A Spectral Algorithm for Latent Tree Graphical Models

Ankur P. Parikh APPARIKH@CS.CMU.EDU
Le Song LESONG@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

Latent variable models are powerful tools for probabilistic modeling, and have been successfully applied to various domains, such as speech analysis and bioinformatics. However, parameter learning algorithms for latent variable models have predominantly relied on local search heuristics such as expectation maximization (EM). We propose a fast, local-minimum-free spectral algorithm for learning latent variable models with arbitrary tree topologies, and show that the joint distribution of the observed variables can be reconstructed from the marginals of triples of observed variables irrespective of the maximum degree of the tree. We demonstrate the performance of our spectral algorithm on synthetic and real datasets; for large training sizes, our algorithm performs comparable to or better than EM while being orders of magnitude faster.

1 Introduction

Latent variable models usually refer to probabilistic graphical models that relate a set of observed variables to an additional set of unobserved or hidden variables. Introducing latent variables can greatly improve the flexibility of probabilistic modeling, allowing it to address a diverse range of problems with hidden factors such as in document analysis (Blei et al., 2002), social network modeling (Hoff et al., 2002), speech recognition (Rabiner & Juang, 1986) and bioinformatics (Clark, 1990). Latent variables can also lead to significant savings in model parametrization. By defining a joint model over observed and latent variables, the marginal distribution of the observed variables is obtained by integrating out the latent ones. This allows complex distributions over observed variables (*e.g.*, clique models) to be expressed in terms of more tractable joint models (*e.g.*, tree models) over the augmented variable space.

Although latent variable models are very flexible and can be represented in a compact way, learning the model parameters has predominantly relied on likelihood maximization and local search heuristics such as expectation maximization (EM) (Dempster et al., 1977). Besides the problem of local minima, EM can require many iterations to reach a prescribed training precision, and high dimensional problems can dramatically slow down EM.

While EM tries to recover the full set of parameters in latent variable models, in many applications it is the inference task that is most interesting. For instance, in speech classification, we are interested in estimating the likelihood of a test sequence under different models; in quantitative finance, we are interested in predicting the price of one stock given the prices of other stocks; or in biological analysis, we are interested in forecasting the expression of one gene given perturbations to other genes. In all these examples, the inference task involves estimating either the joint or conditional distribution of a set of observed variables. Ideally, we want to avoid explicitly recovering the parameters related to latent variables (which leads to non-convex problems), and proceed directly to the interested quantities.

Recently, Hsu et al. (2009) proposed a spectral algorithm for learning hidden Markov models (HMM) which directly estimates the joint distribution of the observed variables without recovering the HMM model parameters. The major computation of the algorithms involves a singular value decomposition (SVD) of small marginal probability matrices involving pairs of observed variables. Compared to EM, this spectral algorithm does not have the problem of local optima, and one can formally study its statistical properties. However, this spectral algorithm is specific to HMMs, and it is not clear whether their techniques can be extend to latent variable models with other topologies.

Mossel & Roch (2006) also proposed a spectral algorithm for latent variable models which applies to arbitrary tree topologies, but they made very restrictive assumptions: all variables (observed and latent) have exactly the same number of states, and all conditional probability tables (CPT) are invertible. Under these conditions, they derived a spectral algorithm that can explicitly recovers all CPTs from marginals of triples of observed variables. In many applications, however, latent variables can represent factors sim-

Appearing in *Proceedings of the 28^{th} International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

pler than the noisy observations, and the number of hidden states can be smaller than that of the observed states. In these cases, the CPTs are no longer invertible, which renders this spectral algorithm no longer applicable.

In this paper, we propose a novel spectral algorithm for latent variable models with arbitrary tree topologies where the number of hidden states is smaller than or equal to that of the observed states. Instead of first explicitly learning the model parameters and then performing inference, we directly compute the joint distribution of the observed variables without explicitly recovering the model parameters.

We first express the joint distribution of the observed variables using 3rd order tensors, and then show that the components in this tensor representation can be reconstructed from the marginals of triples of observed variables. Given a finite number of samples, our spectral algorithm estimates the desired joint distributions by performing singular value decompositions on a collection of small marginal probability matrices, and hence is very efficient. In addition to estimating the joint distribution, our method can also recover the marginal of any set of observed variables. We conducted experiments on both synthetic and real world data, and demonstrated the competitive performance of our algorithm to EM. For large training sizes, our algorithm performs comparably or better than EM while being orders of magnitude faster.

2 Tensor Algebra

We first give a brief introduction to tensor algebra (for more details, see Kolda & Bader (2009)). A tensor is a multidimensional array, and its order is the number of dimensions, also known as modes. In this paper, vectors (tensors of order one) are denoted by boldface lowercase letters, *e.g.*, $\boldsymbol{a}$. Matrices (tensors of order two) are denoted by boldface capital letters, *e.g.*, $\boldsymbol{A}$. Higher-order tensors (order three or higher) are denoted by boldface caligraphic letters, *e.g.*, $\boldsymbol{\mathcal{T}}$. Scalars are denoted by lowercase letters, *e.g.*, a.

Subarrays of a tensor are formed when a subset of the indices is fixed. Particularly, a fiber is defined by fixing every index but one. Fibers are the higher-order analogue of matrix rows and columns. A colon is used to indicate all elements of a mode. Thus, the jth column of a matrix $\boldsymbol{A}$ is $\boldsymbol{A}(:,j)$, and the ith row of $\boldsymbol{A}$ is $\boldsymbol{A}(i,:)$. Analogously, the mode-n fiber of a Nth order tensor $\boldsymbol{\mathcal{T}}$ is then denoted as $\boldsymbol{\mathcal{T}}(i_1, i_2, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$. Fibers can be used to construct higher order tensors from lower order ones. For instance, a third order tensor $\boldsymbol{\mathcal{A}}$ which is diagonal in mode-2 and 3 can be constructed from a matrix $\boldsymbol{B}$ by setting $\boldsymbol{\mathcal{T}}(:,i,i) = \boldsymbol{B}(:,i)$.

Tensors can be multiplied together. For matrices and vectors, we will use standard notation for their multiplications, *e.g.*, $\boldsymbol{Ba}$ and $\boldsymbol{AB}$. For tensors of higher order, we are particularly interested in multiplying a tensor by matrices and vectors. The n-mode matrix product is the multiplication of a tensor with a matrix in mode n of the tensor. Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an Nth order tensor and $\boldsymbol{A} \in \mathbb{R}^{J \times I_n}$ be a matrix. Then

$$\boldsymbol{\mathcal{T}}' = \boldsymbol{\mathcal{T}} \times_n \boldsymbol{A} \;\in\; \mathbb{R}^{I_1 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N}, \qquad (1)$$

where the entries $\boldsymbol{\mathcal{T}}'(i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N)$ are defined as $\sum_{i_n=1}^{I_n} \boldsymbol{\mathcal{T}}(i_1, \ldots, i_n, \ldots, i_N)\boldsymbol{A}(j, i_n)$. We will further introduce two useful properties of n-mode matrix product. First, for distinct modes in a series of multiplications, the order of the multiplication can be exchanged

$$\boldsymbol{\mathcal{T}} \times_n \boldsymbol{A} \times_m \boldsymbol{B} = \boldsymbol{\mathcal{T}} \times_m \boldsymbol{B} \times_n \boldsymbol{A} \quad (m \neq n). \qquad (2)$$

Second, the matrices can be combined first, if the modes in a series of multiplications are the same

$$\boldsymbol{\mathcal{T}} \times_n \boldsymbol{A} \times_n \boldsymbol{B} = \boldsymbol{\mathcal{T}} \times_n (\boldsymbol{BA}). \qquad (3)$$

We note that n-mode matrix product does not change the order of a tensor, but the size of the tensor may change.

Multiplication of a tensor with a vector in mode n of the tensor is called n-mode vector product. Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and $\boldsymbol{a} \in \mathbb{R}^{I_n}$. Then

$$\boldsymbol{\mathcal{T}}' = \boldsymbol{\mathcal{T}} \,\bar{\times}_n\, \boldsymbol{b} \;\in\; \mathbb{R}^{I_1 \times \ldots I_{n-1} I_{n+1} \times \ldots \times I_N} \qquad (4)$$

where the entries $\boldsymbol{\mathcal{T}}'(i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N)$ is defined as $\sum_{i_n=1}^{I_n} \boldsymbol{\mathcal{T}}(i_1, i_2, \ldots, i_n, \ldots, i_N)\boldsymbol{a}(i_n)$. We note that n-mode vector product actually reduces the order of the tensor, *i.e.*, $\boldsymbol{\mathcal{T}}'$ is order $N-1$ if $\boldsymbol{\mathcal{T}}$ is order N. Using n-mode vector product, we can turn a diagonal operation on vector-matrix product into tensor multiplications, *i.e.*,

$$\operatorname{diag}(\boldsymbol{a}^\top \boldsymbol{B}) = \boldsymbol{\mathcal{T}} \,\bar{\times}_1\, \boldsymbol{a}, \;\text{ where } \boldsymbol{\mathcal{T}}(:,i,i) = \boldsymbol{B}(:,i). \qquad (5)$$

3 Latent Tree Graphical Models (LTMs)

In this paper, we will focus on discrete latent variable models where the conditional independence structures are specified by a tree. Furthermore, we follow the convention that uppercase letters denote random variables (*e.g.*, X_i) and lowercase letters their instantiations (*e.g.*, x_i). A latent tree model defines a joint probability distribution over a set of O observed variables $\mathscr{O} = \{X_1, \ldots, X_O\}$ and a set of H hidden variables $\mathscr{H} = \{X_{O+1}, \ldots, X_{O+H}\}$. For simplicity, we assume that all observed variables have S_O states and all hidden variables have S_H states, and $S_O \geq S_H$. The complete set of variables is denoted by $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$.

The joint distribution of $\mathscr{X}$ in a latent tree model is fully characterized by a set of conditional probability tables (CPTs). More specifically, we can select an arbitrary (observed or latent) node in the tree as the root, and sort the nodes in the tree in topological order. Then the set of CPTs between nodes and their parents $\mathbb{P}[X_i | X_{\pi_i}]$ are sufficient to characterize the joint distribution (the root node X_r has no parent, *i.e.*, $\mathbb{P}[X_r | X_{\pi_r}] = \mathbb{P}[X_r]$),

$$\mathbb{P}[x_1, \ldots, x_{O+H}] = \prod_{i=1}^{O+H} \mathbb{P}[x_i | x_{\pi_i}]. \qquad (6)$$

Compared to tree models which are defined solely on ob-

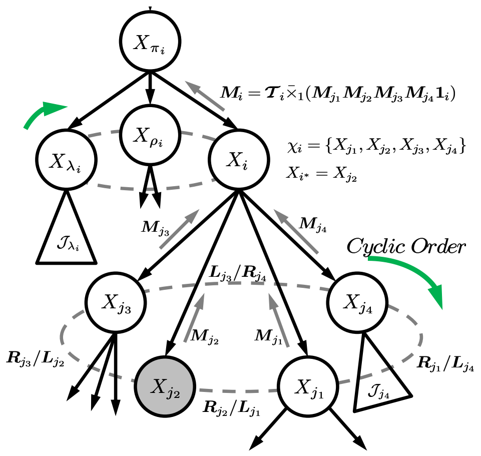


Figure 1. Notation for latent tree models. After rooting the tree and sorting the nodes in topological order, we denote the parent of a node X_i as X_{π_i}, and the set of children of X_i as χ_i. We order the sibling of X_i in a clockwise cyclic order, such that the left (next) sibling of X_i is denoted as X_{λ_i}, and the right (previous) sibling as X_{ρ_i}. We denote the subtree induced by X_i and its descendants as $\mathscr{T}_i$, and an observed variable in $\mathscr{T}_i$ as X_{i^*}. We note that there may be multiple observed variables in $\mathscr{T}_i$, and we will use X_{i^*} to refer to either of them. Similarly, we will also use $X_{\lambda_i^*}$ and $X_{\rho_i^*}$ to denote observed variables at subtrees rooted at X_{λ_i} and X_{ρ_i} respectively. Last, we use shaded nodes to denote observed variables, and un-shaded ones for hidden variables.

served variables (*e.g.*, models obtained from Chow & Liu (1968) algorithm), latent tree models encompass a much larger classes of models, allowing more flexibility in modeling observed variables. This is evident if we compute the marginal distribution of the observed variables by summing out the latent ones,

$$\mathbb{P}[x_1, \ldots, x_O] = \sum_{x_{O+1}} \cdots \sum_{x_{O+H}} \prod_{i=1}^{O+H} \mathbb{P}[x_i | x_{\pi_i}]. \quad (7)$$

This expression leads to complicated conditional independence structures between observed variables depending on the tree topology. In other words, latent tree models allow complex distributions over observed variables (*e.g.*, clique models) to be expressed in terms of more tractable joint models over the augmented variable space. This is a significant saving in model parametrization. We also note that for latent tree models, observed variables can be internal nodes as well as leaf nodes, allowing diverse structures, such as cliques connected by trees, for observed variables. Other notation related to the topological ordering of the nodes in a latent tree model are illustrated in Figure 1.

4 Tensor Representation for LTMs

The computation of the marginal distribution of the observed variables in (7) can be expressed in terms of tensor multiplications. Basically, the information contained in each tensor will correspond to the information in a conditional probability table (CPT) of the model and the tensor multiplications implement the summations. However, there are multiple ways of rewriting (7) using tensor notation, and not all of them provide intuition or easy derivation to a spectral algorithm. In this section, we will derive a specific representation of latent tree models which requires only tensors up to 3rd order and provides us a basis for deriving a spectral algorithm. For simplicity, we assume that all internal nodes of the tree correspond to latent variables and leaf nodes correspond to observed variables. The general case where observed variables can appear as both internal and leaf nodes can be found in the supplementary.

Root. We associate the root node X_r with the marginal probability vector $\boldsymbol{r} = \mathbb{P}[X_r]$ of X_r. Here we use $\mathbb{P}[X_r]$ to denote a vector where its kth dimension is defined as $\mathbb{P}[X_r = k]$ and k ranges over all possible assignments of X_r. Similarly, we use $\mathbb{P}[X_i, X_j]$, $\mathbb{P}[X_i | X_j]$, and $\mathbb{P}[X_i, X_j, X_k]$ to denote the joint probability matrix, conditional probability matrix and joint probability tensor respectively, and we denote $\mathbb{P}[X_i, x_j, X_k]$ as a slice (or a fiber) of the tensor when the middle variable is fixed to x_j.

Internal nodes. We associate each internal node X_i with a 3rd order tensor $\mathcal{T}_i$ related to the conditional probability matrix between X_i and its parent X_{π_i}. This tensor is diagonal in its 2nd and 3rd mode, and hence its nonzero entries can be accessed by two indices k and l. Furthermore, $\mathcal{T}_i(k, l, l) = \mathbb{P}[X_i = k | X_{\pi_i} = l]$. The reason for defining this tensor is to implement the marginalization operation over variable X_i using tensor vector multiplications, and return the result as a diagonal matrix. Let $\boldsymbol{v} = \mathbb{P}[x_j | X_i]$ be a likelihood vector. Then the mode-1 vector product, $\mathcal{T}_i \bar{\times}_1 \boldsymbol{v}$, results in a diagonal matrix with nonzero entries $\boldsymbol{M}_i(l, l) = \sum_k \mathbb{P}[x_j | X_i = k] \mathbb{P}[X_i = k | X_{\pi_i} = l]$ (or $\mathbb{P}[x_j | X_{\pi_i} = l]$).

Leaf nodes. We associate each leaf node x_i, which is always observed, with a diagonal matrix $\boldsymbol{M}_i$ related to the likelihood of x_i, *i.e.*, $\boldsymbol{M}_i(l, l) = \mathbb{P}[x_i | X_{\pi_i} = l]$. Furthermore, we let this $\boldsymbol{M}_i$ be the messages passed from the leaf nodes to their parents. We can show that the marginal probability of the leaf nodes (equation (7)) can be computed recursively using a message passing algorithm (Pearl, 1988): each node in the tree sends a message to its parent according to the reverse topological order of the nodes, and the final messages are aggregated in the root to yield the desired quantity.

Message updates. The outgoing message from an internal node X_i to its parent can be computed as (also see Figure 1)

$$\boldsymbol{M}_i = \mathcal{T}_i \bar{\times}_1 (\boldsymbol{M}_{j_1} \boldsymbol{M}_{j_2} \ldots \boldsymbol{M}_{j_J} \mathbf{1}_i) \quad (8)$$

where each $\boldsymbol{M}_j$ (a diagonal matrix) is an incoming message from a child, and $j_1, j_2, \ldots, j_J \in \chi_i$ range over all children of X_i ($J = |\chi_i|$). The $\mathbf{1}_i$ is a vector of all ones with suitable size, and it is used to reduce the incoming messages (all are diagonal matrices) to a single vector. The computation in (8) essentially implements the message update we often see in ordinary message passing algorithm (Pearl, 1988),

$$\boldsymbol{m}_i(x_{\pi_i}) = \sum\nolimits_{x_i} \mathbb{P}[x_i|x_{\pi_i}]\boldsymbol{m}_{j_1}(x_i)\ldots\boldsymbol{m}_{j_J}(x_i), \quad (9)$$

where $\boldsymbol{m}_j(x_i)$ represents incoming messages to X_i. The $\boldsymbol{M}_{j_1}\boldsymbol{M}_{j_2}\ldots\boldsymbol{M}_{j_J}\ \boldsymbol{1}_i$ corresponds to aggregating all incoming messages $\boldsymbol{m}_{j_1}(x_i)\ldots\boldsymbol{m}_{j_J}(x_i)$, and the $\boldsymbol{\mathcal{T}}_i\ \bar{\times}_1 *$ corresponds to the summation $\sum_{x_i}\mathbb{P}[x_i|x_{\pi_i}]\ *$. The characteristic feature of our update in (8) is that we use 3rd tensors to ensure that given incoming messages as diagonal matrices, the outgoing message is also a diagonal matrix, such that message aggregation can be carried on recursively.

Marginal probability. At root node, all incoming messages are combined to yield the final joint probability,

$$\mathbb{P}[x_1,\ldots,x_O] = \boldsymbol{r}^\top\left(\boldsymbol{M}_{j_1}\boldsymbol{M}_{j_2}\ldots\boldsymbol{M}_{j_J}\ \boldsymbol{1}_r\right), \quad (10)$$

where $\boldsymbol{r}^\top *$ operation basically marginalizes out the root variables, *i.e.*, $\sum_{x_r}\mathbb{P}[x_r]*$. Note that in both (8) and (10), we require that the message multiplications $\boldsymbol{M}_{j_1}\boldsymbol{M}_{j_2}\ldots\boldsymbol{M}_{j_J}$ are ordered according to the cyclic order of the siblings illustrated in Figure 1.

5 Spectral Algorithm for LTMs

The drawback of the representations in (8) and (10) is that they require the exact knowledge of the parameters (CPTs) associated with latent variables, but none of them are available in training. If we are not interested in recovering these model parameters but only in the marginal probability of the observed variables (*i.e.*, inference), we may not need to recover the transition tensors $\boldsymbol{\mathcal{T}}$, the messages $\boldsymbol{M}$ and the root marginal $\boldsymbol{r}$ exactly. Our key observation from (8) and (10) is that as long as we can recover them up to some invertible transformations, we will still be able to compute the marginal probability correctly.

For example, we can introduce a pair of matrices, $\boldsymbol{R}$ and $\boldsymbol{L}$ ($\boldsymbol{R}\boldsymbol{L}^{-1} = \boldsymbol{I}$), between $\boldsymbol{M}_{j_1}$ and $\boldsymbol{M}_{j_2}$ in (10), *i.e.*,

$$\mathbb{P}[x_1,\ldots,x_O] = \boldsymbol{r}^\top\left(\boldsymbol{M}_{j_1}\boldsymbol{R}\boldsymbol{L}^{-1}\boldsymbol{M}_{j_2}\ldots\boldsymbol{M}_{j_J}\ \boldsymbol{1}_r\right),$$

without changing the final marginal probability. This is interesting because the transformed representation ($\boldsymbol{M}_{j_1}\boldsymbol{R}$ and $\boldsymbol{L}^{-1}\boldsymbol{M}_{j_2}$) provides us an additional degree of freedom for algorithm design: we want to choose $\boldsymbol{R}$ and $\boldsymbol{L}$ from the large class of invertible matrices, such that the transformed representation can be recovered from observed quantities without the need for accessing the latent variables.

We will show that such $\boldsymbol{R}$ and $\boldsymbol{L}$ can be constructed from singular vectors $\boldsymbol{U}$ of the joint probability matrices of certain pairs of observed variables. Given a finite number of samples, this leads us to a very efficient algorithm for estimating the joint probability $\mathbb{P}[x_1,\ldots,x_O]$: the main computation only involves a sequence of singular value decompositions of empirical pairwise joint probability matrices. Furthermore, our algorithm's sample complexity will depend on the singular values of the pairwise joint probability matrices. The dependence of our method on the spectral properties of the model give the name "spectral algorithm".

5.1 Transformed Tensor Representation

More specifically, we transform each message $\boldsymbol{M}_j$ by two invertible matrices $\boldsymbol{L}_j$ and $\boldsymbol{R}_j$, one from its left and one from the right (see Figure 1 for illustration). Then the message update in (8) can be re-written as

$$\boldsymbol{M}_i = \boldsymbol{\mathcal{T}}_i\ \bar{\times}_1 \quad (11)$$
$$(\boldsymbol{L}_{j_1}\boldsymbol{L}_{j_1}^{-1}\boldsymbol{M}_{j_1}\boldsymbol{R}_{j_1}\boldsymbol{L}_{j_2}^{-1}\boldsymbol{M}_{j_2}\boldsymbol{R}_{j_2}\ldots\boldsymbol{L}_{j_J}^{-1}\boldsymbol{M}_{j_J}\boldsymbol{R}_{j_J}\boldsymbol{R}_{j_J}^{-1}\boldsymbol{1}_i).$$

We further require that $\boldsymbol{R}_{j_1} = \boldsymbol{L}_{j_2}$, $\boldsymbol{R}_{j_2} = \boldsymbol{L}_{j_3}$ *etc.* such that the transformations cancel out with each other, *e.g.*, $\boldsymbol{R}_{j_1}\boldsymbol{L}_{j_2}^{-1} = \boldsymbol{I}$. Since the message multiplications $\boldsymbol{M}_{j_1}\boldsymbol{M}_{j_2}\ldots\boldsymbol{M}_{j_J}$ are ordered according to the cyclic order of the siblings, this is equivalent to requiring that $\boldsymbol{R}_j$ of X_j be equal to matrix $\boldsymbol{L}_{\rho_j}$ of its right sibling X_{ρ_j}, *i.e.*, $\boldsymbol{R}_j = \boldsymbol{L}_{\rho_j}$; similarly, we require $\boldsymbol{L}_j = \boldsymbol{R}_{\lambda_j}$.

The same can be done with $\boldsymbol{\mathcal{T}}_i$. If we propagate message $\boldsymbol{M}_i$ one step further to its parent X_{π_i} and the outgoing message at X_{π_i} can be written as

$$\boldsymbol{M}_{\pi_i} = \boldsymbol{\mathcal{T}}_{\pi_i}\ \bar{\times}_1\left(\ldots\boldsymbol{L}_i^{-1}\ \boldsymbol{M}_i\ \boldsymbol{R}_i\ldots\boldsymbol{1}_{\pi_i}\right) \quad (12)$$

We now show that we can re-define the tensors $\boldsymbol{\mathcal{T}}$ and messages $\boldsymbol{M}$ by grouping them with these transformations such that the message recursion still works in this transformed representation.

From (11) and (12), we observe that the components in the tensor representation ($\boldsymbol{\mathcal{T}}$, $\boldsymbol{M}$ and $\boldsymbol{1}$) have been "sandwiched" by the invertible transformations ($\boldsymbol{R}$ and $\boldsymbol{L}$). Therefore we can define a set of new quanties

$$\tilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{T}}_i \times_1 \boldsymbol{L}_{j_1}^\top \times_2 \boldsymbol{L}_i^{-1} \times_3 \boldsymbol{R}_i^\top \quad (13)$$
$$\tilde{\boldsymbol{M}}_j = \boldsymbol{L}_j^{-1}\ \boldsymbol{M}_j\ \boldsymbol{R}_j \quad (14)$$
$$\tilde{\boldsymbol{1}}_i = \boldsymbol{R}_{j_J}^{-1}\boldsymbol{1}_i \quad (15)$$

where $\tilde{\boldsymbol{\mathcal{T}}}_i$ in (13) is obtained by absorbing the leading transformation ($\boldsymbol{L}_{j_1}$) from (11), and the other two parent-level transformations from (12). Then the message update can be expressed in these new quantities as

$$\tilde{\boldsymbol{M}}_i = \tilde{\boldsymbol{\mathcal{T}}}_i\ \bar{\times}_1\left(\tilde{\boldsymbol{M}}_{j_1}\ldots\tilde{\boldsymbol{M}}_{j_J}\ \tilde{\boldsymbol{1}}_i\right) \quad (16)$$

Similarly, we can transform the probability vector $\boldsymbol{r}$ at root node by $\boldsymbol{L}_{j_1}$, which leads to $\tilde{\boldsymbol{r}}^\top = \boldsymbol{r}^\top\boldsymbol{L}_{j_1}$ and the final joint probability is

$$\mathbb{P}[x_1,\ldots,x_O] = \tilde{\boldsymbol{r}}^\top\left(\tilde{\boldsymbol{M}}_{j_1}\ldots\tilde{\boldsymbol{M}}_{j_J}\ \tilde{\boldsymbol{1}}_r\right). \quad (17)$$

Next we show that $\boldsymbol{R}$ and $\boldsymbol{L}$ can be chosen smartly such that all quantities in the transformed representation can be recovered from observed quantities.

5.2 Observable Representation

We now show that the transformed representation of latent tree models can be reconstructed from observed quantities. Each component requires at most 3 observed variables for the reconstruction, so the trees have to be triconnected, *i.e.*, each node has at least 3 neighbors. Our strategy is to relate latent quantities to observed quantities using the sum rule of probability; then based on these relations, we solve for the latent quantities. For notation, let $\boldsymbol{O}_{ij}$ be a conditional probability matrix: $\boldsymbol{O}_{ij}(k,l) = \mathbb{P}[X_i = k | X_j = l]$. Recall that X_{λ_i} and X_{ρ_i} are the left/right siblings of X_i respectively, and X_{i^*} is an observed leaf in the subtree rooted at X_i (see Figure 1).

Transition tensor in (13). If we choose $\boldsymbol{L}_{j_1} = \boldsymbol{O}_{j_1^* i}^\top \boldsymbol{U}_{j_1^*}$, $\boldsymbol{L}_i = \boldsymbol{O}_{i^*\pi_i}^\top \boldsymbol{U}_{i^*}$ and $\boldsymbol{R}_i = \boldsymbol{O}_{\rho_i^*\pi_i}\boldsymbol{U}_{\rho_i^*}$, then the tensor in (13) becomes

$$\begin{aligned}\tilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{T}}_i \times_1 & \\ (\boldsymbol{U}_{j_1^*}^\top \boldsymbol{O}_{j_1^* i}) \times_2 (\boldsymbol{O}_{i^*\pi_i}^\top \boldsymbol{U}_{i^*})^{-1} & \times_3 (\boldsymbol{U}_{\rho_i^*}^\top \boldsymbol{O}_{\rho_i^*\pi_i}), \qquad (18)\end{aligned}$$

where $\boldsymbol{U}_{j^*}$ is a matrix specific to $\boldsymbol{O}_{j^*i}$ such that $\boldsymbol{O}_{j^*i}^\top \boldsymbol{U}_j$ is invertible. One choice for $\boldsymbol{U}_{j^*}$ that meets this requirement is to perform a "thin" singular value decompositions (SVDs) of the pair marginal $\mathbb{P}[X_{\lambda_j^*}, X_{j^*}]$, and then take the first S_H right principal singular vectors to form $\boldsymbol{U}_{j^*}$. In the rest of the paper, we use this SVD approach to obtain $\boldsymbol{U}_{j^*}$.

Let $\boldsymbol{a}_i$ be a marginal probability vector with entries $\boldsymbol{a}_i(k) = \mathbb{P}[X_i = k]$, then $\mathbb{P}[X_{\lambda_i^*}, X_{i^*}] = \boldsymbol{O}_{\lambda_i^*\pi_i}\operatorname{diag}(\boldsymbol{a}_{\pi_i})\boldsymbol{O}_{i^*\pi_i}^\top$ by summing out X_{π_i}. Next, we multiply tensor $\tilde{\boldsymbol{\mathcal{T}}}_i$ in its mode-2 by $\mathbb{P}[X_{\lambda_i^*}, X_{i^*}]\boldsymbol{U}_{i^*}$, resulting in

$$\begin{aligned}& \tilde{\boldsymbol{\mathcal{T}}}_i \times_2 (\mathbb{P}[X_{\lambda_i^*}, X_{i^*}]\boldsymbol{U}_{i^*}) && (19)\\ &= \tilde{\boldsymbol{\mathcal{T}}}_i \times_2 \boldsymbol{O}_{\lambda_i^*\pi_i}\operatorname{diag}(\boldsymbol{a}_{\pi_i})\boldsymbol{O}_{i^*\pi_i}^\top \boldsymbol{U}_{i^*} \\ &= \mathbb{P}[X_{j_1^*}, X_{\lambda_i^*}, X_{\rho_i^*}] \times_1 \boldsymbol{U}_{j_1^*}^\top \times_3 \boldsymbol{U}_{\rho_i^*}^\top, && (20)\end{aligned}$$

where in (20) we cancel out the $\boldsymbol{O}_{i^*\pi_i}^\top \boldsymbol{U}_{i^*}$ in mode-2 of $\tilde{\boldsymbol{\mathcal{T}}}_i$ in (18), and use the fact that the tensor multiplication effectively marginalizes out variable X_i and X_{π_i}.

Based on (19) and (20), we can recover the hidden $\tilde{\boldsymbol{\mathcal{T}}}_i$ as

$$\begin{aligned}\tilde{\boldsymbol{\mathcal{T}}}_i = {} & \mathbb{P}[X_{j_1^*}, X_{\lambda_i^*}, X_{\rho_i^*}] \times_1 \boldsymbol{U}_{j_1^*}^\top \\ & \times_2 (\mathbb{P}[X_{\lambda_i^*}, X_{i^*}]\boldsymbol{U}_{i^*})^\dagger \times_3 \boldsymbol{U}_{\rho_i^*}^\top, \qquad (21)\end{aligned}$$

where we multiple the pseudo-inverse of $\mathbb{P}[X_{\lambda_i^*}, X_{i^*}]\boldsymbol{U}_{i^*}$ to the mode-2 of the tensor to recover $\tilde{\boldsymbol{\mathcal{T}}}_i$ from (19). We also note that since X_{j_1} is a child of X_i, we can set $i^* = j_1^*$.

Message in (14). We only need to derive the case for leaf nodes, and the messages from internal nodes will be automatically taken care of by the message recursion. Choose $\boldsymbol{L}_i = \boldsymbol{O}_{i\pi_i}^\top \boldsymbol{U}_i$ and $\boldsymbol{R}_i = \boldsymbol{O}_{\rho_i^*\pi_i}^\top \boldsymbol{U}_{\rho_i^*}$ respectively, then

$$\tilde{\boldsymbol{M}}_i = (\boldsymbol{O}_{i\pi_i}^\top \boldsymbol{U}_i)^{-1}\boldsymbol{M}_i \boldsymbol{O}_{\rho_i^*\pi_i}^\top \boldsymbol{U}_{\rho_i^*}. \qquad (22)$$

Next we relate $\tilde{\boldsymbol{M}}_i$ to the marginal probability of triples of observed variables in the following way

$$\begin{aligned}& \boldsymbol{O}_{\lambda_i^*\pi_i}\operatorname{diag}(\boldsymbol{a}_{\pi_i})\boldsymbol{O}_{i\pi_i}^\top \boldsymbol{U}_i \tilde{\boldsymbol{M}}_i \\ &= \mathbb{P}[X_{\lambda_i^*}, X_i]\boldsymbol{U}_i\tilde{\boldsymbol{M}}_i && (23)\\ &= \boldsymbol{O}_{\lambda_i^*\pi_i}\operatorname{diag}(\boldsymbol{a}_{\pi_i})\boldsymbol{M}_i\boldsymbol{O}_{\rho_i^*\pi_i}^\top \boldsymbol{U}_{\rho_i^*} \\ &= \mathbb{P}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]\boldsymbol{U}_{\rho_i^*}. && (24)\end{aligned}$$

where we use the fact that $\boldsymbol{M}_i$ is a diagonal matrix with nonzero entries coming from the likelihood vector $\mathbb{P}[x_i|X_{\pi_i}]$. Note that $\mathbb{P}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]$ is a slice of the joint probability tensor where the middle variable X_i is fixed at x_i. Based on (23) and (24), we can recover $\tilde{\boldsymbol{M}}_i$ as

$$\tilde{\boldsymbol{M}}_i = (\mathbb{P}[X_{\lambda_i^*}, X_i]\boldsymbol{U}_i)^\dagger\, \mathbb{P}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]\boldsymbol{U}_{\rho_i^*}, \qquad (25)$$

where the tensor slice $\mathbb{P}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]$ behaves as a matrix.

One in (15) and root marginal in (17). We let $\boldsymbol{R}_{j_J} = \boldsymbol{O}_{\rho_{j_J}^* i}^\top \boldsymbol{U}_{\rho_{j_J}^*}$ (the parent of X_{j_J} is X_i), then the transformed $\tilde{\mathbf{1}}$ becomes $(\boldsymbol{O}_{\rho_{j_J}^* i}^\top \boldsymbol{U}_{\rho_{j_J}^*})^{-1}\mathbf{1}_i$. Next we multiply it by $\mathbb{P}[X_{j_J^*}, X_{\rho_{j_J}^*}] = \boldsymbol{O}_{j_J^* i}\operatorname{diag}(\boldsymbol{a}_i)(\boldsymbol{O}_{\rho_{j_J}^* i}^\top \boldsymbol{U}_{\rho_{j_J}^*})$, resulting in

$$\begin{aligned}& \boldsymbol{O}_{j_J^* i}\operatorname{diag}(\boldsymbol{a}_i)(\boldsymbol{O}_{\rho_{j_J}^* i}^\top \boldsymbol{U}_{\rho_{j_J}^*})(\boldsymbol{O}_{\rho_{j_J}^* i}^\top \boldsymbol{U}_{\rho_{j_J}^*})^{-1}\mathbf{1}_i \\ &= \mathbb{P}[X_{j_J^*}, X_{\rho_{j_J}^*}]\boldsymbol{U}_{\rho_{j_J}^*}\tilde{\mathbf{1}}_i = \mathbb{P}[X_{j_J^*}] \qquad (26)\end{aligned}$$

Based on (26), we can recover $\tilde{\mathbf{1}}_i$ as

$$\tilde{\mathbf{1}}_i = (\mathbb{P}[X_{j_J^*}, X_{\rho_{j_J}^*}]\boldsymbol{U}_{\rho_{j_J}^*})^\dagger\, \mathbb{P}[X_{j_J^*}]. \qquad (27)$$

Similarly, we can also recover the transformed $\tilde{\boldsymbol{r}}$ as

$$\tilde{\boldsymbol{r}}^\top = \boldsymbol{r}^\top \boldsymbol{L}_{j_1^*} = \boldsymbol{r}^\top \boldsymbol{O}_{j_1^* r}^\top \boldsymbol{U}_{j_1^*} = \mathbb{P}[X_{j_1^*}]^\top \boldsymbol{U}_{j_1^*}, \qquad (28)$$

where here X_{j_1} refers to a child of X_r.

We note that our derivation of the observable representation requires a variable to have at least two different siblings. This basically requires that each internal node have at least 3 children. However, with slight modifications, our reasoning in this section also applies to the cases where an internal node X_i has only 2 children. In this case, we only need to conceptually "adopt" a child from the sibling X_{λ_i} of X_i, and order this "adopted" child with the other 2 children of X_i in cyclic order.

5.3 Spectral Algorithm

We now present a spectral algorithm for latent tree models based on the observable representation in the previous section. We assume that the topologies of the trees are given. Then, given N *i.i.d.* samples of the observed variables $\{x_1, \ldots, x_O\}$ from a latent tree model (latent variables are not observed at either training or test time), our algorithm first selects a root and then sorts the other nodes in topological order. Then we estimate the empirical marginal distributions of up to triples of variables needed to recover the observable representations in (21), (25), (27) and (28).

Next for each $\boldsymbol{U}_{j^*}$ required in (21), (25), (27) and (28), we perform a "thin" singular value decompositions (SVDs) of the empirical pair marginal $\widehat{\mathbb{P}}[X_{\lambda_j^*}, X_{j^*}]$, and taking the first S_H right principal singular vectors to form an estimate $\widehat{\boldsymbol{U}}_{j^*}$. Since X_{j^*} is an arbitrary observed variable in the subtree induced by X_j and its descendants, there can be multiple choices for X_{j^*}. In practice, we choose X_{j^*} such that the S_H^{th} largest singular value of $\mathbb{P}[X_{\lambda_j^*}, X_{j^*}]$ is large (justified by Theorem 1). Finally, we compute the $\boldsymbol{U}_j$ for each observed variable X_j only once at initialization, and store it for later use. We summarize the algorithm in Algorithm 1.

Algorithm 1 Spectral Algorithm for Latent Tree Models

```
In: Tree topology and $N$ i.i.d. samples $\{x_1^s, \ldots, x_O^s\}_{s=1}^N$
Out: Estimated marginal $\widehat{\mathbb{P}}[x_1, \ldots, x_O]$
1: Select a root node and sort other nodes in topological order.
2: For each $\tilde{\boldsymbol{\mathcal{T}}}_i$, $\tilde{\mathbf{1}}_i$ in an internal node $X_i$, each $\tilde{\boldsymbol{M}}_i$ in a leaf
   node, and $\boldsymbol{r}$ in the root node, estimate the following empirical
   marginals from training samples:
```

	Triple	Pair	Singleton
$\tilde{\boldsymbol{\mathcal{T}}}_i$	$\widehat{\mathbb{P}}[X_{j_1^*}, X_{\lambda_i^*}, X_{\rho_i^*}]$	$\widehat{\mathbb{P}}[X_{\lambda_i^*}, X_{i^*}]$	-
$\tilde{\mathbf{1}}_i$	-	$\widehat{\mathbb{P}}[X_{j_J^*}, X_{\rho_{j_J}^*}]$	$\widehat{\mathbb{P}}[X_{j_J^*}]$
$\tilde{\boldsymbol{M}}_i$	$\widehat{\mathbb{P}}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]$	$\widehat{\mathbb{P}}[X_{\lambda_i^*}, X_i]$	-
$\tilde{\boldsymbol{r}}$	-	-	$\widehat{\mathbb{P}}[X_{j_1^*}]$

```
   ($j_1, j_J \in \chi_i$ for internal nodes and $j_1 \in \chi_r$ for the root.)
3: For each leaf node $X_i$, perform a "thin" singular value de-
   composition of $\widehat{\mathbb{P}}[X_{\lambda_i^*}, X_i] = \boldsymbol{V\Sigma U}^\top$; let $\widehat{\boldsymbol{U}}_i = \boldsymbol{U}(:, 1 : S_H)$
   be the the first $S_H$ principal right singular vectors.
4: Estimate each $\tilde{\boldsymbol{\mathcal{T}}}_i$, $\tilde{\mathbf{1}}_i$ in internal nodes, each $\tilde{\boldsymbol{M}}_i$ in leaf
   nodes, and $\tilde{\boldsymbol{r}}$ via
      $\widehat{\boldsymbol{\mathcal{T}}}_i = \widehat{\mathbb{P}}[X_{j_1^*}, X_{\lambda_i^*}, X_{\rho_i^*}] \times_1 \widehat{\boldsymbol{U}}_{j_1^*}^\top \times_2 (\widehat{\mathbb{P}}[X_{\lambda_i^*}, X_{i^*}]\widehat{\boldsymbol{U}}_{i^*})^\dagger \times_3 \widehat{\boldsymbol{U}}_{\rho_i^*}^\top$   (29)
      $\widehat{\mathbf{1}}_i = (\widehat{\mathbb{P}}[X_{j_J^*}, X_{\rho_{j_J}^*}]\widehat{\boldsymbol{U}}_{\rho_{j_J}^*})^\dagger \, \widehat{\mathbb{P}}[X_{j_J^*}]$   (30)
      $\widehat{\boldsymbol{M}}_i = (\widehat{\mathbb{P}}[X_{\lambda_i^*}, X_i]\widehat{\boldsymbol{U}}_i)^\dagger \, \widehat{\mathbb{P}}[X_{\lambda_i^*}, x_i, X_{\rho_i^*}]\widehat{\boldsymbol{U}}_{\rho_i^*}$   (31)
      $\widehat{\boldsymbol{r}}^\top = \widehat{\mathbb{P}}[X_{j_1^*}]^\top \widehat{\boldsymbol{U}}_{j_1^*}$   (32)
5: In reverse topological order, internal nodes send messages
      $\widehat{\boldsymbol{M}}_i = \widehat{\boldsymbol{\mathcal{T}}}_i \, \bar{\times}_1 \left(\widehat{\boldsymbol{M}}_{j_1} \ldots \widehat{\boldsymbol{M}}_{j_J} \, \widehat{\mathbf{1}}_i\right)$,   (33)
   and at root node, all incoming messages are combined
      $\widehat{\mathbb{P}}[x_1, \ldots, x_O] = \widehat{\boldsymbol{r}}^\top \left(\widehat{\boldsymbol{M}}_{j_1} \ldots \widehat{\boldsymbol{M}}_{j_J} \, \widehat{\mathbf{1}}_r\right)$   (34)
```

5.4 Sample Complexity

We analyze the sample complexity of Algorithm 1 and find that it depends on the tree topology and the spectral properties of the true model. See the supplementary for a proof [1].

Theorem 1 *For any $\epsilon > 0, 0 < \delta < 1$, let*

$$N \geq O\left(\frac{(d_{max} S_H)^{2\ell+1}}{\alpha\beta\epsilon^2}\right) \log\frac{|\mathscr{O}|}{\delta}$$

where $\sigma_{S_H}(*)$ returns the S_H^{th} largest singular value and

$$\alpha = \min_{i \neq j, i,j \in \mathscr{O}} \ \sigma_{S_H}(\mathbb{P}[X_i, X_j])^4$$
$$\beta = \min_{i \in \mathscr{O}} \ \sigma_{S_H}(\boldsymbol{O}_{i\pi_i})^2$$

Then $\sum_{x_1,\ldots,x_O} \left|\widehat{\mathbb{P}}[x_1, \ldots, x_O] - \mathbb{P}[x_1, \ldots, x_O]\right| \leq \epsilon$ with probability $1 - \delta$.

This result implies that the estimation problem gets harder as the maximum degree $d_{\max}$ of the hidden nodes, the number S_H of the hidden states, and the length ℓ of the chain of hidden variables increase. Furthermore, the sample complexity depends exponentially in ℓ, which suggests that we should choose the root of the tree to make ℓ small. However, we believe that such adverse dependence on ℓ is due to the artifact of our analysis.

[1] can be found at http://www.sailing.cs.cmu.edu

A special case of latent tree models is hidden Markov models (HMMs). Recently, Hsu et al. (2009) derived a spectral algorithm specific to HMMs. Their reasoning relies heavily on a single connected chain of hidden variables and each hidden variable has an observed variable attached. Although this excludes many interesting tree topologies, they obtained a tighter sample complexity bound which is $O(\frac{S_H \ell^2}{\alpha\beta\epsilon^2})$ (polynomial in ℓ). This also suggests that our analysis can be further improved.

6 Discussion

There can be other representations of latent tree models. However, we are not aware of other representations for the marginal computation in (7) which lead to an efficient estimation procedure that requires only marginals of triples of observed variables. For instance, we can represent each hidden node X_i by a $(J+1)$th order transition tensor $\boldsymbol{\mathcal{T}}_i$ ($J = |\chi_i|$) where $\boldsymbol{\mathcal{T}}_i$ is diagonal in the first J modes ($\boldsymbol{\mathcal{T}}_i(k, \ldots, k, l) = \mathbb{P}[X_i = k | X_{\pi_i}]$). The root node can be represented as a Jth order diagonal tensor $\boldsymbol{\mathcal{R}}$ ($J = |\chi_r|$, and $\boldsymbol{\mathcal{R}}(k, \ldots, k) = \mathbb{P}[X_r = k]$). If we represent message from leaf node x_i as vectors $\boldsymbol{m}_i = \mathbb{P}[x_i | X_{\pi_i}]$, then the message update can be written as

$$\boldsymbol{m}_i = \boldsymbol{\mathcal{T}}_i \ \bar{\times}_1 \ \boldsymbol{m}_{j_1} \ldots \ \bar{\times}_J \ \boldsymbol{m}_{j_J}. \tag{35}$$

and the final marginal probability is computed as

$$\mathbb{P}[x_1, \ldots, x_O] = \boldsymbol{\mathcal{R}} \ \bar{\times}_1 \ \boldsymbol{m}_{j_1} \ldots \ \bar{\times}_J \ \boldsymbol{m}_{j_J}. \tag{36}$$

Although this representation is more elegant, it does not suggest a simple spectral algorithm. We conjecture that in order to recover an observable representation for $\boldsymbol{\mathcal{R}}$, we will need the marginal of J observed variables, which will be impractical for large J.

Another representation is to use elementwise vector products. Let $\boldsymbol{T}_i = \mathbb{P}[X_i | X_{\pi_i}]$ be the transition matrix at internal node X_i, and represent message from leaf node x_i as vectors $\boldsymbol{m}_i = \mathbb{P}[x_i | X_{\pi_i}]$, then message update becomes

$$\boldsymbol{m}_i = \boldsymbol{T}_i \left(\boldsymbol{m}_{j_1} \circ \ldots \circ \boldsymbol{m}_{j_J}\right), \tag{37}$$

and the final marginal probability is computed as

$$\mathbb{P}[x_1, \ldots, x_O] = \boldsymbol{r}^\top \left(\boldsymbol{m}_{j_1} \circ \ldots \circ \boldsymbol{m}_{j_J}\right). \tag{38}$$

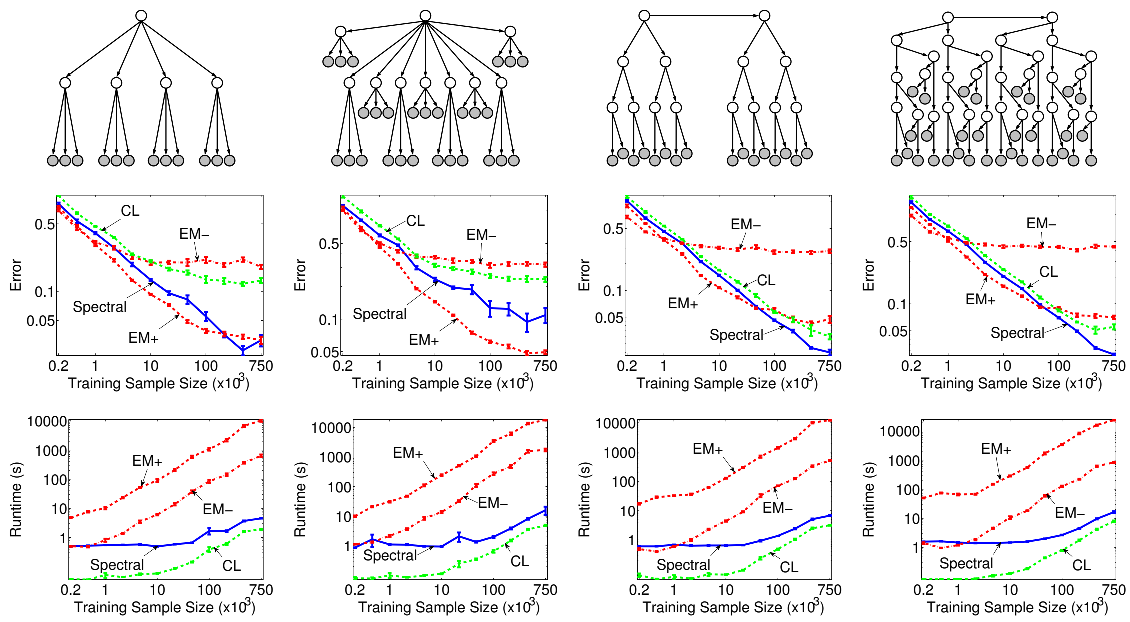
However, in this form, it is not clear how one can derive a observable representation and a spectral algorithm.

7 Experiments

We evaluate our spectral algorithm using 3 sets of experiments. Overall, our spectral algorithm is orders of magnitude faster than EM (main competitor) while being more accurate at sufficiently large sample sizes.

7.1 Comparisons with EM and Chow-Liu Tree

We first generate synthetic data from latent tree models with four different topologies (broad4, broad9, deep4 and deep5 shown in Figure 2). These tree topologies are designed so that the tree either grows broader (broad4 to broad9), or grows deeper (deep4 to deep5). We set $S_O = 6$,



(a) broad4 (b) broad9 (c) deep4 (d) deep5

Figure 2. Comparison of our spectral algorithm (Spectral) to EM algorithm with high precision (EM+) and low precision (EM-), and Chow-Liu tree learning (CL) for 4 different tree topologies shown in the first row. Both errors and runtimes are plotted in log scale.

and $S_H = 2$, *i.e.*, $S_O > S_H$. In this case, the Mossel & Roch (2006) algorithm no longer applies, and therefore we compare with EM and inference on the Chow-Liu tree.

For an experiment on a given tree type with N training points, we randomly generate 10 sets of model parameters and sample N training points and 1000 test points for each parameter set. For EM, we learn the CPTs (with 5 restarts) based on the training points and the true latent tree topology, and then perform inference on test points using message passing. We experiment with a low precision EM (0.01, denoted as EM-) and a high precision EM (0.0005, denoted as EM+) to give a better perspective on the time versus accuracy tradeoff as compared to our approach. For the Chow-Liu tree, we first learn the topology of a fully observable tree model using the Chow-Liu algorithm and the true pairwise marginals; then we learn the CPTs in the Chow-Liu tree using training points and perform inference on test points using message passing. We measure the performance of joint estimation using $\epsilon = \frac{|\widehat{\mathbb{P}}[x_1,\ldots,x_O]-\mathbb{P}[x_1,\ldots,x_O]|}{\mathbb{P}[x_1,\ldots,x_O]}$, and we vary the training sample size N from 200 to 750,000, and report both the runtime for training and the test error for inference in Figure 2. (Test runtimes are about the same for all methods).

Figure 2(a)(b)(c)(d), show that our spectral algorithm can be orders of magnitude faster than EM for large sample sizes and that Chow-Liu tree learning is fastest (we do not count the time for tree topology learning), since it only needs to estimate a collection of CPTs for pairs of random variables. In terms of estimation errors, EM+ and EM- perform the best for small training sizes. However, when the sample sizes go beyond 5,000, the performance of EM- levels off and our spectral algorithm overtakes EM-. This is because EM- learns the models with a low precision (0.01), and as we increase sample size beyond certain point, this fixed precision simply dominates the estimation error. Similarly, we also see that for large sample sizes our algorithm overtakes EM+ on 2 of the 4 experiments (deep4 and deep5) and performs equally on broad4 (EM+ does better on broad9). Furthermore, our spectral algorithm is significantly better than the Chow-Liu tree learning over the range of sample sizes. This is expected since both our spectral algorithm and EM use the correct tree topology while the fully observable tree learned by Chow-Liu has introduced large bias into the model.

Finally, our method's performance does degrade as the topologies become more complex. However, the performance does not seem to degrade exponentially with the length of the chain of hidden variables, which suggests that our sample complexity analysis can be further improved.

7.2 Comparison with Mossel and Roch Algorithm

We now make a separate comparison with the spectral algorithm by (Mossel & Roch, 2006), since it only applies to case where the number of observed states S_O is the same as the number of hidden states S_H. We use the same experimental settings as in the previous section, but set $S_O = S_H = 2$. Although this method is theoretically interesting, it can perform poorly in practice.

The results are shown in Figure 3(a)-(d) (the runtime of both methods are similar, and thus not reported). Our spec-

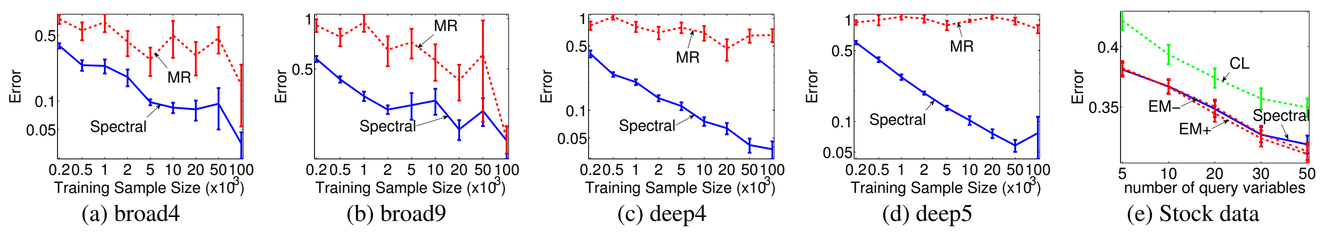


Figure 3. (a)-(d) Comparison of our spectral algorithm (Spectral) with the Mossel and Roch algorithm (MR) for 4 different latent tree topologies. The errors are plotted in log scale. (e) Comparison of our spectral algorithm (Spectral, blue line) with EMs (EM+ and EM-, red lines) and Chow-Liu based algorithm (CL, green line) on stock dataset.

tral algorithm significantly outperforms the MR algorithm on all trees for practically all sample sizes. This is because our method does not explicitly recover the CPTs, and is thus more robust. We also note that our approach is more general: it can allow for the observation state space to be larger than the hidden state space, which may be preferable in many applications where the observation space can be large (*e.g.*, quantization of a continuous variable), but the hidden factors are simple and have lower dimensions.

7.3 Stock Trend Prediction

Finally, we evaluate our algorithm on a stock trend prediction problem. Our goal is to predict whether a stock X_i will go up or down on a particular day given the trends of a set $\mathscr{E}$ of other stocks. We acquired closing prices of 59 stocks from 1984 to 2011, which provides us 6800 samples.[2] We randomly partition these samples to 6300 training points and 500 test points. Since we are only predicting whether a stock goes up or down, the data are binarized. From the training data, we learn the latent tree topology using an algorithm by Choi et al. (2010), and a fully observable Chow-Liu tree (Chow & Liu, 1968). A visualization of the learned tree topologies are in the supplementary.

We compare our spectral algorithm to EM+ and EM- using the latent tree, and with inference over the Chow-Liu tree. For the prediction task, we need to estimate the conditional, *i.e.*, $\mathbb{P}[X_i|x_{j_1},\ldots,x_{j_{|\mathscr{E}|}}]$ and $j_1,\ldots,j_{|\mathscr{E}|} \in \mathscr{E}$. This can be achieved by estimating $\mathbb{P}[x_i,x_{j_1},\ldots,x_{j_{|\mathscr{E}|}}]$ for each instantiation x_i. Then we make prediction by $\hat{x}_i = \operatorname{argmax}_{x_i} \mathbb{P}[x_i,x_{j_1},\ldots,x_{j_{|\mathscr{E}|}}]$. We measure the prediction error using $\epsilon = |\hat{x}_i - x_i^\star|$ where $x_i^\star$ is the true label.

We experiment with a varying number of query sizes. For each query size Q, we randomly pick Q stocks and predict the value of one stock conditioned on the other $Q-1$, (and repeat for 50 trials). Over the entire range of query sizes, the advantage of latent tree approaches (Spectral, EM+/-) is clear over the Chow-Liu tree. Thus, the latent factors help better model the stock data in this case. Due to the small training sample size, the distinction between our method and EM is less clear.

[2]www.finance.yahoo.com

8 Conclusion

We have proposed a local-minimum-free spectral algorithm for latent tree models which only uses information from marginals of triples of observed variables irrespective of the maximum degree of the graph. Our algorithm is computationally efficient even for large sample sizes, and shows good performance in both synthetic and real datasets. There are many future directions: one can kernelize the method like Song et al. (2010) did for HMMs, or design spectral algorithms for loopy latent variable models.

Acknowledgements

APP is thankful for a NSF Graduate Fellowship and LS is supported by a Ray and Stephanie Lane Fellowship. This paper is also supported by NIH 1R01GM093156, NIH 1RC2HL101487, NSF IIS-0713379, NSF DBI-0546594, and an Alfred P. Sloan Fellowship to EPX.

References

Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. In *NIPS*, 2002.

Choi, Myung J., Tan, Vicent Y., Anandkumar, Animashree, and Willsky, Alan S. Learning latent tree graphical models. In *arXiv:1009.2722v1*, 2010.

Chow, C. and Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

Clark, A. Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2): 111–122, 1990.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–22, 1977.

Hoff, Peter D., Raftery, Adrian E., and Handcock, Mark S. Latent space approaches to social network analysis. *JASA*, 97(460): 1090–1098, 2002.

Hsu, D., Kakade, S., and Zhang, T. A spectral algorithm for learning hidden markov models. In *COLT*, 2009.

Kolda, Tamara. and Bader, Brett. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Mossel, E. and Roch, S. Learning nonsingular phylogenies and hidden markov models. *AOAP*, 16(2):583–614, 2006.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

Rabiner, L. R. and Juang, B. H. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

Song, L., Boots, B., Siddiqi, S., Gordon, G., and Smola, A. Hilbert space embeddings of hidden markov models. In *ICML*, 2010.