Department of Computer Science
George Mason University
Technical Report Series

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/
703-993-1530

Fixed Length Representations of Sequence Data Using a Variant of the Hidden Markov Model

Sam Blasiak sblasiak@gmu.edu
Huzefa Rangwala rangwala@cs.gmu.edu

Technical Report GMU-CS-TR-2010-14

Abstract

Sequence classification is central to many practical problems within machine learning. Classification algorithms often center around the notion of a distance metric between examples. Unlike sequences, the Euclidean distance metric between vectors often has an intuitive meaning which transfers naturally to a meaning in the classification domain. Distances metrics between arbitrary pairs of sequences, however, can be harder to define because sequences can vary in both length and the information contained in the order of sequence elements is lost when standard distance metrics are applied. We present a scheme that employs a Hidden Markov Model variant to produce a set of fixed-length vectors from a set of sequences. We then define three inference algorithms, a Baum-Welch variant, a Gibbs Sampling algorithm, and a variational algorithm, to infer model parameters. Finally, we show experimentally that the fixed length representation produced by these inference methods is useful for classifying proteins by structural taxonomy.

1 Introduction

The need to operate on sequence data is prevalent in a variety of real world applications ranging from protein/DNA classification, speech recognition, intrusion detection and text classification. Sequence data can be distinguished from the more-typical vector representation in that the length of sequences within a dataset can vary and that the order of symbols within a sequence carries meaning.

For sequence classification, a variety of strategies, depending on the problem type, can be used to map sequences to a representation that can be handled by traditional classifiers. A simple technique involves selecting a fixed number of elements from the sequence and then using those elements as a fixed-length vector in the classification engine. In another technique, a small subsequence length, ℓ, is selected, and a size M^ℓ vector is constructed containing the counts of all length ℓ subsequences from the original sequence. This vector can then be used for classification. A third method for classifying sequence data requires only a positive definite mapping from be defined between sequences rather than any direct mapping of sequences to vectors. This strategy, known as the kernel trick, is often used in conjunction with support vector machines and allows for a wide variety of sequence similarity measurements to be employed.

Hidden Markov Models (HMM) [20, 6] have a rich history for modeling sequence data (in speech recognition and bioinformatic applications) for the purposes of classification, segmentation, and clustering. HMMs' success is based on the convenience of their simplifying assumptions. The space of probable sequences is contrained by assuming only pairwise dependencies over hidden states. Pairwise dependencies also allow for a class of efficient inference algorithms whose critical steps build on the Forward-Backward algorithm [20].

We present an HMM variant over a set of sequences, with one transition matrix per sequence, as a novel alternative for handling sequence data. After training, the per-sequence transition matrices of the HMM variant are used as fixed-length vector representations for each associated sequence. There

are a number of ways for understanding how the HMM variant represents sequence data, and we connect these ways of understanding to both traditional explanations using simple Hidden Markov Models and more recent interpretations arising from topic models [4]. We then describe three methods to infer the parameters of our HMM variant, explore connections between these methods, and provide rationale for the classification behavior of the parameters derived through each.

We perform a comprehensive set of experiments, evaluating the performance of our method in conjunction with support vector machines, to classify synthetically generated data and sequences of amino acids into structural classes (fold recognition and remote homology detection problem [22]).

The combination of these methods, their interpretations, and their connections to prior work constitutes a new twist on classic ways of understanding sequence data that we believe is valuable to anyone approaching a sequence classification task and constitutes a significant contribution of this work.

2 Problem Statement

Given a set of sequences, we would like to find a set of fixed-length vectors, A, that, when used as input to a function $f(A)$, maximize the probability of reconstructing the original set of sequences. Under our scheme, $f(A)$ is a Hidden Markov Model variant with one transition matrix, A_n, assigned to each sequence, a single emissions matrix, B, and a single start probability vector, a, for the entire set of sequences. By maximizing the likelihood of the set of sequences under the HMM variant model, we will also find the set of transition matrices that best represent our set of sequences. We further postulate that this maximum likelihood representation will achieve good classification results if each sequence is later associated with a meaningful label.

2.1 Model Description We define a Hidden Markov Model variant that represents a set of sequences. Each sequence is associated with a separate transition matrix, while the emission matrix and initial state transition vector are shared across all sequences. We use the value of each transition matrix as a fixed-length representation of the sequence. We define the parameters and notation for the model in Table 1.

The probability of the model is shown below:

Parameter	Description
N	the number of sequences
T_n	the length of sequence n
K	the number of hidden symbols
M	the number of observed symbols
a_i	start state probabilities, where i is the value of the first hidden state
A_{nij}	transition probabilities, where n indicates the sequence, i the originating hidden state, and j the destination hidden state
B_{im}	emission probabilities, where i indicates the hidden state, and m the observed symbol associated with the hidden state
z_{nt}	the hidden state at position t in sequence n
x_{nt}	the observed state at position t in sequence n

Table 1: HMM Variant model parameters

$$(2.1)\quad p(x,z|a,A,B) = \prod_{n=1}^{N}\left(a_{z_{n1}}\left(\prod_{t=2}^{T_n}\mathbf{A_{nz_{nt-1}z_{nt}}}\right)\left(\prod_{t=1}^{T_n}B_{z_{nt}x_{nt}}\right)\right)$$

This differs from the standard hidden Markov model only in the addition of a transition matrix for each sequence. The probability of a set of sequences under the standard HMM is shown below (differences highlighted in bold):

$$(2.2)\quad p(x,z|a,A,B) = \prod_{n=1}^{N}\left(a_{z_{n1}}\left(\prod_{t=2}^{T_n}\mathbf{A_{z_{nt-1}z_{nt}}}\right)\left(\prod_{t=1}^{T_n}B_{z_{nt}x_{nt}}\right)\right)$$

To regularize the model, we further augment the basic HMM by placing Dirichlet priors on a, each row of A, and each row of B. The prior parameters are the uniform Dirichlet parameters γ, α, and β for a, A, and B respectively. The probability of the model with priors is shown below, where the prior probabilities are the first three terms in the product below and take the form $Dir(x;a,K) = \frac{\Gamma(Ka)}{\Gamma(a)^K}\prod_i x_i^{a-1}$:

$$(2.3)\quad p(x,z,a,A,B|\alpha,\beta,\gamma) = \left(\frac{\Gamma(K\gamma)}{\Gamma(\gamma)^K}\prod_i a_i^{\gamma-1}\right)\left(\prod_{ni}\frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K}\prod_j A_{nij}^{\alpha-1}\right)\left(\prod_i\frac{\Gamma(M\beta)}{\Gamma(\beta)^M}\prod_m B_{im}^{\beta-1}\right)\prod_{n=1}^{N}\left(a_{z_{n1}}\left(\prod_{t=2}^{T_n}A_{nz_{nt-1}z_{nt}}\right)\left(\prod_{t=1}^{T_n}B_{z_{nt}x_{nt}}\right)\right)$$

One potential difficulty that could be expected in classifying simple HMMs by transition matrix is that the probability of a sequence under an HMM does not change under a permutation of the hidden states. This problem is avoided when we force each sequence to share an emissions matrix, which locks the meaning of each transition matrix row to a particular emission distribution. If the emission matrix were not shared, then two HMMs with permuted hidden states could have transition matrices that with large Euclidean distances. For instance, the following HMMs have different transition matrices, but the probability of an observed sequence is the same under each.

$$HMM_1 : A_1 = \begin{bmatrix} .9 & .1 \\ .9 & .1 \end{bmatrix},\ B_1 = \begin{bmatrix} .9 & .1 \\ .1 & .9 \end{bmatrix}$$

$$HMM_2 : A_2 = \begin{bmatrix} .1 & .9 \\ .1 & .9 \end{bmatrix},\ B_2 = \begin{bmatrix} .1 & .9 \\ .9 & .1 \end{bmatrix}$$

However, a Euclidean distance between their two transition matrices, A_1 and A_2 is large.

2.2 A simple example To gain an intuitive understanding of how our scheme operates, consider the following scenario. Assume that instead of learning the parameters of our emissions matrix, B, we fix B so that row m describes a multinomial distribution with probability of 1 of emitting the m^{th} observed symbol and zero probability of emitting any other symbol. For instance, if we have three possible emission symbols, $[a,b,c]$, then $M=3$, $K=3$, and B is set to I_3:

$B =$

	x_a	x_b	x_c
z_1	1	0	0
z_2	0	1	0
z_3	0	0	1

Because there is a deterministic correspondence between observed and hidden states, the hidden states are effectively observed, and A_{nij} is simply $P(x_{nt}=j|x_{nt-1}=i)$, which can be estimated by taking the normalized count of the number of pairs of symbols ij in the sequence: $\{\#t : x_{nt-1}=i, x_{nt}=j,\ t>1\}$ divided by the total number of x_{nt} with the value i,$\{\#t : x_{nt}=i,\ t<T_n\}$.

Our HMM variant is similar to this simplified scheme, but the number of hidden states, K, is set beforehand, and an inference algorithm is used to jointly optimize the transition and emission matrices to capture the best representation of the set of input sequences.

2.3 Another interpretation Earlier we noted that we can interpret each transition matrix A_n as the argument to a function that allows us to reconstruct the sequence x_n with minimum error.

Using the basic HMM, we can describe a method to perform this reconstruction: first, an initial hidden state, z_{n1}, is sampled from a. Next, for every z_{nt} with $1<t\leq T_n$, z_{nt} is sampled from a multinomial with parameters $A_{nz_{nt-1}}$. Finally, each observed sequence element, x_{nt} is sampled from a multinomial with parameters $B_{z_{nt}}$.

Given a single sequence, we can create a standard HMM, with a probability of 1 of regenerating the source sequence by setting the number of hidden states equal to the length of the sequence, $K=T$. Next, at each hidden state, k, we set the probability of a transition to the hidden state $k+1$ to one and transitions to any other hidden states to zero. Finally, we set the matrix B so that at hidden state $k=t$ we emit the symbol x_k.

Taking this idea further, we can see intuitively how the size of A relates to some measure of information in the sequence. To best illustrate this, if we take a sequence that consists of two repeated sections, we would need only $K=\frac{T}{2}$ states to reconstruct it without error (with n repeats we would need $K=\frac{T}{n}$ states) because we can set state K to deterministically transition to state 1.

If the set of sequences has varying amounts of information per sequence, then, for small values of K, our scheme will be able to reconstruct low-information sequences with small error but will have a high error rate when reconstructing high-information sequences. If we choose a large K, then our scheme will be able to reconstruct high-information sequences well, but for low-information sequences some values of A will be meaningless.

3 Background

3.1 Mixtures of HMMs HMMs have a rich history in sequence classification and clustering [20, 6]. Smyth introduces a mixture of HMMs in [25] and presents an initialization technique that is similar to our model in that an individual HMM is learned for each sequence, but differs from our model in that the emission matrices are not shared between HMMs. In [25], these initial N models are used to compute the set of all pairwise distances between sequences, defined as the symmetrized log likelihood of each element of the pair under the other's respective model. Clusters are then computed from this distance matrix, which are used to initialize a set of $K < N$ HMMs where each sequence is associated with one of K labels. Smyth notes that while the log probability of a sequence under an HMM is an intuitive distance measure between sequences, it is not intuitive how the parameters of the model are meaningful in terms of defining a distance between sequences. In this research, we demonstrate experimentally that the transition matrix of our model is useful for sequence classification when combined with standard distance metrics and tools.

3.2 Topic Models Simpler precursors of LDA [4] and pLSI [10], which represent an entire corpus of documents with a single topic distribution vector, are very similar to the basic Hidden Markov Model, which assigns a single transition matrix to the entire set of sequences that are being modeled. To extend the HMM to a pLSI analogue, all that is needed is to split the single transition matrix into a per-sequence transition matrix. To extend this model to an LDA analogue, we must go a step further and attach Dirichlet priors to the transition matrices.

Inference of the LDA model (Figure 1a) on a corpus of documents learns a matrix of document-topic probabilities. A row of this matrix, sometimes described as a mixed-membership vector, can be viewed as a measurement of how a given document is composed from the set of topics. In our HMM variant (Figure 1c), a single transition matrix, A_n, can be thought of as the analogue to a document-topic matrix row and can be viewed as a measurement of how a sequence is composed of pairs of adjacent symbols.

More recent topic models contain significant similarities to our HMM variant. Both the Hidden Topic Markov Model (HTMM) (Figure 1b) [9] and Conditional Topic Random Fields (CTRF) [27] are similar to our HMM variant in that they add pairwise dependencies between hidden topics to the LDA model. The key difference between our HMM variant and the HTMM lie in the HTMM's explicit modeling of text. Like LDA, the HTMM assigns one topic composition vector to each document. Dependencies between topics of adjacent words are modeled using a separate binomial parameter and associated set of indicator hidden variables per topic, rather than using a transition matrix like the HMM variant. This binomial parameter has the effect of restricting the possible transitions between topics according to a per-sentence composition. For the CTRF, hidden states (topics) are modeled using a conditional random field .

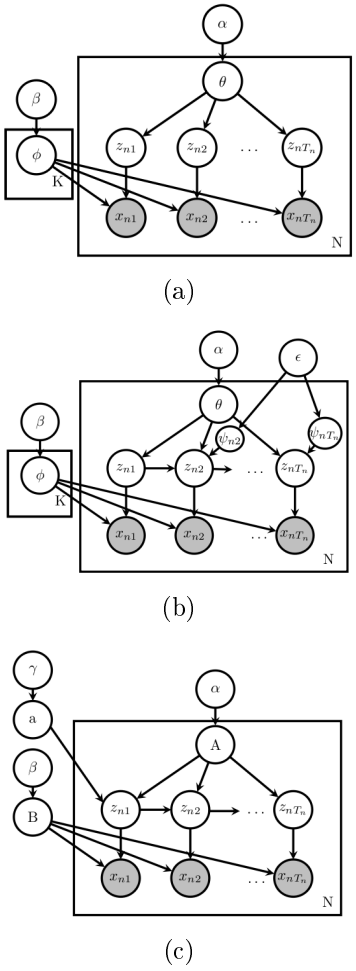

Figure 1: Plate diagrams of the (a) LDA model, expanded to show each word separately, the (b) Hidden Topic Markov Model, and the (c) HMM variant.

4 Learning the model parameters

4.1 Baum-Welch A well-known method for learning HMM model parameters is the Baum-Welch algorithm. The Baum-Welch algorithm is an expectation maximization algorithm for the standard HMM model, and the basic algorithm is easily modified to learn the multiple transition matrices of our

variant. The parameter updates shown below converges to a maximum a priori (MAP) estimate of $p(z, a, A, B|x, \gamma, \alpha, \beta)$ [20]:

$$a_i \propto \sum_n f_{ni}(1) b_{ni}(1) + \gamma - 1 \tag{4.4}$$

$$A_{nij}^{(new)} \propto \sum_{t=2}^{T_n} f_{ni}(t-1) A_{nij} B_{jx_t} b_{nj}(t) + \alpha - 1 \tag{4.5}$$

$$B_{im} \propto \sum_n \sum_{t:x_t=m} f_{ni}(t) b_{nj}(t) + \beta - 1 \tag{4.6}$$

where f and b are the forward and backward recursions defined below:

$$f_{ni}(t) = \begin{cases} \sum_j f_{nj}(t-1) A_{nji} B_{ix_t}, & t > 1 \\ a_i B_{ix_1}, & t = 1 \end{cases} \tag{4.7}$$

$$b_{ni}(t) = \begin{cases} \sum_j A_{nij} B_{jx_{t+1}} b_{nj}(t+1), & t < T_n \\ \frac{1}{K}, & t = T_n \end{cases} \tag{4.8}$$

The complexity of the Baum-Welch-like algorithm for our variant is identical to the complexity of Baum-Welch for the standard HMM. The update for A_{ij} in the original HMM involves summing over $\sum_n T_n$ terms, while the update for a single A_{nij} is a sum over T_n terms, making the total number of terms over all the A_n's in our variant, $\sum_n T_n$, which is the same as number the original algorithm.

4.2 Gibbs Sampling Two Gibbs sampling schemes are commonly used to infer Hidden Markov Model parameters [23]. Unlike the Baum-Welch algorithm which returns a MAP estimate of the parameters, these sampling schemes allow the expectation of the parameters to be computed over the posterior distribution $p(z, a, A, B|x, \gamma, \alpha, \beta)$.

In the Direct Gibbs sampler (DG), hidden states and parameters are initially chosen at random, then new hidden states are sampled using the current set of parameters:

$$p(z_{ti}^{(new)}|z_{t-1}, z_{t+1}) \propto A_{z_{t-1}i} B_{ix_t} A_{iz_{t+1}} \tag{4.9}$$

In the Forward Backward sampler (FB), the initial settings and parameter updates are the same as the DG scheme, but the hidden states are sampled in order from T_n down to 1 using values from the forward recursion. Specifically, each hidden state z_{nt} is sampled given $z_{nt+1} = j$ from a multinomial with parameters

$$p(z_{nT_n}^{(new)}|x_{n1:T_n}) \propto f_{ni}(T_n) \tag{4.10}$$

$$\begin{aligned} p(z_{nt}^{(new)}|x_{n1:T_n}, z_{nt+1}^{(new)}) &= p(z_{nt}^{(new)}|x_{n1:t}, z_{nt+1}^{(new)}) \\ &\propto f_{ni}(t) A_{nij}, \qquad t < T_n \end{aligned} \tag{4.11}$$

In both algorithms, after the hidden states are sampled, parameters are sampled from Dirichlet conditional distributions, shown for A below, where $\mathbb{I}(\omega) = 1$ if ω is true and 0 otherwise:

$$p(A_{nij}|z_n, \alpha) = Dir(\sum_{t=2}^{T_n} \mathbb{I}(z_{nt-1} = i)\mathbb{I}(z_{nt} = j) + \alpha) \tag{4.12}$$

The FB sampler has been shown to mix more quickly than the DG sampler, especially in cases where adjacent hidden states are highly correlated [23]. We therefore use the FB sampler in our implementation.

4.3 Variational Algorithm Another approach for inference of the HMM variant parameters is through variational techniques. We employ a mean field variational algorithm that follows a similar pattern as EM. When the variational update steps are run until convergence, Kullback-Leibler divergence between the variational distribution, $q(z, a, A, B)$, and the model's conditional probability distribution, $p(z, a, A, B|x, \gamma, \alpha, \beta)$, is minimized. The transition matrices returned by the variational algorithm are the expectations of those matrices under the variational distribution. Thus, like the Gibbs sampling algorithm, the parameters returned by the variational algorithm approximate the expectations of the parameters under the conditional distribution.

Our mean field variational approximation is shown below:

SCOP Version	Filtering	Taxonomic type	# sequences	# categories	SVM classifier
1.67	25%	class	4995	7	multiclass
1.67	25%	fold	1127	25	multiclass
1.67	40%	fold	1653	27	multiclass
1.67	40%	superfamily	1044	37	multiclass
1.53	25%	superfamily	4352	23	one-versus-rest

Table 2: Datasets used to evaluate the HMM variant's ability to classify protein sequences.

(4.13)

$$\begin{aligned} q(z,a,A,B) =& q(a)\prod_{n=1}^{N}\prod_{i=1}^{K} q(A_{ni})\prod_{i=1}^{K} q(B_i)\prod_{nt} q(z_{nt}) \\ =& \left(\frac{\Gamma(\sum_i \tilde{\gamma}_i)}{\prod_i \Gamma(\tilde{\gamma}_i)}\prod_i a_i^{\tilde{\gamma}_i - 1}\right) \\ & \left(\prod_{ni}\frac{\Gamma(\sum_j \tilde{\alpha}_{nij})}{\prod_j \Gamma(\tilde{\alpha}_{nij})}\prod_j A_{nij}^{\tilde{\alpha}_{nij}-1}\right) \\ & \left(\prod_i \frac{\Gamma(\sum_m \tilde{\beta}_{im})}{\prod_m \Gamma(\tilde{\beta}_{im})}\prod_m B_{im}^{\tilde{\beta}_{im}-1}\right)\prod_{nti} h_{nti}^{z_{nti}} \end{aligned}$$

with variational parameters h_{nti}, which acts as an approximate mean for each z_{nti}, and $\tilde{\alpha}_{nij}$, $\tilde{\beta}_{im}$, and $\tilde{\gamma}_i$, which can be thought of as Dirichlet parameters approximating α, β, and γ.

When we maximize the variational free energy with respect to the variational parameters, we obtain the following update equations, where $\Psi(x) = \frac{d\log\Gamma(x)}{dx}$:

$$\tilde{\alpha}_{nij} = \sum_t h_{nt-1i} h_{ntj} + \alpha \tag{4.14}$$

$$\tilde{\beta}_{im} = \sum_{nt:x_t=m} h_{nti} + \beta \tag{4.15}$$

$$\tilde{\gamma}_i = \sum_n h_{n1i} + \gamma \tag{4.16}$$

(4.17)

$$h_{nti} \propto \begin{cases} \exp\Big(\Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'}\tilde{\gamma}_{i'}) \\ \quad + \sum_j h_{n2j}\Big(\Psi(\tilde{\alpha}_{nij}) - \Psi(\sum_{j'}\tilde{\alpha}_{nij})\Big) \\ \quad + \Big(\Psi(\tilde{\beta}_{ix_{n1}}) - \Psi(\sum_m \tilde{\beta}_{im})\Big)\Big), \\ t = 1 \\ \exp\Big(\sum_{i'} h_{nt-1i'}\Big(\Psi(\tilde{\alpha}_{ni'i}) - \Psi(\sum_j \tilde{\alpha}_{ni'j})\Big) \\ \quad + \sum_j h_{nt+1j}\Big(\Psi(\tilde{\alpha}_{nij}) - \Psi(\sum_{j'}\tilde{\alpha}_{nij'})\Big) \\ \quad + \Big(\Psi(\tilde{\beta}_{ix_{nt}}) - \Psi(\sum_m \tilde{\beta}_{im})\Big)\Big), \\ 1 < t < T_n \\ \exp\Big(\sum_{i'} h_{nt-1i'}\Big(\Psi(\tilde{\alpha}_{ni'i}) - \Psi(\sum_j \tilde{\alpha}_{ni'j})\Big) \\ \quad + \Big(\Psi(\tilde{\beta}_{ix_{nT_n-1}}) - \Psi(\sum_m \tilde{\beta}_{im})\Big)\Big), \\ t = T_n \end{cases}$$

Notice that the update for h_{nti} depends only on the adjacent h's, h_{nt-1i} and h_{nt+1i} as well as the expectations of the transition probabilities from the adjacent h's and the expectation of the emission probabilities from the current h_{nti}. This mean field algorithm can therefore be understood as an equivalent of the Direct Gibbs sampling method except that at subsequent time steps interactions occur between variational approximations rather than through the sampled values of z. A complete derivation of the variational algorithm is included on the author's website [1].

Other authors have performed variational inference over hidden Markov models. Most notably, Ghahramani derives variational updates to iteratively optimize parameters of factorial HMMs [8] and switching state space models [7]. MacKay derives a variational ensemble for HMM posterior parameters [17]. Our variational algorithm is most similar to the mean field variational algorithm from Ghahramani in [8].

5 Experimental Setup

5.1 Protocol To evaluate our fixed-length representation scheme, for each classification experiment, we created three sets of fixed-length representations per trial over ten trials by running each of the three inference algorithms: (i) Baum-Welch, (ii) Gibbs Sampling, and (iii) the mean field variational algorithm, on the entire set of input data. We varied the number of hidden states from 5 to 20 in increments of 5 ($K = \{5, 10, 15, 20\}$). This procedure created a total of 120 ($3 \times 10 \times 4$) fixed-length representations for each dataset.

The fixed-length vector data was then used as input to a support vector machine (SVM) classifier [2]. We used the SVM to either perform either multiway classification on the dataset under the Crammer-Singer [5] construction or the one-versus-rest approach, where a binary classifier was trained for each of the classes. No attempt was made to optimize the SVM parameters in any of the experiments.

We compare classification results from our

[1] http://www.cs.gmu.edu/~mlbio/sdm10

[2] We used SVM-light and SVM-struct for classification (http://www.cs.cornell.edu/People/tj/svm_light/) [12].

model with results from the Spectrum(2) kernel for all experiments. The Spectrum(ℓ) kernel is a simple string kernel whose vector representation is the set of counts of substrings of of observed symbols length ℓ in a given string [15]. For the one-versus rest experiments, we compare our results to more sophisticated biologically sensitive kernels for protein classification, described in Rangwala et. al [21].

5.2 Protein Datasets The Structural Classification of Proteins (SCOP) [19] database categorizes proteins from the Protein Databank (PDB)[3] into a multilevel hierarchy that captures commonalities between protein structure at different levels of detail. The ASTRAL compendium[4], provides versions of SCOP datasets filtered to remove sequences whose structures are significantly similar, allowing for less biased classification. To evaluate our representation, we ran sets of protein classification experiments on the three top levels of the SCOP taxonomy, class, fold, and superfamily. Table 2 provides a detailed description of the different SCOP-derived protein sequence classification datasets, that were obtained from previous studies [22, 13]. Specifically, the datasets were derived from either the SCOP 1.67 or the SCOP 1.53 versions and filtered at 25% and 40% pairwise sequence identities. A protein sequence dataset filtered at 25% identity will have no two sequences with more than 25% sequence identity.

We partitioned the data into a single test and training set for each category. At the class level, the original dataset was split randomly in to training and test sets. To eliminate high levels of similarity between sequences that could lead to trivially good classification results, we imposed certain constraints on the training/test set partitioning for classification in the fold and superfamily experiments. For the fold level classification problem, the training sets were partitioned so that no examples that shared the fold and superfamily labels were included in both the training and test sets. Similarly, for the superfamily level classification problem (referred to as the remote homology detection problem [15, 21]), no examples that shared the superfamily and family levels were included in both the training and test sets.

5.3 Synthetic Datasets As a basic test of concept, we constructed synthetic datasets by drawing samples from two HMMs with transition matrices

$$A_1 = \begin{bmatrix} .6 & .4 \\ .4 & .6 \end{bmatrix}, \; A_2 = \begin{bmatrix} .4 & .6 \\ .6 & .4 \end{bmatrix}$$

with a discrete emissions matrix where each state's output is an eight state discretized version of a univariate normal distribution with means of 3 and 5 respectively. These HMMs were constructed to be discretized versions of the HMMs used to generate synthetic data in a previous study [25].

We ran experiments on a dataset consisting of 500 samples from each HMM with the length of each sequence Poisson distributed with a mean of 100. We then randomly partitioned the dataset into 20% test and 80% training samples and used the protocol described above to classify training samples by their generating HMM. The experiment was run 10 times on separately generated datasets.

For the synthetic experiments, we included two additional kernels. The first, a Kullback-Leibler Divergence Kernel, [18], was used to attempt to answer the question whether a "natural" metric over the space of transition matrices produces better results than the standard dot product. Because each transition matrix row resides in a K-simplex, we considered the possibility that a symmetrized DKL measurement between rows of different matrices would allow for better comparisons.

We also tested a representation, that we call the gradient kernel, where the feature vector for each sequence was given by the gradient of the log probability of a sequence with respect to the transition matrix associated with that sequence. This representation is similar to the Fisher kernel [11] but differs from the standard Fisher kernel in that the scheme does not take advantage of information about the classes associated with sequences in the training set. A more complete version of the Fisher kernel for the HMM variant would take into account sequence labels in the training set and would require taking the gradient of a sequence under the portion of the HMM variant model associated with each positive and negative training example, a computationally intensive task.

5.4 Evaluation Metrics We evaluated each classification experiment by computing the area under the ROC curve (AUC), a plot of the true positive rate against the false positive rate, constructed by adjusting the SVM's intercept parameter. We also computed the AUC50 value, which is a normalized computation of the area under the ROC curve until the first 50 false positives have been detected. We were worried about variance over different Baum-

[3] http://www.pdb.org/pdb/home/home.do
[4] http://astral.berkeley.edu/

Welch runs due to convergence of the algorithm to different local optima. To mitigate this concern, we ran both the Baum-Welch algorithm and the other inference algorithms, for consistency, 10 separate times on each dataset. The results presented for each inference method are averages over individual results of the 10 trials across the different classes.

6 Results and Discussion

6.1 Protein Sequence Classification Table 3 shows a comparison of results (average AUC scores) across the inference algorithms in three taxonomic categories (class, fold, and superfamily) using the multiclass SVM. Although the AUC scores are close for each algorithm in most cases, the Gibbs sampling algorithm outperforms the other algorithms the majority of the time.

Table 4 shows a comparison of results over the inference algorithms but only for the one-versus-rest superfamily classification experiment on the SCOP 1.53 dataset. Similarly to the multiclass experiments using the linear kernel, the Gibbs sampling algorithm outperforms the other inference methods in the one-versus-rest experiments. Although the values of the AUC and AUC50 scores do not significantly change between the Gaussian and linear kernels, the variational algorithm shows the largest improvement in the AUC scores, ranging from 6% to 30%, when used with the non-linear kernel.

The results over multiple trials also revealed an interesting characteristic of the data. Although the standard deviation of the AUC score between categories within a single trial is high, with an average of 0.17 over 10 trials, the standard deviation of the average AUC score over all categories was low (0.03 averaged over 10 trials), indicating that a trade-off is occurring. If the AUC on a single taxonomic category is high in one trial, then it will generally be offset by a low AUC score on another taxonomic category and vice versa. Figure 2 graphically illustrates this trade-off, showing an overlay of AUC curves for fold classification over 10 Baum-Welch transition matrix representations.

6.2 Synthetic Results Table 5 compares average AUC and AUC50 scores from the synthetic data classification experiments between each inference algorithm and the Spectrum(2) kernel. The table also shows the effects of a variety of additional kernels over the basic description vectors. Notably, the Baum-Welch algorithm outperforms the others inference algorithms under both the linear kernel and all of the external kernels. No HMM variant formulation performs better on the synthetic data than the Spectrum(2) vector under a linear kernel. However, the spectrum kernel results degrade across the non-linear kernels, while the HMM variant results are consistent.

AUC

Alg/Kernel	Linear	Gaussian	DKL	Gradient
Baum Welch	0.894	**0.926**	**0.919**	**0.860**
Gibbs	0.606	0.537	0.563	0.828
Variational	0.562	0.562	0.578	0.546
Spectrum	**0.997**	0.472	0.548	N/A

AUC50

Alg/Kernel	Linear	Gaussian	DKL	Gradient
Baum Welch	0.837	**0.905**	**0.890**	**0.829**
Gibbs	0.570	0.523	0.536	0.797
Variational	0.533	0.553	0.560	0.468
Spectrum	**0.994**	0.511	0.503	N/A

Table 5: AUC and AUC50 results from all synthetic data experiments averaged over 10 trials. For each HMM variant, the number of hidden states is 2. Counts of substrings of length 2 were used to construct the spectrum kernel. The best performing entry is marked in bold.

We compared several kernel functions over the HMM variant sequence representation vectors. Although the DKL kernel appears to perform better than the linear kernel, it does not outperform the Gaussian. Similarly, the gradient kernel does not consistently perform better than the Gaussian. The Gibbs Sampling algorithm performs comparatively better under the Fisher kernel, a trend not reflected in the variational algorithm's performance.

6.3 Analysis of inference algorithms We were initially puzzled by the significant differences in AUC values resulting from the different training algorithms of our model in both the protein classification experiments (Tables 3 and 4) and especially in the synthetic data (Table 5).

A potential explanation of these differences lies in the targets of each algorithms objective function. While the Baum-Welch algorithm returns MAP parameters of the model, both the Gibbs sampling method and the variational algorithm produce expectations of the parameters. These different convergence points would seem to make a much larger difference in the results if the posterior distribution of the transition matrix is highly multimodal. Under a multimodal distribution, we expect the Baum-Welch algorithm to converge to a local maximum at one of these modes. In contrast, the expectation computed under both the Gibbs sampling and the variational algorithm may lie at a point in the parameter space described by the weighted center of mass of these modes. Different topologies of the posterior distribution will clearly have an effect on how

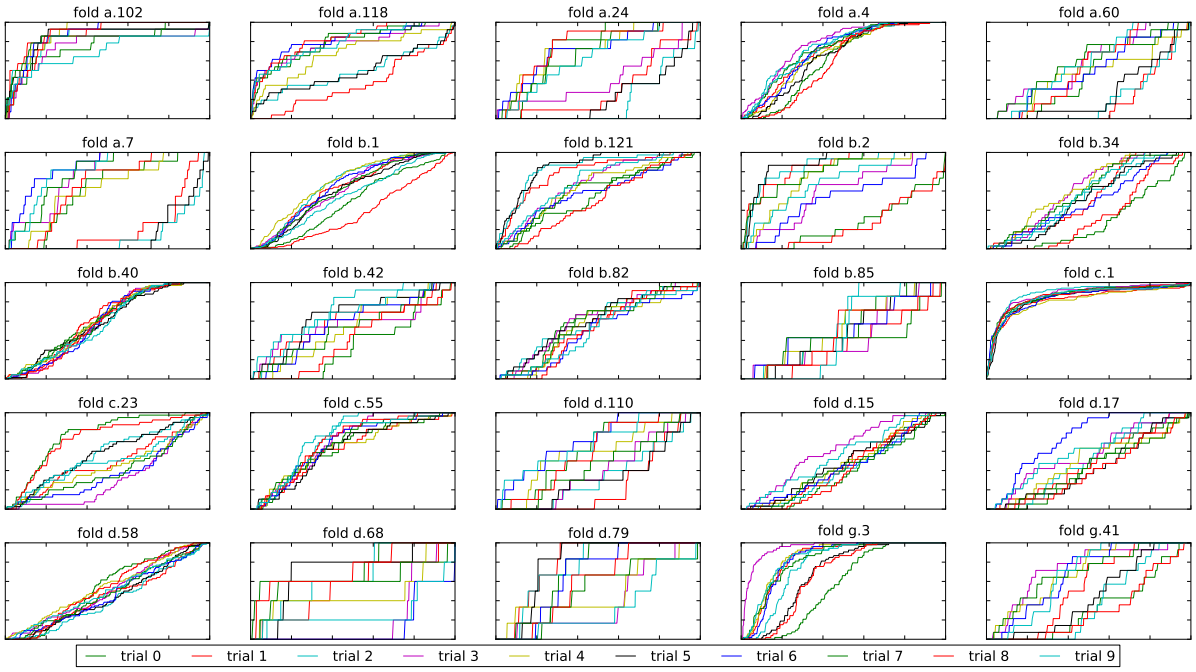

Figure 2: A comparison of AUC plots for the Baum Welch algorithm for $K = 10$ on the SCOP 1.67, 25% fold recognition dataset (25 classes) over a set of 10 parameters learned through randomly initialized Baum-Welch runs. From the plots, we can see that the variance of the classification of individual results can be high, especially for the classes with a small number of examples. However, there was a comparatively smaller amount of variation ($\sim 3\%$) in the average AUC score over all classes.

well the expected value of the parameters compares to the MAP parameters. If many maxima occur in a single region, then the expected parameter may perform well. However, if a small number of maxima are highly separated, then the MAP solution, which will likely reside at one of the maxima, will probably contain more information about the sequence than the expected parameter value.

To test this hypothesis, we constructed histograms of transition matrices using 1000 samples from the Gibbs sampling algorithm. Figure 3 shows a histogram of the 2×2 transition matrix associated with the first sequence from the synthetic data and is clearly bimodal with the modes at opposite ends of the distribution for each matrix entry. Similarly, Figure 4 shows a histogram of the 5×5 transition matrix associated with the first sequence from the SCOP 1.67, 25% Astral filtering dataset. This histogram exhibits much less multimodality than the histogram from the synthetic experiments. Taken together, the histograms, which approximate the posterior distribution of the transition matrices, illustrate how the multimodality seen in the posteriors of the synthetic data but not in posteriors of the the protein data likely causes the Gibbs sampling and variational transition matrices to perform poorly for

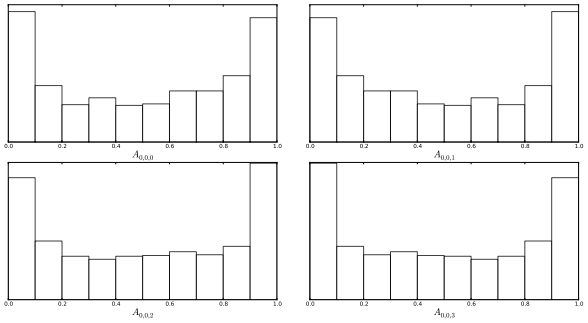

Figure 3: Histogram of the transition matrix associated with the first sequence of the synthetic dataset. The matrix entries have two modes at either end of the probability distribution.

classification.

To further analyze the results on the synthetic data, we observe the emission matrices learned by each inference method. There are clearly observable differences between the emission matrices returned from Baum-Welch and from both the Gibbs sampling and variational algorithm. Although all inferred emissions matrices exhibit some amount of bimodality, the degree of bimodality in the Baum-

Class Level, 7 categories
SCOP 1.67, 25%

Alg/K	5	10	15	**20**
Baum Welch	**0.61**	0.64	0.65	0.63
Gibbs Sampling	**0.61**	**0.65**	**0.66**	**0.68**
Variational	0.60	0.63	0.60	0.60

Fold Level, 25 categories
SCOP 1.67, 25%

Alg/K	5	10	15	**20**
Baum Welch	**0.56**	**0.59**	**0.59**	0.58
Gibbs Sampling	**0.56**	0.58	**0.59**	**0.61**
Variational	0.54	0.57	**0.59**	0.58

Fold Level, 27 categories
SCOP 1.67, 40%

Alg/K	5	10	15	**20**
Baum Welch	0.58	0.60	0.55	0.57
Gibbs Sampling	**0.60**	**0.63**	**0.65**	**0.67**
Variational	0.58	0.58	0.59	0.59

Superfamily, 37 categories
SCOP 1.67, 40%

Alg/K	5	10	**15**	20
Baum Welch	**0.59**	**0.63**	0.63	0.61
Gibbs Sampling	**0.59**	0.62	**0.64**	**0.63**
Variational	**0.59**	0.57	0.58	0.57

Table 3: AUC results from all of the multi-class SVM experiments are displayed. The best performing algorithm, the best performing setting of K, and the best combination of K and algorithm is marked in bold. The Gibbs-Sampling-derived representation most frequently returned the most accurate level of classification on the majority of the datasets.

Linear Kernel

Evaluation Metric	AUC				AUC50			
Algorithm/K	5	10	**15**	**20**	5	**10**	15	20
Baum Welch	0.58	0.55	0.52	0.57	0.18	0.17	0.15	0.24
Gibbs Sampling	**0.64**	**0.67**	**0.69**	**0.69**	**0.18**	**0.37**	**0.32**	**0.29**
Variational	0.63	0.59	0.54	0.58	0.17	0.11	0.19	0.17

Gaussian Kernel

Evaluation Metric	AUC				AUC50			
Algorithm/K	5	10	**15**	20	5	10	15	**20**
Baum Welch	0.61	0.60	0.58	0.59	**0.26**	**0.19**	**0.15**	**0.30**
Gibbs Sampling	0.63	**0.63**	0.63	0.63	0.20	0.11	0.11	0.11
Variational	**0.67**	0.60	**0.70**	**0.68**	0.23	0.16	0.14	0.16

Table 4: AUC and AUC50 results for protein superfamily classification AUC results on the SCOP 1.53 with 25% Astral filtering over a selected set of 23 superfamilies using Gaussian and linear kernels in one-versus-rest SVM classification.

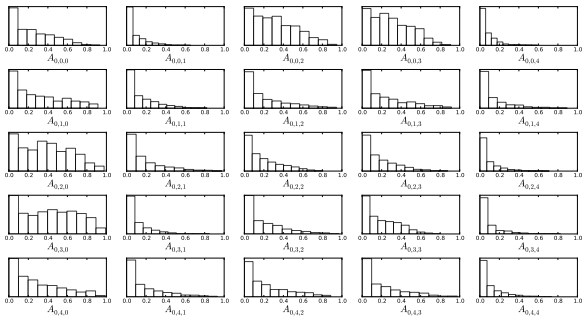

Figure 4: Histogram of the transition matrix associated with the first sequence of the SCOP 1.67 dataset with 25% Astral filtering. Modes of the distribution are more evenly distributed compared to the synthetic data transition matrices.

Welch emissions matrix is much less than the other algorithms (see Figure 5), allowing hidden states, and thus transitions between hidden states, to be more easily distinguished.

The gap in performance between the Gibbs sampling algorithm and the variational algorithm in the protein classification experiments is not as surprising as the different between the Baum-Welch algorithm and the rest. Both the Gibbs sampling algorithm and the variational algorithm compute expectations of the parameters under an approximate posterior distribution, but each uses a different method to construct this approximation. The variational algorithm will be less likely to converge to a good approximation of the marginal distribution because the mean field variational approximation necessarily does away with the direct coupling between adjacent hidden states characteristic of the HMM.

6.4 Comparison with standard structure classification methods Tables 5, 6, and 7 show a comparison between the HMM variant and common classification methods for the synthetic, multiclass, and one-versus rest experiments respectively. The AUC and AUC50 scores indicate that our scheme produces a representation that is roughly equivalent in power to the Spectrum kernel for protein classification but is outperformed by the Spectrum kernel for synthetic data classification. In defense of the HMM variant, the size of the vector representation produced by these kernels is significantly larger than the typical representations produced by our HMM variant. The Mismatch(5,1) kernel, used for SCOP 1.53 superfamily classification (Table 7), is similar to the Spectrum(5) kernel but also counts substrings of

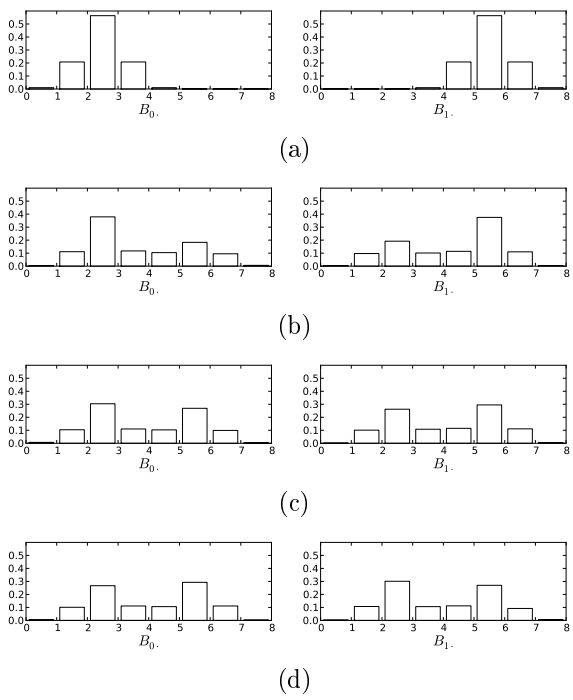

Figure 5: The original emission matrix used to generate the synthetic dataset (a) compared to typical emission matrices inferred the synthetic dataset using the Baum-Welch (b), Gibbs Sampling (c), and variational (d) algorithms. The charts in the first column show the first row of the matrix and the second column shows the second row of the matrix.

length 5 that differ by one amino acid residue from those found in an observed sequence. The size of the vector representation associated with this kernel can be up to 20^5. This value is large compared to the largest vector representation in our experiments, which is 400 for the HMM variant with 20 hidden states. Similarly, in the synthetic data classification task, the HMM variant uses a vector representation of length 4, while the Spectrum kernel vector representation is of length 64. Even though the vector representation associated with these kernels does not need to be explicitly computed, the size of the representation itself is an indication of the relative amounts of information used by these methods compared to the HMM variant.

The HMM variant does not perform as well as profile HMM kernels (AUC scores shown under the "SW-PSSM" entry in Table 7), which are another set of kernels commonly used for protein taxonomy classification. Nearly all of these high-performing kernel methods, unlike the HMM variant, employ domain specific knowledge, such as carefully tuned position-specific scoring matrices, to aid classification. In contrast, only parameter that needs to be adjusted in the HMM variant is the value of K.

Dataset/Kernel	HMM Variant	Spectrum
Class	0.68	0.66
Fold (25 Categories)	0.61	0.62
Fold (27 Categories)	0.67	0.67
Superfamily	0.66	0.64

Table 6: A comparison of results between the Spectrum kernel and the HMM variant under experiments using the multiclass SVM formulation. The HMM variant scores are the best performing from Table 3.

Algorithm	AUC	AUC50
HMM Variant (best)	0.67	0.37
Spectrum(2) [15]	0.712	0.290
Mismatch(5,1) [16]	0.870	0.416
Fisher [11]	0.773	0.250
SW-PSSM [21]	0.982	0.904

Table 7: A selection of AUC and AUC50 scores using a variety of SVM kernels on the same dataset (see [21] for details on additional kernel methods). The HMM variant scores are the best performing from Table 4

6.5 Number of Hidden States Tables 3 and 4 not only show how AUC scores are effected by the inference algorithm used but also how the performance of the algorithms changes as the number of hidden states change. For many of the trials, a maximum AUC or AUC50 score occur at 10 or 15 hidden states (although notably, the Gibbs sampling results appear to increase as K increases), indicating that the best performing value of K in each experiment is probably near the range that we tested.

Although we do not present experimental results with larger numbers of hidden states, our experiments show that larger values of K tend to produce worse AUC results (The Baum-Welch Algorithm with K=75 results in a AUC score of .54 and AUC50 of .18 on the class-level dataset using a Gaussian kernel, which was superior to the linear kernel's performance.). This trend can be accounted for using arguments similar to those that explain overfitting. As the transition matrix size increases it seems likely that the number of "junk" hidden states, hidden states that contribute to generating observed symbols for only a small number of sequences in the dataset, would also increase. These meaningless hidden states would cause distances between transition matrices to be corrupted with noise.

6.6 Higher Order Models In addition to the experiments described above, we also ran a limited number of experiments with higher order HMM variants, where transition matrices are defined between

the group of hidden states $z_{nt-\ell}, \ldots, z_{nt-1}$ and z_{nt}, where ℓ is the order of the HMM. Classification performance under this set of models was also inferior to the first order results presented in Table 3. Decreasing performance with higher order results also occurs in the Spectrum and Mismatch kernels. The best performing Spectrum or Mismatch models on the multiclass protein taxonomy classification problems use substrings of length 2, and performance decreases using counts of larger substrings. This trend of reduced performance under higher order correlations can be explained by observing that the number of parameters in the higher order models increase by a factor of K when ℓ is incremented by 1 but our dataset size remains constant. Thus, as the order of the model increases, the uncertainty in the HMM variant transition matrices will also likely increase.

7 Conclusions and Future Work

Our HMM variant is an extension of the standard HMM that assigns individual transition matrices to each sequence in a dataset. At least two intuitive interpretations describe the mechanisms that allow the HMM variant to capture meaningful information about a set of sequences. In addition, we describe three inference algorithms, two of which, a Baum-Welch-like algorithm and a Gibbs sampling algorithm are similar to standard methods used to infer HMM parameters. A third, the variational inference algorithm, is related to algorithms used for inference on topic models and more complex HMM extensions. We demonstrate, by comparing results on protein sequence classification using our method in conjunction with SVMs, that each of these algorithms infers transition matrices that capture useful characteristics of individual sequences. We further show, through an analysis of the transition matrices, what types of information are best captured by each the inference method. Although classification performance using our model does not outperform either highly optimized kernels or simple string kernels, our HMM variant facilitates new ways of understanding issues in sequence classification.

One basic extension to the HMM variant involves optimization where the parameter K is not specified. This optimization is possible by extending the iHMM [1] for inference on a parameterless version of our variant.

Another extension to the basic scheme would use the values of the the transition matrix priors rather than the transition matrices themselves to construct the fixed-length representation of the sequence. Although the Dirichlet prior parameters are not as intuitively meaningful as the transition matrix, they can still be interpreted as parameters of a function that stochastically generates an associated sequence.

Because our model fits within a large existing body of work on generative models, it is amenable to extensions that could increase classification performance. We are especially interested in related models that perform classification directly rather than mediated through an SVM. These model extensions are also related to sLDA [3], Discriminative LDA [24], as well as models that use only partially generative objective functions such as DiscLDA [14], MedLDA [26], and Conditional Topic Random Fields [27].

Acknowledgments

This work was supported by NSF grant III-0905117 and a presidential fellowship awarded to SB.

References

[1] M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. The infinite hidden Markov model. *Advances in Neural Information Processing Systems*, 1:577–584, 2002.

[2] M.J. Beal and University of London. *Variational algorithms for approximate Bayesian inference*. Citeseer, 2003.

[3] D.M. Blei and J. McAuliffe. Supervised topic models. *Advances in Neural Information Processing Systems*, 20:121–128, 2008.

[4] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[6] S. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[7] Z. Ghahramani and G.E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[8] Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine learning*, 29(2):245–273, 1997.

[9] A. Gruber, M. Rosen-Zvi, and Y. Weiss. Hidden topic Markov models. *Artificial Intelligence and Statistics (AISTATS)*, 2007.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[11] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2):95–114, 2000.

[12] T. Joachims. SVMLight: Support Vector Machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 1999.

[13] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Computational Systems Bioinformatics*, pages 152–160, 2004.

[14] S. Lacoste-Julien, F. Sha, and M.I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. *Advances in Neural Information Processing Systems 21 (NIPS08)*, 2008.

[15] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[16] C. Leslie, E. Eskin, W. S. Noble, and J. Weston. Mismatch string kernels for svm protein classification. *Advances in Neural Information Processing Systems*, 20(4):467–476, 2003.

[17] D.J.C. MacKay. Ensemble learning for hidden Markov models, 1997.

[18] P.J. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *Advances in Neural Information Processing Systems*, 16:1385–1392, 2004.

[19] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[20] L. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE ASSp Magazine*, 3(1 Part 1):4–16, 1986.

[21] H. Rangwala and G. Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239, 2005.

[22] Huzefa Rangwala and George Karypis. Building multiclass classifiers for remote homology detection and fold recognition. *BMC Bioinformatics*, 7:455, 2006.

[23] S.L. Scott. Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351, 2002.

[24] H. Shan, A. Banerjee, and N.C. Oza. Discriminative Mixed-membership Models. In *2009 Ninth IEEE International Conference on Data Mining*, pages 466–475. IEEE, 2009.

[25] P. Smyth. Clustering sequences with hidden Markov models. *Advances in neural information processing systems*, pages 648–654, 1997.

[26] J. Zhu, A. Ahmed, and E.P. Xing. MedLDA: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1257–1264. ACM, 2009.

[27] J. Zhu and E.P. Xing. Conditional Topic Random Fields. In *International Conference on Machine Learning (To appear)*. Citeseer, 2010.

A Derivation of the mean field variational algorithm

Parameter	Description
N	the number of sequences
T_n	the length of sequence n
K	the number of hidden symbols
M	the number of observed symbols
a_i	start state probabilities, where i is the value of the first hidden state
A_{nij}	transition probabilities, where n indicates the sequence, i the originating hidden state, and j the destination hidden state
B_{im}	emission probabilities, where i indicates the hidden state, and m the observed symbol associated with the hidden state
z_{nt}	the hidden state at position t in sequence n
x_{nt}	the observed state at position t in sequence n
γ	Dirichlet prior parameter for a
α	Dirichlet prior parameter for A
β	Dirichelt prior parameter for B
h_{nti}	Variational parameter that approximates the mean of z_{nti}
$\tilde{\gamma}$	Variational parameter that approximates the Dirichlet prior for a
$\tilde{\alpha}_{ni}$	Variational parameter that approximates the Dirichlet prior for A_{ni}
$\tilde{\beta}_i$	Variational parameter that approximates the Dirichlet prior for B_i

Figure 6: Parameters used in the mean field variational algorithm

A.1 Probabilities

$$
\begin{aligned}
p(x,z,a,A,B|\gamma,\alpha,\beta) &= \left(\frac{\Gamma(\sum_i \gamma_i)}{\prod_i \Gamma(\gamma_i)} \prod_i a_i^{\gamma_i - 1}\right) \left(\prod_{ni} \frac{\Gamma(\sum_j \alpha_{nij})}{\prod_j \Gamma(\alpha_{nij})} \prod_j A_{nij}^{\alpha_{nij}-1}\right) \\
&\quad \left(\prod_i \frac{\Gamma(\sum_m \beta_{im})}{\prod_m \Gamma(\beta)_{im}} \prod_m B_{im}^{\beta_{im}-1}\right) \prod_n a_{z_1} \prod_{t=2}^{T_n} A_{nz_{t-1}z_t} \prod_t B_{z_t x_t} \\
&= \left(\frac{\Gamma(\sum_i \gamma_i)}{\prod_i \Gamma(\gamma_i)} \prod_i a_i^{\gamma - 1}\right) \left(\prod_{ni} \frac{\Gamma(\sum_j \alpha_{nij})}{\prod_j \Gamma(\alpha_{nij})} \prod_j A_{nij}^{\alpha-1}\right) \\
&\quad \left(\prod_i \frac{\Gamma(\sum_m \beta_{im})}{\prod_m \Gamma(\beta)_{im}} \prod_m B_{im}^{\beta-1}\right) \prod_i a_i^{n_i} \prod_{nij} A_{nij}^{n_{nij}} \prod_{im} B_{im}^{n_{im}} \\
&= \left(\frac{\Gamma(\sum_i \gamma_i)}{\prod_i \Gamma(\gamma_i)} \prod_i a_i^{\gamma_i - 1 + n_i}\right) \left(\prod_{ni} \frac{\Gamma(\sum_j \alpha_{nij})}{\prod_j \Gamma(\alpha_{nij})} \prod_j A_{nij}^{\alpha_{nij}-1+n_{nij}}\right) \\
&\quad \left(\prod_i \frac{\Gamma(\sum_m \beta_{im})}{\prod_m \Gamma(\beta)_{im}} \prod_m B_{im}^{\beta_{im}-1+n_{im}}\right)
\end{aligned}
$$

A.2 Mean Field Variational Approximation

$$
\begin{aligned}
q(z,a,A,B) =& q(a) \prod_{n=1}^{N} \prod_{i=1}^{K} q(A_{ni}) \prod_{i=1}^{K} q(B_i) \prod_{nt} q(z_{nt}) \\
=& \left(\frac{\Gamma(\sum_i \tilde{\gamma}_i)}{\prod_i \Gamma(\tilde{\gamma}_i)} \prod_i a_i^{\tilde{\gamma}_i - 1}\right) \left(\prod_{ni} \frac{\Gamma(\sum_j \tilde{\alpha}_{nij})}{\prod_j \Gamma(\tilde{\alpha}_{nij})} \prod_j A_{nij}^{\tilde{\alpha}_{nij}-1}\right) \left(\prod_i \frac{\Gamma(\sum_m \tilde{\beta}_{im})}{\prod_m \Gamma(\tilde{\beta}_{im})} \prod_m B_{im}^{\tilde{\beta}_{im}-1}\right) \prod_{nti} h_{nti}^{z_{nti}}
\end{aligned}
$$

Using the standard variational formulation [2], we construct $\mathcal{F}(q)$ by applying Jensen's inequality to create a lower bound on the marginal likelihood:

$$\begin{aligned}\mathcal{F}(q) =& \int da \int dA \int dB \sum_{\vec{z}} q(z,a,A,B) \log \frac{p(x,z,a,A,B|\alpha,\beta)}{q(z,a,A,B)} \\ =& E_q\left[\log p(x,z,a,A,B|\alpha,\beta)\right] - E_q\left[\log q(z,a,A,B)\right]\end{aligned}$$

$$\begin{aligned}\mathcal{F}(q) =& \log\Gamma(\sum_i \gamma_i) - \sum_i \log\Gamma(\gamma_i) + \sum_i (\gamma_i - 1) E_q\left[\log a_i\right] \\ & \sum_{ni}\left(\log\Gamma(\sum_j \alpha_{nij}) - \sum_j \log\Gamma(\alpha_{nij})\right) + \sum_{nij}(\alpha_{nij}-1)E_q\left[\log A_{nij}\right] \\ & \sum_i\left(\log\Gamma(\sum_m \beta_{im}) - \sum_m \log\Gamma(\beta_{im})\right) + \sum_{im}(\beta_{im}-1)E_q\left[\log B_{im}\right] \\ & \sum_i E_q\left[n_i \log a_i\right] + \sum_{nij} E_q\left[n_{nij}\log A_{nij}\right] + \sum_{im} E_q\left[n_{im}\log B_{im}\right] \\ & - \log\Gamma(\sum_i \tilde{\gamma}_i) + \sum_i \log\Gamma(\tilde{\gamma}_i) - \sum_i (\tilde{\gamma}_i - 1) E_q\left[\log a_i\right] \\ & - \sum_{ni}\log\Gamma(\sum_j \tilde{\alpha}_{nij}) + \sum_{nij}\log\Gamma(\tilde{\alpha}_{nij}) - \sum_{nij}(\tilde{\alpha}_{nij}-1)E_q\left[\log A_{nij}\right] \\ & - \sum_i \log\Gamma(\sum_m \tilde{\beta}_{im}) + \sum_{im}\log\Gamma(\tilde{\beta}_{im}) - \sum_{im}(\tilde{\beta}_{im}-1)E_q\left[\log B_{im}\right] \\ & - \sum_{nti} E_q\left[z_{nti}\right]\log h_{nti}\end{aligned}$$

A.3 Expectations The expectations of $\log a$, $\log A$, and $\log B$ are given by the formula for expectations of the log of the parameters under the Dirichlet distribution [4].

$$E_q\left[\log a_i\right] = \Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'} \tilde{\gamma}_{i'})$$

$$E_q\left[\log A_{nij}\right] = \Psi(\tilde{\alpha}_{nij}) - \Psi(\sum_{j'} \tilde{\alpha}_{nij'})$$

$$E_q\left[\log B_{im}\right] = \Psi(\tilde{\beta}_{im}) - \Psi(\sum_{m'} \tilde{\beta}_{im'})$$

$$E_q\left[n_i\right] = \sum_n h_{n1i}$$

$$
\begin{aligned}
E_q\left[n_{nij}\right] &= \sum_Z n_{nij} \prod_{nti'} h_{nti'}^{z_{nti'}} \\
&= \sum_Z \left(\sum_{t=2}^{T_n} I(z_{nt-1i}) I(z_{ntj}) \right) \prod_{nti'} h_{nti'}^{z_{nti'}} \\
&= \sum_Z \left(\sum_{t=2}^{T_n} I(z_{nt-1i}) I(z_{ntj}) \prod_{nti'} h_{nti'}^{z_{nti'}} \right) \\
&= \sum_{t=2}^{T_n} \left(\sum_Z I(z_{nt-1i}) I(z_{ntj}) \prod_{nti'} h_{nti'}^{z_{nti'}} \right) \\
&= \sum_{t=2}^{T_n} \left(h_{nt-1i} h_{ntj} \sum_{Z^{\neg z_{nt-1} z_{nt}}} \prod_{nti'} h_{nti'}^{z_{nti'}} \right) \\
&= \sum_{t=2}^{T_n} h_{nt-1i} h_{ntj}
\end{aligned}
$$

$$
E_q\left[n_{im}\right] = \sum_{nt:x_{nt}=m} h_{nti}
$$

$$
E_q\left[z_{nti}\right] = h_{nti}
$$

A.4 Maximize $F(q)$ with respect to $\tilde{\gamma}_i$ Here we use a Dirichlet prior on a with uniform parameters γ.

$$
\begin{aligned}
\mathcal{L}(\tilde{\gamma}_i) =& (\gamma - 1) \sum_i \left(\Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'} \tilde{\gamma}_{i'}) \right) + \sum_i \sum_n h_{n1i} \left(\Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'} \tilde{\gamma}_{i'}) \right) \\
& - \log \Gamma(\sum_i \tilde{\gamma}_i) + \sum_i \log \Gamma(\tilde{\gamma}_i) - \sum_i (\tilde{\gamma}_i - 1) \left(\Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'} \tilde{\gamma}_{i'}) \right) \\
=& \sum_i \left(\Psi(\tilde{\gamma}_i) - \Psi(\sum_{i'} \tilde{\gamma}_{i'}) \right) \left(\sum_n h_{n1i} + \gamma - \tilde{\gamma}_i \right) \\
& - \log \Gamma(\sum_i \tilde{\gamma}_i) + \sum_i \log \Gamma(\tilde{\gamma}_i)
\end{aligned}
$$

$$
\frac{\partial \mathcal{L}(\tilde{\gamma}_i)}{\partial \tilde{\gamma}_i} = \sum_i \Psi'(\sum_{i'} \tilde{\gamma}_{i'}) \left(\sum_n h_{n1i} + \gamma - \tilde{\gamma}_i \right) - \Psi'(\tilde{\gamma}_i) \left(\sum_n h_{n1i} + \gamma - \tilde{\gamma}_i \right)
$$

Setting the partial derivative to 0, we find the update that maximizes $\tilde{\gamma}_i$ below:

$$
\tilde{\gamma}_i = \sum_n h_{n1i} + \gamma
$$

A.4.1 Maximize $F(q)$ with respect to $\tilde{\alpha}_{nij}$ Here we use a Dirichlet prior on A with uniform parameters α.

$$\begin{aligned}\mathcal{L}(\tilde{\alpha}_{nij}) =&(\alpha-1)\sum_{nij}\left(\Psi(\tilde{\alpha}_{nij})-\Psi(\sum_{j'}\tilde{\alpha}_{nij'})\right)+\sum_{nij}\sum_{t}h_{nt-1i}h_{ntj}\left(\Psi(\tilde{\alpha}_{nij})-\Psi(\sum_{j'}\tilde{\alpha}_{nij'})\right)\\&-\sum_{ni}\log\Gamma(\sum_{j}\tilde{\alpha}_{nij})+\sum_{nij}\log\Gamma(\tilde{\alpha}_{nij})-\sum_{nij}(\tilde{\alpha}_{nij}-1)\left(\Psi(\tilde{\alpha}_{nij})-\Psi(\sum_{j'}\tilde{\alpha}_{nij'})\right)\\=&\sum_{nij}\left(\sum_{t}h_{nt-1i}h_{ntj}+\alpha-\tilde{\alpha}_{nij}\right)\left(\Psi(\tilde{\alpha}_{nij})-\Psi(\sum_{j'}\tilde{\alpha}_{nij'})\right)-\sum_{ni}\log\Gamma(\sum_{j}\tilde{\alpha}_{nij})+\sum_{nij}\log\Gamma(\tilde{\alpha}_{nij})\end{aligned}$$

$$\frac{\partial\mathcal{L}(\tilde{\alpha}_{nij})}{\partial\tilde{\alpha}_{nij}}=\Psi'(\tilde{\alpha}_{nij})\left(\sum_{t}h_{nt-1i}h_{ntj}+\alpha-\tilde{\alpha}_{nij}\right)-\sum_{j}\Psi'(\sum_{j'}\tilde{\alpha}_{nij'})\left(\sum_{t}h_{nt-1i}h_{ntj}+\alpha-\tilde{\alpha}_{nij}\right)$$

Setting the partial derivative to 0, we find the update that maximizes $\tilde{\alpha}_{nij}$ below:

$$\tilde{\alpha}_{nij}=\sum_{t}h_{nt-1i}h_{ntj}+\alpha$$

A.4.2 Maximize $F(q)$ with respect to $\tilde{\beta}_{im}$ Here we use a Dirichlet prior on B with uniform parameters β

$$\begin{aligned}\mathcal{L}(\tilde{\beta}_{im}) =&(\beta-1)\sum_{im}\left(\Psi(\tilde{\beta}_{im})-\Psi(\sum_{m'}\tilde{\beta}_{im'})\right)+\sum_{im}\left(\sum_{nt:x_t=m}h_{nti}\right)\left(\Psi(\tilde{\beta}_{im})-\Psi(\sum_{m'}\tilde{\beta}_{im'})\right)\\&-\sum_{i}\log\Gamma(\sum_{m}\tilde{\beta}_{im})+\sum_{im}\log\Gamma(\tilde{\beta}_{im})-\sum_{im}(\tilde{\beta}_{im}-1)\left(\Psi(\tilde{\beta}_{im})-\Psi(\sum_{m'}\tilde{\beta}_{im'})\right)\\=&\sum_{im}\left(\sum_{nt:x_t=m}h_{tni}+\beta-\tilde{\beta}_{im}\right)\left(\Psi(\tilde{\beta}_{im})-\Psi(\sum_{m'}\tilde{\beta}_{im'})\right)-\sum_{i}\log\Gamma(\sum_{m}\tilde{\beta}_{im})+\sum_{im}\log\Gamma(\tilde{\beta}_{im})\end{aligned}$$

Setting the partial derivative to 0, we find the update that maximizes $\tilde{\beta}_{im}$ below:

$$\tilde{\beta}_{im}=\sum_{nt:x_t=m}h_{nti}+\beta$$

A.4.3 Maximize $F(q)$ with respect to h_{nti}

$$\begin{aligned}\mathcal{L}(h_{nti}) =&\sum_{i}\left(\sum_{n}h_{n1i}\right)\left(\Psi(\tilde{\gamma}_{i})-\Psi(\sum_{i'}\tilde{\gamma}_{i'})\right)+\\&\sum_{nij}\left(\sum_{t=2}^{T_n}h_{nt-1i}h_{ntj}\right)\left(\Psi(\tilde{\alpha}_{nij})-\Psi(\sum_{j'}\tilde{\alpha}_{nij'})\right)+\\&\sum_{im}\left(\sum_{t=1}^{T_n}\mathbb{I}(x_t=m)h_{nti}\right)\left(\Psi(\tilde{\beta}_{im})-\Psi(\sum_{m'}\tilde{\beta}_{im'})\right)\\&-\sum_{nti}h_{nti}\log h_{nti}-\lambda\left(\sum_{i}h_{nti}-1\right)\end{aligned}$$

$$\frac{\partial \mathcal{L}(h_{nti})}{\partial h_{n't'i'}} = \begin{cases} (\Psi(\tilde{\gamma}_{i'}) - \Psi(\sum_{i''} \tilde{\gamma}_{i''})) + & \\ \sum_j h_{n'2j} \left(\Psi(\tilde{\alpha}_{n'i'j}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'i'j'}) \right) + \left(\Psi(\tilde{\beta}_{ix_{n'1}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) - & \\ 1 - \log h_{n't'i'} - \lambda & t' = 1 \\ \sum_i h_{n't'-1i} \left(\Psi(\tilde{\alpha}_{n'ii'}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'ij'}) \right) + & \\ \sum_j h_{n't'+1j} \left(\Psi(\tilde{\alpha}_{n'i'j}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'i'j'}) \right) + \left(\Psi(\tilde{\beta}_{ix_{n't'}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) - & \\ 1 - \log h_{n't'i'} - \lambda & 1 < t' < T' \\ \sum_i h_{n'T_{n'}-1i} \left(\Psi(\tilde{\alpha}_{n'ii'}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'ij'}) \right) + \left(\Psi(\tilde{\beta}_{ix_{n'T_{n'}-1}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) - & \\ 1 - \log h_{n't'i'} - \lambda & t' = T' \end{cases}$$

Setting $\frac{\partial \mathcal{L}(h_{nti})}{\partial h_{n't'i'}}$ to zero, we find the expression for h_{nti} below:

$$h_{n't'i'} \propto \begin{cases} \exp\left(\Psi(\tilde{\gamma}_{i'}) - \Psi(\sum_{i''} \tilde{\gamma}_{i''})\right) & \\ \quad + \sum_j h_{n'2j} \left(\Psi(\tilde{\alpha}_{n'i'j}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'i'j'}) \right) & \\ \quad + \left(\Psi(\tilde{\beta}_{ix_{n'1}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) & t' = 1 \\ \exp \sum_i h_{n't'-1i} \left(\Psi(\tilde{\alpha}_{n'ii'}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'ij'}) \right) & \\ \quad + \sum_j h_{n't'+1j} \left(\Psi(\tilde{\alpha}_{n'i'j}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'i'j'}) \right) & \\ \quad + \left(\Psi(\tilde{\beta}_{ix_{n't'}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) & 1 < t' < T'_{n'} \\ \exp \sum_i h_{n't'-1i} \left(\Psi(\tilde{\alpha}_{n'ii'}) - \Psi(\sum_{j'} \tilde{\alpha}_{n'ij'}) \right) & \\ \quad + \left(\Psi(\tilde{\beta}_{ix_{n'T_{n'}-1}}) - \Psi(\sum_{m'} \tilde{\beta}_{im'}) \right) & t' = T'_{n'} \end{cases}$$