Unsupervised Bayesian Learning of HMM Transition Parameters by Moment Matching

Farheen Omar and Pascal Poupart
Davir R. Cheriton Schol of Computer Science
University of Waterloo
200 University Avenue West, Waterloo, N2L 3G1, ON, Canada
{f2omar,ppoupart}@uwaterloo.ca

Technical Report CS-2013-19

Abstract

Hidden Markov Models (HMMs) provide a simple framework to model the generative process of some types of sequential of data. We consider the challenging problem of estimating the parameters of an HMM incrementally by doing a constant amount of computation per observation. We propose an online moment matching algorithm for Bayesian learning of transition parameters of an HMM. While moment matching suggests that learning is done approximately, we show that the algorithm performs exact Bayesian learning for an implicit prior. The algorithm exploits the fact that only the first, second and third order moments of the prior need to be specified before any data is observed. After each observation, moment matching implicitly specifies additional moments in the prior. Hence, the algorithm specifies the moments of the prior incrementally as they become needed in the computation. Therefore the overall computation is exact with respect to the resulting prior. The algorithm performs a constant amount of computation per observation in contrast to other methods such as naive exact Bayesian learning where the computation grows exponentially and Gibbs Sampling where an approximate solution is obtained after multiple iterations. We demonstrate the performance of the algorithm by comparing it to exact Bayesian Learning and Gibbs Sampling on synthetic data as well as real data for HMMs that arise in activity recognition.

1 Introduction

Over the years, hidden Markov models (HMMs) have emerged as a popular framework to model and reason about sequential data in many applications including activity recognition, speech recognition and natural language processing. Nevertheless, learning the parameters of an HMM remains challenging, especially in the absence of labeled data and in online settings. Optimization techniques based on maximum likelihood face a non-convex optimization problem and online learning restricts algorithms to a constant amount of computation per observation.

In this work, we propose an online Bayesian learning technique that performs moment matching. To our knowledge, this algorithm is the first exact online Bayesian learning technique that can process each observation in a constant amount of time for HMMs. Our approach exploits the fact that only the first, second and third order moments of the prior need to be specified before any data is observed. After each observation, moment matching implicitly specifies additional moments in the prior. Hence, the algorithm specifies incrementally the moments of the prior as they become needed in the computation and therefore the overall computation is exact with respect to the resulting prior.

Delaying the implicit specification of the moments of the prior until they become needed, allows us to set the moments in a way that ensures that computation remains bounded at every step. In contrast, a naive implementation of exact Bayesian learning would require memory and computation that grows exponentially in the number of states with respect to the amount of data observed so far to process each new observation.

The paper is structured as follows. Sec. 2 introduces HMMs and reviews exact Bayesian learning. Sec. 4 describes our moment matching technique and proves that under certain conditions the computation is exact with respect to an implicit prior. Sec. 5 demonstrates our moment matching techniques with synthetic and real data from HMMs that arise in activity recognition. Sec. 6 concludes and discusses future work.

2 Background

2.1 Hidden Markov Model

Consider a hidden Markov model (HMM) defined by a transition function $\Pr(Y_t|Y_{t-1})$ and an observation function $\Pr(E_t|Y_t)$ where Y_t and E_t denote the random variables for the hidden state and the observation (evidence) respectively at time step t. In this work, we assume that the state space is finite, while the observation space may be finite or continuous. We also assume that the observation distribution is completely known.

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 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 performed over a period of time that may range from hours to weeks. Also, the sequences of activities should not be scripted, meaning that users should be free to perform activities as they wish in order to generate natural sequences. 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 transition function as the activities are performed, meaning that learning should be done in an online fashion. Since activities tend to persist for a while before changing, the resulting HMMs are often known as sticky HMMs [5] [4]. In subsequent sections we will describe the algorithm for learning the parameters of a generic transition function with respect to an implicit prior. We

will also explain how we can adapt the algorithm to the sticky HMM case.

2.2 Online Unsupervised Learning

Online unsupervised learning for HMMs is quite challenging even for simple transition functions. Maximum likelihood leads to a non-convex optimization problem and techniques such as Expectation Maximization (EM) may get trapped into local optima. Bayesian learning sidesteps local optima issues by doing inference instead of optimization. However, the posterior distribution may be difficult to represent tractably.

Let Θ be the parameters of the transition function and n be the total number of states where $\Theta = \{\theta_1, \ldots, \theta_n\}, \theta_y = \{\theta_{y,1}, \theta_{y,2}, \ldots, \theta_{y,n}\}$ and $\theta_{y,y'} = \Pr(Y_t = y'|Y_{t-1} = y)$. Starting off with a prior distribution over $P_0(\Theta)$, we can use Baye's theorem to recursively compute a posterior $P_{t+1}^{y'}(\Theta) = \Pr(\Theta|Y_{t+1} = y', e_{1:t+1})$ at time step $t+1$ based on the posterior at time step t using:

$$P_{t+1}^{y'}(\Theta) = k_{t+1}^{y'} \sum_y P_t^y(\Theta) \Pr(Y_{t+1} = y'|Y_t = y, \Theta) c_t^y$$

Here, $k_{t+1}^{y'}$ is a normalization constant equal to $1/\Pr(Y_{t+1} = y'|e_{1:t})$ and $c_t^y = \Pr(Y_t = y|e_{1:t})$. They can be calculated using:

$$k_{t+1}^{y'} = \sum_y c_t^y \int_\Theta P_t^y(\Theta) \theta_{y,y'} d\Theta \tag{1}$$

$$c_t^y = \frac{\Pr(e_t|Y_t = y) k_t^y}{\sum_y \Pr(e_t|Y_t = y) k_t^y} \tag{2}$$

To allow Bayesian learning from multiple sequences, we use the following equation to calculate the posterior if $t+1$ is the start of a sequence

$$P_{t+1}^{y'}(\Theta) = k_{t+1}^{y'} \sum_y \lambda(y) \theta_{y,y'} P_t(\Theta) \tag{3}$$

where $P_t(\Theta) = \Pr(\Theta|e_{1:t}) = \sum_y c_t^y P_t^y(\Theta)$ and $\lambda(y)$ is a pre-specified distribution over y. We now describe how to do exact Bayesian learning for three different parameterizations of the transition function.

2.2.1 Sticky HMM Parametrization

In sticky HMMs that arise in activity recognition problems, a person executes an activity for a while until a change occurs. The probability that a state y persists or "sticks" is given by Bernoulli parameter θ_y. In this model, $\theta_{y,y'} = \theta_y$ if $y = y'$ and $\theta_{y,y'} = (1-\theta_y)/(n-1)$ otherwise. It is natural to represent the prior $P_0(\Theta)$ by a product of Beta distributions $P_0(\Theta) = \prod_{i=1}^n Beta(\theta_y, \alpha_{y,1}, \alpha_{y,2})$. The Beta distribution is a conjugate prior for a Bernoulli likelihood. $Beta(\theta_i; \alpha_i) = k\theta_i^{\alpha_{i,1}-1}(1-\theta_i)^{\alpha_{i,2}-1}$ where k is the normalization constant and $\alpha_{i,j}$ are the hyperparameters. The posterior at time $t+1$ can be computed as:

$$P_{t+1}^{y'}(\Theta) = k_{t+1}^{y'} \sum_{y} c_t^y P_t^y(\Theta) \Big[\delta(y, y')\theta_y + (1 - \delta(y, y'))\,(1 - \theta_y)/(n-1) \Big] \quad (4)$$

Here $\delta(y, y')$ is a Kronecker delta that returns 1 when $y = y'$ and 0 otherwise.

We can further restrict the transition function to have a single Bernoulli parameter denoting the probability of a state to persist. In that case, we replace θ_y by θ in Eq. **??** to get the posterior for this model. Although this model with a single Bernoulli parameter is restrictive, we will use it mostly to simplify the exposition of some concepts. Our algorithms do not rely on this restriction since they can deal with arbitrary transition distributions.

2.2.2 n-1 Multinomial Parameters per State

We model each θ_y as a multinomial random variable. It is natural to start with a prior distribution $P_0(\Theta)$ represented as a product of Dirichlet distributions since Dirichlets are conjugate priors for Multinomial likelihoods and are a generalization of Beta distributions. $P_0(\Theta) = \prod_{i=1}^{n} Dir(\theta_i; \alpha_i)$ and $Dir(\theta_i; \alpha_i) = k \prod_j \theta_{i,j}^{\alpha_{i,j}-1}$ where k is the normalization constant and $\alpha_{i,j}$ are the hyperparameters. The posterior at $t+1$ can be computed using:

$$P_{t+1}^{y'}(\Theta) = k_{t+1}^{y'} \sum_{y} P_t^y(\Theta)\theta_{y,y'} c_t^y \quad (5)$$

The equations described above show that depending on the parametrization of the transition function, when P_t^y is a mixture of products of Dirichlets / products of Betas / Betas, then $P_{t+1}^{y'}$ is also a mixture of products of Dirichlets / products of Betas / Betas because each P_t^y is multiplied by a $\theta_{y,y'}$ at every time step. Even if we start with a single component in the prior, the posterior becomes a mixture because of the summation over y. The number of components in the mixture may increase by a factor of n at each time step. However, the degree of the polynomial in $\theta_{i,j}$ corresponding to the mixture of components increases by 1. Hence, the posterior can be represented by a polynomial with a number of terms that grows exponentially over n with the amount of data. This allows us to do exact Bayesian learning, but not in an online fashion since the amount of computation and memory to process each observation increases with the amount of data. In Sec. 4, we propose a moment matching algorithm that ensures exact computation with respect to an implicit prior while keeping the amount of computation constant at each time step.

3 Related Work

Spectral learning algorithms also use moment matching to estimate the parameters of an HMM [1]. In particular, Hsu [6] showed that an observable operator model parametrization of an HMM can be learnt from a large number of short sequences

of observations. In contrast, we learn from a single sequence of observations. Some spectral algorithms have been adapted to the single sequence case [2], but they yield approximate point estimates of the parameters for finite sequences, where as our approach is exact and provides distributional information about the parameters. Our approach is also related to expectation propagation (EP) [7] in the sense that we match the first few moments of posteriors at each time step. However, EP is an iterative method that yields an approximation, whereas our approach is not iterative and it is exact for an implicit prior. On the other hand, our approach assumes that the transition function is known whereas spectral learning techniques and EP make no such assumption. Sequential Monte Carlo (SMC) methods can also been used for Bayesian Learning. In SMC, we approximate the parameters by sampling sequentially from a series of probability distributions of increasing dimensions. Doucet et. al. [3] have summarized many convergence results for SMC methods that rely on the ability to get an infinite number of samples at every time step. For any finite number of samples N, it can be proven that the number of particles effectively used to sample from the distribution at time t will effectively be reduced to 1. This is called the degeneracy problem and many SMC methods suffer from it. Degeneracy of the sample set does not allow the algorithm to be consistent. Markov chain Monte Carlo techniques such as Gibbs sampling are also used extensively to do approximate Bayesian Learning of parameters. Since MCMC techniques are iterative, they may require a large number of iterations to converge. In addition, they do not lend itself easily to online learning since after each iteration it may be necessary to resample all previous hidden states to ensure convergence. In contrast, our algorithm learns from the data as it becomes available and does a constant amount of work at each iteration.

4 Moment Matching

We will now describe a framework for calculating moments of the posterior at every time step and then show how we can use them to derive an approximate Bayesian learning algorithm.

4.1 Moments of a Distribution

We define the o^{th} order moment of a multivariate distribution in $X = (X_1, \ldots, X_n)$ to be the expectation of a monomial in the X_i's of degree o. For example, $X_1 X_3^2$ is a monomial of degree 3, and $\int_X X_1 X_3^2 \Pr(X) dX$ is a third order moment of $\Pr(X)$ that corresponds to the expectation of this monomial. We will denote by $M_{i_1,i_2,\ldots,i_o}(P)$ the o^{th} order moment of P where the expectation is with respect to the monomial $\prod_{j=1}^{o} X_{i_j}$. For example, $M_{1,2,1}(P) = \int_X X_1 X_2 X_1 \Pr(X) dX$. We will also denote by $M_{I_o}(P)$ an o^{th} order moment where $I_o = (i_1, i_2, ..., i_o)$ is a string of o indices. The set of o^{th} order moments of a distribution P is denoted by $\mathbb{M}_o(P)$ and its cardinality by $|\mathbb{M}_o|$. It is easy to see that the first order moments of a distribution $M_i(P)$ are equivalent to its means. Using Eq. 1, the o^{th} moment of the posterior at time $t+1$ can be calculated as

$$M_{I_o}(P_{t+1}^{y'}) = k_{t+1}^{y'} \sum_y c_t^y \int_\Theta \theta_{i_1} ... \theta_{i_o} P_t^y(\Theta) \theta_{y,y'} d\Theta \tag{6}$$

If a new sequence starts at time $t+1$ the posterior can be calculated using

$$M_{I_o}\left(P_{t+1}^{y'}\right) = \qquad k_{t+1}^{y'} \sum_y \lambda(y) M_{I_o,(y,y')}\left(P_t\right) \tag{7}$$

where $I_o, (y, y')$ denotes a string of $o+1$ indices such that the last index (y, y') refers to parameter $\theta_(y, y')$. It is clear from these equations that the o^{th} order moments at time $t+1$ can be calculated using the $o+1^{th}$ order moments at time t since the length of $I_o, (y, y')$ is $o+1$. This will become a fundamental building block when we prove that our moment matching algorithm is exact with respect to an implicit prior. Also $k_{t+1}^{y'}$ denotes the following normalization constant.

$$k_{t+1}^{y'} = \sum_y c_t^y M_{(y,y')}\left(P_t^y\right) \tag{8}$$

When we consider a transition function parameterized by a single Bernoulli variable θ, we will use a different notation for moments. Since there is a single parameter θ, the string of indices $i_1, ..., i_o$ is cumbersome and all indices are necessarily the same. Instead of using a string of indices $i_1, ..., i_o$ to specify a moment $M_{i_1,...,i_o}$, we will simply indicate the order o of the moment M_o since it is clear that the monomial used in the expectation is θ^o. In this case the o^{th} order moment of the posterior can be calculated as follows:

$$M_o\left(P_{t+1}^{y'}\right) = k_{t+1}^{y'} \sum_y c_t^y \left[\delta(y, y') M_{o+1}\left(P_t^y\right) + \left(1 - \delta(y, y')\right) \frac{1 - M_{o+1}\left(P_t^y\right)}{n-1}\right] \tag{9}$$

4.2 Parameter Learning by Moment Matching

As we have mentioned before, the number of mixture components in the posterior grows with time. In order to avoid this, we propose an algorithm that computes the posteriors $P_t^y(\Theta)$ exactly at each time step and to project them onto a single product of Dirichlets before computing the next posteriors. This projection ensures that the mixture does not get arbitrarily large.

There are several ways of approximating a mixture of products of Dirichlets by a single product term. While one could approximate the mixture by sampling from it or by retaining the mixture component with the largest mixture probability, these approximations may be severe. Since the hyperparameters of a Dirichlet distribution can be determined by its first order moments and a subset of second order moments, we can approximate the mixture by computing its first and second order moments and then replacing the mixture by a single product of Dirichlets that has the same first and second order moments. Let $M_i(DirProd)$ be the first order moments of a product of

Dirichlets where i is the index of $\theta_{y,y'}$. Let $M_{j,j}(DirProd)$ be a second order moment of a product of Dirichlets where j is the index for $\theta_{y,1}$:

$$\begin{aligned} M_i(DirProd) &= \int_\Theta \theta_{y,y'} \prod_{l=1}^{n} Dir(\theta_l; \alpha_l) d\Theta \\ &= \alpha_{y,y'}/\alpha_{y,0} \end{aligned} \tag{10}$$

$$\begin{aligned} M_{j,j}(DirProd) &= \int_\Theta (\theta_{y,1})^2 \prod_{l=1}^{n} Dir(\theta_l; \alpha_l) d\Theta \\ &= \frac{\alpha_{y,1}}{\alpha_{y,0}} \frac{\alpha_{y,1}+1}{\alpha_{y,0}+1} \end{aligned} \tag{11}$$

Here $\alpha_{y,0} = \sum_{y'=1}^{n} \alpha_{y,y'}$. We will use this notation in the rest of the paper. Using Eqs 10 and 11 we can set up a system of equations by matching the moments of a product of Dirichlets to those calculated by using Eq. 6. For each Dirichlet $Dir(\theta_y)$ in the product of Eq. 10 we get n equations in n variables where the variables are $\alpha_{y,y'}$. However, since $\sum_{y'=1}^{n} \theta_{y,y'} = 1$, the last equation becomes redundant. Eq. 11 yields the last equation of the system of equations. We will call the set of moments that allow us to do the update at the next time step the sufficient set of moments. The choice of j in Eq. 11 is arbitrary. In fact, we can choose any second order moment to solve the system of equations. Choosing j in this way allows us to determine the hyperparameters as follows:

$$\alpha_{y,y'} = M_i(M_i - M_{j,j})/((M_{j,j} - (M_i)^2)) \tag{12}$$

Using these equations, the process of computing posteriors where we fit a single product of Dirichlets to a mixture of such products is equivalent to computing moments only at each iteration. This approach takes a constant amount of computation at each iteration because the number of mixture components remains bounded, however inference is approximate. The algorithm is described in Alg. 1

Algorithm 1 approximateMomentMatching

1: **for** $t = 1$ to T **do**
2: For each y', for each i, compute $M_i(P_{t+1}^{y'})$ according to Eq. 6 where i indexes $\theta_{y,y'}$
3: For each y', for each j compute $M_{j,j}(P_{t+1}^{y'})$ according to Eq. 6 where j indexes $\theta_{y,1}$
4: For each y', compute $c_{t+1}^{y'}$ according to Eq. 2
5: For each y', for each i, j, compute $M_{i,j}(P_{t+1}^{y'})$ from $M_i(P_{t+1}^{y'})$ according to Eq. 13
6: For each y', compute $M_{i,j,k}(P_{t+1}^{y'})$ from $M_{i,j}(P_{t+1}^{y'})$ according to Eq. 13
7: **end for**

Lines 2 and 3 allow us to calculate moments of the posterior at time $t+1$. Lines 5 and 6 allow us to match the moments to a single product of Dirichlets. Other second

order and third order moments of P^y_{t+1} can be calculated recursively using the values of $\alpha_{i,j}$ using

$$M_{I_o,(y,y')}(DirProd) = M_{I_o}(DirProd)\frac{\alpha_{y,y'} + n_{y,y'}}{\alpha_{y,0} + n_{(y,0)}} \tag{13}$$

where i_o indexes $\theta_{y,y'}$ and $n_{y,y'}$ is the number of times index i_o is repeated in string I.

Although Eq. 13 allows us to calculate an infinite number of moments for the posterior at time $t+1$, only the first, second and some of the third order moments are necessary to compute the moments necessary for moment matching at the next time step. We can call this the sufficient set of moments. This last observation suggests something really interesting about the posterior at each time step. As long as we use any distribution Q^y_t with the same sufficient moments as P^y_t, then the first and second order moments of $P^{y'}_{t+1}$ and $Q^{y'}_{t+1}$ will be the same.

4.2.1 Sticky HMM

We can restrict the moment matching scheme described above to approximate the posterior in a sticky HMM. Here we approximate a mixture of products of Betas by a single product term using moment matching as follows:

$$M_y(BetaProd) = \alpha_{y,1}/\alpha_{y,0} \tag{14}$$

$$M_{y,y}(BetaProd) = \frac{\alpha_{y,1}}{\alpha_{y,0}}\frac{\alpha_{y,1}+1}{\alpha_{y,0}+1} \tag{15}$$

The hyperparameters can be calculated using:

$$\begin{aligned}
\alpha_{y,1} &= M_y\left(M_y - M_{y,y}\right)/((M_{y,y} - (M_y)^2)) \\
\alpha_{y,2} &= (1 - M_y)\left(M_y - M_{y,y}\right)/((M_{y,y} - (M_y)^2)) \\
M_{I_o,y}(BetaProd) &= M_{I_o}(BetaProd)\frac{\alpha_{y,1} + n_{y,1}}{\alpha_{y,0} + n_{(y,0)}}
\end{aligned}$$

where $n_{y,j}$ is the number of times index i_o is repeated in I.

For a single Bernoulli parameter model, the equations become

$$M_1(Beta) = \alpha_1/\alpha_0 \tag{16}$$

$$M_2(Beta) = \frac{\alpha_1}{\alpha_0}\frac{\alpha_1+1}{\alpha_0+1} \tag{17}$$

$\alpha_0 = \alpha_1 + \alpha_2$. The hyperparameters can be estimated using:

$$\begin{aligned}
\alpha_1 &= M_1\left(M_1 - M_2\right)/((M_2 - (M_1)^2)) \\
\alpha_2 &= (1 - M_1)\left(M_1 - M_2\right)/((M_2 - (M_1)^2)) \\
M_{o+1}(Beta) &= M_0(Beta)\frac{\alpha_1 + n_1}{\alpha_0 + n_{(0)}}
\end{aligned}$$

For the single Bernoulli parameter model, the sufficient set of moments at every time step is the first three moments since we can calculate the first two moments at time step $t+1$ using the first three moments at time step t.

4.3 Exact Inference

Does there exist a prior such that if we did exact computation using this prior, it would lead to the same sequence of first order moments at every time step t? To investigate this question, let us use Eq. 9 to calculate moments for a model with a single parameter and two states at time step 1.

$$M_3(P_1^1)/k_1^1 = c_0^1 M_4(P_0^1) + c_0^2 \left(M_3(P_0^2) - M_4(P_0^2)\right)$$
$$M_3(P_1^2)/k_1^2 = c_0^1 \left(M_3(P_0^1) - M_4(P_0^1)\right) + c_0^2 M_4(P_0^2)$$

We did not specify M_4 at time 0, therefore, we can use this system of equations to calculate it. As $M_3(P_0)$ is already selected, then selecting $M_4(P_0)$ implicitly determines $M_3(P_1^y)$. This suggests that approximating the posterior P_1^y at time step 1 with a single Beta distribution using Alg 1 implicitly determines the 4^{th} moment of the prior P_0. Similarly, selecting the third moment of P_2^y implicitly determines the 5^{th} moment of P_0. However, there are two posteriors P_t^y at each step t, one for each hidden state y, so the third moment of each posterior may yield different values for $M_{t+3}(P_0)$.

We can resolve this conflict in two ways. Instead of starting with a single prior $P_0(\theta)$ that is independent of y, we could assume that we start with two priors $P_0^y(\theta)$, one for each hidden state y. This is weird, but technically fine. Since we have two posteriors at each time step $t > 0$ that depend on the hidden state y, one could argue that the priors at time step 0 are really the posteriors of a previous experiment and therefore naturally yield two priors. Then, Eq. 9 can be used to define a system of linear equations that relate the o^{th} and $o+1^{th}$ moments of the priors P_0^y to the o^{th} moments of the posteriors P_1^y and hence we get a system of linear equations in 2 unknowns and 2 equations. More generally, we can solve a similar system of linear equations to determine the $t+2^{th}$ moment of the priors based on the third moments of the posteriors up to time step t. Note that we are not suggesting to solve all these linear systems of equations, but we are simply pointing out that a solution to those linear systems implicitly determines the priors. However, the solution to these systems of linear equations may not yield positive moments. This means that there may exists two functions for which exact inference yields the same results, but these functions may not be valid priors as they may be negative in some parts of the domain.

In order to address this problem we use the following observation. At any time $t+1$ summing the unnormalized posterior for all states give us

$$\sum_{y'} k_{t+1}^{y'} \left(P_{t+1}^{y'}(\Theta)\right) = \sum_{y'} c_t^{y'} \left(P_t^{y'}(\Theta)\right) \tag{18}$$

We can see that the o^{th} moment of the L.H.S of the equation is equal to the o^{th} moment on the R.H.S. Using this observation we propose a modification to our algorithm. We

calculate first and second order moments for each posterior $P_{t+1}^{y'}$ from $t = 1$ to $T - 1$ as before. Let $S_{t+1} = \sum_{y'} c_t^{y'} \left(P_t^{y'}\right)$. Then we can calculate moments for the n^{th} posterior using

$$k_{t+1}^n P_{t+1}^n(\Theta) = S_{t+1}(\Theta) - \sum_{y'=1}^{n-1} k_{t+1}^{y'} P_{t+1}^{y'}(\Theta) \tag{19}$$

$$k_{t+1}^n M_I\left(P_{t+1}^n\right) = M_I\left(S_{t+1}\right) - \sum_{y'=1}^{n-1} k_{t+1}^{y'} M_I\left(P_{t+1}^{y'}\right)$$

This alternative procedure is described in Alg. 2.

Algorithm 2 exactMomentMatching

1: **for** $t = 1$ to T **do**
2: For each y', and each i compute $M_i(P_{t+1}^{y'})$ according to Eq. 6 where i indexes $\theta_{y,y'}$
3: For each y', for each j compute $M_{j,j}(P_{t+1}^{y'})$ according to Eq. 6 where j indexes $\theta_{y,1}$
4: For each y', compute $c_{t+1}^{y'}$ according to Eq. 2
5: For each $y' \in 1, \ldots, n-1$, for each i, j, compute $M_{i,j}(P_{t+1}^{y'})$ from $M_i(P_{t+1}^{y'})$ according to Eq. 13
6: For $y' = n$, for each i, j, compute $M_{i,j}(P_{t+1}^{y'})$ from $M_i(P_{t+1}^{y'})$ according to Eq. 19
7: For each $y' \in 1, \ldots, n-1$, compute $M_{i,j,k}(P_{t+1}^{y'})$ from $M_{i,j}(P_{t+1}^{y'})$ according to Eq. 13
8: For $y' = n$, compute $M_{i,j,k}(P_{t+1}^{y'})$ from $M_{i,j}(P_{t+1}^{y'})$ according to Eq. 19
9: **end for**

4.3.1 Constructing the Prior

Now we will describe a method to construct a prior Q_0 at time step 0 such that if we started from Q_0 instead of P_0, then the sufficient set of moments (first order moments and some of the second order moments) of the posterior Q_{t+1} computed at time step $t+1$ using exact inference will yield the same moments that we acquire by computing P_{t+1} using moment matching. Note that we only compute enough moments of the posterior at every time t that allow us to compute the moments of the posterior at time step $t+1$. Effectively, all the remaining moments of the posterior at t are "unspecified". When we do a Bayes update and then do moment matching, we can set up a system of equations that allows us to "set" these unspecified moments. The following system of equations will allow us to calculate the $o + 1^{th}$ order moments of the posterior at time step t using the o^{th} order moments of the posterior at time step $t + 1$:

$$M_{I_o}\left(P_{t+1}^{y'}\right) = k_{t+1}^{y'} \sum_{y} c_t^y M_{I_o,(y,y')}\left(P_t^y\right) \ \forall I_o$$

$$\text{where } y' \in 1, \ldots, n-1$$

$$\sum_{y'} k_{t+1}^{y'} M_{I_{o+1}}\left(P_t^{y'}\right) = \sum_{y'} c_t^{y'} M_{I_{o+1}}\left(P_{t-1}^{y'}\right) \ \forall I_{o+1}$$

In this system of equations $M_{I_{o+1}}(P_t^{y'})$ and $M_{I_o,(y,y')}(P_t^y)$ are the variables. The rest of the terms are known. The last equation ensures that the system of equations is consistent at every previous time step. Note that the number of o^{th} order moments $|M_o|$ is smaller than the number of $o+1^{th}$ order moments $|M_{o+1}|$. These equations give us a system in $(n-1)|M_o| + |M_{o+1}|$ equations in $n \times |M_{o+1}|$ variables. It is easy to see that this system is underdetermined and may have infinitely many solutions. Note that $M_{I_o,(y,y')}(P_t^y)$ are the only variables that show up in the first set of equations. In fact, $M_{I_o,(y,y')}(P_t^{y''})$ are free for all $y'' \neq y$. Therefore, we can use the free variables to set the constrained variables such that both equations are satisfied. This means that we can solve this system of equation at every time step. Solving $t-2$ such systems will give us the t^{th} order moments at time step 3.

At time step 2 we only have one set of equations

$$M_{I_o}\left(P_2^{y'}\right) = k_1^{y'} \sum_{y} c_t^y M_{I_o,(y,y')}\left(P_1^y\right)$$

The moments at time 0 can be calculated using

$$\sum_{y}^{'} k_{t+1}^{y'} M_{I_{o+1}}\left(P_1^{y'}\right) = M_{I_{o+1}}\left(P_0\right)$$

Theorem 1. *The moment matching procedure described in Alg. 2 performs exact inference with respect to an implicit prior.*

Proof. In the previous section, we have described an algorithm that allows us to set the moments of the prior recursively. Each time we select second order moments at time $t+1$, we set 3rd order moments at time t, which allows us to set 4^{th} order moments at time $t-1$ and so on until we can set $t+1^{th}$ order moments of the implicit prior at time step 0. This way we can construct a function Q_0 whose moments are equal to the implicit prior at time 0. Solving the t systems of equations is in fact the exact reverse of doing exact inference. Therefore, if we start with Q_0 and do exact inference, the first order and some second order moments of the posterior Q_{t+1} will be the same as those of P_{t+1} computed by moment matching.

This means that there is a function with suitable moments such that exact inference yields the same results as Alg. 2. We can show that this function is a valid prior since it is necessarily positive and it integrates to 1. The implicit prior necessarily integrates to 1 since we can select $M_0(P_0)$ to be 1 when we specify the first few moments of

the prior. Setting $M_0(P_0)$ to 1 is compatible with the systems of linear equations that determine the remaining moments of P_0.

We prove by contradiction that the implicit prior must be positive. Suppose that the function corresponding to the moments of P_0 obtained by solving the linear systems of equations is negative in some part of the domain. As argued above, we can obtain P_t^1 from P_0 by exact inference according to Bayes' theorem:

$$P_t^1(\theta) \propto P_0(\theta) \Pr(Y_t = 1, e_{1:t}|\theta)$$

Since P_0 is assumed to be negative in some part of the domain and the likelihood $\Pr(Y_t = 1, e_{1:t}|\theta)$ is necessarily positive, the posterior P_t^1 must be negative in some part of the domain. This leads to a contradiction since we can always set P_t^1 to be a product of Dirichlet distributions, which is necessarily positive. □

One may object that a prior must be fully specified before any data is observed otherwise it is not a prior. Although Alg. 2 implicitly determines the moments of the prior in an incremental fashion as the data is observed. All moments are determined before they are used in any computation. So even though some moments are defined after we have observed some data, these moments do not impact the likelihood of the data observed so far. The right way to think about this process is that we partially specify a prior and then lazily wait until a moment is needed in the inference to specify it. Waiting till a moment is needed allows us to set it in a way that ensures that computation remains bounded at each time step, therefore facilitating online learning.

5 Experiments

In this section, we evaluate our moment matching techniques on synthetic data and a real world problem i.e. recognition of behaviours performed by older adults while using their walker.

5.1 Synthetic Data

We evaluate our moment matching algorithms on synthetic HMMs of n states and n observations where the observation function is known and the transition function is unknown. The observation distributions are parameterized as follows: $\Pr(E_t = e|Y_t = y) = \gamma$ when $e = y$ and $(1-\gamma)/(1-n)$ otherwise. The higher γ is, the more informative are the observations and when $\gamma = 1/n$, the observations are sampled uniformly and are uninformative.

Let θ_t be the true underlying parameter calculated by counting the number of transitions till time t. We denote by $\bar{\theta}_t$, the expected value of the parameter corresponding to the first moment of the posterior at each time step t calculated using moment matching. We want to report an estimate of $Err = \int_\theta \sum_i (\theta_i - \bar{\theta})^2 P(\theta) d\theta$. We sample 20 sequences and report the mean and standard deviation of the error over time.

In case of a single Bernoulli parameter, the naive exact Bayesian learning approach computes a mixture of Betas that corresponds to a polynomial with an increasing degree, requiring memory and time that grows linearly with the amount of data at each

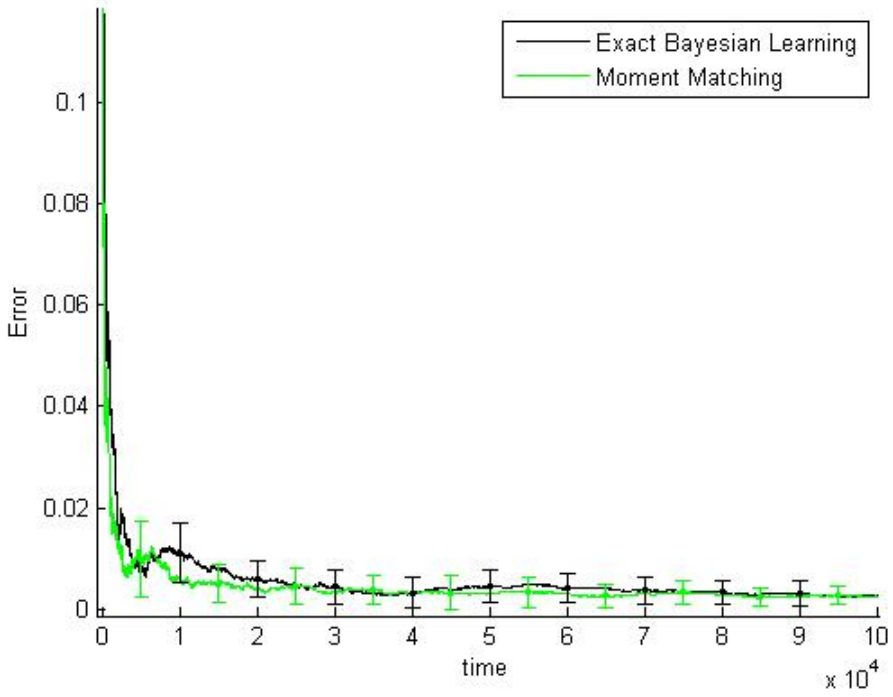


Figure 1: Evolution of Err_{Beta} and Err_{Exact} over time for 2-state HMMs with $\theta = 0.75$ and $\gamma = 0.8$. Comparison between exact Bayesian learning and our moment matching algorithms.

step. We compare our moment matching algorithm for that model to exact Bayesian learning. $\hat{\theta}_t$ is the expected value of the parameter corresponding to the first moment of the posterior calculated using exact naive Bayesian learning. We generate 20 sequences of length 100,000. Fig. 1 compares the error achieved by our moment matching algorithm vs. the error for exact Bayesian learning.

We report results for Alg. 1 and 2. It turns out that even though those two algorithms are different, their results are nearly the same and therefore there is only one visible curve. The two algorithms differ in the choice of the implicit prior(s). However, the first 3 moments of their priors are the same. Only the higher moments may differ. The results suggest that those higher moments are implicitly chosen in almost the same way since the resulting curves are indistinguishable. This makes sense since both algorithms differ only in the choice of the third moment of one P_t^y and hence we are showing only one curve. Both exact Bayesian learning and moment matching converge at a similar rate to the true underlying θ, thereby supporting our claims in the previous section. Moment matching took 9 seconds to process all 10,000 observations where as exact Bayesian learning took 271 seconds.

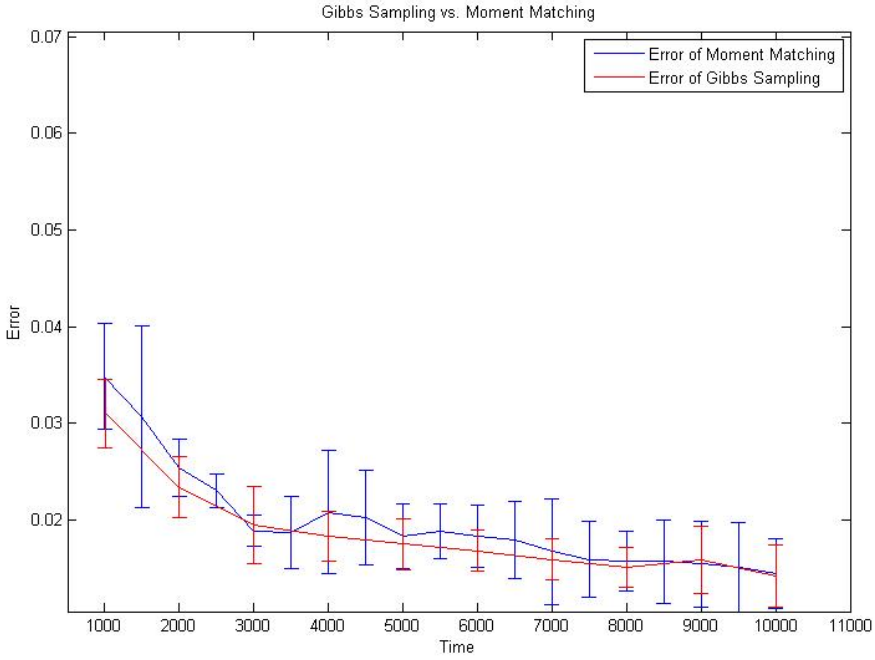


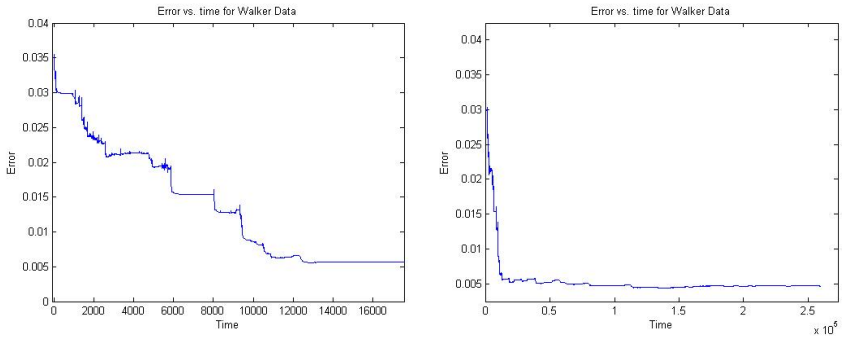
Figure 2: Evolution of Err_{Dir} over time for 4-state HMMs

We also report the results of Alg. 2 for a product of Dirichlets for a 4-state HMM in Fig. 2. Here we use 10 sequences of length 20,000. The transition function of the HMM has been generated randomly. We did not compare to exact Naive Bayesian learning since the computational complexity grows exponentially with the length of sequence, which is intractable. Instead, we compare our results to Gibbs Sampling since Gibbs samping is also exact in the limit. However, Gibbs sampling does not run in an online fashion since the amount of computation increases as the sequence of observations increases. It must resample all previous hidden states multiple times (500 times in our experiments) for each new observation. Gibbs sampling took 17 minutes to process the last observation where as our moment matching technique took only 0.005 seconds per observation (including the last observation). The results of both algorithms are similar (differences are not statistically significant).

5.2 Activity Recognition for Walker users

Rollating walkers are popular mobility aids used by older adults to improve balance control. There is a need to automatically recognize the activities performed by walker users to better understand activity patterns, mobility issues and the context in which falls are more likely to happen. We have access to a walker equipped with various sensors including three accelerometers that record the acceleration across the x, y and

Figure 3: The evolution of error over time. The error converges close to the true value at around time 16,000



z axis, load-cells in each leg of the walker to measure the vertical forces and a wheel encoder to record the distance traveled. We have previously discussed that the class of sticky HMMs serves as a good model for such activity recognition tasks. Here, we assume that each of the 12 states (corresponding to 12 different activities) have the some probability θ_y of persisting and the same probability $(1 - \theta_y)/(n - 1)$ of switching to another state. We learned Θ using the exact moment matching procedure described in Alg. 2. Fig. 3 shows that the error goes down over time and the algorithm is able to converge close to the true value of θ obtained by handlabeling the activities at each time step.

6 Conclusion and Future Work

In this work we described moment matching techniques for Bayesian learning of the transition parameters of HMMs. To our knowledge this is the first time that an exact Bayesian learning algorithm is derived while ensuring a constant amount of computation at each time step, which facilitates online learning. The approach delays the specification of the moments of the prior until they are needed in the computation, which allows us to choose them in a way that keeps the amount of computation bounded. In the future we plan to extend this work to allow for learning the observation function as well.

References

[1] 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.

[2] Byron Boots and Geoffrey J. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In Wolfram Burgard and Dan Roth, editors, *AAAI*. AAAI Press, 2011.

[3] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later, 2011.

[4] 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.

[5] Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. In *COLT*, 2009.

[6] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In Jack S. Breese and Daphne Koller, editors, *UAI*, pages 362–369. Morgan Kaufmann, 2001.

[7] Farheen Omar, Mathieu Sinn, Jakub Truszkowski, Pascal Poupart, James Tung, and Allan Caine. Comparative analysis of probabilistic models for activity recognition with an instrumented walker. In *UAI*, pages 392–400, 2010.