A Probabilistic Model for Mining Labeled Ordered Trees: Capturing Patterns in Carbohydrate Sugar Chains

Nobuhisa Ueda, Kiyoko F. Aoki-Kinoshita, *Member*, *IEEE*, Atsuko Yamaguchi, Tatsuya Akutsu, and Hiroshi Mamitsuka, *Member*, *IEEE*

Abstract—Glycans, or carbohydrate sugar chains, which play a number of important roles in the development and functioning of multicellular organisms, can be regarded as labeled ordered trees. A recent increase in the documentation of glycan structures, especially in the form of database curation, has made mining glycans important for the understanding of living cells. We propose a probabilistic model for mining labeled ordered trees, and we further present an efficient learning algorithm for this model, based on an EM algorithm. The time and space complexities of this algorithm are rather favorable, falling within the practical limits set by a variety of existing probabilistic models, including stochastic context-free grammars. Experimental results have shown that, in a supervised problem setting, the proposed method outperformed five other competing methods by a statistically significant factor in all cases. We further applied the proposed method to aligning multiple glycan trees, and we detected biologically significant common subtrees in these alignments where the trees are automatically classified into subtypes already known in glycobiology. Extended abstracts of parts of the work presented in this paper have appeared in [35], [4], and [3].

Index Terms—Biology and genetics, machine learning, data mining, mining methods and algorithms.

1 Introduction

Labeled ordered trees are drawing considerable attention as essential semistructured data and they have appeared in several major data mining applications such as text mining, Web mining, and bioinformatics. A tree-structured text format called XML has become a popular method for storing documents and has been extensively used recently [2], especially on the World Wide Web. With larger XML documents stored on the web, mining XML documents has become an important data mining domain. Thus, a number of approaches, including mining frequent patterns [13], [33], [38], mining data-streams [6], and detecting fraud [12] have been developed for data sets of labeled ordered trees. Kernels for labeled ordered trees have also been developed in the last few years [24].

Semistructured data also appear in biological domains. Glycans, or carbohydrate sugar chains, are well-known as the third major class of biological molecules, subsequent to DNA and proteins, and the recent advent of glycome informatics has generated an increasing number of glycan structure and annotation data [5]. Glycans are rooted tree structures, whose nodes are labeled by monosaccharides, the molecular base units of glycans. Furthermore, the siblings in glycans, labeled by monosaccharides, are ordered, so we may consider glycans as labeled ordered trees. Because of the difficulty in determining the structure of glycans in detail due to technical challenges inherent in the biology of glycans, there remains much to learn about these glycans essential to the development and function of complex multicellular organisms [9]. Since glycans are known to be used as identifiers for enzymes as well as for recognition by microbes and pathogens [10], there seems to be some underlying complex pattern inherent in the structure of glycans, making our model very well suited for this domain.

The importance of labeled ordered trees in a variety of domains is very clear as shown above, and probabilistic modeling and learning is a standard approach in machine learning and data mining. However, only a few probabilistic models have been developed for labeled ordered trees. Considering semistructured data, a probabilistic model, called the hidden Markov model (HMM) [8], has been successfully applied to numerous applications such as speech recognition [32] and bioinformatics [17]. A variety of models, such as the hierarchical hidden Markov model (HHMM) [18], [36], stochastic context-free grammars (SCFG) [7], [26], and probabilistic tree grammars [1], have already been proposed to extend HMMs in the context of probabilistic models and graphical modeling, but all of these models are only applied to sequence data. That is, each of these models uses a latent variable corresponding to a hidden dependency in a sequence to capture long-range interactions over a sentence. The only extension of HMM known to the best of our knowledge for modeling labeled trees is the hidden tree Markov model (HTMM) [14], [16] (or tree hidden Markov model), which captures a hidden relationship between a vertex and its parent. This dependency is equivalent to that of HMMs, where there exists a dependency between two adjacent letters (labels) in a

• *The authors are with the Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho Uji 6110011, Japan.*
E-mail: {ueda, kiyoko, atsuko, takutsu, mami}@kuicr.kyoto-u.ac.jp.

Manuscript received 22 July 2004; revised 22 Oct. 2004; accepted 10 Feb. 2005; published online 17 June 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0256-0704.

TABLE 1
Extensions of HMM to the Proposed Model

Models	Dependency relation
HMM	Two adjacent labels in a sequence
HTMM	Parent-child in a tree
Proposed model	Parent-child and two adjacent siblings in a tree

sequence. On the other hand, a labeled ordered tree has ordered siblings, and so there must be a dependency between siblings as well. For labeled ordered trees, we need another model to capture not only parent-child dependencies but also those between siblings.

We therefore propose a probabilistic model for labeled ordered trees by extending HTMM to a model in which a vertex depends on its immediately elder sibling as well as on its parent. This extension considers the dependencies between ordered siblings as well as parent-child relationships in a tree, which is enough for capturing various types of patterns in labeled ordered trees. Table 1 summarizes these two extensions from HMM to the proposed model.

We also propose a learning algorithm based on the Expectation-Maximization (EM) algorithm [15] for our proposed model. The time and space complexities of our learning algorithm are roughly $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$ and $O(|S|^3 \cdot |V|)$, respectively, where $|\mathbf{T}|$ is the number of trees in a given data set, $|S|$ is the number of states in the model, and $|V|$ is the maximum number of nodes of any tree in $\mathbf{T}$. We note that this algorithm has several very nice properties in terms of complexity. First, our proposed model may be considered a Bayesian belief network [31], belonging to its computationally intractable subclass called a multiply-connected Bayesian network [31]. A standard learning algorithm for this subclass has an extra step of modifying a given network into a data structure called a junction tree, although the computational complexity of this step may reach $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$. Furthermore, the space complexity of this algorithm reaches $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$, whereas that of our proposed algorithm is $O(|S|^3 \cdot |V|)$, which is independent of the number of trees. Therefore, because our model is constrained within a subset of Bayesian belief networks such that it can focus on labeled ordered trees, we could use an efficient EM algorithm for learning/mining labeled ordered trees. Second, the corresponding computation time for both HMM and HTMM is roughly $O(|\mathbf{T}| \cdot |S|^2 \cdot |V|)$. Although they are faster by a factor of $|S|$, we claim that our method provides far richer expressive power, which cannot be compared to this increase in computation time. Moreover, third, this increase in time by the proposed model is negligible since learning the proposed model is more efficient than learning SCFG, which has been used practically in a number of applications such as natural language processing [26]. The time complexity of learning SCFG is $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|^3)$ time,[1] where $|V|$ is the maximum length of the given sentences. We then claim that the proposed model has reasonable efficiency in practice.

We experimentally evaluated the effectiveness of our proposed method, using a variety of experiments and data sets, including real data sets of glycans. We first evaluated the performance of the proposed method by comparing it with other probabilistic methods in a supervised manner using classification. More concretely, we checked the performance of discriminating positive examples from randomly generated negatives, and we examined the performance of each of the methods by three measures, area under the ROC (AUC), prediction accuracy, and precision. Experimental results show that our proposed method outperforms all of the other methods compared by a statistically significant factor in all cases. We further applied our method to aligning multiple glycans and analyzed the results obtained. We first automatically found common subtrees (i.e., patterns of glycans) from aligned glycans and confirmed that these subtrees match biologically known motifs in glycans. Further empirical findings include the automatic classification of trees into three types using the most likely state transitions, where, in fact, these three types correspond to three biologically well-known types of glycans. Overall, these results confirm that the proposed model is especially effective for mining glycans, a complex biological example of labeled ordered trees.

The rest of this paper is organized as follows: In Section 2, we describe the notations used in this paper as well as the probabilistic structure and a learning algorithm of HTMM. In Section 3, we describe the details of our probabilistic model and algorithms to compute the likelihood of given examples, to compute the most likely transition, and to estimate the probability parameters of our model from a given set of trees. In Section 4, we show the experimental results obtained by applying our method to synthetic data as well as to actual glycan data sets and the performance advantage of our proposed method over other competing methods. We further show the biological findings obtained by applying our method to real data sets of glycans. Finally, we conclude in Section 5.

2 Preliminaries

2.1 Notations

We first describe notations that will be used throughout this paper. A *tree* is an acyclic connected graph. In this paper, we refer to a vertex of a tree as a *node* of the tree. A *rooted tree* is a tree that has a special node called the root. Any node x on a unique path from the root to a node y is called an *ancestor* of node y, in which case y is called a *descendant* of x. Each of the closest descendants of x (that is, a node that is only

1. It is well-known that the time for learning SCFG is reducible to $O(|\mathbf{T}| \cdot |R| \cdot |V|^3)$ time, where $|R| \leq |S|^3$ is the number of rules in an SCFG. In a manner similar to SCFG, we can modify the learning procedure for our proposed model to one that only takes $O(|\mathbf{T}| \cdot |R| \cdot |V|)$ time, where $|R|$ is the number of possible state transitions.

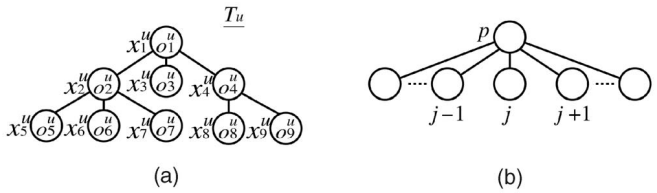

Fig. 1. Notations for labeled ordered trees. (a) Labeled ordered tree T_u. (b) Indices for a parent and its children.

one edge away from node x) is called a *child* of x, in which case x is called the parent of the child. We call nodes x and y siblings if x and y have the same parent. We call a node having no children a *leaf*. A *subtree* of tree T is a tree consisting of all descendants of a node. An *ordered tree* is a rooted tree in which the children of each node are ordered. A *labeled tree* is a tree in which a label is attached to each node. We will often simply use the term *tree* in place of an ordered, labeled, and rooted tree.

Let $\mathbf{T} = \{T_1, \ldots, T_{|\mathbf{T}|}\}$ be a set of labeled ordered trees, where $T_u = (V_u, E_u)$ and $V_u (= \{x_1^u, \ldots, x_{|V_u|}^u\})$ and $E_u \subseteq V_u \times V_u$ are a set of nodes and a set of edges, respectively. Let x_1^u be the root of tree T_u and $|V| = \max_u |V_u|$. We assume that nodes are indexed by level order, which can be done by traversing the tree in breadth-first order. An example of trees with their indices is shown in Fig. 1. From this indexing of nodes, for a node j, we can refer to the immediately elder and younger siblings of j as $j-1$ and $j+1$, respectively. Let $t_u(i)$ be a subtree of T_u, having x_i^u as the root of $t_u(i)$. Let $x_{\leftarrow}^u(p)$ and $x_{\rightarrow}^u(p)$ be the eldest and youngest children of node p, respectively. Let $C_u(p) \subseteq \{1, \ldots, |V_u|\}$ be a set of indices of children of x_p^u in T_u, $|C| = \max_{u,p} |C_u(p)|$, and $Y_u(p) = C_u(p) \backslash \{i\}$ such that $x_i^u = x_{\leftarrow}^u$. Each node x_j^u has label $o_j^u \in \Sigma$, where $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ is a set of labels on nodes. For simplicity, we will often use node j instead of x_j^u and p as a parent node, if understood from the context.

Let θ denote a set of parameters of a probabilistic model. For simplicity, we may use $\theta = \{\theta_1, \ldots, \theta_n\}$ as a set of parameters, such that $\theta_i = \{\theta_{i,1}, \ldots, \theta_{i,|\theta_i|}\}$ and $\sum_{j=1}^{|\theta_i|} \theta_{i,j} = 1$ for $i = 1, \ldots, n$. A probabilistic model has a "state" corresponding to a node, and each state has a probabilistic parameter which probabilistically generates a label at a node. Let $S = \{s_1, \ldots, s_{|S|}\}$ be a set of states and $z_j^u \in S$ be a state for node j in a tree. For simplicity, we may also use j instead of z_j^u and q as the state of a parent node, if understood from the context.

2.2 Hidden Tree Markov Model

We review an existing probabilistic model called the hidden tree Markov model [16] (or hidden Markov tree model [14]) for labeled trees and its learning algorithm. In this model, the state of a node depends only on the state of its parent and, in this sense, this model is a straightforward extension of a hidden Markov model for sequences to a model for labeled trees. Fig. 2 illustrates the dependencies in HTMM for the tree in Fig. 1. HTMM has three types of probability parameters, π, a, and b. The initial state probability $\pi[l] (= P(z_1^u = s_l; \theta))$ is the probability that the state z_1^u of the root node x_1^u is s_l. The state transition probability $a[l, m]$ is the

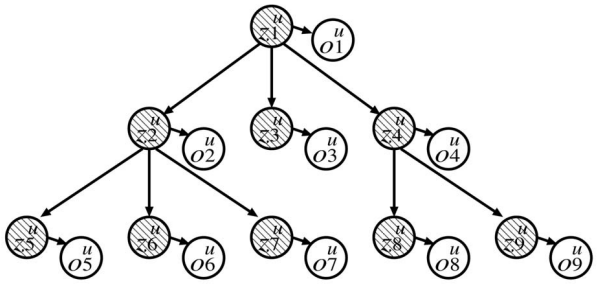

Fig. 2. Graphical representation of HTMM for tree T_u in Fig. 1.

conditional probability that the state of a node is s_m given that the state of its parent is s_l. The label output probability $b[l, \sigma_h] (= P(o_j^u = \sigma_h | z_j^u = s_l; \theta))$ is the conditional probability that the output label of a node is σ_h given that the state of this node is s_l. Note that $\sum_l \pi[l] = 1$, $\sum_m a[l, m] = 1$, and $\sum_h b[l, \sigma_h] = 1$.

To compute these probabilities efficiently, we define an upward probability $\alpha_u(l, j) (= P(t_u(j) | z_j^u = s_l; \theta))$, which is the probability that the labels of subtree $t_u(j)$ are all generated and that the state of node x_j^u is s_l. Note that the upward probability at node j can be calculated from the upward probability at each of the children of j and the probability parameters of HTMM. This can be formulated as follows:

$$\alpha_u(l, p) = \begin{cases} b[l, o_p^u] & \text{if } C_u(p) = \emptyset, \\ b[l, o_p^u] \prod_{j \in C_u(p)} \sum_{m=1}^{|S|} a[l, m] \alpha_u(m, j) & \text{otherwise.} \end{cases}$$

To obtain the upward probabilities for each of the nodes of a given tree, we use a bottom-up dynamic programming procedure. That is, we recursively compute the upward probability at each node from the leaves to the root of the tree.

We then compute the likelihood $L(T_u; \theta)$ for T_u, which is given by HTMM as follows:

$$L(T_u; \theta) = \sum_{l=1}^{|S|} \pi[l] \alpha_u(l, 1).$$

We can finally compute the likelihood for a given set of trees as follows:

$$L(\mathbf{T}; \theta) = \prod_{u=1}^{|\mathbf{T}|} L(T_u; \theta) = \prod_{u=1}^{|\mathbf{T}|} \sum_{l=1}^{|S|} \pi[l] \alpha_u(l, 1).$$

A standard criterion for estimating the probability parameters of HTMM is the maximum likelihood (ML), in which parameters are estimated to maximize the above likelihood for a set of given trees. We can use a general scheme called the EM (Expectation-Maximization) algorithm [15] to obtain the ML estimation of HTMM. In the EM algorithm for HTMM, we define a downward probability $\beta_u(l, i)$, which is the probability that all labels of tree T_u, except for those of subtrees $t_u(i)$, are generated and that the state of node i is s_l. By using the downward probability as well as the upward probability, we can implement the EM algorithm for HTMM in a manner similar to the Baum-Welch algorithm in HMM. Therefore,

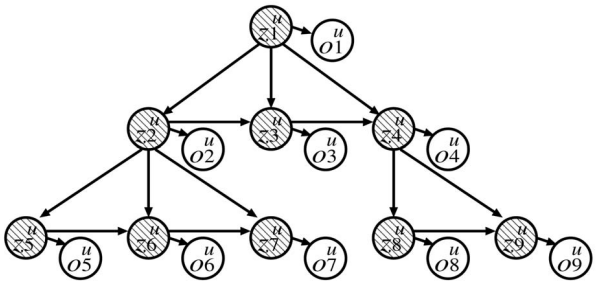

Fig. 3. Graphical representation of PSTMM for tree T_u in Fig. 1.

HTMM would take $O(|T| \cdot |S|^2 \cdot |V| \cdot |C|)$ time to calculate the likelihood.

HTMM can be used for modeling labeled trees when a dependency exists between a node and its parent in a tree. However, if there is a dependency in ordered siblings in a labeled ordered tree, it cannot be captured by HTMM.

3 Proposed Model and Algorithms

We propose a new probabilistic model for labeled ordered trees which we hereafter call *probabilistic sibling-dependent tree Markov model*, or PSTMM for short. In HTMM, the state of a node depends on the state of its parent only, whereas in PSTMM, the state of a node depends on the state of its immediately elder sibling as well as the state of its parent. Fig. 3 illustrates the dependencies in PSTMM embedded in the tree of Fig. 1. We emphasize that incorporating this dependency on the immediately elder sibling drastically improves the performance of HTMM in finding patterns in labeled ordered trees. Recall that HMMs can capture a distant (long-range) dependency in a sequence indirectly if a state transition can be set to capture such a dependency. Similarly, HTMM can capture a long-range dependency in a tree indirectly, but this indirect dependency is limited to the one between a node and its descendant. On the other hand, PSTMM can capture a variety of complex dependencies in a tree as well as the descendant dependency. For example, a dependency between distant siblings may be found by PSTMM indirectly. Furthermore, an indirect dependency between a node and its distant sibling's descendant may also be captured by PSTMM.

PSTMM has three types of probability parameters, π, a, and b, where π and b are the same as those defined in HTMM. However, we note that a differs between HTMM and PSTMM. The state transition probability $a[\{q,l\},m](= P(z_j^u = s_m | z_p^u = s_q, z_i^u = s_l; \theta))$ is newly defined as the conditional probability that the state of a node is s_m given that the states of its parent and the immediately elder sibling are s_q and s_l, respectively. Note that $\sum_{m=1}^{|S|} a[\{q,l\},m] = 1$.

Most applications of PSTMM can be summarized as the following three problems as done for HMMs [32]:

1. Likelihood computation: computing the likelihood of a given tree,
2. Parsing (Prediction): finding the most likely state transition for a given tree, and
3. Learning: estimating probability parameters from a set of given trees.

Thus, in the following sections, we will explain our efficient algorithm of PSTMMs for each of these problems.

3.1 Likelihood Computation

To compute the likelihood of a given tree, we define a backward probability as well as an upward probability. The upward probability $U_u(q,p)$ is the probability that all labels of subtree $t_u(p)$ are generated and the state of node p is s_q. The backward probability $B_u(q,l,j)$ is the probability that, for node j, all labels of a subtree for each of the elder siblings and node j are generated, the state of j is s_l, and the state of parent p is s_q.

We can compute these two probabilities, from leaves to the root, using a bottom-up dynamic programming procedure similar to that used in HTMM. This computation can be formulated as follows:

$$U_u(q,p) = \begin{cases} b[q, o_p^u] & \text{if } C_u(p) = \emptyset, \\ b[q,o_p^u] \sum_{m=1}^{|S|} a[\{q,-\},m] B_u(q,m,j) (\text{s.t. } x_j^u = x_{\leftarrow}^u) & \text{otherwise,} \end{cases} \tag{1}$$

$$B_u(q,m,j) = \begin{cases} U_u(m,j) & \text{if } x_j^u = x_{\rightarrow}^u(p), \\ U_u(m,j) \sum_{l=1}^{|S|} a[\{q,m\},l] B_u(q,l,j+1) & \text{otherwise.} \end{cases} \tag{2}$$

The likelihood for a given tree T_u is obtained by using $U_u(l,1)$, U at the root of the tree, in a manner similar to HTMM, as in the following equation:

$$L(T_u; \theta) = \sum_{l=1}^{|S|} \pi[l] U_u(l,1).$$

The likelihood for a given set of trees is defined as a product of the likelihood for each tree in the set:

$$L(\mathbf{T}; \theta) = \prod_{u=1}^{|\mathbf{T}|} L(T_u; \theta) = \prod_{u=1}^{|\mathbf{T}|} \sum_{l=1}^{|S|} \pi[l] U_u(l,1).$$

In HTMM, the upward probability α at a node is iteratively computed from the αs at its children. On the other hand, in PSTMM, the upward probability U at a node is computed using the backward probability B of the eldest sibling, and B at a node is iteratively computed using the U at the node. Figs. 4a and 4b depict the recursive calculation of U and B, respectively. Time and space complexities of this algorithm are shown in Table 2.

3.2 Parsing

We can find the most likely state transition of a given tree by taking the U and B probability parameters and modifying them to calculate two new probabilities, $\phi_U(q,p)$ and $\phi_B(q,m,j)$. $\phi_U(q,p)$ is the maximum probability that, for a state transition from node p, all labels of subtree $t_u(p)$ are generated and the state of node p is s_q. Similarly, $\phi_B(q,m,j)$ is the maximum probability that, for a state transition from node j, the state of j is s_q and all the labels of the subtrees for each of the younger siblings and node j are

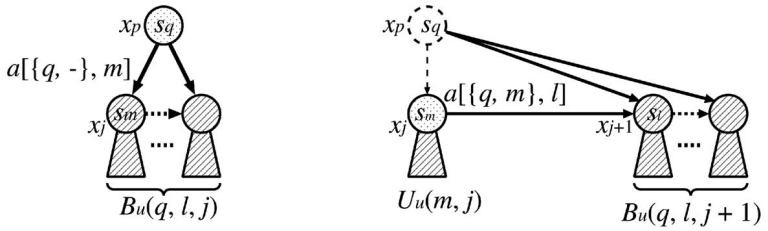

Fig. 4. Updating (a) $U_u(q,p)$ and (b) $B_u(q,m,j)$. The sparse shaded node is p for $U_u(q,p)$ and j for $B_u(q,m,j)$. Dense shaded areas are used for updating.

generated. These probabilities are obtained by replacing $\sum$ in (1) and (2) with $\max$, as follows:

$$\phi_U(q,p) = \begin{cases} b[q,o_p^u] & \text{if } C_u(p)=\emptyset, \\ \max_m b[q,o_p^u]a[\{q,-\},m]\phi_B(q,m,j)(\text{s.t. } x_j^u = x_{\leftarrow}^u(p)) & \text{otherwise,} \end{cases} \tag{3}$$

$$\phi_B(q,m,j) = \begin{cases} \phi_U(m,j) & \text{if } x_j^u = x_{\rightarrow}^u(p), \\ \phi_U(m,j)\max_l a[\{q,m\},l]\phi_B(q,l,j+1) & \text{otherwise.} \end{cases} \tag{4}$$

We then need to retrieve a state transition which gives these values, and so we use $\tau_U(q,p)$ and $\tau_B(q,m,j)$ which can be computed by the following equations, slightly modified from (3) and (4) above, as follows:

$$\tau_U(q,p) = \begin{cases} 0 & \text{if } C_u(p)=\emptyset, \\ \arg\max_m b[q,o_p^u]a[\{q,-\},m]\phi_B(q,m,j)(\text{s.t. } x_j^u = x_{\leftarrow}^u(p)) & \text{otherwise,} \end{cases}$$

$$\tau_B(q,m,j) = \begin{cases} 0 & \text{if } x_j^u = x_{\rightarrow}^u(p), \\ \arg\max_l \phi_U(m,j)a[\{q,m\},l]\phi_B(q,l,j+1) & \text{otherwise.} \end{cases}$$

We now define P^*, which is the probability that all labels are outputted along the most likely state transition. P^* can be computed using ϕ_U above, as follows:

$$P^* = \max_l \pi[l]\phi_U(l,1).$$

We further define q_j^*, which is the most likely state at node j for a given tree. We can compute q_1^* by tracing the states giving the maximum probability at each node x_j ($j = 2,\ldots,|V_u|$) over the tree using τ_U and τ_B above, as follows:

$$q_1^* = \arg\max_l \pi[l]\phi_U(l,1),$$
$$q_j^* = \tau_U(q_p^*) \text{ if } x_j = x_{\leftarrow}^u(p),$$
$$q_j^* = \tau_B(q_{j-1}^*) \text{ otherwise.}$$

From all of these computations, we can finally obtain the most likely state transition for the given tree T_u as the resulting set of states $Z_{u,*} = \{q_1^*,\ldots,q_{|V_u|}^*\}$. Time and space complexities of this parsing algorithm are shown in Table 2.

TABLE 2
Time and Space Complexities

	Time	Space
U, ϕ_U, τ_U	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^2 \cdot \lvert V\rvert)$	$O(\lvert S\rvert \cdot \lvert V\rvert)$
B, ϕ_B, τ_B	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^3 \cdot \lvert V\rvert)$	$O(\lvert S\rvert^2 \cdot \lvert V\rvert)$
F	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^3 \cdot \lvert V\rvert)$	$O(\lvert S\rvert^2 \cdot \lvert V\rvert)$
D	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^3 \cdot \lvert V\rvert)$	$O(\lvert S\rvert \cdot \lvert V\rvert)$
$\mu(a)$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^3 \cdot \lvert V\rvert \cdot \lvert C\rvert)$	$O(\lvert S\rvert^3)$
$\mu(b)$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert \cdot \lvert V\rvert)$	$O(\lvert S\rvert \cdot \lvert\Sigma\rvert)$
$\mu(\pi)$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert \cdot \lvert V\rvert)$	$O(\lvert S\rvert)$
$\hat{a}$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert^3)$	$O(\lvert S\rvert^3)$
$\hat{b}$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert \cdot \lvert V\rvert)$	$O(\lvert S\rvert \cdot \lvert\Sigma\rvert)$
$\hat{\pi}$	$O(\lvert\mathbf{T}\rvert \cdot \lvert S\rvert)$	$O(\lvert S\rvert)$

3.3 Learning: Maximum Likelihood

As we mentioned for HTMM, the maximum likelihood is a general criterion used to estimate the probability parameters of a probabilistic model from given training examples. We use an EM algorithm [15], a general and popular scheme to maximize the likelihood for a given set of examples.

3.3.1 An EM Algorithm

In addition to the two probabilities U and B defined already, we define the forward probability $F_u(q,l,j)$ and the downward probability $D_u(l,j)$. The forward probability $F_u(q,l,j)$ is the probability that, for node j, all labels of the subtrees at each of the elder siblings and node j are generated, the state of node j is s_l, and the state of parent p is s_q. The downward probability $D_u(l,j)$ is the probability that all labels of tree T_u, except for those of subtree $t_u(j)$, are generated and that the state of node x_j^u is s_l.

We can compute these probabilities $F_u(q,l,j)$ and $D_u(l,j)$ as follows:

$$F_u(q,l,j) = \begin{cases} a[\{q,-\},l] & \text{if } x_j^u = x_{\leftarrow}^u(p), \\ \sum_{m=1}^{|S|} F_u(q,m,j-1)U_u(m,j-1)a[\{q,m\},l] & \text{otherwise,} \end{cases}$$

$$D_u(l,j) = \begin{cases} \pi[l] & \text{if } j = 1, \\ \sum_{q=1}^{|S|} D_u(q,p)b[q,o_p^u]F_u(q,m,j) & \text{if } x_j^u = x_{\rightarrow}^u(p), \\ \sum_{q=1}^{|S|} D_u(q,p)b[q,o_p^u]F_u(q,l,j)\sum_{m=1}^{|S|} a[\{q,m\},l]B_u(q,m,j+1) & \text{otherwise.} \end{cases}$$

Figs. 5a and 5b illustrate the recursive calculation of F and D, respectively.

Note that, for any node i, the likelihood of a tree can be computed using the upward and downward probabilities at node i as follows:

$$L(T_u;\theta) = \sum_l U_u(l,i)D_u(l,i).$$

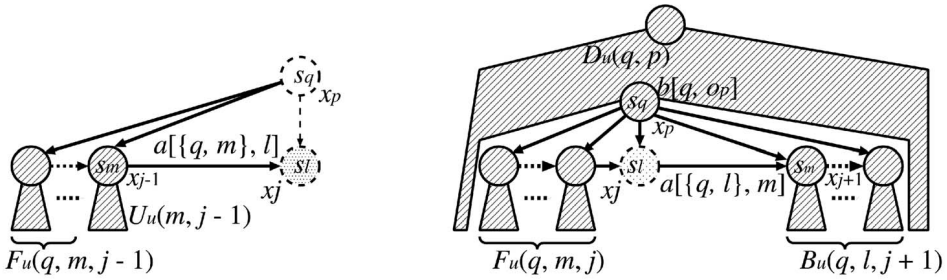

Fig. 5. Updating (a) $F_u(q,l,j)$ and (b) $D_u(l,j)$. The sparse shaded node is j, and dense shaded areas are used for updating.

E-step. For each probability parameter, compute the expectation values $\mu_u(a[\{q,m\},l])$, $\mu_u(b[m,\sigma_h])$, and $\mu_u(\pi[m])$ using the above four probabilities, U, B, F, and D.

$$\mu_u(a[\{q,-\},l]) = \frac{1}{L(T_u;\theta)} \sum_{p:C_u(p)\neq\emptyset} D_u(q,p) b[q,o_p^u] a[\{q,-\},l] B_u(q,l,j'),$$

$$\mu_u(a[\{q,m\},l]) = \frac{1}{L(T_u;\theta)} \sum_{p:C_u(p)\neq\emptyset} D_u(q,p) b[q,o_p^u] \sum_{j:x_j^u\in Y_u(p)} H_u(q,m,l,j),$$

where $H_u(q,m,l,j) = F_u(q,m,i)U_u(m,i)a[\{q,m\},l]B_u(q,l,j)$,

$$\mu_u(b[m,\sigma_h]) = \frac{1}{L(T_u;\theta)} \sum_{i:o_i^u=\sigma_h} D_u(m,i)U_u(m,i),$$

$$\mu_u(\pi[m]) = \frac{1}{L(T_u;\theta)} \pi[m] U_u(m,1),$$

where j' is such that $x_{j'}^u = x_{\rightarrow}^u(p)$.

M-step. Update the probability parameters using the expectation values computed in the E-step:

$$\hat{a}[\{q,-\},l] = \frac{\sum_u \mu_u(a[\{q,-\},l])}{\sum_u \sum_{l'} \mu_u(a[\{q,-\},l'])},$$

$$\hat{a}[\{q,m\},l] = \frac{\sum_u \mu_u(a[\{q,m\},l])}{\sum_u \sum_{l'} \mu_u(a[\{q,m\},l'])},$$

$$\hat{b}[m,\sigma_h] = \frac{\sum_u \mu_u(b[m,\sigma_h])}{\sum_u \sum_i \mu_u(b[m,\sigma_i])},$$

$$\hat{\pi}[m] = \frac{\sum_u \mu_u(\pi[m])}{\sum_u \sum_k \mu_u(\pi[k])}.$$

We repeat these E and M-steps alternately until a certain convergence criterion is satisfied.[2] A possible criterion is that the increase of the likelihood at an iteration is less than a certain small constant.

2. Note that it has been proven that the EM algorithm theoretically converges to a local maximum solution [29].

Fig. 6 shows a sample pseudocode of this EM algorithm for PSTMM. At the initialization step, each parameter is initialized randomly (lines 1-2). We then iterate the E and M-steps of the EM algorithm (lines 4-22) until a certain stopping condition is satisfied (line 21). In the E-step, we first initialize each expectation value for a given data set to zero (line 6), and we then loop through each tree in the data set, calculating each value by summing the expectation values for each tree (line 7). Specifically, for each tree, we first compute U and B using bottom-up and right-to-left dynamic programming (lines 8-10), compute F using bottom-up and left-to-right dynamic programming (lines 11-15), and compute D using top-down and left-to-right dynamic programming (lines 16-17). We then compute the expectation value of each of the parameters (line 18) for the given tree and add it to the expectation value for the entire data set (line 19). In the M-step, we update each parameter by these expectation values (line 20). After the EM iteration, we output the estimated parameters (line 22).

3.3.2 Time and Space Complexities

Table 2 summarizes the time and space complexities of all our algorithms, which include the algorithms for computing the likelihoods, parsing trees, and learning the parameters. Regarding the time complexity, the most time-consuming part is computing $\mu_u(a[\{q,m\},l])$, which is $O(|\mathbf{T}|\cdot|S|^3\cdot|V|\cdot|C|)$. In fact, for this part, we have to take $O(|V|\cdot|C|)$ time to compute each combination of q, m, and l and then repeat this $O(|S|^3)$ times for all possible combinations of states. As for the space complexity, we note that we can initialize the memory space each time the computations for each tree are finished. The space complexity is then upper-bounded by $O(|S|^3\cdot|V|)$, which is independent of $|\mathbf{T}|$ for a given data set. In summary, the time and space complexities of our EM algorithm are upper-bounded by $O(|\mathbf{T}|\cdot|S|^3\cdot|V|)$ and $O(|S|^3\cdot|V|)$, respectively.

We can regard PSTMM as a type of a Bayesian belief network [31]. More precisely, PSTMM belongs to a computationally intractable subclass of Bayesian networks called *multiply-connected Bayesian networks*. We emphasize that our EM algorithm is more efficient than the most popular learning method for a multiply-connected Bayesian network. This subclass is defined as those networks that contain more than one path between any two nodes. Obviously, PSTMM satisfies this, because, for example, there are two paths $z_1^u \rightarrow z_2^u \rightarrow z_3^u$ and $z_1^u \rightarrow z_3^u$ from z_1^u to z_3^u in Fig. 3. The current general approach for estimating the parameters of a multiply connected Bayesian network is the

```
for each θ^(1)_{i,j} do
    initialize θ^(1)_{i,j} randomly s.t. Σ_{j=1}^{|θ_i|} θ^(1)_{i,j} for any i;
L(T|θ^(0)) := −∞; t := 0;
repeat
    t := t + 1;
    for each θ^(t)_{i,j} do μ_sum(θ^(t)_{i,j}) := 0;
    for u := 1 to |T| do
        for j := |V_u| downto 1 do /* bottom-up and right-to-left DP */
            for each q ∈ S do calculate U_u(q, j);
            for each (q, m) ∈ S × S do calculate B_u(q, m, j);
        for j = |V_u| downto 1 do /* bottom-up and left-to-right DP */
            if C_u(j) ≠ ∅ then
                for j' := i downto k s.t. x^u_i = x^u_→(p) and x^u_k = x^u_←(p) do
                    for each (q, l) ∈ S × S do calculate F_u(q, l, j');
        for each (q, l) ∈ S × S do calculate F_u(q, l, 1);
        for j := 1 to |V_u| do /* top-down and left-to-right DP */
            for each q ∈ S do calculate D_u(q, j);
        for each θ^(t)_{i,j} do calculate μ_u(θ^(t)_{i,j});
        for each θ_{i,j} do μ_sum(θ^(t)_{i,j}) := μ_sum(θ^(t)_{i,j}) + μ_u(θ^(t)_{i,j});
    for each θ^(t)_{i,j} do update θ^(t)_{i,j} with μ_sum(θ^(t)_{i,j}) as Σ_u μ_u(θ^(t)_{i,j});
until L(T|θ^(t)) − L(T|θ^(t−1)) < ε
output θ^(t);
```

Fig. 6. Pseudocode of the EM algorithm for PSTMM. $\theta^{(t)}_{i,j}$ stands for a parameter at tth iteration.

junction tree algorithm [27]. The time complexity of this algorithm is $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$, which is equivalent to our algorithm for PSTMM. However, we note that the junction tree algorithm requires extra computation time to construct and store for each tree a specific data structure called a *junction tree*. Furthermore, the space complexity of this algorithm is $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$, while ours is $O(|S|^3 \cdot |V|)$. Therefore, our algorithms are rather efficient in comparison to the existing algorithm.[3]

4 Experimental Results

We first evaluated the performance of our approach by comparing it with those of five simpler probabilistic models, including HTMM. The data sets used in this evaluation are synthetic data sets as well as real glycan data sets. We then further evaluated our approach by aligning a given set of multiple glycans to find common subtrees in the given set. We further checked the validity of those common subtrees from a biological viewpoint.

4.1 Simple Probabilistic Models Used in Experiments

To illustrate the advantage of PSTMM, we used four simple probabilistic models, which we call a label model (LM), a mixture[4] of label models (MLM), a label pair model (LPM), and a mixture of label pair models (MLPM), in addition to HTMM. We note that all of these simpler models do not consider any dependencies between siblings. Learning algorithms of these models are given in Supplementary Information 2 in the Appendix which can be found on the Computer Society Digital Library at http://computer.org/tkde/archives.htm.

LM is defined only with the label output probability $w[\sigma_h]$, which is the probability that label σ_h is outputted at a node, where $\sum_h w[\sigma_h] = 1$. In this model, labels at different nodes are outputted independently. MLM is a mixture of LMs, each of which we here call a *component*. Let c be a component and Z be the number of components. Each component is assigned different probability parameter values, and we have two probability parameters, $w[c, \sigma_h]$ ($\sum_h w[c, \sigma_h] = 1$ for each c) and $v[c]$ ($\sum_c v[c] = 1$). Thus, we have defined two models that cannot capture the probabilistic dependencies between a parent and a child. MLM contains a hidden variable Z, and we then use the EM algorithm as given in Supplementary Information 2 in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tkde/archives.htm) to estimate the parameters of MLM.

LPM captures a dependency between labels on a node and on its child. This model has two parameters, $w[\sigma_h, \sigma_{h'}]$ ($\sum_h w[\sigma_h, \sigma_{h'}] = 1$) and $\pi[h]$ ($\sum_h \pi[\sigma_h] = 1$). $w[\sigma_h, \sigma_{h'}] (= P(o^u_j = \sigma_h, o^u_p = \sigma_{h'}))$ is the label pair probability, or the probability that label σ_h is outputted at a node given that label $\sigma_{h'}$ is outputted at its parent node, and $\pi[\sigma_h]$ is the probability that the root label is σ_h. MLPM has three probability parameters, $w[c, \sigma_h, \sigma_{h'}]$, $\pi[c, \sigma_h]$, and $v[c]$ such that $\sum_h w[c, \sigma_h, \sigma_{h'}] = 1$ for any c and $\sigma_{h'}$, $\sum_h \pi[c, \sigma_h] = 1$ for any c, and $\sum_c v[c] = 1$. $w[c, \sigma_h, \sigma_{h'}] (= P(o^u_j = \sigma_h | c, o^u_p = \sigma_{h'}))$ is the conditional probability that σ_h is outputted at a node given that $\sigma_{h'}$ is outputted at its parent node for component c. $\pi[c, \sigma_h]$ $(= P(o^u_1 = \sigma_h | c))$ is the probability

3. To be more precise, the space complexity for the junction tree algorithm is $O(|\mathbf{T}| \cdot |S|^3 \cdot |V'|)$, where $|V'|$ is the number of nodes in a junction tree converted from a given tree. However, in general, we claim that there is no upper bound for $|V'|$ better than $|V|$.

4. For details on mixture models, we refer the reader to [30].

TABLE 3
Comparison between Models for Labeled Trees and Sequences

For labeled trees	For sequences
LPM	P1MM
MLPM	MP1MM
HTMM	HMM

that σ_h is outputted at the root node for component c. As in MLM, MLPM also contains a hidden variable Z, and its parameters are also estimated by the EM algorithm as described in Supplementary Information 2 in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tkde/archives.htm).

Let us compare these models for trees with models for sequences. First, a hidden Markov model (HMM) corresponds to an HTMM for labeled trees. In HMM, a letter emitting probability is attached to each state and state transitions may occur between any letter regardless of its position in the sequence. If each state outputs only one particular symbol in HMM, this model can be considered to be a model having no hidden variables. This simpler model is usually called a probabilistic first-order Markov model (P1MM), which corresponds to LPM for labeled trees. We can naturally consider a mixture of P1MM (MP1MM), which corresponds to MLPM for labeled trees. Table 3 summarizes these relations between models for trees and models for sequences.

Note that, for capturing sequence patterns based on the first-order Markov property, MP1MM has representational power equal to that of HMM and, as such, MP1MM has been frequently used for obtaining multiple sequence patterns from sequences such as in Web access patterns [11]. Similarly, this equivalence can be applied to MLPM and HTMM for capturing multiple parent-child relationship in a given set of trees.

4.2 Performance Evaluation by Classification

We evaluated each model in a supervised manner. More concretely, we first generated training and test examples, consisting of positive examples only, and we trained each model with the generated training examples. We then generated negative test examples such that their distribution of parent-child pair labels was equal to that of the positive test examples. We evaluated each of the five models by their ability to discriminate positives from negatives in each test data set. Note that since the distribution of parent-child pair labels was made the same for both positive and negative examples, it was a difficult task for the simpler probabilistic models, LM, MLM, LPM, MLPM, and HTMM, to discriminate positives from negatives. We performed a five-fold cross-validation for each of the data sets. That is, we divided each data set into five blocks of roughly equal size and, in each of the five trials, a different block was selected as the test set while the remaining four were used for training. We finally repeated this process five times. The results were then averaged over the $25(= 5 \times 5)$ runs. We evaluated the performance of each of the models by the following three measures: AUC, prediction accuracy, and precision at recall of 30 percent.

The AUC is the area under the ROC (Receiver Operating Characteristic) curve. The ROC curve is a curve plotting recall against the false positive rate for different classification thresholds from $-\infty$ to ∞ [20], where the recall is the proportion of the number of correctly predicted examples to the total number of positive examples, and the false positive rate is the proportion of the number of false positives to the total number of negative examples. Let G denote a classifier. The AUC [20] of G is calculated by

$$\mathrm{AUC}(G) = \int_{x=-\infty}^{\infty} \mathrm{f}(G,x)d\mathrm{r}(G,x),$$

where x is a threshold, $\mathrm{f}(G,x)$ is the recall of G given x, and $\mathrm{r}(G,x)$ is the false positive rate of G given x. AUC takes on a value between 1 and 0, with a larger value indicating better performance.

We computed the prediction accuracy for each of the six models by choosing a threshold that discriminates the positives from the negatives such that the threshold maximizes the discrimination accuracy. In order to find this threshold, we evaluated the discrimination accuracy for all possible thresholds. To be more precise, we first sorted the examples in descending order of their computed likelihoods (or scores). Then, the threshold can be found between the pair of consecutive examples from the sorted list of scores such that the positives above the threshold and the negatives below are most discriminated. The prediction accuracy takes a value between 1 and 0, with a larger value indicating higher accuracy.

We also used precision and recall. Precision is the proportion of the number of correctly predicted examples to the number of those examples predicted to be positives. If we actually check the true label of a predicted example whose label is unknown, we will choose a relatively small subset for which a reasonably high precision can be achieved. For our experiments, we chose 0.3 as a reasonable recall value.

We further used the pairwise mean significance statistical test of "t" values to statistically compare the performance of PSTMM with each of the other models. The t values are calculated using the following formula:

$$t = \frac{|ave(A)|}{\sqrt{\frac{var(A)}{n}}},$$

where A denotes the difference between the performance measures of the two models for each data set in our five trials, $ave(W)$ is the average of W, $var(W)$ is the variance of W, and n is the number of data sets (five in our case). For $n = 5$, if t is greater than 4.604, then it is more than 99 percent statistically significant that PSTMM performs better.

All of our experiments were performed on a Linux machine with dual Intel Xeon 3.0 GHz processors and 8 GBytes of main memory.

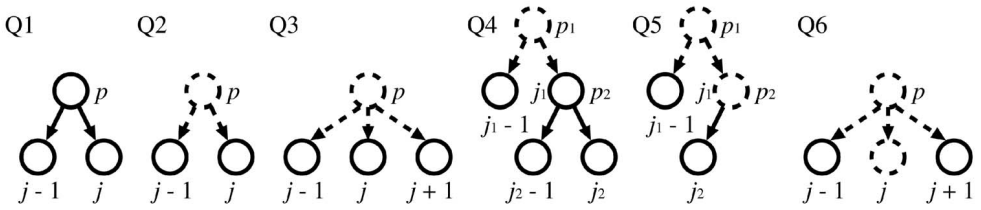

Fig. 7. Six patterns of tree fragments used in our experiments.

4.2.1 Synthetic Data Sets

In our experimental setting for synthetic data, each positive example was embedded with tree fragment patterns, and each negative example was simply randomly generated from the parent-child distribution of the positive examples. To embed such tree fragment patterns into each positive example, we first randomly generated five instances of a pattern with different labels. We then randomly generated tree structures and labels, where each tree contained 20 nodes, and then we randomly selected five nodes from each generated tree and embedded each of the generated patterns into each of the selected nodes. We used six types of tree fragment patterns in our experiments. For all of the tree examples tested, we fixed the parameters of the models in our experiments to the following:[5] $|\mathbf{T}| = 200$ for training (400 for test), $|S| = 10$, $|V_u| = 20$, $|\Sigma| = 10$, $|C| = 5$, and $Z = 10$ in MLM and MLPM. We claim that these models were tested under similar conditions since they approximately have the same number of parameters. That is, the sizes of the parameters in PSTMM and MPLM are equivalent in this setting; the sizes of the initial state probabilities π, the state transition probabilities a, and the label output probabilities b in PSTMM are 10, 1,000 $(= 10 \times 10 \times 10)$, and 100, respectively, and the sizes of the component selection probabilities v, the initial state probabilities π, and the label output probabilities w in MPLM are 10, 100, and 1,000, respectively.

HTMM and MLPM took less time than PSTMM to find the maximum likelihood estimation due to the lack of dependencies among the siblings in the trees. MLM is much simpler than PSTMM, HTMM, and MLPM, so the algorithms for MLM ran faster than those for the above three models. LM and LPM required the least amount of time since these models have no hidden variables, and estimation of these parameters is done by counting the number of labels in the given data set just once, while the other four models need to update their parameters iteratively.

From the time complexities of the algorithms, recall that all six of the models are scalable in terms of the size of the data set since any of their corresponding learning algorithms are able to update parameters in time linear to the size. However, we note that PSTMM is able to capture the most complex patterns, and its time complexity of $O(|\mathbf{T}| \cdot |S|^3 \cdot |V|)$ is within the well-known practical computational upper limit, set by such probabilistic models as SCFGs. Therefore, we claim that its expressive power significantly outweighs the increase in time complexity, especially considering its scalability.

The most important feature of our proposed model is that the dependency between siblings is considered. Five of our six patterns (abbreviated Q1 through Q6) contain two or more siblings, as illustrated in Fig. 7. In this figure, each label in a pattern is attached to a solid circle. Q1 is the most standard pattern, which contains a parent and two siblings. Q2 is a simpler pattern (subfragment of Q1), which contains only two siblings. We can thus check the performance of our model when the dependency of a child on its parent is removed. Q3, an extension of Q2, has three consecutive siblings, so we can check whether or not this consecutive pattern can be captured by PSTMM. Q4 is a combination of Q1 and Q2, where node j of Q2 is node p of Q1. Q5 is a pattern between a node and its immediately younger sibling's child. Finally, Q6 is a modification of Q3, where the label of node j of Q3 is set randomly. Both Q5 and Q6 contain distant nodes, so as to check whether or not such distant relationships in a tree can be captured by PSTMM.

Tables 4, 5, and 6 show AUC performance, prediction accuracies, and precisions (at recall of 0.3), respectively, for the six models tested on these six patterns. In each of these tables, *t*-values are added in parentheses. As shown in these tables, PSTMM outperformed all of the other methods at a statistically significant level. The results show that Q3 is the easiest pattern for which PSTMM can discriminate between positives and negatives, Q2 and Q1 are the second and the third easiest, respectively, and Q5 is the most difficult. In our experimental setting, the distribution of parent-child pairs is set the same for both positives and negatives and, thus, the results indicate that patterns consisting of siblings can be more easily captured by PSTMM. For patterns Q5 and Q6, each of which does not contain any direct dependencies (parent-child or between siblings), PSTMM achieved a high performance, and so we can claim that PSTMM can be applied to find such patterns consisting of indirect, long-range dependencies. The *t*-values for Q4 are the smallest among all the patterns, since Q4 (and Q1) contains parent-child relationships, making it the easiest pattern for HTMM, MLPM, and LPM, all of which attempt to capture parent-child pairs of labels.

4.2.2 Real Data Sets: Glycans

We further used data sets of glycans, which were all obtained from the KEGG GLYCAN database [23], [5]. Glycans are branched tree structures with various types of linkages. The basic component of glycans is the monosaccharide unit, of

5. Experiments using different parameter settings were also performed, particularly in changing the number of states used by PSTMM. Note that, in all cases we tested, the performance advantage advantage of PSTMM over other methods was consistent with what is presented in this paper.

TABLE 4
AUC and *t*-Values (in Parentheses) for Synthetic Data

Pattern	PSTMM	HTMM	MLPM	LPM	MLM	LM
Q1	**87.6**	56.9 (**13.0**)	71.8 (**12.5**)	50.7 (**40.0**)	49.9 (**64.0**)	57.6 (**40.9**)
Q2	**89.1**	51.4 (**24.4**)	48.5 (**32.8**)	58.7 (**27.6**)	52.6 (**39.0**)	53.9 (**33.9**)
Q3	**96.1**	53.0 (**55.0**)	51.2 (**75.2**)	58.4 (**36.8**)	55.6 (**37.8**)	56.3 (**31.2**)
Q4	**80.0**	48.9 (**10.3**)	58.9 (**6.2**)	60.3 (**6.5**)	49.9 (**9.7**)	54.2 (**8.9**)
Q5	**70.3**	50.5 (**13.1**)	51.8 (**16.1**)	50.0 (**18.3**)	49.9 (**21.7**)	47.1 (**16.2**)
Q6	**84.5**	50.3 (**48.4**)	49.1 (**21.5**)	51.5 (**39.1**)	49.9 (**29.6**)	49.8 (**27.4**)

TABLE 5
Prediction Accuracies and *t*-Values (in Parentheses) for Synthetic Data

Pattern	PSTMM	HTMM	MLPM	LPM	MLM	LM
Q1	**80.9**	58.1 (**14.9**)	66.8 (**10.7**)	52.9 (**30.4**)	50.0 (**35.0**)	58.3 (**21.8**)
Q2	**83.8**	53.5 (**30.0**)	51.2 (**47.5**)	58.0 (**32.7**)	58.6 (**26.9**)	55.9 (**31.6**)
Q3	**90.7**	54.3 (**29.9**)	53.2 (**43.4**)	58.7 (**37.4**)	60.7 (**35.0**)	56.5 (**25.9**)
Q4	**73.9**	52.2 (**8.1**)	58.1 (**5.97**)	59.5 (**4.86**)	50.0 (**8.73**)	56.6 (**6.27**)
Q5	**66.1**	52.5 (**16.5**)	53.3 (**10.4**)	52.0 (**15.1**)	50.0 (**18.5**)	52.2 (**12.9**)
Q6	**78.2**	52.3 (**36.2**)	51.4 (**18.7**)	52.9 (**30.2**)	50.0 (**25.1**)	53.3 (**20.1**)

TABLE 6
Precisions at Recall of 0.3 and *t*-Values (in Parentheses) for Synthetic Data

Pattern	PSTMM	HTMM	MLPM	LPM	MLM	LM
Q1	**99.2**	53.5 (**22.0**)	76.8 (**14.0**)	49.7 (**28.3**)	50.0 (**123.5**)	54.3 (**28.8**)
Q2	**95.3**	50.3 (**37.1**)	48.8 (**61.5**)	59.2 (**35.4**)	46.9 (**75.0**)	50.0 (**38.4**)
Q3	**99.0**	51.2 (**46.2**)	53.8 (**39.1**)	64.0 (**13.7**)	48.8 (**55.7**)	55.8 (**36.4**)
Q4	**87.3**	47.6 (**7.3**)	59.1 (**4.64**)	57.8 (**6.44**)	50.0 (**6.99**)	51.2 (**7.03**)
Q5	**75.7**	51.0 (**16.9**)	51.6 (**14.8**)	48.9 (**13.2**)	50.0 (**13.6**)	45.4 (**13.9**)
Q6	**88.2**	50.0 (**31.6**)	49.2 (**19.0**)	52.5 (**44.2**)	50.0 (**29.2**)	47.4 (**28.4**)

which a handful are most common in higher aminal oligosaccharides (See Table 7).

Glycans are classified based on their biological properties, and we selected four of the major classes, called "N-Glycans," "O-Glycans," "Glycosaminoglycans," and "Sphingolipids," as our data sets. For each class, we selected those structures that contained at least one sibling pair to be included in the data sets for our experiment. The parameters used in this experiment were as follows: $|\mathbf{T}| = 200$ for training,[6] $|S| = 10$, $|\Sigma| = 19$, $|C| = 5$, and $Z = 10$ in MLM and MLPM. Since the trees in our data sets vary in size, the likelihood for a given tree drastically changes depending on the number of nodes in the tree. In order to correct for this discrepancy, each probability parameter value was multiplied by its size. That is, for example, we multiplied $a[\{s_q, s_m\}, s_l]$ by $|S|$. We used these corrected parameter values to calculate the likelihood of each tree and, thus, used the corrected likelihood as the score for each tree to evaluate each model. Table 8 shows the AUC, prediction accuracies, and precisions (at recall of 0.3) for the six models tested on our data sets. *t*-values are added in parentheses in this table. The results show that PSTMM clearly outperformed the other five models tested in all cases by a statistically significant margin. The performance results obtained by PSTMM for N-Glycan are higher than those for O-Glycan most likely due to the larger tree sizes and increased number of sibling pairs. The HTMM performance on the Sphingolipid data set was rather high compared to those of the other classes using the same model. This may be attributed to the fact that the structures of Sphingolipids tend to be generally linear. In fact, more than half of the Sphingolipid structures in our database are strictly linear, meaning they do not contain even one sibling pair. Because we selected those structures containing at least one sibling pair, we may assume that the majority of

6. Two hundred examples were randomly chosen from the training blocks, and the test size depended on the data set.

TABLE 7
Common Monosaccharide Names, Their Abbreviations, and Their Symbols

Monosaccharide name	Abbr.	Sym.
Glucose	Glc	▲
Galactose	Gal	●
Mannose	Man	○
N-acetyl neuraminic or sialic acid	NeuNAc	◆
N-acetylglucosamine	GlcNAc	■
N-acetylgalactosamine	GalNAc	□
Fucose	Fuc	△
Xylose	Xyl	▽
Glucuronic acid	GlcA	◆
Iduronic acid	IdA	◆

TABLE 8
Experimental Results Listing Prediction Accuracy, Precision, and AUC Performance for Glycan Data Sets: (a) N-Glycans, (b) O-Glycans, (c) Glycosaminoglycans, and (d) Sphingolipidsa

		PSTMM	HTMM	MLPM	LPM	MLM	LM
a	AUC	**92.0**	66.7 (**29.7**)	67.8 (**28.5**)	55.1 (**45.5**)	58.9 (**38.2**)	51.2 (**57.8**)
	Acc.	**85.5**	64.4 (**26.0**)	64.5 (**22.5**)	55.4 (**33.6**)	58.1 (**31.6**)	52.9 (**39.2**)
	Prec.	**95.6**	68.4 (**22.5**)	66.8 (**29.2**)	55.7 (**39.8**)	58.2 (**46.3**)	50.8 (**62.2**)
b	AUC	**80.1**	62.4 (**11.9**)	64.9 (**11.4**)	54.9 (**24.4**)	51.4 (**29.4**)	49.4 (**31.8**)
	Acc.	**75.3**	62.1 (**11.1**)	63.8 (**10.5**)	57.1 (**19.2**)	54.7 (**24.5**)	53.4 (**24.6**)
	Prec.	**84.1**	61.9 (**11.4**)	62.7 (**13.5**)	55.0 (**20.1**)	51.3 (**23.2**)	49.5 (**25.6**)
c	AUC	**91.9**	50.7 (**23.1**)	69.6 (**10.3**)	48.7 (**24.0**)	56.7 (**19.7**)	50.7 (**23.8**)
	Acc.	**86.4**	55.7 (**22.6**)	67.2 (**11.3**)	53.7 (**23.6**)	57.6 (**20.3**)	54.4 (**22.6**)
	Prec.	**96.3**	54.6 (**18.5**)	72.4 (**9.6**)	48.9 (**24.4**)	60.2 (**18.1**)	53.7 (**28.1**)
d	AUC	**88.3**	71.2 (**11.6**)	65.1 (**14.3**)	59.0 (**28.0**)	55.2 (**25.9**)	49.4 (**39.8**)
	Acc.	**83.1**	70.1 (**10.2**)	65.0 (**12.8**)	61.7 (**19.0**)	57.6 (**20.0**)	54.1 (**26.2**)
	Prec.	**92.9**	73.0 (**10.5**)	64.1 (**14.5**)	61.3 (**19.3**)	56.6 (**20.2**)	51.2 (**29.8**)

The PSTMM performance is statistically significant in all classes, implying that complex, sibling-dependent patterns indeed exist in these glycan classes.

those selected had exactly one, thus explaining the higher performance of the structures in this class using the HTMM model compared to other classes. Of course, PSTMM still outperforms HTMM in this class by a significant margin. As for the Glycosaminoglycans, these structures also tend to be highly sulfatized linear structures, where for the majority, only sulfates branch off the main "backbone" by single edges, and only a few structures contain siblings, which tend to exist off the root node. Thus, it is not surprising that HTMM performs along the lines of the LM and MLM models. PSTMM, on the other hand, was able to capture some dependency in Glycosaminoglycans because of the siblings at the root; we may assume that there is some dependency existing across the breadth of these structures that could not be captured whatsoever by HTMM. Fig. 8 includes the ROC curves for N-Glycans and O-Glycans, clearly illustrating the significant performance increase of PSTMM over the other models. Overall, these results are almost equal to those obtained from our synthetic data sets, implying that there must be some complex pattern that is not limited to parent-child (or, more generally, ancestor-descendant) relationships, in the real data sets of glycans. Furthermore, the extremely high performance of PSTMM over all classes of glycans clearly illustrate the importance of being able to capture more complex relationships in biological data.

4.3 Most Likely State Transition for New Patterns and Multiple Tree Alignment

The next step was to analyze the probabilities of the states learned from our data sets and to find the most likely state transitions. By doing so, we would be able to find common patterns in the data sets as well as to perform so-called multiple tree structure alignment. Fig. 9 illustrates three N-Glycan tree structures that PSTMM found to have similar patterns. The state transition model learned from these structures is given below each glycan structure. In this figure, we can see that we would find ambiguities in aligning these structures simply by looking at the monosaccharides. By using the states learned by PSTMM, however, the multiple tree alignment is clear in that there is only one way to match the branches to one another. More specifically, we notice that the three mannoses in the center of all three structures each are assigned different states. Clearly, the parent-sibling relationship has been captured in this known core-structure of N-Glycans. Second, despite the fact that each of the child mannoses have similar children and grandchildren (i.e., the same □ → • pair in sequence),

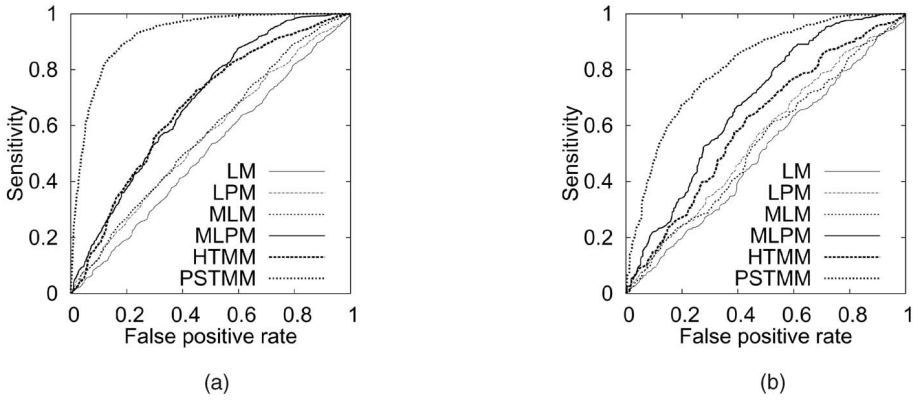

Fig. 8. ROC curves for (a) N-Glycans and (b) O-Glycans.

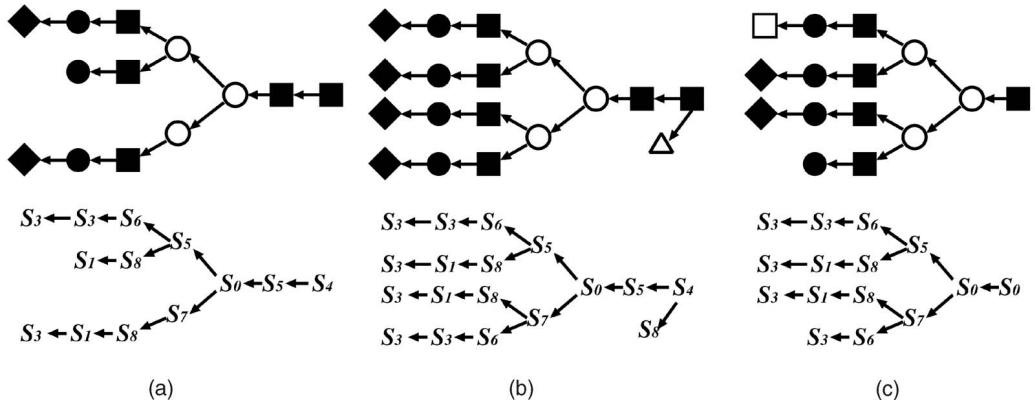

Fig. 9. Example of a glycan tree alignment of three similar glycan structures. Note how the leaves are exactly alike according to monosaccharide chains, but their state paths differ to enable them to be distinctly aligned.

no two state sequences originates from the same parent. Thus, the more complex pattern across siblings has been captured. One may regard an entire glycan structure as one unique pattern considering their compact sizes.

It was interesting to run this prediction procedure another time to see how different state transitions were learned. For example, Figs. 10, 11, and 12 come from a different prediction run. Looking at the state transitions of these figures, we see the pattern of $S_4 - S_2 - S_2$ at the root portions of the trees (on the right-hand side) in Fig. 10. A similar pattern can be seen in the trees in Fig. 11 except with an extra S_9 for the Fucose. At first glance, one would assume those in Fig. 12 would also be the same as in Fig. 11, but, in fact, the former turns out to contain a completely different state path of $S_0 - S_5 - S_4$ plus an S_8. From an overall perspective, we can see how this comes about, and a biological explanation even exists. The three types of patterns on the right-hand side of these structures actually correspond to known vertebrate N-Glycan diversification types. There are three N-Glycan subtypes: high-mannose, hybrid, and complex [37], which are defined by the patterns across the leaves of the structures. The high-mannose N-Glycan subtype is characterized by groups of mannoses clustered together, the hybrid N-Glycan subtype has a combination of both triple-mannose groups and GlcNAcs, and the complex N-Glycan type has GlcNAcs linked to the triple-mannose core. Amazingly, PSTMM was able to capture these three types despite the fact that all three have the same core structures. That is, we see again that the $\square \rightarrow \bullet$ sequence appears in some of these structures, and this time, they are assigned one of the $S_3 - S_6$ or $S_1 - S_2$ state paths. In this run, however, a complex pattern concerning the large number of mannoses has been found. We see that, in a consecutive sequence of mannoses, if a mannose has no mannose child and has a younger sibling, it will be assigned S_5. Otherwise, it will be S_4 and the mannose child S_0. There may be further complex rules associated with these state transition paths estimated in this run, which may be an interesting route for future research. One may then deduce that other patterns which have yet to be discovered may be found by our model.

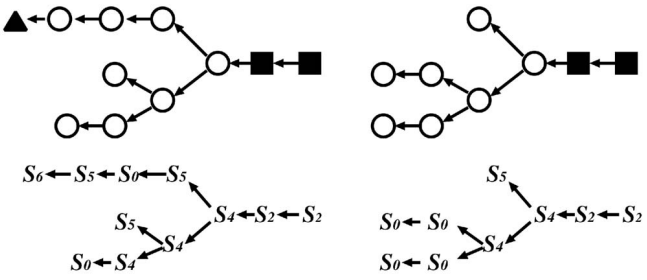

Fig. 10. High-mannose N-Glycan type where groups of mannoses cluster together.

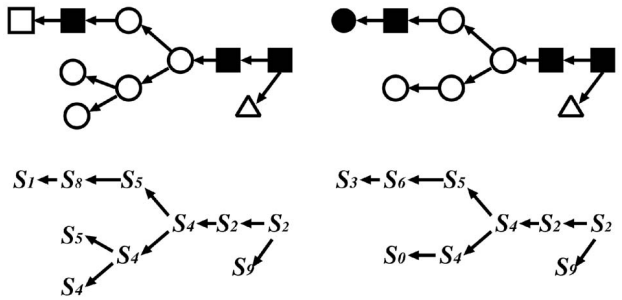

Fig. 11. Hybrid N-Glycan type where both triple-mannose groups and GlcNAcs exist.

5 Concluding Remarks

We have proposed a probabilistic model, called PSTMM, and its effective learning algorithm for mining labeled

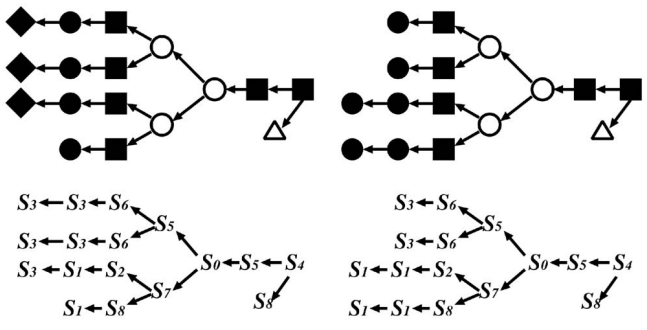

Fig. 12. Complex N-Glycan type where GlcNAcs are linked to the triple-mannose core.

ordered trees, and we have experimentally shown the effectiveness of our approach by using a variety of experiments using real-world glycan data sets. We emphasize again that our learning algorithm has a very nice complexity order, which is right within a practically well-known computational limit. We also note that, due to the nature of the EM algorithms, it is straightforward to modify our learning algorithm for special data sets such as large-scale data sets and data streams. Details of these modified algorithms are shown in Supplementary Information 1 in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tkde/archives.htm). Possible future work of this approach is to apply PSTMM to align a variety of multiple glycans in a database and to find biologically new patterns embedded in them.

The performance of PSTMM was evaluated in a supervised learning manner in our experiments, but only positive examples were given for training. The incorporation of both positive and negative examples as training examples to discriminate positives from negatives is a general supervised learning task, popular, and important in machine learning. The recent high-performance supervised learning method comes in the form of kernel methods, such as support vector machines. A kernel is a distance measure between two examples, and an important theme in current machine learning research is to develop a kernel which is best suited to the given examples. For example, a graph kernel capable of computing the distance between two given input graphs is one important and popular problem in which the performance of kernels is being investigated [19], [25], [28], [21]. It would thus be interesting to develop a kernel for labeled ordered trees based on PSTMM, as was done for strings based on HMMs [22], [34].

ACKNOWLEDGEMENTS

The authors would like to thank anonymous referees for useful comments and suggestions. Nobuhisa Ueda thanks Taisuke Sato, Taku Yoshioka, and Kenichi Kurihara for valuable discussions. This work is supported in part by Grant-in-Aid for Scientific Research on Priority Areas (C) "Genome Information Science" and Grant-in-Aid for Encouragement of Young Scientists from the Ministry of Education, Culture, Sports, Science, and Technology of Japan. Nobuhisa Ueda and Kiyoko F. Aoki-Kinoshita equally contributed to this work.

REFERENCES

[1] N. Abe and H. Mamitsuka, "Predicting Protein Secondary Structure Using Stochastic Tree Grammars," *Machine Learning,* vol. 29, pp. 275-301, 1997.

[2] S. Abitebould, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 2000.

[3] K.F. Aoki, N. Ueda, A. Yamaguchi, T. Akutsu, M. Kanehisa, and H. Mamitsuka, "Managing and Analyzing Carbohydrate Data," *ACM SIGMOD Record,* vol. 33, no. 2, 2004.

[4] K.F. Aoki, N. Ueda, A. Yamaguchi, M. Kanehisa, T. Akutsu, and H. Mamitsuka, "Application of a New Probabilistic Model for Recognizing Complex Patterns in Glycans," *Proc. 12th Int'l Conf. Intelligent Systems for Molecular Biology,* 2004.

[5] K.F. Aoki, A. Yamaguchi, N. Ueda, T. Akutsu, H. Mamitsuka, S. Goto, and M. Kanehisa, "KCaM (KEGG Carbohydrate Matcher): A Software Tool for Analyzing the Structures of Carbohydrate Sugar Chains," *Nucleic Acids Research,* vol. 32, (Web Server issue), pp. W267-W272, 2004.

[6] T. Asai, H. Arimura, K. Abe, S. Kawasoe, and S. Arikawa, "Online Algorithms for Mining Semistructured Data Streams," *Proc. Second Int'l Conf. Data Mining,* pp. 158-174, 2002.

[7] J.K. Baker, "Trainable Grammars for Speech Recognition," *Speech Comm. Papers Presented at the 97th Meeting of ASA,* pp. 547-550, 1979.

[8] L. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Infinite State Markov Chains," *Annals of Math. Statistics,* vol. 37, no. 6, pp. 1554-1563, 1966.

[9] C. Bertozzi and L. Keissling, "Chemical Glycobiology," *Science,* vol. 291, pp. 2357-2364, 2001.

[10] S. Brooks, M. Dwek, and U. Schumacher, "Functional and Molecular Glycobiology," *BIOS Scientific,* 2002.

[11] I.V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Model-Based Clustering and Visualization of Navigation Patterns on a Web Site," *Data Mining and Knowledge Discovery,* vol. 7, no. 4, pp. 399-424, 2003.

[12] G. Chang, G. Patel, L. Relihan, and J.T.L. Wang, "A Graphical Environment for Change Detection in Structured Documents," *Proc. 21st Int'l Computer Software and Applications Conf.,* pp. 536-541, 1997.

[13] G. Cong, L. Yi, B. Liu, and K. Wang, "Discovering Frequent Substructures from Hierarchical Semistructured Data," *Proc. Second SIAM Int'l Conf. Data Mining (SDM '02),* 2002.

[14] M.S. Crouse, R.D. Nowak, and R.G. Baraniuk, "Wavelet-Based Statistical Signal Processing Using Hidden Markov Models," *IEEE Trans. Signal Processing,* vol. 46, no. 4, pp. 886-902, 1998.

[15] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Series B,* vol. 39, pp. 1-38, 1977.

[16] M. Diligenti, P. Frasconi, and M. Gori, "Hidden Tree Markov Models for Document Image Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 4, pp. 519-523, 2003.

[17] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge Univ. Press, 1998.

[18] S. Fine, Y. Singer, and N. Tishby, "Hierarchical Hidden Markov Model: Analysis and Applications," *Machine Learning,* vol. 32, no. 1, pp. 41-62, 1998.

[19] T. Gärtner, P.A. Flach, and S. Wrobel, "On Graph Kernels: Hardness Results and Efficient Alternatives," *Proc. 16th Ann. Conf. Learning Theory,* pp. 129-143, 2003.

[20] D.J. Hand and R.J. Till, "A Simple Generalisation of the Area under the ROC Curve for Multiple Class Classification Problems," *Machine Learning,* vol. 45, pp. 171-186, 2001.

[21] T. Horvath, T. Gartner, and S. Wrobel, "Cyclic Pattern Kernels for Predictive Graph Mining," *Proc. 10th ACM SIGKDD,* pp. 158-167, 2004.

[22] T.S. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers," *Proc. Conf. Neural Information Processing Systems (NIPS 11),* pp. 487-493, 1999.

[23] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori, "The KEGG Resource for Deciphering the Genome," *Nucleic Acids Research,* vol. 32, pp. D277-D280, 2004.

[24] H. Kashima and T. Koyanagi, "Kernels for Semistructured Data," *Proc. 19th Int'l Conf. Machine Learning,* pp. 411-417, 2002.

[25] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized Kernels between Labeled Graphs," *Proc. 20th Int'l Conf. Machine Learning,* pp. 321-328, 2003.

[26] K. Lari and S.J. Young, "The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm," *Computer Speech and Language,* vol. 4, pp. 35-56, 1990.

[27] S.L. Lauritzen and D.J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems (with Discussion)," *J. Royal Statistical Soc., Series B,* vol. 50, no. 2, pp. 157-224, 1988.

[28] P. Mahe, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, "Extensions of Marginalized Graph Kernels," *Proc. 21st Int'l Conf. Machine Learning,* pp. 552-559, 2004.

[29] G.J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions.* Wiley-Interscience, 1996.

[30] G.J. McLachlan and D. Peel, *Finite Mixture Models.* John Wiley & Sons, 2000.
[31] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, 1988.
[32] L. Rabiner and S. Juang, *Fundamentals of Speech Recognition.* Prentice Hall, 1993.
[33] A. Termier, M.C. Rousset, and M. Sebag, "Treefinder: A First Step towards XML Data Mining," *Proc. Second IEEE Int'l Conf. Data Mining,* pp. 450-457, 2002.
[34] K. Tsuda, T. Kin, and K. Asai, "Marginalized Kernels for Biological Sequences," *Bioinformatics,* vol. 18 (Supplement 1), pp. 268-275, 2002.
[35] N. Ueda, K.F. Aoki, and H. Mamitsuka, "A General Probabilistic Framework for Mining Labeled Ordered Trees," *Proc. SIAM Int'l Conf. Data Mining,* pp. 357-368, 2004.
[36] N. Ueda and T. Sato, "Simplified Training Algorithms for Hierarchical Hidden Markov Models," *Proc. Fourth Int'l Conf. Discovery Science,* pp. 401-415, 2001.
[37] A. Varki, R. Cummings, J. Esko, and J. Freeze, *Essentials of Glycobiology.* Cold Spring Harbor Laboratory Press, 1999.
[38] M. Zaki and C. Aggarwal, "An Effective Structural Classifier for XML Data," *Proc. Ninth ACM SIGKDD,* pp. 316-325, 2003.

Nobuhisa Ueda received the BEng, MEng, and PhD degrees from the Tokyo Institute of Technology, Japan, in 1996, 1998, and 2002, respectively. He is currently a research associate at the Bioinformatics Center, Institute for Chemical Research, Kyoto University. His main research interests include machine learning, probabilistic modeling, and bioinformatics.

Kiyoko F. Aoki-Kinoshita, PhD, received the BS and MS degrees in computer science from Northwestern University in 1996, followed by her doctorate in computer engineering from Northwestern University in December, 1999. She is currently a research associate at the Bioinformatics Center of Kyoto University's Institute for Chemical Research, where her interests are in chemo/bioinformatics, carbohydrate sugar chains (glycans), and probabilistic models. She is a member of the IEEE.

Atsuko Yamaguchi received the BS degree in 1991 from Kyoto University, the MS degree in information systems from Kyushu University in 1993, and the PhD degree in information science from Kyoto University in 2004. She is a research associate at the Bioinformatics Center of Kyoto University. Her professional interests include chemoinformatics and graph algorithms.

Tatsuya Akutsu received the BEng and MEng degrees in aeronautics and the DEng degree in information engineering from the University of Tokyo, 1984, 1986, and 1989, respectively. From 1989 to 1994, he was with the Mechanical Engineering Laboratory. From 1994 to 1996, he was an associate professor in the Department of Computer Science at Gunma University. From 1996 to 2001, he was an associate professor in the Human Genome Center, Institute of Medical Science, University of Tokyo. Since 2001, he has been a professor in the Bioinformatics Center, Institute for Chemical Research, Kyoto University. His research interests include bioinformatics and the design and analysis of algorithms.

Hiroshi Mamitsuka received the BS degree in biochemistry and biophysics, the ME degree in information engineering, and the PhD degree in information sciences from the University of Tokyo in 1988, 1991, and 1999, respectively. He is an associate professor at the Institute for Chemical Research, Kyoto University. He worked with NEC Research Laboratories for 11 years beginning in 1991. He became a faculty member of Kyoto University in 2002. His research interests include bioinformatics, machine learning, and data mining. He is a member of the IEEE and the IEEE Computer Society.