée est que plus les classes sont proches, plus le HMM fait d'erreurs de prédiction les concernant, ce qui est bien reflété par sa matrice de confusion (notée $confusMat$) calculée sur la base de validation.

$$classSim = confusMat + confusMat^T$$

Nous normalisons chaque ligne de classSim de façon à ce que les coefficients soient entre 0 et 1. Finalement, la i^{eme} ligne de la matrice simClass définit le vecteur de variables contextuelles pour la classe i et exprime sa proximité vis à vis de toutes les autres classes.

Les résultats du tableau 9.6 montrent des différences notables entre l'utilisation d'un code de classe $\boldsymbol{\theta}_y$ de type hotcode et un code de classe de type similarité. L'utilisation d'un vecteur contextuel de type similarité produit une meilleure généralisation alors que la performance en apprentissage est un peu inférieure. Ces résultats sont encourageants car ils montrent qu'un changement dans la représentation du code de classe peut encoder une information utile à la généralisation pendant l'apprentissage.

modèle	nb d'états	nb gauss	Train moyenne F1	Test moyenne F1
$\mu\Sigma$CHMM (hcode)	8	1	91.4(1.2)	67.4(2.9)
$\mu\Sigma$CHMM (hcode)	8	5	97.1(0.6)	70.9(3.3)
$\mu\Sigma$CHMM (classSim)	8	1	89.5(0.9)	71.8(2.7)
$\mu\Sigma$CHMM (classSim)	8	5	96.5(0.7)	72.6(4.2)

TABLE 9.6: Mesure F1 de performance pour les CHMMs globaux paramétrisés par un vecteur contextuel de classe de type hot code (hcode) comparé à un vecteur contextuel encodant la similarité entre les classes (classSim)

9.6 Conclusion

A travers cette thèse, nous avons défini le paradigme des modèles de Markov Contextuels. Le point de départ de notre approche est qu'une part importante de la variabilité des séquences d'observations peut être expliquée par quelques variables contextuelles qui sont fixes ou qui varient au court du temps. Plusieurs déclinaisons des CHMMs ont été proposées pour incorporer l'influence de variables contextuelles dans la modélisation des HMMs. En particulier nous avons montré comment paramétriser à la fois les matrices de covariances des distributions Gaussiennes et les probabilités de transitions par des variables contextuelles dynamiques.

Nous avons ensuite pu proposer une méthode efficace pour exploiter l'information contextuelle dans un modèle discriminant les Champs de Markov Conditionnels et Contextuels (CHCRFs). Cette approche s'est révélée être un schéma d'apprentissage plus efficace pour exploiter l'information contextuelle dans les HCRFs que la forme d'apprentissage standard.

Nous avons pu expérimenter la performance de ces modèles sur diverses tâches de classification de caractères ainsi qu'en reconnaissance et en synthèse. Dans toutes ces tâches les capacités de modélisation des CHMM ont permis d'améliorer les performances des HMMs.

Enfin, nous avons développé un dernier type de technique permettant d'apprendre des CHMMs avec peu d'exemples. Cette approche basée sur un modèle global appris sur l'ensemble des données utilise des variables contextuelles encodant une représentation de la classe. Ce mécanisme permet de partager de l'information entre différentes classes pendant la phase d'apprentissage alors que l'apprentissage génératif classique les apprend indépendamment. Nos résultats on montré que cette méthode permet de mieux généraliser comparé à des HMMs testés sur diverses topologies.