Abstraction Augmented Markov Models

Cornelia Caragea[1] **Adrian Silvescu**[2] **Doina Caragea**[3] **Vasant Honavar**[1]
[1]Computer Science Department, Iowa State University
[2]Yahoo! Labs, Sunnyvale, CA
[3] Computer and Information Sciences, Kansas State University
{cornelia,honavar}@cs.iastate.edu, silvescu@yahoo-inc.com, dcaragea@ksu.edu

Abstract

We present abstraction augmented Markov models (AAMMs), which are directed acyclic graphical models that simplify the data representation used by the standard Markov models (MMs). AAMMs group similar entities to generate more abstract entities that are organized in an abstraction hierarchy. *Abstraction* reduces the MM size and improves the statistical estimates of complex models by reducing the number of parameters to be estimated from data. We evaluate the AAMMs on two protein subcellular localization prediction tasks. The results of our experiments show that: (1) AAMMs can achieve significantly lower model sizes (by 1 to 3 orders of magnitude) for a minor drop in accuracy over the standard MMs, and in some cases even higher accuracy while simultaneously lowering the model size; and (2) AAMMs substantially outperforms MMs in settings where only a small fraction of available data is labeled.

1 Introduction

Many real-world problems can be regarded as sequence classification tasks. For example, in computational biology predicting protein function or protein subcellular localization can be addressed as sequence classification problems, where the amino acid sequence of the protein is used to classify a protein in functional or localization classes. Markov models (MMs) are effective sequence models [3] that capture dependencies between neighboring elements in a sequence and thus provide more accurate models of the data. In fixed-order MMs the elements of a sequence satisfy the *Markov property*: Each element in the sequence directly depends on a fixed number of previous elements, called *parents*, and is independent of the rest of the elements in the sequence. Interpolated MMs [5] combine several fixed-order MMs that capture important sequence patterns that would otherwise be ignored by a single fixed-order MM.

While dependencies between neighboring elements provide a way to improve the predictive accuracy, the number of model parameters increases exponentially with the range of direct dependencies, thereby increasing the risk of overfitting when the data set is limited in size. We present abstraction augmented Markov models (AAMMs) aimed at addressing this dificulty by reducing the number of model parameters through *abstraction*. AAMMs construct an abstraction hierarchy over the values of the *parents* of each element using hierarchical agglomerative clustering [2]. AAMMs extend the standard MMs by constructing new variables that represent abstractions over the values of the *parents* of each element.

We evaluate AAMMs on two protein subcellular localization prediction tasks. The results of our experiments show that adapting data representation by abstraction makes it possible to construct predictive models that use substantially smaller number of parameters (by 1 to 3 orders of magnitude) than standard MMs without sacrificing predictive accuracy. Our results also show that AAMMs substantially outperform MMs in semi-supervised settings where there exist only a small fraction of labeled data, but a vast amount of unlabeled data.

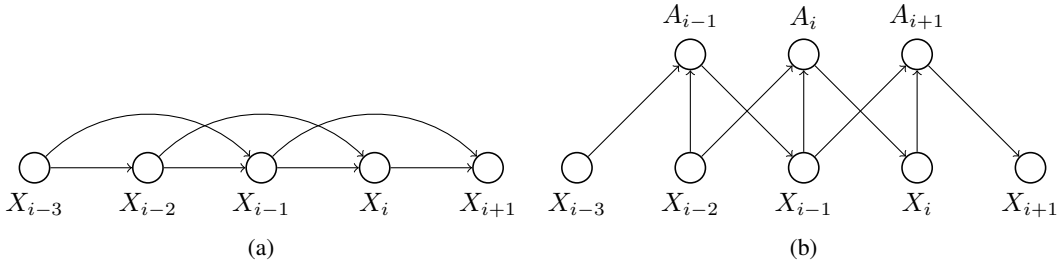


Figure 1: (a) 2^{nd} Order Markov Model; (b) 2^{nd} Order Abstraction Augmented Markov Model

2 From Markov Models To Abstraction Augmented Markov Models

2.1 Markov Models

Let $\mathbf{x} = x_0, \cdots, x_{n-1}$ be an input sequence over an alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$. An MM represents the joint probability distribution of $\mathbf{x}$ under the Markov assumption. The graphical representation of an MM is a directed linear-chain graph where the nodes X_i represent random variables corresponding to the sequence elements x_i, $i = 0, \cdots, n-1$, and the edges represent direct dependencies between neighboring elements. For a k^{th} order MM, the conditional independencies are:

$$X_i \perp\!\!\!\perp \{X_0, \cdots, X_{i-k-1}\} \,|\, \{X_{i-k}, \cdots, X_{i-1}\} \text{ for } i = k, \cdots, n-1$$

That is, X_i is conditionally independent of $X_0, \cdots, X_{i-k-1}$ given $X_{i-k}, \cdots, X_{i-1}$ for any $i = k, \cdots, n-1$. $X_{i-k}, \cdots, X_{i-1}$ are called the *parents* of X_i. The joint probability distribution $p(\mathbf{X})$, where $\mathbf{X}$ denotes the set of nodes, can be factorized as follows:

$$p(\mathbf{X}) = p(X_0, \cdots, X_{k-1}) \prod_{i=k}^{n-1} p(X_i | X_{i-k}, \cdots, X_{i-1})$$

The directed graph for a 2^{nd} order Markov model on a subset of nodes of sequence $\mathbf{x}$, $\{X_{i-3}, \cdots, X_{i+1}\}$, is shown in Figure 1a.

2.2 Abstraction Augmented Markov Models

Abstraction is the operation of grouping similar entities to generate more abstract entities. Let S_{i-1} denote the *parents* $X_{i-k} \cdots X_{i-1}$ of X_i in a standard k^{th} order MM. The values of S_{i-1} represent instantiations of $X_{i-k} \cdots X_{i-1}$, i.e., substrings of length k over $\mathcal{X}$, or k-grams. Let $\mathcal{S}$ denote the set of k-grams over $\mathcal{X}$. Hence, the cardinality of $\mathcal{S}$ is $N = |\mathcal{X}|^k$. To reduce the cardinality of $\mathcal{S}$, AAMMs construct an abstraction hierarchy over the set of k-grams.

Definition 1 (Abstraction Hierarchy) *An abstraction hierarchy $\mathcal{T}$ associated with a set of k-grams $\mathcal{S}$ is a rooted tree such that: (1) The root of $\mathcal{T}$ corresponds to the abstraction that consists of all k-grams in $\mathcal{S}$; (2) The tree $\mathcal{T}$ has exactly N leaves corresponding to the N k-grams in $\mathcal{S}$; (3) The internal nodes of $\mathcal{T}$ correspond to abstractions over k-grams (i.e., subsets of "similar" k-grams); (4) The edges of $\mathcal{T}$ correspond to partial order relations $\prec$ between their corresponding nodes.*

Definition 2 ($\mathbf{m}$-Cut) *An m-cut γ_m through the abstraction hierarchy $\mathcal{T}$ is a subset of m nodes of $\mathcal{T}$ satisfying the following properties: (1) For any leaf s_i, either $s_i \in \gamma_m$ or s_i is a descendant of a node $a_j \in \gamma_m$; and (2) for any two nodes $a_j, a_l \in \gamma_m$, a_j is neither a descendant nor an ancestor of a_l. The set of abstractions $\mathcal{A}$ at any given m-cut γ_m forms a partition of the set of k-grams.*

AAMMs extend the graphical structure of the standard MMs by constructing new variables A_i that represent abstractions over the set of values of S_{i-1} for $i = k, \cdots, n-1$. More precisely, the variables A_i take values in a set of abstractions $\mathcal{A} = \{a_1, \cdots, a_m\}$ corresponding to an m-cut γ_m. We model the fact that A_i is an abstraction of S_{i-1} by defining $p(A_i = a_i | S_{i-1} = s_{i-1}) = 1$ if $s_{i-1} \in a_i$, and 0 otherwise, where $s_{i-1} \in \mathcal{S}$ and $a_i \in \mathcal{A}$, respectively. Furthermore, in AAMMs, the node X_i directly depends on A_i instead of being directly dependent on S_{i-1}, as in the standard

Algorithm 1 Abstraction Construction

Input: A set of sequences $\mathcal{D} = \{\mathbf{x}_l\}$, $\mathbf{x}_l \in \mathcal{X}^*$; the set of k-grams $\mathcal{S} = \{s_1, \cdots, s_N\}$ in $\mathcal{D}$
Output: An abstraction hierarchy $\mathcal{T}$ over $\mathcal{S}$
Initialize $\mathcal{A} = \{a_1 : \{s_1\}, \cdots, a_N : \{s_N\}\}$, and $\mathcal{T} = \{a_1 : \{s_1\}, \cdots, a_N : \{s_N\}\}$
for $w = N + 1$ **to** $2N - 1$ **do**
  $(u_{min}, v_{min}) = \arg\min_{u,v \in \mathcal{A}} d_{\mathcal{D}}(a_u, a_v)$
  $a_w = a_{u_{min}} \cup a_{v_{min}}$
  $\mathcal{A} = \mathcal{A} \backslash \{a_{u_{min}}, a_{v_{min}}\} \cup \{a_w\}$
  $\mathcal{T} = \mathcal{T} \cup \{a_w\}$ s.t. $par(a_{u_{min}}) = a_w$, $par(a_{v_{min}}) = a_w$
end for

MMs. We define the conditional distribution of X_i given A_i as $p(X_i = x_i | A_i = a_i) = \theta_{xa}^{(i)}$, where $x_i \in \mathcal{X}$ and $a_i \in \mathcal{A}$, respectively. The prior distribution of S_{k-1} is defined as $p(S_{k-1}) = \theta_s$.

The directed graph for a 2^{nd} order AAMM on a subset of variables $\mathbf{X} \cup \mathbf{A}$ is shown in Figure 1b. The joint probability distribution over the entire set of variables can be factorized as follows:

$$p(\mathbf{X}, \mathbf{A}) = p(S_{k-1}) \cdot \prod_{i=k}^{n-1} p(X_i | A_i) \cdot p(A_i | S_{i-1})$$

In what follows we show how to learn AAMMs from data which involves two steps: learning abstraction hierarchies (AHs) and learning model parameters.

2.2.1 Learning Abstraction Hierarchies

The procedure for constructing AHs over the set of k-grams is shown in Algorithm 1. The *input* of the algorithm consists of a set $\mathcal{D}$ of sequences over $\mathcal{X}$, $\mathcal{D} = \{\mathbf{x}_l\}$, $\mathbf{x}_l \in \mathcal{X}^\star$, and the set $\mathcal{S}$ of k-grams extracted from $\mathcal{D}$. The *output* of the algorithm is a binary tree $\mathcal{T}$, specifically an AH over $\mathcal{S}$.

The algorithm starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction a_j corresponds to a k-gram s_j in $\mathcal{S}$, $j = 1, \cdots, N$. For each a_j the algorithm creates a node in $\mathcal{T}$. Pairs of abstractions are recursively merged until one abstraction is obtained. Specifically, $N - 1$ times, the algorithm searches for the most "similar" two abstractions $a_{u_{min}}$ and $a_{v_{min}}$, adds a new abstraction a_w to $\mathcal{A}$ by taking the union of their k-grams, and removes $a_{u_{min}}$ and $a_{v_{min}}$ from $\mathcal{A}$. Simultaneously, the nodes in $\mathcal{T}$ corresponding to $a_{u_{min}}$ and $a_{v_{min}}$ are merged into a newly created node which is their parent. After $N - 1$ steps, the algorithm returns the AH $\mathcal{T}$ over $\mathcal{S}$, which is stored in a last-in-first-out (LIFO) stack. For a given choice of the size m of an m-cut through $\mathcal{T}$, the set of abstractions that define an AAMM is obtained by discarding $m - 1$ elements from the stack.

To complete the description of our algorithm we need to define the similarity between abstractions. We define a distance $d_{\mathcal{D}}(a_u, a_v)$ between the *contexts* of two abstractions a_u and a_v and identify the most "similar" abstractions as those that have the smallest distance between their contexts (see below). Since an abstraction is a subset of k-grams, we define the context of such a subset by aggregating the contexts of its constituent k-grams.

Context of an abstraction. Given a set $\mathcal{D} = \{\mathbf{x}_l\}$ of sequences, we define the context of a k-gram $s_j \in \mathcal{S}$, $j = 1, \cdots, N$, wrt $\mathcal{D}$ as follows:

$$Context_{\mathcal{D}}(s_j) := [p(X_i | s_j), \#s_j] = \left[\left[\frac{\#[s_j, x_i]}{\sum_{x_i \in \mathcal{X}} \#[s_j, x_i]} \right]_{x_i \in \mathcal{X}}, \sum_{x_i \in \mathcal{X}} \#[s_j, x_i] \right]$$

That is, the context of a k-gram s_j wrt $\mathcal{D}$ is the conditional distribution of X_i given s_j, $p(X_i | s_j)$ estimated from $\mathcal{D}$, along with the frequency counts of the k-gram s_j in $\mathcal{D}$, $\#s_j$.

More generally, we define the context of an abstraction $a_j = \{s_{j_1}, \cdots, s_{j_q}\}$, $a_j \in \mathcal{A}$, as:

$$Context_{\mathcal{D}}(a_j) := \left[\sum_{r=1}^{q} \pi_r \cdot p(X_i | s_{j_r}), \sum_{r=1}^{q} \#s_{j_r} \right] \text{ where } \pi_r := \frac{\#s_{j_r}}{\sum_{r=1}^{q} \#s_{j_r}}$$

Note that if we "abstract out" the difference between all the k-grams in the abstraction a_j and replace their occurrences in $\mathcal{D}$ by a_j, then $\#a_j = \sum_{r=1}^{q} \#s_{j_r}$ and $\#[a_j, x_i] = \sum_{r=1}^{q} \#[s_{j_r}, x_i]$. Furthermore, for any $x_i \in \mathcal{V}$:

$$p(x_i|a_j) = \frac{\#[a_j, x_i]}{\#a_j} = \frac{\sum_{r=1}^{q} \#[s_{j_r}, x_i]}{\sum_{r=1}^{q} \#s_{j_r}} = \sum_{r=1}^{q} \frac{\#s_{j_r}}{\sum_{r=1}^{q} \#s_{j_r}} \frac{\#[s_{j_r}, x_i]}{\#s_{j_r}} = \sum_{r=1}^{q} \pi_r \cdot p(x_i|s_{j_r})$$

Hence, we have shown that $Context_{\mathcal{D}}(a_j) = [p(X_i|a_j), \#a_j]$. Next, we define a distance between contexts of two abstractions, motivated by ideas from information theory [1].

Distance between abstractions. Our goal is to find an abstraction (or compression) A_i of the variable S_{i-1} and, at the same time, preserve the direct dependency between A_i and X_i as much as possible. One way to measure the dependency between two variables is to use mutual information [1]. Hence, we want to construct an abstraction A_i of S_{i-1} such that the reduction in the mutual information between A_i and X_i, $I(A_i, X_i)$, is minimized at each step of Algorithm 1.

The reduction in mutual information between a node and its *parents* (in the graphical model corresponding to an AAMM) due to a single merge of Algorithm 1 can be calculated as follows: Let γ_m be an m-cut through the abstraction hierarchy $\mathcal{T}$ and γ_{m-1} be the $(m-1)$-cut through $\mathcal{T}$ that results after one merge $\{a_u, a_v\} \rightarrow a_w$. Let $\mathcal{A}_m$ and $\mathcal{A}_{m-1}$ denote the set of abstractions corresponding to γ_m and γ_{m-1}, respectively. Furthermore, let π_u and π_v denote the prior probabilities of a_u and a_v in the set a_w, i.e. $\pi_u = \frac{p(a_u)}{p(a_u)+p(a_v)}$ and $\pi_v = \frac{p(a_v)}{p(a_u)+p(a_v)}$.

Proposition 1: *The reduction in the mutual information between each variable X_i and its parent, due to the above merge is given by $\delta I(\{a_u, a_v\}, a_w) = (p(a_u) + p(a_v)) \cdot JS_{\pi_u,\pi_v}(p(X_i|a_u), p(X_i|a_v)) \geq 0$, where $JS_{\pi_u,\pi_v}(p(X_i|a_u), p(X_i|a_v))$ represents the weighted Jensen-Shannon divergence [6] between two probability distributions $p(X_i|a_u)$ and $p(X_i|a_v)$ with weights π_u and π_v, respectively. (We ignore the proof due to lack of space.)*

We define the distance between two abstractions a_u and a_v in $\mathcal{D}$, denoted by $d_{\mathcal{D}}(a_u, a_v)$, as follows:

$$d_{\mathcal{D}}(a_u, a_v) \quad = \quad \delta I(\{a_u, a_v\}, a_w) \text{ where } a_w = \{a_u \cup a_v\}$$

2.3 Learning AAMM Parameters

The AAMM is a completely observed directed graphical model, i.e. there are no hidden variables in the model. The parameters $\theta_{xa}^{(i)}$ and θ_s of an AAMM can be estimated using maximum likelihood estimation. Hence, learning AAMMs reduces to collecting the *sufficient statistics* $\#[s_j, x_i]$ from the set of training sequences, for all $s_j \in \mathcal{S}$ and $x_i \in \mathcal{X}$. Furthermore, for an abstraction $a_j = \{s_{j_1}, \cdots, s_{j_q}\}$, the *sufficient statistics* $\#[a_j, x_i]$ are $\sum_{r=1}^{q} \#[s_{j_r}, x_i]$. The marginal counts of s_j and a_j are obtained by summing out x_i. The parameters θ_s are estimated as in the standard MM.

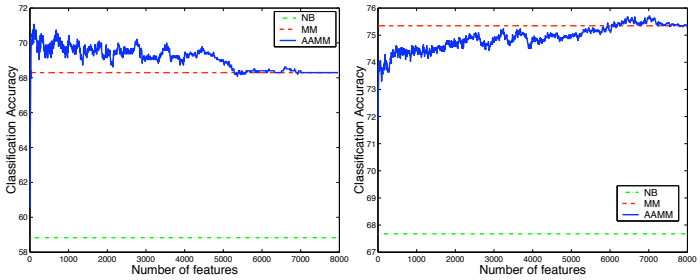
3 Experimental Results

Experimental Design. We evaluated AAMMs on two protein subcellular localization prediction tasks. The problem of predicting subcellular localization is important in cell biology, because it can provide valuable information for predicting protein function and protein-protein interactions, among others. The data sets used in our experiments are **plant** and **non-plant**[1], first introduced in [4].

Note that a generative model can be used for classification tasks by learning a model for each class and selecting the model with the highest posterior probability when classifying new data. We trained AAMMs with several choices of m, where m is the cardinality of the set of abstractions $\mathcal{A}_m$.

We considered two sets of experiments. In our first set of experiments, we compared the performance of AAMM with that of MM and Naïve Bayes (NB) on classification tasks in a supervised setting, where each sequence in the training data is associated with a class from a finite set. In this case, we learned a class-specific AH separately for each class. Furthermore, we learned an AAMM for each class (based on the class-specific AH for that class).

[1] Available online at http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php



(a) plant (b) non-plant

Figure 2: Comparison of AAMM with MM and NB on **plant** (a) and **non-plant** (b), respectively. An abstraction hierarchy is constructed for each class.

In our second set of experiments, we investigated the effect of *abstraction* on the performance of AAMMs in a semi-supervised setting, where only a subset of the training data is labeled and the rest are unlabeled. We performed experiments with 1%, 10%, and 100% of the training data being used as labeled data, with the rest being treated as unlabeled data (by ignoring the class). Because the class is unavailable for a large fraction of data, we learned a single AH from the entire data and used it to learn an AAMM for each class.

For both sets of experiments, we report the average classification accuracy obtained in a 10-fold cross-validation experiment as a function of the number m of abstractions used as features in the classification model, where m ranges from 1 to $|\mathcal{S}| = N$ (the number of unique k-grams).

Comparison of AAMMs with MMs and NB. Figures 2a and 2b show the results of the comparison of AAMMs with MMs using 3-grams on **plant** and **non-plant** data sets, respectively. On the **plant** data set, AAMM substantially outperforms MM over a broad range of choices of *m* (the size of the chosen cut through the AH). The performance of AAMM matches that of MM for AAMMs trained with more than 5285 features (AAMM trained on $|\mathcal{S}|$ features is the same as MM). On the **non-plant** data set, AAMM achieves performance comparable to that of MM with approximately 2500 features.

Hence, AAMMs can achieve significantly lower model sizes (by 1 to 3 orders of magnitude) than MMs for comparable classification performance. Figures 2a and 2b also show the comparison of AAMMs with NB. As expected, for the same number of features used to train the models (20 features), both AAMM and MM (3^{rd} order) are superior in performance to NB (0^{th} order MM).

Evaluation of Abstraction in a Semi-supervised Setting. Figure 3 shows the results of the comparison of AAMMs with MMs using 3-grams for 1%, 10%, and 100% of labeled data for both **plant** and **non-plant** data sets, respectively. Note that an MM is trained on the same fraction of labeled data as its AAMM counterpart. As can be seen in the figure, in the case when only 1% of the data is labeled, AAMMs substantially outperforms MMs on both data sets. When we increase the number of labeled data to 10%, AAMMs still have a much higher performance than MMs. In the case when the entire data set is labeled (100%), the performance of AAMMs is comparable to that of MMs on both data sets.

Note that Figure 3 shows results obtained with AAMM based on a single class-independent AH. In contrast, the results shown in Figure 2 correspond to AAMMs that are based on class-specific AHs. Not surprisingly, comparison of results in Figures 2 and 3 (third column with 100% of the training data being labeled), AAMMs constructed using class-specific AHs outperform AAMMs constructed using a single class-independent AH.

4 Summary and Discussion

We have presented AAMMs that simplify the data representation used by the standard MMs. The results of our experiments have shown that organizing data in a hierarchy using *abstraction* makes it

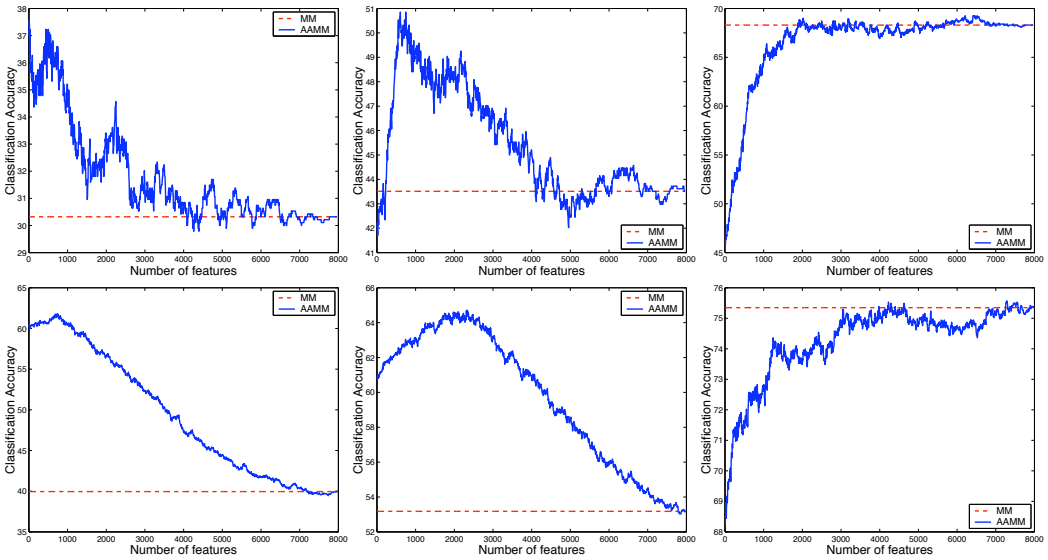


Figure 3: Comparison of AAMM and MM for 1% (first column), 10% (second column), and 100% (third column) of labeled data for **plant** (upper row) and **non-plant** (lower row), respectively. An abstraction hierarchy $\mathcal{T}$ is constructed from all training data, independent of the class variable.

possible to construct predictive models that have significantly smaller size (by 1 to 3 orders of magnitude) as compared to the size of the corresponding MM (which is exponential). The performance of AAMM is similar, and in some cases better, than that of standard MMs. Moreover, the results have shown that *abstraction* allows us to make better use of unlabeled training data.

Segal et al. [7] AHs over *classes* to inprove classification accuracy. Zhang and Honavar [9] have used AHs over nominal variables to build compact yet accurate classifiers. Slonim and Tishby [8] have introduced the so-called *agglomerative information bottleneck* (IB) methods that perform model compression by identifyng a set of *bottleneck variables*. Unlike IB methods, AAMMs exploit AHs that compress k-grams based on conditional distributions of each element in the sequence given its *parents*, rather than class conditional distributions.

Further research includes a thorough comparison of the approach presented here with hidden Markov models [3] and agglomerative IB approaches [8] on a broad range of sequence classification tasks, as well as extensions of our model to settings where the data have a richer structure (e.g., images).

References

[1] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2001.

[3] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press., 2004.

[4] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *J. Mol. Biol.*, 300:1005–1016, 2000.

[5] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. *Pattern Recognition in Practice*, pages 381–397, 1980.

[6] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

[7] E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In *Advances in Neural Information Processing Systems (NIPS 2001)*, pages 913–920, Vancouver, Canada, December 2002.

[8] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems (NIPS-12)*, pages 617–623. MIT Press, 1999.

[9] J. Zhang and V. Honavar. Learning decision tree classifiers from attribute value taxonomies and partially specified data. In *Proceedings of ICML 2003*, Washington, DC, 2003.