A Hidden Markov Model Variant for Sequence Classification[*]

Sam Blasiak and **Huzefa Rangwala**
Computer Science, George Mason University
sblasiak@gmu.edu, rangwala@cs.gmu.edu

Abstract

Sequence classification is central to many practical problems within machine learning. Distances metrics between arbitrary pairs of sequences can be hard to define because sequences can vary in length and the information contained in the order of sequence elements is lost when standard metrics such as Euclidean distance are applied. We present a scheme that employs a Hidden Markov Model variant to produce a set of fixed-length description 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 sequences of amino acids into structural classes.

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 [Leslie *et al.*, 2002]. A third method for classifying sequence data requires only a positive definite mapping defined over pairs of 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 (SVMs) and allows for a wide variety of sequence similarity measurements to be employed.

Hidden Markov Models (HMM) [Rabiner and Juang, 1986; Eddy, 1998] have a rich history in sequence data modeling (in speech recognition and bioinformatics 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 constrained 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 [Rabiner and Juang, 1986].

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. The HMM variant is also similar to a number of topic models, and we describe it in the context of Latent Dirichlet Allocation [Blei *et al.*, 2003]. 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 sequences of amino acids into structural classes (fold recognition and remote homology detection problem [Rangwala and Karypis, 2006]). 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.

2 Problem Statement

Given a set of N sequences, we would like to find a set of fixed-length vectors, $A_{1...N}$, that, when used as input to a function $f(A)$, maximize the probability of reconstructing the original set of sequences. Under our scheme,

[*]Funding: NSF III-0905117.

$f(A)$ is a Hidden Markov Model variant with one transition matrix, A_n, assigned to each sequence, and a single emissions matrix, B, and 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.

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 indexed by the value of the first hidden state
A_{nij}	transition probabilities, where n is an index of a training 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 symbol at position t in sequence n

Table 1: HMM Variant model parameters

The joint probability of the model is shown below:

(1) $p(x, z|a, A, B) =$

$$\prod_{n=1}^{N}\left(a_{z_{n1}}\left(\prod_{t=2}^{T_n}\mathbf{A_{n\,z_{nt-1}\,z_{nt}}}\right)\left(\prod_{t=1}^{T_n}B_{z_{nt}\,x_{nt}}\right)\right)$$

This differs from the standard hidden Markov model only in the addition of a transition matrix, A_n (highlighted in bold in Equation 1), for each sequence, where the index n indicates a sequence in the training set. Under the standard HMM, a single transition matrix, A, would be used for all sequences.

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 $\mathrm{Dir}(x; a, K) = \frac{\Gamma(Ka)}{\Gamma(a)^K}\prod_i x_i^{a-1}$:

(2) $p(x, z, a, A, B|\alpha, \beta, \gamma) =$

$$\left(\frac{\Gamma(K\gamma)}{\Gamma(\gamma)^K}\prod_i a_i^{\gamma-1}\right)\left(\prod_{ni}\frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K}\prod_j A_{nij}^{\alpha-1}\right)\left(\prod_i\frac{\Gamma(M\beta)}{\Gamma(\beta)^M}\prod_m B_{im}^{\beta-1}\right)$$

$$\prod_{n=1}^{N}\left(a_{z_{n1}}\left(\prod_{t=2}^{T_n}A_{nz_{nt-1}z_{nt}}\right)\left(\prod_{t=1}^{T_n}B_{z_{nt}x_{nt}}\right)\right)$$

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:

$$\text{HMM}_1\text{: } A_1 = \begin{bmatrix} .9 & .1 \\ .9 & .1 \end{bmatrix},\ B_1 = \begin{bmatrix} .9 & .1 \\ .1 & .9 \end{bmatrix}$$

$$\text{HMM}_2\text{: } A_2 = \begin{bmatrix} .1 & .9 \\ .1 & .9 \end{bmatrix},\ B_2 = \begin{bmatrix} .1 & .9 \\ .9 & .1 \end{bmatrix}$$

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

3 Background

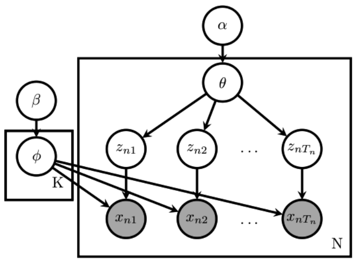
3.1 Mixtures of HMMs

Smyth introduces a mixture of HMMs in [Smyth, 1997] 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 [Smyth, 1997], 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.

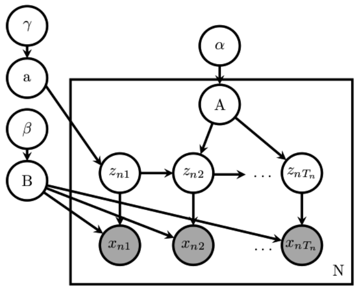
3.2 Topic Models

Simpler precursors of LDA [Blei *et al.*, 2003] and pLSI [Hofmann, 1999], 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, as in our model.

Inference of the LDA model (Figure 1a) on a corpus of documents learns a matrix of document-topic proba-



(a)



(b)

Figure 1: Plate diagrams of the (a) LDA model, expanded to show each word separately and the (b) HMM variant. The model parameters in the LDA model are defined as follows: K - number of topics, ϕ_k - a vector of word probabilities given topic k, β - parameters of the Dirichlet prior of ϕ_k, θ_n - a vector of topic probabilities in document n, α - parameters of the Dirichlet prior of θ_n. A row of the matrix B in the HMM variant has exactly the same meaning as a topic-word vector, ϕ_k, in the LDA model.

bilities. 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 1b), 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. The LDA model also includes a "topic-word" matrix, which indicates the probability of a word given a topic assignment. This matrix has the same meaning as the emissions matrix, B, in the HMM variant.

The Fisher kernel [Jaakkola and Haussler, 1999] and the Probabilistic Product Kernel [Jebara *et al.*, 2004] (PPK), are principled methods that allow probabilistic models to be incorporated into SVM kernels. The HMM variant is similar to these methods in that it uses latent information from a generative model as input to a discriminative classifier. It differs from these methods, however, both in which portions of the generative model that are incorporated into the discriminative classifier and in the assumptions about how differences in generating distributions comparisons between training examples.

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 posteriori (MAP) estimate of $p(z, a, A, B|x, \gamma, \alpha, \beta)$ [Rabiner and Juang, 1986]:

$$a_i \propto \sum_n f_{ni}(1)b_{ni}(1) + \gamma - 1 \tag{3}$$

$$A_{nij}^{(new)} \propto \sum_{t=2}^{T_n} f_{ni}(t-1)A_{nij}B_{jx_t}b_{nj}(t) + \alpha - 1 \tag{4}$$

$$B_{im}^{(new)} \propto \sum_n \sum_{t:x_t=m} f_{ni}(t)b_{nj}(t) + \beta - 1 \tag{5}$$

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

$$f_{ni}(t) = \begin{cases} \sum_j f_{nj}(t-1)A_{nji}B_{ix_t}, & t > 1 \\ a_i B_{ix_1}, & t = 1 \end{cases} \tag{6}$$

$$b_{ni}(t) = \begin{cases} \sum_j A_{nij}B_{jx_{t+1}}b_{nj}(t+1), & t < T_n \\ \frac{1}{K}, & t = T_n \end{cases} \tag{7}$$

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, $\sum_n T_n$, the same number as the original algorithm.

4.2 Gibbs Sampling

Two Gibbs sampling schemes are commonly used to infer Hidden Markov Model parameters [Scott, 2002]. 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:

$$p(z_{ti}^{(new)}|z_{t-1}, z_{t+1}) \propto A_{z_{t-1}i}B_{ix_t}A_{iz_{t+1}} \tag{8}$$

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

$$p(z_{nT_n}^{(new)}|x_{n1:T_n}) \propto f_{ni}(T_n) \tag{9}$$

$$\begin{aligned} p(z_{nt}^{(new)}|x_{n1:T_n}, z_{nt+1}^{(new)}) &= p(z_{nt}^{(new)}|x_{n1:t}, z_{nt+1}^{(new)}) \\ &\propto f_{ni}(t)A_{nij}, \qquad t < T_n \end{aligned} \tag{10}$$

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:

$$p(A_{nij}|z_n, \alpha) = \text{Dir}(\sum_{t=2}^{T_n} \mathbb{I}(z_{nt-1} = i)\mathbb{I}(z_{nt} = j) + \alpha) \tag{11}$$

The FB sampler has been shown to mix more quickly than the DG sampler, especially in cases where adjacent hidden states are highly correlated [Scott, 2002]. 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:

$$q(z,a,A,B) = q(a)\prod_{n=1}^{N}\prod_{i=1}^{K} q(A_{ni})\prod_{i=1}^{K} q(B_i)\prod_{nt} q(z_{nt}) \tag{12}$$

$$= \left(\frac{\Gamma(\sum_i \tilde{\gamma_i})}{\prod_i \Gamma(\tilde{\gamma_i})}\prod_i a_i^{\tilde{\gamma_i}-1}\right)\left(\prod_{ni}\frac{\Gamma(\sum_j \tilde{\alpha}_{nij})}{\prod_j \Gamma(\tilde{\alpha}_{nij})}\prod_j A_{nij}^{\tilde{\alpha}_{nij}-1}\right)$$

$$\left(\prod_i \frac{\Gamma(\sum_m \tilde{\beta}_{im})}{\prod_m \Gamma(\tilde{\beta}_{im})}\prod_m B_{im}^{\tilde{\beta}_{im}-1}\right)\prod_{nti} h_{nti}^{z_{nti}}$$

with variational parameters h_{nti}, which approximate 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 $\Psi(x) = \frac{d \log \Gamma(x)}{dx}$:

$$\tilde{\alpha}_{nij} = \sum_t h_{nt-1i} h_{ntj} + \alpha \tag{13}$$

$$\tilde{\beta}_{im} = \sum_{nt:x_t=m} h_{nti} + \beta \tag{14}$$

$$\tilde{\gamma_i} = \sum_n h_{n1i} + \gamma \tag{15}$$

$$h_{nti} \propto \exp\left(\sum_{i'} h_{nt-1i'}\left(\Psi(\tilde{\alpha}_{ni'i}) - \Psi(\sum_j \tilde{\alpha}_{ni'j})\right) + \right. \tag{16}$$

$$\left. \sum_j h_{n\,t+1\,j}\left(\Psi(\tilde{\alpha}_{nij}) - \Psi(\sum_{j'} \tilde{\alpha}_{nij'})\right) + \left(\Psi(\tilde{\beta}_{ix_{nt}}) - \Psi(\sum_m \tilde{\beta}_{im})\right)\right),$$

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 parameters rather than through the sampled values of z. A complete derivation of the variational algorithm is included on the authors' website[1].

[1]http://www.cs.gmu.edu/~mlbio/ijcai11

7 Class 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

25 Fold 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

27 Fold 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

37 Superfamily 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 2: 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 best AUC score on the majority of the datasets.

5 Experimental Setup

5.1 Protocol

To evaluate our fixed-length representation scheme, for each dataset (described in Section 5.2), 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, K, from 5 to 20 in increments of 5. 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 [Crammer and Singer, 2002] construction or the one-versus-rest approach, where a binary classifier was trained for each of the classes.

We compare classification results from our model with results from the Spectrum(2) kernel for all experiments. The Spectrum(ℓ) kernel is a string kernel whose vector representation is the set of counts of substrings of observed symbols length ℓ in a given string [Leslie *et al.*, 2002]. For the one-versus rest experiments, we compare our results to more biologically sensitive kernels for protein classification, described in Rangwala et. al [Rangwala and Karypis, 2005].

5.2 Protein Datasets

The Structural Classification of Proteins (SCOP) [Murzin *et al.*, 1995] database categorizes proteins into a multilevel hierarchy that captures commonalities between protein structure at different levels of detail. To evaluate our representation, we ran sets of protein classi-

[2]We used SVM-light and SVM-struct for classification (http://www.cs.cornell.edu/People/tj/svm_light/) [Joachims, 1999].

fication experiments on the three top levels of the SCOP taxonomy: class, fold, and superfamily. Our datasets, which were obtained from previous studies [Rangwala and Karypis, 2006; Kuang *et al.*, 2004], 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 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 [Leslie *et al.*, 2002; Rangwala and Karypis, 2005]), no examples that shared the superfamily and family levels were included in both the training and test sets.

5.3 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-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 2 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.

Table 3 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. Similar 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 best performing algorithm's AUC and AUC50 scores do not significantly change from the linear to the Gaussian kernel, the variational algorithm shows a large improvement, ranging from 6% to 30%.

Linear Kernel

Metric	AUC				AUC50			
Alg/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

Metric	AUC				AUC50			
Alg/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 3: 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.

6.2 Analysis of inference algorithms

The differences in AUC values resulting from the different training algorithms (Tables 2 and 3) can be explained, at least in part, by a high level overview of how each algorithm operates. While the Baum-Welch algorithm returns MAP parameters of the model, both the Gibbs sampling method and the variational algorithm return expectations of the parameters under an approximate of the posterior distribution. The MAP solution from the Baum-Welch algorithm is likely to reach a local maximum of the posterior, while the other algorithms should tend to average over posterior parameters.

The Gibbs sampling algorithm and the variational algorithm each 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.

6.3 Comparative Performance

Tables 4 and 5 show a comparison between the HMM variant and common classification methods for the 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. In defense of the HMM variant, the size of the vector representation produced by the spectrum kernel 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 5), is similar to the Spectrum(5) kernel but also counts substrings of 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. Nearly

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 4: 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 2.

Algorithm	AUC	AUC50
HMM Variant (best)	0.67	0.37
Spectrum(2) [Leslie *et al.*, 2002]	0.712	0.290
Mismatch(5,1) [Leslie *et al.*, 2003]	0.870	0.416
Fisher [Jaakkola *et al.*, 2000]	0.773	0.250
SW-PSSM [Rangwala and Karypis, 2005]	0.982	0.904

Table 5: A selection of AUC and AUC50 scores for the Remote Homology Detection problem using a variety of SVM kernels on the SCOP 1.53, 25% dataset using 1-vs-rest classification. The HMM variant scores are the best performing from Table 3.

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, the only parameter that needs to be adjusted in the HMM variant is the number of hidden states.

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 but keeps a single emissions matrix for the entire dataset. 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. Because our model fits within a large existing body of work on generative models, we are especially interested in related models that perform classification directly.

References

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

[Crammer and Singer, 2002] 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.

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

[Hofmann, 1999] 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.

[Jaakkola and Haussler, 1999] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[Jaakkola *et al.*, 2000] 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.

[Jebara *et al.*, 2004] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

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

[Kuang *et al.*, 2004] 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.

[Leslie *et al.*, 2002] 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.

[Leslie *et al.*, 2003] 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.

[Murzin *et al.*, 1995] 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.

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

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

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

[Scott, 2002] 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.

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