# PROBABILISTIC TIME SERIES CLASSIFICATION

by

Yusuf Cem Sübakan

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2013

# ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis supervisor Bülent Sankur. Without his infinite support and encouragement I would surely be less of a student.

Taylan Cemgil, my thesis supervisor, changed the way I think and live, by teaching me in technical and non-technical aspects of research life. I owe him most of the technical stuff I know today and I am really grateful for his patience, wisdom and mentorship. I am genuinely thankful to F. Kerem Harmancı for introducing me to Taylan hoca.

I would also thank Çağatay Dikici for the endless hours of technical chats, and ping-pong games. Research is certainly more fun when you have someone to talk casually about the stuff you are working on. I would not like to forget Ceyhun Burak Akgül for the technical discussions we had on my first year, and the guys in pi-lab for being fellow Bayesians nearby.

I would like to thank fellow BUSIMers, first of all Oya Çeliktutan for being a patient listener to my problems, Erinç Dikici to answer all my paperwork related inquiries, and being patient to my loud music, Mehmet Yamaç for being my late night work-buddy. I also salute the other fellas. It would be wrong to forget the WCL HAXBALL brotherhood. Life is certainly better when you have the high-school boy spirit on with Alican Gök and Numan Su. My machinist friends, Erdem Eren and Osman Yüksel also contributed immensely in this aspect. A word also goes to an infinite list of musicians whose music I was listening to with my headphones, throughout my time in BUSIM. Without their music on, I would unquestionably be less productive.

Finally, I would like to thank my high school math teacher, Özcan Baytekin. It is due to him that I like math, and pursue a career in research.

# ABSTRACT

## PROBABILISTIC TIME SERIES CLASSIFICATION

In this thesis, we investigate probabilistic methods for time series classification and clustering problems. For various classification and clustering tasks, we survey different time series models such as Markov models, hidden Markov models (HMM), mixture of Markov models and mixture of Hidden Markov models. We also investigate discriminative versions of Markov models and Hidden Markov models. The novel contribution of this thesis is the derivation of algorithms for learning mixtures of Markov models and mixtures of hidden Markov models. Mixture models are special latent variable models that require the usage of local search heuristics such as Expectation Maximization (EM) algorithm, that can only provide locally optimal solutions. In contrast, we make use of the spectral learning algorithms, recently popularized in the machine learning community. Under mild assumptions, spectral learning algorithms are able to estimate the parameters in latent variable models by solving systems of equations via eigendecompositions of matrices or tensors of observable moments. As such, spectral methods can be viewed as an instance of the method of moments for parameter estimation, an alternative to maximum likelihood. The popularity stems from the fact that these methods provide a computationally cheap and local optima free alternative to EM. We conduct classification experiments on human action sequences extracted from videos, clustering experiments on motion capture data and network traffic data to illustrate the viability of our approach. We conclude that the spectral methods are a practical and useful alternative in terms of computational effort and solution quality to standard iterative techniques such as EM in many application areas.

# ÖZET

## OLASILIKSAL ZAMAN SERİSİ SINIFLANDIRMA

Bu tezde, olasılıksal zaman serisi sınıflandırma ve topaklandırma yöntemleri üzerine çalışılmıştır. Bu amaçla, Markov modelleri, saklı Markov modelleri, Markov modeli karışımları ve saklı Markov modeli karışımları kullanılmıştır. Ayırıcı Markov modelleri ve ayırıcı saklı Markov modelleri de incelenmiştir. Bu tezde, Markov modeli karışımları ve saklı Markov modeli karışımları öğrenmek için iki yeni algoritma öneriyoruz. Literatürde, karışım modellerinin öğrenilmesi çoğunlukla global optimum bulma garantisi olmayan Beklenti Enbüyütme algoritması ile yapılmaktadır. Biz, yapay öğrenme literatüründe son dönemde popüler olmaya başlayan spektral öğrenme yöntemlerinin kullanılmasını öneriyoruz. Çok kuvvetli olmayan varsayımlar altında, spektral öğrenme algoritmaları saklı değişken modelleri için sadece gözlemlenebilir moment matris veya tensörlerinin özdeğer-özvektör ayrışımını alarak parametre kestirimi yapma olanağı sunmaktadır. Böylelikle, spektral öğrenme yöntemleri olabilirlik enbüyütmeye dayalı beklenti enbüyütme algoritması için alternatif olarak görülebilir. Spektral öğrenme algoritmalarının popülerliği, hesap yükü bakımından ucuz olmaları ve lokal optimumlara takılma sorunu olmadan parametre kestirimi yapabilmelerinden kaynaklanmaktadır. Önerdiğimiz algoritmalarının geçerliliğini göstermek için, insan edim videolarından çıkarılmış diziler üstünde sınıflandırma, insan hareket yakalama (motion capture) verileri ve internet ağı verilerinden çıkarılmış diziler üstünde topaklandırma deneyleri yapıyoruz. Neticede, önerdiğimiz spektral öğrenme tabanlı algoritmaların, bir çok uygulama sahasında, beklenti en büyütme algoritması yerine kullanılabilecek; yüksek topaklandırma başarısı gösteren ve hızlı algoritmalar olduğunu gösteriyoruz.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
1.1. Introduction . . . 1
1.2. Background . . . 4
1.3. Frequently used notation . . . 8
2. TIME SERIES MODELS . . . 10
2.1. Generative Time Series Models . . . 10
2.1.1. Markov Model . . . 10
2.1.2. Hidden Markov Model . . . 12
2.2. Time series clustering models . . . 14
2.2.1. Mixture of Markov Models . . . 15
2.2.2. Mixture of Hidden Markov Models . . . 17
2.3. Discriminative Time Series Models . . . 19
2.3.1. Discriminative Markov Model . . . 20
2.3.2. Discriminative Hidden Markov Model . . . 22
3. MAXIMUM LIKELIHOOD BASED LEARNING ALGORITHMS . . . 24
3.1. Expectation Maximization Algorithm . . . 24
3.2. Learning HMM with EM . . . 28
3.2.1. Inference in HMMs . . . 30
3.3. Learning mixture of Markov models with EM . . . 31
3.4. Learning mixture of HMMs with EM . . . 32
3.5. Training Discriminative Markov Model . . . 35
3.6. Training Discriminative HMM . . . 36

3.6.1. Inference in discriminative HMM . . . . . 38
4. METHOD OF MOMENTS BASED LEARNING ALGORITHMS . . . . . 40
4.1. Spectral Learning for Mixture Models . . . . . 43
4.2. Spectral Learning for Hidden Markov Models . . . . . 48
4.2.1. Parameter Learning in HMMs . . . . . 52
4.3. Spectral Learning for Mixture of Markov Models . . . . . 55
4.3.1. Spectral Learning for a Mixture of Dirichlet Distributions . . . . 59
4.4. Incorporating Spectral Learning for learning HMM mixtures . . . . . 61
4.4.1. Learning Infinite Mixtures of HMMs using spectral learning . . 63
5. EXPERIMENTS AND RESULTS . . . . . 67
5.1. Sequence Clustering via mixture of Markov models . . . . . 67
5.1.1. Synthetic Data . . . . . 67
5.1.2. Real World Data . . . . . 68
5.1.3. Finding number of clusters . . . . . 69
5.2. Sequence clustering via learning mixtures of HMMs . . . . . 73
5.2.1. Toy Problem . . . . . 73
5.2.2. Clustering Motion Capture Data with Finite Mixture Models . . 74
5.2.3. Application: Human Action Recognition from Videos with HMMs 75
5.3. Generative vs. Discriminative Models . . . . . 77
6. CONCLUSIONS . . . . . 81
6.1. Spectral learning algorithm for mixture of Markov models . . . . . 81
6.2. Incorporating spectral learning in learning mixtures of Hidden Markov models . . . . . 82
6.3. Discussion on Spectral Learning and EM . . . . . 84
APPENDIX A: Generative Models and Discriminative Models . . . . . 85
APPENDIX B: Gibbs Sampling for an Infinite mixture of HMMs using auxiliary parameter method . . . . . 89
B.1. Gibbs Sampling in HMM . . . . . 90
APPENDIX C: EM algorithm for learning mixture of Dirichlet distributions . 93
REFERENCES . . . . . 94

# LIST OF FIGURES

Figure 1.1. Illustration of sequence classification/clustering task. . . . . 1

Figure 1.2. KTH Human Action Database. . . . . 2

Figure 1.3. DAGs of Markov model and HMM. . . . . 6

Figure 1.4. DAGs of Markov models and mixture of HMMs. . . . . 6

Figure 1.5. Undirected graphs of discriminative Markov model and discriminative HMM. . . . . 7

Figure 2.1. Example sequences generated from Markov model. . . . . 11

Figure 2.2. Example observation sequence generated from a HMM with a Gaussian observation model. . . . . 14

Figure 2.3. Example observation sequence generated from a HMM with a discrete observation model. . . . . 14

Figure 2.4. Example time series data generated from mixture of Markov models. 16

Figure 2.5. Transition matrices used to generate the data in Figure 2.4. . . . . 16

Figure 2.6. Example time series data generated from mixture of HMMs. . . . 18

Figure 2.7. Transition matrices and observation matrices used to generate the data in Figure 2.6. . . . . 18

Figure 3.1. Expectation Maximization Algorithm. . . . . 26

Figure 3.2. Joint distribution $p(x_1, x_2)$ used in the toy problem for demonstration of the EM algorithm. . . . . 27

Figure 3.3. EM runs with two different initializations. . . . . 28

Figure 4.1. Parameter estimation from $N$, i.i.d. data from $\mathcal{G}(x_n; a = 1, b = 5)$. 42

Figure 4.2. DAG of a mixture model. . . . . 44

Figure 4.3. Spectral Learning Algorithm for Mixture Model. . . . . 47

Figure 4.4. Spectral Learning Algorithm for HMM. . . . . 54

Figure 4.5. DAG of the mixture of Dirichlet distributions. . . . . 60

Figure 4.6. Algorithm for clustering sequences via spectral learning of mixture of Dirichlet distributions. . . . . 61

Figure 4.7. Algorithm for learning a mixture of HMMs with spectral learning. 62

Figure 4.8. Learning an infinite mixture of HMMs with spectral learning . . . 65

Figure 5.1. Comparison of clustering accuracies on synthetic data. . . . . 68

Figure 5.2. Exploratory data analysis on network traffic data. . . . . 70

Figure 5.3. Finding the number of clusters on various data types. . . . . 72

Figure 5.4. Clustering handwritten L, V, 7. . . . . 73

Figure 5.5. Clustering handwritten 7 and 1. . . . . 74

Figure 5.6. Representative examples from clustering of "punching" and "kicking" sequences. . . . . 75

Figure 5.7. Example scenes from KTH Human Action Database [1]. . . . . 76

Figure 5.8. Feature extraction process for human action videos. . . . . 77

Figure 5.9. Sequence data used in [2]. . . . . 79

Figure 5.10. Number of data items used in training vs. classification accuracy of generative and discriminative models. . . . . 80

Figure A.1. Logistic Regression vs. Naive Bayes Classifiers. . . . . 87

# LIST OF TABLES

Table 5.1. Classification accuricies of three clustering algorithms on network traffic data. . . . . 71

Table 5.2. Speed comparison of Hard EM, Soft EM and Spectral algorithm. . 75

Table 5.3. Confusion matrices obtained from human action video classification. 78

Table 5.4. Confusion matrices obtained from classification experiment for generative and discriminative Markov models. . . . . 78

# LIST OF SYMBOLS

| | |
|---|---|
| $A$ | Transition matrix of a Markov chain |
| $\mathbb{E}_{p(x)}[x]$ | Expectation of a variable $x$ with respect to density $p(x)$ |
| $h_n$ | Class/cluster indicator of $n$'th data item |
| $I$ | Identity matrix |
| $K$ | Number of clusters in a mixture model |
| $L$ | Cardinality of the observation set in discrete model, dimensionality of observations in a continuous model |
| $\mathcal{M}_x$ | Forward messages in HMM, in matrix form |
| $O$ | Observation matrix |
| $p(x)$ | Probability distribution of a variable $x$ |
| $P_2$ | Second order empirical statistics |
| $P_3$ | Third order empirical statistics |
| $\mathcal{PO}(.)$ | Poisson distribution |
| $\mathcal{N}(.)$ | Gaussian distribution |
| $\mathcal{Q}(\theta, \theta^*)$ | EM lower bound computed with parameters $\theta^*$ |
| $q(x)$ | Variational distribution with variable $x$ in EM algorithm |
| $r_{1:T}$ | Latent state sequence of a hidden Markov model |
| $\mathbf{x}_{1:N}$ | Sequence dataset of $N$ data items |
| $x_{1:T}$ | Observation sequence of length $T$ |
| | |
| $\alpha(r_t)$ | Forward message in HMM inference |
| $\beta(r_t)$ | Backward message in HMM inference |
| $\eta$ | Vector to increase the robustness of a spectral algorithm |
| $\theta$ | Parameter set of a particular model |
| $\lambda$ | Intensity parameter of a Poisson distribution |
| $\mu$ | Mean parameter of a Gaussian distribution |
| $\pi$ | Initial state of a Markov chain |
| $\sigma$ | Standard deviation of a Gaussian distribution |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| DAG | Directed Acyclic Graph |
| EM | Expectation Maximization |
| HMM | Hidden Markov Model |
| KL | Kullback-Leibler Divergence |
| ML | Maximum Likelihood Estimation |

# 1. INTRODUCTION

## 1.1. Introduction

A significant portion of the data today can be found in sequential form. That is, the data is in an ordered manner, capturing the sequential information in addition to the information stored in each individual data item. Some relevant examples are speech signals, protein sequences, hand trajectories, human action videos. In this thesis, we are interested in classification and clustering of time series. Overall, we want to be able to teach a machine so that it is able to assign a discrete class (or cluster) label to an observed sequence. In Figure 1.1, we illustrate the sequence classification concept: Each class/cluster is shaded with a distinct color. We see that the sequences within a particular class share a common pattern. For instance, the class shaded by turquoise is mostly formed of low frequency sinusoids and the class on the top right is mostly formed of triangular waves.

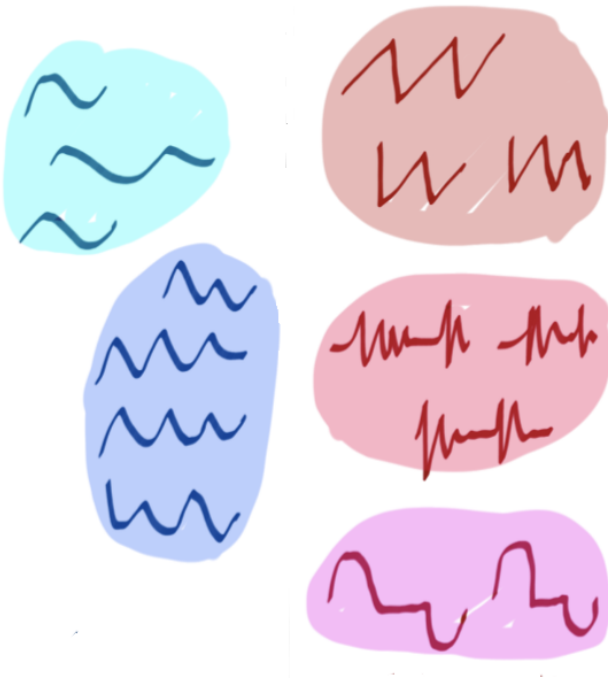


Figure 1.1. Illustration of sequence classification/clustering task (Courtesy of A. T. Cemgil).

Although the sequences in the same class seem to be similar to each other, they are not exactly the same. There are variations in duration, amplitude, phase, and each sequence may be contaminated by noise in various ways. Therefore, a simple heuristic is not sufficient to implement a robust sequence classifier. In Figure 1.2, we see some example scenes from the KTH human action database [1]. We observe that data instances within a class can show a significant amount of variability. Note that humans can successfully recognize human actions, despite the significant variations present in data items. It is possible to distinguish them, since they share a common pattern.

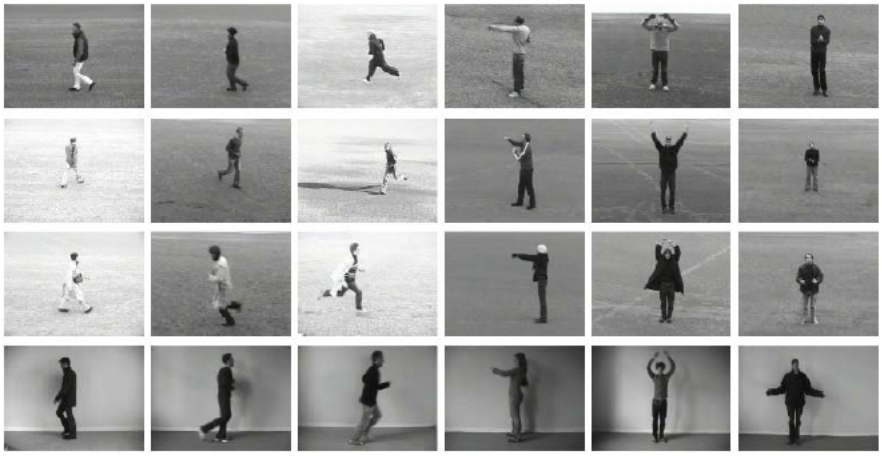


Figure 1.2. KTH Human Action Database [1]. Each column is reserved for a distinct action class. Action classes from left to right: Walking, Jogging, Running, Boxing, Hand waving and Hand Clapping.

So, the aim in these types problems should be to infer some shared structure from the sequences, in order to be able to successfully group them in classes/clusters. With this purpose in mind, we use probabilistic modeling in this thesis. We model the sequences in form of probability distributions so as to be able to model the natural variation in addition to the noise that might be present.

The probabilistic classification models can be categorized as supervised and unsupervised models. We call the supervised classification simply as "classification" and unsupervised classification as "clustering", conforming with the convention in the machine learning literature. Supervised methods make use of a available labeled training set, to "train" the models. Whereas unsupervised models, cluster the data without

using labeled training data. As expected, in practice supervised models yield superior classification performances compared to the unsupervised models. However, not in all applications a labeled training data is available. Moreover, we may be interested in discovering novel patterns that may not be present in the training data.

The main methodological contribution of this thesis is to develop novel learning algorithms for mixture of Markov models and mixture of hidden Markov models, which are time series clustering models. Mixture models are special type of latent variable models which require the usage of local search algorithms such expectation maximization (EM) algorithm. The EM algorithm can only provide locally optimum solutions, with indefinite number of iterations. In this thesis we propose using the recently popularized method of moments based spectral learning methods in learning mixture of Markov models and mixture of hidden Markov models, which give local optima free solutions by using only some observable moments.

We show that [3], the spectral learning approach in [4] as applied to mixture of Markov models would require the usage of a fifth order observable moment. Then, we consider an alternative learning scheme, inspired by a hierarchical Bayesian viewpoint, that only requires the second order moment. We experimentally show on synthetic data that, this approach is superior than the standard spectral learning algorithm in terms of the required number of samples. We also conduct experiments on real world data such as network traffic data and human motion capture capture data to show the validity of our approach.

We also propose an algorithm [5], for learning a mixture of Hidden Markov models, which uses spectral learning in the parameter estimation step. This algorithm results in a computationally cheaper alternative to a standard EM algorithm for HMM mixtures. We also extend our algorithm in [6], to handle infinite mixture of hidden Markov models, which do not require the user to specify the number of clusters of HMMs before running the algorithm. We perform experiments on human action recognition sequences extracted from videos and motion capture data to show that our clustering algorithm performs well.

There are two types of supervised classification, which are generative and discriminative classification [7]. In generative classification, we model the probability distribution of the data. That is, we model the data generation process. In discriminative models, we directly model the posterior of the class labels. This increases the ability of the model to distinguish between classes. In this thesis, we study the difference of generative and discriminative modeling in time series modeling by surveying Markov model, hidden Markov model and their discriminative counterparts. We discuss the differences between generative and discriminative models in more detail, in Appendix A. A performance comparison of generative and discriminative models are made on the sequence dataset in [2].

## 1.2. Background

According to the survey in [8], there are three main approaches in the literature for sequence classification, which are feature based classification, distance based classification and model based classification.

In the feature based classification, a number of features is extracted from the sequences. A prominent example for such a feature can be n-grams statistics. If $n = 2$, this corresponds to counting the transitions (bi-gram statistics). For instance, for the following short base sequence $AGCTTCGC$, the bigram sequence would be $AG$, $GC$, $CT$, $TT$, $TC$, $CG$, $GC$. This would result in the following $4 \times 4$, bi-gram statistics matrix:

| | $A$ | $T$ | $G$ | $C$ |
|---|---|---|---|---|
| $A$ | 0 | 0 | 0 | 0 |
| $T$ | 0 | 1 | 0 | 1 |
| $G$ | 1 | 0 | 0 | 1 |
| $C$ | 0 | 1 | 2 | 0 |

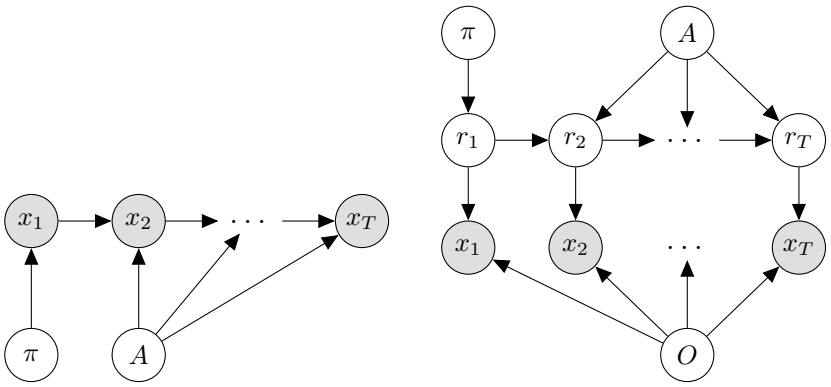
Then, this matrix (possibly a normalized version of this matrix so that the the elements add up to one) is input to a generic classifier such as support vector machines,

or decision trees. Some examples regarding this approach can be found in [9, 10]. There are other feature extraction mechanisms such as choosing a representative short segment from each sequence, so that a good classification accuracy is obtained [11]. These approaches may suffer from generalization issues as the extracted features alone may not be totally representative.

The distance based classification defines a similarity measure between the sequences. If the sequences are of the same length, this simply amounts to computing a valid distance function for two sequences, such as Euclidean distance [12], so that a simple K-nearest-neighbor classifier can be applied. This approach however requires the sequences to be of the same length. If not, a procedure called dynamic time warping can be used to align the sequences [13], which can be computationally prohibitive for a database with large number of sequences.

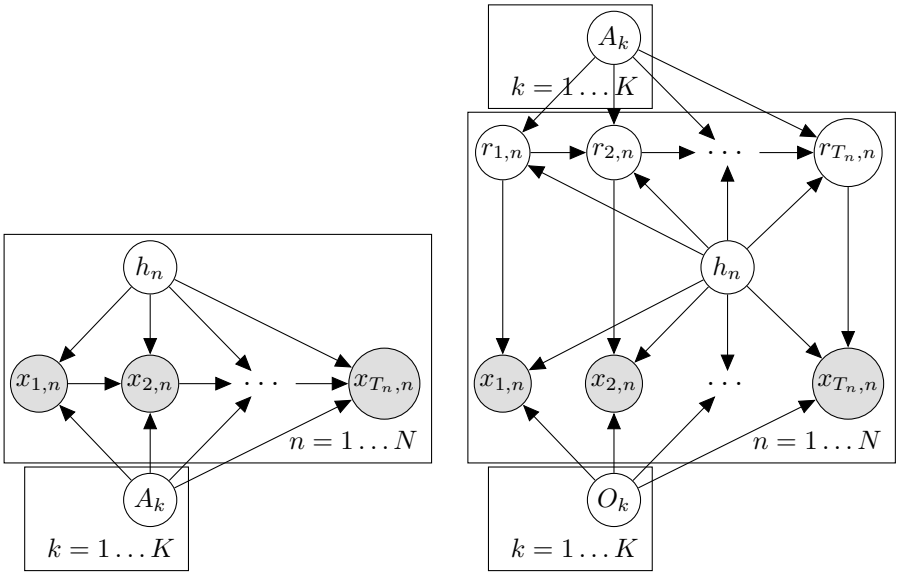
The model based classification is the approach that will be taken in this thesis. In model based classification, we define a probabilistic model to either model the generation process of the data (generative models), or simply to classify them (discriminative model). A detailed discussion on generative and discriminative models is given in Appendix A. Let us first briefly consider the generative models we will use in this thesis (A detailed explanation of the models is given in Chapter 2.1). Generative models are represented using the directed acyclic graph (DAG) formalism [14]. The graphical models of Markov model and hidden Markov model are respectively given in Figure 1.3a and Figure 1.3b. Using graphical models helps us to better understand and design these models. Furthermore, they enable us to derive inference algorithms more easily by making the conditional independence structure of the models explicit.

We can for instance see from Figure 1.3a that, in Markov model an observation $x_t$ is generated conditioned on the observation $x_{t-1}$, and the transition matrix $A$ (Note that in graphical models, the observed variables are shown with gray shaded circles. Hollow circles represent the latent variables.). The benefit of using the graphical model formalism is more evident when we consider a more sophisticated model such as HMM. For instance we see that, conditioned on $r_t$, three consecutive observations $x_{t-1}$, $x_t$ and




a. Markov Model b. HMM

Figure 1.3. DAGs of Markov model and HMM.

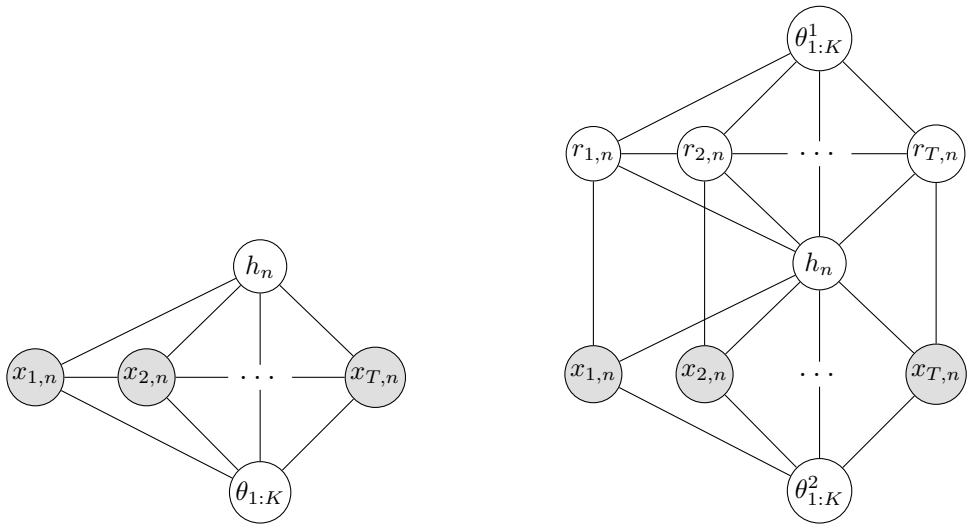



a. Mixture of Markov models b. Mixture of HMMs

Figure 1.4. DAGs of Markov models and mixture of HMMs.

$x_{t+1}$ become conditionally independent. We make use of this property in inference in Section 3.2.1, to show that the posterior $p(r_t|x_{1:T})$ can be computed via forward-backward algorithm [14]. Now, let us modify these models slightly so that we obtain unsupervised classification (clustering) models, mixture of Markov models and mixture of HMMs respectively in Figure 1.4a and Figure 1.4b.

We see in Figure 1.4a and Figure 1.4b that by adding a cluster indicator variable $h_n$, and replicating the variables $x_t$ and $r_t$, $N$ times using the plate notation, we can represent the generation process of a dataset consisting of $N$ data items, each having a Markov model or HMM structure. We can represent the discriminative versions of




a. Discriminative Markov model b. Discriminative HMM

Figure 1.5. Undirected graphs of discriminative Markov model and discriminative HMM.

Markov model and hidden Markov model using an *undirected* graph, which are given respectively in Figure 1.5a and Figure 1.5b.

In discriminative models, we do not use a directed model, because unlike generative models, we do not model the distribution of the observations $x_{1:T}$. We only consider them as inputs to an energy function $\psi(.)$, which produces a "score" of a particular class, given the observations $x_{1:T}$. The details are given in Section 2.3. All in all, we conclude that using a model based approach provides a principled approach for modeling, learning, and consequently inference.

As mentioned in previous section, in this thesis we propose novel spectral learning algorithms for learning mixture of Markov models and mixture of HMMs. In the spectral learning literature, which can be considered very young, there exist algorithms for inference in HMMs [15] and learning in multi-view mixture models [4,16]. The latter also proposes an algorithm for learning HMMs via converting the HMM into a multi-view mixture model. These algorithms are reviewed in Chapter 4.

In Section 4.3, using the tensor algebra defined in [17], we show that learning a mixture of Markov models via the standard procedure in [4], requires the usage of

moments up to order five. So, we propose an alternative scheme based on learning a mixture of Dirichlet distributions in Section 4.3.1 inspired by hierarchical Bayesian modeling, which only requires a second order moment.

In Section 4.4, we propose a mixture of HMMs learning algorithm, which uses spectral learning algorithm in [4] as a subroutine of a k-means clustering algorithm. We also extend this algorithm to handle an infinite mixture of HMMs. In infinite mixture models [18], we take the number of cluster to infinity, which enables the algorithm to automatically determine the number of clusters. An infinite mixture of HMMs would require an intractable joint integral and summation over the model parameters and latent state sequences, as shown in Section 4.4.1. Using spectral learning for this task yields a simple and fast learning algorithm, which is analogous to the Dirichlet-process means algorithm in [19], an infinite analogue of the standard k-means clustering algorithm for mixture of Gaussians.

The organization of the thesis is as follows: In Chapter 2 we introduce the time series models that we use in the thesis. We describe the maximum likelihood based learning algorithms in Chapter 3. We introduce the method of moments approaches in Chapter 4. In Chapter 5, we give experimental results. We conclude with Chapter 6.

### 1.3. Frequently used notation

We used squared brackets $[x = y]$ for the indicator function. If the argument inside is true, indicator function returns 1. Otherwise, it returns 0. In this case, if $x$ is equal to $y$, the function returns 1. We use MATLAB notation to pick a particular column or row of a matrix. For example, $O(:, k)$ denotes the $k$'th column of $O$ matrix. Similarly, $O(k, :)$ denotes the $k$'th row of $O$ matrix. We also use the same notation for tensors. The notation $A(i, :, j)$ tells us to take all entries of $A$ in second dimension corresponding to $i$'th and $j$'th entries in first and third dimension. Throughout the thesis, we switch between subscript notation and MATLAB notation, as necessary. The outer product operator $\otimes$ is defined as (using MATLAB notation), $(a \otimes b)(i, j) = a(i)b(j)$, where, $a \in \mathbb{R}^K$, $b \in \mathbb{R}^L$ and $a \otimes b \in \mathbb{R}^{K \times L}$. We denote a sequence of length $T$ as

$x_{1:T} = \{x_1, x_2, \ldots, x_T\}$. We use boldface $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ to denote a sequence dataset of $N$ sequences. We denote the estimate as $\theta^*$ for parameter $\theta$. The notation $=^+$ is used to denote that equality is satisfied with the addition of some constants, e.g. $f(\theta) =^+ \theta$ is equivalent to writing $f(\theta) = \theta + c$, where $c$ is a constant independent of variable $\theta$.

# 2. TIME SERIES MODELS

In this chapter, we give the detailed definitions of generative and discriminative models used in this thesis. We first describe the generative time series models, Markov model and HMM. Then, we introduce time series clustering models mixture of Markov models and mixture of HMMs. Finally we give the discriminative versions of Markov model and HMM.

## 2.1. Generative Time Series Models

As discussed in Appendix A, for doing sequence classification with generative models (generative classification), we simply train a model for each class in training via solving the problem defined in Equation A.1. When it comes to testing, we assign the data item $x^{test}$ to the class $\hat{c}$ that has the best fitting set of parameters $\theta_{class}$:

$$\hat{c} = \arg\max_{k} p(x^{test}|\theta_k) \tag{2.1}$$

The "goodness" of a the model parameter $\theta_c$ is as high as this probability, so the class that has the highest probability parameter $\theta_c$ is the assigned class. In this thesis, we use two generative models for sequence classification: Markov model (in Section 2.1.1) and Hidden Markov model (in Section 2.1.2).

### 2.1.1. Markov Model

In a Markov model, a generated observation sequence is a first order Markov chain. The likelihood of an observation sequence $\mathbf{x} = x_{1:T} = \{x_1, x_2, \ldots, x_T\}$, given

the model parameters $\theta = \{A, \pi\}$, is defined as follows:

$$\begin{aligned} p(x_{1:T}|\theta) &= \prod_{t=1}^{T} p(x_t|\ x_{t-1}) \\ &= \prod_{l=1}^{L} \pi_l^{[x_1=l]} \prod_{t=2}^{T} \prod_{l_1=1}^{L} \prod_{l_2=1}^{L} A_{l_1,l_2}^{[x_t=l_1][x_{t-1}=l_2]} \end{aligned} \tag{2.2}$$

where, the observations are dependent on each other in a chain fashion: In order to generate the observation at time instant $t$, $x_t$, we need the previous observation $x_{t-1}$. Note that observation $x_t$ is a discrete variable, and $x_t \in \{1, \ldots, L\}$. The model parameters are defined as follows:

- $A(u, v) := p(x_t = u|\ x_{t-1} = v)$, for $t \geq 2$, time invariant observation transition matrix
- $\pi(u) := p(x_1 = u|x_0) = p(x_1 = u)$, initial observation distribution

The directed graph model corresponding to a Markov model is given in Figure 1.3a. Two example sequences generated from a Markov model are given in Figure 2.1.

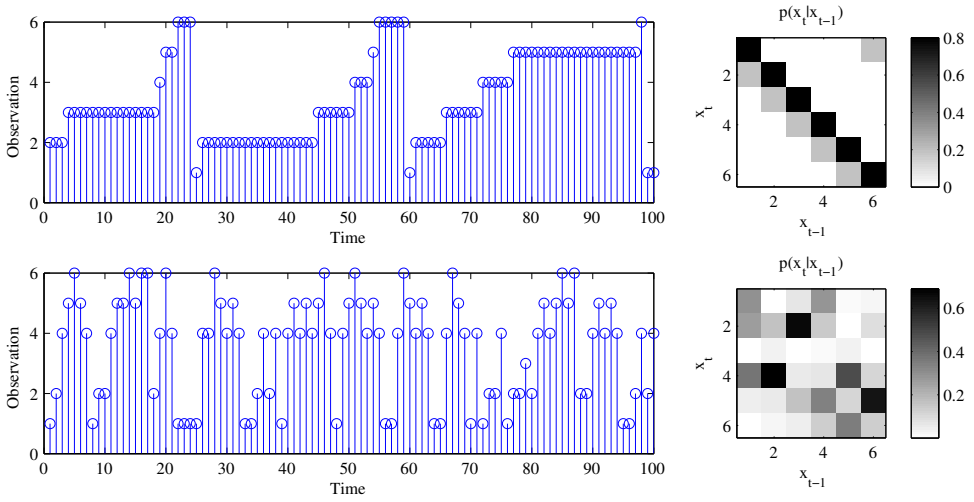



Figure 2.1. Example sequences generated from Markov model. Corresponding transition matrices $p(x_t|\ x_{t-1})$ are given next to the sequences. We see the effect of the transition matrix on the generated sequence.

A maximum likelihood estimate for parameters of a Markov model can be made in closed form. So, there is no need for an iterative algorithm such as EM. As the intuition suggests, it can be shown that learning the transition matrix $A$ given some sequences $\mathbf{x}_{1:N}$ simply amounts to counting the state transitions and then normalizing the empirical counts so that we have a conditional distribution. Similarly, according to the empirical counts for the first observation, we can estimate $\pi$. We omit the derivation because of its similarity to derivation of the EM algorithm for learning mixture of Markov models.

### 2.1.2. Hidden Markov Model

In Hidden Markov Models (HMMs), we do not observe a Markov chain directly, but instead have a hidden Markov chain $r_{1:T}$, with $r_t \in \{1, \dots M\}$, and we observe a sequence $\mathbf{x} = x_{1:T}$ dependent on this latent state sequence. According to an HMM, given the model parameters $\theta = (O, A, \pi)$, the likelihood $p(x_{1:T}|\theta)$ of an observation sequence is defined as follows:

$$\begin{aligned} p(x_{1:T}|\theta) &= \sum_{r_{1:T}} p(x_{1:T}, r_{1:T}|\theta) \\ &= \sum_{r_{1:T}} \prod_{t=1}^{T} p(x_t|r_t) p(r_t|r_{t-1}) \end{aligned} \tag{2.3}$$

The model parameters are defined as follows:

- $\pi(u) = p(r_1 = u|r_0) = p(r_1 = u)$, the initial latent state distribution
- $A(u, v) = p(r_t = u|r_{t-1} = v), \quad \text{for} \quad t \geq 2$, latent state transition matrix
- $O(:, u) = \mathbb{E}[x_t|r_t = u]$, observation matrix

where, $\pi \in \mathbb{R}^M$, $A \in \mathbb{R}^{M \times M}$ and $O \in \mathbb{R}^{L \times M}$. A column $O(:, u)$ of the observation matrix $O$ is defined as the expectation of the observation at time instant $x_t$, conditioned on the latent state. According to how one models $p(x_t|\ r_t)$, the observation matrix $O$ corresponds to some model parameters. Some frequently used modeling choices are as follows:

- Gaussian: $p(x_t|r_t = u) = \mathcal{N}(x_t; \mu_u, \sigma^2)$
  then, $O(:, u) = \mathbb{E}[x_t|r_t = u] = \mu_u$.
- Poisson: $p(x_t|r_t = u) = \mathcal{PO}(x_t; \lambda_u)$
  then, $O(:, u) = \mathbb{E}[x_t|r_t = u] = \lambda_u$.
- Discrete: $p(x_t|r_t = u) = \text{Discrete}(p_u)$
  then, $O(:, u) = \mathbb{E}[x_t|r_t = u] = p_u$.

where, the first choice is multivariate, isotropic Gaussian with mean $\mu_u \in \mathbb{R}^L$. As one might wonder, it is also possible to choose the observation density as a Gaussian with an arbitrary covariance matrix. For the sake of simplicity, we only consider the mean as a model parameter. Gaussian observation density is often used for applications with continuous observations. The second distribution is Poisson with intensity parameter $\lambda_u \in \mathbb{R}^L$. This choice is particularly useful for count data, as Poisson distribution is used to model count data. The last density is a generic discrete distribution, with parameter $p_u \in \mathbb{R}^L$. It is suitable for every application with discrete observations. However, since the model is quite flexible, one may suffer from overfitting, in a supervised classification scenario. The graphical model of a Hidden Markov model is given in Figure 1.3b.

An example data generated from a Hidden Markov model with Gaussian observation model is given in Figure 2.2 and discrete observation model in Figure 2.3. Given a sequence $x_{1:T}$, to learn the parameters $\theta = (O, A, \pi)$, of an HMM, we can maximize likelihood:

$$\theta^* = \arg\max_{\theta} p(x_{1:T}|\theta) = \arg\max_{\theta} \sum_{r_{1:T}} p(x_{1:T}, r_{1:T}|\theta) \tag{2.4}$$

However, since the likelihood $p(x_{1:T}|\theta)$ is defined via a summation over the latent state sequences $r_{1:T}$, we have to resort to some iterative optimization method. One of the most common ways is to iteratively maximize an EM lower bound:

$$Q(\theta^{old}, \theta^{new}) = \mathbb{E}_{p(r_{1:T}|\ x_{1:T}, \theta^{old})}[\log(p(x_{1:T}, r_{1:T}|\theta^{new}))] \tag{2.5}$$

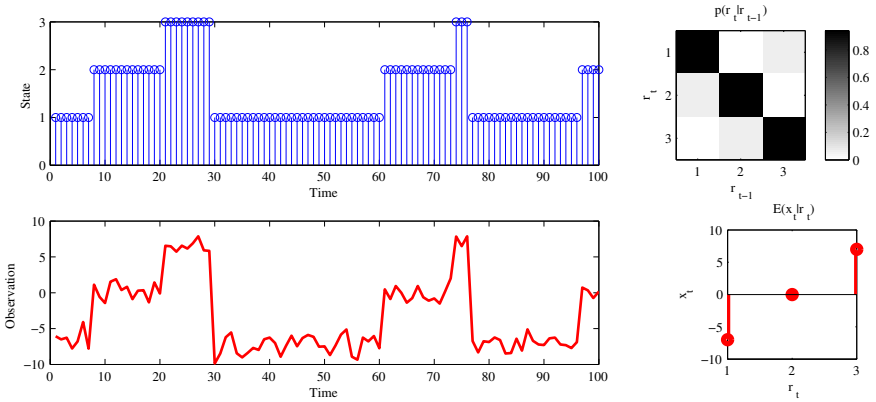



Figure 2.2. Example observation sequence generated from a hidden Markov model with a Gaussian observation model, with $L = 1$, $M = 3$. Corresponding transition matrix $p(r_t|\ r_{t-1})$ and observation matrix $\mathbb{E}[x_t|r_t]$ are given next to the sequences.

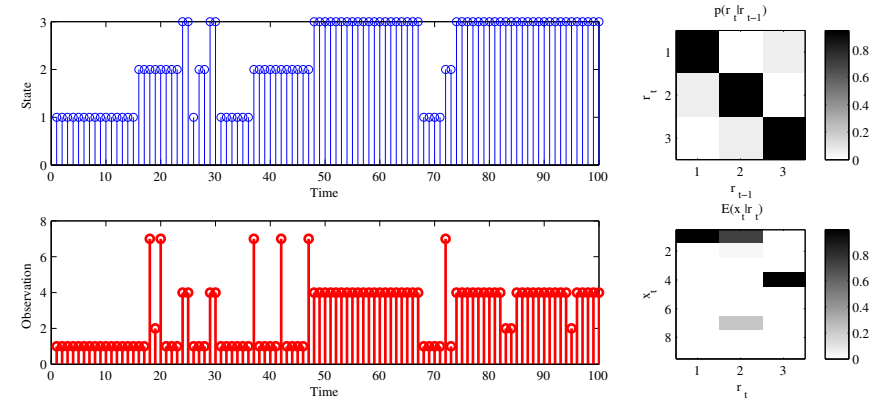



Figure 2.3. Example observation sequence generated from a hidden Markov model with a discrete observation model, with $L = 6$, $M = 3$.

However, this approach requires clever initialization for $\theta$ as the optimization problem does not have unique solution. In Section 4.2, we will discuss the local optima free spectral learning algorithms in [4, 15].

## 2.2. Time series clustering models

For clustering time series data with unsupervised models, an unlabeled dataset consisting of $N$ data items $x_{1:N}$ is input to the clustering algorithm. The algorithm returns the cluster labels $h^*_{1:N}$ as output. We use mixture Markov models (Section 2.2.1) and mixture of HMMs (Section 2.2.2) as time series clustering models. These models

are by nature generative models, since we model the data distribution $p(\mathbf{x}_{1:N}|\ \theta_{1:K})$.

### 2.2.1. Mixture of Markov Models

In Mixture of Markov models each data item is generated from one of the $K$ clusters of Markov models. The likelihood of a sequence is modeled as a convex combination of $K$ Markov models with transition matrices $A_{1:K}$. The likelihood of an observation sequence $\mathbf{x}_n = \{x_{1,n}, x_{2,n}, \ldots, x_{T_n,n}\ \}$ of length $T_n$ is defined as follows:

$$\begin{aligned} p(\mathbf{x}_n|\ A_{1:K}) &= \sum_{k=1}^{K} p(h_n = k) p(\mathbf{x}_n | h_n = k) \\ &= \sum_{k=1}^{K} p(h_n = k) \prod_{t=1}^{T_n} p(x_{t,n} | x_{t-1,n}, h_n = k) \\ &= \sum_{k=1}^{K} p(h_n = k) \prod_{t=1}^{T_n} \prod_{l_1=1}^{L} \prod_{l_2=1}^{L} A_{k,l_1,l_2}^{[x_{t,n}=l_1][x_{t-1,n}=l_2]} \end{aligned} \tag{2.6}$$

where, the variable $h_n \in \{1, 2, \ldots, K\}$ is the latent cluster indicator of a sequence $\mathbf{x}_n$. Note that $x_{t,n}$, the observation at time $t$ of sequence $n$ is a discrete variable which takes on values from $\{1, 2, \ldots, L\}$, and $p(x_t|x_{t-1}, h_n = k) = A_k \in \mathbb{R}^{L \times L}$ is the transition matrix of the sequences in cluster $k$. For the sake of simplicity, we omit the initial observation distribution $p(x_1, h_n) = \pi_{h_n}$. The corresponding graphical model is given in Figure 1.4a.

In this model, each sequence is generated using a transition matrix dependent on a cluster label. That is, we have a mixture of Markov chains where each mixture component is specified by a distinct transition matrix $A_k$. Example observation sequences are given in Figure 2.4 and corresponding transition matrices are given in Figure 2.5.

Given $N$ observation sequences $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, to learn the transition matrices of all clusters $A_{1:K}$, we can maximize the likelihood:

$$A_{1:K}^* = \arg\max_{A_{1:K}} p(\mathbf{x}_{1:N}|A_{1:K}) = \arg\max_{A_{1:K}} \sum_{h_{1:N}} \prod_{n=1}^{N} p(\mathbf{x}_n, h_n | A_{1:K}) \tag{2.7}$$

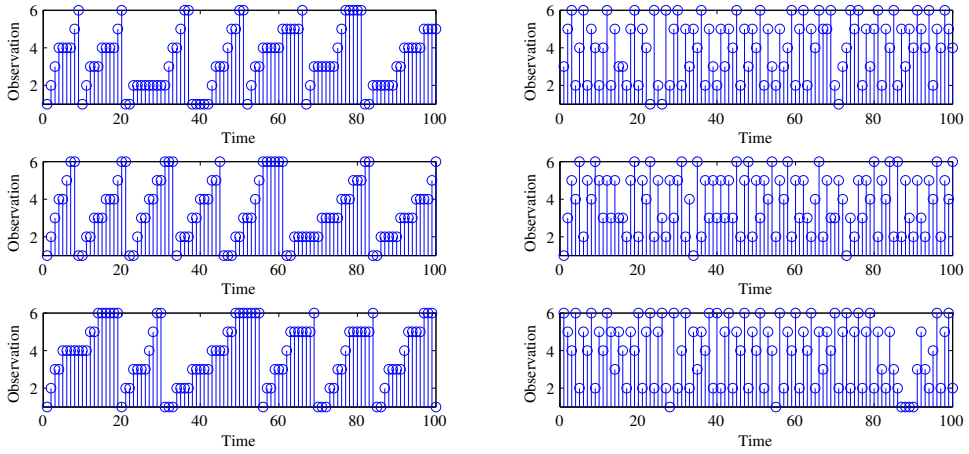



Figure 2.4. Example time series data generated from mixture of Markov models. The sequences generated from the first and second cluster are respectively given in first and second columns.

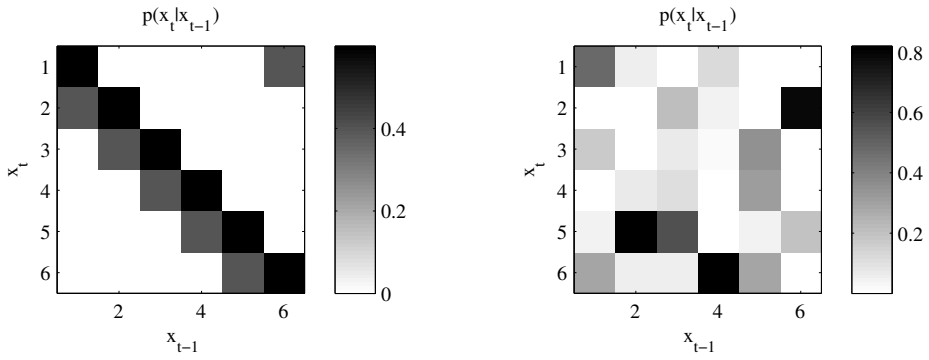



Figure 2.5. Transition matrices used to generate the data in Figure 2.4. The transition matrices of the first and second clusters are respectively given in first and second columns.

However, since the likelihood $p(\mathbf{x}_{1:N}|A_{1:K})$ is defined via a summation over all possible combinations of the cluster indicator variables $h_{1:N}$, we have to resort to some iterative optimization method. One of the most common ways is to iteratively maximize an EM lower bound:

$$Q(A_{1:K}^{old}, A_{1:K}^{new}) = \mathbb{E}_{p(h_{1:N}|\ \mathbf{x}_{1:N}, A_{1:K}^{old})}[\log(p(\mathbf{x}_{1:N}, h_{1:N}|A_{1:K}^{new}))] \tag{2.8}$$

However, this approach requires clever initialization for $A_{1:K}$ as the optimization problem does not have unique solution. In Section 4.3, we propose a local optima-free

alternative learning algorithm for learning mixture using the spectral learning algorithms in [4].

### 2.2.2. Mixture of Hidden Markov Models

In Mixture of HMMs each data item is generated from one of the $K$ clusters of HMMs. The likelihood of an observation sequence $\mathbf{x}_n = \{x_{1,n}, x_{2,n}, \ldots, x_{T_n,n}\}$ of length $T_n$ is modeled as a convex combination of $K$ hidden Markov models with parameters $\theta_{1:K} = (O_{1:K}, A_{1:K}, \pi_{1:K})$:

$$\begin{aligned} p(\mathbf{x}_n|\ \theta) &= \sum_{k=1}^{K} p(h_n = k) p(\mathbf{x}_n | h_n = k) \\ &= \sum_{k=1}^{K} p(h_n = k) \sum_{\mathbf{r}_n} p(\mathbf{x}_n, \mathbf{r}_n | h_n = k) \\ &= \sum_{k=1}^{K} p(h_n = k) \sum_{r_{1:T_n,n}} \prod_{t=1}^{T_n} p(x_{t,n}|r_{t,n}) p(r_{t,n}|r_{t-1,n}) \end{aligned} \tag{2.9}$$

where, $h_n \in \{1, 2, \ldots, K\}$ is the latent cluster indicator, and $\mathbf{r}_n = \{r_{1,n}, r_{2,n}, \ldots, r_{T_n,n}\}$ is the latent state sequence of the observed sequence $\mathbf{x}_n$. As discussed in Section 2.2.1, the observation model $p(x_{t,n}|r_{t,n})$, can be chosen flexibly, in accordance with the application's demands.

The corresponding graphical model is given in Figure 1.4b. Note that for the sake of simplicity, we omit the initial state distribution $p(r_{1,n}|h_n) = \pi_{h_n}$. In this model, each sequence is generated using a state transition matrix $A_k$ and an observation matrix $O_k$ dependent on the cluster indicator variable $h_n$. That is, we have a mixture of Markov chains where each mixture component is specified by the model parameters $\theta_k$. Example generated sequences from a mixture of HMMs with unit variance Gaussian observation model are given in Figure 2.6 and corresponding model parameters are given in Figure 2.7.

Given $N$ observation sequences $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, to learn the model pa-

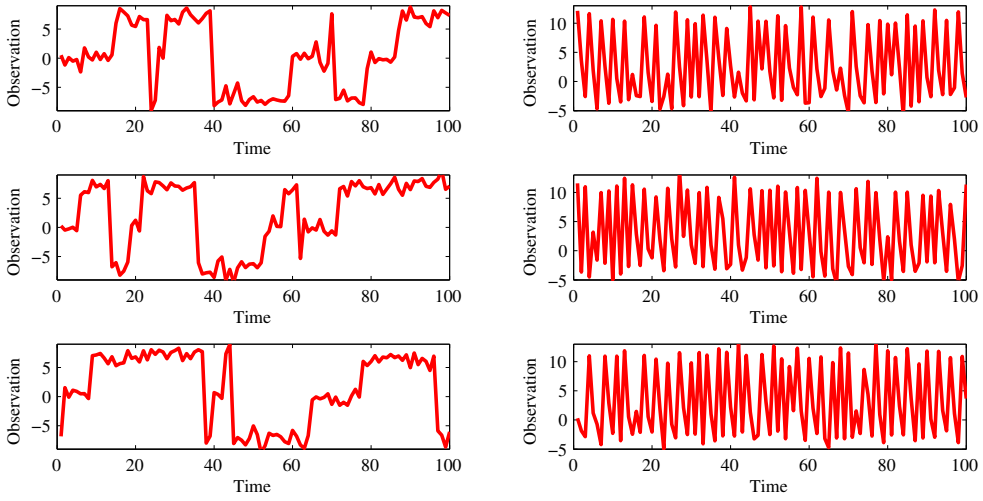



Figure 2.6. Example time series data generated from mixture of HMMs. The sequences generated from the first and second cluster are respectively given in first and second columns.

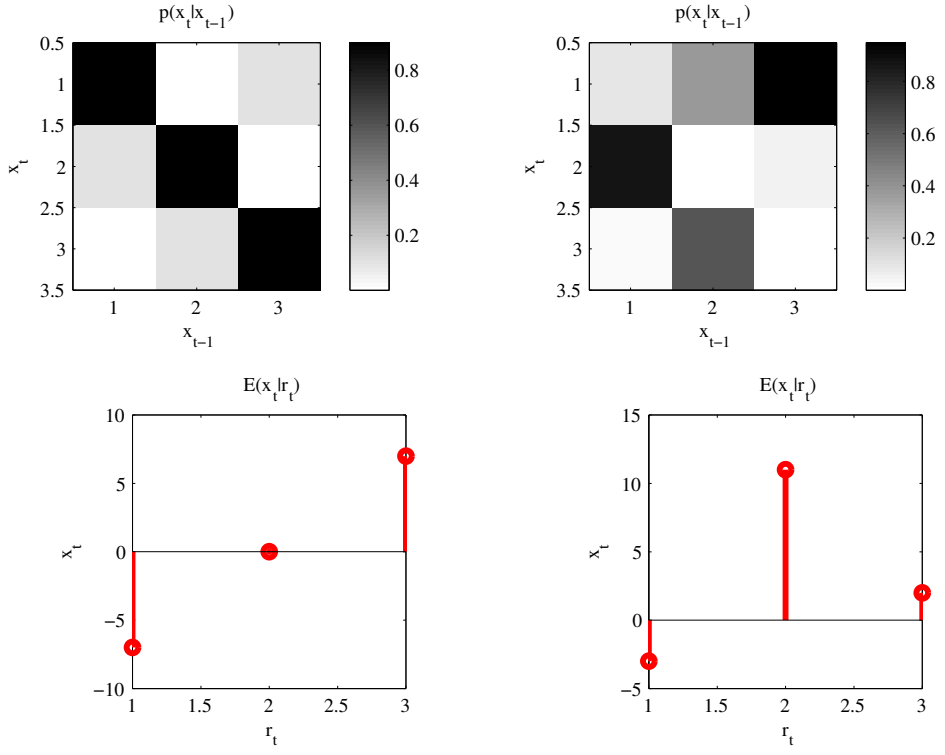



Figure 2.7. Transition matrices and observation matrices used to generate the data in Figure 2.6. The parameters of the first and second clusters are respectively given in first and second columns.

rameters of all clusters $\theta_{1:K}$, we can maximize the likelihood;

$$\theta^*_{1:K} = \arg\max_{\theta_{1:K}} p(\mathbf{x}_{1:N}|\theta_{1:K}) = \arg\max_{\theta_{1:K}} \sum_{h_{1:N}} \sum_{\mathbf{r}_n} \prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{r}_n, h_n|\theta_{1:K}) \tag{2.10}$$

However, since the likelihood $p(\mathbf{x}_{1:N}|\theta_{1:K})$ is defined via a summation over all possible combinations of the cluster indicator variables $h_{1:N}$, and for each sequence $x_n$ all possible combinations of the latent state sequence $\mathbf{r}_n$, we have to resort to some iterative optimization method. One of the most common ways is to iteratively maximize an EM lower bound:

$$Q(\theta^{old}_{1:K}, \theta^{new}_{1:K}) = \mathbb{E}_{p(h_{1:N}, \mathbf{r}_{1:N}|\ \mathbf{x}_{1:N}, \theta^{old}_{1:K})}[\log(p(\mathbf{x}_{1:N}, h_{1:N}|\theta^{new}_{1:K}))] \tag{2.11}$$

However, this approach requires clever initialization for $\theta_{1:K}$ as the optimization problem does not have a unique solution. A spectral learning algorithm that learns the model parameters $\theta_{1:K}$ is hard to derive because of the permutation ambiguity caused by the presence of two layers of hidden variables $h_n$ and $\mathbf{r}_n$. Instead, we propose using spectral learning algorithm as a subroutine in a k-means type algorithm in Section 4.4.

## 2.3. Discriminative Time Series Models

As discussed extensively in Appendix A, the reason for using discriminative models is to achieve the maximum class separability in training. That is, we optimize the model parameters so that the classes are separated maximally. This is equivalent to maximizing the product of the posterior of the class labels. In this section we derive the discriminative counterparts of Markov model and HMM. The general methodology to derive the discriminative equivalent of a generative model amounts to converting the model to an undirected graphical model by defining an appropriate energy function [20]. This way, the data $\mathbf{x}_{1:N}$ is only considered as an input to the model. The output of a discriminative model are the class labels $h_{1:N}$.

### 2.3.1. Discriminative Markov Model

To derive a learning algorithm for the discriminative Markov model, we first write the log-posterior of the class label $h_n$;

$$
\begin{aligned}
&\log p(h_n|\mathbf{x}_n, A_{1:K}) \\
&= \log \frac{p(\mathbf{x}_n, h_n|A_{1:K})}{\sum_{h_n} p(\mathbf{x}_n, h_n|A_{1:K})} \\
&= \log p(\mathbf{x}_n, h_n|A_{1:K}) - \log \sum_{h_n} p(\mathbf{x}_n, h_n|A_{1:K}) \\
&= \log\{p(h_n)p(\mathbf{x}_n|\ h_n, A_{1:K})\} - \log\{\sum_{h_n=1}^{K} p(h_n)p(\mathbf{x}_n|\ h_n, A_{1:K})\} \\
&= \log\{p(h_n)\prod_{t=1}^{T_n}\prod_{k=1}^{K}\prod_{l_1=1}^{L}\prod_{l_2=1}^{L} A_{k,l_1,l_2}^{[h_n=k][l_1=x_t][l_2=x_{t-1}]}\} \\
&\quad - \log\{\sum_{k=1}^{K} p(h_n)\prod_{t=1}^{T_n}\prod_{k=1}^{K}\prod_{l_1=1}^{L}\prod_{l_2=1}^{L} A_{k,l_1,l_2}^{[h_n=k][l_1=x_t][l_2=x_{t-1}]}\} \\
&= \log p(h_n) + \sum_{t=1}^{T_n}\sum_{l_1=1}^{L}\sum_{k=1}^{K}\sum_{l_2=1}^{L}[h_n=k][l_1=x_t][l_2=x_{t-1}]\log A_{k,l_1,l_2} \\
&- \log\left\{\sum_{k=1}^{K}\exp\left(\log p(h_n) + \sum_{t=1}^{T_n}\sum_{k=1}^{K}\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}[h_n=k][l_1=x_t][l_2=x_{t-1}]\log A_{k,l_1,l_2}\right)\right\}
\end{aligned}
\tag{2.12}
$$

Assuming uniform prior $p(h_n)$ for the sake of simplicity, and defining the model parameters as $\theta_{k,l_1,l_2} := \log A_{k,l_1,l_2}$, the log-posterior becomes;

$$
\begin{aligned}
&\log p(h_n = k|\mathbf{x}_n, A_{1:K}) \\
&\qquad = \log p(h_n = k|\mathbf{x}_n, \theta_{1:K}) \\
&\qquad = \sum_{t=1}^{T_n}\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}[h_n=k][l_1=x_t][l_2=x_{t-1}]\theta_{k,l_1,l_2} \\
&\qquad\quad - \log\left\{\sum_{h_n=1}^{K}\exp\left(\sum_{t=1}^{T_n}\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}[h_n=k][l_1=x_t][l_2=x_{t-1}]\theta_{k,l_1,l_2}\right)\right\}
\end{aligned}
\tag{2.13}
$$

The expression inside the exponential in Equation 2.13 is defined as the energy function $\psi(\mathbf{x}_n, h_n; \theta_k)$, and for discriminative Markov model, it is defined as follows;

$$\psi(\mathbf{x}_n, h_n; \theta_k) = \sum_{t=1}^{T_n} \sum_{k=1}^{K} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} [h_n = k][l_1 = x_t][l_2 = x_{t-1}]\theta_{k,l_1,l_2} \tag{2.14}$$

The likelihood $p(\mathbf{x}_n|h_n = k, \theta_{1:K})$ is in fact expressed as $\exp(\psi(\mathbf{x}_n, h_n = k; \theta_k))$. So, the overall procedure to derive the discriminative counterpart of a generative model mostly amounts to determining the corresponding energy function $\psi(\mathbf{x}_n, h_n; \theta_k)$. Notice that, deriving an energy function for a discriminative model is also equivalent to converting a directed graphical model of a generative model to an undirected one. The undirected graph corresponding to discriminative model is given in Figure 1.5a. Given the data $\mathbf{x}_{1:N}$ and class labels $h_{1:N}$, we simply solve the optimization problem in Equation 2.15 to learn the model parameters $\theta_{1:K}$:

$$\begin{aligned}
\theta^*_{1:K} &= \arg\max_{\theta_{1:K}} \prod_{n=1}^{N} p(h_n|\mathbf{x}_n, \theta_{1:K}) \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log p(h_n|\mathbf{x}_n, \theta_{1:K}) \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log \frac{\exp(\psi(\mathbf{x}_n, h_n; \theta_k))}{\sum_{h_n=1}^{K} \exp(\psi(\mathbf{x}_n, h_n; \theta_k))} \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \psi(\mathbf{x}_n, h_n; \theta_k) - \sum_{n=1}^{N} \log \sum_{h_n=1}^{K} \exp(\psi(\mathbf{x}_n, h_n; \theta_k))
\end{aligned} \tag{2.15}$$

We took the logarithm of $\prod_{n=1}^{N} p(h_n|\mathbf{x}_n, \theta_{1:K})$, since taking the logarithm makes it easier to differentiate with respect to the unknowns. Since the logarithm is a monotonic function, and does not change the maximizer, we were able to do this. Given $N$ sequences $\mathbf{x}_{1:N}$ and $N$ class labels $h_{1:N}$, we maximize this expression by gradient descent. We give the detailed derivation of the gradient descent updates in Section 3.5. One final note is that, we do not constrain the model parameter $\theta_k$ to be the logarithm of a probability table (a transition matrix in this case), as we do in Markov model. This is because of the reasons we discussed extensively in Appendix A: We do not care about the distribution (how we generate) the data, but our only concern is to obtain a good

decision boundary to separate the two classes.

### 2.3.2. Discriminative Hidden Markov Model

As we have discussed in the previous section, deriving the discriminative counterpart of a generative model amounts to deriving the corresponding energy function. The energy function of the discriminative counterpart of a hidden Markov model is specified as follows:

$$\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta) = \sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}[r_{t,n}=m_1][r_{t-1,n}=m_2][h_n=k]\theta^1_{k,m_1,m_2} + \sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{m=1}^{M}[x_{t,n}=l][r_{t,n}=m][h_n=k]\theta^2_{k,l,m} \quad (2.16)$$

Note that in this model $\theta_{1:K} = (\theta^1_{1:K} = \log A_{1:K}, \theta^2_{1:K} = \log O_{1:K})$. So, the first term basically stands for the state transitions and the second term is the discrete observation matrix. We have considered a discrete observation model, but it is possible to derive an energy function for arbitrary observation model. Given the data $\mathbf{x}_{1:N}$ and class labels $h_{1:N}$, we simply solve the optimization problem in Equation 2.17 to learn the model parameters $\theta_{1:K}$:

$$\begin{aligned}
\theta^*_{1:K} &= \arg\max_{\theta_{1:K}} \prod_{n=1}^{N} p(h_n|\mathbf{x}_n, \theta_{1:K}) \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log p(h_n|\mathbf{x}_n, \theta_{1:K}) \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log \frac{\sum_{\mathbf{r}_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_k))}{\sum_{h'_n=1}^{K}\sum_{\mathbf{r}_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h'_n; \theta_k))} \\
&= \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log \sum_{\mathbf{r}_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_k)) \\
&\qquad - \sum_{n=1}^{N} \log \sum_{h'_n=1}^{K}\sum_{\mathbf{r}_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h'_n; \theta_k))
\end{aligned} \quad (2.17)$$

This expression can be minimized using an iterative algorithm such as gradient descent where the necessary probability distributions are computed using forward-backward message passing. The details can be found in Section 3.6. The corresponding undirected graphical model of discriminative HMM is given in Figure 1.5b. Note that the class of discriminative models with a latent state sequence layer $\mathbf{r}_n$, and a class indicator $h_n$ is also called hidden conditional random field (HCRF) [21].

# 3. MAXIMUM LIKELIHOOD BASED LEARNING ALGORITHMS

In this chapter, we describe the maximum likelihood parameter learning algorithms for the models we introduced in Chapter 2. We also describe the theoretical and practical details of the expectation-maximization (EM) algorithm.

## 3.1. Expectation Maximization Algorithm

Expectation maximization algorithm [22] is a very popular algorithm for parameter learning in latent variable models. The problem setting in a typical Expectation-Maximization scenario is as follows:

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \log p(x|\theta) \\ &= \arg\max_{\theta} \log \sum_{h} p(x,h|\theta)\end{aligned} \tag{3.1}$$

where, $h$ is the latent variable over which we take a summation, $x$ is the observation, and $\theta$ is the parameter to be optimized. Since the likelihood $p(x|\theta)$ is defined with a summation over a latent variable $h$, which prevents the logarithm to act directly on the distribution, we have a complicated function to optimize. EM algorithm provides an alternative way for solving the maximum likelihood problem in latent variable models, by optimizing functions which are easier to optimize. Let us consider the log-likelihood $\log p(x|\theta)$: (Usage of log may seem arbitrary, but will become obvious shortly.)

$$\begin{aligned}\log p(x|\theta) &= \log \sum_{h} p(x,h|\theta) \\ &= \log \sum_{h} q(h) \frac{p(x,h|\theta)}{q(h)}\end{aligned} \tag{3.2}$$

where $q(h)$ is an arbitrary distribution on the latent variable $h$. Now, notice that we have a convex combination of the functions $\frac{p(x,h|\theta)}{q(h)}$ inside the logarithm. Since logarithm

is a concave function, we can apply the Jensen's inequality to obtain:

$$\begin{aligned}\log \sum_h q(h) \frac{p(x,h|\theta)}{q(h)} &\geq \sum_h q(h) \log \frac{p(x,h|\theta)}{q(h)} \\ &= \underbrace{\mathbb{E}_{q(h)}[\log p(x,h|\theta)] - \mathbb{E}_{q(h)}[\log q(h)]}_{\mathcal{Q}(\theta):=}\end{aligned} \tag{3.3}$$

So, we have found a lower bound $\mathcal{Q}(\theta)$ on the log-likelihood, where the first term is the expectation of logarithm of the joint distribution, which is generally easy to compute, and second term is the entropy of the $q(h)$ distribution, which is a constant with respect to model parameter $\theta$. The next thing is to maximize this bound with respect to the arbitrary distribution $q(h)$ for a fixed $\theta$.

*Claim:* The bound $\mathcal{Q}(\theta)$ is maximized for a fixed $\theta$ when $q(h) = p(h|x,\theta)$.

*Proof.*

$$\begin{aligned}\mathcal{Q}(\theta) =& \mathbb{E}_{q(h)}[\log p(x,h|\theta)] - \mathbb{E}_{q(h)}[\log q(h)] \\ =& \mathbb{E}_{q(h)}[\log p(h|x,\theta)] + \mathbb{E}_{q(h)}[\log p(x|\theta)] - \mathbb{E}_{q(h)}[\log q(h)] \\ =& \mathbb{E}_{q(h)}[\log p(x|\theta)] - KL(q(h)\|p(h|x,\theta)\ ) \\ =& \log p(x|\theta) - KL(q(h)\|p(h|x,\theta)\ )\end{aligned} \tag{3.4}$$

where, $KL(.)$ is the KL divergence. Since KL divergence is always positive, the bound $\mathcal{Q}(\theta)$ attains $\log p(x|\theta)$ iff $q(h) = p(h|x,\theta)$, which is the posterior of the hidden variable. □

Moreover, from Equation 3.4 we see that if $q(h) = p(h|x,\theta)$, we have a tight bound, i.e. at the $\theta$ value where the bound is computed, $\mathcal{Q}(\theta) = \log p(x|\theta)$. In order to sketch the overall algorithm let us denote the parameter $\theta$ with which we compute the bound $\theta^{old}$. That is, we define $\mathcal{Q}(\theta, \theta^{old}) := \mathcal{Q}(\theta^{old})$. (The bound is computed using

$\theta^{old}$, and it is a function of $\theta$.) We maximize this bound with respect to $\theta$, to find $\theta^{new}$:

$$\theta^{new} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{old}) \tag{3.5}$$

Then, we compute a new bound $\mathcal{Q}(\theta, \theta^{new})$, using $p(h|x, \theta^{new})$. The step where we compute a novel bound is called the $E$ step. The step we maximize this bound is called the $M$ step, and hence the name $EM$ algorithm. We repeat these steps until convergence. The overall algorithm is given in Figure 3.1.

Input: Sequences Data $x$
Output: Estimated model parameter(s) $\theta^*$
Randomly initialize model parameters $\theta^0$.
**for** $\tau = 1$ **to** $maxiter$ **do**
  E-Step: Compute the bound $\mathcal{Q}(\theta, \theta^{\tau-1})$ using the posterior $p(h|x, \theta^{\tau-1})$
  M-Step: Update the model parameters $\theta^{\tau}$ by maximizing the bound $\mathcal{Q}(\theta, \theta^{\tau-1})$
**end for**

Figure 3.1. Expectation Maximization Algorithm.

*Claim:* EM iterations increase (at worst do not decrease) the log-likelihood $\log p(x|\theta^{\tau})$.

*Proof.* Let $\mathcal{Q}(\theta^{\tau}, \theta^{\tau-1}) := \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\tau-1})$, then

$$\log p(x|\ \theta^{\tau-1}) = \mathcal{Q}(\theta^{\tau-1}, \theta^{\tau-1}) \leq \mathcal{Q}(\theta^{\tau}, \theta^{\tau-1}). \tag{3.6}$$

since we have found $\mathcal{Q}(\theta^{\tau}, \theta^{\tau-1})$ by maximizing $\mathcal{Q}(\theta, \theta^{\tau-1})$. We also know from Equation 3.4 for a fixed parameter $\theta^{\tau}$, the bound is maximized when the bound is computed using the posterior $p(h|x, \theta^{\tau})$. That is, updating the bound can only increase the bound value at $\theta^{\tau}$. So $\mathcal{Q}(\theta^{\tau}, \theta^{\tau-1}) \leq \mathcal{Q}(\theta^{\tau}, \theta^{\tau})$. Thus, we conclude that $\log p(x|\theta^{\tau-1}) = \mathcal{Q}(\theta^{\tau-1}, \theta^{\tau-1}) \leq \log p(x|\theta^{\tau}) = \mathcal{Q}(\theta^{\tau}, \theta^{\tau})$. □

Although, this gives us a local convergence guarantee, we do not have a global

convergence guarantee. For global convergence, we do need to have proper initializations to obtain good solutions. Let us demonstrate this on a toy problem. The problem is to find the maximizing $x_1$ for the marginal distribution $p(x_1)$:

$$x_1^* = \arg\max_{x_1} \sum_{x_2} p(x_1, x_2) \tag{3.7}$$

In the typical EM setting in Equation 3.1, $x_2$ is the latent variable ($h$), and $x_1$ is the parameter ($\theta$) to be optimized (For the sake of simplicity we do not have observation $x$). In Figure 3.2, we see the joint distribution $p(x_1, x_2)$. In the Figure 3.3, we see the EM iterations with bounds in each iteration. The log likelihood $\log p(x_1)$ is shown with a blue curve. As shown mathematically, the EM iterations do not decrease the log-likelihood $\log p(x_1)$, and EM bounds are tight to the log-likelihood, on the $x_1$ value with which it is computed. Furthermore, we see that we only have local convergence guarantee as we see that initializations from different $x_1$ values result in convergence to different locally optimal $x_1^*$.

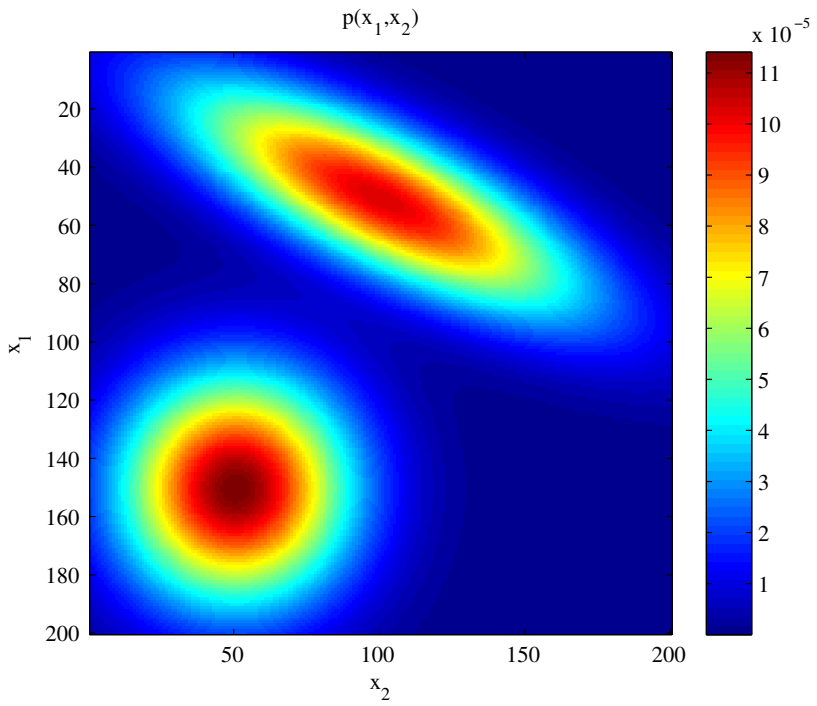
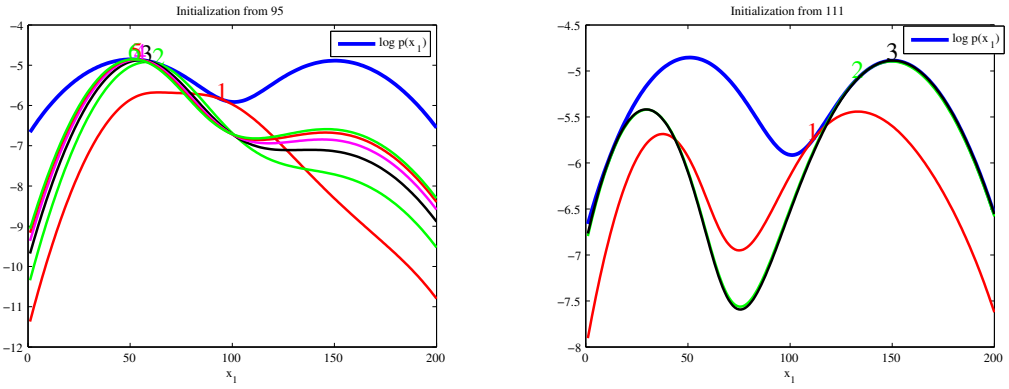



Figure 3.2. Joint distribution $p(x_1, x_2)$ used in the toy problem for demonstration of the EM algorithm.




a. EM iterations, initialization from $x_1 = 95$ b. EM iterations, initialization from $x_1 = 111$

Figure 3.3. EM runs with two different initializations. The log-likelihood $\log p(x_1, x_2)$ is shown with a blue curve. Bound computed at each iteration is enumerated with the current iteration count, with a corresponding color. We see that bounds are tight on the parameter value, with which they are computed.

### 3.2. Learning HMM with EM

In order to derive an EM algorithm for a latent variable model, mainly we have to do two things. First, we have to derive an EM lower bound $\mathcal{Q}(\theta, \theta^{old})$, which corresponds to the E-step. Then, we have to maximize this bound with respect to model parameters $\theta$, which corresponds to the M-step. Let us first write the EM lower bound for a Hidden Markov model:

$$
\begin{aligned}
\mathcal{Q}(\theta, \theta^*) =^{+} & \mathbb{E}_{p(r_{1:T}|x_{1:T},\theta^*)}[\log p(x_{1:T}, r_{1:T}|\theta)] \\
= & \mathbb{E}_{p(r_{1:T}|x_{1:T},\theta^*)}[\log \prod_{t=1}^{T} p(x_t|r_t, O)p(r_t|r_{t-1}, A)] \\
= & \sum_{t=1}^{T} \mathbb{E}_{p(r_t|\ x_{1:T},\theta^*)}[\log p(x_t|r_t, O)] + \sum_{t=1}^{T} \mathbb{E}_{p(r_t, r_{t-1}|\ x_{1:T},\theta^*)}[\log p(r_t|r_{t-1}, A)] \\
= & \sum_{t=2}^{T} \sum_{m=1}^{M} \mathbb{E}([r_t = m]) \log p(x_t|\ r_t, O(:, k)) \\
& + \sum_{t=2}^{T} \sum_{j=1}^{M} \sum_{k=1}^{M} \mathbb{E}([r_{t-1} = m_1][r_t = m_2]) \log A_{m_1, m_2}
\end{aligned}
\tag{3.8}
$$

Note that for the sake of simplicity we do not consider the initial state distribution $\pi$. Taking it uniform, generally works well in practice. In order to keep the notation

uncluttered, we do not show the distribution with respect to which we are taking expectations. The next thing is to maximize this lower bound with respect to, $A$, and $O$. Since this is an unconstrained optimization problem, we take derivatives with respect to $O$ and $A$ and equate the expressions to zero. (We use Lagrange multipliers to ensure that $A$ is a probability table.) The update equation for the transition matrix $A$ turn out to be as follows;

$$A^{new}_{m_1,m_2} = \frac{\sum_{t=2}^{T} \mathbb{E}([r_{t-1} = m_1][r_t = m_2])}{\sum_{t=2}^{T} \sum_{m_2=1}^{M} \mathbb{E}([r_{t-1} = m_1][r_t = m_2])} \tag{3.9}$$

intuitively, this update equation counts the expected state transitions: Since we do not observe $r_{1:T}$ directly, we compute an expectation, and count the expected transitions to update the state transition matrix $A$. Update equations for the observation matrix $O$. The update equations for the frequently used models used in Section 2.1.2 are as follows:

- Poisson; $O = \lambda$, the intensity parameter:

$$\lambda^{new}_m = \frac{\sum_{t=1}^{T} \mathbb{E}[r_t = m] x_t}{\sum_{t=1}^{T} \mathbb{E}[r_t = m]} \tag{3.10}$$

- Gaussian; $O = \mu$, the mean parameter:

$$\mu^{new}_m = \frac{\sum_{t=1}^{T} \mathbb{E}[r_t = m] x_t}{\sum_{t=1}^{T} \mathbb{E}[r_t = m]} \tag{3.11}$$

- Discrete; $O = p$, the discrete distribution:

$$p^{new}_{l,m} = \frac{\sum_{t=1}^{T} \mathbb{E}([r_t = m])[x_t = l]}{\sum_{t=1}^{T} \sum_{l'=1}^{L} \mathbb{E}([r_t = m])[x_t = l']} \tag{3.12}$$

### 3.2.1. Inference in HMMs

To implement these update equations, we of course need to compute the expectations $\mathbb{E}([r_t = m])$ and $\mathbb{E}([r_{t-1} = m_1][r_t = m_2])$. Note that, as mentioned before, to keep the notation uncluttered, we do not indicate the density with respect to which we take these expectations. In an EM algorithm, the expectations are always with respect to the posterior of the latent variable(s). In HMM, the posterior distribution is $p(r_{1:T}|\ x_{1:T}, \theta)$, the posterior of the latent state trajectory. Let us consider the expectation $\mathbb{E}([r_t = m])$.

$$\begin{aligned}\mathbb{E}([r_t = m]) =& \sum_{r_{1:T}} p(r_{1:T}|\ x_{1:T}, \theta)[r_t = m] \\ =& \sum_{r_t} p(r_t|\ x_{1:T}, \theta)[r_t = m] \\ =& p(r_t = m|\ x_{1:T}, \theta) \end{aligned} \tag{3.13}$$

So, we need the posterior of a single latent state variable $r_t$. The computation of this posterior can be seen to be daunting at first sight, since we have take a some over all possible trajectories (except $r_t$), which is a huge sum, consisting of $M^{T-1}$ possible states. However, due to the conditional independence properties of the HMM graph structure, we can in fact compute this posterior very efficiently using forward-backward algorithm. Let us elaborate more: (We drop $\theta$ in probabilities for uncluttered notation)

$$\begin{aligned} p(r_t|\ x_{1:T}) \propto&\quad p(x_{1:T}, r_t) \\ =& \underbrace{p(x_{1:t}, r_t)}_{:=\alpha(r_t)} \underbrace{p(x_{t+1:T}|r_t)}_{:=\beta(r_t)} \end{aligned} \tag{3.14}$$

where,

$$\alpha(r_t) = p(x_t|r_t) \sum_{r_{t-1}} p(r_t|r_{t-1}) \underbrace{p(x_{t-1}|r_{t-1}) \dots \underbrace{p(x_2|\ r_2) \sum_{r_1} p(r_2|r_1) p(x_1|r_1) \underbrace{p(r_1)}_{\alpha(r_1)}}_{\alpha(r_2)}}_{\alpha(r_{t-1})} \tag{3.15}$$

So, we see that we can recursively compute the $\alpha(r_t)$ (forward) message. Notice that, due to the Markovian property of the model, we are able to distribute the sums over the factor. This reduces the computational complexity to order $\mathcal{O}(M^2T)$ to compute all $\alpha$ messages. Similarly, the backward $\beta$ messages are;

$$\begin{aligned}\beta(r_t) &= \sum_{r_{t+1:T}} \prod_{t=t+1:T} p(x_t|r_t)p(r_t|r_{t-1}) \\ &= \sum_{r_{t+1}} p(r_t|r_{t+1})p(x_{t+1}|r_{t+1}) \dots \underbrace{\underbrace{\sum_{r_T} p(r_T|r_{T-1})p(x_T|r_T)\underbrace{1}_{\beta(r_T)}}_{\beta(r_{T-1})}}_{\beta(r_{t+1})}\end{aligned} \tag{3.16}$$

In the HMM literature product $\alpha(r_t)\beta(r_t) \propto p(r_t|x_{1:T})$ is called the $\gamma(r_t)$ message. Similarly, the expectation $\mathbb{E}([r_{t-1} = m_1][r_t = m_2])$ can be computed via $\alpha$ and $\beta$ messages. Details can be found in standard machine learning textbooks such as [14], so we do not reproduce the derivation here.

### 3.3. Learning mixture of Markov models with EM

According to the typical EM scenario as defined in Section 3.1, the latent variable is $h_{1:N}$, parameters to be optimized are the transition matrices $A_{1:K}$, observations are the sequences $\mathbf{x}_{1:N}$. Let us first derive the EM lower bound for the Markov model mixture:

$$\begin{aligned}\mathcal{Q}(A_{1:K}, A^*_{1:K}) &=^{+} \mathbb{E}_{p(h_{1:N}|\mathbf{x}_{1:N}, A^*_{1:K})}[\log p(\mathbf{x}_{1:N}, h_{1:N}|A_{1:K})] \\ &= \mathbb{E}_{p(h_{1:N}|\mathbf{x}_{1:N}, A^*_{1:K})}[\sum_{n=1}^{N} \log p(\mathbf{x}_n, h_n|A_{1:K})] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T_n}\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}[x_{t,n} = l_1][x_{t-1,n} = l_2]\mathbb{E}([h_n = k]) \log A_{k,l_1,l_2} \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l_1=1}^{L}\sum_{l_2=1}^{L} c^n_{l_1,l_2}\mathbb{E}([h_n = k]) \log A_{k,l_1,l_2}\end{aligned} \tag{3.17}$$

where, $c^n_{l_1,l_2} = \sum_{t=1}^{T_n}[x_{t,n} = l_1][x_{t-1,n} = l_2]$, the transition counts matrix for sequence $\mathbf{x}_n$. Note that again for the sake of simplicity, we do not include the initial observation distribution. In practice, taking it as uniform generally works well except for very short sequences. We maximize this bound using a Lagrange multiplier for the constraint $\sum_{l_1} A_{k,l_1,l_2} = 1$ to obtain the update equation for $A_{1:K}$:

$$A^{new}_{1:K} = \frac{\sum_{n=1}^{N} c^n_{l_1,l_2}\mathbb{E}([h_n = k])}{\sum_{l_1=1}^{N}\sum_{n=1}^{N} c^n_{l_1,l_2}\mathbb{E}([h_n = k])} \tag{3.18}$$

The expectation needed for the parameter update equation is:

$$\begin{aligned}\mathbb{E}([h_n = k]) &= \sum_{h_n=1}^{K} p(h_n|\mathbf{x}_{1:N}, A^*_{1:K})[h_n = k] \\ &= p(h_n = k|\mathbf{x}_{1:N}, A^*_{1:K}) = \frac{p(\mathbf{x}_n\ |h_n = k, A^*_k)p(h_n = k)}{\sum_{k=1}^{K} p(\mathbf{x}_n|h_n = k, A^*_k)p(h_n = k)}\end{aligned} \tag{3.19}$$

which tells us to compute the likelihood $p(\mathbf{x}_n|h_n = k, A^*_k)$ for each cluster $k$, and then by weighting the likelihoods by prior $p(h_n = k)$, we compute a cluster assignment probability vector for each data item $\mathbf{x}_n$. When the data cluster assignments are made with high confidence, that is generally when the EM iterations are close to convergence, probability vector becomes a vector with only one entry one and others zero.

## 3.4. Learning mixture of HMMs with EM

According to the typical EM scenario as defined in Section 3.1, the latent variables are; latent cluster indicators $h_{1:N}$ and latent state sequences $\mathbf{r}_{1:N}$, parameters $\theta_{1:K}$ to be optimized are the transition matrix of each cluster $A_{1:K}$, and observation matrix of each cluster $O_{1:K}$, observations are the sequences $\mathbf{x}_{1:N}$. Let us first derive the EM lower bound for the mixture of HMMs: Let us use $q(\mathbf{r}_{1:N}, h_{1:N}) = p(\mathbf{r}_{1:N}, h_{1:N}|\mathbf{x}_{1:N}, \theta^*_{1:K})$ to

keep the notation clutter minimal.

$$
\begin{aligned}
&\mathcal{Q}(\theta_{1:K}, \theta^*_{1:K}) =^+ \mathbb{E}_{q(\mathbf{r}_{1:N}, h_{1:N})}[\log p(\mathbf{x}_{1:N}, \mathbf{r}_{1:N}, h_{1:N}|\theta_{1:K})] \\
=&\mathbb{E}_{q(\mathbf{r}_{1:N}, h_{1:N})}[\sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{r}_n, h_n|\theta_{1:K})] \\
=&\sum_{n=1}^{N} \mathbb{E}_{q(h_n)}[\mathbb{E}_{q(\mathbf{r}_n|h_n)}[\log p(\mathbf{x}_n, \mathbf{r}_n, h_n|\theta_{1:K})]] \\
=&\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{q(h_n)}([h_n = k])\mathbb{E}_{q(\mathbf{r}_n|h_n=k)}\left[\sum_{t=1}^{T_n}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}[r_{t,n} = m_1][r_{t-1,n} = m_2]\log A_{k,m_1,m_2}\right] \\
&+\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{q(h_n)}([h_n = k])\mathbb{E}_{q(\mathbf{r}_n|h_n=k)}\left[\sum_{t=1}^{T_n}\sum_{m=1}^{M}[r_{t,n} = m]\log p(x_{t,n}|\ r_{t,n}, O(:,k))\right] \\
=&\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}([h_n = k])\sum_{t=1}^{T_n}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M} \mathbb{E}_{q(r_{t-1:t,n}|h_n=k)}\left([r_{t,n} = m_1][r_{t-1,n} = m_2]\right)\log A_{k,m_1,m_2} \\
&+\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}([h_n = k])\sum_{t=1}^{T_n}\sum_{m=1}^{M} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])\log p(x_{t,n}|\ r_{t,n}, O(:,k)) \qquad (3.20)
\end{aligned}
$$

So, we have derived the EM lower bound for mixture of HMMs. The next thing is to maximize this bound with respect to the parameters $\theta_{1:K}$. Update equations turn out to be as follows:

- Transition matrices $A_{1:K}$:

$$
A_{k,m_1,m_2} = \frac{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t-1:t,n}|h_n=k)}([r_{t,n} = m_1][r_{t-1,n} = m_2])}{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \sum_{m_1'=1}^{M} \mathbb{E}_{q(r_{t-1:t,n}|h_n=k)}([r_{t,n} = m_1'][r_{t-1,n} = m_2])} \qquad (3.21)
$$

- Observation matrices $O_{1:K}$:

  (i) Poisson observation model:

$$
\lambda_{k,l,m} = \frac{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])x_{t,n}}{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])} \qquad (3.22)
$$

(ii) Discrete observation model:

$$p_{k,l,m} = \frac{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])[x_{t,n} = l]}{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \sum_{l'=1}^{L} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])[x_{t,n} = l']} \tag{3.23}$$

(iii) Gaussian observation model:

$$\mu_{k,l,m} = \frac{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m]) x_{t,n}}{\sum_{n=1}^{N} \mathbb{E}([h_n = k]) \sum_{t=1}^{T_n} \mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])} \tag{3.24}$$

The necessary expectations to compute in the E-step are $\mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])$, $\mathbb{E}_{q(r_{t,n},r_{t-1,n}|h_n=k)}([r_{t,n} = m_1][r_{t-1,n} = m_2])$ and $\mathbb{E}_{q(h_n)}([h_n = k])$. The first two can be computed via forward-backward algorithm described in Section 3.2.1. The last one is computed as follows:

$$\mathbb{E}_{q(h_n)}([h_n = k]) = \frac{p(h_n = k)p(\mathbf{x}_n|\theta_k)}{\sum_{k=1}^{K} p(h_n = k)p(\mathbf{x}_n|\theta_k)} \tag{3.25}$$

where, $p(\mathbf{x}_n|\theta_k)$ is the likelihood of $\mathbf{x}_n$ given the parameters of cluster $k$. So, we compute the likelihood for all clusters, and then by weighting with $p(h_n = k)$, we compute the necessary expectation. Note that, in the M-step the parameters of $k$'th cluster $\theta_k = (O_k, A_k)$, are updated using the expectations $\mathbb{E}_{q(r_{t,n},r_{t-1,n}|h_n=k)}([r_{t,n} = m_1][r_{t-1,n} = m_2])$ and $\mathbb{E}_{q(r_{t,n}|h_n=k)}([r_{t,n} = m])$. Therefore, in each EM iteration we need to compute these expectations for $N$ sequences and $K$ clusters. Consequently, for sequences of average length $T$ and $M$ latent states, since each EM iteration is computationally dominated by the E-step, we have a computational complexity on the order of $\mathcal{O}(M^2TNK)$ in every iteration (These expectations can be computed using forward-backward algorithm on the order of $\mathcal{O}(M^2T)$.). Finally, note that we can have a hard clustering algorithm just by changing the update of the expectations $\mathbb{E}([h_n = k]) = 1$ for $k = \arg\max_k p(y_i|\theta_k)$, and $\mathbb{E}([h_n = k']) = 0$ for $k' \neq k$. In this case the necessary expectations for $k$'th HMM can be computed just by considering the sequences in $k$'th cluster. This way, the complexity can be reduced to $\mathcal{O}(M^2TN)$. In Section 4.4, we propose a novel mixture of HMMs learning algorithm which further

reduces the computational complexity by using a spectral learning algorithm in the parameter estimation step.

### 3.5. Training Discriminative Markov Model

The likelihood function to be maximized for training a discriminative Markov model is as follows:

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{n=1}^{N} \psi(\mathbf{x}_n, h_n; \theta_k) - \sum_{n=1}^{N} \log \sum_{k=1}^{K} \exp(\psi(\mathbf{x}_n, h_n; \theta_k)) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T_n} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} [h_n = k][l_1 = x_t][l_2 = x_{t-1}]\theta_{k,l_1,l_2} \\ &\quad - \sum_{n=1}^{N} \log \sum_{h_n=1}^{K} \exp(\sum_{k=1}^{K} \sum_{t=1}^{T_n} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} [h_n = k][l_1 = x_t][l_2 = x_{t-1}]\theta_{k,l_1,l_2}) \end{aligned} \tag{3.26}$$

then, we compute the gradient of this expression with respect to $\theta$ to derive the gradient descent update equations:

$$\begin{aligned}\frac{\partial \mathcal{L}(\theta)}{\partial \theta_{k,l_1,l_2}} &= \sum_{n=1}^{N} \sum_{t=1}^{T_n} [h_n = k][l_1 = x_t][l_2 = x_{t-1}] \\ &\quad - \sum_{n=1}^{N} \sum_{h_n=1}^{K} \underbrace{\frac{\exp(\psi(\mathbf{x}_n, h_n; \theta_k))}{\sum_{h_n=1}^{K} \exp(\psi(\mathbf{x}_n, h_n; \theta_k))}}_{p(h_n|\mathbf{x}_n, \theta^*)} \sum_{t=1}^{T_n} [h_n = k][l_1 = x_t][l_2 = x_{t-1}] \\ &= \sum_{n=1}^{N} \sum_{t=1}^{T_n} [h_n = k][l_1 = x_t][l_2 = x_{t-1}] - \sum_{n=1}^{N} \sum_{t=1}^{T_n} \mathbb{E}([h_n = k])[l_1 = x_t][l_2 = x_{t-1}] \\ &= \sum_{n=1}^{N} \Big( [h_n = k] - \mathbb{E}([h_n = k]) \Big) c^n_{l_1,l_2} \end{aligned} \tag{3.27}$$

where, $c^n_{l_1,l_2} = \sum_{t=1}^{T_n} [x_{t,n} = l_1][x_{t-1,n} = l_2]$, the transition counts matrix for sequence $\mathbf{x}_n$. In practice, in order to avoid overfitting, it is customary to use a regularizer by placing a prior $\theta \sim \mathcal{N}(\theta; 0, \sigma^2 I)$ (which corresponds to a vector $l_2$ norm) [21]. In this

case, the gradient descent updates become:

$$\theta^{new}_{k,l_1,l_2} \leftarrow \theta^{old}_{k,l_1,l_2} + \zeta\left(\frac{\partial L(\theta)}{\partial \theta_{k,l_1,l_2}} - \frac{1}{2\sigma^2}\theta_{k,l_1,l_2}\right) \tag{3.28}$$

where, $\zeta$ is the learning rate. In practice, it is generally necessary to make a line search for an appropriate step size $\zeta$, so that we do not decrease the likelihood $\mathcal{L}(\theta)$ with gradient descent steps. Note that, the gradient in Equation 3.27 makes sense since, when we have the parameter $\theta$ so that all the classes are well-seperated, the gradient vanishes. In such a case, the expectation of the class indicator $\mathbb{E}[h_n]$ become equal to the to the actual class indicator $h_n$, and thus the gradient vanishes and we do not update the parameter.

## 3.6. Training Discriminative HMM

The likelihood function to be maximized in the training of a discriminative HMM is as follows:

$$\begin{aligned}
\mathcal{L}(\theta_{1:K}) = &\sum_{n=1}^{N} \log \sum_{\mathbf{r}_n} \exp\left(\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}[r_{t,n}=m_1][r_{t-1,n}=m_2][h_n=k]\theta^1_{k,m_1,m_2}\right.\\
&\left.+\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{m=1}^{M}[x_{t,n}=l][r_{t,n}=m][h_n=k]\theta^2_{k,l,m}\right)\\
&-\sum_{n=1}^{N} \log \sum_{h'_n=1}^{K}\sum_{\mathbf{r}_n} \exp\left(\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}[r_{t,n}=m_1][r_{t-1,n}=m_2][h_n=k]\theta^1_{k,m_1,m_2}\right.\\
&\left.+\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{m=1}^{M}[x_{t,n}=l][r_{t,n}=m][h_n=k]\theta^2_{k,l,m}\right)
\end{aligned} \tag{3.29}$$

Then, we compute the partial derivatives $\partial\mathcal{L}(\theta_{1:K})/\partial\theta^1_{k,m_1,m_2}$ and $\partial\mathcal{L}(\theta_{1:K})/\partial\theta^2_{k,l,m}$ to

derive the gradient descent update equations.

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta_{1:K})}{\partial \theta^1_{k,m_1,m_2}} =& \sum_{n=1}^{N} \sum_{\mathbf{r}_n} \frac{\exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_{1:K}))}{\sum_{\mathbf{r}'_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}'_n, h_n; \theta_{1:K}))} \sum_{t=1}^{T} [r_{t,n} = m_1][r_{t-1,n} = m_2][h_n = k] \\
& - \sum_{n=1}^{N} \sum_{\mathbf{r}_n, h_n} \frac{\exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_{1:K}))}{\sum_{\mathbf{r}'_n, h'_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}'_n, h'_n; \theta_{1:K}))} \sum_{t=1}^{T} [r_{t,n} = m_1][r_{t-1,n} = m_2][h_n = k] \\
=& \sum_{n=1}^{N} \sum_{\mathbf{r}_n} p(\mathbf{r}_n | h_n, \mathbf{x}_n, \theta_{1:K}) \sum_{t=1}^{T} [r_{t,n} = m_1][r_{t-1,n} = m_2][h_n = k] \\
& - \sum_{n=1}^{N} \sum_{h_n, \mathbf{r}_n} p(\mathbf{r}_n, h_n | \mathbf{x}_n, \theta_{1:K}) \sum_{t=1}^{T} [r_{t,n} = m_1][r_{t-1,n} = m_2][h_n = k] \\
=& \sum_{n=1}^{N} \sum_{t=1}^{T} [h_n = k] \mathbb{E}_{p(r_{t,n}, r_{t-1,n} | h_n, \mathbf{x}_n, \theta_{1:K})}([r_{t,n} = m_1][r_{t-1,n} = m_2]) \\
& - \sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{p(r_{t,n}, r_{t-1,n}, h_n | \mathbf{x}_n, \theta_{1:K})}([r_{t,n} = m_1][r_{t-1,n} = m_2][h_n = k])
\end{aligned} \tag{3.30}$$

Similarly for $\theta^2_{k,l,m}$;

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta_{1:K})}{\partial \theta^2_{k,l,m}} =& \sum_{n=1}^{N} \sum_{\mathbf{r}_n} \frac{\exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_{1:K}))}{\sum_{\mathbf{r}'_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}'_n, h_n; \theta_{1:K}))} \sum_{t=1}^{T} [x_{t,n} = l][r_{t,n} = m][h_n = k] \\
& - \sum_{n=1}^{N} \sum_{\mathbf{r}_n, h_n} \frac{\exp(\psi(\mathbf{x}_n, \mathbf{r}_n, h_n; \theta_{1:K}))}{\sum_{\mathbf{r}'_n, h'_n} \exp(\psi(\mathbf{x}_n, \mathbf{r}'_n, h'_n; \theta_{1:K}))} \sum_{t=1}^{T} [x_{t,n} = l][r_{t,n} = m][h_n = k] \\
=& \sum_{n=1}^{N} \sum_{\mathbf{r}_n} p(\mathbf{r}_n | h_n, \mathbf{x}_n, \theta_{1:K}) \sum_{t=1}^{T} [x_{t,n} = l][r_{t,n} = m][h_n = k] \\
& - \sum_{n=1}^{N} \sum_{h_n, \mathbf{r}_n} p(\mathbf{r}_n, h_n | \mathbf{x}_n, \theta_{1:K}) \sum_{t=1}^{T} [x_{t,n} = l][r_{t,n} = m][h_n = k] \\
=& \sum_{n=1}^{N} \sum_{t=1}^{T} [h_n = k][x_{t,n} = l] \mathbb{E}_{p(r_{t,n} | h_n, \mathbf{x}_n, \theta_{1:K})}([r_{t,n} = m]) \\
& - \sum_{n=1}^{N} \sum_{t=1}^{T} [x_{t,n} = l] \mathbb{E}_{p(r_{t,n}, h_n | \mathbf{x}_n, \theta_{1:K})}([r_{t,n} = m][h_n = k])
\end{aligned} \tag{3.31}$$

Then, using gradient descent, we update the parameters.

$$\theta^{1,new}_{k,m_1,m_2} \leftarrow \theta^{1,old}_{k,m_1,m_2} + \zeta\left(\frac{\partial \mathcal{L}(\theta_{1:K})}{\partial \theta^1_{k,m_1,m_2}} - \frac{1}{2\sigma^2}\theta^1_{k,m_1,m_2}\right) \tag{3.32}$$

$$\theta^{2,new}_{k,l,m} \leftarrow \theta^{2,old}_{k,l,m} + \zeta\left(\frac{\partial \mathcal{L}(\theta_{1:K})}{\partial \theta^2_{k,l,m}} - \frac{1}{2\sigma^2}\theta^2_{k,l,m}\right) \tag{3.33}$$

The last terms in these update equations are due to regularizer.

### 3.6.1. Inference in discriminative HMM

The necessary expectations we have to compute are,

$$\begin{aligned}\mathbb{E}_{p(r_{t,n}|h_n,\mathbf{x}_n,\theta_{1:K})}[r_{t,n}=m] &= \sum_{r_{t,n}=1}^{K} p(r_{t,n}|h_n,\mathbf{x}_n,\theta_{1:K})[r_{t,n}=k] \\ &= p(r_{t,n}=k|h_n,\mathbf{x}_n,\theta_{1:K})\end{aligned} \tag{3.34}$$

$p(r_{t,n}|h_n,\mathbf{x}_n,\theta_{1:K})$, can be estimated using forward-backward algorithm described in Section 3.2.1.

$$\begin{aligned}&\mathbb{E}_{p(r_{t,n},h_n|\mathbf{x}_n,\theta_{1:K})}([r_{t,n}=m][h_n=k]) \\ &\quad= \sum_{r_{t,n},h_n} p(r_{t,n},h_n|\mathbf{x}_n,\theta_{1:K})[r_{t,n}=m][h_n=k] \\ &\quad= \sum_{r_{t,n},h_n} p(r_{t,n}|h_n,\mathbf{x}_n,\theta_{1:K})p(h_n|\mathbf{x}_n,\theta_{1:K})[r_{t,n}=m][h_n=k] \\ &\quad= p(r_{t,n}=m|h_n=k,\mathbf{x}_n,\theta_{1:K})p(h_n=k|\mathbf{x}_n,\theta_{1:K}) \\ &\quad= p(r_{t,n}=m|h_n=k,\mathbf{x}_n,\theta_{1:K})\frac{p(\mathbf{x}_n|h_n=k,\theta_{1:K})p(h_n=k)}{p(\mathbf{x}_n|\theta_{1:K})}\end{aligned} \tag{3.35}$$

The other two necessary expectations are pairwise expectations of $r_t$ and $r_{t-1}$.

$$\begin{aligned}
&\mathbb{E}_{p(r_{t,n},r_{t-1,n}|h_n,\mathbf{x}_n,\theta_{1:K})}([r_{t,n}=m_1][r_{t-1,n}=m_2]) \\
&\qquad = \sum_{r_{t,n}}\sum_{r_{t-1,n}} [r_{t,n}=m_1][r_{t-1,n}=m_2]p(r_{t,n},r_{t-1,n}|h_n,\mathbf{x}_n,\theta_{1:K}) \\
&\qquad = p(r_{t,n}=m_1,r_{t-1,n}=m_2|h_n,\mathbf{x}_n,\theta_{1:K}) \qquad (3.36)
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}_{p(r_{t,n},r_{t-1,n},h_n|\mathbf{x}_n,\theta_{1:K})}([r_{t,n}=m_1][r_{t-1,n}=m_2][h_n=k]) \\
&\quad = \sum_{r_{t,n},r_{t-1,n},h_n} p(r_{t,n},r_{t-1,n},h_n|\mathbf{x}_n,\theta_{1:K})[r_{t,n}=m_1][r_{t-1,n}=m_2][h_n=k] \\
&\quad = \frac{p(r_{t,n}=m_1,r_{t-1,n}=m_2,h_n=k|\mathbf{x}_n,\theta_{1:K})p(\mathbf{x}_n|h_n=k,\theta_{1:K})p(h_n=k)}{p(\mathbf{x}_n|\theta_{1:K})} \qquad (3.37)
\end{aligned}$$

All of the necessary probabilities can be computed efficiently using forward-backward algorithm in Section 3.2.1.

# 4. METHOD OF MOMENTS BASED LEARNING ALGORITHMS

In this chapter, we describe the method of moments based learning algorithms for latent variable models. The main idea in these algorithms is to do (or bypass) the parameter estimation step by using the fact that model parameters can be expressed as a function of some observable moments. This approach is rooted in Pearson's early work in 1936 [23], for learning mixtures of Gaussians. Now, let us consider the following example problem: Suppose we want to learn the parameters of a Gamma distribution $\mathcal{G}(x; a, b)$, from its first and second order moments:

$$M_1 := \mathbb{E}[x] = ab \tag{4.1}$$

$$M_2 := \mathbb{E}[x^2] = ab^2 + a^2b^2 \tag{4.2}$$

We can solve this system of equations for $a$ and $b$:

$$a = \frac{M_1^2}{M_2 - M_1^2} \tag{4.3}$$

$$b = \frac{M_2 - M_1^2}{M_1} \tag{4.4}$$

So, we conclude that we can relate the moments of the Gamma distribution to its parameters $a$ and $b$. The main advantage of method of moments compared to the classical maximum likelihood (ML) estimation is that, we can estimate the model parameters in closed form. In some cases, as in this one, a closed form solution may not be possible in ML approach. For a ML estimation, we first need to solve the following equation via a numerical method:

$$\log(a) - \psi(a) = \log\left(\frac{1}{N}\sum_{n=1}^{N} x_n\right) - \frac{1}{N}\sum_{n=1}^{N}\log(x_n) \tag{4.5}$$

then, $b$ is estimated from,

$$b = \frac{1}{aN} \sum_{n=1}^{N} x_n \tag{4.6}$$

where, $N$ is the number of data points and $\psi(.)$ is the digamma function. So, we see that a ML solution for parameters $a$, $b$ is more complicated than a method of moments approach. Note that, for a method moments approach, in practice we replace $M_1$ and $M_2$ with empirical moment estimates;

$$\widehat{M_1} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{4.7}$$

$$\widehat{M_2} = \frac{1}{N} \sum_{n=1}^{N} x_n^2 \tag{4.8}$$

Therefore, the accuracy of the empirical moment estimates are of crucial importance for the overall accuracy of the method. Naturally, to have accurate moment estimates, we have to have large number of samples $N$. Otherwise, inaccurate moment estimates lead to inaccurate parameter estimates. In Figure 4.1, we show the parameter estimations from data where $x_n \sim \mathcal{G}(x_n; a = 1, b = 5)$, with ML and method of moments. We do parameter estimations for cases $N = 10, 40, 100, 200$. We see that method of moments fails to perform well when $N$ is not large. However, as $N$ increases, we see that ML estimation and the method of moment estimates get closer. In cases where $N$ is not large, method of moments fail to perform well because of inaccurate empirical moment estimates.

Inaccurate empirical moment estimates may even cause parameter estimates that are not in the feasible region. For instance in Gamma distribution, the parameters $a$ and $b$ are strictly positive. The method of moment estimators in Equation 4.3 and Equation 4.4 may return negative signed parameter estimates. On the other hand, maximum likelihood approach gives reliable estimates, guaranteed to yield solutions in the feasible region, since it is after all a constrained optimization approach. The downside of ML

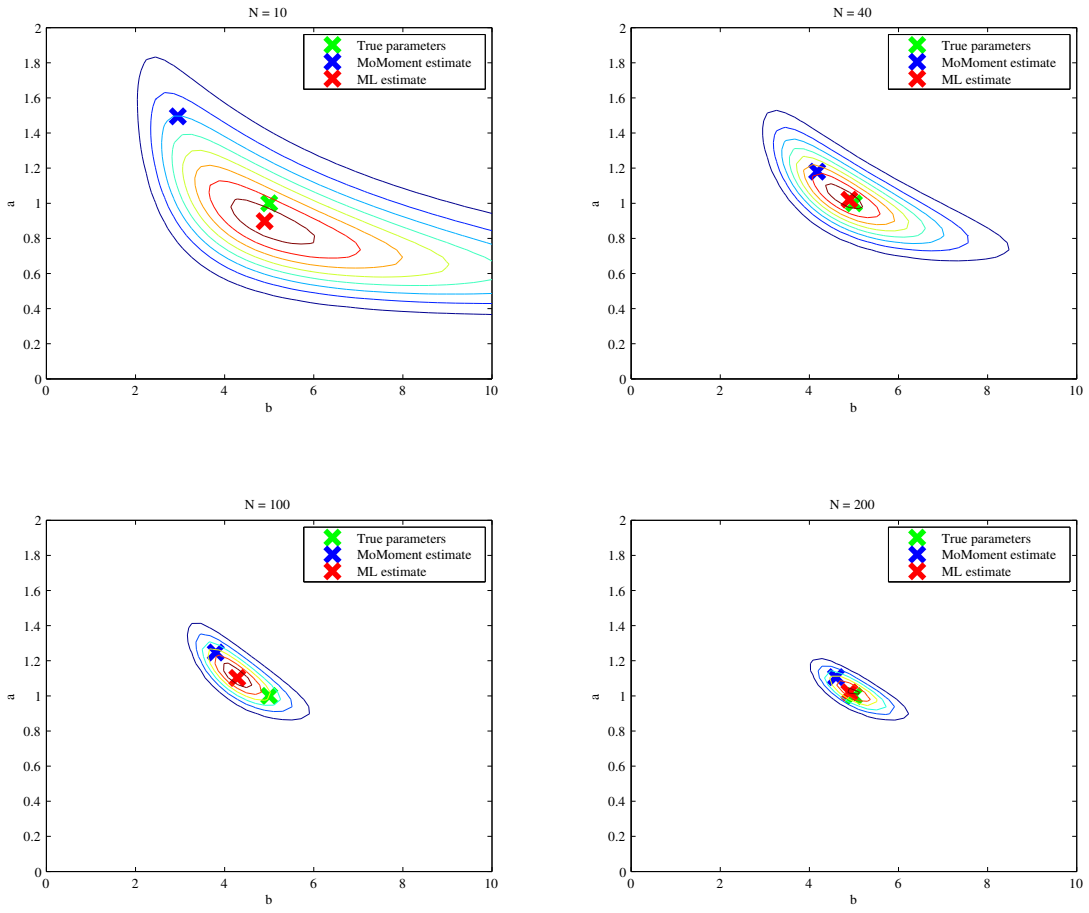



Figure 4.1. Parameter estimation from $N$, i.i.d. data from $\mathcal{G}(x_n; a = 1, b = 5)$. We plot the likelihood surface $\mathcal{G}(x_{1:N}, a, b)$ for each case.

is that, the optimization problem may not have a unique solution, resulting possibly from the non-convexity of the objective function. So, in general it yields solutions which are only locally optimal. Moreover, to have "good" local solutions we have to do "good" initializations for the optimization problem. And of course, possibly lacking a closed form solution, we have to apply some iterative solution methods like gradient descent which have a computational burden. Whereas the method of moments give us a solution in one step, without necessitating any initialization of any sort. In other words, we directly obtain a solution for the parameters as a function of the empirical moment estimates.

Recently, there has been work on method of moments based algorithms for learning latent variable models such as Hidden Markov Model [4, 15, 16], Gaussian Mixture

Model [4,16], Latent Dirichlet Allocation [16,24], community membership models [25]. These methods are called spectral learning methods in the machine learning community. The reason is that, one can use the observable moments to construct a matrix whose eigen-decomposition gives information about the model parameters. In the next section, we will exemplify the general methodology by learning a mixture model. Later on, we will describe the spectral learning algorithms for learning time series models such as HMM, mixture of Markov models and mixture of Hidden Markov models.

### 4.1. Spectral Learning for Mixture Models

In this section we describe the spectral algorithm in [4] for learning a simple mixture model. Spectral learning algorithm for a simple mixture model is a good example to sketch the general algorithm, with which we also learn sequential models such as HMMs and mixture of Markov models. Let us first briefly introduce the mixture model. The likelihood of a data item $x_n$ is defined as follows;

$$p(x_n|\ O) = \sum_{k=1}^{K} p(h_n = k)p(x_n|h_n = k, O) \tag{4.9}$$

where, we follow the notation used in Section 3.1 for a general latent variable model. We use $h_n \in \{1, \dots K\}$, for latent cluster indicator variable for data item $x_n \in \mathbb{R}^L$. The model parameter $\theta$ is the observation matrix $O(l, k) = \mathbb{E}[x_{n,l}|h_{n,k}], \forall n \in \{1, \dots, N\}$. Note that, $O \in \mathbb{R}^{L \times K}$. According to the choice of the observation model, a column $O(:, k)$ of the observation matrix correspond to different parameters, as we did for HMM observation matrix in Section 2.1.2 (e.g. if the observation model is Gaussian, it corresponds to the mean vector of the $k$'th cluster $O(:, k) = \mu_k$, or similarly if we have a discrete observation model, we have $O(:, k) = p_k$). To generate a data set $x_{1:N}$, the generative process is as follows:

- For each data item $\mathbf{x}_n$, we sample a cluster indicator variable $h_n \in \{1, \dots, K\}$ from $p(h_n)$, which is a generic discrete distribution.
- Conditioned on $h_{1:N}$, we generate $x_{1:N}$ via $p(x_n|h_n, O)$.

The corresponding graphical model is given in Figure 4.2.

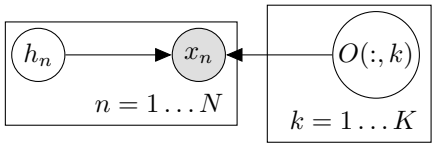



Figure 4.2. DAG of a mixture model.

To estimate the $O$ matrix given the data $x_{1:N}$, we can maximize the likelihood $p(\mathbf{x}_{1:N}|O)$:

$$O^* = \arg\max_O p(\mathbf{x}_{1:N}|O) = \arg\max_O \sum_{h_{1:N}} \prod_{n=1}^{N} p(\mathbf{x}_n, h_n|O) \tag{4.10}$$

However, since the likelihood $p(\mathbf{x}_{1:N}|O)$ is defined via a summation over the all possible combinations of the cluster indicator variables $h_{1:N}$, we have to resort to some iterative optimization method. One of the most common ways is to iteratively maximize an EM lower bound, as discussed extensively in the previous sections:

$$\mathcal{Q}(O, O^{old}) = \mathbb{E}_{p(h_{1:N}|\ \mathbf{x}_{1:N}, O^{old})}[\log(p(\mathbf{x}_{1:N}, h_{1:N}|O))] \tag{4.11}$$

However, as we discussed in Section 3.1, EM algorithm is dependent on good initialization. At this point we introduce a spectral learning algorithm for learning this mixture model [4]. As discussed at the beginning of this chapter, the spectral learning algorithms for latent variable models are based on method of moments. That is, we aim to express the model parameters as a function of some observable moments. With this aim, we use the second order moment $\mathbb{E}[x \otimes x]$ and third order moment $\mathbb{E}[x \otimes x \otimes x_i]$. In the following lemma, we establish a relation between the model parameters and these observable moments.

**Lemma 4.1.**

$$\mathbb{E}[x \otimes x] = O diag(p(h)) O^T \tag{4.12}$$

$$\mathbb{E}[x \otimes x \otimes x_i] = O diag(O(i,:)) diag(p(h)) O^T \tag{4.13}$$

Note that we drop the subscript $n$ to keep the notation uncluttered.

*Proof.* We write the definition of these moments in index notation:

$$\begin{aligned}\mathbb{E}[x_i x_j] &= \sum_h \mathbb{E}[x_i|h]p(h)\mathbb{E}[x_j|h] \\ &= \sum_h O_{i,h}p(h)O_{j,h}\end{aligned}$$

Then, by definition of the matrix multiplication, we can see that the full $\mathbb{E}[x \otimes x]$ matrix is given by 4.12. Similarly; for triples $\mathbb{E}[x \otimes x \otimes x_i]$,

$$\begin{aligned}\mathbb{E}[x_i x_j x_k] &= \sum_h \mathbb{E}[x_i|h]p(h)\mathbb{E}[x_j|h]\mathbb{E}[x_k|h] \\ &= \sum_h O_{i,h}p(h)O_{j,h}O_{k,h}\end{aligned}$$

Again, by the definition of the matrix multiplication, we see that the full $\mathbb{E}[x \otimes x \otimes x_i]$ matrix is given by 4.13. □

**Lemma 4.2.** *Given that $L \geq K$, and $O$ has $K$ linearly independent columns then,*

$$B_i := (U^T O)diag(O(i,:))(U^T O)^{-1} = (U^T \mathbb{E}[x \otimes x \otimes x_i]V)(U^T \mathbb{E}[x \otimes x]V)^{-1} \quad (4.14)$$

*where, $U$ and $V$ are respectively the first $K$ left and right singular vectors of the second order statistics matrix $\mathbb{E}[x \otimes x]$ as in [4].*

*Proof.*

$$\begin{aligned}U^T \mathbb{E}[x \otimes x \otimes x_i]V =& U^T O diag(O(i,:))diag(p(h))O^T V \\ =& U^T O diag(O(i,:))(U^T O)^{-1} \underbrace{U^T O diag(p(h))O^T V}_{U^T \mathbb{E}[x \otimes x]V}\end{aligned}$$

so, if we multiply both sides by $(U^T \mathbb{E}[x \otimes x \otimes x_i]V)^{-1}$ from right, we see that we obtain the equality in Equation 4.14. Note that $U^T \mathbb{E}[x \otimes x \otimes x_i]V$ is invertible since it is

diagonal and full rank. □

So, we conclude that we can estimate the model parameters $O$ by doing eigen-decomposition (hence the name spectral learning) of the matrix in the left hand side of Equation 4.14, which can be approximately computed solely by using observable moments: The eigenvectors are $U^T O$, and eigenvalues are the $i$'th row of the observation matrix $O$. Notice that the choice of $K$ when computing the SVD of $\mathbb{E}[x \otimes x]$ determines the number of clusters $K$ as well.

Let us denote the approximate (collected) second order moment $\mathbb{E}[x \otimes x]$ with $P_2$, and third order approximate moment $\mathbb{E}[x \otimes x \otimes x]$ with $P_3$. Estimation of $P_2$ and $P_3$ from data $x_{1:N}$ is as follows:

$$P_2 = \frac{1}{N} \sum_{n=1}^{N} x_n \otimes x_n = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^T \tag{4.15}$$

$$P_3 = \frac{1}{N} \sum_{n=1}^{N} x_n \otimes x_n \otimes x_n \tag{4.16}$$

Note that we can apply this moment estimator for a discrete distribution as well, if we take discrete distribution as a multinomial distribution with number of trials parameter as 1. In this case, the support of the distribution are vectors with of length $L$, with only one element one and else zero.

Notice that the choice of $B_i$ is arbitrary and in practice cause the algorithm to be unstable for certain $i$ values: For some $i$ values the eigenvalues of $B_i$ may get too close to each other, which in turn causes instability in eigenvector estimation. To overcome this problem, it is suggested in [4] that using $\mathbb{E}[x \otimes x \otimes x](\eta) = \sum_{i=1}^{L} \eta_i \mathbb{E}[x \otimes x \otimes x_i]$ instead of using $\mathbb{E}[x \otimes x \otimes x_i]$ would help to make the algorithm more robust, where $\eta \in \mathbb{R}^{L \times 1}$. That is, when we compute $B_i$, we use $\mathbb{E}[x \otimes x \otimes x](\eta)$ instead of $\mathbb{E}[x \otimes x \otimes x_i]$. To choose an appropriate $\eta$ vector, we generate random $\eta$ until we get eigenvalues separated enough (as judged by a certain threshold). We denote this auxiliary matrix as $B(\eta)$ instead of $B_i$. The algorithm for spectral learning a mixture models is given

in Figure 4.1.

Input: Data $x_{1:N}$

Output: Estimated observation matrix $\widehat{O}$

1. Compute $P_2$ and $P_3$ from input data.
2. Compute rank $K$ SVD of $P_2$ such that $P_2 = U\Sigma V^T$.
3. Compute $B(\eta) = (U^T P_3(\eta) V)(U^T P_2 V)^{-1}$
4. Compute the eigen-decomposition of $B(\eta) = \epsilon \Lambda \epsilon^{-1}$.
5. Set $temp = U\epsilon$.
6. Normalize the columns of *temp* to have columns sum up to one and set $\widehat{O} =$ *temp*.

Figure 4.3. Spectral Learning Algorithm for Mixture Model.

Finally, note that although with Figure 4.1 we have a way for solving for the $O$ matrix solely from the observations $x_{1:N}$, we do not have a guarantee on recovering $O$ with the right column order and scaling, since we recover it with an eigen-decomposition, which has permutation and scale ambiguity. Column ordering, i.e. permutation ambiguity does not prevent us from doing clustering or classification.

However, scale ambiguity is a nuisance if we need to know the right scale. For example, if have a discrete model, i.e. we know that columns of $O$ must sum up to one, then scaling ambiguity does not matter, since we can always normalize the output of the algorithm. But, for instance if we have a Gaussian observation model, the scale of the mean vectors is important. In such a case, we can still use this algorithm, by reading the $O$ matrix not from the eigenvectors but from the eigenvalues of $B(\eta)$. If set $\eta$ such that it chooses a particular column of $P_3$ e.g., $\eta = [\ 0\ 1\ 0\ \ldots\ 0]^T$, then it is easy to see that, eigenvalues of $B_i$ correspond to a row of $O$. We can also still use general $\eta$, the details are given in [4], and we do not reproduce them here.

## 4.2. Spectral Learning for Hidden Markov Models

In this section we introduce a spectral learning algorithm for HMMs. We have introduced a spectral learning algorithm for mixture models in the previous section, where latent cluster indicator variables are independent. HMM is also a mixture model, with the only difference being, the latent cluster indicators form a Markov chain. It seems unclear how to immediately adapt the algorithm in the previous section for a temporally connected mixture model such as HMM.

We will first introduce the spectral learning algorithm for HMMs in [15]. Then we will also show that, it is possible to see HMMs as multi-view mixture models, and specify the corresponding algorithm.

The main idea in [15], is to express the likelihood $p(x_{1:T}\ |\theta)$ as a series of matrix multiplications, so as to rewrite it as a series of similarity transformations of the actual forward message updates, which can be expressed in terms of observable moments. Let us first show that the likelihood $p(x_{1:T}|\theta)$ can be written as a series of matrix multiplications.

*Claim:* Given the observed sequence $x_{1:T}$ and HMM parameters $\theta = \{O, A, \pi\}$;

$$p(x_{1:T}|\theta) = 1_M^T \mathcal{M}_{x_T} \mathcal{M}_{x_{T-1}} \dots \mathcal{M}_{x_t} \dots \mathcal{M}_{x_1} \pi \tag{4.17}$$

where, $\mathcal{M}_x = Adiag(O(x,:))$ and $1_M$ is a vector of all ones of length $M$.

*Proof.*

$$p(x_{1:T}|\theta) = \underbrace{\sum_{r_{T+1}} \sum_{r_T} p(r_{T+1}|r_T) p(x_T|r_T) \dots \underbrace{\sum_{r_2} p(r_3|r_2) p(x_2|r_2) \underbrace{\sum_{r_1} p(r_2|r_1) p(x_1|r_1) p(r_1)}_{\mathcal{M}_{x_1}\pi}}_{\mathcal{M}_{x_2}\mathcal{M}_{x_1}\pi}}_{\mathcal{M}_{x_T}\dots\mathcal{M}_{x_2}\mathcal{M}_{x_1}\pi}}_{1_M^T \mathcal{M}_{x_T}\dots\mathcal{M}_{x_2}\mathcal{M}_{x_1}\pi}$$

Note that $(\mathcal{M}_x)_{ij} = p(r_{t+1} = i|r_t = j)p(x_t|r_t = j)$. This concludes the proof. □

Next, we apply some similarity transformations on the messages $\mathcal{M}_x$, so that the complete likelihood expression can be computed via observable moments $\mathbb{E}[x_1]$,$\mathbb{E}[x_1 \otimes x_2]$ and $\mathbb{E}[x_1 \otimes x_2 \otimes x_3]$.

$$\begin{aligned} p(x_{1:T}|\theta) =& 1_M^T \mathcal{M}_{x_T} \mathcal{M}_{x_{T-1}} \dots \mathcal{M}_{x_t} \dots \mathcal{M}_{x_1} \pi \\ =& \underbrace{1_M^T (U^T O)^{-1}}_{:=b_\infty^T} \underbrace{(U^T O)\mathcal{M}_{x_T}(U^T O)^{-1}}_{:=B_{x_T}} \dots \underbrace{(U^T O)\mathcal{M}_{x_t}(U^T O)^{-1}}_{:=B_{x_t}} \dots \\ & \times \dots \underbrace{(U^T O)\mathcal{M}_{x_1}(U^T O)^{-1}}_{:=B_{x_1}} \underbrace{(U^T O)\pi}_{:=b_1} \\ =& b_\infty^T B_{x_T} B_{x_{T-1}} \dots B_{x_t} \dots B_{x_1} b_1 \end{aligned} \tag{4.18}$$

Note that, we have to choose $U$ such that $U^T O$ is invertible as discussed in the previous section. The next thing is to express the messages $\mathcal{M}_x$, so that they can be computed via observable moments. Let us denote the observable moments as follows:

$$\begin{aligned} (P_1)_i :=& (\mathbb{E}[x_1])_i \\ (P_{2,1})_{ij} :=& (\mathbb{E}[x_2 \otimes x_1])_{i,j} \\ (P_{3,l,1})_{ij} :=& (\mathbb{E}[x_3 \otimes x_{2,l} \otimes x_1])_{i,j} \end{aligned}$$

We establish the connection between these observable moments and the model parameters in the following lemma:

**Lemma 4.3.**

$$P_1 = O\pi \tag{4.19}$$

$$P_{2,1} = OAdiag(\pi)O^T \tag{4.20}$$

$$P_{3,x,1} = O\mathcal{M}_x Adiag(\pi)O^T \tag{4.21}$$

*Proof.* The first expression is easy to prove since,

$$P_1 = \sum_k O(:,k)\pi_k$$

so, in matrix notation $P_1 = O\pi$. For the second moment expression, we rewrite it in the index notation:

$$P_{2,1} = \underbrace{\sum_{r_2} p(x_2|r_2) \underbrace{\sum_{r_1} p(r_2|\ r_1) \underbrace{p(r_1)p(x_1|r_1)}_{diag(\pi)O^T}}_{Adiag(\pi)O^T}}_{OAdiag(\pi)O^T}$$

where, we use the definition of the matrix multiplication to convert the expression from index notation to matrix notation. Finally we have:

$$P_{3,l,1} = \underbrace{\sum_{r_3} p(x_3|r_3) \underbrace{\sum_{r_2} p(r_3|r_2)p(x_2=l|r_2) \underbrace{\sum_{r_1} p(r_2|\ r_1) \underbrace{p(r_1)p(x_1|r_1)}_{diag(\pi)O^T}}_{Adiag(\pi)O^T}}_{\mathcal{M}_l Adiag(\pi)O^T}}_{O\mathcal{M}_l Adiag(\pi)O^T}$$

□

Now, we are ready to express the messages $b_1, b_\infty, B_x$ in terms of the observable moments $P_1$, $P_2$, $P_3$.

**Lemma 4.4.**

$$b_1 = U^T P_1 \tag{4.22}$$

*Proof.*

$$\begin{aligned} b_1 =& U^T P_1 \\ =& U^T O\pi \end{aligned}$$

This proof is easy, due to the fact that $P_1 = \sum_k O(:,k)\pi_k$. □

**Lemma 4.5.**

$$b_\infty = (V^T P_{2,1}^T U)^{-1} V^T P_1 \tag{4.23}$$

*Proof.* We first express $P_1^T$ as follows:

$$\begin{aligned} P_1^T =& 1_M^T A diag(\pi) O^T \\ =& 1_M^T (U^T O)^{-1} (U^T O) A diag(\pi) O^T \\ =& 1_M^T (U^T O)^{-1} U^T P_{2,1} \end{aligned}$$

Note that, the expressions for $P_1$ and $P_{2,1}$ can be easily verified using a similar approach we used for 4.17.Then we plug this expression in:

$$\begin{aligned} b_\infty^T =& P_1^T V (U^T P_{2,1} V)^{-1} \\ =& 1_M^T (U^T O)^{-1} U^T P_{2,1} V (U^T P_{2,1} V)^{-1} \\ =& 1_M^T (U^T O)^{-1} \end{aligned}$$

□

**Lemma 4.6.**

$$B_x = U^T P_{3,x,1} V (U^T P_{2,1} V)^{-1} \tag{4.24}$$

*Proof.* Let us express $U^T P_{3,x,1} V$ as follows:

$$\begin{aligned} U^T P_{3,x,1} V =& U^T O \mathcal{M}_x A diag(\pi) O^T V \\ =& U^T O \mathcal{M}_x (U^T O)^{-1} (U^T O) A diag(\pi) O^T V \\ =& U^T O A_x (U^T O)^{-1} U^T P_{2,1} V \end{aligned}$$

So, we conclude that $B_x = U^T P_{3,x,1} V (U^T P_{2,1} V)^{-1}$. □

Note that we can estimate with $P_1 = \frac{1}{N} \sum_{n=1}^{N} x_n$ and $P_2$, $P_3$ with Equation 4.15 and Equation 4.16. As done in [4], we choose $U$ and $V$ matrices such that $P_2 = U \Sigma V^T$, where $U$ and $V$ consist of the first $K$, respectively, left and right singular vectors of $P_2$. Notice that, the resulting algorithm from the discussion in this section is able to estimate $p(x_{1:T}|\theta)$, solely by using some observable moments.

However, we did not specify a parameter estimation procedure to estimate $\theta$. In the next section, we will show that, we can in fact also recover the HMM parameters $O$ and $A$ with some modification on the Figure 4.1.

### 4.2.1. Parameter Learning in HMMs

In order to estimate the model parameters $\theta$ of a hidden Markov model, we can in fact use an algorithm similar to 4.1 as given in [4]. With this aim let us define $P_{3,1} := \mathbb{E}[x_3 \otimes x_1]$.

**Lemma 4.7.**

$$P_{3,1} = OA A diag(\pi) O^T \tag{4.25}$$

*Proof.* As we did for $P_{2,1}$, we consider the index notation;

$$P_{3,1} = \underbrace{\sum_{r_4} p(x_3|\ r_3) \underbrace{\sum_{r_2} p(r_3|r_2) \underbrace{\sum_{r_1} p(r_2|r_1) \underbrace{p(r_1)p(x_1|r_1)}_{diag(\pi)O^T}}_{Adiag(\pi)O^T}}_{AAdiag(\pi)O^T}}_{OAAdiag(\pi)O^T}$$

□

Given this observable moment and $P_{3,x,1}$ from the previous section, we can in fact recover the model parameters. Let us establish this with the following lemma:

**Lemma 4.8.**

$$\begin{aligned} B_x :=& (U^T OA) diag(O(x,:))(U^T OA)^{-1} \\ =& U^T P_{3,x,1} V (U^T P_{2,1} V)^{-1} \end{aligned} \tag{4.26}$$

*Proof.* Let us express $U^T P_{3,x,1} V$ as follows:

$$\begin{aligned} U^T P_{3,x,1} V =& U^T O \mathcal{M}_x Adiag(\pi) O^T V \\ =& U^T O Adiag(O(x,:)) Adiag(\pi) O^T V \\ =& U^T O Adiag(O(x,:))(U^T OA)^{-1}(U^T OA) Adiag(\pi) O^T V \\ =& (U^T OA) diag(O(x,:))(U^T O)^{-1} U^T P_{3,1} V \end{aligned}$$

So we conclude that $B_x$ can be recovered via $U^T P_{3,x,1} V (U^T P_{2,1} V)^{-1}$. □

Notice that $B_x$ is an eigen-decomposition form where eigenvalues are in fact $x$'th row of $O$ and eigenvectors are $U^T OA$. As we did similarly in previous sections, we set $P_{3,1} = U\Sigma V^T$. The algorithm for parameter estimation in HMM is given in Figure 4.2.1.

Input: Sequence $x_{1:T}$

Output: Estimated HMM parameters $\widehat{\theta} : (\widehat{O}, \widehat{A})$.

1. Estimate $P_{3,1}$ and $P_{3,2,1}$ from the sequence $x_{1:T}$.
2. Compute $U$ and $V$ from $P_{3,1} = U\Sigma V^T$ using SVD.
3. Estimate the first row of $\widehat{O}$ matrix and $R = \widehat{U^T O A}$ matrix by doing the eigen-decomposition $B_1 = Rdiag(O(1,:))R^{-1}$
4. For $l \in \{2, \ldots, L\}$ set $\widehat{O}_{l,:} = R^{-1}B(l)R$.
5. Set $temp = (U\hat{O})^{-1}R$
6. Normalize $temp$ to have columns sum up to one, and set $\widehat{A} = temp$.

Figure 4.4. Spectral Learning Algorithm for HMM.

Note that we did not use $\eta$ in Section 4.1, for the sake of simplicity. In order to increase the robustness of the algorithm, in practice it is necessary to use $P_{3,2,1}(\eta)$ instead of $P_{3,x,1}$. The details on usage of $\eta$ are given in detail in [4].

Also notice that the ordering of the eigenvalues of $B_x$ is dependent upon the ordering of its eigenvectors. The reason for using $R$ in steps 3 and 4 is to preserve the consistency in the ordering of the eigenvalues. Finally, note that when computing the empirical moments, despite the notation $P_{3,2,1}$, we use all of the observations in the sequence $x_{1:T}$ (not just the first three) since the moments we compute are based on the assumption that the underlying latent Markov chain has converged to a stationary distribution [15]. For instance, in empirical moment equations (e.g. Equation 4.25), the distribution $\pi$ is actually not the first state distribution, but the stationary distribution that the Markov chain converges. So, in fact what we do is to see HMM as a multi-view mixture model, with following conditional expectations (observation matrices):

$$\mathbb{E}[x_1|h] = Odiag(\pi)A^T diag(\pi)^{-1} \tag{4.27}$$

$$\mathbb{E}[x_2|h] = O \tag{4.28}$$

$$\mathbb{E}[x_3|h] = OA \tag{4.29}$$

Then, we see that $\mathbb{E}[x_3 \otimes x_1] = \mathbb{E}[x_3|h]diag(\pi)\mathbb{E}[x_1|h]^T = OAAdiag(\pi)O^T$, which is accordance with 4.25. Also, $\mathbb{E}[x_3 \otimes x_{2,l} \otimes x_1] = OAdiag(O(l,:))Adiag(\pi)O^T = O\mathcal{M}_l Adiag(\pi)O^T$, which is also in accordance with 4.21.

### 4.3. Spectral Learning for Mixture of Markov Models

We introduced the mixture of Markov models in Section 2.2.1. As indicated in that chapter, learning a mixture of Markov models would require to use local search algorithm such as EM. With spectral learning, we aim to estimate the transition matrix $A_k$ of each cluster, solely based on some observable moments, in a local optima free fashion.

As we discussed in the previous sections, according to the method of moments described in [4], the approach is to express the moments of the distribution as a matrix multiplication (or possibly tensor as in our case), so that an eigen-decomposition form which reveals information about the model parameters can be computed as a function of the observable moments.

In order to derive a spectral learning algorithm for a mixture of Markov models following this methodology, since we have three dimensional tensor $p(x_t|\ x_{t-1}, h)$, we use the tensor algebra used in [17]. Let us start with the tensor notation: The tensor multiplication $Z = X \times_k Y$ is defined as follows:

$$Z(i_2, \ldots, i_N, j_2, \ldots, j_M) = \sum_k X(k, i_2, \ldots, i_N)Y(k, j_2 \ldots, j_M) \tag{4.30}$$

where $Z \in \mathbb{R}^{I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M}, X \in \mathbb{R}^{I_1 \times \cdots \times I_N}, Y \in \mathbb{R}^{J_1 \times \cdots \times J_N}$. Note that, the tensor multiplication in Equation 4.30 can also be on multiple indices as in Equation 4.35.

The diagonalization operation is important to express joint distributions. The symbol $\oslash_k$ diagonalizes a particular index $k$ times. For example, $p(\oslash_2 h) = p(h)\delta(h, h') = diag(p(h))$, where $\delta(h, h') = 1$ if $h = h'$, and else zero. Similarly, $p(\oslash_3 h) = p(h)\delta(h, h', h'')$.

Let us consider $p(x_1, x_2)$ to illustrate both tensor multiplication and diagonalization:

$$\begin{aligned} p(x_1, x_2) =& \sum_{k=1}^{K} p(x_2|x_1, h = k)p(x_1|h = k)p(h = k) \\ =& (p(x_2| \oslash_2 x_1, h) \times_{h,x_1} p(x_1| \oslash_2 h)) \times_h p(h) \end{aligned} \tag{4.31}$$

where, we see the tensor algebra equivalent of the regular probability notation. Note that, we diagonalize $x_1$ and $h$, in order to have element wise multiplication with tensor multiplication. The next thing we need is the mode-specific tensor inversion. For this, we need to define an identity tensor with respect to an index set: $I_\sigma$ is the identity tensor with respect to the index set $\sigma$ iff,

$$X \times_\sigma I_\sigma = X \tag{4.32}$$

The inverse tensor of $X$ with respect to identity tensor $I_\sigma$ is defined as follows:

$$X^{-1} \times_w X = I_\sigma \tag{4.33}$$

Here, $w = \mathcal{X} \backslash \sigma$ and $\mathcal{X}$ is the set of all indices of $X$. We will use the tensor algebra to show that, for a mixture of Markov models, we have to use at least a fifth order moment to be able to have an eigen-decomposition of the form,

$$B(i, j, k) := (A(:, i, :) \times_h A(j, k, :)) \times_h A^{-1}(:, i, :) \tag{4.34}$$

which reveals the parameters as eigenvectors and eigenvalues [4]. We use the MATLAB array notation for the transition matrix so that $A(:, :, k) := A_k$. For the sake of simplicity, we assume that $A(:, i, :)$ is invertible.

In practice, we use SVD to deal with invertibility as done in previous sections and [4, 15, 17]. To show the impossibility of identifying $A$ from low order moments, let

us first consider the third order observable moment:

$$p(x_1, x_2, x_3) = ((p(x_3|x_2, h) \times_{x_2,h} p(\oslash_2 x_2|\ x_1, \oslash_2 h)) \times_{x_1,h} p(\oslash_2 x_1|h)) \times_h p(h) \quad (4.35)$$

Note that since the observation model is discrete, $\mathbb{E}[x_1 \otimes x_2 \otimes x_3] = p(x_1, x_2, x_3)$. In this observable moment, the variables $x_2$ and $x_1$ are shared variables, on which we have products. To have a form similar to Equation 4.34, where the products are solely on $h$, we have to clamp the shared variables $x_1$ and $x_2$ to particular values. This reduces the third order observable moment to a vector, on which we cannot do an eigen-decomposition. So, we move up and we consider the fourth order moment. We clamp the shared variables $x_3$ and $x_2$ to particular values $j$ and $i$:

$$\begin{aligned} p(x_4, x_3 = i, x_2 = j, x_1) &= (p(x_4|x_3 = i, h) \times_h p(x_3 = i|x_2 = j, \oslash_2 h)) \times_h p(x_2 = j, x_1, h) \\ &= [(p(x_4|x_3 = i, h) \times_h p(x_3 = i|x_2 = j, \oslash_2 h)) \times_h p(x_3^{'}|x_2 = j, h)^{-1}] \\ &\quad \times_{x_3^{'}} \underbrace{[p(x_3^{'}|x_2 = j, h) \times_h p(x_2 = j, x_1, h)]}_{p(x_3^{'}, x_2 = j, x_1)} \end{aligned} \quad (4.36)$$

From Equation 4.36, one can see that;

$$\begin{aligned} B(x_4, x_3^{'}, x_3 = i, x_2 = j) &= (p(x_4|x_3 = i, h) \times_h p(x_3 = i|x_2 = j, \oslash_2 h)) \times_h p(x_3^{'}|x_2 = j, h)^{-1} \\ &= (A(:, i, :) \times_h A(i, j, :)) \times_h A(:, j, :)^{-1} \\ &= p(x_4, x_3 = i, x_2 = j, x_1) \times_{x_1} p(x_3^{'}, x_2 = j, x_1)^{-1} \end{aligned} \quad (4.37)$$

At this stage it appears that we can obtain an eigen-decomposition form by using the fourth and third order moments. In this case, the eigenvectors are $p(x_4|\ x_3 = i, h) = p(x_3^{'}|x_2 = j, h)$ and eigenvalues are $p(x_3 = i|\ x_2 = j, h)$. However, we realize that following this approach there exists no way to resolve the permutation ambiguity since the ordering of the eigenvalues are arbitrary. Therefore, it is not possible for us to recover the transition matrix of a cluster since columns of an estimated transition matrix can be permuted with those of another cluster's transition matrix. The condition to resolve this nuisance is to have the indices of the eigenvalue matrix independent of the

eigenvalues, which can ensure the correctness in the columns of the transition matrices. For this reason we consider the fifth order moment. We clamp the shared variables $x_1, x_2, x_4$.

$$
\begin{aligned}
&p(x_5, x_4 = i, x_3, x_2 = j, x_1 = k) = \\
&[(p(x_5|x_4 = i, h) \times_h p(x_4 = i|x_3, \oslash_2 h)) \times_{h,x_3} p(\oslash_2 x_3|x_2 = j, \oslash_2 h)] \times_h p(x_2 = j, x_1 = k, h) \\
&= [(p(x_5|x_4 = i, h) \times_h p(x_2 = j|\ x_1 = k, \oslash_2 h)) \times_h p(x_5^{'}|x_4 = i, h)^{-1}] \\
&\qquad \times_{x_5^{'}} \underbrace{([p(x_5^{'}|x_4 = i, h) \times_h p(x_4 = i|x_3, \oslash_2 h)] \times_{h,x_3} p(\oslash_2 x_3, x_2 = j, h))}_{p(x_5^{'}, x_4 = i, x_3, x_2 = j)} \qquad (4.38)
\end{aligned}
$$

Then, we see that the eigendecomposition form $B(i, j, k)$ is;

$$
\begin{aligned}
B(x_5, x_5^{'}, x_4 = i, x_2 = j, x_3 = k) = & \\
&(p(x_5|x_4 = i, h) \times_h p(x_2 = j|\ x_1 = k, \oslash_2 h)) \times_h p(x_5^{'}|x_4 = i, h)^{-1} \\
=&(A(:, i, :) \times_h A(j, k, :)) \times_h A^{-1}(:, i, :) \\
=&p(x_5, x_4 = i, x_3, x_2 = j, x_1 = k) \times_{x_3} p(x_5^{'}, x_4 = i, x_3, x_2 = j)^{-1} \qquad (4.39)
\end{aligned}
$$

Note that, although $x_3$ is a shared variable we do not clamp it since it does not spoil the eigen-decomposition form $B$. Now, the good news is, for differing $x_4$, we have the same eigenvalues if we use the same $x_2$ and $x_1$. Therefore, we can use the eigenvalue-eigenvector correspondence to ensure the consistency in the columns of the transition matrix estimates (In practice, eigenvalues for every $B(i, j, k)$ may not be the same, but one can order the eigenvectors according to the eigenvalue ordering.).

However, the downside of it is that we have to use the fifth order moment. To have accurate estimates of moments this order, we need large number of samples. In this respect, it is simply not practical to use this method, as in Equation 4.39 with the fifth order moment. In the next section, we propose an alternative scheme for learning mixture of Markov models, to reduce the sample complexity.

### 4.3.1. Spectral Learning for a Mixture of Dirichlet Distributions

Suppose we place a prior distribution for transition matrices on the class conditional likelihood $p(\mathbf{x}_n|A_{h_n}, h_n)$. Placing a Dirichlet prior $p(A_{h_n}) \sim \text{Dirichlet}(\beta, \ldots, \beta)$ would result in a Dirichlet posterior:

$$\begin{aligned} p(A_{h_n}|\mathbf{x}_n, h_n) \propto & \quad p(\mathbf{x}_n|A_{h_n}, h_n)p(A_{h_n}) \\ = & \quad \text{Dirichlet}(c^n_{1,1} + \beta - 1, c^n_{1,2} + \beta - 1, \ldots, c^n_{L,L} + \beta - 1) \end{aligned}$$

where, $c^n_{l_1,l_2}$ stores the state transition counts of sequence $\mathbf{x}_n$. So, we can indeed characterize a sequence generated by a Markov model with a Dirichlet distribution, since Dirichlet distribution is the posterior of the transition matrix $A_{h_n}$. Setting $\beta = 1$ (having a uniform prior), we see that the posterior distribution becomes

$$p(A_{h_n}|\ \mathbf{x}_n, h_n) = \text{Dirichlet}(c^n_{1,1}, c^n_{1,2}, \ldots, c^n_{L,L}).$$

Therefore, we can treat a normalized sufficient statistics matrix as a sample from the posterior of the transition matrix. In the mixture of Markov models, every sequence $\mathbf{x}_n$ has a distinct cluster label $h_n$. So, sufficient statistics matrix of each sequence is indeed a sample from the posterior of $A_{h_n}$. Therefore, we can effectively cluster sequences by using normalized second order statistics matrices instead of the sequences themselves. A spectral learning algorithm for a mixture of Dirichlet distributions would be simpler compared to directly learning a mixture of Markov models, since the former requires a second order moment (sufficient statistics matrix) whereas the latter requires a fifth order moment, as shown in Section 4.3. Let us denote an observed second order statistics (vectorized) matrix by $s_n \in \mathbb{R}^{(L\times L)\times 1}$. Note that $s_{n,l_1,l_2} = c^n_{l_1,l_2}/(\sum_{l_1,l_2} c^n_{l_1,l_2})$. The likelihood of a sufficient statistics observation is defined as follows:

$$p(s_n|\alpha) = \sum_{k=1}^{K} p(h_n = k)\text{Dirichlet}(s_n, \alpha(:, k)) \tag{4.40}$$

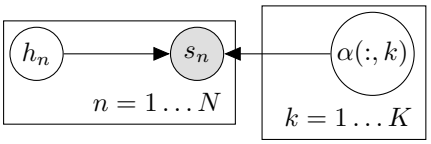



Figure 4.5. DAG of the mixture of Dirichlet distributions.

where, $\alpha \in \mathbb{R}^{(L\times L)\times K}$ is the matrix that stores the Dirichlet parameters in its columns. We write the second and third order observable moments as follows;

$$\begin{aligned}\mathbb{E}[s_n \otimes s_n] =& \sum_{k=1}^{K} p(h_n = k)\mathbb{E}[s_n \otimes s_n | h_n = k] \\ & \mathbb{E}[s_n|h_n] diag(p(h_n))\mathbb{E}[s_n|h_n]^T \\ =& \alpha diag(p(h_n))\alpha^T \end{aligned} \tag{4.41}$$

$$\begin{aligned}\mathbb{E}[s_n \otimes s_n \otimes s_{n,i}] =& \sum_{k=1}^{K} p(h_n = k)\mathbb{E}[s_n \otimes s_n \otimes s_{n,i} | h_n = k] \\ =& \mathbb{E}[s_n|h_n] diag(\mathbb{E}[s_{n,i}|h_n]) diag(p(h_n))\mathbb{E}[s_n|h_n]^T \\ =& \alpha diag(\alpha(i,:)) diag(p(h_n))\alpha^T \end{aligned} \tag{4.42}$$

where, $\otimes$ is the outer product operator and $s_{n,i}$ denote the $i$'th entry of the observed statistics vector $s_n$. Then, following [4], we can conclude that the eigenvectors of the auxilary matrix,

$$\begin{aligned} B_i =& (U^T\mathbb{E}[s_n \otimes s_n \otimes s_{n,i}]V)(U^T\mathbb{E}[s_n \otimes s_n]V)^{-1} \\ =& (U^T\alpha) diag(\alpha(i,:))(U^T\alpha)^{-1} \end{aligned} \tag{4.43}$$

Based on the fact that $\mathbb{E}[s_{n,i}|h_n = k] = \alpha(i,k)/\alpha_0$, where $\alpha_0 = \sum_i \alpha(i,k)$, we can in fact recover the Dirichlet parameters $\alpha(:,k)$ for each cluster up to a scaling factor $\alpha_0$ (as done in [24], we fix $\alpha_0$ to a particular value), i.e. the $\alpha$ matrix by doing the eigendecomposition of $B_i$. We compute the SVD, $\mathbb{E}[s_n \otimes s_n] = U\Sigma V^T$ to estimate $U$ and $V$ as done in [4]. Finally, note that in order to increase the robustness of the algorithm, we use $\mathbb{E}[s_n \otimes s_n \otimes s_n](\eta) = \sum_{i=1}^{L\times L} \eta_i \mathbb{E}[s_n \otimes s_n \otimes s_{n,i}]$ instead of using $\mathbb{E}[s_n \otimes s_n \otimes s_{n,i}]$, where $\eta \in \mathbb{R}^{(L\times L)\times 1}$, which results in using $B(\eta)$. The details on choosing $\eta$ can be found in [4]. We give the overall sequence clustering algorithm in Figure 4.3.1.

**Input:** Sequences $\mathbf{x}_{1:N}$

**Output:** Clustering assignments $\widehat{h}_{1:N}$

1. Extract the sufficient statistics $s_n$ from $x_n, \forall n \in \{1, \ldots, N\}$
2. Compute empirical moment estimates for $\mathbb{E}[s_n \otimes s_n]$ and $\mathbb{E}[s_n \otimes s_n \otimes s_n]$. Compute $U$ and $V$ such that $\mathbb{E}[s_n \otimes s_n] = U\Sigma V^T$.
3. Estimate $\alpha$ by doing eigen-decomposition of $B(\eta)$.
4. $\forall n \in \{1, \ldots N\}, \widehat{h}_n = \arg\max_k \text{Dirichlet}(s_n, \alpha(:, k))$.

Figure 4.6. Algorithm for clustering sequences via spectral learning of mixture of Dirichlet distributions.

### 4.4. Incorporating Spectral Learning for learning HMM mixtures

As discussed in Section 3.4, learning a mixture HMMs can be an expensive task with expectation maximization algorithm, since we have to do forward-backward recursions for each sequence at each iteration of EM. In this section, we propose a novel algorithm for clustering HMMs. According to the spectral method in Section 4.2, we can learn a HMM just by using some low order observable moments. The proposed HMM clustering algorithm replaces the parameter estimation step (M-Step) in the hard clustering algorithm in Section 3.4 with a spectral learning algorithm. This way, required computations for parameter estimation step become $K$ low-rank SVDs and $K$ eigen decompositions, which is substantially cheaper compared to $N$ forward-backward recursions. The proposed algorithm is given in Figure 4.4.

At the start of the algorithm, we collect the un-normalized empirical statistics

$$\tilde{P}^n_{2,1} = \sum_{t=2}^{T_n} x_{t,n} \otimes x_{t-1,n} \tag{4.44}$$

$$\tilde{P}^n_{3,2,1} = \sum_{t=3}^{T_n} x_{t,n} \otimes x_{t-1,n} \otimes x_{t-2,n} \tag{4.45}$$

Input : Sequences $\mathbf{x}_{1:N}$, Number of Cluster $K$

Output : Cluster assignments $h_{1:N}$.

**for** $n = 1 \rightarrow N$ **do**

1. Collect unnormalized statistics $\tilde{P}_2^n$ and $\tilde{P}_3^n$

**end for**

Randomly initialize $h_{1:N}$

**for** $e = 1 \rightarrow maxiter$ **do**

**for** $k = 1 \rightarrow K$ **do**

Compute $P_{2,1}^k$ and $P_{3,2,1}^k$

$\theta_k \leftarrow$ SpectralHMM( $P_{2,1}^k$, $P_{3,2,1}^k$) - (Algorithm in Figure 4.2.1)

**end for**

**for** $n = 1 \rightarrow N$ **do**

$h_i = \arg\max_k p(y_n|\theta_k)$

**end for**

**end for**

Figure 4.7. Algorithm for learning a mixture of HMMs with spectral learning.

Then, according to the cluster assignments $h_{1:N}$ at iteration $e$, we use the corresponding $\tilde{P}_{3,1}^n$ and $\tilde{P}_{3,2,1}^n$ to compute the empirical statistics of the cluster $k$. That is, we set

$$P_{2,1}^k = \frac{1}{\tilde{N}} \sum_{n:h_{1:N}=k} \tilde{P}_{3,1}^n \tag{4.46}$$

$$P_{3,2,1}^k = \frac{1}{\tilde{N}} \sum_{n:h_{1:N}=k} \tilde{P}_{3,2,1}^n \tag{4.47}$$

where $\tilde{N}$ is the appropriate normalizer so that $P_{2,1}^k$ and $P_{3,2,1}^k$ are probability distributions. And then we input these empirical statistics to the spectral learning algorithm given in Figure 4.2.1). Note that, since we input the empirical statistics, we skip the first step of the algorithm in Figure 4.2.1.

Note that, unlike the EM algorithm for mixture of HMMs, the computational complexity of the parameter estimation step is independent from the number of sequences $N$, since we extract the empirical statistics at the beginning of the algorithm, and use them according to the latest update of the cluster indicator variables $h_{1:N}$. This heuristic scheme gives us a very fast and simple algorithm. In Section 5.2.2, we compare the speed and performance of this algorithm with expectation-maximization approach. Finally note that, we give the number the number of cluster $K$ as an input to the algorithm. In following section, we propose an algorithm that automatically determines the number of clusters.

### 4.4.1. Learning Infinite Mixtures of HMMs using spectral learning

It is possible to extend the algorithm 4.4 to automatically find the number of clusters $K$. In order to do that, we consider an infinite mixture of HMMs. To derive a learning algorithm for an infinite mixture of HMMs, one has to take an intractable integral over the HMM parameters. Incorporating spectral methods in learning infinite mixtures of HMMs provides a natural way of handling with the arising intractable integrals. To illustrate the difficulty in learning an infinite mixture of HMMs, let us first consider the mixture model in Section 4.1, where the parameters of the $k$'th cluster is $\theta_k = O(:,k)$. We place a prior on the model parameters $p(\theta_{1:K}) = \prod_{k=1}^{K} p(\theta_k)$. Moreover, let us place a Dirichlet prior on the mixing proportions $\pi := p(h_n)$, $\pi \sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K)$. Then, one can show that;

$$\begin{aligned} p(h_n = k | h_{1:N}^{-n}, \mathbf{x}_n) &= \frac{1}{Z} \int p(\mathbf{x}_n, h_{1:N} |\ \theta, \pi) p(\theta) p(\pi) d\theta d\pi \\ &= \frac{N_k^{-n} + \alpha/K}{N + \alpha - 1} \int p(\mathbf{x}_n | \theta_k) p(\theta_k | \{\mathbf{x}_l : l \neq n, r_l = k\}) d\theta_k \end{aligned} \tag{4.48}$$

where, $p(\theta_k | \{\mathbf{x}_l : l \neq n, r_l = k\})$ is the posterior of the parameter of cluster $k$, with data assigned to cluster $k$ (except data item $n$). The notation $N_k^{-n}$ denotes the number of data items in cluster $k$ except data item $n$. We use the same notation for excluding variables as well. For example $x_{1:N}^{-n}$ denotes all $\mathbf{x}_{1:N}$ except $\mathbf{x}_n$.

Now, let $K^*$ denote the number of occupied clusters. So we have $K - K^*$ empty clusters. Let us lump all the empty clusters together. So, assignment to an empty cluster has the probability;

$$p(h_n = k|h_{1:N}^{-n}, \mathbf{x}_n) = \frac{\alpha \frac{K-K^*}{K}}{N + \alpha - 1} \int p(\mathbf{x}_n|\theta_k)p(\theta_k)d\theta_k \tag{4.49}$$

Let us take $K \to \infty$. Then we have the following densities:

- For an assignment to an occupied cluster, $\forall k \leq K$:

$$p(h_n = k|h_{1:N}^{-n}, \mathbf{x}_{1:N}) = \frac{N_k^{-i}}{N + \alpha - 1} p(x_n|\{\mathbf{x}_l : l \neq n, r_l = k\}) \tag{4.50}$$

- For an assignment to an empty cluster, $\forall k > K$:

$$p(h_n = k|h_{1:N}^{-n}, \mathbf{x}_{1:N}) = \frac{N_k^{-i}}{N + \alpha - 1} p(\mathbf{x}_n) \tag{4.51}$$

For a model with conjugate prior on $\theta_{1:K}$ the integrals required for Equation 4.50 and Equation 4.51 are tractable. For instance, for a Gaussian mixture model with observation model $\theta_k = \mu_k$, and $p(\mu_k) = \mathcal{N}(\mu_k; \mu_0, \sigma_0^2)$, these integrals are tractable.

However, for a mixture of HMMs, Equation 4.50 and Equation 4.51 require integrating over the model parameters $\theta = (A, O)$ and summing over the latent state sequences $\mathbf{r}_n$. This joint sum-integral is not tractable since we no longer have the conditional independence properties of the latent state sequence $\mathbf{r}_n$, due to the fact that we integrate over the transition matrix $A$, which makes the latent state variables $\mathbf{r}_n$ dependent. So, we conclude that a conventional collapsed Gibbs sampler for an infinite mixture of HMMs is not feasible. Assuming that the sequences are long-enough, we make the following approximation:

$$\int p(\mathbf{x}_n|\theta_k)p(\theta_k|\{\mathbf{x}_l : l \neq n, r_l = k\})d\theta_k \approx p(\mathbf{x}_n|\theta_k^{spectral}) \tag{4.52}$$

where, the right hand side is the likelihood of the sequence for the HMM parameters

corresponding to cluster $k$, estimated with spectral learning algorithm given in algorithm 4.2.1. This approximation is sensible since, in real world scenarios the sequences are long enough to make the posterior distribution of the HMM parameters $\theta_k$ peaked enough. Next, we consider the Equation 4.51 for creating new clusters. This requires the marginal likelihood $p(\mathbf{x}_n)$ of a sequence $\mathbf{x}_n$. There are well established methods in the Bayesian inference literature to compute the marginal likelihood [26]. In a clustering algorithm, marginal likelihoods $p(\mathbf{x}_n), n \in \{1, \ldots, N\}$ can be inferred before starting the algorithm and then be utilized in each iteration without having need to recompute them in each iteration. Having made these observations, we propose the algorithm given in Figure 4.4.1. Note that we dropped the multiplicative terms in front

Input : Sequences $\mathbf{x}_{1:N}$, Number of Cluster $K$

Output : Cluster assignments $r_{1:N}$.

**for** $n = 1 \rightarrow N$ **do**

1. Collect unnormalized statistics $P_2^n$ and $P_3^n$
2. Compute marginal likelihood $p(\mathbf{x}_n)$

**end for**

Initialize $h_n = 1$, for $i = 1, \ldots, N$

**for** $e = 1 \rightarrow maxiter$ **do**

**for** $k = 1 \rightarrow K$ **do**

Compute $P_{2,1}^k$ and $P_{3,2,1}^k$

$\theta_k \leftarrow$ SpectralHMM( $P_{2,1}^k$, $P_{3,2,1}^k$) - (Algorithm in Figure 4.2.1)

**end for**

**for** $n = 1 \rightarrow N$ **do**

$h_n = \arg\max_k [p(\mathbf{x}_n|\theta_k) \quad p(\mathbf{x}_n)]$

**if** $h_n > K$ **then**

$K = K + 1$

**end if**

**end for**

**end for**

Figure 4.8. Learning an infinite mixture of HMMs with spectral learning

of the Equation 4.50 and Equation 4.51 since their contribution is neglible compared to the result of the integral. Note that algorithm in Figure 4.4.1 has the same procedural steps with the hard clustering algorithm for Dirichlet processes recently proposed in [19].

# 5. EXPERIMENTS AND RESULTS

In this chapter, the results are organized into three main topics. First, we give the mixture of Markov model related sequence clustering results. Secondly, we give the mixture of HMMs related sequence clustering results. Lastly, we present the generative vs. discriminative model related sequence classification results.

## 5.1. Sequence Clustering via mixture of Markov models

In this section, we give our results corresponding to our spectral learning algorithm for mixture of Markov models in Section 4.3. We compare the related learning algorithms to learning mixture of Markov models on synthetic data and network flow data. We also show on motion capture data that the algorithm in Section 4.3 can be used for finding the number of clusters in data.

### 5.1.1. Synthetic Data

We generated 100 synthetic data sets from mixture of Markov models. Each data set consist of 60 sequences divided equally among 3 clusters. The transition matrices of Markov models are generated randomly from a uniform Dirichlet distribution, i.e. $A(:, j, k) \sim \text{Dirichlet}(1, \ldots, 1)$.

We compare the clustering accuracies of the spectral learning algorithms in Section 4.3, Section 4.3.1 and expectation maximization algorithms for mixture of Markov models and mixture of Dirichlet distributions (Derivation of the EM algorithm for mixture of Dirichlet distributions can be found in Appendix C). The clustering accuracy is defined as the ratio of true cluster assignments in a data set and calculated by resolving the permutation ambiguity introduced by the clustering processes. After each clustering, the estimated assignments are compared with the true assignments for all possible permutations of cluster identifiers ($K!$ permutations for $K$ clusters), and the maximum ratio is chosen.

Figure 5.1 shows the average accuracies of the methods in the data sets with respect to different sequence lengths. In the experiments each data set is used 6 times where the sequences are cropped at different lengths and the number of clusters in the data is given to the algorithms apriori.

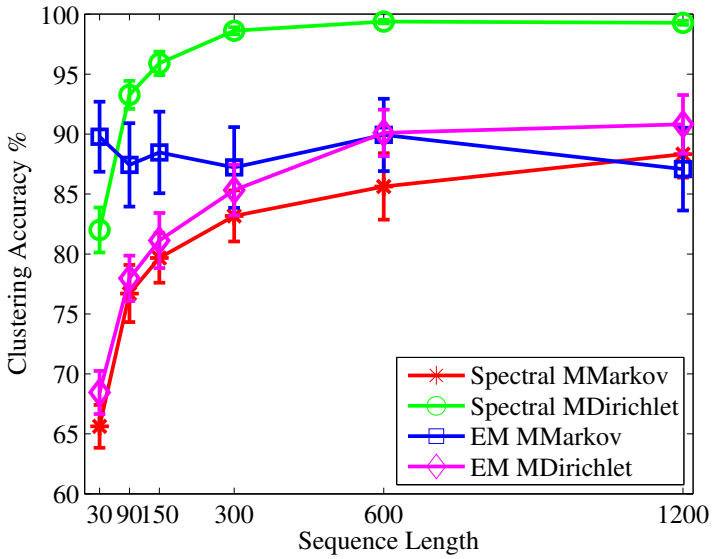



Figure 5.1. Comparison of clustering accuracies on synthetic data.

We see that except when the sequences are short, the spectral learning algorithm for mixtures of Dirichlet approach outperforms other algorithms. As expected, spectral learning for Mixture of Markov model algorithm fails to perform well in the absence of abundant data. EM algorithms do not to perform as well as the spectral Mixture of Dirichlet algorithm on the average, because of the possible unlucky random initializations. We conclude that spectral learning algorithm for a mixture of Dirichlet distributions perform better than EM algorithms since it does not require good initialization. It is better than the spectral learning algorithm for mixture of Markov models since it requires less data for accurate moment estimates.

### 5.1.2. Real World Data

We conducted real world experiments on a network traffic data set, collected by recording the network activity at Boğaziçi university. The observations in the data set are network flows, where a network flow is defined as a series of network packets

transferred between two *IP-Port* pairs. We select the length and the transfer direction of a packet as features. By quantizing the packet sizes into five levels, and adding direction information, we produce an alphabet of 10 discrete symbols. The flows are represented as arrays of those symbols, and assumed to be generated by a Markov model. Hence, our data set is assumed to be generated by a mixture of Markov models.

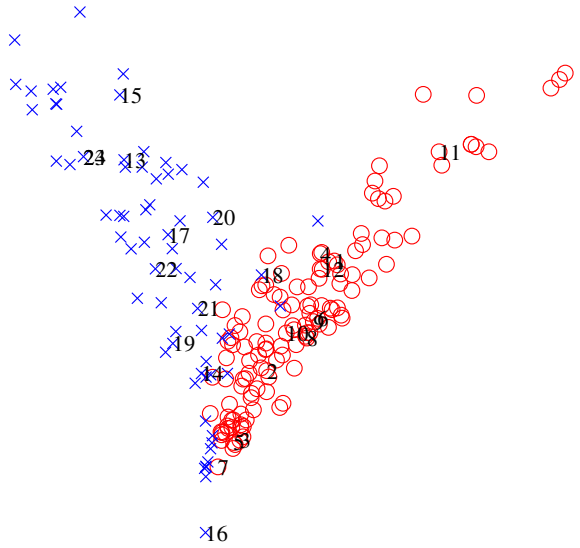
On top of Figure 5.2, we see the multidimensional scaling of a fraction of the data set, which contains 2 clusters, found by the spectral mixtures of Dirichlet algorithm, mapped to a two dimensional space for visualization. On bottom are the sufficient statistics, i.e. $10 \times 10$ state transition counts of 24 randomly selected flows. We see that flows from the blue cluster (crosses) has more mass assigned to the top left corner, whereas the red cluster (circles) tend to have more mass around the center.

In our flow classification experiment, we try to infer the web application to which the flow belongs. We gathered the true application labels for the data by using *IPOQUE* [27] application classification software. This software uses deep packet inspection technique, which is the most accurate known way of application detection [28], and we accept its results as our ground truth.

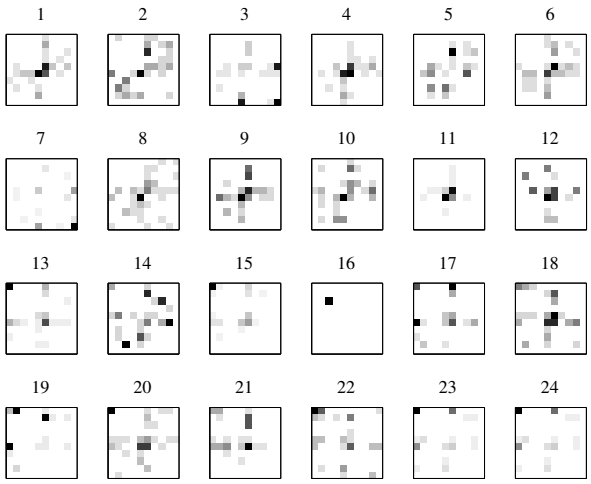
We choose the class conditional densities of flows as a mixture of Markov model. In other words, each web application is composed of different types of flows. We tried to classify two different applications: *Skype* and *BitTorrent*. 1574 *Skype* and 415 *BitTorrent* flows are divided into training and test sets equally and our algorithms estimated mixture model parameters for each application type. In classification, a test flow is labeled according to its likelihood under the two mixture models. Table 5.1 shows the classification accuracy of the three different algorithms.

### 5.1.3. Finding number of clusters

Using the spectral method provides a natural way for finding the number of clusters. Since the second order moment matrix $\mathbb{E}[s_n \otimes s_n]$ is a sum of rank one




a. Multi Dimensional Scaling of the data to 2D.




b. Multi Dimensional Scaling of the data to 2D.

Figure 5.2. Exploratory data analysis on network traffic data.

matrices, we can determine the number of clusters by using singular values:

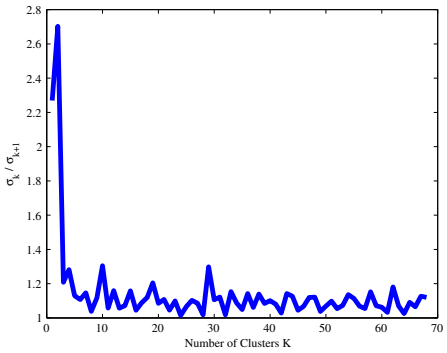
$$\mathbb{E}[s_n \otimes s_n] = \alpha diag(p(h))\alpha^T = \sum_{k=1}^{K} p(h=k)\alpha(:,k)\alpha(:,k)^T \tag{5.1}$$

Table 5.1. Classification accuricies of three clustering algorithms on network traffic data.

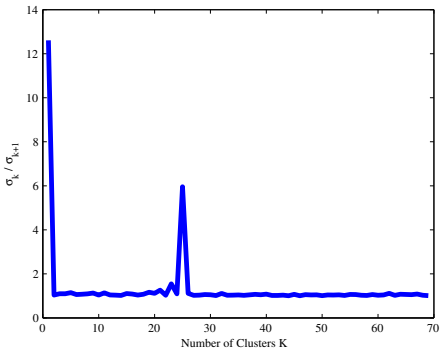
| Algorithm | Classification Accuracy |
|---|---|
| No Clustering | 63.41% |
| Mixture of Dirichlet (Spectral) | 84.85% |
| Mixture of Dirichlet (EM) | 79.39% |
| Mixture of Markov (EM) | 83.51% |

We observe that the ratio of the consecutive singular values $\sigma_K/\sigma_{K+1}$ is a sensible measure for determining the number of clusters $K$. We plot $\sigma_K/\sigma_{K+1}$ vs $K$ curves. To justify the number of clusters found by the algorithm, we also plot example cluster assignments.

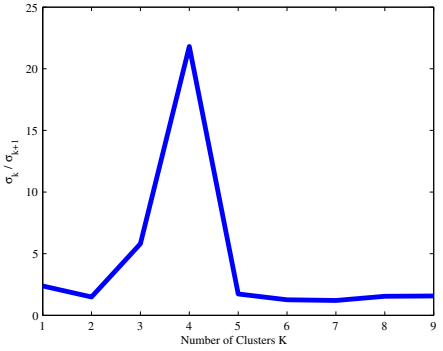
In Figure 5.3a, we used network data used in Figure 5.2 for exploratory data analysis. We see that that maximum $\sigma_K/\sigma_{K+1}$ is attained when $K = 2$, as we also saw a two cluster behavior in the previous section. In Figure 5.3b, we used synthetic data with 25 clusters. We see that the biggest jump corresponds to cutting off from the first and 25'th singular values. Since having only one cluster would be trivial, we identify the number of clusters as 25. In Figure 5.3c and Figure 5.3d, we used respectively, sequence sets Simple Walk vs Run and, Run(#9) vs Run/Jog(#35) from CMU motion capture dataset [29], as used in [30]. The data in its original form is 62 dimensional and continuous. We discretize the data using vector quantization by k-means clustering. To demonstrate that maximum of the consecutive singular values ratio vector indeed gives the number of clusters, we also show the example cluster assignments in Figure 5.3e and Figure 5.3f, where each column shows the data items in the same cluster. In Figure 5.3e, we see that the algorithm successfully identifies the clusters, where each cluster has distinctive features. We see that even if the second cluster has only one data item in it, the algorithm successfully identifies it as a cluster.



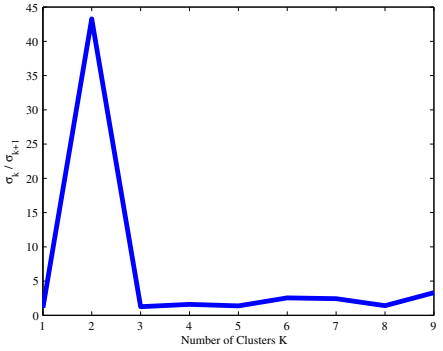

a. Network Traffic data.



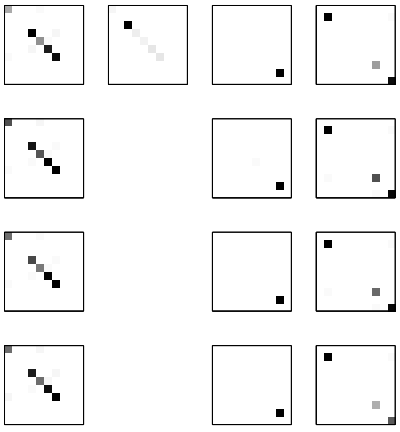

b. Synthetic data.



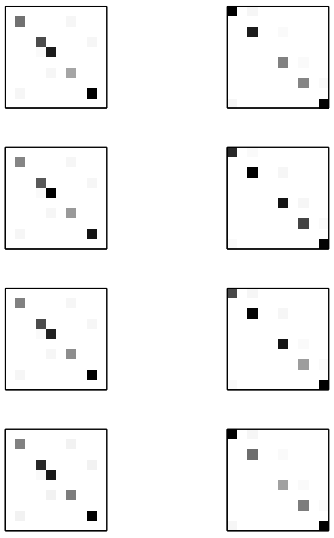

c. Simple Walk vs Run




d. Run(#9) vs Run/Jog(#35)



e. Simple Walk vs Run cluster representatives.



f. Run(#9) vs Run/Jog(#35) cluster representatives.

Figure 5.3. Finding the number of clusters on various data types.

## 5.2. Sequence clustering via learning mixtures of HMMs

In this section, we present our results obtained with the algorithms we propose in Section 4.4. We first present the results in our work [5], based on sequence clustering using a finite mixture of HMMs. We then present our work in [6], which is based on sequence clustering using an infinite mixture of HMMs. We use clustering to increase the classification accuracy of human action sequences.

### 5.2.1. Toy Problem

As a toy problem, we clustered shape sequences, recorded by manual mouse clicking. We input the first temporal derivative of the 2D coordinates to the algorithm. We used a 2D Isotropic Gaussian emission HMM. Example clustering results obtained with the proposed algorithm are given Figure 5.4 and Figure 5.5. Sequences that are assigned to the same cluster are plotted on top of each other. We have observed that our method converges to the true solution very fast and accurately.

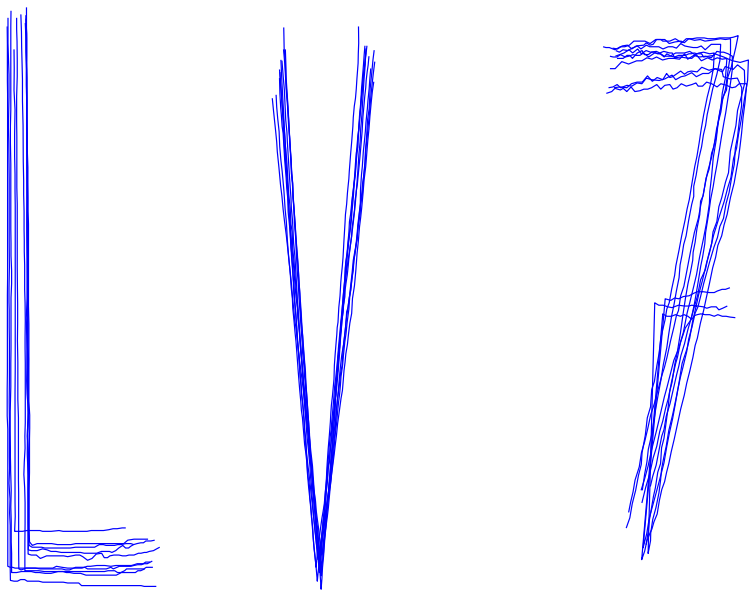


Figure 5.4. Clustering handwritten L, V, 7.

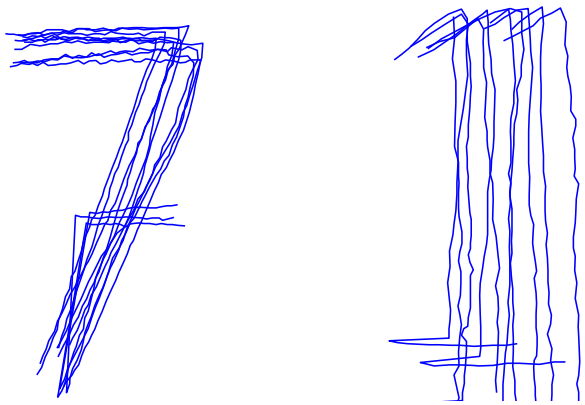


Figure 5.5. Clustering handwritten 7 and 1.

### 5.2.2. Clustering Motion Capture Data with Finite Mixture Models

We tested our algorithm also for clustering motion capture data. We compared it with soft and hard versions of the EM algorithm presented in Section 3.4. We used the database HDM05 [31]. We input the first temporal derivative of the 3D trajectories of 32 joints. The sequences are on average of length 500, and $L = 96$ dimensional. We tested on several binary clustering scenarios for different action pairs. We used a $M = 3$ state HMM. The observation model is taken as multidimensional isotropic Gaussian. In Table 5.2, we compare the average clustering accuracies, maximum accuracy, time required for one iteration, and average convergence time, for $N = 20$ "kicking" and "punching" actions, for 10 restarts with random initializations for cluster indicator variables $h_{1:N}$. We have seen that EM tends to get stuck in local optima because of poor parameter initialization, which results in sub-optimal clustering accuracy. Since the spectral method does not suffer from initialization problem, this algorithm reaches the optimal clustering more often than EM. In addition to the improvement in clustering accuracy, the real gain is in computational effort. The time required per iteration is almost two times smaller than the hard version of EM and three times smaller than the soft version of EM. Average number of iterations required for EM seems to be less than spectral method. This makes sense since, when initialized with a "bad" partitioning, EM tends to converge to local optima rather quickly, while the spectral learning method strives to find the true solution, which takes more iterations. An example clustering result is given in Figure 5.6.

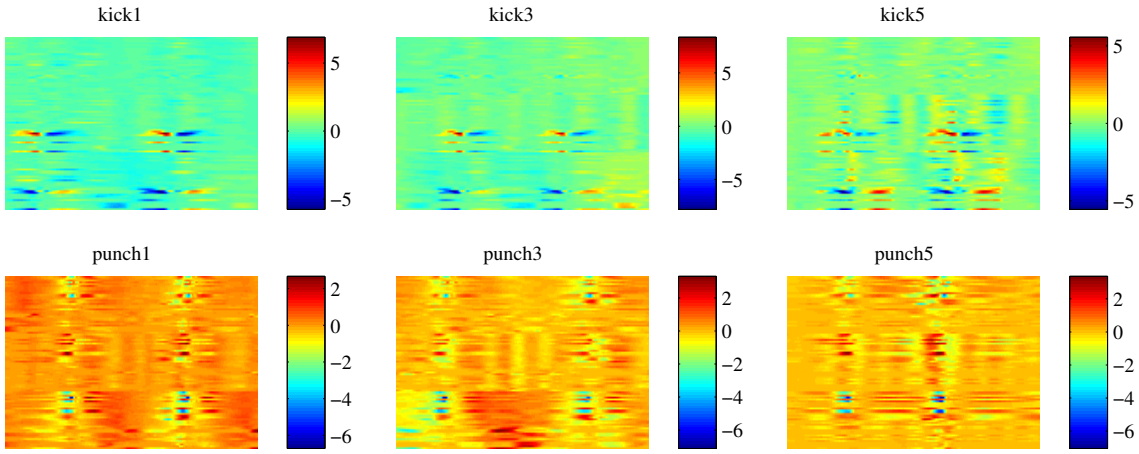



Figure 5.6. Representative examples from clustering of "punching" and "kicking" sequences. The horizontal axis is time and the vertical axis indexes the joints on the body.

Table 5.2. Speed comparison of EM1:Hard EM, EM2: Soft EM.

| | Average Success(%) | Max Success(%) | Iteration time* (s) | Average convergence |
|---|---|---|---|---|
| EM1 | 70 | 100 | 7.5 | 3.2 |
| EM2 | 73 | 100 | 12 | 2.9 |
| Spectral | 76 | 100 | 3.1 | 3.8 |

*PC: 3.33 GHz dual core CPU, 4 GB RAM, Software: MATLAB

### 5.2.3. Application: Human Action Recognition from Videos with HMMs

We tested our clustering algorithm on KTH Action Database [1]. This dataset contains 600 sequences of 25 people collected in four sessions. There are in total 6 actions: Boxing, hand clapping, hand waving, running, jogging and walking. Example sequences are given in Figure 1.2.

In each action frame we place a bounding box on the human body. Then, this box is divided into $L$ blocks. We set $L = 9$ in our experiments. We characterize each block by the count of the spatio-temporal interest points (STIPs). In other words, for any given time $t$, the count of STIPs in each box are the observations $x_t \in \mathbb{Z}^L$. Since it

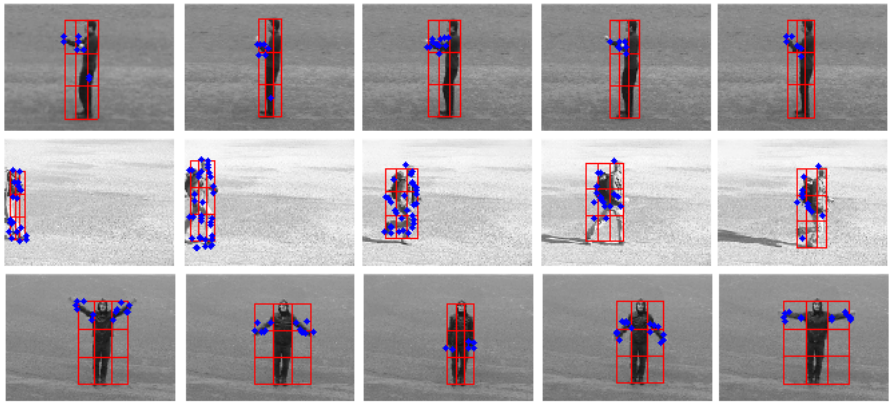


Figure 5.7. Example scenes from KTH Human Action Database [1]. Each row correspond to an example action sequence. The feature vectors are extracted by counting the STIPs that fall into different regions on human body.

is a count data, we use a multivariate Poisson observation model. The box and interest points are automatically extracted by using the off-the-shelf code in [32]. Example scenes with corresponding STIPs are shown in Figure 5.7. We illustrate the sequence extraction process in Figure 5.8.

For each class, we use 64 sequences for training and 36 sequences for testing. In the training step, we trained a separate multiple observation HMM per action class using EM and spectral learning described in Section 4.2. In the testing step, we assign a test sequence to the highest likelihood action class. We achieved 70% and 68% test accuracies with EM and spectral method, respectively. Secondly, we did clustering and trained more than one HMM for each action class using the Algorithm for learning infinite mixture of HMMs suggested in Section 4.4.1. We also compare our algorithm with a more conventional Gibbs sampling with auxiliary parameter method as described in Appendix B. During testing, we did the class assignments according to the likelihoods averaged over the clusters for each class. Consequently, the performance has increased by 6% to reach 74%. Confusion matrices are given in Table 5.3. This result indicates that our clustering algorithm based on learning Infinite mixture of HMMs yields comparable increments in the classification accuracy, which validates our algorithm.

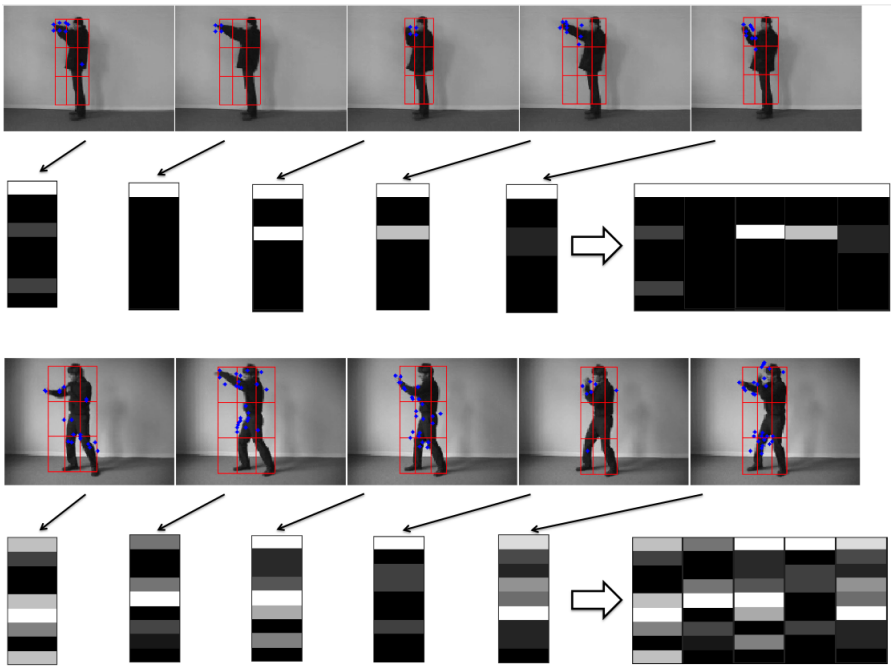


Figure 5.8. Feature extraction process for human action videos. Notice that, action instantiations within a same action class (boxing in this case) tend show variability. In the first sequence there is no motion in the legs, whereas in the second we have substantial leg movement. So, it makes sense to learn a mixture of HMMs for each class in training phase, to better capture the distribution.

### 5.3. Generative vs. Discriminative Models

We use the sequence data used in [2], to compare the classification accuracy of the discriminative Markov model and the generative Markov model. The dataset has 6 classes of sequences, with 100 instances for each class. Each sequence is univariate, continuous and of length 50. We discretize the sequences using 60 levels. Example sequences are given in Figure 5.9. Confusion matrices obtained with 70 sequences for each class in training, and 30 sequences for each class in testing are given in Table 5.4.

We see that discriminative Markov model outperforms its generative counterpart. Seeing this result, one may conclude that discriminative models are superior than generative models in supervised classification.

However, this is true, only when there is an abundance of training data [33]. In

Table 5.3. Confusion matrices obtained from human action video classification. B: Box, HC: Handclap, HW: Handwave, J: Jog, R: Run, W: Walk. The confusion matrices obtained for training with EM and Spectral learning are given in the top row. In the bottom row, we show the confusion matrices corresponding to doing clustering in training phase.

EM, 70.1 %

| | B | HC | HW | J | R | W |
|---|---|---|---|---|---|---|
| B | **32** | 4 | 1 | 0 | 1 | 0 |
| HC | 1 | **31** | 6 | 0 | 1 | 0 |
| HW | 0 | 1 | **29** | 0 | 0 | 0 |
| J | 1 | 0 | 0 | **17** | 20 | 3 |
| R | 0 | 0 | 0 | 7 | **10** | 0 |
| W | 2 | 0 | 0 | 12 | 4 | **33** |

Spectral, 68.1 %

| | B | HC | HW | J | R | W |
|---|---|---|---|---|---|---|
| B | **32** | 7 | 0 | 0 | 2 | 0 |
| HC | 0 | **20** | 0 | 0 | 1 | 0 |
| HW | 1 | 9 | **36** | 0 | 0 | 0 |
| J | 0 | 0 | 0 | **7** | 7 | 0 |
| R | 0 | 0 | 0 | 11 | **17** | 1 |
| W | 3 | 0 | 0 | 18 | 10 | **35** |

Clustering with Sampling, 74.0 %

| | B | HC | HW | J | R | W |
|---|---|---|---|---|---|---|
| B | **32** | 2 | 1 | 0 | 2 | 0 |
| HC | 1 | **33** | 2 | 0 | 0 | 0 |
| HW | 0 | 1 | **34** | 0 | 0 | 0 |
| J | 0 | 0 | 0 | **18** | 19 | 6 |
| R | 0 | 0 | 0 | 4 | **13** | 0 |
| W | 3 | 0 | 0 | 14 | 4 | **30** |

Clustering with Spectral, 74.1 %

| | B | HC | HW | J | R | W |
|---|---|---|---|---|---|---|
| B | **32** | 5 | 1 | 0 | 2 | 0 |
| HC | 1 | **25** | 0 | 0 | 0 | 0 |
| HW | 0 | 6 | **35** | 0 | 0 | 0 |
| J | 0 | 0 | 0 | **13** | 10 | 1 |
| R | 0 | 0 | 0 | 8 | **20** | 0 |
| W | 3 | 0 | 0 | 15 | 4 | **35** |

Table 5.4. Confusion matrices obtained from classification experiment for generative and discriminative Markov models. Corresponding classification accuracies are given on top of the confusion matrices.

Discriminative Markov model, 84.67%

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | **50** | 0 | 0 | 0 | 4 | 4 |
| B | 0 | **31** | 0 | 0 | 0 | 0 |
| C | 0 | 0 | **48** | 0 | 8 | 0 |
| D | 0 | 16 | 0 | **50** | 0 | 9 |
| E | 0 | 2 | 2 | 0 | **38** | 0 |
| F | 0 | 1 | 0 | 0 | 0 | **37** |

Generative Markov model, 53.57%

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | **50** | 0 | 0 | 0 | 0 | 0 |
| B | 0 | **50** | 21 | 18 | 28 | 36 |
| C | 0 | 0 | **29** | 0 | 22 | 0 |
| D | 0 | 0 | 0 | **32** | 0 | 14 |
| E | 0 | 0 | 0 | 0 | **0** | 0 |
| F | 0 | 0 | 0 | 0 | 0 | **0** |

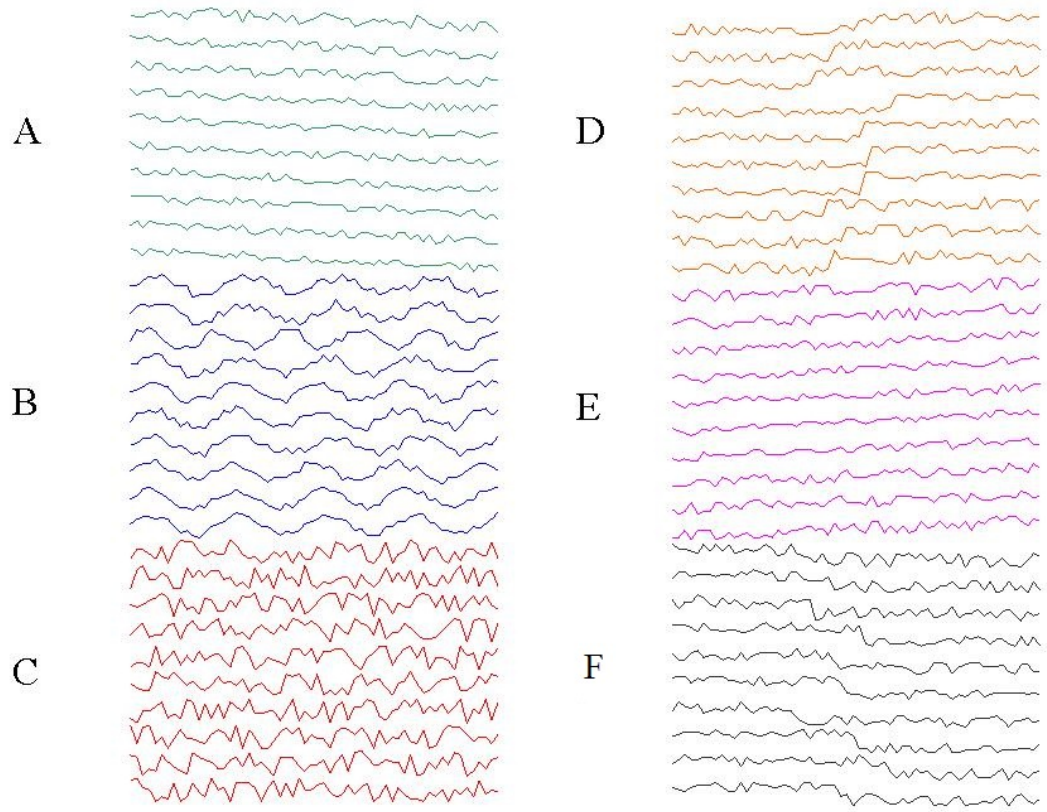



Figure 5.9. Sequence data used in [2]. Classes are, A: Downward trend, B: Cyclic, C:Normal, D:Upward Shift, E:Upward Trend, F:Downward Shift

Figure 5.10, we compare the classification accuracies of the discriminative and generative Markov models for varying size training sets. We divide the as 70 sequences from each class for training and 30 sequences from each class for testing. We keep the test set's size as 30 per class for varying training set sizes. We observe that it is only after we have a substantial amount of training data that discriminative model gain an advantage.

Intuitively, it makes sense since, if there are not enough variability in the training set, the discriminative tend to over-fit to this particular data, which is bad for generalization. However, generative models assume a certain data distribution, so their fitting ability is less than their discriminative counterparts as discussed in Appendix A.

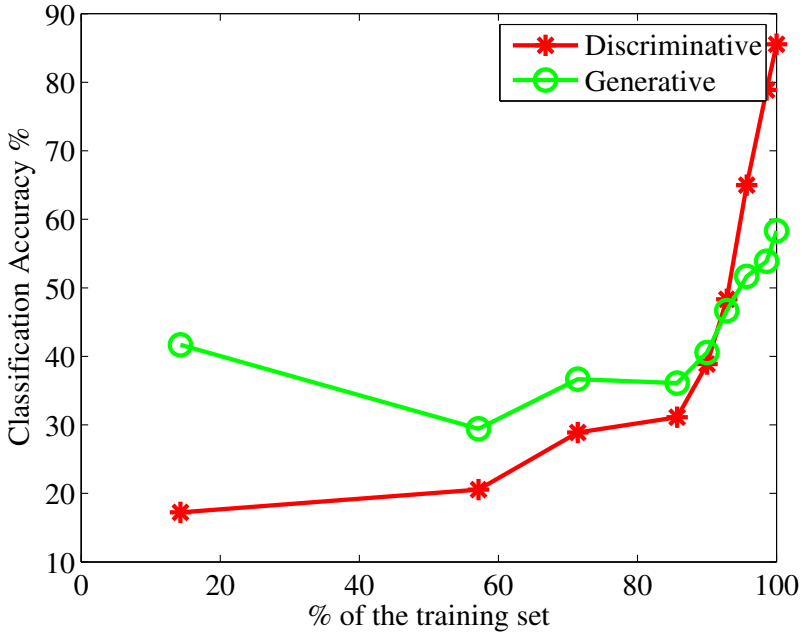



Figure 5.10. Number of data items used in training vs. classification accuracy of generative and discriminative models.

# 6. CONCLUSIONS

In this thesis, we have studied four generative and two discriminative models for sequence classification and clustering. The main offering of this thesis is the spectral learning algorithms for mixture of Markov models and mixture of HMMs. We give the conclusions and future prospects for each algorithm separately in upcoming sections.

## 6.1. Spectral learning algorithm for mixture of Markov models

In Section 4.3, we showed that a standard spectral learning algorithm as in [4,15] for a mixture of Markov models requires an observable moment up to order five. In Section 4.3.1, we have proposed a novel scheme for spectral learning of mixture of Markov models. We propose learning a mixture of Dirichlet distributions, instead of learning a mixture of Markov models. The experimental results on synthetic data demonstrates that, spectral learning of mixture Dirichlet distributions yields better clustering accuracies in all sample regimes (meaning the amount of available data items) than spectral learning of a mixture of Markov models. Our approach also performs better than learning a mixture Dirichlet distributions with expectation maximization, since EM may suffer from unlucky initializations, unlike our method which does not require initialization. We also observe that our approach gives meaningful clustering results on a complicated dataset of network traffic data. Moreover, in a supervised classification scenario, we see that doing clustering with our algorithm increases the classification accuracy more than the expectation maximization approaches. Furthermore, using our algorithm it is possible to find the number of clusters in a dataset as shown in Section 5.1.3. Overall, the approach is very easy to implement and computationally efficient, and competitive in terms of clustering accuracy.

As shown in detail in Section 4.3, for models with temporally connected observations such as mixture of Markov models, high order moments are necessary to express an eigen-decomposition form which reveals information about the parameters in terms of observable moments. As future work, we plan on investing the hierarchi-

cal Bayesian modeling viewpoint employed in Section 4.3.1, to derive similar spectral learning algorithms for more complicated time series models such an HMM with temporally connected observations. Also, as a short term object, we plan to derive a similar algorithm for the continuous analogue of Markov model, the autoregressive model.

Also, we did not provide a proof for the sample complexity bound for our algorithm as in [4]. As a theoretical part of the future work, we will provide a sample complexity bound for our mixture of Dirichlet distributions algorithm.

## 6.2. Incorporating spectral learning in learning mixtures of Hidden Markov models

Learning a mixture of Hidden Markov models requires the usage of a complicated EM algorithm, which is computationally expensive, as discussed in detail in Section 3.4. Replacing the parameter estimation step with a spectral learning algorithm results in a fast, simple and well-performing algorithm, given in Section 4.4. The experimental results show that, the resulting algorithm is faster and more accurate than applying a more conventional expectation maximization algorithm.

Deriving a standard collapsed Gibbs sampler for an infinite mixture of HMMs, which finds the number HMM clusters is not possible since it requires an intractable joint integral-summation over the model parameters and latent state sequence as shown in Section 4.4.1. However, it is possible to generalize the algorithm in Section 4.4, to handle infinite mixture of HMMs. We have validated our spectral mixture of HMMs learning algorithm via doing clustering in training phase of a supervised classification scenario. Our clustering algorithm results in an improved test accuracy.

Note that, this algorithm can not be considered as a complete spectral learning algorithm, since we have a k-means type algorithm in the outer loop and we only utilize the spectral learning as a subroutine. A fully spectral learning algorithm for learning a mixture of HMMs is hard to achieve since we have an additional permutation ambiguity caused by the cluster indicator variable $h_n$. For instance, the observation matrix can not be identified without an arbitrary interchanging of the columns $O$. Since there

are two layers of hidden variables, meaning a cluster indicator $h_n$ on top, and then a latent sequence $\mathbf{r}_n$, it is not possible to estimate the parameters with a standard eigen-decomposition because of the permutation ambiguity.

However, we can avoid the permutation ambiguity by learning a single HMM with a block-diagonal transition matrix. A block diagonal transition matrix would result in a reducible latent Markov chain with segregated regimes. Once sequence starts in a certain regime, it can not get out of it. So, learning a HMM with block-diagonal transition matrix for $N$ sequences is equivalent to learning a mixture of HMMs. It is not clear if it is possible to learn a HMM with a block diagonal constraint using the standard spectral learning algorithms in Section 4.2. However, it is possible to consider another method of moments based approach based on applying a matrix decomposition on the empirical moments matrix $P_2$ in Section 4.2. Inspired by the HMM learning algorithm based on non-negative matrix factorization in [34], we consider the following, optimization problem:

$$\begin{aligned}
&\min_{O,A,U} \| P_2 - OA \underbrace{diag(\pi) O^T}_{U} \|_F^2 \\
&s.t. \quad \sum_i O_{i,j} = 1, \sum_k A_{k,l} = 1, M. * A = 0, 1^T U 1 = 1, \\
&\quad O \geq 0, A \geq 0, U \geq 0
\end{aligned}$$

where, $M$ is a masking matrix that ensures the block diagonal structure of transition matrix $A$, and $.*$ is the element-wise product operator. We have considered solving this problem via coordinate-wise updates. In other words, we fix two variables from $O$, $A$, $U$, and then update one via solving a convex optimization problem. This approach offers a speed up compared to a traditional EM, since we do not have to do forward-backward message passing. However, we have observed that the success of the solution is highly dependent on the initialization. We are planning on investigating this problem further by applying more clever optimization techniques other than coordinate-wise updates.

### 6.3. Discussion on Spectral Learning and EM

Spectral learning algorithms are very exciting, first of all because of their simplicity. Implementing an EM algorithm for a HMM is not easy. The code is lengthy and there are a lot of potential sources for numeric instabilities. However, as the pseudocode in Figure 4.2.1 suggests, an alternative spectral learning algorithm is very easy to understand and implement.

The simplicity also offers speed. A spectral learning for learning an HMM simply amounts to accumulating some empirical moments, and then computing a low rank SVD and some eigen-decompositions. Also, the algorithm does not require any sort of initialization or iterations as EM does.

In spite of the elegance and simplicity of the algorithm, there are some drawbacks in the works [4, 15, 16] which may prevent the widespread usage of the algorithm. First of all, the constraint $K \leq L$ can be prohibitive in some applications. However, there is some very recent theoretical work [35] which shows that it is possible to derive tensor decompositions that does not require this constraint.

The other aspect which may prevent the spectral learning algorithms from getting in machine learning textbooks is the lack of generality. EM offers a very general framework for learning in almost all class of latent variable models. Although there are spectral learning papers for inference in general latent trees [36] and latent junction trees [17], these works do not consider learning (parameter estimation). As shown in the Section 4.3 and [16], for each different class of latent variables, we have to derive a spectral learning algorithm almost from scratch, meaning there is a lack of general framework which tells us how to relate the empirical moments to the model parameters.

It is also almost certain that spectral learning algorithms require more data than EM for stable parameter estimates. As shown in Figure 5.1, we see that the performance of the spectral learning algorithms is proportional to the amount of available data.

# APPENDIX A: Generative Models and Discriminative Models

Generative models and discriminative models are the two principal classes of models used in machine learning. The essence of the difference of these two model classes lies in their objective function. In a supervised learning scenario given the data $\mathbf{x}_{1:N}$ and labels $h_{1:N}$, to learn the model parameters $\theta$, we train the generative models via solving the following optimization problem:

$$\theta^* = \arg\max_{\theta} \prod_n p(\mathbf{x}_n|\ \theta, h_n) \tag{A.1}$$

This expression maximizes the likelihood $p(\mathbf{x}_{1:N}|\theta, h_{1:N})$ of the data points $\mathbf{x}_{1:N}$ and class labels $h_{1:N}$, given the model parameters $\theta$. Note that we separately train a model for each class. Thus, primarily we aim to maximize the ability of the model to fit the data, and separability of the classes is not directly addressed.

The discriminative models differ fundamentally from their generative counterparts by taking into account the class separability in training. In training discriminative models, we directly maximize the posterior of the class labels in order to learn the model parameters:

$$\theta^* = \arg\max_{\theta} \prod_n p(h_n|\mathbf{x}_n,\ \theta) = \arg\max_{\theta} \prod_n \frac{p(\mathbf{x}_n, h_n|\ \theta)}{\sum_{h_n} p(\mathbf{x}_n, h_n|\theta)} \tag{A.2}$$

The canonical examples for generative and discriminative models, respectively are the Naive Bayes model and logistic regression model. Suppose we want to do supervised classification on a dataset of $N$ two dimensional datapoints. Let the true class conditional densities be:

$$p(\mathbf{x}_n|\ \theta, h_n = 1) = \mathcal{N}(\mathbf{x}_n; \mu_1, \Sigma_1)$$
$$p(\mathbf{x}_n|\ \theta, h_n = 2) = \mathcal{N}(\mathbf{x}_n; \mu_2, \Sigma_2)$$

Let us first consider classification with Naive Bayes classifier, which is a generative model. Using a generative approach would require a modeling assumption for $p(\mathbf{x}_n|\theta, h_n)$. Given that these are the underlying true class conditional distributions, the best possible classifier would use these class conditionals for classification. However, in real life we do not know the true distribution of the data. Suppose for this toy problem we assume unit variance spherical Gaussians $\mathcal{N}(\mathbf{x}_n|\mu_k, \sigma^2 I)$ for class conditionals. In training, we fit the class conditional densities, which in this case amounts to simply computing the mean of the data items, for estimating $\mu_1$ and $\mu_2$. In test, when a new data item $\mathbf{x}_n$ comes in, we classify according to the ratio;

$$\log \frac{p(\mathbf{x}_{test}|\ \theta, h_n = 1)}{p(\mathbf{x}_{test}|\ \theta, h_n = 2)} = x^T(\mu_2 - \mu_1)/2\sigma^2 - \mu_1^T\mu_1/2\sigma^2 + \mu_2^T\mu_2/2\sigma^2 \tag{A.3}$$

If this ratio is greater than 0, the test data item $x_{test}$ is assigned to the first class, otherwise to the second class. Next, to derive a discriminative classifier, let us consider the posterior of the class label $h_n$ (assuming uniform prior $p(h_n)$):

$$\begin{aligned} p(h_n = 1|\mathbf{x}_n, \theta) =& \frac{p(\mathbf{x}_n|h_n = 1, \theta)}{p(\mathbf{x}_n|h_n = 1, \theta) + p(\mathbf{x}_n|h_n = 2, \theta)} \\ =& \frac{1}{1 + \exp(\log(\frac{p(\mathbf{x}_n|h_n=2,\theta)}{p(\mathbf{x}_n|h_n=1,\theta)}))} \\ =& \frac{1}{1 + \exp(x^T \underbrace{(\mu_2 - \mu_1)/2\sigma^2}_{\theta'} + \underbrace{\mu_2^T\mu_2/2\sigma^2 - \mu_1^T\mu_1/2\sigma^2}_{\theta'_0})} \\ =& \frac{1}{1 + \exp(x^T\theta' + \theta'_0))} \end{aligned} \tag{A.4}$$

where, the expression in Equation A.4 is known as the logistic function $\sigma(x^T\theta' + \theta'_0))$ (and hence the name logistic regression). Notice that, altough we have assumed that the class conditional densities $p(\mathbf{x}_n|h_n = 1, \theta)$ are isotropic Gaussians with equal variances, at the end, we have a generic linear discriminant as the argument of the logistic function. We see that, in logistic regression, the choice of the model is equivalent to choosing the type of discriminant function. For instance, having arbitrary covariance matrix Gaussians as class conditionals would result in a quadratic discriminant function. To, better appreciate the difference between generative and discriminative models, let us

consider Figure A.1.

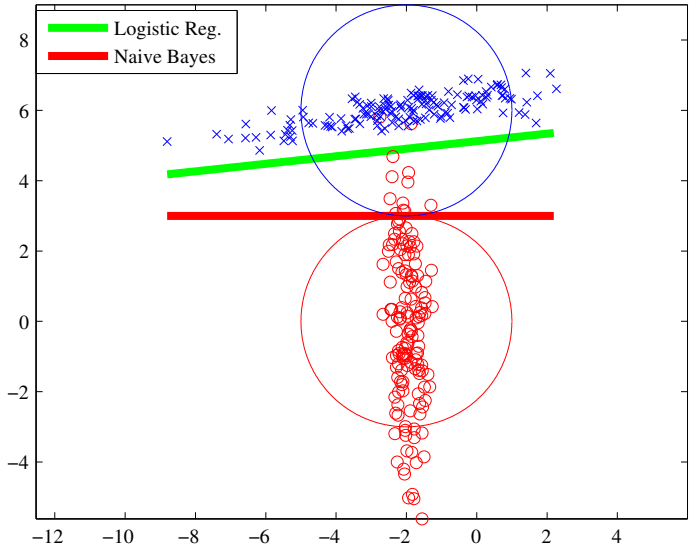



Figure A.1. Logistic Regression vs. Naive Bayes Classifiers. The decision boundary are respectively shown with green and red lines.

When we learn the parameters $\theta'$ of the discriminant function, the main objective of the optimization problem is to maximize the class separation as seen from Equation A.2, and as illustrated in Figure A.1: Logistic regression is able to classify more data points correctly than naive Bayes classifier. Naive Bayes classifier fails to find a decision boundary as good as its discriminative counterpart logistic regression which takes into account the tail of the red data points distribution, because of the model mis-specification caused by the isotropic variance assumption. Although it is simple to fit two dimensional arbitrary covariance matrix Gaussians, in more sophisticated real life models, there is almost always a model mis-specification associated with using generative classification.

When labelled training data is plentiful, generative models give great generalization performances [7, 33] (meaning they perform good on test data). However, in practice we may encounter unlabeled data. Discriminative models can not handle unlabeled data, so we have to use generative models, in this type of datasets.

Even though the Naive Bayes and Logistic Regression Models we have considered in this section are not sequential models, we have used them for their simplicity to demonstrate the concepts of the generative and discriminative models. According to the terminology in [33], logistic regression and naive Bayes models form a generative-discriminative pair. Following the generative-discriminative pair convention, in the thesis, we introduce Markov model-discriminative Markov model and HMM-discriminative HMM (which is in fact a linear chain hidden conditional random field (HCRF) [21]).

# APPENDIX B: Gibbs Sampling for an Infinite mixture of HMMs using auxiliary parameter method

As discussed in Section 4.4.1, to derive a sampler for an infinite mixture model, we have to take an integral over the model parameters $\theta$. In [37], there is an alternative algorithm called Gibbs sampling with auxiliary parameters. We consider a mixture model with $H$ clusters, although there are only $K$ occupied clusters. According to this method, a sampling iteration is as follows (We continue the notation convention in Section 4.4.1):

- For occupied $K$ clusters;
  We sample the parameters of cluster $k$, $\theta_k \sim p(\theta_k|\ \{\mathbf{x}_l : l, r_l = k\})$, from the posterior of the parameters of cluster $k$.
- For unoccupied $H - K$ clusters;
  We sample the parameters of cluster $k$, $\theta \sim p(\theta_k)$, from the prior of the model parameters
- Then, we sample $h_{1:N}$, according to,
  (i) For occupied $K$ clusters;

$$p(h_n = k|\ h_{1:N}^{-n}, \mathbf{x}_{1:N}, \theta_{1:K}) \propto \frac{N_k^{-n}}{N - 1 + \alpha} p(\mathbf{x}_n|\ \theta_k) \tag{B.1}$$

  (ii) For unoccupied $H - K$ clusters;

$$p(h_n = k|\ h_{1:N}^{-n}, \mathbf{x}_{1:N}, \theta_{1:K}) \propto \frac{\alpha/(H - K)}{N - 1 + \alpha} p(\mathbf{x}_n|\ \theta_k) \tag{B.2}$$

  (iii) If $k > K$, we set $K = K + 1$.

It is claimed in the paper that as $H \to \infty$, this algorithm converges to a collapsed Gibbs sampler for an infinite mixture model. So, $H$ should be chosen large, which is the reason why this algorithm is considered slow. In order to apply this algorithm for

a HMM, we have to first know how sample from the HMM parameters $\theta = (O, A, \mathbf{r}_n)$. In the next section, we derive a Gibbs sampler for a HMM.

## B.1. Gibbs Sampling in HMM

The full Bayesian model is as follows:

$$
\begin{aligned}
r_t | r_{t-1} &\sim \text{Discrete}(A_{r_{t-1}}) && \text{(B.3)} \\
A_m &\sim \text{Dirichlet}(\alpha, \ldots, \alpha) && \text{(B.4)} \\
O_m &\sim H && \text{(B.5)} \\
x_t | r_t &\sim p(x_t | O_{r_t}) && \text{(B.6)}
\end{aligned}
$$

The variables are defined as follows:

- $r_t \in \{1, \ldots, M\}$ is the state indicator variable at discrete time $t$.
- $A_m = A(:, m)$ is the state transition density from state $m$. (Alternatively, it can be considered as the $m$'th column of the state transition matrix)
- $p(x_t|O)$ is the observation density
- $O_m = O(:, m)$ is the parameter set used in the observation density at state $m$.
- $H$ is the prior density on the observation matrix $O$.

Note that in this model definition we did not include the first state distribution for the sake of simplicity. Next, we have to derive the full conditionals. In order to do this, we write the full-joint distribution and consider functional dependencies on each variable to be sampled.

$$
\begin{aligned}
p(x_{1:T}, r_{1:T}, O, A) &= \prod_{t=1}^{T} p(x_t | r_t, O) \prod_{t=1}^{T} p(r_t | r_{t-1}, A) \prod_{m=1}^{M} p(A_m) \prod_{m=1}^{M} p(O_m) \\
&\propto \prod_{t=1}^{T} \prod_{j=1}^{M} p(x_t | O_j)^{[r_t = j]} \prod_{t=1}^{T} \prod_{m=1}^{M} \prod_{j=1}^{M} A_{mj}^{[r_{t-1}=m, r_t=j]} \prod_{m=1}^{M} \prod_{j=1}^{M} A_{mj}^{\alpha-1} \prod_{m=1}^{M} p(O_m) \qquad \text{(B.7)}
\end{aligned}
$$

$[a = b]$ is the indicator notation and returns 1 if $a = b$, 0 if $a \neq b$. Then the full conditionals are derived as follows:

$$\begin{aligned}
p(A_m|others) &\propto \prod_{j=1}^{M} A_{m,j}^{n_{mj}} \prod_{j=1}^{M} A_{m,j}^{\alpha-1} \\
&\propto \prod_{j=1}^{M} A_{m,j}^{n_{mj}+\alpha-1} \propto \text{Dirichlet}(n_{m,1}+\alpha, \ldots, n_{m,M}+\alpha) \qquad \text{(B.8)} \\
p(O_j|others) &\propto \prod_{t=1}^{T} p(x_t|O_j)^{[r_t=j]} p(O_j) = \prod_{t:r_t=j} p(x_t|O_j)p(O_j) \qquad \text{(B.9)} \\
p(r_t|others) &\propto \prod_{j=1}^{M} p(x_t|O_j)^{[r_t=j]} \prod_{j=1}^{M} A_{r_{t-1},j}^{[r_t=j]} \prod_{j=1}^{M} A_{j,r_{t+1}}^{[r_t=j]} \qquad \text{(B.10)} \\
&\propto \prod_{j=1}^{M} (p(x_t|O_j)A_{r_{t-1},j}A_{j,r_{t+1}})^{[r_t=j]}, \text{for} \quad 1 \leq t \leq T-1 \qquad \text{(B.11)} \\
&\propto \prod_{j=1}^{M} (p(x_t|O_j)A_{r_{t-1},j})^{[r_t=j]}, \text{for} \quad t = T \qquad \text{(B.12)}
\end{aligned}$$

Here $n_{m,j}$ denotes the number transitions from state $m$ to state $j$. Note that this is the most straightforward Gibbs sampling scheme. In the sequel, we will sample the indicator sequence $r_{1:T}$ in blocks. That is, we will sample trajectories. The reason behind this is as follows; the basic Gibbs sampler which samples from each of the variables independently tends to get stuck in a particular region. Consider the following scenario: Suppose $r_{t-1} = 1$, $r_{t+1} = 1$, $\pi_{1,j} = 0.99$, $\pi_{k,1} = 0.99$, then it is very unlikely that $r_t \neq 1$, even though it is still possible. Therefore, if sample from the trajectories, instead of the individual variables, we may better explore the search space. Generally, the Gibbs sampling performs badly if there is a correlation between the variables. The way to sample from the trajectories $r_{1:T}$ is as follows. Consider the posterior distribution over sequences:

$$p(r_{1:T}|x_{1:T}) = p(r_T|x_{1:T})p(r_{T-1}|r_T, x_{1:T-1}) \ldots p(r_{t-1}|r_t, x_{1:t-1}) \ldots p(r_1|r_2, x_1) \qquad \text{(B.13)}$$

Note that,

$$\begin{aligned}
p(r_{t-1}|r_t, x_{1:t-1}) &\propto p(r_{t-1}, x_{1:t-1})p(r_t|r_{t-1}) \\
&\propto \alpha(r_{t-1})A_{r_{t-1},r_t} \qquad \text{(B.14)}
\end{aligned}$$

where,

$$p(r_T, x_{1:T}) \propto \alpha(r_T) \tag{B.15}$$

$\alpha(r_t)$ is the forward message defined in Section 3.2.1, in Equation 3.15. We first compute the all of the forward messages. And then, we first sample last indicator variable $r_T$ from $p(r_T, x_{1:T})$ . Then conditioned on it, we sample $r_{T-1}$ from $p(r_{T-1}|r_T, x_{1:T-1})$. We go on like this until the first indicator variable.

# APPENDIX C: EM algorithm for learning mixture of Dirichlet distributions

As discussed in detail in Section 3.1, in order to derive an EM algorithm for a latent variable model, we first write the EM lower bound (We use the notation in Section 4.3.1):

$$\begin{aligned}
\mathcal{Q}(\theta,\theta^*) &=^{+} \mathbb{E}_{p(r_{1:N}|s_{1:N},\alpha^*)}[\log p(s_{1:N},r_{1:N}|\alpha)] \\
&= \mathbb{E}_{p(r_{1:N}|x_{1:N},\alpha^*)}[\log \prod_{n=1}^{N} p(s_n|r_n,\alpha)p(r_n)] \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}([r_n=k]) \log \text{Dirichlet}(\alpha(:,k)) + \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}([r_n=k]) \log p(h_n=k) \quad \text{(C.1)}
\end{aligned}$$

Omitting the second term (mixing proportions $p(h_n)$) for the sake of simplicity, we are left with the first term to maximize. We compute the gradient of it, with respect to $\alpha(l,k)$:

$$\frac{\partial \mathcal{Q}(\theta,\theta^*)}{\partial \alpha(l,k)} = \sum_{n=1}^{N} \mathbb{E}([r_n=k]) \Big( \psi(\sum_{l'=1}^{L} \alpha(l',k)) - \psi(\alpha(l,k)) + \log s_{l,n} \Big) \quad \text{(C.2)}$$

where, $\psi(.)$ is the Digamma function. Note that there is no closed form update equation for $\alpha(:,k)$. We have to find the roots of this equations via some numerical method. One additional thing to note is that, in order to ensure that $\alpha > 0$, we can use an interior point method. To do this, we introduce an additional term to be maximized so that we maximize the function $\mathcal{Q}(\theta,\theta^*) + \sum_l \log(\alpha(l,k))$: If the alpha gets closer to 0, the log(.) function goes to infinity. So, we penalize the values that comes near zero. Overall, we have the following gradient descent update:

$$\alpha(l,k)^{new} = \alpha(l,k) + \zeta \Big( \frac{\partial \mathcal{Q}(\theta,\theta^*)}{\partial \alpha(l,k)} + \frac{1}{\alpha(l,k)} \Big) \quad \text{(C.3)}$$

where, $\zeta$ is the learning rate.

# REFERENCES


1. Ning, H., Y. Hu and T. S. Huang, "Searching Human Behaviours Using Spatial-Temporal Words", *International Conference on Image Processing*, 2004.

2. Pham, D. T. and A. B. Chain, "Control Chart Pattern Recognition Using a New Type of Self Organizing Neural Network", *In Proceedings of The Institution of Mechanical Engineers Part I-Journal of Systems and Control Engineering*, Vol. 212, No. 2, pp. 115–127, 1998.

3. Subakan, Y. C., B. Kurt, A. T. Cemgil and B. Sankur, "Spectral Learning for Mixture of Markov Models", *Neural Information Processing Systems*, 2013, (Submitted).

4. Anandkumar, A., D. Hsu and S. Kakade, "A Method of Moments for Mixture Models and Hidden Markov Models", *Conference on Learning Theory*, 2012.

5. Subakan, Y. C., O. Celiktutan, A. T. Cemgil and B. Sankur, "Spectral Learning of Mixtures of Hidden Markov Models", *21'st IEEE Signal Processing Applications Conference (SIU)*, 2013.

6. Subakan, Y. C., O. Celiktutan, A. T. Cemgil and B. Sankur, "Spectral Learning of Infinite Mixtures of Hidden Markov Models for Human Action Recognition", *Neural Information Processing Systems, Spectral Learning for Latent Variable Models Workshop*, 2012.

7. Bishop, C. M. and J. Lasserre, "Generative or Discriminative? Getting the Best of Both Worlds", *Bayesian Statistics*, Vol. 8, pp. 3–24, 2007.

8. Xing, Z., J. Pei and E. Keogh, "A Brief Survey on Sequence Classification", *SIGKDD Explorations Newsletter*, Vol. 12, No. 1, pp. 40–48, 2010.

9. Leslie, C. S., E. Eskin and W. S. Noble, "The Spectrum Kernel: A String Kernel for SVM Protein Classification", *Pacific Symposium on Biocomputing*, 2002.

10. Chuzhanova, N. A., A. J. Jones and S. Margetts, "Feature Selection for Genetic Sequence Classification", *Bioinformatics*, Vol. 12, No. 1, pp. 139–143, 1998.

11. Lesh, N., M. J. Zaki and M. Ogihara, "Mining Features for Sequence Classification", *In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pp. 342–346, ACM, New York, NY, USA, 1999.

12. Wei, L. and E. Keogh, "Semi-Supervised Time Series Classification", *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 748–753, 2006.

13. Keogh, E. J. and M. J. Pazzani., "Scaling Up Dynamic Time Warping for Data Mining Applications", *In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 285–289, 2000.

14. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

15. Hsu, D., S. M. Kakade and T. Zhang, "A Spectral Algorithm for Learning Hidden Markov Models", *Journal of Computer and System Sciences*, Vol. 78, No. 5, pp. 1460–1480, 2012.

16. Anandkumar, A., R. Ge, D. Hsu, S. Kakade and M. Telgarsky, "Tensor Decompositions for Learning Latent Variable Models", *arXiv:1210.7559v2*, 2012, `http://arxiv.org/abs/1210.7559v2`, accessed at June 2013.

17. Parikh, A. P., L. Song, M. Ishteva, G. Teodoru and E. P. Xing, "A Spectral Algorithm for Latent Junction Trees", *The 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

18. Rasmussen, C. E., "The Infinite Gaussian Mixture Model", *Neural Information*

*Processing Systems*, pp. 554–560, MIT Press, 2000.

19. Kulis, B. and M. I. Jordan, "Revisiting K-means: New Algorithms via Bayesian Nonparametrics", *arXiv:1210.7559v2*, 2011, `http://arxiv.org/abs/1111.0352`, accessed at June 2013.

20. Sutton, C. and A. McCallum, "An Introduction to Conditional Random Fields", *arXiv:1011.4088*, 2010, `http://arxiv.org/abs/1011.4088`, accessed at June 2013.

21. Quattoni, A., S. Wang, L. Morency, M. Collins and T. Darrell, "Hidden Conditional Random Fields", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 29, No. 10, pp. 1848–1852, 2007.

22. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society, Series B*, Vol. 39, No. 1, pp. 1–38, 1977.

23. Pearson, K., "Method of Moments and Method of Maximum Likelihood", *Biometrika*, Vol. 28, No. 1/2, pp. 34–59, 1936.

24. Anandkumar, A., D. P. Foster, D. Hsu, S. Kakade and Y. Liu, "Two SVDs Suffice: Spectral Decompositions for Probabilistic Topic Modeling and Latent Dirichlet Allocation", *Neural Information Processing Systems*, 2012.

25. Anandkumar, A., R. Ge, D. Hsu and S. M. Kakade, "A Tensor Spectral Approach to Learning Mixed Membership Community Models", *Conference on Learning Theory*, 2013.

26. Nial, F. and W. Jason, "Estimating the Evidence - A Review", *Statistica Neerlandica*, Vol. 66, No. 3, pp. 288–308, 2012.

27. Ipoque, "Ipoque PACE: Protocol and Application Classification Engine", `http://www.ipoque.com/en/products/protocol-and-application-`

`classification-engine`, accessed at June 2013.

28. Nguyen, T. and G. Armitage, "A Survey of Techniques for Internet Traffic Classification Using Machine Learning", *IEEE Communications Surveys and Tutorials*, Vol. 10, No. 4, pp. 56–76, 2008.

29. Carnegie Mellon University, "CMU Motion Capture Database", `http://mocap.cs.cmu.edu/`, accessed at June 2013.

30. Jebara, T., Y. Song and K. Thadani, "Spectral Clustering and Embedding with Hidden Markov Models", *In Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pp. 164–175, Springer-Verlag, Berlin, Heidelberg, 2007.

31. Müller, M., T. Röder, M. Clausen, B. Eberhardt, B. Krüger and A. Weber, *Documentation Mocap Database HDM05*, Tech. Rep. CG-2007-2, Universität Bonn, June 2007.

32. Bregonzio, M., S. Gong and T. Xiang, "Recognising Action as Clouds of Space-Time Interest Points", *Computer Vision and Pattern Recognition*, pp. 1948–1955, IEEE, 2009.

33. Ng, A. Y. and M. I. Jordan, "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes", *Neural Information Processing Systems*, 2001.

34. Lakshminarayanan, B. and R. Raich, "Non-Negative Matrix Factorization for Parameter Estimation in Hidden Markov Models", *Machine Learning for Signal Processing (MLSP), IEEE International Workshop on*, pp. 89–94, 2010.

35. Bhaskara, A., M. Charikar and A. Vijayaraghavan, "Uniqueness of Tensor Decompositions with Applications to Polynomial Identifiability", *arXiv: 1304.8087*, 2013, `http://arxiv.org/abs/1304.8087`, accessed at June 2013.

36. Parikh, A. P., L. Song and E. P. Xing, "A Spectral Algorithm for Latent Tree

Graphical Models", *In Proceedings of the 28th International Conference on Machine Learning*, 2011.

37. Neal, R. M., "Markov Chain Sampling Methods for Dirichlet Process Mixture Models", *Journal of Computational and Graphical Statistics*, Vol. 9, No. 2, pp. 249–265, 2000.