Online Bayesian Learning in Probabilistic Graphical Models using Moment Matching with Applications

by

Farheen Omar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

Abstract

Probabilistic Graphical Models are often used to efficiently encode uncertainty in real world problems as probability distributions. Bayesian learning allows us to compute a posterior distribution over the parameters of these distributions based on observed data. One of the main challenges in Bayesian learning is that the posterior distribution can become exponentially complex as new data becomes available. Secondly, many algorithms require all the data to be present in memory before the parameters can be learned and may require retraining when new data becomes available. This is problematic for big data and expensive for streaming applications where new data arrives constantly.

In this work I have proposed an online moment matching algorithm for Bayesian learning called Bayesian Moment Matching (BMM). This algorithm is based on Assumed Density Filtering (ADF) and allows us to update the posterior in a constant amount of time as new data arrives. In BMM, after new data is received, the exact posterior is projected onto a family of distributions indexed by a set of parameters. This projection is accomplished by matching the moments of this approximate posterior with those of the exact one. This allows us to update the posterior at each step in constant time. The effectiveness of this technique has been demonstrated on two real world problems.

Topic Modelling: Latent Dirichlet Allocation (LDA) is a statistical topic model that examines a set of documents and based on the statistics of the words in each document, discovers what is the distribution over topics for each document.

Activity Recognition: Tung et al [29] have developed an instrumented rolling walker with sensors and cameras to autonomously monitor the user outside the laboratory setting. I have developed automated techniques to identify the activities performed by users with respect to the walker (e.g.,walking, standing, turning) using a Bayesian network called Hidden Markov Model. This problem is significant for applied health scientists who are studying the effectiveness of walkers to prevent falls.

My main contributions in this work are:

- In this work, I have given a novel interpretation of moment matching by showing that there exists a set of initial distributions (different from the prior) for which exact Bayesian learning yields the same first and second order moments in the posterior as moment matching. Hence the Bayesian Moment matching algorithm is exact with respect to an implicit posterior.

- Label switching is a problem which arises in unsupervised learning because labels can be assigned to hidden variables in a Hidden Markov Model in all possible permutations without changing the model. I also show that even though the exact posterior has $n!$ components each corresponding to a permutation of the hidden states, moment matching for a slightly different distribution can allow us to compute the moments without enumerating all the permutations.

- In traditional ADF, the approximate posterior at every time step is constructed by minimizing KL divergence between the approximate and exact posterior. In case the prior is from the exponential family, this boils down to matching the "natural" moments. This can lead to a time complexity which is the order of the number of variables in the problem at every time step. This can become problematic particularly in LDA, where the number of variables is of the order of the dictionary size which can be very large. I have derived an algorithm for moment matching called Linear Moment Matching which updates all the moments in $O(n)$ where n is the number of hidden states.

- I have derived a Bayesian Moment Matching algorithm (BMM) for LDA and compared the performance of BMM against existing techniques for topic modelling using multiple real world data sets.

- I have developed a model for activity recognition using Hidden Markov Models (HMMs). I also analyse existing parameter learning techniques for HMMs in terms of accuracy. The accuracy of the generative HMM model is also compared to that of a discriminative CRF model.

- I have also derived a Bayesian Moment Matching algorithm for Activity Recognition. The effectiveness of this algorithm on learning model parameters is analysed using two experiments conducted with real patients and a control group of walker users.

Acknowledgements

Pascal Poupart was my supervisor during my research on this thesis. He guided and challenged me throughout my research by forcing me to approach my research with crystal clarity and focus. He was thoroughly engaged with my work and was always available whenever I needed assistance and feedback. He always worked with me to untangle each roadblock that I encountered. Even after I had a baby and was working remotely towards the end, his support never flagged. I am truly indebted to him for his mentoring, supervision and kind understanding and patience.

I am also grateful to my thesis committee: Dan Lizzotte and Jesse Hoey for their critical analysis of my work, John Zelek who gave me an engineering perspective on how this work can be utilized and Richard Zemel my external examiner who gave me wonderful insights on how to polish this research further and suggesting future directions. I would also like to thank Han Zhao, Mathieu Sinn and Jakub Truszkowski for contributing to this research.

Margaret Towell in the Computer Science Grad office has always been an invaluable resource who was untiring in her readiness to assist students whenever any help was needed in the catacombs of paperwork.

My husband Omar has been instrumental in helping me become who I am today. He has given me courage and support and also served as a good sounding board for my ideas. He also gave me great critical insights on my work. Above all he convinced me that I could accomplish all I dream of even when I had grown up with a mindset that discouraged achievement for women.

I would also like to give a shout out to my mother-in-law Mrs Ghazala Yawar who provided me with a support system without which I would not have been able to do justice to my young child and my research. I want to thank my mother Neelofur Shahid who is the kindest being that I know. Her support throughout my early education and university and her kindness as a parent is the reason why I am here today. I want to thank my father who despite being the biggest source of challenges in my life, inculcated a thirst of knowledge, reading and planted the seeds of inquisitiveness in me that have led me to pursue research professionally. I would also like to thank my brother Salman Shahid for being a partner in crime when I was younger and a source of support even now. A special thanks to my brother-in-law Awais Zia Khan who was always ready to babysit and offer other help when needed. And lastly a big thanks to my little boy Rohail who is the most wonderful of children. If he were not the sweet person that he is I would not have been able to complete what I had started.

Dedication

To my rock and love Omar and to the source of light in my life Rohail

Table of Contents

List of Tables **xi**

List of Figures **xiv**

1 Introduction **1**

1.1 Moment Matching . . . 2

1.2 Topic Modeling . . . 3

1.2.1 Contributions . . . 4

1.3 Activity Recognition with an Instrumented Walker . . . 5

1.3.1 Contributions . . . 6

1.4 Organization . . . 8

2 Background **10**

2.1 Probability Theory and Bayes Rule . . . 10

2.1.1 Bayes Theorem . . . 11

2.2 Probabilistic Graphical Models . . . 11

2.2.1 Bayesian Networks . . . 12

2.2.2 Inference in Bayesian Networks . . . 12

2.2.3 Parameter Learning in Bayesian Networks . . . 12

2.3 Bayesian Learning . . . 14

2.3.1 Mean-Field Variational Inference (Variational Bayes) . . . 14

2.3.2 Gibbs Sampling 15
2.4 Latent Dirichlet Allocation 16
2.4.1 Learning and Inference 17
2.5 Dynamic Bayesian Networks 18
2.5.1 Hidden Markov Models 18
2.5.2 Inference 20
2.5.3 Bayesian Filtering 21
2.5.4 Parameter Learning 21
2.6 Conditional Random Fields 23
2.6.1 Linear Chain CRF 24

3 Related Work 26

3.1 Parameter Learning for Latent Dirichlet Allocation 26
3.1.1 Online Parameter Learning for LDA 27
3.2 Activity Recognition for Instrumented Walker 28
3.2.1 Parameter Learning in HMM 28
3.3 Assumed Density Filtering 30

4 Online Bayesian Learning Using Moment Matching 31

4.1 Moments 31
4.1.1 Sufficient Set of Moments 32
4.2 Moment Matching 32
4.2.1 Moment Matching for the Dirichlet Distribution 34
4.3 LDA with known Observation Distribution 36

5 Topic Modeling 39

5.1 Latent Dirichlet Allocation 39
5.2 Known Topic-Word Distribution Case 41

5.2.1 Analysis 42
5.3 Learning the Word-Topic Distribution 46
5.3.1 Label Switching and Unidentifiability 46
5.3.2 Sufficient Moments 47
5.3.3 Moment Matching 49
5.3.4 Linear Moment Matching 52
5.3.5 Discussion 54
5.4 Results 55
5.4.1 UCI Bag of Words Document Corpus 55
5.4.2 Wikipedia Corpus 55
5.4.3 Twitter Data 56
5.4.4 Synthetic Data 56
5.4.5 Experiments 57

6 Activity Recognition with Instrumented Walker **65**
6.1 The Walker and Experimental Setup 65
6.1.1 Experiment 1 66
6.1.2 Experiment 2 67
6.1.3 Sensor Data 68
6.2 Activity Recognition Model 69
6.2.1 Hidden Markov Model 71
6.3 Prediction 72
6.4 Maximum Likelihood Parameter Learning 74
6.4.1 Supervised Learning 74
6.4.2 Unsupervised Maximum Likelihood Learning 75
6.4.3 Bayesian Learning for HMMs 76
6.4.4 Gibbs Sampling 76
6.5 Conditional Random Field 78

6.6 Results and Discussion 80
6.6.1 Discussion 82
6.6.2 Experiment 1 vs. Experiment 2 88
6.6.3 CRF vs. HMM 88
6.6.4 Maximum Likelihood vs. Bayesian Learning 88

7 Moment Matching For Activity Recognition 90
7.1 Bayesian Learning for HMMs 90
7.2 Moment Matching for Hidden Markov Models 92
7.2.1 The Known Observation Model 93
7.2.2 Learning the Observation Model 94
7.2.3 Efficient Moment Matching 98
7.2.4 Multiple Sensors 102
7.3 Discussion 103
7.4 Experiments and Results 104
7.4.1 Comparison with Online EM 106

8 Conclusions 108

A Derivation of Expectation Maximization for Activity Recognition 112

APPENDICES 112
A.1 Maximum Likelihood Supervised Learning 112
A.2 Maximum Likelihood Unsupervised Learning 114
A.2.1 Avoiding Underflows 116

References 118

List of Tables

5.1 Summary Statistics of the data sets 57

5.2 Time taken in seconds by each algorithm to go over complete data. 63

6.1 Activities performed in Experiment 1 67

6.2 Additional activities performed during Experiment 2 68

6.3 Highest value for each sensor s with respect to activity y, $\arg\max_e \Pr\left(e^s|y\right)$. The second row for each activity is the probability of this highest value $\max_e \Pr\left(e^s|y\right)$. This table is based on the data from the second experiment 70

6.4 Empirical Transition Distribution for Experiment 2 71

6.5 Confusion matrix for HMM model Experiment 1 activities using load sensor values. Observation model learned from data. Activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, Load cell values. Overall accuracy is 91.1%. 82

6.6 Confusion matrix for HMM model for Experiment 1 using center of pressure. Observation model learned from data. Activity persistence parameter: $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 88%. 83

6.7 Confusion matrix for CRF model for Experiment 1 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and Total Weight. Overall accuracy is 93.8%. 83

6.8 Confusion matrix for HMM model for Experiment 2 data using load sensor values. Observation model learned from data. activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, load cell values. Overall accuracy is 79.1%. 84

6.9 Confusion matrix for HMM model for for Experiment 2 data using center of pressure. Observation model learned from data. activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 77.2%. 84

6.10 Confusion matrix for CRF model for Experiment 2 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 80.8%. 85

6.11 Confusion matrix for HMM model results for Experiment 2 data using normalized load sensor values. Observation model and transition model learned from data. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measuremens, Speed, Normalized load cell values. Overall accuracy is 80.8%. 85

6.12 Experiment 1 percentage accuracy for each activity. COP implies that center of pressure feature is used instead of load cell values. Prediction is done using the Forward-Backward Algorithm 86

6.13 Experiment 2 percentage accuracy for each activity. NL means the normalized load values are used. COP implies that center of pessure feature is used instead of normalized load values. 87

7.1 Moment Matching results for Experiment 1 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and Total Weight. Overall accuracy is 76.2%. 104

7.2 Experiment 1 percentage accuracy for each activity for unsupervised learning techniques. Prediction is done using the Forward-Backward Algorithm . . 105

7.3 Experiment 2 percentage accuracy for each activity. NL means the normalized load values are used. COP implies that center of pessure feature is used instead of normalized load values. 105

7.4 Average training time for unsupervised algorithms 106

List of Figures

1.1 The instrumented walker developed by [73] et. al. The walker has a 3-D accelerometer, 4 load cells, 1 wheel encoder and 2 cameras 7

2.1 Latent Dirichlet Allocation Topic Model 16

2.2 A Bayesian Hidden Markov Model 20

4.1 Expectation of θ with respect to the approximate posterior vs. the number of observations. $Q(\theta)$ is the approximate posterior learnt by matching $M_{\log(\theta)}$ where as $Q'(\theta)$ is the approximate posterior learnt by matching $M_{(\theta)}$. The true value of θ used to generate the observations is 0.7. 38

5.1 Latent Dirichlet Allocation Topic Model 40

5.2 Comparison of different learning algorithms for topic modeling for the NIPS dataset. The number of topics $T = 25$. The second figure is the perplexity for vblp, gs and bplp zoomed 58

5.3 Comparison of different learning algorithms for topic modeling for the Enron dataset. The number of topics $T = 100$. The top right figure is the perplexity of vblp, gs and bplp and the bottom right is the preplexity of AWR-KL, AWR-L2 and Spectral LDA 59

5.4 Comparison of different learning algorithms for topic modeling for the Twitter dataset. The number of topics $T = 9$. The top right figure is the perplexity of vblp, gs and bplp and the bottom right is the preplexity of AWR-KL, AWR-L2 and Spectral LDA 60

5.5 Comparison of different learning algorithms for topic modeling for the NY-Times dataset. The number of topics $T = 100$. 61

5.6 Comparison of different learning algorithms for topic modeling for the Wikipedia dataset. The number of topics $T = 100$. 62
5.7 Comparison of different learning algorithms for topic modeling for the PubMed dataset. The number of topics $T = 100$. 63
5.8 EP vs MM on synthetic data. $T = 5$ 64

6.1 Course for data collection experiment done with the walker. Healthy young subjects were asked to follow this course 66
6.2 Course used in data collection experiment for older subjects who are walker users 67
6.3 A Hidden Markov Model For Activity Recognition 72
6.4 A Hidden Markov Model For Activity Recognition with Bayesian Priors . . 73
6.5 Accuracy, Precision and Recall for various algorithms in each experiment. . 86

7.1 A Hidden Markov Model For Activity Recognition with Bayesian Priors . . 91
7.2 Accuracy, Precision and Recall for various algorithms in each experiment. . 106
7.3 Accuracy, Precision and Recall for Online EM and Moment Matching on synthetic data. 107

Chapter 1

Introduction

Probability theory allows us to capture inherent uncertainty that arises in real world processes. This uncertainty may stem from the fact that some parts of the process are not observable directly or because we are trying to predict the values of future events based on observations in the past. The probability of an event can be informally defined as the degree of belief that this event will happen. A probability distribution is a function that assigns a real number $Pr\left(X = x\right)$ to a random variable X which represents the degree of belief that X will take value x. In some cases, the distribution has the form $\Pr\left(X = x|\theta\right) = f\left(x, \theta\right)$. θ is called the parameter of the distribution. Probabilistic Graphical Models allow us to encode probability distributions over high dimensional event spaces efficiently. One of the key tasks in constructing these models is to learn the value of parameters of each distribution that best describe the observed data. Bayesian learning provides a paradigm which allows us to compute a posterior belief about model parameters, based on a prior belief about their values and observed data. This posterior belief is usually encoded in the form of a posterior distribution over the model parameters.

One of the main challenges in Bayesian learning is that the posterior distribution can become very complex as new information becomes available. In some cases, the posterior can be an exponentially large mixture of distributions. Hence representing the posterior and calculating its statistics can be computationally expensive. Typically algorithms aim to construct an approximate posterior either by minimizing some distance metric between the exact and approximate posterior, or by drawing samples from the exact posterior. Most of these algorithms require the complete data to be present in the memory at the time of learning and usually require the model to be retrained if new data arrives. More recently, online versions of some of these algorithms have been developed that are able to incorporate new information without retraining the complete model. These online techniques use a sample set of the data known as a mini-batch to update the parameters such that some distance metric is minimized.

Maximum Likelihood Learning is an alternative to Bayesian Learning where instead of computing a posterior distribution over the parameters, we compute point estimates for the values of parameters that maximize the likelihood of the observed data. This often involves solving a non convex optimization problem and the algorithms are prone to get stuck in local optima.

Another class of algorithms including Assumed Density Filtering, Expectation Propagation [56] and the online algorithm described in [5] project the complex exact posterior onto a simpler family of distributions indexed by a set of parameters. This projection is done by minimizing the a distance measure between the exact posterior and a member of this family of distributions.

In this work I have explored and extended the concepts central to assumed density filtering by concentrating on the moment matching framework of the algorithm and propose efficient extensions to it. I also give a novel interpretation of this procedure that allows it to be viewed as an exact inference technique instead of an approximate one.

1.1 Moment Matching

In many applications, Bayesian learning can be intractable since the posterior becomes an exponentially large mixture of products of Dirichlets. However, the full posterior is rarely used in practice. A few moments (i.e, mean and variance) are often sufficient to estimate the parameters with some form of confidence.

I have proposed a moment matching technique called Bayesian Moment Matching (BMM) which is based on Assumed Density Filtering. In Bayesian Moment Matching, at every time step, the exact posterior is projected onto a family of distributions indexed by a set of parameters. Given a prior that belongs to this family of distributions; after every observation is received, the exact posterior is computed through a Bayesian update. Then this exact posterior is again projected onto the same family as the initial prior distribution such that some of the moments of the exact posterior are equal to those of the approximate posterior. This allows us to compute all the parameters required to specify the posterior. The moment matching can be done by solving a linear system of equations which can be solved analytically.

Exact Learning w.r.t. an Implied Prior

In Assumed Density Filtering, the primary objective is to find a target distribution in a family of distributions such that the distance between this target distribution and the exact posterior is minimized. This minimization leads to matching of some moments of the exact and approximate posteriors. Given that the observations are informative (i.e. help us distinguish between hidden states), after seeing a very large number of observations, most of the weight of the posterior will be concentrated at one point which will correspond to the first order moment of the posterior (or mean) at that time step. Therefore, in the Bayesian moment matching algorithm proposed in this thesis, we match the first and second order raw moments of the posterior. I have also presented a novel motivation for this moment matching procedure by showing that there exists a set of initial distributions (different from the prior) for which exact Bayesian learning yields the same first and second order moments in the posterior as moment matching. My approach exploits the fact that a small set of moments of the prior need to be specified before any data is observed. After receiving each observation, moment matching allows us to set moments of the posterior. We can solve multiple systems of equations that use the posterior moments to implicitly specify higher order moments in the prior. Hence, the algorithm incrementally specifies the moments of the prior as they become needed in the computation. If we start with this implied prior and perform exact moment matching, then the first order moments of the posterior will be the same as those of the posterior acquired by moment matching. Therefore the overall computation is exact with respect to this prior. To our knowledge, this algorithm is the first Bayesian learning technique that can process each additional word in a constant amount of time and is exact with respect to an implied prior.

I have demonstrated the effectiveness of this technique on two real world problems.

1.2 Topic Modeling

In natural language processing statistical topic models are constructed to discover the abstract topics that occur in a collection of documents. A document typically concerns multiple topics in different proportions. A topic model captures this intuition as a statistical model that examines a set of documents and based on the statistics of the words in each document, discovers what the topics might be and what is the balance of topics for each document. These topics can be used to cluster and organize the corpus. The main computational problem is to infer the conditional distribution of these variables given an observed set of documents [54]. The exact posterior in LDA is an exponentially large mixture of products of Dirichlets. Hence exact inference and parameter learning can be challenging.

Algorithms for topic modeling attempt to construct approximations to the exact posterior either by minimizing some distance metric or by sampling from the exact posterior. Most of these algorithms require the complete learning data to be present in the memory at the time of learning and usually require the model to be retrained if new data arrives. This can be problematic for large streaming corpora. Online versions of some of these algorithms exist that are able to incorporate new information without retraining the complete model. These online algorithms compute the parameters of the approximate posterior by sampling a small set of documents called a mini-batch and then updating the parameters based on this mini-batch such that the overall distance between the exact and approximate posterior is minimized.

1.2.1 Contributions

- **Bayesian Moment Matching for Latent Dirichlet Allocation Model**: In this work, I have proposed a novel algorithm for Bayesian learning of topic models using moment matching called Bayesian Moment Matching (BMM) which processes each new word in a constant amount of time. I derive a method for constructing an initial distribution (different from the prior) for which exact Bayesian Learning yields the same first order moment in the posterior as BMM.

- **State Switching**: State switching occurs because the labels of the hidden states can be permuted $n!$ times. Under each permutation of the model, the probability of generating the observation sequence remains the same rendering all these $n!$ models identical. In this work, I show that even though the exact posterior has $n!$ components, each corresponding to a permutation of the hidden states, moment matching for a slightly different distribution can allow us to compute the moments without enumerating all the permutations.

- **Linear Moment Matching**: I also propose an improvement to the moment matching algorithm called Linear Moment Matching which computes all the necessary moments required to specify the posterior (which is a mixture of product of Dirichlets) in $O(n)$ time where n is the number of topics. In contrast to this, if we use Assumed Density Filtering for a mixture of Dirichlets, we need to solve a system of $n^2 + nm$ non linear equations where m is the total number of observations.

- **Evaluation Against the State of the Art**: We also compare the performance of BMM with other state of the art algorithms for online and offline parameter learning. We demonstrate the effectiveness of each algorithm in terms of perplexity of the test set. We measure the perplexity over various real world document corpora including the UCI bag of words corpora [51] and corpus created from the English version of Wikipedia [40].

- **Topic Modeling in Social Media**: People are increasingly using online social media forums such as twitter, facebook etc. to express their opinions and concerns. With the advent of analytical techniques for natural language processing, many companies are interested in knowing about chatter in the cyber space related to their brand. One way of solving this problem is to tag each tweet or post related to that company with a topic and analyze this data over different time frames. However manual tagging of posts is an expensive process and therefore automated solutions are desirable.

  I have collaborated with an industry partner that does social media mining on behalf of other companies. The data set is comprised of tweets related to a cell phone provider. I have used Bayesian Moment Matching for learning the topics in this data and present a comparison with other methods for topic modeling.

1.3 Activity Recognition with an Instrumented Walker

Mobility aids such as walkers improve the mobility of individuals by helping with balance control and compensating for some mobility restrictions. However, we believe that augmenting these devices with various sensors can create an intelligent assistive technology platform which may allow us to

- Improve the mobility of the user by providing feedback about correct usage.
- Allow care-givers to monitor individuals from afar and react to emergency situations in a timely fashion.
- Provide experts with quantitative measurements to better understand mobility issues and improve care-giving practices.

To better understand the mobility in everyday contexts, Tung et al. [29] developed an instrumented rolling walker shown in Figure 1.1. The walker has a 3-D accelerometer that measures acceleration across x, y and z axis. It also has four load cells mounted just above the wheels that measure the weight distribution on each leg of the walker. There is also a wheel encoder that measures the distance covered by the walker wheels. It also has two cameras mounted on it: one looking forward while the other one looks back at the user's legs. The walker was designed with the aim of autonomously monitoring users outside the laboratory setting. My aim was to develop automated techniques to identify the activities performed by users with respect to their walker (e.g.,walking, standing, turning). This problem is significant for applied health scientists who are studying the effectiveness of walkers to prevent falls. Currently they have to hand label the data by looking at a video feed of the user, which is a time consuming process. An automated activity recognition system would enable clinicians to gather statistics about the activity patterns of users, their level of mobility and the context in which falls are more likely to occur.

Two sets of experiments were conducted to collect data for this analysis. One set of experiments was done with healthy subjects and the other one was done with older adults some of whom were regular walker users. In each experiment users were asked to use the walker to navigate a course and the sensor data was recorded along-with the video. The video is then synced with the sensor data and then we manually label the sensor data with the activities seen in the video to establish ground truth.

1.3.1 Contributions

- **HMM based Model for Activity Recognition using Walker**: In this work I present a probabilistic model for activity recognition of walker users based on Hidden Markov Models (HMMs).
- **Discriminative Model for Activity Recognition using Conditional Random Field** I also derive a discriminative model based on Linear Chain Conditional Random Fields [46] to illustrate the advantages of discriminative models over generative models for this problem.
- **Analysis of Supervised Parameter Learning Techniques**: I evaluate these techniques and models in terms of how well they can predict the actual activity given a sequence of sensor readings. I use labeled data to compute the prediction accuracy for the HMM model and then compare it against the CRF Model.
- **Online Moment Matching Algorithm for Activity Recognition** I also derive an Online Moment Matching algorithm for learning the parameters of the activity recognition HMM. In this approach, I model the posterior as a mixture of Dirichlet

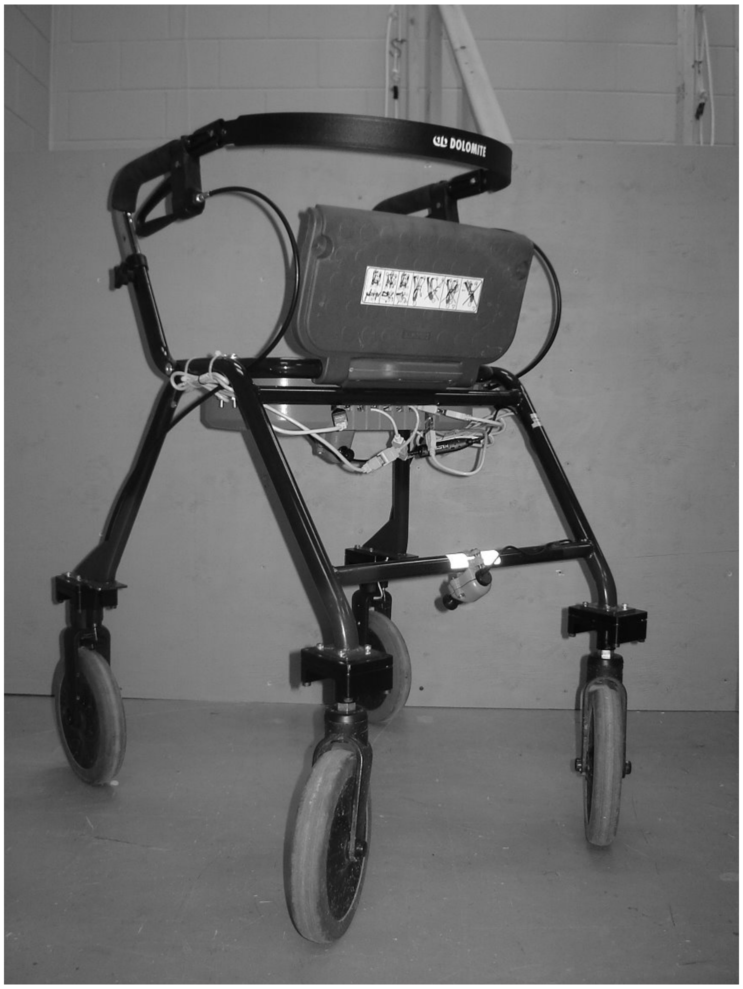


Figure 1.1: The instrumented walker developed by [73] et. al. The walker has a 3-D accelerometer, 4 load cells, 1 wheel encoder and 2 cameras

distributions to optimize the first and second order moments which gives us a linear system of equations which has a closed form solution. This model is distinct from LDA because in LDA model, the order of words in the documents is not important, whereas in time-series models such as HMM, the current activity of the user is dependent on the previous activity of the user.

- **Linear Moment Matching for HMM** In addition to that, I have optimized the moment matching algorithm so that it allows us to specify all the moments with only $O(n)$ computations at every time step where n is the number of activities.

- **Evaluation Against Other Unsupervised Learning Techniques for HMM** I also compare the prediction accuracy of moment matching with the other unsupervised learning algorithms for parameter learning.

1.4 Organization

The rest of this thesis is organized as follows:

In Chapter 2 we introduce some basic concepts about probability and Bayesian learning. We briefly discuss some of the parameter learning techniques for these models. We also introduce Probabilistic Graphical Models (PGMs) and describe one model for topic modeling called Latent Dirichlet Allocation (LDA) and another model for Activity Recognition called Hidden Markov Models.

In Chapter 3 we review some of the previous techniques used for Bayesian learning and discuss how they are different from the work presented in this thesis.

In Chapter 4 we describe an abstract moment matching algorithm for learning a posterior distribution over the parameters of interest given the current data. We also discuss a choice of family of distributions from which we choose our prior. We also describe a method to project other distributions onto this family by moment matching.

In Chapter 5 we derive a moment matching algorithm for the topic modeling problem. We compare our algorithm to other state of the art parameter learning algorithms for topic modeling on some synthetic and some real world data. We also describe a method to construct an implied prior such that if we do exact Bayesian learning by starting off with this prior, the the first moment of the posterior after seeing n words will be the the same as the first moment of the posterior computed using moment matching after seeing the same n words.

In Chapter 6 we present a graphical model for the activity recognition problem based on HMMs. We also describe the set of experiments used for data collection using the instrumented walker. We compare various parameter learning techniques based on their ability to predict the activity correctly given the sensor readings at time t.

In Chapter 7 we give a moment matching algorithm for learning the parameters of an HMM and use it to learn the parameters of the activity recognition HMM described in the previous section.

In Chapter 8 we present conclusions and discuss future work.

Chapter 2

Background

Probability theory has been used for a long time to quantify uncertainty in the real world in the form of probability distributions. Probabilistic Graphical Models allow us to encode and manipulate distributions in high dimensional spaces efficiently. In this chapter we will discuss how real life processes can be modeled as Probabilistic Graphical Models and how they can be used to draw conclusions about events that can not be directly measured.

2.1 Probability Theory and Bayes Rule

The probability of an event can be informally defined as the degree of belief that this event will happen. A random variable is a variable whose possible values are outcomes of a random experiment.

Definition 2.1.1. A probability distribution is a function that assigns a real number $\Pr(X = x)$ to a random variable X which represents the degree of belief that X will take value x. This function must satisfy the following axioms

- $\Pr(X = x) \geq 0$
- $\sum_x \Pr(X = x) = 1$, where the summation is over all possible random events
- $\Pr(X \cup Y) = \Pr(X) + \Pr(Y) - \Pr(X \cap Y)$

In some cases, the distribution has the form $\Pr(X = x|\theta) = f(x, \theta)$. θ is called the parameter of the distribution. In many cases, the parameter represents some statistical characteristic of the model.

Definition 2.1.2. If $g(X)$ is a function of x, then the expectation of g with respect to the distribution $P(x)$ is defined as

$$\mathbb{E}(g(X)) = \int_x g(x)\,P(x)\,dx$$

The "conditional probability" of a random variable $\Pr(X = x|Y = y)$ is the probability that the random variable $X = x$ given $Y = y$. The "joint probability" of two random variables $Pr(X = x, Y = y)$ is the probability that X takes the value x and Y takes the value y. X and Y are "independent" if $\Pr((X|Y) = \Pr(X)$ and $\Pr(Y|X) = \Pr(Y)$.

2.1.1 Bayes Theorem

Bayes theorem allows us to update our prior belief about the value of a random variable given some evidence.

$$\Pr(Y|X) = \frac{\Pr(X|Y)\Pr(Y)}{\Pr(X)} \tag{2.1}$$

$\Pr(X|Y)$ is called the "likelihood", $\Pr(Y)$ is called the "prior distribution", and $\Pr(Y|X)$ is called the "posterior". $\Pr(X)$ is the normalization constant and can be calculated as $\int_y \Pr(X|y)\Pr(y)\,dy$.

Definition 2.1.3. If the posterior $\Pr(Y|X)$ and the prior $\Pr(Y)$ are in the same family of distributions, then the prior $\Pr(Y)$ is called the **conjugate prior** for the likelihood $\Pr(X|Y)$.

2.2 Probabilistic Graphical Models

Probabilistic Graphical Models are used to encode uncertainty in real world processes as probability distributions over high dimensional spaces. Below we will describe a type of probabilistic graphical model called Bayesian Networks. Bayesian Networks are generative models which means that they can be used to generate (i.e. sample) data from the joint distribution over the variables. Later we will also describe Conditional Random Fields (CRF) which are discriminative Graphical Models. In CRFs we model the conditional probability of the hidden variables given the observed variables.

2.2.1 Bayesian Networks

A Bayesian Network or Bayes Net is a directed acyclic graph [63] in which each node represents a random variable and each arc represents a conditional dependence. The node from which the arc originates is called the parent of the node on which the arc terminates. The arc captures the effect of the value of the parent node on the probability that the child will take a certain value. For a random variable X_i, $Parents(X_i)$ is the set of random variables that are its parents. A conditional probability distribution is associated with each node which quantifies the effect of all its parents. Bayesian Networks can be used to calculate the full joint distribution using the following equation

$$\Pr\left(X_1 = x_1, \ldots, X_n = x_n\right) = \prod_{i=1}^{n} \Pr\left(x_i | Parents\left(x_i\right)\right) \tag{2.2}$$

Here $Parents\left(x_i\right)$ refers to a possible assignment of specific values to each random variable in the set $Parents(X_i)$. Each conditional distribution can be of the form $\Pr\left(x_i | Parents(x_i)\right) = f\left(x_i | \theta_{Parents(x_i)}\right)$ where $\theta_{Parents(x_i)}$ is the value of θ corresponding to an assignment of values to members of $Parents(X_i)$.

2.2.2 Inference in Bayesian Networks

Inference or prediction involves computing the posterior probability distributions of some query variables Y given some observed random variables or evidence using

$$\Pr\left(Y | e\right) = \frac{\Pr\left(Y, e\right)}{\Pr\left(e\right)} = \frac{\sum_x \Pr\left(Y, e, x\right)}{\Pr\left(e\right)} \tag{2.3}$$

Here X are non-query and non-observable variables called ”hidden variables” which may be some quantity of interest for example an activity that is not directly observable. ”Variable Elimination” is an efficient algorithm which can solve the inference problem exactly. However, Variable elimination can have exponential space and time complexity (in the number of variables) in the worst case [67].

2.2.3 Parameter Learning in Bayesian Networks

Learning the parameters of a Bayesian network from observable data is an important and challenging problem. Each conditional distribution in Eq. 2.2 can have a particular parametric form. A possible assignment of values to these parameters can be treated as a hypothesis. Parameter learning involves choosing the hypothesis that best describes the observed data. We now describe some of the learning techniques for Bayesian Networks.

Bayesian Learning

In Bayesian Learning, we compute a distribution over the hypothesis space for the parameters given the observed data using Bayes Rule. Let the observations be represented by e, then the probability of the parameters Θ is given by

$$\Pr(\Theta|e) = \frac{1}{\Pr(e)} \underbrace{\Pr(e|\Theta)}_{\text{likelihood}} \underbrace{Pr(\Theta)}_{\text{prior}} \tag{2.4}$$

Predictions are made by using all hypotheses weighed by their probabilities [67]. In order to make a prediction about an unknown quantity Y, we use

$$\Pr(Y|e) = \int \Pr(Y|e,\Theta) \Pr(\Theta|e)\, d\Theta \tag{2.5}$$

Bayesian predictions are optimal even when the data set is small. Given the prior, any other prediction will be correct less often on average[67]. However, for real learning problems, the parameter space may be very large or infinite and the evaluation of the integral in Equation 2.5 becomes intractable. In the next section we will discuss some of the techniques commonly used for Bayesian learning.

Maximum A Posteriori Hypothesis (MAP)

MAP is a common approximation to Bayesian learning. We make predictions based on the most probable set of parameters called the "maximum a posteriori" hypothesis [67]. Equation 2.4 is approximated to

$$\theta_{MAP} = \max_{\theta} \Pr(e|\theta) \Pr(\theta) \tag{2.6}$$

$$\Pr(Y|e) \approx \Pr(Y|e, \theta_{MAP}) \tag{2.7}$$

Maximum Likelihood Learning (ML)

A further simplification is to choose a uniform prior over the hypothesis space[67], then MAP learning reduces to

$$\theta_{ML} = \max_{\theta} \Pr(e|\theta) \tag{2.8}$$

$$\Pr(Y|e) \approx \Pr(Y|e, \theta_{ML}) \tag{2.9}$$

Even in cases where the integral has a closed form solution, the presence of hidden variables may make the posterior over the variables complex

$$\Pr(\Theta|e) = \sum_{y} \Pr(\Theta, Y = y|e) \tag{2.10}$$

For high dimensional hidden variable Y, this sum could be an infinitely large mixture. Therefore, we have to resort to approximate methods.

2.3 Bayesian Learning

As mentioned before, Bayesian predictions are optimal even when the data set is small. However, for real learning problems, the parameter space may be very large or infinite and the evaluation of the integral in Equation 2.5 becomes intractable. In the next section we will discuss some of the techniques commonly used for Bayesian learning. In addition to that, hidden variables introduce an extra level of complexity. According to Equation 2.3 the full posterior is constructed by summing out all possible values of the hidden variables. If we have t hidden variables, each of which can take d different values, this expression will grow in the order of d^t. Therefore, we have to resort to techniques that try to approximate the posterior. Below we will discuss a few of them.

2.3.1 Mean-Field Variational Inference (Variational Bayes)

In Mean-field variational inference, a family of distributions over the hidden variables is chosen that is indexed by a set of free parameters. The parameters are optimized to find the member of the family that is closest to the posterior of interest. This closeness is measured with Kullback-Leibler divergence. [41]. The resulting distribution, called the variational distribution, is then used to approximate the posterior. Variational inference minimizes the Kullback-Leibler (KL) divergence from the variational distribution to the posterior distribution. It maximizes the evidence lower bound (ELBO), a lower bound on the logarithm of the marginal probability of the observations. The ELBO is equal to the negative KL divergence up to an additive constant. We want to project the complex posterior $\Pr(\Theta|e) = P(\Theta)$ into a space of simpler factored distributions $Q(\Theta)$ such that the KL divergence between the P and Q is minimized. [15] [11].

$$\begin{aligned} D_{KL}\{Q||P\} &= \int_{\Theta} Q(\Theta) \log \frac{Q(\Theta)}{P(\Theta)} = \int_{\Theta} Q(\Theta) \log \frac{Q(\Theta)}{\Pr(\Theta|e)} \qquad (2.11) \\ &= \int_{\Theta} Q(\Theta) \log \frac{Q(\Theta)}{\Pr(\Theta, e)} + \log \Pr(e) \\ \log \Pr(e) &= D_{KL}\{Q||P\} - \int_{\Theta} Q(\Theta) \log \frac{Q(\Theta)}{\Pr(\Theta, e)} \\ &= D_{KL}\{Q||P\} + \mathcal{L}(\Theta) \qquad (2.12) \end{aligned}$$

$\mathcal{L}(\Theta)$ is called evidence lower bound (ELBO). It is a lower bound on the logarithm of the marginal probability of the observations. Since $\log \Pr(e)$ is constant, maximizing $\mathcal{L}(\Theta)$ minimizes the KL divergence. In traditional mean-field variational inference, this optimization is done using coordinate ascent. Each variational parameter is iteratively optimized while holding the other parameters fixed.

Online Variational Bayes

The coordinate ascent algorithm becomes inefficient for large data sets because the local variational parameters must be optimized for each data point. Stochastic variational inference uses stochastic optimization to fit the global variational parameters. The data is repeatedly subsampled to form noisy estimates of the natural gradient of the ELBO, and these estimates are followd with a decreasing step-size [41].

2.3.2 Gibbs Sampling

Monte Carlo Markov Chain (MCMC) methods are a popular class of methods that allow us to construct approximations of integrals that are difficult to compute analytically. The idea behind MCMC is to construct a Markov chain whose stationary distribution is the posterior distribution of interest. The algorithm simulates the chain until it gets close to stationarity. After this “burn-in” period, samples from the chain serve as an approximation to the true posterior distribution $P(\Theta|e)$. The most common variations of this approach are the Metropolis-Hastings algorithm and Gibbs sampling. For a more extensive overview of MCMC techniques, see [30, 66].

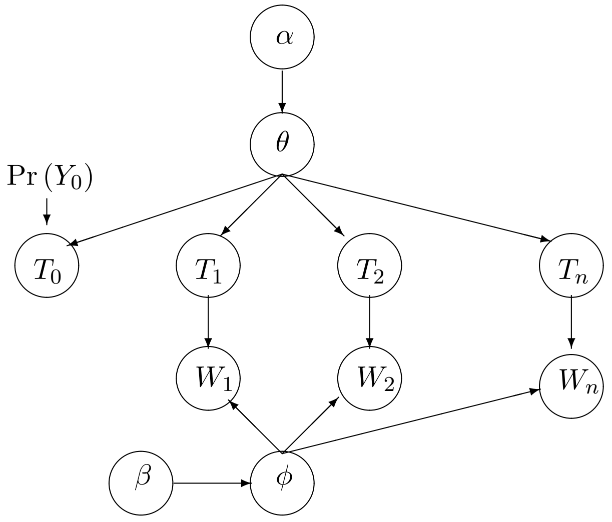


Figure 2.1: Latent Dirichlet Allocation Topic Model

Gibbs sampling is a class of MCMC algorithms which consists of repeatedly sampling each variable in our model from the conditional distribution given all the other variables. For example, if θ is a vector of parameters $\theta = \theta_1, \theta_2, ..., \theta_k$ and e is the observation, the Gibbs sampling algorithm samples θ_1 from $\Pr(\theta_1|\theta_{2:k}, e)$ followed by sampling θ_2 from $\Pr(\theta_2|\theta_1, \theta_{3:k}, e)$ all the way to θ_k, after which we start again from sampling θ_1. It can be shown that the resulting Markov chain has stationary distribution $P(\theta_1, ..., \theta_k|e)$.

Gibbs Sampling is very popular owing to its simplicity. However, one of the major obstacles with Gibbs Sampling is to decide on a stopping criterion.

2.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative Bayesian model that describes a joint distribution over the topics and words in a document corpus. It was introduced by Blei et. al. in [15]. Each word in each document is associated with a hidden topic. Let the total number of topics be T, the total number of documents be D and the total words in the vocabulary be W. The number of words in document d is N_d. We denote the hidden topic of the n^{th} word by t_n. Then the model is is parametrized by the following distributions

- **Word-Topic Distribution** Each document d is represented by a multinomial topic distribution $\theta_d = \{\theta_{d,1}, \ldots, \theta_{d,T}\}$ where $\theta_{d,t}$ is the probability of topic t in document d.

- **Document-Topic Distribution** The distribution over words given a particular topic t is called the Topic-Word distribution and is denoted by $\phi_t = \{\phi_{t,1}, \ldots, \phi_{t,W}\}$, where W is the total number of words in the dictionary.

Each θ_d is usually modeled as a multinomial distribution and is sampled from a prior $f(\theta_d; \alpha)$. For LDA the prior is a Dirichlet distribution of the form

$$Dir\,(\theta_d, \alpha_d) = \frac{\Gamma(\sum_{i=1}^{T} \alpha_{d,i})}{\prod_{i=1}^{T} \Gamma(\alpha_{d,i})} \prod_{i=1}^{n} \theta_{d,i}^{\alpha_{d,i}-1} \; 0 \leq \theta_{d,i} \leq 1 \; \sum_i \theta_{d,i} = 1 \tag{2.13}$$

For a corpus, each ϕ_t is sampled from another prior $g(\phi_t; \beta)$ which is a dirichlet parmaterized by β. The n^{th} word in document d can be generated by first sampling a topic t from θ_d and then sampling a word w_n from ϕ_t. We will use the following terminology frequently for notational convenience

$\Theta = \{\theta_1, \ldots, \theta_D\}$

$\Phi = \{\phi_1, \ldots, \phi_T\}$.

$w_{a:b} = w_a, w_{a+1} \ldots w_b$ a sequence of words from time a to b

$t_{a:b} = t_a, t_{a+1} \ldots t_b$ a sequence of hidden states from time a to b

$P_n\,(\Theta, \Phi) = \Pr\,(\Theta, \Phi | w_{1:n})$

$k_n = \Pr\,(w_n | w_{1:n-1})$

$c_n = \Pr\,(w_{1:t})$

2.4.1 Learning and Inference

In LDA, given a corpus, we are interested in identifying the mixture of topics pertaining to a particular document and estimating Θ and Φ. We can use Bayes rule to estimate Θ and Φ for a corpus by computing the posterior $P_n(\Theta, \Phi) = \Pr(\Theta, \Phi | w_{1:n})$ using Bayesian

Learning. If the n^{th} word w_n lies in document d, the posterior can be calculated by

$$\begin{aligned} P_n(\Theta, \Phi) &= \Pr(\Theta, \Phi | w_{1:n}) \qquad (2.14) \\ &= \sum_t \Pr(\Theta, \Phi, t_n = t | w_{1:n}) \\ &= \frac{\sum_t \Pr(t_n{=}t|\theta_d) \Pr(w_n|t_n{=}t, \phi_t) \Pr(\Theta, \Phi|w_{1:n-1})}{\Pr(w_n|w_{n-1})} \\ &= \frac{1}{k_n} \sum_t \theta_{d,t}\phi_{t,w_n} P_{n-1}(\Theta, \Phi) \\ k_n &= \sum_t \int_\Theta \int_\Phi \theta_{d,t}\phi_{t,w_n} P_{n-1}(\Theta, \Phi) d\Theta d\Phi \qquad (2.15) \end{aligned}$$

Let the prior $P_0(\Theta, \Phi) = f(\Theta, \Phi|\alpha, \beta)$ be a distribution in θs and ϕs where α and β are sets of parameters of the distribution f. Then according to Eq. 2.14 the number of terms in the posterior grows by a factor of T after each word due to the summation over t. After n words, the posterior will consist of a mixture of T^n terms, which is intractable.

2.5 Dynamic Bayesian Networks

Dynamic Bayesian Networks are generative Bayesian models that describes a joint distribution over sequential data. Evidence is acquired at every time step and based on this evidence the values of the hidden variables are updated.

2.5.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a DBN in which each observation e_t is associated with a hidden state y_t that depicts some quantity of interest that is not observable directly. An HMM is shown in Figure 2.2. In HMMs, the Markov assumption states that the current state is only dependent on the state at the previous time step. The Stationary process assumption states that the model parameters do not change over time. This setting is useful in many domains, including activity recognition, speech recognition, natural language processing. In activity recognition, activities are the hidden states and sensor measurements are the observations. The user performs a sequence of activities that are not directly observable but can be measured through sensors. Since the activities are not scripted nor directly observable, then we are faced with an unsupervised learning problem. It may also be desirable to learn the model parameters as the activities are performed, meaning that learning should be done in an online fashion. We denote the number of states by N and the number of observations by M. There are two distributions that represent this model

- **Transition Model**: The transition distribution models the change in the value of the hidden state over time. The distribution over the current state Y_t given that the previous state is y is denoted by $\theta_y = \Pr(Y_t|Y_{t-1} = y)$ where $\theta_y = \{\theta_{y,1}, \ldots, \theta_{y,N}\}$ and $\theta_{y,i} = \Pr(Y_t = i|Y_{t-1} = y)$ is the probability that the current state is i given that the previous state was y.
- **Observation Model**: The observation function models the effect of the hidden state on the observation at any given time t. The distribution over observations given a particular state is denoted by $\phi_y = \Pr(E_t|Y_t = y)$ where $\phi_y = \{\phi_{y,1}, \ldots, \phi_{y,M}\}$ and $\phi_{y,e} = \Pr(E_t = e|Y_t = y)$ is the probability that observation e is seen if the current state is y.

To generate the observation sequence, for each $y \in \{1, \ldots, N\}$ we first sample θ_y from a prior $f(\theta_y; \alpha_y)$ and ϕ_y from another prior $f(\phi_y; \beta_y)$. Then we generate each observation by first sampling the current state y' from θ_y where y is the state sampled at the previous time step and then sample an observation e from $\phi_{y'}$.

We will use the following terminology frequently for notational convenience

$\Theta = \{\theta_1, \ldots, \theta_N\}$

$\Phi = \{\phi_1, \ldots, \phi_N\}$.

α set of hyperparameters for the transition distribution

β set of hyperparameters for the observation distribution

$e_{a:b} = e_a, e_{a+1} \ldots e_b$ a sequence of evidence from time a to b

$y_{a:b} = y_a, y_{a+1} \ldots y_b$ a sequence of hidden states from time a to b

$P_t(\Theta, \Phi) = \Pr(\Theta, \Phi|e_{1:t})$

$P_t^y(\Theta, \Phi) = \Pr(\Theta, \Phi|Y_t = y, e_{1:t})$

$k_t^y = \Pr(Y_t = y, e_t|e_{1:t-1})$

$k_t = \Pr(e_t|e_{1:t-1})$

$c_t^y = \Pr(Y_t = y|e_{1:t})$

$c_t = \Pr(e_{1:t})$

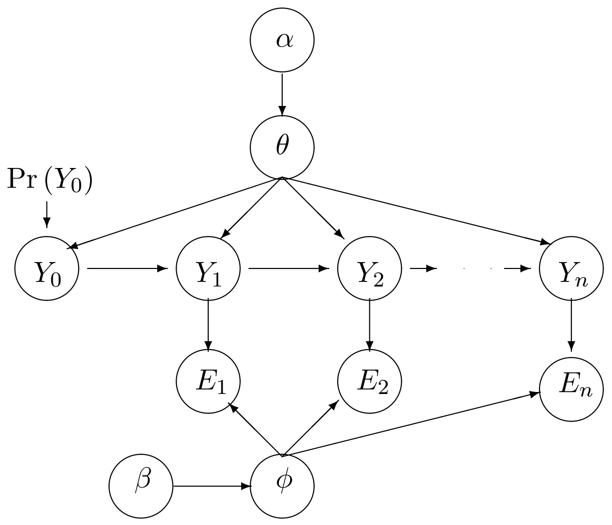


Figure 2.2: A Bayesian Hidden Markov Model

2.5.2 Inference

There are 3 inference tasks that are typically performed with an HMM:

- Filtering : This is the task of computing the posterior over the hidden variables given all previous evidence by integrating out the parameters Θ and Φ. $Pr(Y_t = y|e_{1:t}, \Theta, \Phi)$ can be computed using

$$\Pr(Y_t = y|e_{1:t}, \Theta, \Phi) = \frac{1}{\Pr(e_{1:t}|\Theta, \Phi)} \Pr(e_t|Y_t = y, \Phi) \sum_{i=1}^{N} \Pr(Y_t = y|Y_{t-1} = i, \Theta) \Pr(Y_{t-1} = i|e_{1:t-1}, \Theta, \Phi) \tag{2.16}$$

- Smoothing: This is the task of computing the posterior distribution over a past state $0 < k < t$ given all evidence up to time t: $\Pr(Y_k = y|e_{1:t})$

$$\Pr(Y_k = y|e_{1:t}, \Theta, \Phi) = \frac{1}{\Pr(e_{1:t}|\Theta, \Phi)} \Pr(Y_k = y|e_{1:k}, \Theta, \Phi) \Pr(e_{k+1:t}|Y_k = y, \Theta, \Phi) \tag{2.17}$$

This is also called the Baum Welch algorithm. It is important to note this posterior can not be computed online as it requires future observations. $\Pr(e_{k+1:t}|Y_k)$ can be recursively calculated using

$$\Pr\left(e_{k+1:t}|Y_k = y, \Theta, \Phi\right) = \sum_{Y_{k+1}=1}^{N} \Pr\left(e_{k+1}|y_{k+1}, \Phi\right) \Pr\left(e_{k+2:t}|y_{k+1}, \Theta, \Phi\right) \\ \Pr\left(y_{k+1}|Y_k = y, \Theta\right) \tag{2.18}$$

- Most likely explanation: This is the task of computing the most likely sequence of states that generated a particular sequence of observations $argmax_{y_{1:t}} \Pr\left(y_{1:t}|e_{1:t}\right)$. This is also called the "Viterbi Algorithm". It can be calculated using the following recursive equation,

$$\begin{aligned} &\max_{y_{1:t}} \left(\Pr\left(y_{1:t}, Y_{t+1} = y|e_{1:t+1}, \Theta, \Phi\right)\right) \\ \propto &\Pr\left(e_{t+1}|Y_{t+1} = y, \Phi\right) \max_{y_t} \left(\Pr\left(Y_{t+1} = y|y_t, \Theta\right) \max_{y_{1:t-1}} \left(\Pr\left(y_{1:t-1}, y_t|e_{1:t}, \Theta, \Phi\right)\right)\right) \end{aligned} \tag{2.19}$$

2.5.3 Bayesian Filtering

Filtering : In Bayesian Filtering, we compute the posterior over the hidden variables given all previous evidence $Pr(Y_t = y|e_{1:t})$ by integrating Θ and Φ. We use the symbol c_t^y to denote this posterior as defined in section 2.5.1. We will see later that in Bayesian Learning for HMMs, this posterior will be used as a normalization constant.

$$\begin{aligned} c_t^y =& \Pr\left(Y_t = y|e_{1:t}\right) \\ =& \frac{1}{\Pr\left(e_t|e_{1:t-1}\right)} \int_{\Theta,\Phi} \Pr\left(e_t|Y_t = y, \Phi\right) \sum_{i=1}^{N} \Pr\left(Y_t = y|Y_{t-1} = i, \Theta\right) \\ & \Pr\left(\Theta, \Phi|Y_{t-1} = i, e_{1:t-1}\right) \Pr\left(Y_{t-1} = i|e_{1:t-1}\right) d\Theta d\Phi \\ =& \frac{1}{k_t} \int_{\Theta,\Phi} \phi_{y,e_t} \sum_{i=1}^{N} \theta_{i,y} P_{t-1}^i\left(\Theta, \Phi\right) c_{t-1}^i d\Theta d\Phi \end{aligned} \tag{2.20}$$

2.5.4 Parameter Learning

We will now discuss how to learn the parameters of an HMM.

- Supervised Maximum Likelihood Parameter Learning : If the values of both the hidden variables $y_{1:T}$ and the evidence $e_{1:t}$ are known, then learning from both sequences is called "supervised learning". The optimal parameters Θ^* and Φ^* can be learned from data by maximizing the log likelihood of the data.

$$\Theta^*, \Phi^* = argmax_{\Theta,\Phi} \log\left(\Pr\left(y_{1:T}, e_{1:T} | \pi, \theta, \psi\right)\right) \tag{2.21}$$

subject to

$$\sum_{y'=1}^{N} \theta_{y',y} = 1 \forall y \in \{1, \ldots, N\} \quad \sum_{e=1}^{M} \phi_{y,e} = 1 \forall y \in \{1, \ldots, N\}$$

This optimization problem has an analytical solution which we will discuss in subsequent chapters.

- Unsupervised Maximum Likelihood Parameter Learning : If the values of the hidden variables are not available, then this is called unsupervised learning. Ideally we should maximize $\sum_{y_{1:t}} \Pr\left(y_{1:T}, e_{1:T} | \Theta, \Phi\right)$ over Θ and Φ. However this results in a non convex optimization problem. Instead, we use the "Expectation Maximization Algorithm" [13] which optimizes a convex approximation of this function by maximizing the expected value of the log likelihood

$$\Theta^{i+1}, \Phi^{i+1} = argmax_{\Theta,\Phi} \sum_{y_{1:t}} \left(\Pr\left(y_{1:T}, e_{1:T} | \Theta^i, \Phi^i\right)\right) \log\left(\Pr\left(y_{1:T}, e_{1:T} | \Theta, \Phi\right)\right) \tag{2.22}$$

subject to

$$\sum_{y'=1}^{N} \theta_{y',y} = 1 \forall y \in \{1, \ldots, N\} \quad \sum_{e=1}^{M} \phi_{y,e} = 1 \forall y \in \{1, \ldots, N\}$$

Here we sum over all possible sequences $y_{1:T}$ where each $y_i = 1, \ldots, N$.

- Bayesian Learning in an HMM : In Bayesian Learning we calculate a posterior distribution over the parameters given a prior distribution and the data using

$$\Pr\left(\Theta, \Phi | e_{1:t}\right) = \sum_{y} \Pr\left(\Theta, \Phi | Y_t = y, e_{1:t}\right) \Pr\left(Y_t = y | e_{1:t}\right) = \sum_{y} P_t^y\left(\Theta, \Phi\right) c_t^y \tag{2.23}$$

Using Bayes Rule,

$$\begin{aligned}
&P_t^y(\Theta,\Phi) \qquad (2.24)\\
&=\Pr(\Theta,\Phi|Y_t=y,e_{1:t})=\frac{\Pr(\Theta,\Phi,Y_t=y,e_{1:t})}{\Pr(Y_t=y,e_{1:t})}\\
&=\frac{\Pr(e_t|Y_t=y,\Phi)}{\Pr(Y_t=y,e_t|e_{1:t-1})}\sum_{i=1}^{N}\Big[\Pr(Y_t=y|Y_{t-1}=i,\Theta)\Pr(\Theta,\Phi,Y_{t-1}=i|e_{1:t-1})\\
&\quad\Pr(Y_{t-1}=i|e_{1:t-1})\Big]\\
&=(1/k_t^y)\,\phi_{y,e_t}\sum_{i=1}^{N}\theta_{i,y}c_{t-1}^{i}P_{t-1}^{i}(\Theta,\Phi)
\end{aligned}$$

where

$$k_t^y=\int_{\Phi,\Theta}\phi_{y,e_t}\sum_i\theta_{i,y}c_{t-1}^iP_{t-1}^i(\Theta,\Phi)\,d\Phi d\Theta \qquad (2.25)$$

We can also calculate the posterior $\Pr(Y_t=y|e_{1:t})=c_t^y=k_t^y/\sum_i k_t^i$.

Let the prior $P_0(\Theta,\Phi)=f(\Theta,\Phi|\alpha,\beta)$ be some distribution in θs and ϕs where α and β are the parameters of the distribution f. Then according to Eq. 2.24 the number of terms in the posterior grows by a factor of N after each observation due to the summation over i. After t observations, the posterior will consist of a mixture of N^t terms, which is intractable. In the next Chapters we will discuss how to approximate this posterior in an online fashion.

2.6 Conditional Random Fields

Conditional Random Fields (CRFs) are an alternative to HMMs [46]. Unlike HMMs, CRFs are discriminative, i.e., they model only the conditional distribution of the hidden variables given the evidence. An important advantage of this approach is that there are no assumptions on the distribution of the evidence.

Definition 2.6.1. A clique in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.

Given a set of observable random variables $E_{1:t}$ and hidden variables $Y_{1:t}$, a CRF can be defined by a tuple $(E, Y, \mathcal{G}, \mathcal{F}, \lambda)$. Here $\mathcal{G}$ is an undirected graph with vertices V corresponding to the random variables and a set of cliques $\mathcal{C}$ that model the interactions between them. $\mathcal{F} = (f_C)_{C \in \mathcal{C}}$ is a family of real valued functions. For every clique $C \in \mathcal{C}$ that has vertices E_C and Y_C,

$$f_C : \mathbb{E}_{\mathbb{C}} \times \mathbb{Y}_{\mathbb{C}} \to \mathbb{R}^{n(C)} \tag{2.26}$$

where $n(C) \in \{1, 2, \ldots\}$ is the dimension of the feature vector, $\mathbb{E}_{\mathbb{C}}$ is the set of all possible assignments to nodes in E_C and $\mathbb{Y}_{\mathbb{C}}$ is the set of all possible assignments to nodes in Y_C. For all $C \in \mathcal{C}$, $\lambda_C \in \mathbb{R}^{n(C)}$ is a family of weights. Usually the feature functions of a CRF are fixed while the weights are learned from data. The probability of $Y_{1:T} = y_{1:T}$ conditioned on $e_{1:T}$ is given by

$$\Pr{}_{\lambda}\left(y_{1:t}|e_{1:t}\right) = \frac{1}{Z_\lambda\left(e_{1:t}\right)} exp \left\{ \sum_{C \in \mathcal{C}} \lambda_C f_C\left(e_C, y_C\right) \right\} \tag{2.27}$$

The normalization constant can be evaluated as

$$Z_\lambda\left(e_{1:t}\right) = \sum_{y_{1:t} \in \mathbb{Y}_{1:t}} \exp \left\{ \sum_{C \in \mathcal{C}} \lambda_C f_C\left(e_C, y_C\right) \right\} \tag{2.28}$$

2.6.1 Linear Chain CRF

In a linear chain CRF with observations $e_{1:T}$ and labels $y_{1:T}$, the set of cliques $\mathcal{C}$ is given by $\mathcal{C} = \{\{e_t, y_t\} \forall t \in \{1, \ldots, T\}\} \cup \{\{y_t, y_{t-1}\}, \forall t \in \{2, \ldots, T\}\}$. The CRF includes two kinds of feature functions $f_t\left(e_t, y_t\right) = f\left(e_t, y_t\right)$ with weight $\lambda_t = \mu$ and $f_{t-1,t}(e_{1:t}, y_{t-1}, y_t) = g\left(y_{t-1}, y_t\right)$ with weight $\lambda_{t-1,t} = \nu$.

$$\begin{aligned} \Pr\left(y_{1:t}|e_{1:t}\right) &= \frac{1}{Z_\lambda\left(e_{1:t}\right)} \exp \left\{ \sum_{C \in \mathcal{C}} \lambda_C f_C\left(e_C, y_C\right) \right\} \\ &= \frac{1}{Z_\lambda\left(e_{1:t}\right)} \exp \left\{ \sum_{t=1}^{T} \mu f\left(e_t, y_t\right) + \sum_{t=2}^{T} \nu g\left(y_{t-1}, y_t\right) \right\} \end{aligned} \tag{2.29}$$

Inference

For inference in a CRF, let M be a $|\mathbb{Y}| \times |\mathbb{Y}|$ matrix, such that

$$M_t\left(i, j\right) = \exp\left\{\mu\, f\left(e_t, j\right) + \nu\, g\left(i, j\right)\right\} \tag{2.30}$$

We can use a Baum-Welch type procedure similar to the HMM and use the following recursions to generate the forward and backward messages

$$\alpha_y\left(1\right) = \mu\, f\left(e_1, y\right) \qquad \alpha_y\left(t\right) = \alpha_{t-1} M_t \qquad t = 2, \ldots, T \tag{2.31}$$

$$\beta_y\left(N\right) = 1 \qquad \beta_y\left(t\right) = \alpha_{t-1} M_{t+1} \beta_{t+1}^T \qquad t = T-1, \ldots, 1 \tag{2.32}$$

It can be shown that $Z_\lambda = \sum_{y \in \mathbb{Y}} \alpha_y\left(T\right)$.

Parameter Learning

Learning the parameters of the CRF corresponds to learning the weights μ and ν of each feature function. We setup the following optimization problem to maximize the conditional log likelihood of the data

$$\arg\max_{\mu,\nu} \left(-\log\left(Z_\lambda\left(e_{1:T}\right)\right) + \sum_{t=1}^{T} \mu\, f\left(e_t, y_t\right) + \sum_{t=2}^{T} \nu\, g\left(y_{t-1}, y_t\right) - \frac{\lambda^T \lambda}{2\sigma^2}\right) \tag{2.33}$$

The last term on the right hand side is a regularizer which penalizes large weights. In a Bayesian framework, it can be regarded as a Gaussian prior with mean 0, variance σ^2 and all weights are uncorrelated. This is a convex optimization problem and can be solved using gradient based search.

Chapter 3

Related Work

In this chapter, we review some of the techniques that have been previously used for Bayesian learning. We also review the literature in the context of Latent Dirichlet Allocation Models as well as models for activity recognition with instrumented devices.

3.1 Parameter Learning for Latent Dirichlet Allocation

One of the main challenges in Bayesian learning is that the posterior distribution can become very complex as new information becomes available. In some cases, the posterior can be an exponentially large mixture of distributions. Hence representing the posterior and calculating its statistics can be computationally expensive. Typically algorithms aim to construct an approximate posterior.

We have described Gibbs Sampling [52] briefly in the previous section. Griffiths et. al. [34] use Gibbs Sampling to estimate the parameters of the LDA model. Gibbs Sampling is popular for its simplicity and its ability to produce good estimates. The algorithms passes over the data multiple times, however to process each observation, it only requires $O(n)$ time where n is the number of hidden states. One of the major weaknesses of Gibbs Sampling is that convergence of the Markov chain is difficult to detect reliably. In addition to that, the stochastic nature of the algorithm leads to estimates that vary with each run.

Another popular class of techniques called Variational Bayesian techniques construct an approximation to the posterior by minimizing the KL divergence between the exact posterior and a simpler distribution. The latent nature of the hidden variables leads to a non-convex optimization problem, therefore, instead of minimizing the KL divergence, an upper bound of the KL divergence is optimized. For LDA Variational Bayes exploits the exchangeability assumption and is able to process multiple occurrences of the same word in a document at the same time. Variational Bayes methods can be much faster than sampling based approaches but tend to underestimate the posterior variance[14]. Blei et. al. [15] have utilized Variational Bayesian techniques to learn the parameters of the LDA Model.

Spectral learning algorithms([8],[7],[16],[42]) are based on spectral decomposition of moment matrices or other algebraic structures of the model. In Excess Correlation Analysis [7] the parameters are estimated by matching the moments of the model with the empirical moments. The technique is computationally simple (e.g., matrix operations and singular value decomposition) and it ensures consistency. Despite its theoretical guarantees, ECA often generates negative probabilities. This arises from the fact that ECA does not enforce non-negative solutions (which would be NP-hard) and the empirical moments are necessarily approximations of the true underlying moments.

Most of the above mentioned algorithms require the complete data to be present in the memory at the time of learning and usually require the model to be retrained if new data arrives. This can be problematic for large data sets and applications where new data becomes available frequently. The Moment Matching algorithm that is proposed in this thesis is an online technique which means that updating the model parameters after receiving each new observation requires a constant amount of time.

3.1.1 Online Parameter Learning for LDA

More recently, online versions of some of the algorithms mentioned above have been developed that are able to incorporate new information without retraining the complete model. The Online Variational Bayes algorithm [40] and the sparse stochastic online inference algorithm [54], both utilize online stochastic optimization to compute an approximate posterior distribution for LDA. At each iteration, these methods sample some documents from the corpus and then update the parameters by taking a step in the direction of the gradient that minimizes KL divergence between the approximate posterior and the exact posterior. The authors show that the algorithm converges to a local optimum of the variational bound. However, these algorithms suffer from the same limitation as offline Variational Bayes. In addition to that, the approximation produced by these algorithms is only as good as a single pass of the Variational Bayes algorithm and therefore they require a lot of data to produce reasonable estimates.

3.2 Activity Recognition for Instrumented Walker

There has been some previous work for activity recognition with instrumented walkers. In [6] the authors describe a method that assesses basic walker assisted gait characteristics. Their model is based on the measurement of weight transfer between the user and the walker by two load cells in the handles of the walker. They use a total of 8 participants for their experiments with each users performing a total of 50 experiments emulating 16 navigational scenarios including walking, turns and docking to a chair. A simple thresholding approach is used to detect peaks and valleys in the load measurements, which are assumed to be indicative of certain events in the gait cycle. This work focuses on low level gait statistics where as we are interested to recognize complex high level activities. Hirata et. al. [39] instrumented a walker with sensors and actuators. They recognize three user states: walking, stopped and emergency (including falling). These states are inferred based on the distance between the user and the walker (measured by a laser range finder) and the velocity of the walker. This work is limited to the three states mentioned above and would not be able to differentiate between activities that exhibit roughly the same velocity and distance measurements (e.g., walking, turning, going up a ramp).

A significant amount of work has been done on activity recognition in other contexts. In particular, Liao et. al. [49] use a Hierarchical Markov Model to learn and infer a user's daily movements through an urban community. The model uses multiple levels of abstraction in order to bridge the gap between raw GPS sensor measurements and high level information such as a user's destination and mode of transportation. They use Rao-Blackwellized particle filters for state estimation.

Patel et. al. [62] have designed a walker equipped with the following sensors namely, IRt (Infra-red Torso), IRw (Infra-red Waist), LSG (Left Strain Gauge), RSG (Right Strain Gauge), RF (wireless Radio Frequency switch), LOC (Localisation) and TOD (Time-of-Day). They generate artificial data for a users navigating through a home. They use a Dynamic Bayesian Network and an SVM to do activity recognition. Their work is inspired by the work presented in Chapter 6 and in [61].

3.2.1 Parameter Learning in HMM

Parameter Learning in time series data is a challenging problem. Many of the methods described in the previous section have also been used for parameter learning in HMMs. Beal [11] has used Variational Bayes to learn the parameters of an HMM with discrete hidden states and observations in an offline fashion.

A popular class of methods for learning the parameters of an HMM is a technique called Expectation Maximization which finds point estimates of the parameter values such that the expected likelihood of the observed data is maximized [13]. We have introduced this algorithm in the previous section. In expectation maximization, a concave underestimator of the log likelihood is maximized. Expectation Maximization is also an offline technique which means that the full data is required to be in the memory for learning and multiple passes are required for the algorithm to converge. EM is useful for several reasons: conceptual simplicity, ease of implementation, and the fact that each iteration improves the value of the parameters. The rate of convergence on the first few steps is typically quite good, but can become slow as you approach a local optimum. Generally, EM works best when the fraction of missing information is small and the dimensionality of the data is not too large. EM can require many iterations, and higher dimensionality can dramatically slow down the E-step [69].

An online version of the Expectation Maximization algorithm for discrete Hidden Markov Models has been developed by Mongillo et. al. [57]. Cappe et. al. have also derived an online EM algorithm for HMMs with Gaussian observation distribution. Both of these require $O(n^4 + n^3 \times m)$ operations at every time step where n is the number of hidden states and m is the number of observations. In comparison to this, I have proposed a version of the Moment Matching algorithm that only requires $O(n)$ operations at every time step. Both Mongillo and Cappe provide a limited analysis of their work using synthetic data with $n = 2$.

Foti et. al [28] have also derived an Online Variational Bayes algorithm for learning the parameters of an HMM with discrete states and Gaussian observation distribution. In order to break the dependencies in the chain, they consider mini-batches of observations which seems natural in Bayes nets where the exchangeability assumption holds, however for HMMs it creates edge cases and breaks dependencies between consecutive hidden states at the start and end of a mini-batch. In order to tackle this, Foti, et. al. buffer some observations before and after the mini-batch and make the assumption that the sub-chain is representative of the full time series chain. Their algorithm harnesses the memory decay of the chain to adaptively bound errors arising from edge effects. They have demonstrate the effectiveness of their algorithm on synthetic experiments and a large genomics dataset. In their implementation, they assume a Gaussian observation distribution which allows them to make efficient updates at every time step.

3.3 Assumed Density Filtering

The moment matching algorithm described in this thesis is very close to Assumed Density Filtering. In Assumed Density Filtering, KL divergence between an approximate posterior and the exact one is minimized [55]. This leads to matching the natural moments of the distribution. Alamino et. al [5] have explored this approach to learn the parameters of a discrete HMM with Dirichlet priors. For Dirichlet priors, matching the natural moments involves solving multiple systems of non linear equations which is a computationally intensive process. The authors have suggested matching the first and second order moments which can be done by solving a linear system of equations that have an analytical solution. However, they do not provide any explanation for why this is a good idea and how this will affect the convergence properties of the algorithm. In addition to that, they do not handle the state switching problem that we have described previously. In this thesis, I provide an alternative argument for matching the first and second order moments. Matching the first and second order moments allows us to define higher order moments in terms of an implicit prior such that if we did exact inference with respect to the implicit prior, then the value first order moment of the posterior will be the same as the one acquired by starting with a different prior and doing moment matching to compute the posterior. In addition to that, each new observation can be incorporated in time linear in the number of hidden states which has not been discussed before in terms of ADF. I also provide an explicit treatment for state switching that is ignored in ADF.

Minka et. al.[55],[56] have proposed an algorithm called Expectation Propagation where they use an iterative technique that repeatedly approximates the posterior over all hidden variables with a simple distribution by matching a few moments of the posterior for inference. This step is then embedded in an EM algorithm for learning. The convergence properties of this approach are not well understood and when convergence arises it is very slow. In practice the algorithm is very slow and does not scale well to large data sets for the case of LDA.

All of the above techniques have been used with success in practice, however they are each approximate and further approximations must be done to obtain online algorithms that perform a single scan of the data. In this thesis, I present an online Bayesian moment matching (BMM) technique which is simple to implement and it outperforms existing techniques both in terms of accuracy and time and is exact with respect to an implicit prior. In the next chapter I will describe the motivation and framework for Bayesian Moment Matching.

Chapter 4

Online Bayesian Learning Using Moment Matching

In this chapter I describe a technique for Bayesian learning called Bayesian Moment Matching (BMM) which is based on Assumed Density Filtering (ADF). The main idea is to update the posterior exactly after each observation and then approximate this exact posterior with fewer terms in order to prevent the number of terms from growing exponentially. As we will see in later chapters this is a reasonable approximation since there exists a set of initial distributions (different from the prior) for which exact inference yields the same moments as Bayesian Moment Matching.

4.1 Moments

In Mathematics, a moment is, loosely speaking, a quantitative measure of the shape of a distribution. The first raw moment, or simply the first moment, is referred to as the distribution's mean. For this work, we use an extended definition of moments. Let $f(x;\alpha)$ be a distribution over an N-dimensional random variable $x = \{x_1, \ldots, x_N\}$ and $\mu(x)$ be a function of x Then the moment $M_\mu(f)$ is the expectation of $\mu_j(x)$ with respect to f:

$$M_{\mu_j}(f) = \int_x \mu_j(x) f(x) dx \tag{4.1}$$

In the rest of the thesis, we shall use μ_j instead of $\mu_j(x)$ as a short hand. If μ is a monomial of degree j of the form $\mu_j(x) = \prod_i x_i^{n_i}$ such that $\sum_i n_i = j$, then we call it a j^{th} order moment of the distribution. For example if f is the Beta distribution, $f(x) = Beta(x; \alpha_1, \alpha_2) = k\theta^{\alpha_1} (1-\theta)^{\alpha_2}$, its order 1 moment (also called the mean) and its order 2 moment can be calculated using

$$M_x\,(Beta) = \frac{\alpha_1}{\alpha_1 + \alpha_2} \tag{4.2}$$

$$M_{x^2}\,(Beta) = \frac{\alpha_1}{\alpha_1 + \alpha_2}\frac{\alpha_1 + 1}{\alpha_1 + \alpha_2 + 1} \tag{4.3}$$

Another interesting moment $M_{\log(x)}$ can be calculated as

$$M_{\log(x)}\,(Beta) = \psi\,(\alpha_1) - \psi\,(\alpha_1 + \alpha_2) \tag{4.4}$$

Where ψ is called the digamma function.

4.1.1 Sufficient Set of Moments

For some distributions f, there exists an alternative parametrization of the distribution based on a set of sufficient moments. For example for the Beta distribution, it is easy to set up a linear system of equations using M_x and M_{x^2} to calculate α_1 and α_2. Making this definition more concrete

Definition 4.1.1. For some distributions f there exists a set of monomials $S(f)$ such that for $\forall\mu \in S(f)$, knowing $M_\mu(f)$ allows us to calculate the parameters of f.

For example for the Beta distribution $S(f) = \{x, x^2\}$. Note that $S\,(f)$ is not unique. There may exist multiple sufficient sets for a particular distribution. For example, in case of the Beta distribution, $S(f) = \{\log(x), \log(1-x)\}$ is also sufficient to calculate its parameters.

4.2 Moment Matching

In this section, we describe a moment matching technique for Bayesian learning. The algorithm approximates the posterior after each observation with fewer terms in order to prevent the number of terms to grow exponentially. Let the prior $P_0\,(\Theta) = f\,(\Theta|\alpha)$ be such that the sufficient set of moments of f exists and is denoted by $S\,(f)$. Then the posterior over the parameters of interest Θ after seeing t observations is denoted by $P_t\,(\Theta)$.

Alg. 1 describes a generic procedure to approximate the posterior P_t after each observation with a simpler distribution Q_t by moment matching. More precisely, a set of moments sufficient to define Q_t are matched to the moments of the exact posterior P_t. In Line 4, we calculate the exact posterior $P_t(\Theta)$ based on the t^{th} observed data point. Then, we compute some moments of P_t in Line 5. Specifically, for each $\mu \in S(f)$, we calculate the moments $M_\mu(P_t)$ of the posterior exactly. In line 6 we compute the parameters α of a distribution Q that belongs to the same family as $f(\Theta|\alpha)$ based on the set of sufficient moments. This determines a specific distribution Q_t in the family f that we use to approximate P_t in line 7. Note that the moments in the sufficient set $S(f)$ of the approximate posterior are the same as those of the exact posterior. However, the moments outside of the sufficient set are not necessarily the same. Although the presence of moments that differ suggests an approximation, we will show in the next chapter that in the case of LDA and HMMs, there exists a set of priors for which exact inference would arrive at the same moments at every step.

Algorithm 1 genericMomentMatching

```
1: Let $f(\Theta|\alpha)$ be a family of distributions with parameters $\alpha$
2: Initialize the prior $P_0(\Theta)$
3: for $t = 1$ to $T$ do
4:     Compute $P_t(\Theta)$ from $P_{t-1}(\Theta)$ exactly
5:     For $\forall \mu \in S(f)$, compute $M_\mu(P_t)$
6:     Compute $\alpha_t$ from the $M_\mu(P_t)$'s
7:     Approximate $P_t$ with $Q_t(\Theta) = f(\Theta|\alpha_t)$
8: end for
```

Family of Approximating Distributions

In theory f could belong to any family of distributions for which we can calculate a sufficient set of moments. A convenient choice of prior is usually the conjugate prior for the likelihood function. If the likelihood is an n-dimensional multinomial distribution of the form

$$\text{Mult}(x_1, \ldots, x_n; \theta_1, \ldots, \theta_n) = k \prod_i \theta_i^{x_i} \tag{4.5}$$

then a convenient choice for the conjugate prior is the family of Dirichlet distributions

$$\text{Dir}(\theta_1, \ldots, \theta_n; \alpha_1, \ldots, \alpha_n) = k \prod_i \theta_i^{\alpha_i}, \quad k = \frac{\Gamma(\sum_t \alpha_t)}{\prod_i \Gamma(\alpha_i)} \tag{4.6}$$

Elements of the family of Dirichlet distribution can be parametrized by the hypercounts α. Given the values of α_i's, we can calculate some important moments of the Dirichlet using the following equations

$$M_{\prod_i \theta_i^{n_i}}(f) = \int_\theta \prod_i \theta_i^{n_i} f(\theta) d\theta = \frac{\Gamma(\sum_i \alpha_i)}{\Gamma(\sum_i \alpha_i + n_i)} \prod_i \frac{\Gamma(\alpha_i + n_i - 1)}{\Gamma(\alpha_i)} \tag{4.7}$$

$$M_{\log(\theta_i)}(f) = \int_\theta \log(\theta_i) f(\theta) d\theta = \psi(\alpha_i) - \psi\left(\sum_i \alpha_i\right) \tag{4.8}$$

4.2.1 Moment Matching for the Dirichlet Distribution

In our moment matching algorithm, the prior lies in the family of Dirichlet distributions. After seeing the first observation, Algorithm 1 will update the exact posterior. Depending on the form of the likelihood function, the posterior may not lie in the family of Dirichlet distributions. We then project the posterior back into the family of Dirichlet distributions to keep the computation bounded. Approximating a mixture of Dirichlets with a single Dirichlet at each step is a reasonable thing to do for two reasons. First, since exact Bayesian learning is consistent in this setting, it will converge in the limit (i.e., infinite amount of data) to a Dirac distribution at the true underlying parameters. Hence, it is reasonable to approximate a multimodal mixture of Dirichlets with a single unimodal Dirichlet since the exact posterior becomes unimodal in the limit. Second, we will later show that for a certain sufficient set of moments, the moments calculated by BMM at each step are exact with respect to a set of initial distributions. We are deliberately using the expression *initial distributions* instead of priors since this set of initial distributions can only be determined after seeing some of the data and therefore the initial distributions are not priors in the Bayesian sense. We will now describe some sufficient sets of moments for the family of Dirichlet distributions and discuss the pros and cons of using them for moment matching.

Moment Matching by Minimizing KL Divergence

When we project the posterior to the family of Dirichlet distributions, we want to make sure that the projection is as close to the true posterior as possible. Here we describe a sufficient set of moments for the Dirichlets such that if we match those moments, the KL divergence between the Dirichlet and the distribution to be approximated is minimized.

Definition 4.2.1. The exponential family of distributions is a class of distributions sharing a certain form given by $f(\theta_1, .., \theta_n | \alpha_1, .., \alpha_n) = h(\theta) \exp\{\sum_i \eta_i(\alpha) . T_i(\theta) - A(\alpha)\}$. $T(\theta)$ is also called the sufficient statistic.

We can represent the Dirichlet distribution as a member of the exponential family by setting $\eta_i(\alpha) = \alpha_i$, $h(\theta) = 1$, $T_i(\theta) = \log(\theta_i)$

$$\mathrm{Dir}(\theta_1, \ldots, \theta_n; \alpha_1, \ldots, \alpha_n) = \exp\left\{\sum_i \alpha_i \log(\theta_i) + \log(k)\right\} \tag{4.9}$$

In [38], Herbrich shows that for any distribution Q that is a member of the exponential family of distributions, the distribution P which minimises the Kullback-Leibler divergence, $KL(Q||P)$ is implicitly given by

$$\int T_i(\theta)\, Q(\theta|\alpha)\, d\theta = \int T_i(\theta)\, P(\theta)\, d\theta \tag{4.10}$$

For the Dirichlet distribution, this expression becomes

$$M_{\log(\theta_i)}(Dir) = M_{\log(\theta_i)}(P) \tag{4.11}$$

This implicitly gives us a set of moments $S(f)\{\log(\theta_i); 1 \leq i \leq n\}$ such that if we project a distribution P onto the family of Dirichlet distributions by calculating the moments in $S(f)$ with respect to P and then set them equal to those of Q which is a Dirichlet, then the KL divergence between the Q and P will be minimized. However it still remains to be shown that this set is sufficient to calculate the parameters α_is of the Dirichlet distribution. To this end, Alamino et. al. [5] use equations 4.8 and 4.11

$$\alpha_i = \psi^{-1}\left(M_{\log(\theta_i)}(P) + \psi\left(\sum_i \alpha_i\right)\right) \tag{4.12}$$

Summing this equation for all i, we get

$$\sum_i \alpha_i = \sum_i \psi^{-1}\left(M_{\log(\theta_i)}(P) + \psi\left(\sum_i \alpha_i\right)\right) \tag{4.13}$$

This gives us an iterative procedure to compute $\alpha_0 = \sum_i \alpha_i$

$$\alpha_0^{t+1} = \sum_i \psi^{-1}\left(M_{\log(\theta_i)}(P) + \psi\left(\alpha_0^t\right)\right) \tag{4.14}$$

This value can be substituted back in equation 4.12 to compute each α_i.

Matching First and Second Order Moments

Given that the observation distribution is informative, after seeing a very large number of observations, most of the weight of the posterior will be concentrated at one point which will correspond to the mean of the posterior at that time step. Therefore, another way of projecting this posterior to the manifold of Dirichlets is that instead of minimizing KL divergence, we want a projection such that the mean of the approximate posterior is as close as possible to the mean of the exact posterior. In order to do this we use a different set of sufficient moments namely $S(f) = \{\theta_i; 1 \leq i \leq n, \theta_1^2\}$ [1]. We can use Eq. 4.7 to set up the following system of equations and solve it analytically to calculate the hyperparameters of the approximate Dirichlet distribution. We can determine the parameters α_t of a Dirichlet based on those moments by setting up a system of equations.

$$M_{\theta_i}(f) = \frac{\alpha_i}{\sum_t \alpha_t} \quad \forall i \in \{1, \ldots, n-1\}$$
$$M_{\theta_1^2}(f) = \frac{\alpha_1(\alpha_1+1)}{(\sum_t \alpha_t)(\sum_t \alpha_t + 1)}$$

This gives us n equations in n variables that can be solved analytically as follows

$$\alpha_i = M_{\theta_i}(f) \frac{M_{\theta_1}(f) - M_{\theta_1^2}(f)}{M_{\theta_1^2}(f) - (M_{\theta_1}(f))^2} \quad \forall t \tag{4.15}$$

Note that we can use any second order moment instead of θ_1^2 in order to solve this system of equations analytically.

4.3 LDA with known Observation Distribution

To illustrate our idea further, let's consider a special case of the LDA model described in 2.4. There are 2 topics and the number of words in the vocabulary is 2. We assume that the observation model parameters ϕ are known and there is a single document for which we are estimating the probability of topic 1 which is equal to θ. We use the family of Dirichlets to approximate the posterior after each observed word. The exact posterior P_n after observing w_n is

$$P_n(\theta) = (1/k_n)\left(\theta\phi_{1,w_n} + (1-\theta)\,\phi_{2,w_n}\right) P_{n-1}(\theta) \tag{4.16}$$

[1] We can't use all T first-order moments since the last moment is one minus the sum of the other moments, which will lead to a redundant linearly dependent equation.

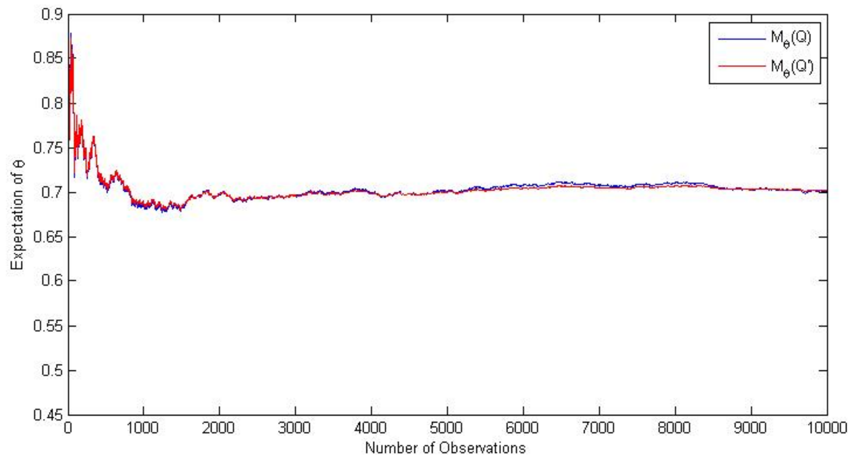
At every time step we can approximate this posterior by a Dirichlets either by matching the $M_{\log(\theta)}$ or by matching M_θ. Let the approximation constructed with respect to $M_{\log(\theta)}$ be denoted by $Q(\theta)$ and the approximation with respect to M_θ be denoted by $Q'(\theta)$. In Figure 4.1, we plot the expected value of θ for Q and Q' vs. the number of observations used for learning with a randomly generated sequence. The graph shows that $Q(\theta)$ is very close to $Q'(\theta)$ in terms of its mean.

In case of computing the new hypercounts by matching M_θ and M_{θ^2}, we have to compute 2 moments at every time step. Each of these moments can be computed by summing up 2 terms. Therefore the complexity of computing the posterior after every observation is constant and is $O(4)$. If the total length of the sequence is T, then the total complexity becomes $O(4T)$. In contrast if we do moment matching by matching $M_{log(\theta)}$, we have to solve the iterative equation 4.14. We are not aware of any convergence results for solving this iterative equation. In practise it tends to converge slowly. Also at each iteration, we have to compute ψ^{-1} and ψ which is time consuming. Let the process of solving equation 4.14 be $O(L_i)$ for time step i. Then the total computation becomes $O(\sum_{i=1}^{T} L_i)$. In our simple example, it takes 56 seconds to process all observations if we match M_θ whereas it takes about 30 minutes if we match $M_{\log(\theta)}$.

In light of the previous discussion, we can see that even though we have guarantees of minimizing KL divergence if we do moment matching by matching $M_{\log(\theta)}$, in practise, it tends to be slow and the computation at every time step is not constant. Also in practise, the resulting posterior obtained by matching $M_{\log(\theta)}$ is very close in expectation to the posterior obtained if we do moment matching using M_θ. In this simplified version of the LDA model, we only learn the document-topic distribution. Later when we will learn the word-topic distribution as well, we encounter the label switching problem. In the next chapter we will discuss this problem in detail and show that if we do moment matching with respect to $M_{\log(\theta)}$, is that we are unable to deal with the label switching.

In this chapter I have described a moment matching algorithm BMM for online Bayesian Learning such that the computation at each step is computationally bounded. I have provided the motivation for why the Dirichlet prior is suitable for this algorithm and how we can compute the projections of distributions onto a family of Dirichlet distributions. In later chapters, this technique will be applied on two real world problems i.e. topic modeling and activity recognition.

Figure 4.1: Expectation of θ with respect to the approximate posterior vs. the number of observations. $Q(\theta)$ is the approximate posterior learnt by matching $M_{\log(\theta)}$ where as $Q'(\theta)$ is the approximate posterior learnt by matching $M_{(\theta)}$. The true value of θ used to generate the observations is 0.7.



Chapter 5

Topic Modeling

In this chapter, I have described an algorithm for topic modeling based on the Bayesian Moment Matching algorithm discussed in Chapter 4. The main contributions in this chapter are

- Derive an algorithm for Bayesian learning of topic models using moment matching called Bayesian Moment Matching (BMM)
- Show that there exists an initial distribution (different from the prior) for which exact Bayesian Learning yields the same first order moment in the posterior as BMM
- Show how to handle state switching in BMM
- Derive an efficient version of the BMM where the posterior is computed in time linear in the number of topics at each time step
- Compare the performance of BMM with other state of the art algorithms used for topic modeling on synthetic and real data from twitter feeds

5.1 Latent Dirichlet Allocation

Recall the LDA model described in Section 2.4. In LDA, each word in each document is associated with a hidden topic. Let the total number of topics be T, the total number of documents be D and the total words in the vocabulary be W. The number of words in document d is N_d. The hidden topic of the n^{th} word is denoted by t_n. Then the model is is parametrized by the following distributions

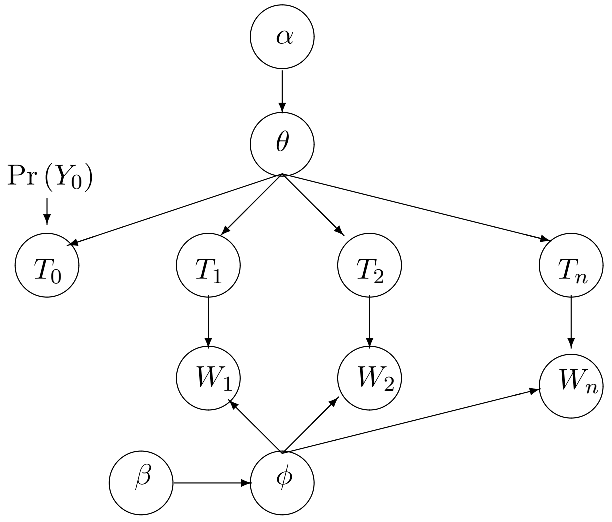


Figure 5.1: Latent Dirichlet Allocation Topic Model

- **Word-Topic Distribution** Each document d is represented by a multinomial topic distribution $\theta_d = \{\theta_{d,1}, ..., \theta_{d,T}\}$ where $\theta_{d,t}$ is the probability of topic t in document d.
- **Topic-Topic Distribution** The distribution over words given a particular topic t is called the Topic-Word distribution and is denoted by $\phi_t = \{\phi_{t,1}, \ldots, \phi_{t,W}\}$, where W is the total number of words in the dictionary.

Each θ_d is sampled from a prior $f(\theta_d; \alpha)$. For a corpus, each ϕ_t is sampled from another prior $g(\phi_t; \beta)$. The n^{th} word in document d can be generated by first sampling a topic t from θ_d and then sampling a word w_n from ϕ_t. The following terminology is reintroduced

$\Theta = \{\theta_1, \ldots, \theta_D\}$

$\Phi = \{\phi_1, \ldots, \phi_T\}$.

$w_{a:b} = w_a, w_{a+1} \ldots w_b$ a sequence of words from time a to b

$t_{a:b} = t_a, t_{a+1} \ldots t_b$ a sequence of hidden states from time a to b

$P_n(\Theta, \Phi) = \Pr(\Theta, \Phi | w_{1:n})$

$k_n = \Pr(w_n | w_{1:n-1})$

Recall that the exact posterior for LDA after seeing n words can be calculated using the following equation

$$P_n(\Theta, \Phi) = \Pr(\Theta, \Phi | w_{1:n}) = \frac{1}{k_n} \sum_t \theta_{d,t} \phi_{t,w_n} P_{n-1}(\Theta, \Phi) \tag{5.1}$$

$$k_n = \sum_t \int_\Theta \int_\Phi \theta_{d,t} \phi_{t,w_n} P_{n-1}(\Theta, \Phi) d\Theta d\Phi \tag{5.2}$$

Here $\theta_{d,t}$ is the probability that the document d is about topic t and $\phi_{t,w}$ is the probability that word w is related to topic t. Any moment $M_\mu(P_n)$ can be calculated using

$$M_\mu(P_n) = \frac{\sum_t M_{\mu\,\theta_{d,t}\phi_{t,w_n}}(P_{n-1})}{\sum_t M_{\theta_{d,t}\phi_{t,w_n}}(P_{n-1})} \tag{5.3}$$

Here $M_{\mu\,\theta_{d,t}\phi_{t,w_n}}(P_{n-1})$ denotes the moment associated with the monomial $\mu\,\theta_{d,t}\phi_{t,w_n}$ obtained by multiplying $\mu(\Theta,\Phi)$ by $\theta_{d,t}$ and ϕ_{t,w_n}.

Algorithm. 2 is the adaptation of Algorithm. 1 to the LDA model by specifying a set of equations to compute the exact posterior at each time step in line 4. It describes a generic procedure to approximate the posterior P_n after seeing each new word with a simpler distribution Q_n by moment matching. A set of moments sufficient to define Q_n are matched to the moments of the exact posterior P_n. Next, we compute the parameters α and β based on the set of sufficient moments. This determines a specific distribution Q_n in the family f that was used to approximate P_n.

Algorithm 2 genericMomentMatching

```
1: Let $f(\Theta,\Phi|\alpha,\beta)$ be a family of distributions with parameters $\alpha$ and $\beta$
2: Initialize the prior $P_0(\Theta,\Phi)$
3: for $n = 1$ to $N$ do
4:     Compute $P_n(\Theta,\Phi)$ from $P_{n-1}(\Theta,\Phi)$ (Eq. 5.1)
5:     For $\forall\mu \in S(f)$, compute $M_\mu(P_n)$
6:     Compute $\alpha$ and $\beta$ from the $M_\mu(P_n)$'s
7:     Approximate $P_n$ with $Q_n(\Theta,\Phi) = f(\Theta,\Phi|\alpha,\beta)$
8: end for
```

5.2 Known Topic-Word Distribution Case

Let's illustrate Alg. 2 with a simplified version of the problem in which the topic-word distributions Φ are known. In this case, the topic distribution θ_d of each document can be estimated separately. So I focus on the estimation of the topic distribution for a single document. To simplify the notation, the index d is dropped and this topic distribution is generically denoted by θ. f can be chosen to be any distribution in the family of Dirichlets $f(\theta) = Dir(\theta;\alpha)$. Any moment of the posterior can be calculated using

$$M_\mu(P_n) = (1/k_n)\sum_t \phi_{t,w_n} M_{\mu\,\theta_t}(P_{n-1}) \tag{5.4}$$

Note that this equation is different from Eq. 5.3 because Φ is known. We calculate the sufficient set of moments of the posterior $S(f)\,\{\log(\theta_i); 1 \leq i \leq n\}$. For $j = \{1, \ldots, T-1\}$

$$M_{log(\theta_i)}(P_n) = (1/k_n) \sum_t \phi_{t,w_n} \frac{\alpha_t}{\sum_{t'} \alpha_{t'}} \left(\psi\left(\alpha_t + \delta(t,i)\right) - \psi\left(\sum_{t'} \alpha_{t'} + 1 \right) \right) \tag{5.5}$$

Eq. 4.14 can then be used to determine the α_t's of the approximating distribution $Q_n(\theta) = f(\theta; \alpha)$. Let the computation complexity of solving equation Eq. 4.14 iteratively be $O(L)$, then the computational complexity of the moment matching process is $O(NLT^2)$ as there are N iterations in which each word is processed by updating the hyper-parameters of the approximating Dirichlets in $O(LT^2)$ time. In practice, however, L tends to be large and the convergence of the iterative process is very slow.

An alternate $S(f)$ can also be used as described in the previous section where $S(f) = \{\theta_i; 1 \leq i \leq n, \theta_1^2\}$. For each i

$$\begin{aligned} M_{\theta_i}(P_n) &= (1/k_n) \sum_t \phi_{t,w_n} M_{\theta_i \theta_t}(P_{n-1}) && (5.6) \\ &= (1/k_n) \sum_t \phi_{t,w_n} \frac{\alpha_t}{\sum_{t'} \alpha_{t'}} \frac{\alpha_t + \delta(t,i)}{\sum_{t'} \alpha_{t'} + 1} \\ M_{\theta_1^2}(P_n) &= (1/k_n) \sum_t \phi_{t,w_n} M_{\theta_1^2 \theta_t}(P_{n-1}) && (5.7) \\ &= (1/k_n) \sum_t \phi_{t,w_n} \frac{\alpha_t}{\sum_{t'} \alpha_{t'}} \frac{\alpha_t + \delta(t,1)}{\sum_{t'} \alpha_{t'} + 1} \frac{\alpha_t + \delta(t,1) + 1}{\sum_{t'} \alpha_{t'} + 2} \end{aligned}$$

We can then use Eq. 4.15 to determine the α_t's of our approximating distribution $Q_n(\theta) = f(\theta; \alpha)$. This process is summarized in Alg. 3. The computational complexity of Alg. 3 is $O(NT^2)$. This is an online algorithm since the data is processed in a single sweep, the amount of computation to process each word is independent of the amount of data, and we obtain an estimate of the parameters (in the form of an approximate posterior) after processing each word.

5.2.1 Analysis

In this section, I will show that the moments calculated by BMM using $S(f) = \{\theta_i; 1 \leq i \leq n, \theta_1^2\}$ are exact with respect to a set of initial distributions. I am deliberately using

Algorithm 3 momentMatching (known Φ)

```
1: Let $f(\theta|\alpha)$ be the family of Dirichlets
2: Initialize $P_0(\theta) = Dir(\theta; \alpha)$
3: for $n = 1$ to $N$ do
4:     Compute $P_n(\theta)$ from $P_{n-1}(\theta)$
5:     $\forall t \in \{1, \ldots, T-1\}$, compute $M_{\theta_t}(P_n)$ (Eq. ??)
6:     Compute $M_{\theta_1^2}(P_n)$ using Eq. ??
7:     $\forall t \in \{1, \ldots, T\}$ compute $\bar{\alpha}_t$ using Eq. 4.15
8:     Approximate $P_n$ with $Q_n(\theta) = Dir(\theta; \bar{\alpha})$
9: end for
```

the expression *initial distributions* instead of priors since this set of initial distributions can only be determined after seeing some of the data and therefore the initial distributions are not priors in the Bayesian sense. Nevertheless, this analysis is helpful to understand the nature of the approximation performed by BMM.

Let's rewrite Alg. 3 purely in terms of moment computation as done in Alg. 4. This will be useful to show that there exists a set of initial distributions for which exact inference yields the same moments. In order to compute the sufficient moments, Eq. 5.4 tells us that we need only the following moments of P_{n-1}:

$M_{\theta_t}(P_{n-1}); 1 \leq t \leq T$

$M_{\theta_t \theta_j}(P_{n-1}); 1 \leq t \leq T \;\; t \leq j \leq T$

$M_{\theta_t \theta_1^2}(P_{n-1}); 1 \leq t \leq T$

Let us denote this set of moments by $Prev(f)$. At every iteration of Alg. 4, we first compute the moments of P_n in $S(f)$ based on the moments of P_{n-1} in $Prev(f)$ (Line 4). Then we compute the moments of P_n in $Prev(f) - S(f)$ based on the moments of P_n in $Prev(f)$ (Line 5). Hence, at every iteration we compute exactly the moments of the posterior that are in $S(f)$ and set additional moments (those that are in $Prev(f) - S(f)$), which allows the process to keep on going.

Let's now show that Alg. 4 (and Alg. 3) performs exact inference with respect to an implicit set of initial distributions. The key is to realize that we start with a partially defined P_0. In Alg. 4, we initialize only the moments of P_0 that are in $Prev(f)$. The other moments are undefined and therefore we really have a set of initial distributions corresponding to all distributions that share the same moments in $Prev(f)$. Then, at each iteration, we compute the moments of P_n in $S(f)$ exactly. When we set the moments of P_n in $Prev(f) - S(f)$, we are implicitly constraining the set of initial distributions by setting

Algorithm 4 momentMatching equivalent to Alg. 3

```
1: Let $f(\theta|\alpha)$ be the family of Dirichlets
2: Set $M(P_0), M_{\theta_t}(P_0), M_{\theta_t\theta_j}(P_0), M_{\theta_t\theta_1^2}(P_0)\ \forall tj \leq T$
3: for $n = 1$ to $N$ do
4:     $\forall \mu \in S(f)$ compute $M_\mu(P_n)$ based on all $M_{\mu'}(P_{n-1})$ s.t. $\mu' \in Prev(f)$ (Eq. 5.4)
5:     $\forall \mu \in Prev(f) - S(f)$ compute $M_\mu(P_n)$ based on all $M_{\mu'}(P_n)$ s.t. $\mu' \in S(f)$
       (Eq. 4.7, 4.15)
6: end for
```

additional moments of P_0.

Consider the case where $T = 2$ (i.e., there is only two topics). In this case, the family f of distributions corresponds to the family of Beta distributions, $S(f) = \{\theta_1, \theta_1^2\}$ and $Prev(f) = \{\theta_1, \theta_1^2, \theta_1^3\}$. In the start, we initialize $M_{\theta_1}(P_0)$, $M_{\theta_1^2}(P_0)$ and $M_{\theta_1^3}(P_0)$. The other moments of P_0 are undefined. Hence, there is a set of distributions that are consistent with those moments and any of them could be the initial distribution. At the first iteration $M_{\theta_1}(P_1)$ and $M_{\theta_1^2}(P_1)$ are computed exactly according to Eq. 5.4. This gives us a set of posteriors that share the same first and second order moments and are consistent with the set of initial distributions. Next, $M_{\theta_1^3}(P_1)$ is computed according to Eqs 4.7 and 4.15. One view is that this is an approximation. However, we can also ask whether we could have obtained the same moment by exact inference from some of the initial distributions in our set. If we do exact inference, then $M_{\theta_1^3}(P_1)$ must be computed according to Eq. 5.3, which gives:

$$M_{\theta_1^3}(P_1) = \frac{\phi_{1,w_n} M_{\theta_1^4}(P_0) + \phi_{2,w_n}(M_{\theta_1^3}(P_0) - M_{\theta_1^4}(P_0))}{k_1}$$

This equation shows that $M_{\theta_1^3}(P_1)$ is really a function of $M_{\theta_1^3}(P_0)$ and $M_{\theta_1^4}(P_0)$. Since $M_{\theta_1^4}(P_0)$ is unspecified, it could be set to ensure that exact inference yields the same $M_{\theta_1^3}(P_1)$. In other words, setting $M_{\theta_1^3}(P_1)$ according to Eqs 4.7 and 4.15 is equivalent to setting $M_{\theta_1^4}(P_0)$ and then arriving at the same $M_{\theta_1^3}(P_1)$ by exact inference according to Eq. 5.4. Similarly, at the second iteration, when $M_{\theta_1^3}(P_2)$ is set, there exists a value for $M_{\theta_1^5}(P_0)$ that ensures that exact inference yields precisely $M_{\theta_1^3}(P_2)$. This follows from the fact that $M_{\theta_1^3}(P_2)$ depends on $M_{\theta_1^4}(P_1)$, which in turn depends on $M_{\theta_1^5}(P_0)$ according to Eq. 5.4. Hence, at every iteration, when the third order moment of the posterior is set, we are really setting one more moment of P_0. Since we do one iteration per word in the dataset and there is a finite amount of data, this process will specify a finite number of moments for P_0, which means that the computation is exact with respect to an implicit set of initial distributions that share those moments.

We can extend this analysis to the case where $T > 2$ and f is the family of Dirichlets. At each iteration, we calculate moments in $S(f)$ exactly and we set the moments in $Prev(f) - S(f)$. Let's have a closer look at the relationship between the third order moments $M_{\mu_3}(P_0)$ and the second order moments $M_{\mu_2}(P_1)$. According to Eq. 5.4, any second order moment $M_{\mu_2}(P_1)$ is a sum of T third order moments of P_0. Therefore, whenever a second order moment is set at iteration n, we can set up a system of equations to compute the moments at iteration $n-1$. For example, we can compute $M_{\theta_1\theta_2}(P_1)$ using the parameters we calculated exactly for P_1 by moment matching. Then using Eq. 5.4, we can setup a system of equations as follows:

$$M_{\theta_1\theta_2}(P_n) = (1/k_n)\sum_t \phi_{t,w_n} M_{\theta_t\theta_1\theta_2}(P_{n-1})$$

For a given iteration n and moment order j, this system of equations is underdetermined because the number of moments of order j is larger than the number of moments of order $j-1$. This underdetermined system has more than one solution that can be used to set higher order moments of P_{n-1}. In general, we can setup a system of equations based on Eq. 5.3 that relates the moments of P_n we set in $Prev(f) - S(f)$ to higher order moments of P_0 that are still undefined. Hence, the process of setting moments in $Prev(f) - S(f)$ at each iteration is equivalent to setting some higher order moments of P_0 such that inference is exact with respect to the initial distributions that satisfy those moments. We formalize this in a Theorem.

Theorem 5.2.1. *Under the assumption that the likelihood of each word* $\Pr(w|\theta)$ *is non-zero, the moment matching procedure described in Alg. 4 (and Alg. 3) performs exact inference with respect to at least one initial distribution.*

Proof. First I will prove that there exists a function P_0 such that exact inference from this function yields the same moments as those computed by BMM at each step. Then I will prove that P_0 is a valid distribution.

In the case of a T-dimensional Dirichlet, $S(f)$ contains T moments, $Prev(f)$ contains $(T-1)(T/2+2)$ and therefore $Prev(f) - S(f)$ contains $(k-1)(T/2+2) - T$ moments. Since at each iteration, BMM sets the moments in $Prev(f) - S(f)$, it will set a total of $N[(T-1)(T/2+2) - T]$ moments after N iterations. These moments are related to the moments of order 1 to $N+3$ in P_0 via a system of equations obtained by composing Eq. 4.7 with itself N times. This system has $N[(T-1)(T/2+2) - T]$ equations corresponding to the moments that are set. The system also has a number of variables corresponding to the moments of order 1 to $N+3$ that were not initially set. Since the number of variables is at least $N[(T-1)(T/2+2) - T]$, the system of equations has a solution and there exists a function P_0 that has moments consistent with the moments set by BMM at each iteration.

We show that P_0 is a valid distribution by showing that P_0 is non-negative in its domain and it integrates to 1. P_0 necessarily integrates to 1 since we select $M_0(P_0)$ to be 1 when we select the first few moments of P_0 at the beginning of Alg. 4. Since P_n is obtained from P_0 by exact inference then P_0 satisfies Bayes' theorem.

$$P_n(\theta) \propto P_0(\theta) \Pr(w_{1:n}|\theta)$$

$$P_n(\theta) = \frac{P_0(\theta) \Pr(w_{1:n}|\theta)}{\int_\theta P_0(\theta) \Pr(w_{1:n}|\theta) d\theta}$$

Since $P_n(\theta)$ is non-negative and $\Pr(w_{1:n}|\theta)$ is strictly greater than 0 (by assumption), then there exists a non-negative P_0 that satisfies Bayes' theorem. □

5.3 Learning the Word-Topic Distribution

In this section, I will consider the general case where the document-topic and topic-word distributions are all unknown. A natural choice for the family of approximating distributions is a product of Dirichlets of the form $f(\Theta, \Phi) = \prod_d Dir(\theta_d; \alpha_d) \prod_t Dir(\phi_t; \beta_t)$. The first problem that we encounter is label switching and unidentifiability.

5.3.1 Label Switching and Unidentifiability

Since the topic assigned to each word is abstract and hidden, we can permute the labels assigned to each topic and get a new set of parameters Θ and Φ. The model corresponding to this new set of parameters, can generate the document corpus $w_{1:n}$ with the same probability as the first permutation. Therefore, there will be $T!$ components in the posterior P_n, each corresponding to a permutation of the hidden topics. In the limit (i.e., infinite amount of data), the posterior will have $T!$ modes and therefore a unimodal approximation with a product of Dirichlets will not work well. In order to cater for each mode that is expected to arise from each permutation, a mixture of products of Dirichlets is considered. The parameters of each product of Dirichlets are permuted according to the topic labels assigned to that permutation. Let Σ^T be all permutations of the vector $1, \ldots, T$. Then f becomes

$$f(\Theta, \Phi) = \frac{1}{T!} \sum_{\sigma \in \Sigma^T} f^\sigma(\Theta, \Phi) \tag{5.8}$$

Here σ is a permutation of the vector $1,\ldots,T$ and $\sigma(t)$ is the label of the topic t in that permutation. f^{σ} is a product of Dirichlets corresponding to the permutation σ. For example, for $T=2$, $W=3$ and $D=1$ there are two possible permutations of topics.

$$\begin{aligned}f\left(\Theta,\Phi\right) = 1/2 Dir\left(\theta_{1,1},\theta_{1,2};\alpha_{1,1},\alpha_{1,2}\right) Dir\left(\phi_{1,1},\phi_{1,2},\phi_{1,3};\beta_{1,1},\beta_{1,2},\beta_{1,3}\right)\\ Dir\left(\phi_{2,1},\phi_{2,2},\phi_{2,3};\beta_{2,1},\beta_{2,2},\beta_{2,3}\right)\\ + 1/2 Dir\left(\theta_{1,1},\theta_{1,2};\alpha_{1,2},\alpha_{1,1}\right) Dir\left(\phi_{1,1},\phi_{1,2},\phi_{1,3};\beta_{2,1},\beta_{2,2},\beta_{2,3}\right)\\ Dir\left(\phi_{2,1},\phi_{2,2},\phi_{2,3};\beta_{1,1},\beta_{1,2},\beta_{1,3}\right)\end{aligned}$$

Each permutation is assigned the weight $1/T!$ so that the mixture sums up to 1. Unfortunately, the symmetric nature of the distribution is problematic. The odd central moments of symmetric distributions are always zero. Because of this, matching all the first order moments leads to the same linear equation which is problematic as we need at least $n-1$ equations corresponding to the first order moments. Instead, we consider a family of slightly asymmetric mixtures

$$f\left(\Theta,\Phi\right) = \sum_{\sigma_i \in S^T} w_{\sigma_i} f^{\sigma_i}\left(\Theta,\Phi\right) \tag{5.9}$$

Let's assign a weight $w_{\sigma_1} = ((T-1)!+1)/(T!+(T-1)!)$ to the first component that is $(T-1)!$ higher than the weight $w_{\sigma_j} = 1/(T!+(T-1)!)\ \forall j \neq 1$ for the remaining components. This slight asymmetry ensures that all moments vary and therefore can be used in moment matching. It also preserves most of the symmetry, which will allow us to simplify moment calculations later. Furthermore, the hyperparameters of only one product of Dirichlets needs to be stored since all $T!$ components share the same (permuted) hyperparameters. The moments $M_{\theta^k_{d,t}\phi^l_{t,w}}(f^{\sigma_i})$ can be calculated using

$$M_{\theta^k_{d,t}\phi^l_{t,w}}(f^{\sigma_i}) = \frac{\Gamma(\alpha_{d,\sigma_i(t)}+k-1)}{\Gamma(\alpha_{d,\sigma_i(t)})} \frac{\Gamma(\alpha_{d,.})}{\Gamma(\alpha_{d,.}+k-1)} \frac{\Gamma(\beta_{\sigma_i(t),w})+l-1}{\Gamma(\beta_{\sigma_i(t),w})} \frac{\Gamma(\beta_{\sigma_i(t),.})}{\Gamma(\beta_{\sigma_i(t),.}+l-1)} \tag{5.10}$$

where $\alpha_{d,.} = \sum_t \alpha_{d,t}$ and $\beta_{i,.} = \sum_w \beta_{i,w}$.

5.3.2 Sufficient Moments

Since all mixture components f^{σ_i} share the same (permuted) hyperparameters, f has a total of DT topic distribution hyperparameters and TW word distribution hyperparameters.

As mentioned before, the following sufficient set of moments can be used to compute the hyperparameters

$$S(f) = \{\log(\theta_{d,t}) | 1 \leq d = D, 1 \leq t \leq T\} \cup \{\log(\phi_{t,w}), 1 \leq t \leq T, 1 \leq w \leq W\}$$

To simplify the computation of the sufficient moments, I exploit the fact that $M_{\log(\theta_{d,t})}$ $(f^{\sigma}) = M_{\log(\theta_{d,t})}(f^{\sigma'})$ for all permutations σ and σ' such that $\sigma(t) = \sigma'(t)$. There are $(T-1)!$ such permutations. The sufficient moments in θ are computed as follows:

$$M_{log(\theta_{d,t})}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} M_{log(\theta_{d,i})}(f^{\sigma_1}) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} (\psi(\alpha_{d,i}) - \psi(\alpha_{d,.})) \quad (5.11)$$

This will give us a set of non linear equations in $\psi(\alpha_i)$s which has no closed form solution. On the other hand, we can also calculate the hyperparameters using the sufficient set of moments given by

$$S(f) = \{\theta_{d,t}, \theta_{d,1}^2 | 1 \leq d = D, 1 \leq t \leq T-1\} \quad \cup \{\phi_{t,w}, \phi_{t,1}^2 | 1 \leq t \leq T, 1 \leq w \leq W-1\}$$

To simplify the computation of the sufficient moments, we exploit the fact that $M_{(\theta_{d,t}^k \phi_{t,w}^l)}$ $(f^{\sigma}) = M_{(\theta_{d,t}^k \phi_{t,w}^l)}(f^{\sigma'})$ for all permutations σ and σ' such that $\sigma(t) = \sigma'(t)$. There are $(T-1)!$ such permutations. In addition, $M_{(\theta_{d,t}^k \phi_{t,w}^l)}(f^{\sigma}) = M_{(\theta_{d,t'}^k \phi_{t',w}^l)}(f^{\sigma'})$ whenever $\sigma(t) = \sigma'(t')$. Using these two facts, the sufficient moments in θ are computed as follows:

$$M_{(\theta_{d,t}^k)}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} M_{(\theta_{d,i}^k)}(f^{\sigma_1}) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\Gamma(\alpha_{d,i}+k-1)}{\Gamma(\alpha_{d,i})} \frac{\Gamma(\alpha_{d,.})}{\Gamma(\alpha_{d,.}+k-1)} \quad (5.12)$$

Specifically for $k = 1$ and $k = 2$ we have the following equations,

$$M_{(\theta_{d,t})}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\alpha_{d,i}}{\alpha_{d,.}} = \frac{1}{T+1}\left(\frac{\alpha_{d,t}}{\alpha_{d,.}} + 1\right) \quad (5.13)$$

$$M_{(\theta_{d,1}^2)}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\alpha_{d,1}}{\alpha_{d,.}} \frac{\alpha_{d,1}+1}{\alpha_{d,.}+1} \quad (5.14)$$

Similarly, the sufficient moments in ϕ are computed as follows:

$$M_{(\phi_{t,w}^l)}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} M_{(\phi_{i,w}^l)}(f^{\sigma_1}) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\Gamma(\beta_{i,w}+k-1)}{\Gamma(\beta_{i,w})} \frac{\Gamma(\beta_{i,.})}{\Gamma(\beta_{i,.}+k-1)} \quad (5.15)$$

For $l = 1$ and $l = 2$ we have the following equations

$$M_{(\phi_{t,w})}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\beta_{i,w}}{\beta_{i,\cdot}} \tag{5.16}$$

$$M_{(\phi_{t,1}^2)}(f) = \sum_{i=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\beta_{i,1}}{\beta_{i,\cdot}} \frac{\beta_{i,1}+1}{\beta_{i,\cdot}+1} \tag{5.17}$$

5.3.3 Moment Matching

For this choice of family of distributions f the posterior P_n is obtained from P_{n-1} according to

$$P_n(\Theta, \Phi) = \frac{1}{k_n} \sum_{\sigma \in \Sigma^T} w_\sigma \sum_t \theta_{d,t} \phi_{t,w_n} P_{n-1}^{\sigma}(\Theta, \Phi) \tag{5.18}$$

$$\begin{aligned} k_n &= \sum_{\sigma \in \Sigma^T} w_\sigma \sum_t \int_\Theta \int_\Phi \theta_{d,t} \phi_{t,w_n} P_{n-1}^{\sigma}(\Theta, \Phi)\, d\Theta d\Phi \\ &= \sum_i \frac{\alpha_i}{\sum_j \alpha_j} \frac{\beta_{i,w_n}}{\sum_j \beta_{i,j}} \end{aligned} \tag{5.19}$$

The moment update equation is

$$M_\mu(P_n) = \frac{1}{k_n} \sum_{\sigma \in \Sigma^T} w_\sigma \sum_t M_{\mu \theta_{d,t} \phi_{t,w_n}}\left(P_{n-1}^{\sigma}\right) \tag{5.20}$$

The following equations allow us to compute the sufficient set of moments of the posterior P_n

$$\begin{aligned}
M_{(\theta^a_{d,t})}(P_n) &= \int_\Theta \int_\Phi \theta^a_{d,t} P_n(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{\sigma \in S^T} w^\sigma \int_\Theta \int_\Phi \theta^a_{d,t} P^\sigma_n(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{\sigma \in S^T} w^\sigma \int_\Theta \int_\Phi \theta^a_{d,t} \sum_i \theta_{d,i} \phi_{i,w_n} P^\sigma_{n-1}(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{i=1}^{T} \sum_{j=1}^{T} \frac{1+\delta(i,t)}{T+1} M_{\theta^a_{d,t}\theta_{d,j}\phi_{j,w_n}} \left(P^{\sigma_1}_{t-1}\right) \\
&= \sum_{i=1}^{T} \frac{\alpha_{d,i}\beta_{i,w_n}}{\alpha_{d,.}\beta_{i,.}} \sum_{j=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\Gamma\left(\alpha_{d,.}\right)}{\Gamma\left(\alpha_{d,.}+a+1\right)} \frac{\Gamma\left(\alpha_{d,j}+\delta(i,j)+a\right)}{\Gamma\left(\alpha_{d,j}+\delta(i,j)\right)}
\end{aligned} \tag{5.21}$$

$$\begin{aligned}
M_{(\phi^b_{t,l})}(P_n) &= \int_\Theta \int_\Phi \phi^b_{t,l} P_n(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{\sigma \in S^T} w^\sigma \int_\Theta \int_\Phi \phi^b_{t,l} P^\sigma_t(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{\sigma \in S^T} w^\sigma \int_\Theta \int_\Phi \phi^b_{t,l} \sum_i \theta_{d,i} \phi_{i,w_n} P^\sigma_{t-1}(\Theta, \Phi) d\Theta d\Phi \\
&= \sum_{i=1}^{T} \sum_{j=1}^{T} \frac{1+\delta(i,t)}{T+1} M_{\phi^b_{j,l}\theta_{d,j}\phi_{j,w_n}} \left(P^{\sigma_1}_{t-1}\right) \\
&= \sum_{i=1}^{T} \frac{\alpha_{d,i}\beta_{i,w_n}}{\alpha_{d,.}\beta_{i,.}} \sum_{j=1}^{T} \frac{1+\delta(i,t)}{T+1} \frac{\Gamma(\beta_{j,.}+\delta(i,j))}{\Gamma(\beta_{j,.}+\delta(i,j)+b)} \frac{\Gamma\left(\beta_{j,l}+\delta(i,j)\delta(l,w_n)+b\right)}{\Gamma\left(\left(\beta_{j,l}+\delta(i,j)\delta(l,w_n)\right)\right.}
\end{aligned} \tag{5.22}$$

Now systems of equations can be set up to match these moments and compute αs and βs at the next time step. We have T variables of the form $M_{\theta_t}(P_n)$. We can set up a system of T equations by equating 5.13 and 5.21. However since $\sum_t M_{\theta_t}(P_n) = 1$, the last equation becomes redundant. To resolve this, we use one second order moment to get a system of $T \times T$ equations. Due to the special structure of the problem, we can use the following system of equations to compute α and β at the next time step

$$M_{\theta_{d,t}}\left(P_n\right)=\sum_i \frac{\alpha_{d,i}}{\sum_j \alpha_{d,j}} \frac{\beta_{i,w}}{\sum_j \beta_{i,j}} \frac{\alpha_{d,i}+\delta\left(i,t\right)}{\sum_j \alpha_{d,j}+1} \qquad 1 \leq t \leq T-1 \tag{5.23}$$

$$M_{\theta^2_{d,1}}\left(P_n\right)+\sum_k M_{\theta^2_{d,k}}\left(P_n\right)=\sum_k \sum_i \frac{\alpha_{d,i}}{\sum_j \alpha_{d,j}} \frac{\beta_{i,w}}{\sum_j \beta_{i,j}} \frac{\alpha_{d,i}+\delta\left(i,k\right)}{\sum_j \alpha_{d,j}+1} \frac{\alpha_{d,i}+\delta\left(i,k\right)+1}{\sum_j \alpha_{d,j}+2} \tag{5.24}$$

$$+\frac{\alpha_{d,i}}{\sum_j \alpha_{d,j}} \frac{\beta_{i,w}}{\sum_j \beta_{i,j}} \frac{\alpha_{d,i}+\delta\left(i,1\right)}{\sum_j \alpha_{d,j}+1} \frac{\alpha_{d,i}+\delta\left(i,1\right)+1}{\sum_j \alpha_{d,j}+2} \tag{5.25}$$

These equations can be solved to compute the new αs at step n using

$$\sum \alpha_d(n)=\frac{M_{\theta_{d,1}}\left(P_{n-1}\right)+\sum_i M_{\theta_{d,i}}\left(P_{n-1}\right)-M_{\theta^2_{d,1}}\left(P_{n-1}\right)-\sum_i M_{\theta^2_{d,i}}\left(P_{n-1}\right)}{M_{\theta^2_{d,1}}\left(P_{n-1}\right)+\sum_i M_{\theta^2_{d,i}}\left(P_{n-1}\right)-\left(M_{\theta_{d,1}}\left(P_{n-1}\right)\right)^2+\sum_i^T\left(M_{\theta_{d,i}}\left(P_{n-1}\right)\right)^2} \tag{5.26}$$

$$\alpha_{d,i}(n)=M_{\theta_{d,i}}\left(P_n\right)\sum \alpha_d(n) \tag{5.27}$$

Where $\alpha(n)$ is the value of α after seeing the n^{th} word. A similar treatment of the moments with respect to ϕ gives us the following equations

$$M_{\phi_{t,w}}\left(P_n\right)=\sum_i \frac{\alpha_{d,i}}{\sum_j \alpha_{d,j}} \frac{\beta_{i,w}}{\sum_j \beta_{i,j}} \frac{\beta_{d,i}+\delta\left(i,t\right)\delta(w,w_n)}{\sum_j \beta_{d,j}+\delta(i,t)} \qquad 1 \leq t \leq T-1 \tag{5.28}$$

$$M_{\theta_{d,1}}\left(P_n\right)+\sum_k M_{\theta_{d,k}}\left(P_n\right)=\sum_k \sum_i \frac{\alpha_{d,i}}{\sum_j \alpha_{d,j}} \frac{\beta_{i,w}}{\sum_j \beta_{i,j}} \frac{\alpha_{d,i}+\delta\left(i,1\right)}{\sum_j \alpha_{d,j}+1} \frac{\alpha_{d,i}+\delta\left(i,1\right)+1}{\sum_j \alpha_{d,j}+2} \tag{5.29}$$

These equations can be solved to compute the new αs at step n using

$$\sum \alpha_d(n)=\frac{M_{\theta_{d,1}}\left(P_{n-1}\right)+\sum_i M_{\theta_{d,i}}\left(P_{n-1}\right)-M_{\theta^2_{d,1}}\left(P_{n-1}\right)-\sum_i M_{\theta^2_{d,i}}\left(P_{n-1}\right)}{M_{\theta^2_{d,1}}\left(P_{n-1}\right)+\sum_i M_{\theta^2_{d,i}}\left(P_{n-1}\right)-\left(M_{\theta_{d,1}}\left(P_{n-1}\right)\right)^2+\sum_i^T\left(M_{\theta_{d,i}}\left(P_{n-1}\right)\right)^2} \tag{5.30}$$

$$\alpha_{d,i}(n)=M_{\theta_{d,i}}\left(P_n\right)\sum \alpha_d(n) \tag{5.31}$$

Alg. 5 summarizes Bayesian moment matching for the general case where both Θ and Φ are estimated.

Algorithm 5 momentMatching (Θ and Φ unknown)

```
1: Let $f(\Theta, \Phi|\alpha, \beta)$ be a family of dist. as in Eq. 5.9
2: $P_0(\Theta, \Phi) \leftarrow \sum_{\sigma_i \in S^T} w_{\sigma_i} f^{\sigma_i}(\Theta, \Phi|\alpha, \beta)$
3: for $n = 1$ to $N$ do
4:     Compute $P_n(\Theta, \Phi)$ from $P_{n-1}(\Theta, \Phi)$ (Eq. 5.18)
5:     Compute $S(f)$ for $P_n$ (Eq. 5.21, 5.22)
6:     Compute $\bar{\alpha}, \bar{\beta}$ (Eq. 5.13, 5.14, 5.16, 5.17)
7:     $P_n(\Theta, \Phi) \leftarrow \sum_{\sigma_i \in S^T} w_{\sigma_i} f^{\sigma_i}(\Theta, \Phi|\bar{\alpha}, \bar{\beta})$
8: end for
```

The computational complexity of Alg. 5 is $O(NWT^2)$. There are N words and each word is processed by updating the hyperparameters of the approximating mixtures in $O(WT^2)$ time. The main bottleneck is the computation $\bar{\alpha}$ and $\bar{\beta}$ on Line 6 which requires the solution of $O(W)$ systems of T linear constraints in T variables. These linear systems possess structure that allow us to solve them in $O(T^2)$ (instead of $O(T^3)$). Alg. 5 is an online algorithm since the data is processed in a single sweep, the amount of computation to process each word is independent of the amount of data, and we obtain an estimate of the parameters (in the form of an approximate posterior) after processing each word.

5.3.4 Linear Moment Matching

In the previous section, I showed that the computational complexity of Alg. 5 is $O(NWT^2)$. This is problematic especially for corpora with large dictionary sizes. In this section, I shall discuss an alternative representation of the moment update equations. Let

$$c_t = \frac{1}{k} \frac{\alpha_t}{\sum_i \alpha_i} \frac{\beta_{t,w}}{\sum_i \beta_{t,i}} \tag{5.32}$$

Where k is as defined in Equation 5.19. Then the moment update equations can be rewritten as

$$M_{\theta_t}(P_n) = c_t \frac{\alpha_t + 1}{\sum_i \alpha_i + 1} + (1 - c_t) \frac{\alpha_t}{\sum_i \alpha_i + 1} \tag{5.33}$$

$$M_{\theta_t^2}(P_n) = c_t \frac{\alpha_t + 1}{\sum_i \alpha_i + 1} \frac{\alpha_t + 2}{\sum_i \alpha_i + 2} + (1 - c_t) \frac{\alpha_t}{\sum_i \alpha_i + 1} \frac{\alpha_t + 1}{\sum_i \alpha_i + 2} \tag{5.34}$$

Similarly the moment updates for ϕ become

$$M_{\phi_{t,e}}(P_n) = c_t \frac{\beta_{t,e} + \delta(e,w)}{\sum_i \beta_{t,i} + 1} + (1 - c_t)\frac{\beta_{t,e}}{\sum_i \beta_{t,i}} \tag{5.35}$$

$$M_{\phi^2_{t,e}}(P_n) = c_t \frac{\beta_{t,e} + \delta(e,w)}{\sum_i \beta_{t,i} + 1} \frac{\beta_{t,e} + \delta(e,w) + 1}{\sum_i \beta_{t,i} + 2} + (1 - c_t)\frac{\beta_{t,e}}{\sum_i \beta_{t,i}} \frac{\beta_{t,e} + 1}{\sum_i \beta_{t,i} + 1} \tag{5.36}$$

Where e represents all words other than the current observed word and w represents the current observed word. This reduces the complexity to the algorithm but it is still dependent on the size of the vocabulary W. In order to resolve that the following identities can be used:

$$\alpha_t = M_{\theta_t} \sum_i \alpha_i \qquad \beta_{t,w} = M_{\phi_{t,w}} \sum_i \beta_{t,i} \tag{5.37}$$

The moment update equations can now be rewritten as

$$M_{\theta_t}(P_n) = M_{\theta_t}(P_{n-1}) \left[\frac{\sum \alpha + c_t / M_{\theta_t}(P_{n-1})}{\sum \alpha + 1} \right] \tag{5.38}$$

$$M_{\theta_t^2}(P_n) = M_{\theta_t^2}(P_{n-1}) \left[\frac{\sum \alpha + 2c_t / M_{\theta_t}(P_{n-1})}{\sum \alpha + 2} \right] \tag{5.39}$$

Using a similar treatment for ϕ, for all words e which are not equal to the current observed word

$$M_{\phi_{t,e}}(P_n) = M_{\phi_{t,e}}(P_{n-1}) \left[1 - \frac{c_t}{\sum \beta_t + 1} \right] \tag{5.40}$$

Define

$$x = \left[1 - \frac{c_t}{\sum \beta_t + 1} \right] \qquad y = \frac{c_t}{\sum \beta_t + 1} \frac{1}{x M_{\phi_{t,w}}(P_{n-1})} \tag{5.41}$$

The update equations now become

$$M_{\phi_{t,e}}(P_n) = \left[M_{\phi_{t,e}}(P_{n-1}) \right] x \; e \neq w \tag{5.42}$$

$$M_{\phi_{t,w}}(P_n) = \left[M_{\phi_{t,w}}(P_{n-1}) + y \right] x \tag{5.43}$$

Similar equations can be derived for the second order moments. Equations 5.42 and 5.43 show that the full topic-word matrix does not need to be updated at every step. In fact, if we update the column corresponding to the observed word w by adding y to it, then it is sufficient to store x in order to recover the full topic-word matrix at the next time step. This is significant as this removes the dependency of moment update on the size of the vocabulary.

We can now choose a pivot $1 \leq piv \leq T$ and derive the equations for computing the sum of counts for both α and β. For readability I use $piv = 1$ while deriving the equations, however it is not required.

$$\sum \alpha(n+1) = \frac{M_{\theta_1}(P_{n-1}) - M_{\theta_1^2}(P_{n-1})}{M_{\theta_1^2}(P_{n-1}) - (M_{\theta_1}(P_{n-1}))^2} \tag{5.44}$$

$$\sum \beta_j(n+1) = \frac{M_{\phi_{j,1}}(P_{n-1}) + -M_{\phi_{j,1}^2}(P_{n-1})}{M_{\phi_{j,1}^2}(P_{n-1}) - \left(M_{\phi_{j,1}}(P_{n-1})\right)^2} \tag{5.45}$$

These equations give us a moment matching algorithm for LDA that has $O(NT)$ complexity. Alg. 6 summarizes this approach.

Algorithm 6 momentMatching (Θ and Φ unknown)

```
 1: Initialize $M_{\theta_t}(P_0)$, $M_{\theta_t^2}(P_0)$, $M_{\phi_{t,w}}(P_0)$, $M_{\phi_{t,1}^2}(P_0)$ and $x_t = 1\ \forall 1 \leq t \leq T,\ 1 \leq w \leq W$

 2: for $n = 1$ to $N$ do
 3:    Let $w_n$ be the current word at time step $n$
 4:    Set $k = \sum_t M_{\theta_t}(P_{n-1})\, x_t M_{\phi_{t,w_n}}(P_{n-1})$
 5:    Compute $c_t = (1/k)\, M_{\theta_t}(P_{n-1})\, x_t M_{\phi_{t,w_n}}(P_{n-1})$ for each $1 \leq t \leq T$
 6:    Update the moments with respect to $\theta$ as $M_{\theta_t}(P_n) = M_{\theta_t}(P_{n-1}) \left[\frac{\sum\alpha + c_t/M_{\theta_t}(P_{n-1})}{\sum\alpha+1}\right]$
       and $M_{\theta_t^2}(P_n) = M_{\theta_t^2}(P_{n-1}) \left[\frac{\sum\alpha + 2c_t/M_{\theta_t}(P_{n-1})}{\sum\alpha+2}\right]$
 7:    Update $x_t = 1 - \frac{c_t}{\sum\beta_t+1}$   $y_t = \frac{c_t}{\sum\beta_t+1}\frac{1}{xM_{\phi_{t,w}}(P_{n-1})}$ for all $1 \leq t \leq T$
 8:    Update $M_{\phi_{t,w_n}}(P_n) = M_{\phi_{t,w_n}}(P_{n-1}) + y_t$ for all $1 \leq t \leq T$
 9:    Compute the new sum of counts using Equations 5.44 and 5.45.
10: end for
```

5.3.5 Discussion

An important issue in Bayesian learning is unidentifiablity, which arises when several solutions can explain the data equally well. These equivalent solutions are usually due to symmetries in the problem. By approximating the posterior with a mixture of products of Dirichlets with permuted hyperparameters, BMM properly handles symmetries arising from permutations of the topic labels. If there are fewer than K topics, BMM will not be affected since this simply means that fewer mixture components could have been used since some of them will become identical. It is possible that other types of symmetries may arise, although the authors are not aware of any. If they lead to equivalent solutions that are not captured by the current mixture of products of Dirichlets, BMM could be modified to work with a family of distributions f that takes into account those symmetries.

Theorem 5.3.1. *Under the assumption that the likelihood of each word* $\Pr(w|\theta)$ *is non-zero, the moment matching procedure described in Alg. 5 and 6 performs exact inference with respect to at least one initial distribution.*

The above theorem is a generalization of Theorem 5.2.1 and its proof follows the same steps. Again, BMM can be viewed as lazily waiting till a moment of the initial distribution is needed in the computation to set that moment, which ensures that inference can proceed tractably.

5.4 Results

In this section, I will present results of a comparison between BMM and existing parameter learning techniques for LDA. The performance of these algorithms is demonstrated with synthetic and real corpora.

5.4.1 UCI Bag of Words Document Corpus

The UCI bag of words data sets are a publicly available bag of words corpora which contains 4 data sets [51]. Each document is represented as a term frequency vector. The NIPS data set consists of complete NIPS papers. Then Enron data set contains emails of Enron employees. The NYTimes dataset contains New York Times articles. The PubMed database is a set of abstracts from PubMed.

5.4.2 Wikipedia Corpus

I have also trained these algorithms on 100,000 documents from the english version of Wikipedia. This corpus was shared with us by Hoffman et.al. [40]. They removed all words not from a fixed vocabulary of 7,995 common words. This vocabulary was obtained by removing words less than 3 characters long from a list of the 10,000 most common words in Project Gutenberg texts obtained from http://en.wiktionary.org/wiki/Wiktionary:Frequency lists.

5.4.3 Twitter Data

I collaborated with an industry partner "In the Chat" that does social media mining on behalf of other companies. The data set is comprised of tweets related to a cell phone provider. There is about 850 days of Twitter data. The tweets are manually labeled as Customer Service, Pricing Promo, Sales, Hardware, Billing Payments, Technical, Cancel, Rant and Other. The company is interested in the topic distribution during a given period of time in order to identify trending topics. They are also interested in tweets about their competitors to pro-actively engage customers that are thinking of leaving a competitor. The data was divided into documents where each document contains 10 days of tweets. Very common and extremely rare words were removed.

Note that the topics discovered by any unsupervised algorithm may or may not correspond to the topics identified by human judges. In addition, some pairs of topics (e.g. Customer Service vs Rant, Hardware vs Technical, Sales vs Pricing Promo) are difficult to distinguish even for humans. Tweets are different from other corpora such as scientific papers or wikipedia because the vocabulary used in tweets includes a lot of slang and abbreviations that cannot be disregarded since they may have strong correlations with some topics.

5.4.4 Synthetic Data

The algorithms were also also tested on synthetic data generated from an LDA model. We generate 20 observation sequences by choosing α and β, setting $T = 5$, $W = 1000$. The length of each document varies between 2000 and 5000. We choose these numbers to mimic Twitter data.

The datasets are summarized in Table 5.1. The number of tokens refers to unique words in the document.

Data Set	**NIPS**	**Enron**	**NYTimes**	**PubMed**	**Wikipedia**	**Twitter**
Vocabulary Size	10433	26186	299751	128972	7702	4632
Total Documents	1736	39733	99064	8200000	3764100	832
Total Words	2122250	5991406	95073360	691810664	3764100	527128
Av. Words/Document	1222.5	150.8	317.2	84.4	126.7	633.6
Av. Tokens/Document	470	87.7	222.8	56.1	65.1	269.7
Total Chunks	30	50	101	100	100	25
Av. Tokens/Chunk	27241	69708	661360	4619900	2451600	8976
Av. Words/Chunk	70742	119830	941320	6918100	4771900	21085

Table 5.1: Summary Statistics of the data sets

5.4.5 Experiments

I have compared the Bayesian Moment Matching (BMM) algorithm to various state of the art algorithms for topic modeling. The algorithms I compare against include Variational Bayes (VBLP) [15], Belief Propagation (BPLP) [76], Gibbs Sampling (GS)[70], Expectation Propagation (EP) [56], Spectral LDA (spectralLDA)[7], Online Variational Bayes (OVB)[40] and Sparse Online Variational Bayes (SOI)[54]. I have used the Matlab toolbox by [76] for BP, GS and VB. We implemented Spectral LDA based on [7] since there is no publicly available implementation. For OVB Hoffman et al.'s [40] toolbox has been used. For Sparse Online Inference (SOI), the Mallet toolbox [53] was used. For the offline algorithms (Gibbs Sampling, Varitional Bayes, Belief Propagation and Spectral LDA), the algorithm is run on the entire training set and the perplexity is computed on the test set. Since the larger datasets including NYTimes, Pubmed and Wikipedia were unable to fit in the memory in memory, the comparison with offline algorithms for these datasets is not shown. In the case of the online algorithms such as MM, OVB and SOI, they can incorporate new observations with a constant amount of computation. For online experiments, the data is given to the algorithm in the form of batches. The perplexity is measured after learning the parameters by incorporating each new batch. In MM, if the first few observations belong to the same document, then the parameters of the other documents are momentarily unidentifiable, which may cause some problems. To avoid this, the first few documents are permuted randomly.

Each data set is divided into small batches to measure perplexity after each batch is trained. I have used 10 fold cross validation such that for each fold 10% of the documents are in the test set while 90% are in the training set.

I have compared the parameters computed by each algorithm by measuring the likelihood of held out test data. To quantify this measure we compute the perplexity:

$$perplexity = exp\left\{-\frac{\sum_{d=1}^{D} log(\Pr(w_{d,1:N_d}))}{\sum_{d=1}^{D} N_d}\right\} \tag{5.46}$$

where D is the total number of documents in the test set and $w_{d,1:N_d}$ is the sequence of words of the d^{th} document in the test set. The perplexity is computed using Gibbs Sampling for inference. I use 10-fold cross validation and report the mean and standard deviation of the 10 runs.

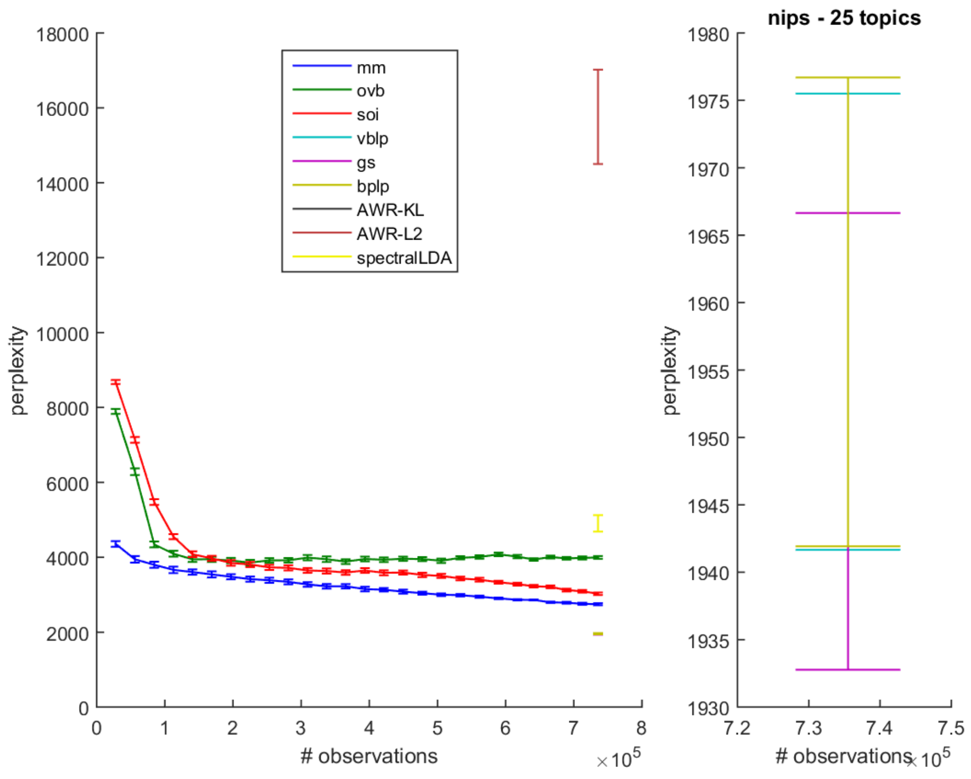


Figure 5.2: Comparison of different learning algorithms for topic modeling for the NIPS dataset. The number of topics $T = 25$. The second figure is the perplexity for vblp, gs and bplp zoomed

Most of these algorithms require a parameter β that defines the prior for the topic-word distribution is a uniform Dirichlet. This value is set to 1 (this corresponds to a uniform

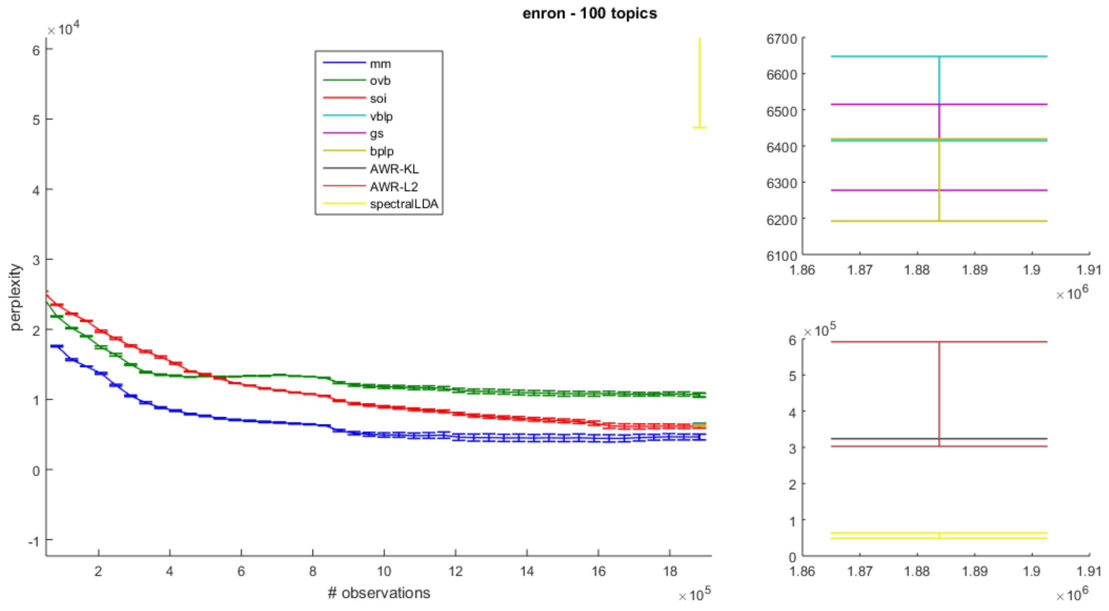


Figure 5.3: Comparison of different learning algorithms for topic modeling for the Enron dataset. The number of topics $T = 100$. The top right figure is the perplexity of vblp, gs and bplp and the bottom right is the preplexity of AWR-KL, AWR-L2 and Spectral LDA

distribution) for all our experiments. I have used 500 iterations for Gibbs Sampling. For Variational Bayes I use 8 inner iterations and 500 outer iterations. For Online Variational Bayes and Sparse Online Inference, the total number of documents changes as more observations become available.

Figures 5.2, 5.3, 5.4 show the how the perplexity changes with increasing number of observations at every time step. The graphs show that moment matching competes favorably with other algorithms in terms of perplexity of test data. Displaying the perplexity of some of the algorithms on the same graph makes it difficult to visualize the difference between various online techniques. Therefore, I have provided side panels in various figures that show the perplexity for some of the off line algorithms. We find that moment matching is actually comparable with offline techniques such as Variational Bayes, Gibbs Sampling and Belief Propagation. Both Online Variational Bayes and the Sparse Online Inference algorithm by Mimno. et. al. [54] are stochastic gradient descent algorithms. In the SOI code, the authors discard samples with low weight. Therefore, most of the topic-word matrix learned by that algorithm is actually equal to the zero. Then β is added to the whole matrix for smoothing. Therefore, most of the matrix entries consist of initial value

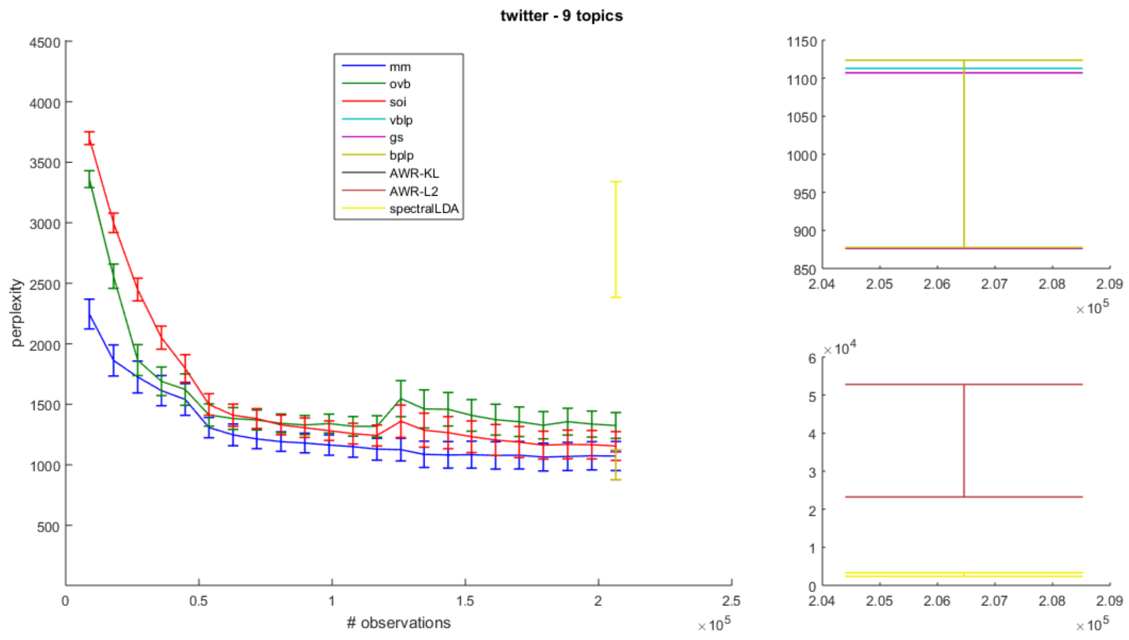


Figure 5.4: Comparison of different learning algorithms for topic modeling for the Twitter dataset. The number of topics $T = 9$. The top right figure is the perplexity of vblp, gs and bplp and the bottom right is the preplexity of AWR-KL, AWR-L2 and Spectral LDA

of the parameter β. This may distorts the perplexity as it smooths over some noise in the data. For the larger datasets including NYTimes, PubMed and Wikipedia, we only show the results for the online algorithms as the data can not be fully loaded into the memory. The results are shown in Figs. 5.5, 5.6 and 5.7. Moment matching shows clear gains in all data sets with significant gains in the Wikipedia data set. This may be due to the fact that in the wikipedia dataset, the vocabulary has only about 7000 words as opposed to pubmed and nytimes where the vocabulary has around 100,000 words. Therefore, the data set is less noisy and the words are more informative. Another reason why we see gains for moment matching is because both OVB and SOI are stochastic gradient descent methods that aim to construct noisy estimates of the gradient by sub-sampling a block of data. This estimate improves over time but can be noisy in the start.

I also compared the amount of time taken by each algorithm, although it is important to note that the programming language and the degree of code optimization can affect the running times. The results are summarized in 5.2. In terms of time, OVB and SOI have an advantage over MM in some cases. This is due to the fact that the SOI, OVB, GS, VB,

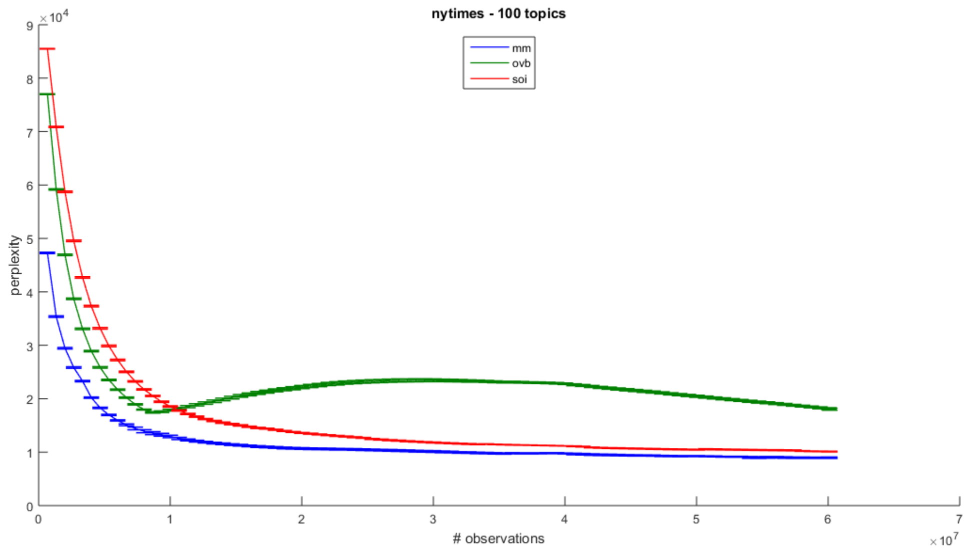


Figure 5.5: Comparison of different learning algorithms for topic modeling for the NYTimes dataset. The number of topics $T = 100$.

BP all depend on processing each token at one time step where each token represents a unique word in the document. If the word is repeated multiple times in the document, it is still processed once. On the other hand Moment Matching currently relies on processing each word to compute the exact posterior. SOI also has a slight advantage in terms of time as it is implemented in java whereas the other algorithms are implemented using Matlab and Python. Despite this, BMM is the second best algorithm in terms of training time.

I also tried to compare Bayesian Moment Matching to Expectation Propagation (EP) on Twitter data. It took about 10 days for EP to go over the sequence once (it still needed to do many more iterations before convergence) and the perplexity remained high. We present a comparison of both algorithms on a smaller synthetic data set in Fig. 5.8. MM outperforms EP.

In this chapter, we have described an online Bayesian moment matching technique that incrementally estimates the parameters of LDA by performing a single sweep of the data. The approach is simple and it compares favorably to existing approaches in terms of time and perplexity. This can be explained by the fact that the approach is performing exact inference with respect to an implicit set of initial distributions. In subsequent chapters, we will apply this technique to other models such as HMMs for activity recognition.

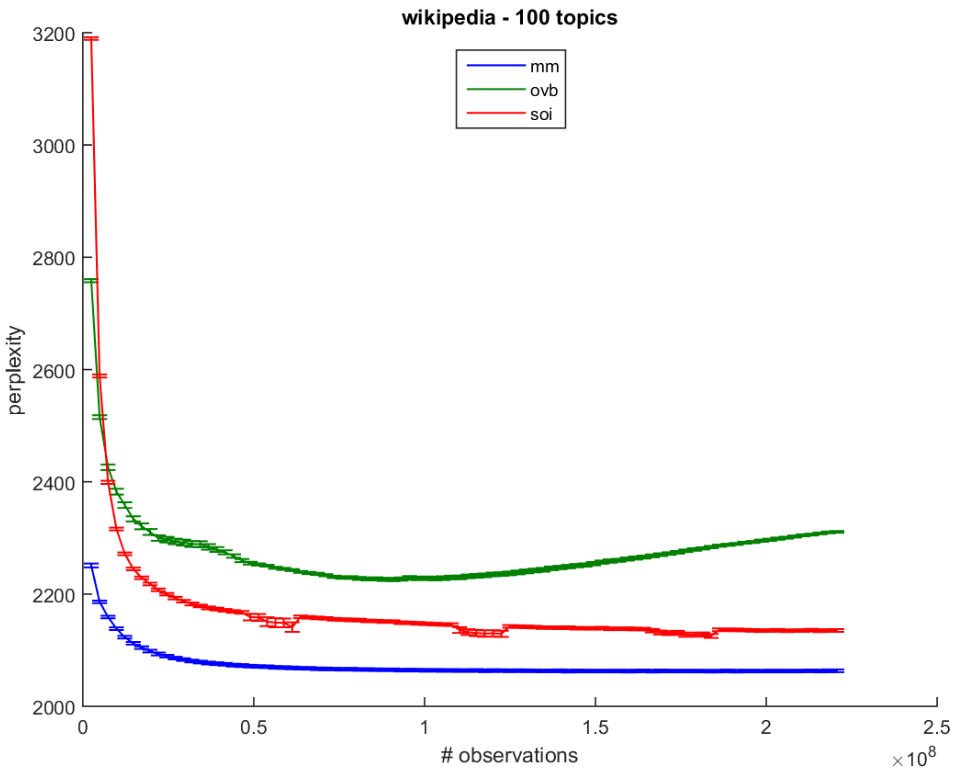


Figure 5.6: Comparison of different learning algorithms for topic modeling for the Wikipedia dataset. The number of topics $T = 100$.

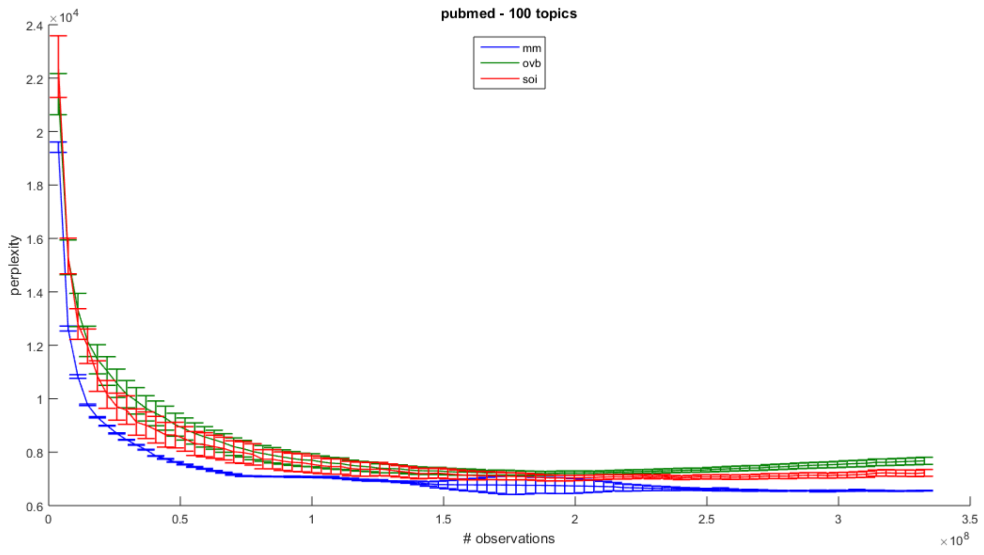


Figure 5.7: Comparison of different learning algorithms for topic modeling for the PubMed dataset. The number of topics $T = 100$.

Data Set	**enron**	**NIPS**	**Twitter**	**NYTimes**	**PubMed**	**Wikipedia**	**Code Language**
# of Topics	100	25	9	100	100	100	
GS	1904.2	591.4	60.1				c++,Matlab
VBLP	212053	12969.1	1684.2				c++,Matlab
BPLP	6334.1	172.35	20.3				c++,Matlab
AWR-KL	3105.1	661.2	58.2				Matlab
AWR-L2	2535.2	740.5	106.6				Matlab
Spectral	4332.1	1261.9	107				Matlab
OVB	504.12	78.2	15.1	7221.6	58438.8	126.7	Python
SOI	183.25	0.51	1.26	2041.2	13680	65.1	Java
MM	327.1	123.6	9.43	2475.9	43164.2	100	Matlab

Table 5.2: Time taken in seconds by each algorithm to go over complete data.

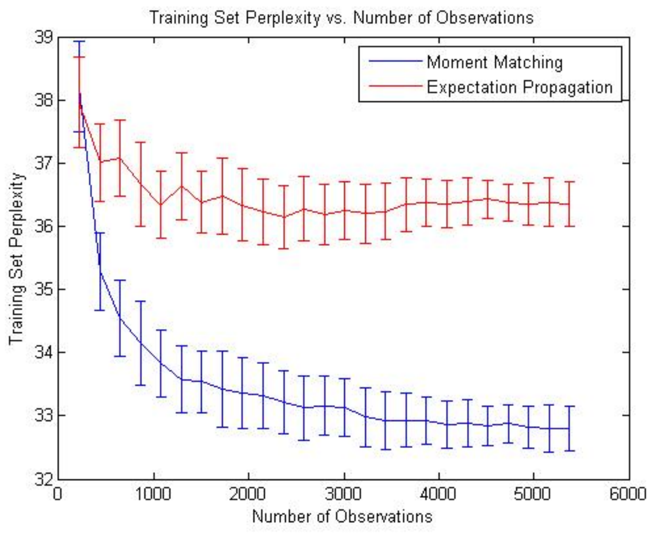


Figure 5.8: EP vs MM on synthetic data. $T = 5$

Chapter 6

Activity Recognition with Instrumented Walker

In Chapter 1, I introduced the activity recognition problem for instrumented walkers. Previous approaches for activity recognition such as [39] and [6] have tried to solve this problem by constructing complex physical models of the walker. These approaches have limited applicability since they predict a primitive set of activities and are not easily adaptable to changes in the physical configurations of the walker. In this chapter I describe how to construct a probabilistic graphical model for activity recognition. I will also describe the setup for some data collection experiments. The main contributions in this chapter are

- Design and training (supervised and unsupervised) of probabilistic graphical models (HMMs and CRFs) tailored to activity recognition with instrumented walkers
- Comprehensive analysis of these techniques with real data collected with control subjects and regular walker users living in a long term care facility
- Comprehensive analysis of the ease/difficulty of recognizing common walker user activities with existing algorithms

6.1 The Walker and Experimental Setup

As mentioned in Chapter 1, I had access to a walker developed by [73]. The walker is equipped with various sensors including a 3-D accelerometer in the seat of the walker that records the acceleration across the x-axis, y-axis and z-axis. In addition to these sensors, there is a load-cell in each leg of the walker. The walker is also equipped with a wheel encoder, which measures the distance traveled by the wheel. The sensor readings vary between 0 and $2^{16} - 1$ and the data is channeled via bluetooth to a PDA for acquisition. The data rate is 50 readings/second. In addition to these sensors, there are two cameras on the walker. One of them is facing backwards and provides the video feed of the user's legs. The other one is looking forwards and provides the video feed of the environment. The video frame rate is approximately 30 frames/second.

In order to collect data for our models, two different sets of experiments were conducted that are described below.

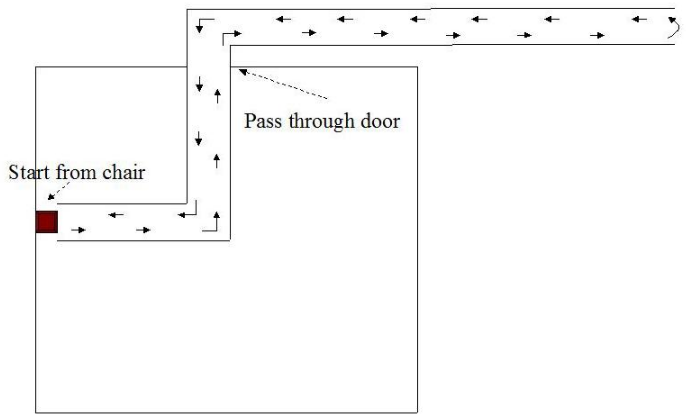


Figure 6.1: Course for data collection experiment done with the walker. Healthy young subjects were asked to follow this course

6.1.1 Experiment 1

17 healthy young subjects (age between 19 and 53) were asked to use the walker and go twice through the course shown in Figure 6.1. The activities performed by the participants are shown in Table 6.1.

Not Touching the Walker (NTW)
Stop/Standing (ST)
Walking Forward (WF)
Turn Left (TL)
Turn Right (TR)
Walking Backwards (WB)
Transfers (Sit to Stand/Stand to Sit) (TRS)

Table 6.1: Activities performed in Experiment 1

6.1.2 Experiment 2

In a second experiment, 8 older adults (age 84 to 97) who are regular walker users, were asked to follow the course shown in Figure 6.2. These experiments were conducted at a retirement home and the participants were residents of that facility. 12 older adults (age 80 to 89) who do not live in a retirement home and are not regular walker users were also asked to follow the same course. The activities performed during this experiment include those of Table 6.1 and some additional activities listed in Table 6.2.

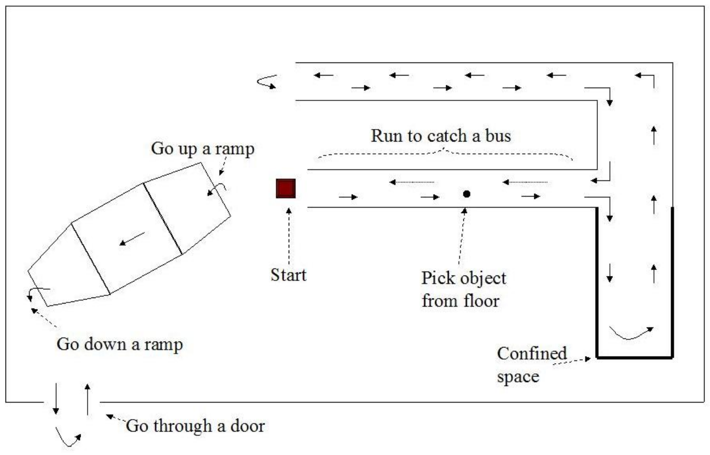


Figure 6.2: Course used in data collection experiment for older subjects who are walker users

Going up Ramp (GUR)
Going down Ramp (GDR)
Sitting on Walker (SW)
Reaching Task (RT)
Going up Curb (GUC)
Going down Curb (GDC)

Table 6.2: Additional activities performed during Experiment 2

6.1.3 Sensor Data

My goal was to perform activity recognition based on accelerometers, load cells and wheel encoder. These sensors only measure the activities of the person indirectly, therefore, it is not obvious a priori which activities can be easily recognized. To establish ground truth, I synchronized the video feed from the rear facing camera with the data feed from all sensors. I developed a tool that allowed me to annotate the data based on the video feed. The tool also helped in visualizing the data synchronized with the video. Since the video camera only looks at the legs and feet of the person for privacy reasons, we are unable to know for sure when a person is exactly touching the walker apart from looking at the load cell readings. In addition to that this process is prone to human error on the part of the labeler. For example, the time that marks the start of a right turn may be different depending on the person labeling the data.

We define the reading on the load sensor mounted on the front right leg of the walker as L_{FR}, reading on the load sensor mounted on the front left leg as L_{FL}, reading on the load sensor mounted on the rear left leg as L_{RL} and reading on the load sensor mounted on the rear right leg as L_{RR}. Instead of using the raw data of the load sensors, we consider total load on the walker, the sagittal plane center of pressure (COP_{SP}) and the frontal plane center of pressure (COP_{FP}). Formally, these quantities are given by

$$\text{TotalLoad} = L_{RR} + L_{RL} + L_{FR} + L_{FL} \tag{6.1}$$

$$COP_{SP} = \frac{L_{FL} + L_{FR} - L_{RL} - L_{RR}}{\text{TotalLoad}} \tag{6.2}$$

$$COP_{FP} = \frac{21.25(L_{FL} - L_{FR}) + 26.6(L_{RL} - L_{RR})}{\text{TotalLoad}} \tag{6.3}$$

where the constants 21.25 and 26.6 correspond to the distances of the front/rear load cells to the midline of the walker (in centimeters).

In general, activities that involve moving forward including turns and vertical transitions (ramp and curb negotiation tasks) are difficult to differentiate from each other. Since the load sensors are mounted on the legs instead of being close to the hands, we only get indirect information about how users vary their weight between both hands to keep themselves balanced while negotiating a turn or a vertical transition. Also, vertical transitions may be particularly difficult to recognize due to the wide range of strategies used by people to lift or lower the walker. Similarly, transfers (sit to stand or stand to sit) and reaching tasks (such as opening doors and picking objects) are also expected to be difficult to recognize due to a wide range of strategies.

In Table 6.3 the most frequent value of the sensor for each activity is shown. The second row for each activity gives the probability of seeing that particular value on the sensor for the activity. The higher the probability, the easier it would be for an automated algorithm to recognize this activity when it sees this reading on the sensor. We can see that different sensors help in identifying different activities. When the person is Not touching the walker (NTW) the center of pressure provides a strong signal as there is no load on the walker. For Walking forward (WF), the wheel encoder provides the requisite information. For left turns the x and y axis accelerometers along with COP_{FP} provide a strong signal. The same sensors help us with right turns (TR). For walking backwards (WB) the strongest signal comes from total load indicating that there is very little load on the walker when the person is walking backwards. This is logical as when people walk backwards, they tend to lean back thus removing the weight from the walker. For sitting on walker (SW), the total load alongwith the center of pressure features give an adequate signal. It is interesting to note that for vertical transitions i.e. (going up/down curbs and ramps) the signal from the accelerometers is very strong suggesting that a vertical transition has happened.

In Table 6.4, each element at row i and column j indicates the probability that the current state is j given the previous state was i. This matrix is diagonal heavy owing to the fact that people tend to keep doing the current activity most of the time.

6.2 Activity Recognition Model

I assume that the total number of activities is N and the total number of sensors is S. The sensor readings were discretized by dividing the range of the sensor data into M discrete intervals. The activity at time t is represented by the random variable Y_t. The reading on sensor s at time t is given by the random variable e_t^s where $s \in \{1, \ldots, S\}$.

Activity	Accelerometer			Total Load	COP_{SP}	COP_{FP}	Wheel Encoder	Frequency
	x-axis	y-axis	z-axis					
NTW	10	9	13	6	7	9	16	8554
	.13	.145	.118	.316	.387	.31	.11	
ST	14	8	13	3	8	10	17	51638
	.07	.09	.07	.1	.10	.09	.08	
WF	20	3	21	11	8	11	13	71417
	.08	.1	.09	.09	.07	.07	.13	
TL	2	20	20	15	2	2	8	16867
	.21	.13	.1	.17	.19	.33	.13	
TR	21	20	3	10	21	12	2	15832
	.22	.11	.09	.08	.13	.10	.14	
WB	3	2	4	2	2	2	7	1045
	.09	.09	.08	.37	.27	.24	.18	
RT	19	5	13	3	4	8	8	12287
	.08	.07	.06	.14	.08	.08	.07	
SW	7	14	6	18	20	21	17	71334
	.08	.11	.08	.18	.11	.17	.09	
GUR	2	2	2	15	2	3	4	2998
	.31	.67	.22	.21	.20	.10	.11	
GDR	21	21	2	3	21	11	2	3194
	.14	.64	.24	.11	.11	.08	.15	
GUC	21	2	2	2	2	5	8	3087
	.14	.45	.27	.64	.4	.12	.1	
GDC	21	21	2	2	21	21	7	1700
	.11	.61	.26	.13	.13	.13	.12	

Table 6.3: Highest value for each sensor s with respect to activity y, $\arg\max_e \Pr(e^s|y)$. The second row for each activity is the probability of this highest value $\max_e \Pr(e^s|y)$. This table is based on the data from the second experiment

	NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC
NTW	.998	.001	.0001	.0002	.0001	0	0	0	0	0	0	0
ST	.0002	.9969	.0012	.0005	.0004	0	.0001	.0007	0	0	0	0
WF	0	.0009	.9959	.0012	.0009	0	.0004	0	.0001	.0002	.0001	.0002
TL	0	.0017	.0044	.9930	.0001	0	.0007	0	.0002	0	0	0
TR	0	.0011	.0048	.0001	.9934	0	.0002	0	0	0	.0003	0
WB	0	.0019	.0038	0	0	.9923	.0019	0	0	0	0	0
RT	.0002	0	.0028	.0002	.0004	.0006	.9958	0	0	0	0	0
SW	.0001	.0004	0	0	0	0	0	.9995	0	0	0	0
GUR	0	0	.005	0	0	0	0	0	.9950	0	0	0
GDR	0	0	.0009	0	0	0	0	0	0	.9953	0	0
GUC	0	.0006	.005	0	0	0	0	0	0	0	.9945	0
GDC	.0006	.0006	.0071	0	.0006	0	0	0	0	0	0	.9912

Table 6.4: Empirical Transition Distribution for Experiment 2

6.2.1 Hidden Markov Model

I have used a Hidden Markov Model (HMM) for this problem where the activities are the hidden variables and the sensor readings are observations. The hidden Markov model is given in Figure 6.3. The parameters that I model are as follows:

- **Transition Model**: The transition distribution models the change in activity being performed over time. The distribution over the current activity Y_t given that the previous activity is y is denoted by $\theta_y = \Pr(Y_t|Y_{t-1} = y)$ where $\theta_y = \{\theta_{y,1}, \ldots, \theta_{y,N}\}$ and $\theta_{y,i} = \Pr(Y_t = i|Y_{t-1} = y)$ is the probability that the current activity is i given the previous activity was y. Note that in the activity recognition problem, activities tend to have a high persistence probability, i.e. user is very likely to keep performing the same activity for long intervals of time.

- **Observation Model**: The observation function models the effect of the current activity on sensor readings at time t. The distribution over sensor reading given that the user is performing activity y is denoted by $\phi_{y,s} = \Pr(E_t^s|Y_t = y)$ where $\phi_{y,s} = \{\phi_{y,1,s}, \ldots, \phi_{y,M,s}\}$ and $\phi_{y,e,s} = \Pr(E_t^s = e|Y_t = y)$ is the probability that we observe reading e on sensor s if the current state is y.

In Figure 6.4 the model is given a full Bayesian treatment by allowing the parameters to be sampled from priors. To generate the observation sequence, for each $y \in \{1, \ldots, N\}$ we

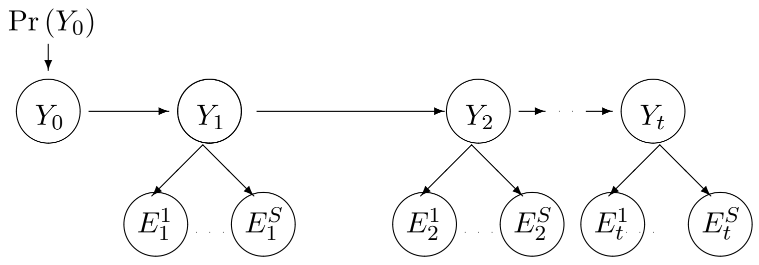


Figure 6.3: A Hidden Markov Model For Activity Recognition

first sample $\theta_{y,y'}$ from a prior $f(\theta_y; \alpha_y)$ where y' is the previous state. Then for each sensor s, we sample $\phi_{y,s}$ from another prior $f(\phi_{y,s}; \beta_{y,s})$. Then we generate each observation by first sampling the current state y' from θ_y where y is the state sampled at the previous time step and then for each sensor s, we sample an observation e from $\phi_{y',s}$.

The following terminology will be used frequently for notational convenience

$\Theta = \{\theta_{i,j} : 1 \le i \le N, 1 \le j \le N\}$

$\Phi = \{\phi_{y,e,s} : 1 \le y \le N, 1 \le e \le M, 1 \le s \le S\}$.

$e^s_{a:b} = e^s_a, e^s_{a+1} \dots e^s_b$ a sequence of readings on sensor s from time a to b

$e^{1:S}_{a:b} = e^{1:S}_a, e^{1:S}_{a+1} \dots e^{1:S}_b$ a sequence of readings from all sensors from time a to b

$y_{a:b} = y_a, y_{a+1} \dots y_b$ a sequence of activities from time a to b

$\tilde{y}_t$ predicted activity at time t

$F^y_t = \Pr\left(Y_t = y, e^{1:S}_{1:t} | \Theta, \Phi\right)$

$B^y_t = \Pr\left(e^{1:S}_{t+1:T} | Y_t = y, \Theta, \Phi\right)$

$\gamma^y_t = \Pr\left(Y_t = y | e^{1:S}_{1:T}, \Theta, \Phi\right)$

$\xi^{y,j}_t = \Pr\left(Y_t = y, Y_{t+1} = j | e^{1:S}_{1:T}, \Theta, \Phi\right)$

6.3 Prediction

I have used three techniques to do prediction with the HMM. Following is a brief description of the prediction algorithms used. For detailed derivations, please refer to [67] and [13].

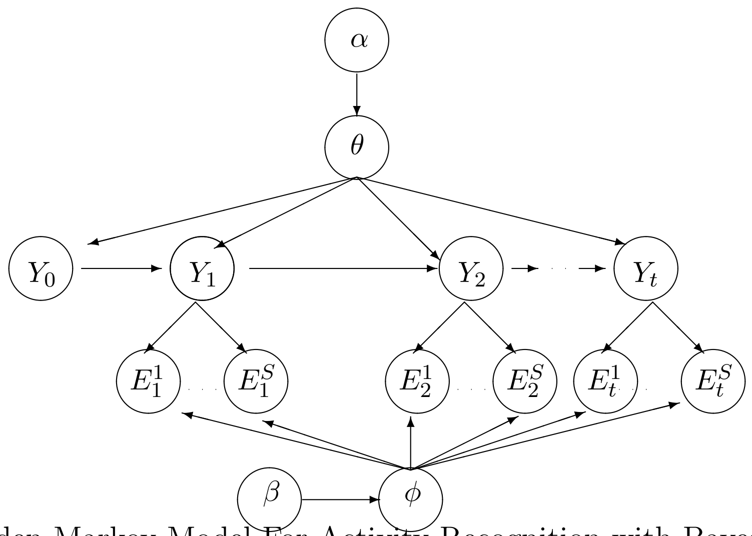


Figure 6.4: A Hidden Markov Model For Activity Recognition with Bayesian Priors

Viterbi Algorithm: Given a sequence of observations, the Viterbi algorithm finds the sequence of states most likely to generate those observations. These probabilities can be calculated online as only past observations are used. However, to retrieve the most likely sequence, we have to do a backtracking step that is explained in [67] and it can not be performed online.

$$\begin{aligned}\tilde{y}_{t+1} &= \arg\max_y \left(\arg\max_{y_{1:t}} \left(\Pr\left(y_{1:t}, Y_{t+1} = y | e^{1:S}_{1:t+1}, \Theta, \Phi\right)\right)\right) \\ &= \arg\max_y \prod_i \phi_{y,e_t,i} \arg\max_{y'} \left(\theta_{y,y'} \left(\arg\max_{y_{1:t-1}} \left(\Pr\left(y_{1:t-1}, y' | e^{1:S}_{1:t}, \Theta, \Phi\right)\right)\right)\right) \qquad (6.4)\end{aligned}$$

Filtering: This is the task of estimating a distribution over the current activity at time t given all the observations before it ($e^{1:S}_{1:t}$). Since, this process returns a distribution over the current activity, the final prediction is chosen by picking the activity with the highest probability. This process can be performed online as only past observations are used.

$$\begin{aligned}\tilde{y}_t =& \arg\max_y \Pr\left(Y_t = y | e^{1:S}_{1:t}, \Theta, \Phi\right) \qquad (6.5) \\ =& \arg\max_y \frac{\Pr\left(Y_t = y, e^{1:S}_{1:t} | \Theta, \Phi\right)}{\Pr\left(e^{1:S}_{1:t} | \Theta, \Phi\right)} = \arg\max_y \frac{F^y_t}{\sum_{t'} F^{y'}_t} \\ F^y_t \propto& \prod_s \phi_{y,e^s_t} \sum_i \theta_{i,y} F^i_t \qquad (6.6)\end{aligned}$$

Forward Backward (Baum-Welch Algorithm): This is the task of estimating a distribution over the activity at time t given all the observations $e^{1:S}_{1:T}$. As future observations are used, this process can not be performed online. Similar to filtering, the final prediction is chosen by picking the activity with the highest probability.

$$\begin{aligned}
\tilde{y}_t =& \arg\max_y \Pr\left(Y_t = y|e_{1:T}^{1:S}, \Theta, \Phi\right) && (6.7)\\
=& \arg\max_y \frac{\Pr\left(Y_t = y|e_{1:T}^{1:S}, \Theta, \Phi\right)\Pr\left(e_{1:T}^{1:S}|Y_t = y, \Theta\Phi\right)}{\sum_{y'}\Pr\left(Y_t = y'|e_{1:T}^{1:S}, \Theta\Phi\right)\Pr\left(e_{1:T}^{1:S}|Y_t = y', \Theta\Phi\right)}\\
=& \arg\max_y \frac{F_t^y B_t^y}{\sum_{y'} F_t^{y'} B_t^{y'}}\\
B_t^y =& \sum_i \theta_{y,i} \prod_s \phi_{i,e_{t+1},s} B_{t+1}^i && (6.8)\\
\gamma_t^y =& \frac{F_t^y B_t^y}{\sum_{y'} F_t^{y'} B_t^{y'}} && (6.9)
\end{aligned}$$

6.4 Maximum Likelihood Parameter Learning

Maximum likelihood parameter learning was introduced in section 2.2.3. As mentioned before, we find the values of the parameters that maximize the log likelihood of data

$$\theta_{ML} = \max_\theta \log\left(\Pr\left(e|\theta\right)\right) \tag{6.10}$$

In this section, I will derive equations for maximum likelihood parameter learning in the context of activity recognition.

6.4.1 Supervised Learning

For supervised learning, I labeled the data by hand by synchronizing the data feed with the video feed. Therefore, the optimal parameters Θ^* and Φ^* can be learned from a single labeled sequence of length T using the following equation

$$\begin{aligned}
(\Theta^*, \Phi^*) = & argmax_{\Theta,\Phi} \log\left(\Pr\left(y_{1:T}, e_{1:T}^{1:S}|\Theta, \Phi\right)\right)\\
& \text{subject to} && (6.11)\\
& \sum_i \theta_{y,i} = 1 \,\forall y \in \{1, \ldots, N\}\\
& \sum_{e=1}^{M} \phi_{y,e,s} = 1 \,\forall y \in \{1, \ldots, N\} \forall k \in \{1, \ldots, S\}
\end{aligned}$$

Solving this optimization problem gives us the following values for the transition parameters

$$\theta_{y,y'} = \frac{\sum_{t=1}^{T} \delta\left(y_t = y' \& y_{t-1} = y\right)}{\sum_{t=1}^{T} \delta\left(y_{t-1} = y\right)} = \frac{n_{y,y'}}{n_y},\ \forall y', y \in \{1, \dots, N\} \tag{6.12}$$

Where $n_{y,y'}$ is the number of times y is followed by y' in labeled activity sequence and n_y is the number of times we observe y in the labeled activity sequence. Note that during the experiment, the participants are following a fixed course and thus the activity transitions are biased and do not reflect real life situations. To avoid this pitfall I use the fact that activities tend to persist. Therefore, we can learn the parameters of the transition model using the following distribution:

$$\theta_{y,y'} = \begin{cases} \frac{\tau}{N+\tau-1} & \text{if } y = y' \\ \frac{1}{N+\tau-1} & otherwise \end{cases} \tag{6.13}$$

The observation parameters can be calculated using

$$\phi_{y,e,s} = \frac{\sum_{t=1}^{T} \delta\left(y_t = y \& e_t^s = e\right)}{\sum_{t=1}^{T} \delta\left(y_t = y\right)} = \frac{n_{y,e,s}}{n_y} \tag{6.14}$$

6.4.2 Unsupervised Maximum Likelihood Learning

For unsupervised Learning, I have used the EM algorithm for learning transition probabilities and maximize the expected value of the log likelihood. New values of the parameters are iteratively estimated by solving the following optimization problem

$$\Theta^{i+1}, \Phi^{i+1} = \underbrace{argmax_{\pi,\theta,\phi} \overbrace{\left(\sum_{y_{1:T} \in Y_{1:T}} \Pr\left(y_{1:T}, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \log\left(\Pr\left(y_{1:T}, e_{1:T}^{1:S} | \Theta, \Phi\right)\right) \right)}^{Expectation}}_{Maximization} \tag{6.15}$$

subject to

$$\sum_{y'=1}^{N} \theta_{y,y'} = 1 \forall y \in \{1, \dots, N\}$$

$$\sum_{e=1}^{M} \phi_{y,e,s} = 1 \forall y \in \{1, \dots, N\} \forall s \in \{1, \dots, S\}$$

Where $Y_{1:T}$ represents all possible activity sequences of length T. The solution for this optimization problem gives us the following equations for estimating parameters.

$$\theta_{y,y'} = \frac{\sum_{t=1}^{T} \Pr\left(Y_t = y', Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)}{\sum_{t=1}^{T} \Pr\left(Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)} \tag{6.16}$$

$$\phi_{y,e,s} = \frac{\sum_{t=1}^{T} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \delta\left(e_t^s = e\right)}{\sum_{t=1}^{T} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)} \tag{6.17}$$

The detailed derivation of these equations is given in Appendix A.2. For efficient calculation of these values, we use the terminology defined in section 6.2. Recall that

$$\gamma_t^y = \Pr\left(Y_t = y | e_{1:T}^{1:S}, \Theta, \Phi\right) = \frac{F_t^y B_t^y}{\sum_{y'} F_t^{y'} B_t^{y'}}$$

$$\xi_t^{y,j} = \Pr\left(Y_t = y, Y_{t+1} = j | e_{1:T}^{1:S}, \Theta, \Phi\right) = \frac{F_t^y \theta_{y,j} \prod_s \phi_{j,e_{t+1},s} B_{t+1}^j}{B_t^i}$$

Then we can estimate the parameters using the following equations

$$\theta_{i,j} = \frac{\sum_{t=1}^{T} \xi_t^{i,j}}{\sum_{t=1}^{T} \gamma_t^i}, \quad \phi_{i,j,s} = \frac{\sum_{t=1}^{t} \delta(e_t^s = j)\gamma_t^i}{\sum_{t=1}^{T} \gamma_t^i} \tag{6.18}$$

In Appendix A.2.1 we describe how to avoid underflow issues in this computation.

6.4.3 Bayesian Learning for HMMs

As discussed before, we can use Bayesian priors for the observation and transition distributions for a full Bayesian treatment of this problem. In this section I will describe how we can use Gibbs Sampling to learn the parameters of the HMM.

6.4.4 Gibbs Sampling

As discussed previously, Monte Carlo Markov Chain (MCMC) methods are a popular class of methods that allow us to construct approximations of integrals that are difficult to compute analytically. The idea behind MCMC is to construct a Markov chain whose stationary distribution is the posterior distribution of interest. The algorithm simulates the chain until it gets close to stationarity. After this "burn-in" period, samples from the chain serve as an approximation to the true posterior distribution $P(\Theta|e)$. The most common variations of this approach are the Metropolis-Hastings algorithm and Gibbs sampling. Here, we use Gibbs sampling to approximate the posterior distribution. For a more extensive overview of MCMC techniques, see [30, 66].

Algorithm 7 Gibbs Sampling

```
1: Let Start with an arbitrary state path $y_{1:T}$ which has non zero probability
2: for $t = 1$ to $T$ do
3:     while $\Pr(Y_{1:T}|e_{1:T})$ has not converged do
4:         Sample $y_t$ from $P(Y_t = y|y_{1:t-1}, y_{t+1:T}, e_{1:T})$ according to Equation 6.20
5:     end while
6:     Generate $n$ samples from $\Pr(Y_{1:T}|e_{1:T})$
7:     For each sampled state path $y_{1:t}$ sample from $\Pr(\Theta, \Phi|Y_{1:T}, e_{1:T})$
8: end for
```

Gibbs sampling consists of repeatedly sampling each variable in the model from the conditional distribution given all the other variables. For example, if θ is a vector of parameters $\theta = \theta_1, \theta_2, ..., \theta_k$, the Gibbs sampling algorithm samples θ_1 from $\Pr(\theta_1|\theta_{2:k}, e)$ followed by sampling θ_2 from $\Pr(\theta_2|\theta_1, \theta_{3:k}, e)$ all the way to θ_k, after which we start again from sampling θ_1. It can be shown that the resulting Markov chain has stationary distribution $P(\theta_1, ..., \theta_k|e)$. In the case of HMMs, a straightforward Gibbs sampler would involve sampling all hidden states Y_i followed by sampling the parameters. Here, we use what is known as the *collapsed* Gibbs sampler [52] which integrates out the parameters and samples from $\Pr(Y_{1:T}|e_{1:T})$ to speed up the convergence. Once we have sampled the state paths, sampling the parameters can be done directly given the state paths.

As described in Chapter 4 we choose the prior distribution over the parameters to be a product of Dirichlets. The posterior becomes a mixture of exponentially many products of Dirichlets, each product corresponding to one possible state path. The joint distribution $\Pr(Y_{1:T}, e_{1:T})$ can be computed analytically using:

$$\begin{aligned}\Pr(y_{1:T}, e_{1:T}) &= \int_{\Theta\Phi} \Pr(y_{1:T}, e_{1:T}|\Theta, \Phi) \Pr(\Theta, \Phi) d\Theta d\Phi \\ &\propto \prod_y \frac{\prod_{y'} \Gamma(n_{y,y'})}{\Gamma(\sum_{y'} n_{y,y'})} \prod_y \frac{\prod_e \Gamma(n_{y,e})}{\Gamma(\sum_e n_{y,e})}\end{aligned} \tag{6.19}$$

where $n_{y,y'}$ is the number of transitions observed from state y to y', $n_{y,e}$ is the number of times e was observed given the state is y. Using $\Pr(Y_t = y|y_{1:t-1}, y_{t+1:T}, e_{1:T}) = \Pr(Y_t =$

$y, y_{1:t-1}, y_{t+1:T}, e_{1:T}) / \sum_{y'} P(Y_t = y', y_{1:t-1}, y_{t+1:T}, e_{1:T})$ and simplifying the terms above, we get:

$$P(Y_t = y | y_{1:t-1}, y_{t+1:T}, e_{1:T}) \propto \frac{n_{y_{t-1},y}}{\sum_{y'} n_{y_{t-1},y'}} \cdot \frac{n_{y,y_{t+1}}}{\sum_{y'} n_{y,y'}} \frac{n_{y,e_t}}{\sum_{e'} n_{y,e'}} \tag{6.20}$$

The algorithm is described in Algm 7. The Gibbs sampling procedure can be used both for learning the parameters and for inference about activities, since it produces samples of both state paths and parameter sets. Gibbs Sampling is a popular algorithm owing to its simplicity and ease of implementation. However one of the key issues with Gibbs Sampling is deciding on the convergence criteria i.e. when to stop sampling. Stopping at the wrong time may give us sub optimal estimates of the parameters. In the next Chapter I will describe an alternative technique of estimating the posterior based on moment matching.

6.5 Conditional Random Field

Conditional random fields are another class of probabilistic graphical models used for creating models of real world processes (CRFs, see [46]). In contrast to Hidden Markov Models, CRFs are discriminative, i.e., they model only the conditional distribution of the activities given the the sensor measurements. An important advantage of this approach is that no assumptions on the distribution of the sensor measurements need to be made. In these experiments, we have restricted ourselves to linear-chain CRFs. This subclass of CRF models is well suited for segmenting and labeling sequential data (such as the time-ordered sensor measurements in our experiments), and efficient algorithms for training and inference based on dynamic programming are available. Using the framework of linear-chain CRFs, the conditional probability of the sequence of user activities $y_{1:T}$, given the sequence of sensor measurements $e_{1:T}^{1:S}$, is given by

$$P_{\boldsymbol{\lambda}}(y_{1:T} | e_{1:T}^{1:S}) = \frac{1}{Z_{\boldsymbol{\lambda}}\left(e_{1:T}^{1:S}\right)} \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{f}\left(e_{1:T}^{1:S}, y_{1:T}\right)\right) \tag{6.21}$$

$$Z_{\boldsymbol{\lambda}}(e_{1:T}^{1:S}) = \sum_{\tilde{y}_{1:T}} \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{f}(e_{1:t}^{1:S}, \tilde{y}_{1:t})\right) \tag{6.22}$$

The key ingredient to Equation 6.22 is the inner product of model weights $\boldsymbol{\lambda}$ and feature functions $\boldsymbol{f}(e_{1:t}^{1:S}, y_{1:t})$ ($\cdot$ represents the inner product). Since the CRF has a linear-chain structure, there are two types of feature functions, namely, state and transition feature functions. The inner product of model weights and feature functions can be written as

$$\boldsymbol{\lambda} \cdot \boldsymbol{f}(e_{1:T}^{1:S}, y_{1:T}) = \sum_{t=1}^{T} \boldsymbol{\lambda}^{\text{state}} \cdot \boldsymbol{f}^{\text{state}}(e_t^{1:S}, y_t) \; + \; \sum_{t=2}^{T} \boldsymbol{\lambda}^{\text{trans}} \cdot \boldsymbol{f}^{\text{trans}}(y_{t-1}, y_t) \tag{6.23}$$

Intuitively, the weighted state feature function $\boldsymbol{\lambda}^{\text{state}} \cdot \boldsymbol{f}^{\text{state}}(e_t^{1:S}, y_t)$ relates the user activity at time t to the corresponding sensor measurements; the higher the particular value for $\boldsymbol{\lambda}^{\text{state}} \cdot \boldsymbol{f}^{\text{state}}(e_t^{1:S}, y_t)$, the more likely it is that the user is performing activity y_t at time t. Similarly, the weighted transition feature function $\boldsymbol{\lambda}^{\text{trans}} \cdot \boldsymbol{f}^{\text{trans}}(y_{t-1}, y_t)$ relates the user's activities at subsequent time points.

The state feature functions in this model are based on thresholding: for each pair of activities i, j and each sensor s, the actual observation e_t^s is compared to a fixed threshold value τ_{ij}^s. If the value is exceeded, i.e., if $e_t^s \geq \tau_{ij}^s$, then some model weight $\lambda_{ijs}^{(e)}$ is added, otherwise a different model weight $\lambda_{ijs}^{(n)}$ is added. The threshold values are chosen manually by a visual inspection of the data. If the labels i, j cannot be well discriminated by looking at the data from sensor s, then the threshold is chosen as the average value from sensor s; later on, such "non-informative" thresholds will be given very low weights in the training of the model.

A very simple transition model is used for the same reason as for HMMs (namely, we want to avoid a bias towards certain transitions due to the design of the walker course). In particular, the transition feature function is given by $\boldsymbol{f}^{\text{trans}}(y_{t-1}, y_t) = \delta(y_{t-1} = y_t)$, hence the corresponding model weight $\boldsymbol{\lambda}^{\text{trans}}$ is a scalar.

Training and Inference

Given a sequence of labeled training data, the most common approach to learn the weights of the CRF is maximizing the log-likelihood. The objective function is given by

$$\mathcal{L}(\boldsymbol{\lambda}) \;=\; \boldsymbol{\lambda} \cdot \boldsymbol{f}\left(e_{1:T}^{1:S}, y_{1:T}\right) - \log Z_{\boldsymbol{\lambda}}\left(e_{1:T}^{1:S}\right) - \frac{\boldsymbol{\lambda} \cdot \boldsymbol{\lambda}}{2\sigma^2} \tag{6.24}$$

The last term on the right hand side is a regularizer which penalizes large weights. Since the objective function $\mathcal{L}(\boldsymbol{\lambda})$ is concave, hence its unique maximum can be found using gradient-based search. The gradient of $\mathcal{L}$ in $\boldsymbol{\lambda}$ is given by

$$\nabla\mathcal{L}(\boldsymbol{\lambda}) \;=\; \boldsymbol{f}\left(e_{1:T}^{1:S}, y_{1:T}\right) \;-\; E_{\boldsymbol{\lambda}}\big(\boldsymbol{f}\left(e_{1:T}^{1:S}, Y_{1:T}\right)\big) \;-\; \frac{\boldsymbol{\lambda}}{\sigma^2} \tag{6.25}$$

where $E_{\boldsymbol{\lambda}}\big(\boldsymbol{f}\left(e^{1:S}_{1:T}, Y_{1:T}\right)\big)$ is the expectation of $\boldsymbol{f}\left(e^{1:S}_{1:T}, Y_{1:T}\right)$ under $P_{\boldsymbol{\lambda}}$, that is,

$$E_{\boldsymbol{\lambda}}\big(\boldsymbol{f}\left(e^{1:S}_{1:T}, Y_{1:T}\right)\big) = \sum_{\tilde{y}_{1:T}} P_{\boldsymbol{\lambda}}(Y_{1:T} = \tilde{y}_{1:T} \,|\, e^{1:S}_{1:T})\, \boldsymbol{f}\left(e^{1:S}_{1:T}, \tilde{y}_{1:T}\right) \tag{6.26}$$

We used the conjugate gradient method to find the maximum of $\mathcal{L}(\boldsymbol{\lambda})$, where we stopped the search after 100 Newton iterations.

The main challenge in evaluating $\mathcal{L}(\boldsymbol{\lambda})$ is the computation of the partition function $Z_{\boldsymbol{\lambda}}\left(e^{1:S}_{1:T}\right)$. Also for the evaluation of the gradient $\nabla\mathcal{L}(\boldsymbol{\lambda})$ the marginal distributions of $Y_{1:T}$ under $P_{\boldsymbol{\lambda}}$ need to be computed in order to determine $E_{\boldsymbol{\lambda}}\big(\boldsymbol{f}\left(e^{1:S}_{1:T}, Y_{1:T}\right)\big)$. Both problems can be efficiently solved using dynamic programming: Let $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_T$ be $(N+1)\times(N+1)$-matrices with the non-zero entries $\boldsymbol{M}_1(i,j) = \exp\left(\boldsymbol{\lambda}^{\text{state}} \cdot \boldsymbol{f}^{\text{state}}(e^{1:S}_1, j)\right)$ then

$$\boldsymbol{M}_t(i,j) = \exp\left(\boldsymbol{\lambda}^{\text{state}} \cdot \boldsymbol{f}^{\text{state}}(e^{1:S}_t, j) + \boldsymbol{\lambda}^{\text{trans}} \cdot \boldsymbol{f}^{\text{trans}}(i,j)\right)$$

Furthermore, let $\boldsymbol{F}_0 = (0,\ldots,0,1)$ and $\boldsymbol{B}_T = (1,\ldots,1,0)$ be $(N+1)$-dimensional row-vectors, and set $\boldsymbol{F}_t = \boldsymbol{F}_{t-1}\boldsymbol{M}_t$ and $\boldsymbol{B}'_t = \boldsymbol{M}_{t+1}\boldsymbol{B}'_{t+1}$ (where $\boldsymbol{B}'_t$ is the transpose of the vectors $\boldsymbol{B}_t$). Then the partition function can be written as

$$Z_{\boldsymbol{\lambda}}(e^{1:S}_{1:T}) = \boldsymbol{F}_T \cdot \boldsymbol{B}_T$$

and the marginals of $Y_{1:T}$ can be obtained by

$$P_{\boldsymbol{\lambda}}(Y_t = y_t, \ldots, Y_{t+k} = y_{t+k} | e^{1:S}_{1:T}) = \frac{\boldsymbol{F}_t(y_t)\cdot \boldsymbol{B}_{t+k}(y_{t+k})}{Z_{\boldsymbol{\lambda}}(e^{1:S}_{1:T})} \prod_{i=1}^{k} \boldsymbol{M}_{t+i}(y_{t+i-1}, y_{t+i})$$

where as usual the product over an empty index set is equal to 1 (see [46]).

Given the trained model and a sequence of observations, the labels can be predicted by the sequence $\tilde{y}_{1:T}$ which maximizes the a-posteriori probability of $Y_{1:T}$, that is, we compute

$$\tilde{y}_{1:T} = \arg\max_{y_{1:T}} P(Y_{1:T} = y_{1:T} \,|\, e^{1:S}_{1:T})$$

where the maximization is over all label sequences of length T. Note that $\tilde{b}_{1:T}$ can be computed efficiently similar to the Viterbi algorithm for HMMs (see [46]).

6.6 Results and Discussion

In this section, I present the analysis of the performance of the different algorithms discussed in this section on the activity recognition problem. I use leave one out cross validation for each round of training and prediction. Specifically, if we want to predict the activity sequence for a certain participant in Experiment 2, we learn the parameters from all other participants in Experiment 2. For Experiment 1, each person goes through the course twice. I have included one instance of the course in the training data along with data from other participants while testing for the same person. The range of sensor readings is divided into 20 intervals and their length is set in such a way that the same number of readings fall into each interval. Therefore $M = 20$.

Ground truth is established by hand labeling the data based on the video. This process is not perfect since the labeler can make mistakes while identifying activity transitions. For example, the labeler may interpret that a left turn started at time t, while the turn may actually start some time before t, but it only becomes evident in the video at time t. Therefore, in order to calculate error, I introduce the concept of window size. If the window size is x, then for the activity at time t, if we find the same activity in the window between time $t - x$ and time $t + x$ in the predicted sequence, we count it as a correct prediction. I vary our window size from 0 to 50 in intervals of 5. If we make a correct prediction within a window width of 25, then we are only off by half a second. Since older people perform activities at a rate that is much slower than half a second, this may still be considered accurate. Let the predicted value of the activity be $\tilde{y}$ and the actual value of the activity identified by a human be $\hat{y}$. Then, the accuracy can be defined as

$$Accuracy = \frac{\sum_{t=1}^{T} \delta\left(\hat{y}_t = \tilde{y}_t\right)}{T} \tag{6.27}$$

Note that random predictions would yield accuracy of $1/7 = 14\%$ in Experiment 1 and $1/13 = 7\%$ in Experiment 2.

We calculate the number of actual transitions as

$AT = \sum_{t=1}^{T} \delta\left(\hat{y}_t \neq \hat{y}_{t-1}\right)$

The number of predicted transitions can be calculated using

$PT = \sum_{t=1}^{T} \delta\left(\tilde{y}_t \neq \tilde{y}_{t-1}\right)$

Then, the number of correctly predicted transitions are calculated as

$CPT = \sum_{t=1}^{T} \delta\left(\tilde{y}_t \neq \tilde{y}_{t-1} \,\&\, \hat{y}_t = \tilde{y}_t \,\&\, \hat{y}_{t-1} = \tilde{y}_{t-1}\right)$

Then

$precision = CPT/PT$

and

$recall = CPT/AT$

Table 6.5: Confusion matrix for HMM model Experiment 1 activities using load sensor values. Observation model learned from data. Activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, Load cell values. Overall accuracy is 91.1%.

	NTW	ST	WF	TL	TR	WB	TRS	Accuracy %
NTW	8161	915	0	2	8	0	3409	65.3
ST	101	4488	2	6	1	33	448	88.4
WF	3	404	63455	885	1403	14	578	95.1
TL	6	10	223	12834	109	30	202	95.7
TR	0	52	164	107	6515	0	283	91.5
WB	0	54	0	106	23	8314	244	95.1
TRS	260	312	0	59	20	0	3661	84.9

The results are expressed in the form of a confusion matrix where each matrix entry at row i and column j indicates how many times activity i was predicted as activity j. I have reported results obtained by using two different sets of features for the HMM. The first set includes accelerometer measurements, speed and load sensor values while the second set includes accelerometer measurements, speed, frontal center of pressure, saggital center of pressure and total weight. The results obtained by doing supervised learning for the HMM with the first set of features are shown in Table 6.5 for Experiment 1 and Table 6.8 for Experiment 2. The results obtained by doing supervised learning for the HMM with the second set of features are shown in Table 6.6 for Experiment 1 and Table 6.9 for Experiment 2. The results obtained obtained by using the CRF with the center of pressure features are shown in Table 6.7 for Experiment 1 and Table 6.10 for Experiment 2. Table 6.11 shows the results obtained by doing supervised learning when the transition model is also learned form data for Experiment 2.

Tables 6.12 and 6.13 summarize the accuracy of all the algorithms for each activity.

In some situations, we are interested in identifying transitions from some activity to another. Figure 6.5 shows the precision and recall of activity transitions (in addition to recognition accuracy) as a function of the window size.

6.6.1 Discussion

In this section, we will analyze the results obtained from our experiments.

Table 6.6: Confusion matrix for HMM model for Experiment 1 using center of pressure. Observation model learned from data. Activity persistence parameter: $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 88%.

	NTW	ST	WF	TL	TR	WB	TRS	Accuracy %
NTW	9343	1055	0	15	35	0	2047	74.8
ST	159	4308	2	75	34	53	448	84.8
WF	10	342	59379	5361	1545	7	98	89.0
TL	0	58	322	12528	99	170	237	93.4
TR	0	15	185	425	6367	3	126	89.4
WB	30	8	0	76	0	8520	107	97.5
TRS	273	344	0	208	22	106	3359	77.9

Table 6.7: Confusion matrix for CRF model for Experiment 1 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and Total Weight. Overall accuracy is 93.8%.

	NTW	ST	WF	TL	TR	WB	TRS	Accuracy %
NTW	10722	102	0	0	9	26	335	74.8
ST	102	4193	140	147	4	153	399	84.8
WF	0	48	67599	864	317	1	20	89.0
TL	0	31	1172	12316	19	180	57	93.4
TR	0	54	805	381	5855	0	200	89.4
WB	0	248	6	111	29	8578	117	97.5
TRS	386	507	33	250	15	148	3105	77.9

Table 6.8: Confusion matrix for HMM model for Experiment 2 data using load sensor values. Observation model learned from data. activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measurements, Speed, load cell values. Overall accuracy is 79.1%.

	NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Accuracy %
NTW	4301	3270	0	38	0	740	102	40	17	0	33	3	50.3
ST	2175	36432	252	2786	451	846	7030	333	466	76	724	67	70.6
WF	52	2327	52256	3986	7347	618	2272	0	1010	576	585	388	73.2
TL	11	436	702	13647	505	309	573	29	264	243	110	38	80.9
TR	16	518	1426	691	11563	245	819	0	185	222	64	83	73.0
WB	0	77	11	209	70	221	335	0	77	3	42	0	21.1
RT	209	2879	420	954	623	318	6400	0	277	65	126	16	52.1
SW	38	108	60	120	55	16	56	70812	37	14	18	0	99.3
GUR	0	0	8	4	25	7	11	0	2580	116	247	0	86.1
GDR	0	32	41	40	123	2	23	0	2	2748	69	114	86.0
GUC	0	24	17	60	11	41	27	0	2	0	2888	17	93.6
GDC	0	5	54	14	102	30	1	0	16	8	31	1439	84.6

Table 6.9: Confusion matrix for HMM model for for Experiment 2 data using center of pressure. Observation model learned from data. activity persistence probability $\tau = 4000$. Prediction using Filtering. Window size is 25. Features used are Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 77.2%.

	NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Accuracy %
NTW	3962	3887	0	50	0	126	2	68	13	7	0	0	46.3
ST	3604	35073	1457	1815	788	1048	3893	509	807	166	646	146	67.9
WF	798	2146	50259	3040	7674	1251	2003	0	1271	420	477	238	70.4
TL	529	525	1892	11372	392	714	624	42	299	116	175	30	67.4
TR	50	330	1137	734	11748	695	578	0	140.0	197	133	90	74.2
WB	0	5	30	160	21	588	141	0	41.0	0	46	13	56.3
RT	502	3290	553	1077	494	495	5199	18	211.0	74	219	155	42.3
SW	26	129	16	177	49	0	94	70784	39.0	0	13	7	99.2
GUR	0	0	57	24	20	0	9	0	2528.0	113	132	115	84.3
GDR	0	17	32	142	98	22	40	0	8.0	2595	72	168	81.2
GUC	0	18	15	45	18	2	45	0	0.0	0	2937	7	95.1
GDC	0	18	59	12	127	6	54	0	0.0	9	19	1396	77.2

Table 6.10: Confusion matrix for CRF model for Experiment 2 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and total weight. Overall accuracy is 80.8%.

	NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Accuracy %
NTW	6674	1834	0	0	0	0	0	36	0.0	0	0	0	78.1
ST	1017	48677	1100	296	0	0	129	374	0.0	0	45	0	94.3
WF	0	2490	67738	744	270	0	25	112	0.0	0	38	0	94.8
TL	0	2081	5958	8528	97	0	72	117	0.0	12	2	0	50.6
TR	0	1517	10632	350	3129	0	62	34	0.0	69	39	0	19.8
WB	0	383	501	55	29	0	59	0	18.0	0	0	0	0.0
RT	387	7787	3071	327	24	0	634	46	11.0	0	0	0	5.2
SW	0	271	147	77	0	0	0	70839	0.0	0	0	0	99.3
GUR	0	173	1647	28	0	0	0	32	938.0	0	180	0	31.3
GDR	0	95	1753	0	575	0	0	0	0.0	720	41	10	22.5
GUC	0	436	375	0	89	0	122	0	0.0	0	2065	0	66.9
GDC	0	156	982	0	1	0	0	0	0.0	339	9	213	12.5

Table 6.11: Confusion matrix for HMM model results for Experiment 2 data using normalized load sensor values. Observation model and transition model learned from data. Prediction using Filtering. Window size is 25. Features used are Accelerometer Measuremens, Speed, Normalized load cell values. Overall accuracy is 80.8%.

	NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Accuracy %
NTW	4227	3356	1	38	0	709	117	40	24	0	32	0	49.5
ST	1777	37952	281	2485	458	649	6531	337	408	49	649	62	73.5
WF	36	2155	55910	3499	5814	468	1827	0	599	343	508	258	78.3
TL	8	473	769	13614	536	200	560	58	254	223	138	34	80.7
TR	20	562	1552	694	11515	190	806	0	146	198	70	79	72.7
WB	0	90	15	210	70	218	326	0	77	2	37	0	20.9
RT	207	3201	524	936	586	252	6116	0	249	61	140	15	49.8
SW	40	116	63	117	56	12	51	70812	38	13	16	0	99.3
GUR	0	0	36	4	26	8	11	0	2551	113	249	0	85.1
GDR	0	33	50	62	128	2	20	0	2	2723	70	104	85.3
GUC	0	25	22	59	11	34	30	0	1	0	2888	17	93.6
GDC	0	6	67	14	96	33	1	0	15	7	30	1431	84.2

Table 6.12: Experiment 1 percentage accuracy for each activity. COP implies that center of pressure feature is used instead of load cell values. Prediction is done using the Forward-Backward Algorithm

Window Size	Learning Algorithm	Percentage Accuracy							
		NTW	ST	WF	TL	TR	WB	TRS	Total
0	Supervised HMM NL	62.0	74.6	81.2	79.5	76.3	85.9	67.6	78.2
	Supervised HMM COP	69.6	59.9	78.2	71.5	72.6	89.2	42.6	74.9
	Supervised CRF	90.9	67.4	95.0	75.3	63.6	89.5	49.6	87.1
	Unsupervised HMM EM	85.3	6.1	57.3	48.1	56.5	59.1	19.0	55.9
	Unsupervised HMM Gibbs	98.1	6.8	92.5	58.2	57.3	41.1	12.2	75.2
25	Supervised HMM NL	65.1	87.0	95.1	94.4	90.3	94.4	84.1	90.7
	Supervised HMM COP	73.3	79.0	86.7	88.3	86.9	97.0	62.6	85.0
	Supervised CRF	95.8	81.6	98.2	89.4	80.3	94.4	69.9	93.8
	Unsupervised HMM EM	90.1	16.1	88.6	71.5	75.4	73.7	31.3	79.3
	Unsupervised HMM Gibbs	99.9	17.5	95.5	65.2	71.3	43.1	17.2	79.9

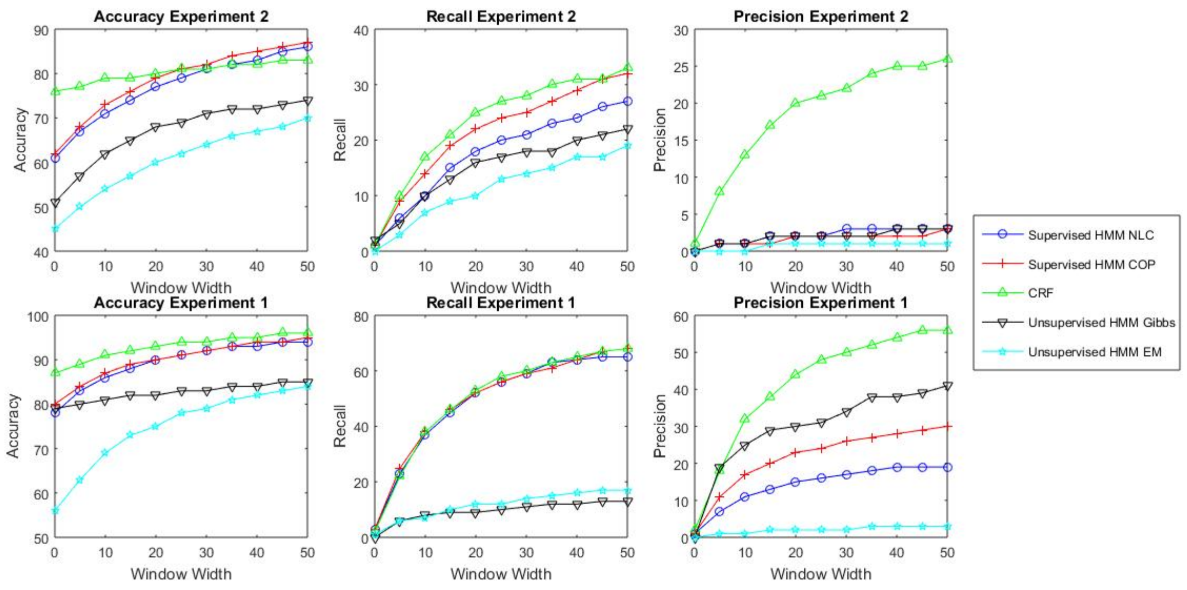


Figure 6.5: Accuracy, Precision and Recall for various algorithms in each experiment.

Table 6.13: Experiment 2 percentage accuracy for each activity. NL means the normalized load values are used. COP implies that center of pessure feature is used instead of normalized load values.

Window	Learning	Accuracy												
Size		NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Total
0	Supervised HMM NL	45.0	50.6	43.2	60.8	50.6	8.0	27.5	98.6	71.7	57.9	75.2	62.8	61.7
	Supervised HMM COP	45.6	50.0	43.5	50.4	48.6	25.6	21.3	98.3	75.5	56.5	70.3	53.6	59.2
	Supervised CRF	74.9	88.6	89.4	37.3	14.7	0.0	2.9	98.8	19.1	19.4	41.2	8.6	76.2
	Unsupervised HMM EM	75.6	30.0	31.2	46.9	21.2	38.0	28.5	68.7	8.3	4.6	4.8	22.6	42.8
	Unsupervised HMM Gibbs	90.0	35.1	35.0	45.6	62.3	0	50.2	87.5	39.6	25.6	37.9	0.1	51.0
25	Supervised HMM NL	49.1	66.1	70.8	78.7	69.6	11.0	42.8	99.3	82.8	81.4	93.0	82.9	76.4
	Supervised HMM COP	48.1	74.0	81.6	71.7	74.5	53.6	41.1	99.2	94.0	84.7	90.1	81.6	81.5
	Supervised CRF	78.1	94.3	94.8	50.6	19.8	0.0	5.2	99.3	31.3	22.5	66.9	12.5	80.8
	Unsupervised HMM EM	78.0	48.0	69.6	74.3	50.9	63.3	53.5	70.3	23.2	17.6	57.0	49.4	61.8
	Unsupervised HMM Gibbs	95.0	42.2	76.9	68.8	80.8	0	62.3	90.9	57.4	52.1	46.2	1.9	69.3

6.6.2 Experiment 1 vs. Experiment 2

It is obvious from Tables 6.12 and 6.13 and Figure 6.5 that the overall accuracy is much higher for Experiment 1. Difficult activities such as TR, TL and TRS are also predicted accurately. One reason for the high accuracy may be that in Experiment 1, each person executes the course twice and one instance of the course execution is included in the training data while testing for the same person. Therefore, the training data may include information that is more specific to the particular person e.g., how people put load on the walker for different activities. Secondly, for Experiment 1, activities such as NTW, ST, WF and WB are easily distinguishable from each other. Confusion is only likely between WF, TL and TR and between ST and TRS. There is a larger number of activities in Experiment 2 that are difficult to distinguish from each other based on sensor information e.g. WF, TL, TR, GUR, GDR, GUC and GDC. Similarly it is difficult to distinguish between RT and ST.

6.6.3 CRF vs. HMM

Tables 6.12 and 6.13 show that the total accuracy of the CRF is much higher than that of the HMM. This result is further validated by comparing Tables 6.6 and 6.7 and Tables 6.9 and 6.10. From Figure 6.5 it is obvious that the precision is higher and recall is lower for the CRF than for any other algorithm. This is largely due to the fact that the CRF models $Pr(Activity|Observations)$ directly and optimizes the parameters of this distribution. On the other hand, the HMM models $Pr(Observations|Activity)$ as well as $Pr(Activitys$, and then uses Bayes rule to calculate $Pr(Activity|Observations)$. Therefore, the HMM model is more complex and the number of parameters that we have to learn is larger. Also, the HMM makes an explicit assumption about the conditional independence of sensor measurements over time. The CRF avoids this (problematic) assumption since it does not model any distribution over the observations. However, techniques for unsupervised learning are better established for HMMs than CRFs. This becomes an important advantage since the data does not need to be labeled in unsupervised learning.

6.6.4 Maximum Likelihood vs. Bayesian Learning

As explained previously, manually labeling the data is time consuming and error-prone. Unsupervised learning algorithms avoid this problem. However, they take longer to converge and the solutions are usually approximations to the optimal parameters. One important difficulty with unsupervised learning is state matching. On one hand, unsupervised learning algorithms may pick sub-activities of composite activities as a state. For example, the ramp transition can be broken up into getting on the ramp (corresponding to a blip in the vertical acceleration), and walking on the ramp. The algorithm may find a state that matches either one or both instead of the going up/down ramp. On the other hand, if two activities are similar, the algorithm might treat them as the same state. I observed that in Table 6.12, TRS is almost never predicted correctly. This may be due to the fact that TRS is very similar to ST and hence they are merged into one state.

For EM and Gibbs sampling, once a model is learned, each latent state is associated with the activity that it is matched with most frequently. In general we used a number of latent states equal to the number of activities, except for Gibbs sampling in Experiment 1 (Table 6.12) where latent states (11) states were used than activities (7), the accuracy improved.

Since EM often gets stuck in local optima, I did 20 random restarts and showed the results of the model with the highest likelihood. In contrast, Gibbs sampling does not suffer from this problem. It is evident from Tables 6.12 and 6.13 as well as Figure 6.5 that Gibbs sampling performs better than EM.

In this chapter I have presented a novel and significant application of activity recognition in the context of instrumented walkers. I designed several algorithms based on HMMs and CRFs, and tested them with real users at the Village of Winston Park (retirement community in Kitchener, Ontario). A comprehensive analysis of the results showed that activities associated with walker usage tend to induce load, speed and acceleration patterns that are sufficient to distinguish them with reasonable accuracy. We have used a Hidden Markov Model for activity recognition. In future variants of HMMs such as semi-Markov HMMs can be used to model this problem as well. In semi-Markov HMMs, the persistence probability of an activity along-with the duration for which the activity persists are explicitly modeled. However this is beyond the scope of this thesis and is left as future work. In the next chapter we will describe an algorithm for activity recognition based on moment matching.

Chapter 7

Moment Matching For Activity Recognition

In the previous chapter I described how we can use probabilistic graphical models to do activity recognition using an HMM. Most of the algorithms discussed previously either require manual labeling of the data or else require the full data to be present in the memory. In this chapter I will describe a moment matching algorithm for learning the parameters of the HMM arising in the activity recognition domain in an online fashion. Each new observation is incorporated in a constant amount of time without needing to retrain the model.

7.1 Bayesian Learning for HMMs

In Figure 7.1 a Bayesian model for activity recognition is shown where the transition and observation parameters are sampled from Bayesian priors. The following generative process allows us to sample an observation sequence given a set of hyper-parameters for the priors. For each $y \in \{1, \ldots, T\}$ we first sample θ_y from a prior $f(\theta_y; \alpha_y)$. Then for each sensor s, we sample $\phi_{y,s}$ from another prior $f(\phi_{y,s}; \beta_{y,s})$. Then, each observation is generated by first sampling the current state y' from θ_y where y is the state sampled at the previous time step. Then for each sensor s, we sample an observation e from $\phi_{y',s}$. We will use the following additional terminology frequently for notational convenience

α set of hyper-parameters for the transition distribution

β set of hyperparameters for the observation distribution

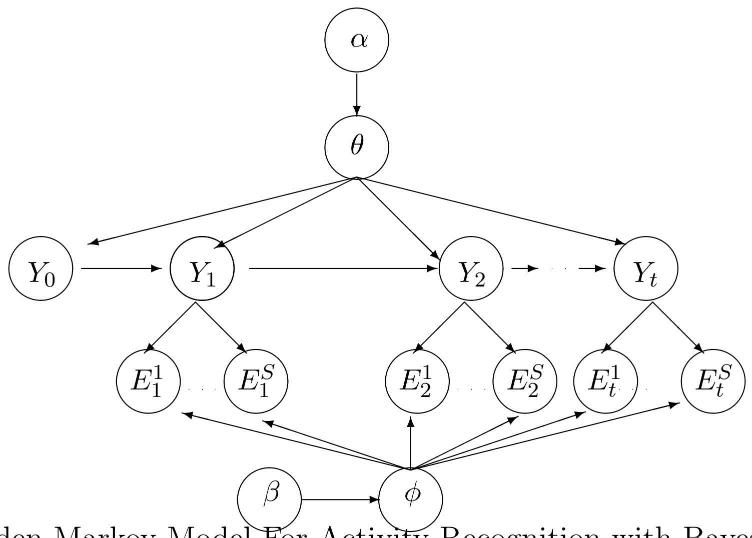


Figure 7.1: A Hidden Markov Model For Activity Recognition with Bayesian Priors

$P_t(\Theta,\Phi) = \Pr\left(\Theta,\Phi|e_{1:t}^{1:S}\right)$
$P_t^y(\Theta,\Phi) = \Pr\left(\Theta,\Phi|Y_t = y, e_{1:t}^{1:S}\right)$
$k_t^y = \Pr\left(Y_t = y, e_t^{1:S}|e_{1:t-1}^{1:S}\right)$
$k_t = \Pr\left(e_t^{1:S}|e_{1:t-1}^{1:S}\right)$
$c_t^y = \Pr\left(Y_t = y|e_{1:t}^{1:S}\right)$
$c_t = \Pr\left(e_{1:t}^{1:S}\right)$

For now we will assume that there is only 1 sensor. Later on I will extend the discussion to include more sensors. Given an observation sequence, we want to estimate Θ and Φ. Recall that the posterior $P_t(\Theta,\Phi)$ can be computed using

$$\Pr(\Theta,\Phi|e_{1:t}) = \sum_y P_t^y(\Theta,\Phi)\, c_t^y \tag{7.1}$$

where

$$k_t^y = \Pr(Y_t = y, e_t|e_{1:t-1}) = \int_{\Phi,\Theta} \phi_{y,e_t} \sum_i \theta_{i,y} c_{t-1}^i P_{t-1}^i(\Theta,\Phi)\, d\Phi d\Theta \tag{7.2}$$

$$k_t = \Pr(e_t|e_{1:t-1}) = \sum_a k_t^a \tag{7.3}$$

$$c_t^y = \Pr(Y_t = y|e_{1:t}) = k_t^y / k_t \tag{7.4}$$

$$c_t^y P_t^y(\Theta,\Phi) = \Pr(\Theta,\Phi,Y_t = y|e_{1:t}) = (1/k_t)\,\phi_{y,e_t} \sum_i \theta_{i,y} c_{t-1}^i P_{t-1}^i(\Theta,\Phi) \tag{7.5}$$

Let the prior $P_0(\Theta, \Phi) = f(\Theta, \Phi|\alpha, \beta)$ be some distribution in θs and ϕs where α and β are the parameters of the distribution f. Then according to Eq. 7.1 the number of terms in the posterior grows by a factor of $O(N^2)$. After t observations, the posterior will consist of a mixture of $O(N^{2t})$ terms, which is intractable to express.

7.2 Moment Matching for Hidden Markov Models

In Algm.8, I describe a moment matching technique for Bayesian learning of HMM parameters. This algorithm is based on Algm. 1 from Chapter 4. For each y the algorithm approximates the posterior P_t^y after seeing an observation with fewer terms in order to prevent the number of terms from growing exponentially. The posterior P_t^y is approximated with a simpler distribution Q_t^y by moment matching. More precisely, a set of moments sufficient to define Q_t^y are matched to the moments of the exact posterior P_t^y. Then, the exact posterior $P_t^y(\Theta, \Phi)$ is computed according to 7.5 based on the t^{th} observation. Then, for each P_t^y, compute the moments $S(f)$ are computed that are sufficient to define a distribution in the family f. For each $\mu \in S(f)$, the moments $M_\mu(P_t^y)$ of the posterior are computed exactly using

$$M_\mu(P_t^y) = \frac{\sum_i c_{t-1}^i M_{\mu\,\theta_{i,y}\phi_{y,e_t}}\left(P_{t-1}^i\right)}{\sum_i c_{t-1}^i M_{\theta_{i,y}\phi_{y,e_t}}\left(P_{t-1}^i\right)} \tag{7.6}$$

Algorithm 8 genericMomentMatching

```
1: Let $f(\Theta, \Phi|\alpha, \beta)$ be a family of distributions with parameters $\alpha$ and $\beta$
2: Initialize the prior $P_0(\Theta, \Phi)$
3: for $t = 1$ to $T$ do
4:     for $y = 1$ to $N$ do
5:         Compute $c_t^y P_t^y(\Theta, \Phi)$ from $c_{t-1}^a P_{t-1}^a(\Theta, \Phi)$ according to (Eq. 7.5)
6:         For $\forall \mu \in S(f)$, compute $M_\mu(P_t^y)$
7:         Compute $\alpha^y$ and $\beta^y$ from the $M_\mu(P_t^y)$'s
8:         Approximate $P_t^y$ with $Q_t^y(\Theta, \Phi) = f(\Theta, \Phi|\alpha^y, \beta^y)$
9:         Compute $P_t(\Theta, \Phi)$ using Equation 7.1
10:    end for
11: end for
```

$M_{\mu\,\theta_{i,y}\phi_{y,e_t}}\left(P_{t-1}^y\right)$ denotes the moment associated with the monomial $\mu\,\theta_{i,y}\phi_{y,e_t}$ obtained by multiplying $\mu(\Theta, \Phi)$ by $\theta_{i,y}$ and ϕ_{y,e_t}. Next, we compute the parameters α^y and β^y based on the set of sufficient moments (line 6)[1]. This determines a specific distribution Q_t^y in the family f that we use to approximate P_t^y. The key difference between this algorithm and the one described in Algm. 2 is that the moment matching is done for each posterior P_t^y separately and then we use 7.1 to compute the posterior P_t (line 9).

7.2.1 The Known Observation Model

We will first illustrate Algm. 8 with a simplified version of the problem in which the observation distribution Φ is known. This setting is useful in many domains, including activity recognition. The observation distribution can be estimated in isolation by asking participants to perform specific activities while recording the sensor measurements. In contrast, the transition distribution cannot be easily estimated in controlled experiments since sequences of activities must be unscripted and performed over a period of time that may range from hours to weeks. This yields an unsupervised learning problem.

Again we let $f(\Theta)$ be a product of Dirichlets such that $f(\Theta) = \prod_a Dir(\theta_a; \alpha_a)$. Recall that any moment M_μ where $\mu(\Theta) = \prod_i \prod_j \theta_{i,j}^{n_{i,j}}$ can be calculated using

$$M_\mu(f_y) = \int_\Theta \prod_i \prod_j \theta_{i,j}^{n_{i,j}} f(\theta) d\theta \qquad = \prod_i \frac{\Gamma(\alpha_{i,.})}{\Gamma(\alpha_{i,.} + \sum_b n_{i,b})} \prod_j \frac{\Gamma(\alpha_{i,j} + n_{i,j} - 1)}{\Gamma(\alpha_{i,j})} \tag{7.7}$$

Where $\alpha_{i,.} = \sum_j \alpha_{i,j}$. Similar to the procedure described in Chapter 4 the following sufficient set can be used to compute the parameters of the distribution

$$S(f) = \{\theta_{i,j}, \theta_{i,1}^2 | 1 \le i \le N, 1 \le j \le N-1\} \tag{7.8}$$

We can determine the parameters $\alpha_{i,j}$ of a Dirichlet based on those moments by setting up a system of N equations in N variables

$$M_{\theta_{i,j}}(f) = \frac{\alpha_{i,j}}{\alpha_{i,.}} \; 1 \le j \le N-1 \qquad\qquad M_{\theta_{i,1}^2}(f) = \frac{\alpha_{i,1}(\alpha_{i,1}+1)}{(\alpha_{i,.})(\alpha_{i,.}+1)}$$

This system of equations can be solved analytically as follows

$$\alpha_{i,j} = M_{\theta_{i,j}}(f) \frac{M_{\theta_{i,j}}(f) - M_{\theta_{i,1}^2}(f)}{M_{\theta_{i,1}^2}(f) - (M_{\theta_{i,j}}(f))^2} \tag{7.9}$$

[1]Since the parameters of P_t^y are different for different values of y, the superscript y on the αs and βs allows us the differentiate between these values.

Once we have the parameters $\alpha_{i,j}$, we can use Eq. 7.7 to calculate all other moments of this distribution. The sufficient set of moments of the exact posterior can be computed using the following equations

$$c_t^y M_\mu(P_t^y) = (1/k_t) \sum_i \phi_{y,e_t} c_{t-1}^i M_{\mu\,\theta_{i,y}}(P_{t-1}^a) \tag{7.10}$$

The key difference between this algorithm and the algorithm described in Chapter 5 lies in the fact that at every time step, we have y different posteriors P_t^y that are approximated by a product of Dirichlets of the form $f_y(\theta_y)$. Whereas in the case of LDA, there is only one posterior that is approximated. We will use $\alpha_{a,b}^y$ to denote the parameters for the y^{th} posterior P_t^y. For $j = \{1, \ldots, N-1\}$

$$\begin{aligned} c_t^y M_{\theta_{i,j}}(P_t^y) &= (1/k_t) \sum_a \phi_{y,e_t} c_{t-1}^a M_{\theta_{i,j}\theta_{a,y}}(P_{t-1}^a) && (7.11)\\ &= (1/k_t) \sum_a \phi_{y,e_t} c_{t-1}^a \frac{\alpha_{a,y}^a}{\alpha_{a,.}^a} \frac{\alpha_{i,j}^a + \delta(a,i)\delta(y,j)}{\alpha_{i,.}^a + \delta(a,i)} \\ c_t^y M_{\theta_{i,1}^2}(P_t^y) &= (1/k_t) \sum_a c_{t-1}^a \phi_{y,e_t} M_{\theta_{i,1}^2\theta_{a,y}}(P_{n-1}) && (7.12)\\ &= (1/k_t) \sum_a c_{t-1}^a \phi_{y,e_t} \frac{\alpha_{a,y}^a}{\alpha_{a,.}^a} \frac{\alpha_{i,1}^a + \delta(a,i)\delta(y,1)}{\alpha_{i,.}^a + \delta(a,i)} \frac{\alpha_{i,1} + \delta(a,i)\delta(y,1) + 1}{\alpha_{i,.}^a + \delta(a,i) + 1} \end{aligned}$$

We can then use Eq. 7.9 to determine the $\alpha_{a,b}$'s of the approximating distribution $Q_t^y(\Theta) = f(\Theta; \alpha)$. This process is summarized in Alg. 9.

The computational complexity of Alg. 9 is $O(TN^3)$. There are T iterations in which an observation is processed by updating the hyper-parameters in $O(N^3)$. This is an online algorithm since the data is processed in one sweep, the amount of computation to process each observation is independent of the amount of data. We obtain an estimate of the parameters (as an approximate posterior) after processing each observation.

7.2.2 Learning the Observation Model

Now lets consider the general case where the transition and observation distributions are all unknown. A natural choice for the family of approximating distributions is a product of Dirichlets of the form $f(\Theta, \Phi) = \prod_y Dir(\theta_y; \alpha_y) \prod_y Dir(\phi_y; \beta_y)$. Similar to LDA we have to deal with the label switching problem.

Algorithm 9 momentMatching (known Φ)

```
1:  Let $f(\theta|\alpha)$ be the family of Dirichlets
2:  Initialize $P_0(\theta) = \prod_y Dir(\theta_y; \alpha_y)$
3:  for $t = 1$ to $T$ do
4:      for $y = 1$ to $N$ do
5:          Compute $P_t^y(\theta)$ from $P_{t-1}^a(\theta)$ (Eq. 7.5)
6:          Compute $M_{\theta_{i,j}}(P_t^y)$ for $1 \le i \le N$ and $1 \le j \le N-1$ (Eq. 7.11)
7:          Compute $M_{\theta_{i,1}^2}(P_t^y)$ for $1 \le i \le N$ using Eq. 7.12
8:          Compute $\bar{\alpha}_{i,j}$ for $1 \le i \le N$ and $1 \le j \le N-1$ using Eq. 7.9
9:          Approximate $P_t^y$ with $Q_t^y(\theta) = \prod_a Dir(\theta_a; \bar{\alpha}_a^y)$
10:         Compute $P_t(\Theta, \Phi)$ using Equation 7.1
11:     end for
12: end for
```

Label Switching and Unidentifiability

I discussed label switching in details in 5 for LDA. HMMs also suffer from the same issue where the labels assigned to each hidden state can be permuted and we get a new set of parameters Θ and Φ. The model corresponding to this new set of parameters, can generate the observations $e_{1:t}$ with the same probability as the first model. The treatment of unidentifiability is slightly different from LDA since in LDA the moment update equation is defined in terms of the full posterior while in HMMs, the moment update is defined in terms of the joint distribution distribution $P(\Theta, \Phi, Y_t = y|e_{1:t})$. Each $P(\Theta, \Phi, Y_t = y|e_{1:t})$ has there are $N!$ components, each corresponding to a permutation of the hidden states. In the limit (i.e., infinite amount of data), the full posterior $\Pr(\Theta, \Phi|e_{1:t})$ will have $N!$ modes and therefore a unimodal approximation with a product of Dirichlets will not work well. In order to cater for each mode that is expected to arise from each permutation, we consider a mixture of products of Dirichlets. The parameters of each product of Dirichlets are permuted according to the hidden state labels assigned to that permutation. Let Σ^N be all permutations of the vector $1, \ldots, N$, then we can write

$$\Pr(\Theta, \Phi, Y_t = y|e_{1:t}) = \frac{1}{N!} \sum_{\sigma_i \in S^N} \Pr(\Theta, \Phi, Y_t = \sigma_i(y)|e_{1:t}) \quad = \frac{1}{N!} \sum_{\sigma_i} c_t^{\sigma_i(y)} P_t^{\sigma_i(y)}(\Theta, \Phi) \tag{7.13}$$

Here σ is a permutation of the vector $1, \ldots, N$ and $\sigma(y)$ is the label of the hidden state y in that permutation. f^σ is a product of Dirichlets corresponding to the permutation

σ.Each permutation is assigned the weight $1/N!$ so that the mixture sums up to 1. We approximate $P(\Theta, \Phi, Y_t = y|e_{1:t})$ with an f such that

$$P(\Theta, \Phi, Y_t = y|e_{1:t}) \approx f^y(\Theta, \Phi) = \frac{1}{N!} \sum_{\sigma_i \in S^N} c_t^{\sigma_i(y)} f^{\sigma_i(y)}(\Theta, \Phi) \tag{7.14}$$

This can be illustrated by the following example where $N = 2$ and $M = 3$. The conditional $P(\Theta, \Phi, Y_t = 1|e_{1:t})$ can be approximated by

$$\begin{aligned} f^1(\Theta, \Phi) = 1/2\, c_t^1 & Dir\left(\theta_{1,1}, \theta_{1,2}; \alpha_{1,1}^1, \alpha_{1,2}^1\right) Dir\left(\theta_{2,1}, \theta_{2,2}; \alpha_{2,1}^1, \alpha_{2,2}^1\right) \\ & Dir\left(\phi_{1,1}, \phi_{1,2}, \phi_{1,3}; \beta_{1,1}^1, \beta_{1,2}^1, \beta_{1,3}^1\right) Dir\left(\phi_{2,1}, \phi_{2,2}, \phi_{2,3}; \beta_{2,1}^1, \beta_{2,2}^1, \beta_{2,3}^1\right) \\ & + 1/2\, c_t^2 Dir\left(\theta_{1,1}, \theta_{1,2}; \alpha_{2,2}^2, \alpha_{2,1}^2\right) Dir\left(\theta_{2,1}, \theta_{2,2}; \alpha_{1,2}^2, \alpha_{1,1}^2\right) \\ & Dir\left(\phi_{1,1}, \phi_{1,2}, \phi_{1,3}; \beta_{2,1}^2, \beta_{2,2}^2, \beta_{2,3}^2\right) Dir\left(\phi_{2,1}, \phi_{2,2}, \phi_{2,3}; \beta_{1,1}^2, \beta_{1,2}^2, \beta_{1,3}^2\right) \end{aligned}$$

Similarly $P(\Theta, \Phi, Y_t = 2|e_{1:t})$ can be approximated by

$$\begin{aligned} f^2(\Theta, \Phi) = 1/2\, c_t^2 & Dir\left(\theta_{1,1}, \theta_{1,2}; \alpha_{1,1}^2, \alpha_{1,2}^2\right) Dir\left(\theta_{2,1}, \theta_{2,2}; \alpha_{2,1}^2, \alpha_{2,2}^2\right) \\ & Dir\left(\phi_{1,1}, \phi_{1,2}, \phi_{1,3}; \beta_{1,1}^2, \beta_{1,2}^2, \beta_{1,3}^2\right) Dir\left(\phi_{2,1}, \phi_{2,2}, \phi_{2,3}; \beta_{2,1}^2, \beta_{2,2}^2, \beta_{2,3}^2\right) \\ & + 1/2\, c_t^2 Dir\left(\theta_{1,1}, \theta_{1,2}; \alpha_{2,2}^1, \alpha_{2,1}^1\right) Dir\left(\theta_{2,1}, \theta_{2,2}; \alpha_{1,2}^1, \alpha_{1,1}^1\right) \\ & Dir\left(\phi_{1,1}, \phi_{1,2}, \phi_{1,3}; \beta_{2,1}^1, \beta_{2,2}^1, \beta_{2,3}^1\right) Dir\left(\phi_{2,1}, \phi_{2,2}, \phi_{2,3}; \beta_{1,1}^1, \beta_{1,2}^1, \beta_{1,3}^1\right) \end{aligned} \tag{7.15}$$

The full posterior $\Pr(\Theta, \Phi) = f^1(\Theta, \Phi) + f^2(\Theta, \Phi)$ then becomes symmetric and its odd central moments of symmetric distributions are zero and therefore cannot be used in moment matching. Similar to LDA, I consider a family of slightly asymmetric mixtures

$$f(\Theta, \Phi) = \sum_{\sigma_i \in S^N} w_{\sigma_i} f^{\sigma_i}(\Theta, \Phi) \tag{7.16}$$

I assign a weight $w_{\sigma_1} = ((N-1)! + 1)/(N! + (N-1)!)$ to the first component that is $(N-1)!$ higher than the weight $w_{\sigma_j} = 1/(N! + (N-1)!)\ \forall j \neq 1$ for the remaining components. In the next section I will show how this slight asymmetry ensures that all moments vary and therefore can be used in moment matching. Furthermore, we only need to store the hyper-parameters of one product of Dirichlets since all $N!$ components share the same (permuted) hyper-parameters.

Sufficient Moments

Since all mixture components f^{σ_i} share the same (permuted) hyper-parameters, f has a total of N^2 transition hyper-parameters and NMS observation hyperparameters. Thus we include $N^2 + NMS$ moments in the set of sufficient moments

$$S(f) = \{\theta_{a,y}, \theta^2_{a,1} | 1 \le a \le N, 1 \le y \le N-1\} \cup \{\phi_{y,e}, \phi^2_{y,1} | 1 \le y \le N, 1 \le e \le M-1\}$$

Using equations 7.13 and 7.14, we can set up a system of equations such that allow us to match the moments of each $\Pr(\Theta, \Phi, Y_t = \sigma_i(y) | e_{1:t})$ and $f^y(\Theta, \Phi)$, we get

$$\sum_{\sigma_i} w_{\sigma_i} c_t^{\sigma_i(y)} M_{\theta_{a,b}}\left(P_t^{\sigma_i(y)}\right) = \sum_{\sigma_i \in S^N} w_{\sigma_i} c_t^{\sigma_i(y)} M_{\theta_{a,b}}\left(f^{\sigma_i(y)}\right) \tag{7.17}$$

where $\sigma(y)$ is the index of y under the permutation σ. If we use symmetric weighing scheme, this system of equations will give us $N!$ equations for each moment that are all the same. However, if we use the assymetric weighing scheme, the moment matching equations become,

$$\begin{aligned} &\frac{(N-1)!+1}{N!+(N-1)!} M_{\theta_{a,b}}\left(P_t^{\sigma_1(y)}\right) + \frac{1}{N!+N-1!} \sum_{\sigma_i, i\neq 1} c_t^{\sigma_i(y)} M_{\theta_{a,b}}\left(P_t^{\sigma_i(y)}\right) \\ =&\frac{(N-1)!+1}{N!+(N-1)!} M_{\theta_{a,b}}\left(P_t^{\sigma_1(y)}\right) + \frac{1}{N!+N-1!} \sum_{\sigma_i, i\neq 1} c_t^{\sigma_i(y)} M_{\theta_{a,b}}\left(f^{\sigma_i(y)}\right) \end{aligned}$$

This will give us $N!$ equations that can be analytically solved to get

$$M_{\theta_{a,b}}\left(P_t^{\sigma_1(y)}\right) = M_{\theta_{a,b}}\left(f^{\sigma_1(y)}\right) \tag{7.18}$$

This is an important result as it shows that doing moment matching using even one permutation actually allows us to cater for all permutations.

The sufficient moments in θ are computed as follows:

$$M_{(\theta_{i,j})}(f^y) = \frac{\alpha^y_{i,j}}{\alpha^y_{i,.}} \tag{7.19}$$

$$M_{(\theta^2_{i,1})}(f^y) = \frac{\alpha^y_{i,j}}{\alpha^y_{i,.}} \frac{\alpha^y_{i,j}+1}{\alpha^y_{i,.}+1} \tag{7.20}$$

Similarly, using $\beta_{t,.} = \sum_w \beta_{t,w}$ the sufficient moments can be computed as:

$$M_{(\phi_{i,j})}(f^y) = \frac{\beta^y_{i,j}}{\beta^y_{i,.}} \tag{7.21}$$

$$M_{(\phi^2_{i,1})}(f^y) = \frac{\beta^y_{i,j}}{\beta^y_{i,.}} \frac{\beta^y_{i,j}+1}{\beta^y_{i,.}+1} \tag{7.22}$$

For matching the moments with respect to θ, we can use Equations 7.11 and 7.12. For ϕ, we can use the following set of equations

$$\begin{aligned} c^y_t M_{\phi_{i,j}}(P^y_t) &= (1/k_t) \sum_a c^a_{t-1} M_{\phi_{i,j}\theta_{a,y}\phi_{y,e_t}}(P^a_{t-1}) && (7.23) \\ &= (1/k_t) \sum_a c^a_{t-1} \frac{\alpha^a_{a,y}}{\alpha^a_{a,.}} \frac{\beta^a_{y,e}}{\beta^a_{y,.}} \frac{\beta^a_{i,j} + \delta(y,i)\delta(i,e)}{\beta^a_{i,.} + \delta(y,i)} \\ c^y_t M_{\phi_{i,j}}(P^y_t) &= (1/k_t) \sum_a c^a_{t-1} M_{\phi^2_{i,1}\theta_{a,y}\phi_{y,e_t}}(P^a_{t-1}) && (7.24) \\ &= (1/k_t) \sum_a c^a_{t-1} \frac{\alpha^a_{a,y}}{\alpha^a_{a,.}} \frac{\beta^a_{y,e}}{\beta^a_{y,.}} \frac{\beta^a_{i,j} + \delta(y,i)\delta(i,e)}{\beta^a_{i,.} + \delta(y,i)} \frac{\beta^a_{i,j} + \delta(y,i)\delta(i,e) + 1}{\beta^a_{i,.} + \delta(y,i) + 1} \end{aligned}$$

Then the new counts can be computed using

$$\beta_{i,j} = M_{\phi_{i,j}}(f) \frac{M_{\phi_{i,j}}(f) - M_{\phi^2_{i,1}}(f)}{M_{\phi^2_{i,1}}(f) - (M_{\phi_{i,j}}(f))^2} \tag{7.25}$$

Alg. 10 summarizes Bayesian moment matching for the case where both Θ and Φ are estimated. The computational complexity of Alg. 10 is $O(T \times (N^4 + N^3 M))$. There are T iterations in which an observation is processed by updating the hyper-parameters of the approximating mixtures in $O(N^4 + N^3 M)$ time. The main bottleneck is the computation of all moments, each of which takes $O(N)$ time. In the next section I will describe a method for doing this computation efficiently.

7.2.3 Efficient Moment Matching

As we get more observations, $\Pr(\Theta, \Phi | Y_t = y, e_{1:t})$ becomes the same for all values of y. Therefore, we can make an approximation such that $\Pr(\Theta, \Phi | Y_t = y, e_{1:t}) \approx \Pr(\Theta, \Phi | e_{1:t})$.

Algorithm 10 momentMatching (Θ and Φ unknown)

```
1: Let $f(\theta|\alpha)$ be the family of Dirichlets
2: for $t = 1$ to $T$ do
3:   for $y = 1$ to $N$ do
4:     Compute $M_{\theta_{i,j}}(P_t^y)$ for $1 \le i \le N$ and $1 \le j \le N-1$ (Eq. 7.11)
5:     Compute $M_{\theta^2_{i,1}}(P_t^y)$ for $1 \le i \le N$ using Eq. 7.12
6:     Compute $M_{\phi_{i,j}}(P_t^y)$ for $1 \le i \le N$ and $1 \le j \le M$ (Eq. 7.23)
7:     Compute $M_{\phi^2_{i,1}}(P_t^y)$ for $1 \le i \le N$ using Eq. 7.12
8:     Compute $\bar{\alpha}_{i,j}$ for $1 \le i \le N$ and $1 \le j \le N-1$ using Eq. 7.9
9:     Compute $\bar{\beta}_{i,j}$ for $1 \le i \le N$ and $1 \le j \le M$ using Eq. 7.25
10:    Approximate $P_t^y$ with $Q_t^y(\Theta,\Phi) = \prod_a Dir(\theta_a; \bar{\alpha}_a^y) \prod_b Dir(\phi_b; \bar{\beta}_b^y)$
11:    Compute $P_t(\Theta,\Phi)$ using Equation 7.1
12:  end for
13: end for
```

This gives us a simplified set of equations for the moment update.

$$\Pr(\Theta,\Phi|e_{1:t}) = (1/k_t) \sum_y \sum_i \theta_{i,y} \phi_{y,e_t} c_{t-1}^i \Pr(\Theta,\Phi|e_{1:t-1}) \tag{7.26}$$

$$k_t^y = \Pr(Y_t = y, e_t|e_{1:t-1}) = \int_{\Phi,\Theta} \phi_{y,e_t} \sum_i \theta_{i,y} c_{t-1}^i P_{t-1}^i(\Theta,\Phi)\, d\Phi d\Theta \tag{7.27}$$

$$k_t = \Pr(e_t|e_{1:t-1}) = \sum_a k_t^a \tag{7.28}$$

$$c_t^y = \Pr(Y_t = y|e_{1:t}) = k_t^y / k_t \tag{7.29}$$

The moment update equation for θ can be written as

$$\begin{aligned}
M_{\theta_{a,b}}\left(P_t\right) &= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1} M_{\theta_{a,b}\theta_{i,y}\phi_{y,e_t}}\left(P_{t-1}\right) \\
&= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1}\frac{\beta_{y,e_t}}{\beta_{y,.}}\frac{\alpha_{i,y}}{\alpha_{i,.}}\frac{\alpha_{a,b}+\delta(a,i)\delta(b,y)}{\alpha_{a,.}++\delta(a,i)} \\
&= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1}\frac{\beta_{y,e_t}}{\beta_{y,.}}\frac{\alpha_{i,y}}{\alpha_{i,.}}\left[\frac{\alpha_{a,b}+\delta(a,i)\delta(b,y)}{\alpha_{i,.}+\delta(a,i)}+\delta(a,i)\frac{\alpha_{a,b}}{\alpha_{a,.}}-\delta(a,i)\frac{\alpha_{a,b}}{\alpha_{a,.}}\right] \\
&= \frac{\alpha_{a,b}}{\alpha_{a,.}}\left\{1+\sum_y\frac{\beta_{y,e_t}}{\beta_{y,.}}\left[\frac{\alpha_{a,b}+\delta(b,y)}{\alpha_{a,.}+1}-\frac{\alpha_{a,b}}{\alpha_{a,.}}\right]\frac{c^a_{t-1}}{k_t}\right\} \\
&= M_{\theta_{a,b}}\left(P_{t-1}\right)\left\{1+\sum_y M_{\phi_{y,e_t}}\left(P_{t-1}\right)\left[\frac{\alpha_{a,b}+\delta(b,y)}{\alpha_{a,.}+1}-\frac{\alpha_{a,b}}{\alpha_{a,.}}\right]\frac{c^a_{t-1}}{k_t}\right\} \\
&= M_{\theta_{a,b}}\left(P_{t-1}\right)\left\{1+\frac{1}{\alpha_{a,.}+1}\left[M_{\phi_{b,e_t}}\left(P_{t-1}\right)-M_{\theta_{a,b}}\left(P_{t-1}\right)\sum_y M_{\phi_{y,e_t}}\left(P_{t-1}\right)\right]\frac{c^a_{t-1}}{k_t}\right\}
\end{aligned} \tag{7.30}$$

Since we can compute $\sum_y M_{\phi_{y,e_t}}\left(P_{t-1}\right)$ once for each observation, each moment can now be updated in $O(1)$ time.

For the observation distribution,

$$\begin{aligned}
M_{\phi_{a,b}}\left(P_t\right) &= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1} M_{\phi_{a,b}\theta_{i,y}\phi_{y,e_t}}\left(P_{t-1}\right) \\
&= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1}\frac{\beta_{y,e_t}}{\beta_{y,.}}\frac{\alpha_{i,y}}{\alpha_{i,.}}\frac{\beta_{a,b}+\delta(a,y)\delta(b,e_t)}{\beta_{a,.}++\delta(a,y)} \\
&= \frac{1}{k_t}\sum_y\sum_i c^i_{t-1}\frac{\beta_{y,e_t}}{\beta_{y,.}}\frac{\alpha_{i,y}}{\alpha_{i,.}}\frac{\beta_{a,b}+\delta(a,y)\delta(b,e_t)}{\beta_{a,.}++\delta(a,y)}-\delta(a,y)\sum_i c^i_{t-1}\frac{\beta_{y,e_t}}{\beta_{y,.}}\frac{\alpha_{i,y}}{\alpha_{i,.}}\frac{\beta_{a,b}}{\beta_{a,.}} \\
&= M_{\phi_{a,b}}\left(P_{t-1}\right)-M_{\phi_{a,b}}\left(P_{t-1}\right)\frac{c^a_t}{\beta_{a,.}+1}+\delta(b,e_t)\frac{c^a_t}{\beta_{a,.}+1}
\end{aligned} \tag{7.31}$$

Let

$$X^a_t = \left[1-\frac{c^a_t}{\beta_{a,.}+1}\right] \tag{7.32}$$

$$Y^a_t = \frac{c^a_t}{\beta_{a,.}+1} \tag{7.33}$$

Then

$$M_{\phi_{a,b}}(P_t) = \left[M_{\phi_{a,b}}(P_{t-1}) + \delta(b, e_t)\frac{Y_t^a}{X_t^a}\right] X_t^a \tag{7.34}$$

$$\begin{cases} = \left[M_{\phi_{a,b}}(P_{t-1})\right] X_t^a & if\ b = e_t \\ = \left[M_{\phi_{a,b}}(P_{t-1}) + \delta(b, e_t)\frac{Y_t^a}{X_t^a}\right] X_t^a & otherwise \end{cases} \tag{7.35}$$

Note that in this equation, the only term dependent on b is $\delta(b, e_t)\frac{Y_t^a}{X_t^a}$. Therefore, we do not have to update the whole observation matrix at every time step. In fact, at every time step, we can do the following upate

$$X_t^a = \left[1 - \frac{c_t^a}{\beta_{a,.} + 1}\right] X_{t-1}^a \tag{7.36}$$

$$\overline{M}_{\phi_{a,b}}(P_t) = \left[\overline{M}_{\phi_{a,b}}(P_{t-1}) + \delta(b, e_t)\frac{Y_t^a}{X_t^a}\right] \tag{7.37}$$

$$M_{\phi_{a,b}}(P_{t-1}) = \overline{M}_{\phi_{a,b}}(P_{t-1}) \times X_t^a \tag{7.38}$$

$$\beta_{a,b} = \overline{M}_{\phi_{a,b}}(P_{t-1}) \times X_t^a \times b_{a,.} \tag{7.39}$$

The second order moments can be computed using $\beta_{a,b}$ and $\beta_{a,.}$ and Eq. 7.24. This algorithm is summarized in Algm. 11. The complexity of this algorithm is $O(TN^2)$. For each observation, we compute $O(N^2)$ θ moments each of which is computed in $O(1)$ time. The observation distribution can be updated in $O(N)$ time by updating only N X_as and $\overline{M}_{\phi_{y,e_t}}(P_t)$.

Algorithm 11 momentMatching (Θ and Φ unknown)

```
1: Let $f(\theta|\alpha)$ be the family of Dirichlets
2: Initialize $X_0^i = 1 \;\; \forall 1 \le i \le N$
3: for $t = 1$ to $T$ do
4:     Compute $M_{\theta_{i,j}}(P_t)$ for $1 \le i \le N$ and $1 \le j \le N-1$ (Eq. 7.30)
5:     Compute $M_{\theta^2_{i,1}}(P_t)$ for $1 \le i \le N$ using Eq. 7.12
6:     Compute $X_t^i$ using Eq. 7.37 for all $1 \le i \le N$
7:     Compute $\overline{M}_{\phi_{i,e_t}}(P_t)$ for $1 \le i \le N$ (Eq. 7.37)
8:     Compute $M_{\phi^2_{i,1}}(P_t)$ for $1 \le i \le N$ using Eq. 7.24
9:     Compute $\alpha_{i,.}$ for $1 \le i \le N$ using Eq. 7.9
10:    Compute $\beta_{i,.}$ for $1 \le i \le N$ using Eq. 7.25
11: end for
```

7.2.4 Multiple Sensors

In the Activity Recognition HMM described in Fig. 7.1 there are multiple observations at each time step coming from multiple sensors. The full posterior then becomes

$$\Pr(\Theta, \Phi | e_{1:t}) = (1/k_t) \sum_y \sum_i \theta_{i,y} \prod_s \phi_{y,e_t} c^i_{t-1} \Pr(\Theta, \Phi | e_{1:t-1}) \tag{7.40}$$

$$k^y_t = \Pr(Y_t = y, e_t | e_{1:t-1}) = \int_{\Phi,\Theta} \prod_s \phi_{y,e_t,s} \sum_i \theta_{i,y} c^i_{t-1} P^i_{t-1}(\Theta, \Phi)\, d\Phi d\Theta \tag{7.41}$$

$$= \prod_s \frac{\beta_{y,e_t,s}}{\beta_{y,.,s}} \sum_i \frac{\alpha_{i,y}}{\alpha_{i,.}}$$

$$k_t = \sum_y k^y_t$$

$$c^y_t = \frac{k^y_t}{k_t}$$

Where $\beta i, ., s = \sum_e \beta i, e, s$. The equations for the moments with respect ot θ remain the same. For the observation distribution, we can extend the equations derived before to handle multiple sensors

$$X_t^a(s) = \left[1 - \frac{c_t^a}{\beta_{a,.,s}+1}\right] X_{t-1}^a(s) \tag{7.42}$$

$$\overline{M}_{\phi_{a,b,s}}(P_t) = \left[\overline{M}_{\phi_{a,b,s}}(P_{t-1}) + \delta(b,e_t)\frac{Y_t^a(s)}{X_t^a(s)}\right] \tag{7.43}$$

$$M_{\phi_{a,b}}(P_{t-1}) = \overline{M}_{\phi_{a,b,s}}(P_{t-1}) \times X_t^a(s) \tag{7.44}$$

$$\beta_{a,b,s} = \overline{M}_{\phi_{a,b,s}}(P_{t-1}) \times X_t^a(s) \times b_{a,.,s} \tag{7.45}$$

Algorithm 12 momentMatching (multiple sensors)

```
1: Let $f(\theta|\alpha)$ be the family of Dirichlets
2: Initialize $X_0^i(s) = 1 \;\; \forall 1 \le i \le N \;\; 1 \le s \le S$
3: for $t = 1$ to $T$ do
4:     Compute $M_{\theta_{i,j}}(P_t)$ for $1 \le i \le N$ and $1 \le j \le N-1$ (Eq. 7.30)
5:     Compute $M_{\theta^2_{i,1}}(P_t)$ for $1 \le i \le N$ using Eq. 7.12
6:     Compute $X_t^i(s)$ using Eq. 7.37 for all $1 \le i \le N \;\; 1 \le s \le S$
7:     Compute $\overline{M}_{\phi_{i,e_t,s}}(P_t)$ for $1 \le i \le N \;\; 1 \le s \le S$ (Eq. 7.37)
8:     Compute $M_{\phi^2_{i,1,s}}(P_t)$ for $1 \le i \le N \;\; 1 \le s \le S$ using Eq. 7.24
9:     Compute $\alpha_{i,.}$ for $1 \le i \le N$ using Eq. 7.9
10:    Compute $\beta_{i,.,s}$ for $1 \le i \le N \;\; 1 \le s \le S$ using Eq. 7.25
11: end for
```

7.3 Discussion

Moment matching allows us to compute the counts of approximate posterior distribution in a constant amount of time. The time complexity of Algm. 12 is $O(N^2 + NS)$. It is also easy to extend the discussion from LDA to the HMM and make a case that there exists an implicit prior for which exact Bayesian updates will yield a posterior that has the same first and second order moments as the posterior retrieved by Bayesian Moment Matching. Initially, we specify some moments of the prior. After receiving an observation, we do an exact Bayesian update and then do moment matching. The process of moment matching allows us to set higher order moments in the posterior. We can easily set up a system of equations such that the moments of the posterior can be used to compute higher order moments in the previous time step. As new observations come, we use Bayesian Moment matching to set higher order moments in the prior.

Table 7.1: Moment Matching results for Experiment 1 data using center of pressure. Window size is 25. Features used are Accelerometer Measurements, Speed, Frontal plane COP, Saggital plane COP and Total Weight. Overall accuracy is 76.2%.

	NTW	ST	WF	TL	TR	WB	TRS	Accuracy %
NTW	11409	68	0	1	0	545	247	92.9
ST	2070	681	67	326	60	1543	191	13.8
WF	1187	1090	52325	1222	4080	3685	549	81.5
TL	566	326	547	9213	469	1691	123	71.4
TR	25	2	778	262	5305	465	98	71.4
WB	1366	100	0	604	243	6025	148	71.0
TRS	1634	265	77	431	222	790	694	16.9

7.4 Experiments and Results

I have used Bayesian Moment Matching to compute the parameters of the activity recognition HMM. The same experimental setup is used as described in Section 6.6. I use leave one out cross validation where for each sequence i in our data set, the parameters are learned from all sequences other than i and then use these parameters to do inference. The definitions of prediction accuracy, precision and recall are the same as the ones used in the previous chapter.

First I show how well the algorithm performs on synthetic data. For synthetic data, I set the number of states to be 8, the number of sensors to be 6 and the total readings on each sensor to be 15. I generated 20 sequences, each of length 20,000. The experiments was set up such that there is at least one sensor for which the observation matrix is full-rank. i.e. for each state, there exists an observation e such that $\Pr(e|Y_t = y) > \Pr left(e|Y_t = y') \ \forall y' \neq y$. In terms of prediction accuracy, moment matching performs almost as well as if we do inference using the actual parameters that generated the data. The overall accuracy if using the actual parameters is 95.5% whereas the for moment matching it is 94.7%. The prediction accuracy for the EM algorithm is 94.9%. The average training time for the EM algorithm was 2301.5 seconds while the average training time for moment matching was 302.4 seconds.

I also use the moment matching algorithm to do prediction accuracy, precision and recall analysis on the Walker data. Table 7.1 shows a confusion matrix for Experiment 1 data where the parameters are learnt using moment matching. It is easy to see that the prediction accuracy is comparable with that of EM and Gibbs Sampling.

Table 7.2: Experiment 1 percentage accuracy for each activity for unsupervised learning techniques. Prediction is done using the Forward-Backward Algorithm

Window Size	Learning Algorithm	Percentage Accuracy							
		NTW	ST	WF	TL	TR	WB	TRS	Total
0	Expectation Maximization	85.3	6.1	57.3	48.1	56.5	59.1	19.0	55.9
	Gibbs Sampling	98.1	6.8	92.5	58.2	57.3	41.1	12.2	75.2
	Moment Matching	91.6	8.2	66.5	50.2	63.3	53.7	12.9	73.1
25	Expectation Maximization	90.1	16.1	88.6	71.5	75.4	73.7	31.3	79.3
	Gibbs Sampling	99.9	17.5	95.5	65.2	71.3	43.1	17.2	79.9
	Moment Matching	93.1	13.7	81.5	71.2	76.3	71.1	16.8	82.0

Table 7.3: Experiment 2 percentage accuracy for each activity. NL means the normalized load values are used. COP implies that center of pessure feature is used instead of normalized load values.

	Learning	Accuracy												
		NTW	ST	WF	TL	TR	WB	RT	SW	GUR	GDR	GUC	GDC	Total
0	EM	75.6	30.0	31.2	46.9	21.2	38.0	28.5	68.7	8.3	4.6	4.8	22.6	42.8
	Gibbs	90.0	35.1	35.0	45.6	62.3	0	50.2	87.5	39.6	25.6	37.9	0.1	51.0
	MM	86.7	31.5	41.9	49.1	23.5	9.7	26.9	76.4	18.9	13.5	36.9	51.1	49.1
25	EM	78.0	48.0	69.6	74.3	50.9	63.3	53.5	70.3	23.2	17.6	57.0	49.4	61.8
	Gibbs	95.0	42.2	76.9	68.8	80.8	0	62.3	90.9	57.4	52.1	46.2	1.9	69.3
	MM	89.3	43.8	67.2.9	68.2	38.8	26	38.3	81.9	41.1	30.1	58.9	82.4	63.3

Table 7.2 shows the prediction accuracy for each activity on the experiment 1 data for unsupervised algorithms. We see that the prediction accuracy of Moment matching is comparable to that of the offline algorithms and actually surpasses the offline algorithms for certain activities. Table 7.3 shows the prediction accuracy for each activity on the experiment 2 data for unsupervised algorithms. I also plot the accuracy, precision and recall for the unsupervised algorithms in Fig. 7.2. Moment matching performs well compared to offline techniques while giving us the advantage of adding new data as it arrives with constant amount of work.

The average training time over all folds for each algorithm is summarized in Table 7.4. For EM if the change in likelihood is below a certain tolerance, the algorithm declares convergence. Therefore the training time between folds varies greatly. In our implementation of Gibbs Sampling, we do 500 iterations for every fold. It is easy to see that moment

Table 7.4: Average training time for unsupervised algorithms

DataSet/Algorithm	Moment Matching	Gibbs Sampling	Expectation Maximization
Synthetic	333.5	3460.1	2310.5
Experiment 1	91.5	7205.1	2564.3
Experiment 2	423.5	54302	28524

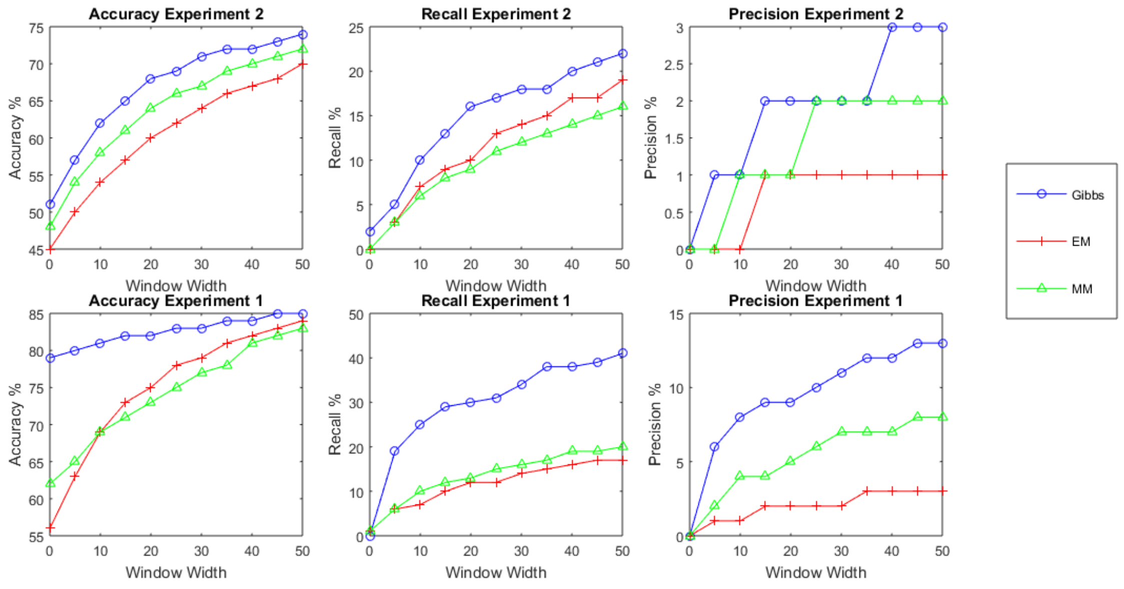


Figure 7.2: Accuracy, Precision and Recall for various algorithms in each experiment.

matching has an advantage in terms of time.

7.4.1 Comparison with Online EM

An online version of the Expectation Maximization algorithm for Hidden Markov Models has been proposed by Cappe et. al [21]. The HMM they consider has discrete states and continuous observations with a Gaussian observation distribution and a single sensor. In their work, Cappe et. al. provide a limited analysis of their algorithm using synthetic data with two hidden states. I derived an Online EM algorithm based on their approach that works for discrete observations for a single sensor. I compared their algorithm with Moment Matching on a synthetic dataset which consists of 10 sequences each of length 10,000

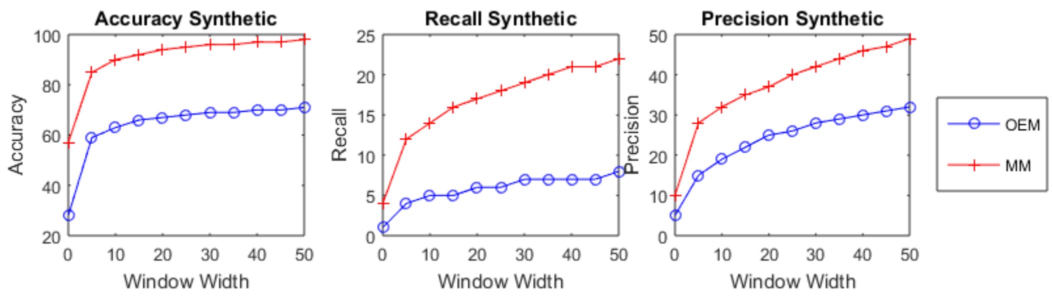


Figure 7.3: Accuracy, Precision and Recall for Online EM and Moment Matching on synthetic data.

each. We use leave one out cross validation for evaluation. The average likelihood of each sequence for Online EM is -27033.86053 while that of moment matching is -26457.78912. The precision, recall and accuracy analysis is shown in Figure 7.3. It is easy to see that moment matching surpasses online EM in terms of prediction accuracy. The likelihood of data is also higher in the case of moment matching. This shows that moment matching has considerable advantage over this approach. This maybe due to the fact that in online EM, a lower bound of the negative log likelihood is minimized by using stochastic optimization. In some sense this is itself an approximation of an approximation which explains why moment matching outperforms online EM.

In this chapter I have described an unsupervised online technique for learning the parameters of the activity recognition HMM described in the previous section. I derive an efficient $O(N^2)$ algorithm for parameter learning. I have also shown how the algorithm is able to deal with label switching. The algorithm is compared with established offline techniques for parameter learning. Moment matching holds its own against these algorithms in terms of accuracy and performs better than EM in some scenarios.

Chapter 8

Conclusions

This thesis presents an online Bayesian moment matching technique that incrementally estimates the parameters of a Probabilistic Graphical Model by performing a single sweep of the data. The approach is simple to implement and understand. the approach is tested on two real world applications i.e. Topic Modeling and Activity Recognition and compares favorably to existing approaches in terms of time, perplexity and prediction accuracy.

The Bayesian Moment Matching for LDA is able to estimate the parameters of the model in an online fashion. The algorithm is able to incorporate each new observation in time that is linear in the number of topics which is a significant improvement over the current Assumed Density Filtering algorithm. The algorithm compares favourably with other offline and online state of the art techniques both in terms of time and perplexity and shows significant gains for some data sets.

This thesis also presents a novel and significant application of activity recognition in the context of instrumented walkers. We designed several algorithms based on HMMs and CRFs, and tested them with real users at the Village of Winston Park (retirement community in Kitchener, Ontario). A comprehensive analysis of the results showed that activities associated with walker usage tend to induce load, speed and acceleration patterns that are sufficient to distinguish them with reasonable accuracy. This work can be turned into a clinical tool that can be used to assess the mobility patterns of walker users and the contexts in which they are more likely to fall.

In addition to this analysis, an algorithm for parameter learning in HMMs using Bayesian Moment Matching techniques is derived. The experiments against state of the art offline and online techniques show very promising results both in terms of prediction accuracy and time. In fact Moment Matching adapts itself beautifully to the time-series scenario and the implicit prior interpretation can be adapted easily to this scenario. In addition to that, the efficient $O(n)$ algorithm for moment updates is much more efficient than the online update for Online EM. In addition to that, it does not suffer from issues encountered by Online Variational Bayes where it breaks the time series into mini-batches.

The gain in perplexity for Moment matching can be explained by the fact that that the approach is performing exact inference with respect to an implicit set of initial distributions. Since this set is determined after seeing the data, the initial distributions are not priors in the Bayesian sense. Hence there is an important open question: is Bayesian moment matching consistent? The empirical results suggest that it may be, so it will be interesting to see if consistency can be proved.

In addition to that, moment matching and ADF are sensitive to ordering of data. This is less of a problem in the case of HMMs since the ordering of the data has a significance, however in models where the observations are exchangeable such as LDA, this may cause problems. As the number of observations grows, $\sum_{\beta_i}$ also grows which reduces the variance of the posterior. As described previously, the moment update equation for ϕ is

$$M_{\phi_{t,e}}(P_n) = \left[M_{\phi_{t,e}}(P_{n-1})\right] x \; e \neq w \tag{8.1}$$

$$M_{\phi_{t,w}}(P_n) = \left[M_{\phi_{t,w}}(P_{n-1}) + y\right] x \tag{8.2}$$

where

$$x = \left[1 - \frac{c_t}{\sum \beta_t + 1}\right] \qquad y = \frac{c_t}{\sum \beta_t + 1} \frac{1}{x M_{\phi_{t,w}}(P_{n-1})} \tag{8.3}$$

Since $\sum_{\beta_t}$ is in the denominator, as this value grows large, the change in the moments becomes very small. Therefore, we need to come up with techniques which allow a smoothing of the variance so that it does not grow too small too fast.

As data sets are becoming larger, algorithms that process these data sets need to be parallelized. With this fact in mind, a very neat extension for the Bayesian Moment Matching algorithm would be to derive a parallel version of the algorithm where the computation can be distributed to various nodes. The main challenge is to combine the estimate of the posterior from different nodes in a way that assures that the approximate posterior moments do not diverge from the true posterior moments.

This thesis has mainly explored the application of moment matching to some directed probabilistic graphical models. However, moment matching can be potentially used to learn posterior distributions in other types of probabilistic and undirected graphical models such as Conditional Random Fields (CRFs). Recently Rashwan et. al [3] have proposed an online an distributed moment matching algorithm for parameter learning in Sum Product networks. This can potentially open many research directions in terms of models where moment matching can be successfully applied.

APPENDICES

Appendix A

Derivation of Expectation Maximization for Activity Recognition

A.1 Maximum Likelihood Supervised Learning

For supervised learning, we label the data by hand by synchronizing the data feed with the video feed. Therefore, the optimal parameters Θ^* and Φ^* can be learned from a single labeled sequence of length T using the following equation

$$\begin{aligned}(\Theta^*, \Phi^*) = argmax_{\Theta,\Phi} \log\left(\Pr\left(y_{1:T}, e_{1:T}^{1:S} | \Theta, \Phi\right)\right) \\ \text{subject to} \\ \sum_i \theta_{y,i} = 1 \, \forall y \in \{1, \ldots, N\} \\ \sum_{e=1}^{M} \phi_{y,e,s} = 1 \, \forall y \in \{1, \ldots, N\} \forall k \in \{1, \ldots, S\}\end{aligned} \tag{A.1}$$

Expanding the log likelihood function and folding the constraints into the objective function using Lagrange multipliers, we get

$$
\begin{aligned}
L(\Theta,\Phi,\lambda) = {} & \sum_{t=1}^{T}\sum_{y}\sum_{y'} \delta\left(Y_t = y', y_{t-1} = y\right)\log\left(\theta_{y',y}\right) \\
& + \sum_{t=1}^{T}\sum_{s}\sum_{y} \delta\left(e_t^s = e, y_t = y\right)\log\left(\phi_{y,e,s}\right) - \sum_{y}\lambda_{1y}\left(\sum_{i}\theta_{y,i} - 1\right) \\
& - \sum_{s}\left(\sum_{y}\lambda_{2ys}\left(\sum_{e=1}^{M}\phi_{y,e,s} - 1\right)\right)
\end{aligned}
\tag{A.2}
$$

Here, $\delta(.) = 1$ if the condition in the parenthesis is true. Since this function is convex, we can differentiate Equation A.2 with respect to each variable and set the value of the derivative to 0 to obtain the optimal values of the parameters.

- **Transition Model**: Differentiating the Lagrangian with respect to $\theta_{y,y'}$ and λ_{1b} we get

$$
\begin{aligned}
\frac{\partial L(\Theta,\Phi,\lambda)}{\partial \theta_{y,y'}} &= \frac{\delta\left(y_t = y', y_{t-1} = y\right)}{\theta_{y,y'}} - \lambda_{1b} = 0 \\
\frac{\partial L(\Theta,\Phi,\lambda)}{\partial \lambda_{1b}} &= \sum_{y'}\theta_{y,y'} - 1 = 0
\end{aligned}
$$

These are $m \times (m+1)$ equations in $m \times (m+1)$ variables. Solving for $\theta_{y,y'}$, we get

$$
\theta_{y,y'} = \frac{\sum_{t=1}^{T}\delta\left(y_t = y' \& y_{t-1} = y\right)}{\sum_{t=1}^{T}\delta\left(y_{t-1} = y\right)} = \frac{n_{y,y'}}{n_y},\ \forall y', y \in \{1,\ldots,N\} \tag{A.3}
$$

Where $n_{y,y'}$ is the number of times y is followed by y' in labeled activity sequence and n_y is the number of times we observe y in the labeled activity sequence. Note that during the experiment, the participants are following a fixed course and thus the activity transitions are biased and do not reflect real life situations. To avoid this pitfall we provide experimental results for another technique for estimating the transition parameters. We use the fact that activities tend to persist. We can learn the parameters of the transition model using the following distribution:

$$
\theta_{y,y'} = \begin{cases} \frac{\tau}{N+\tau-1} & \text{if } y = y' \\ \frac{1}{N+\tau-1} & \textit{otherwise} \end{cases} \tag{A.4}
$$

Therefore, the user is τ times more likely to maintain his behaviour than to change it.

- **Observation Model**: We learn $\phi_{y,e,s}$ by differentiating the Lagrangian by $\phi_{y,e,s}$ and λ_{2ys}

$$\begin{aligned}
\frac{\partial L(\Theta,\Phi,\lambda)}{\partial \phi_{y,e,s}} &= \frac{\delta\left(e_t^s = e, y_t = y\right)}{\phi_{y,e,s}} - \lambda_{2ys} = 0 \\
\frac{\partial L(\Theta,\Phi,\lambda)}{\partial \lambda_{2kb}} &= \sum_{e=1}^{M} \phi_{y,e,s} - 1 \qquad = 0
\end{aligned}$$

Solving for $\phi_{y,e,s}$, we get

$$\phi_{y,e,s} = \frac{\sum_{t=1}^{T} \delta\left(y_t = y \& e_t^s = e\right)}{\sum_{t=1}^{T} \delta\left(y_t = y\right)} = \frac{n_{y,e,s}}{n_y} \tag{A.5}$$

A.2 Maximum Likelihood Unsupervised Learning

For unsupervised Learning, we have used the EM algorithm for learning transition probabilities and maximize the expected value of the log likelihood. We iteratively estimate the new values of the parameters by solving the following optimization problem

$$\Theta^{i+1}, \Phi^{i+1} = \underbrace{argmax_{\pi,\theta,\phi} \overbrace{\left(\sum_{y_{1:T} \in Y_{1:T}} \Pr\left(y_{1:T}, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \log\left(\Pr\left(y_{1:T}, e_{1:T}^{1:S} | \Theta, \Phi\right)\right) \right)}^{Expectation}}_{Maximization} \tag{A.6}$$

subject to

$$\sum_{y'=1}^{N} \theta_{y,y'} = 1 \forall y \in \{1, \ldots, N\}$$

$$\sum_{e=1}^{M} \phi_{y,e,s} = 1 \forall y \in \{1, \ldots, N\} \forall s \in \{1, \ldots, S\}$$

Where $Y_{1:T}$ represents all possible behaviour sequences of length T. The Lagrangian can be written as

$$\begin{aligned}
L\left(\Theta^{i+1}, \Phi^{i+1}, \lambda\right) = & \sum_{t=1}^{T}\sum_{y}\sum_{y'} \Pr\left(Y_t = y', Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \log\left(\theta_{y,y'}\right) \\
& + \sum_{t=1}^{T}\sum_{s}\sum_{y} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \log\left(\phi_{y,e,s}\right) \\
& - \sum_{y} \lambda_{1y}\left(\sum_{y'} \theta_{y,y'} - 1\right) - \sum_{s}\left(\sum_{y} \lambda_{2ys}\left(\sum_{e} \phi_{y,e,s} - 1\right)\right)
\end{aligned} \tag{A.7}$$

The Lagrangian is a convex function and we can differentiate it and set the derivative to 0 to recover the parameters.

- **Transition model**: Differentiating the Lagrangian with respect to $\theta_{y,y'}$ and λ_{1y} and setting it to zero, we get

$$\frac{\partial L\left(\Theta^{i+1}, \Phi^{i+1}, \lambda\right)}{\partial \theta_{y,y'}} = \frac{\Pr\left(Y_t = y', Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)}{\theta_{y,y'}} - \lambda_{1y} = 0 \tag{A.8}$$

$$\frac{\partial L\left(\Theta^{i+1}, \Phi^{i+1}, \lambda\right)}{\partial \lambda_{1y}} = \sum_{y'} \theta_{y,y'} - 1 = 0 \tag{A.9}$$

  Solving for $\theta_{y,y'}$ we get

$$\theta_{y,y'} = \frac{\sum_{t=1}^{T} \Pr\left(Y_t = y', Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)}{\sum_{t=1}^{T} \Pr\left(Y_{t-1} = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)} \tag{A.10}$$

- Sensor model: Differentiating the Lagrangian with respect to $\phi_{y,e,s}$ and λ_{2ys} and setting it to zero, we get

$$\frac{\partial L\left(\Theta^{i+1}, \Phi^{i+1}, \lambda\right)}{\partial \phi_{y,e,s}} = \frac{\sum_{t=1}^{T} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \delta\left(e_t^s = e\right)}{\phi_{y,e,s}} - \lambda_{2kb} = 0 \tag{A.11}$$

$$\frac{\partial L\left(\Theta^{i+1}, \Phi^{i+1}, \lambda\right)}{\partial \lambda_{2ys}} = \sum_{e} \phi_{y,e,s} - 1 = 0 \tag{A.12}$$

Solving for $\phi_{y,e,s}$, we get

$$\phi_{y,e,s} = \frac{\sum_{t=1}^{T} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right) \delta\left(e_t^s = e\right)}{\sum_{t=1}^{T} \Pr\left(Y_t = y, e_{1:T}^{1:S} | \Theta^i, \Phi^i\right)} \tag{A.13}$$

For efficient calculation of these values, we use the terminology defined in section 6.2. Recall that

$$\gamma_t^y = \Pr\left(Y_t = y | e_{1:T}^{1:S}, \Theta, \Phi\right) = \frac{F_t^y B_t^y}{\sum_{y'} F_t^{y'} B_t^{y'}}$$

$$\xi_t^{y,j} = \Pr\left(Y_t = y, Y_{t+1} = j | e_{1:T}^{1:S}, \Theta, \Phi\right) = \frac{F_t^y \theta_{y,j} \prod_s \phi_{j,e_{t+1},s} B_{t+1}^j}{B_t^i}$$

Then we can estimate the parameters using the following equations

$$\theta_{i,j} = \frac{\sum_{t=1}^{T} \xi_t^{i,j}}{\sum_{t=1}^{T} \gamma_t^i}, \quad \phi_{i,j,s} = \frac{\sum_{t=1}^{t} \delta(e_t^s = j) \gamma_t^i}{\sum_{t=1}^{T} \gamma_t^i} \tag{A.14}$$

A.2.1 Avoiding Underflows

To avoid underflow errors, we use the technique explained in [14]. From the definition of F_t^y and B_t^y in section 6.2.1 we can see that at each step we obtain F_t^y from F_{t-1}^i by multiplying it with two probabilities. Since probabilities are less than one, therefore for long enough chains, the probability of F_t^y drops to zero. To avoid this, we define

$$\hat{F}_t^y = \Pr\left(Y_t = y | e_{1:T}^{1:S}, \Theta, \Phi\right) = \frac{F_t^y}{\Pr\left(e_{1:t}^{1:S}\right)} \tag{A.15}$$

We expect this distribution to behave better numerically since its a conditional distribution instead of a joint. We also define

$$k_t = \Pr\left(e_t^{1:|S} | e_{1:t-1}^{1:S}\right) \tag{A.16}$$

Therefore, $\Pr\left(e_{1:t}^{1:|S}\right) = \prod_{m=1}^{t} k_m$.

$$F_t^y = \left(\prod_{m=1}^{t} k_m\right) \hat{F}_t^y$$

Now

$$k_t \hat{F}_t^y = k\,\phi_{y,e_t,s} \sum_{y'} \theta_{y',y} \hat{F}_{t-1}^{y'} \tag{A.17}$$

We can similarly rescale B to obtain

$$B_t^y = \left(\prod_{m=t+1}^{T} k_m \right) \hat{B}_t^y$$

Then

$$k_{t+1} \hat{B}_t^y = k \sum_{y'} \left(\theta_{y,y'} \phi_{y',e_{t+1},s} \hat{B}_{t+1}^{y'} \right) \tag{A.18}$$

Also

$$\begin{aligned} \hat{F}_t^y \hat{B}_t^y &= \frac{F_t^y}{\prod_{m=1}^{t} k_m} \frac{B_t^y}{\prod_{m=t+1}^{T} k_m} \\ &= \gamma_b\,(t) \end{aligned}$$

Using this rescaling, we can avoid underflow errors for long chains.

References

[1] *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

[2] Statistics Canada. Estimates of population, by age group and sex for july 1, Canada, provinces and territories, annual (CANSIM Table 051-0001). Ottawa:. Statistics Canada, 2010.

[3] H. Zhao A. Rashwan and P. Poupart. Online and distributed bayesian moment matching for spns. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, 2016.

[4] H. Abelson, G. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, Cambridge, Massachusetts, 1985.

[5] Roberto C Alamino and Nestor Caticha. Bayesian online algorithms for learning in discrete hidden markov models. *Discrete and continuous dynamical systems: series B*, 9(1):1–10, 2008.

[6] M. Alwan, G. Wasson, P. Sheth, A. Ledoux, and C. Huang. Passive derivation of basic walker-assisted gait characteristics from measured forces and moments. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS San Francisco, CA, USA*, September 2004.

[7] Anima Anandkumar, Dean P Foster, Daniel Hsu, Sham Kakade, and Yi-Kai Liu. A spectral algorithm for latent dirichlet allocation. In *NIPS*, pages 926–934, 2012.

[8] Animashree Anandkumar, Daniel Hsu, and Sham M. Kakade. A method of moments for mixture models and hidden markov models. *Journal of Machine Learning Research - Proceedings Track*, 23:33.1–33.34, 2012.

[9] T. S. Barger, D. E. Brown, and M. Alwan. Health-status monitoring through analysis of behavioral patterns. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 35(1):22–27, 2005.

[10] R. Baumgartner, G. Gottlob, and S. Flesca. Visual information extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Databases*, pages 119–128, Rome, Italy, September 2001. Morgan Kaufmann.

[11] Matthew James Beal. *Variational algorithms for approximate Bayesian inference.* PhD thesis, University of London, 2003.

[12] M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. The infinite hidden markov model. In Z. Ghahramani T. Dietterich, S. Becker, editor, *Advances in Neural Information Processing Systems 14*, pages 577–584. MIT Press, 2002.

[13] J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, International Computer Science Institute Berkeley CA and Computer Science Division Department of Electrical Engineering and Computer Science U.C. Berkeley, 1998.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[15] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning*, 3:993–1022, 2003.

[16] Byron Boots and Geoffrey J. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In Burgard and Roth [20].

[17] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April–June 1985.

[18] Jack S. Breese and Daphne Koller, editors. *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001.* Morgan Kaufmann, 2001.

[19] H. H. Bui, D. Q. Phung, and S. Venkatesh. Hierarchical hidden markov models with general state hierarchy. In *Proceedings of AAAI*, 2004.

[20] Wolfram Burgard and Dan Roth, editors. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011w*. AAAI Press, 2011.

[21] Olivier Cappé. Online em algorithm for hidden markov models. *Journal of Computational and Graphical Statistics*, 20(3), 2011.

[22] Olivier Cappé. Online em algorithm for hidden markov models. *Journal of Computational and Graphical Statistics*, 20(3), 2011.

[23] C. Carvalho, M. Johannes, H. F. Lopes, and N. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2011.

[24] O. Chuy, Y. Hirata, Z. D. Wang, and K. Kosuge. A control approach based on passive behavior to enhance user interaction. *IEEE Transactions on Robotics*, 23(5):899–908, 2007.

[25] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[26] Bui H Venkatesh S. Duong T, Phung D. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 2009.

[27] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[28] Nicholas Foti, Jason Xu, Dillon Laird, and Emily Fox. Stochastic variational inference for hidden markov models. In *Advances in Neural Information Processing Systems*, pages 3599–3607, 2014.

[29] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319. ACM, 2008.

[30] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, 2 edition, 2004.

[31] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks.

[32] Georg Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[33] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, May 2002.

[34] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[35] J.M. Hammersley and D.C. Handscomb. *Monte Carlo methods*. Methuen's monographs on applied probability and statistics. Methuen, 1975.

[36] Waddell D. Oliver A. Smith D. Fleming R. Wolf S. Hass C., Gregor R. The influence of tai chi training on the center of pressure trajectory during gait initiation in older adults. *Archives of Physical Medicine and Rehabilitation*, 85(10):1593–1598, 2004.

[37] K. A. Heller, Y. W. Teh, and D. Görür. Infinite hierarchical hidden Markov models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 12, 2009.

[38] Ralf Herbrich. Minimising the kullbackleibler divergence. Technical report, Microsoft Research, 2005.

[39] Y. Hirata, A. Muraki, and K. Kasuge. Motion control of intelligent passive-type walker for fall prevention function based on estimation of user state. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006.

[40] M. Hoffman, D. Blei, , and F. Bach. Online learning for latent dirichlet allocation. In *Neural Information Processing Systems*, 2010.

[41] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[42] Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. In *COLT - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009* [1].

[43] Richard Zhi-Ling Hu, Adam Hartfiel, James Tung, Adel Fakih, Jesse Hoey, and Pascal Poupart. 3d pose tracking of walker users' lower limb with a structured-light camera on a moving platform. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 29–36. IEEE, 2011.

[44] Stam H.J. Janssen W.G., Bussmann H.B. Determinants of the sit-to-stand movement: a review. *Physical Therapy*, 82:866–879, 2002.

[45] N. Kantas, A. Doucet, S.S. Singh, and J. M. Maciejowski. Overview of sequential monte carlo methods for parameter estimation on general state space models. In *In Proc. 15th IFAC Symposium on System Identification (SYSID)*, 2009.

[46] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML01*, 2001.

[47] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 198–202, Austin, Texas, August 1984. American Association for Artificial Intelligence.

[48] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155–212, July 1984.

[49] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *International Journal of Robotics Research*, 26(1):119–134, 2007.

[50] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171:311–331, April 2007.

[51] M. Lichman. UCI machine learning repository, 2013.

[52] J.S. Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *J. of the Am. Statistical Association*, 89(427):958 – 966, 1994.

[53] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[54] David Mimno, Matt Hoffman, and David Blei. Sparse stochastic inference for latent dirichlet allocation. *ICML*, 2012.

[55] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[56] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In Breese and Koller [18], pages 362–369.

[57] Deneve S. Mongillo G. Online learning with hidden markov models. *Neural Com*, 20(7):1706–1716, July 2008.

[58] K. Murphy and M. Paskin. Linear time inference in hierarchical hmms. 2001.

[59] B. Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

[60] S. Ng, A. Fakih, A. Fourney, P. Poupart, and J. Zelek. Towards a Mobility Diagnostic Tool: Tracking Rollator Users Leg Pose With a Monocular Vision System. In *International Conference of IEEE Engineering in Medicine and Biology Society (EMBC)*, 2009.

[61] F. Omar, M. Sinn, J. Truszkowski, P. Poupart, J. Tung, and A. Caine. Comparative analysis of probabilistic models for activity recognition with an instrumented walker. In *Proceedings of the Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 392–400, Corvallis, Oregon, 2010. AUAI Press.

[62] M. Patel, R. Khushaba, J. Miro, and G. Dissanayak. Probabilistic models versus discriminate classiers for human activity recognition with an instrumented mobility-assistance aid. In *Australasian Conference on Robotics and Automation*, 2010.

[63] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* The Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann Publishers, 1988.

[64] MS Press. PrimeSense supplies 3-D-Sensing technology to Project Natal for Xbox 360. Press Release, March 2010.

[65] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77 of *257-268*, Feb 1989.

[66] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 1 edition, August 1999.

[67] S. J. Russell and P. Norvig. *Artificial intelligence: A modern approach*. Prentice Hall, 2003.

[68] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-Time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*, June 2011.

[69] Ajit Singh. Em algorithm.

[70] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[71] Geir Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, 50(2):281–289, 2002.

[72] Charles Sutton and Andrew McCallum. *Introduction to Statistical Relational Learning*, chapter An Introduction to Conditional Random Fields for Relational Learning. MIT Press, 2006.

[73] J. Tung, K. Zabjek, W. Gage, B. Maki, and W. McIlroy. Frontal plane balance control with rollators: Perturbed stance and walking. *Archives of Physical Medicine and Rehabilitation, Volume 89, Issue 10, Pages e50-e50*, 89:e50, 2008.

[74] Md. Zia Uddin and Tae-Seong Kim. Human activity recognition based on independent depth silhouette components and optical flow features. *The Imaging Science Journal*, 59(5), 2011.

[75] L. Wesserman. *All of statistics : a concise course in statistical inference.* Springer, 2004.

[76] Jia Zeng, William K Cheung, and Jiming Liu. Learning topic models by belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1121–1134, 2013.