Learning Latent Tree Graphical Models

Myung Jin Choi   MYUNGJIN@MIT.EDU
Stochastic Systems Group
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Vincent Y. F. Tan   VTAN@WISC.EDU
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706

Animashree Anandkumar   A.ANANDKUMAR@UCI.EDU
Center for Pervasive Communications and Computing
Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697

Alan S. Willsky   WILLSKY@MIT.EDU
Stochastic Systems Group
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We study the problem of learning a latent tree graphical model where samples are available only from a subset of variables. We propose two consistent and computationally efficient algorithms for learning *minimal* latent trees, that is, trees without any redundant hidden nodes. Unlike many existing methods, the observed nodes (or variables) are not constrained to be leaf nodes. Our algorithms can be applied to both discrete and Gaussian random variables and our learned models are such that all the observed and latent variables have the same domain (state space). Our first algorithm, *recursive grouping*, builds the latent tree recursively by identifying sibling groups using so-called information distances. One of the main contributions of this work is our second algorithm, which we refer to as *CLGrouping*. CLGrouping starts with a pre-processing procedure in which a tree over the observed variables is constructed. This global step groups the observed nodes that are likely to be close to each other in the true latent tree, thereby guiding subsequent recursive grouping (or equivalent procedures such as neighbor-joining) on much smaller subsets of variables. This results in more accurate and efficient learning of latent trees. We also present regularized versions of our algorithms that learn latent tree approximations of arbitrary distributions. We compare the proposed algorithms to other methods by performing extensive numerical experiments on various latent tree graphical models such as hidden Markov models and star graphs. In addition, we demonstrate the applicability of our methods on real-world data sets by modeling the dependency structure of monthly stock returns in the S&P index and of the words in the 20 newsgroups data set.

Keywords: graphical models, Markov random fields, hidden variables, latent tree models, structure learning

1. Introduction

The inclusion of latent variables in modeling complex phenomena and data is a well-recognized and a valuable construct in a variety of applications, including bio-informatics and computer vision, and the investigation of machine-learning methods for models with latent variables is a substantial and continuing direction of research.

There are three challenging problems in learning a model with latent variables: learning the number of latent variables; inferring the structure of how these latent variables relate to each other and to the observed variables; and estimating the parameters characterizing those relationships. Issues that one must consider in developing a new learning algorithm include developing tractable methods; incorporating the tradeoff between the fidelity to the given data and generalizability; deriving theoretical results on the performance of such algorithms; and studying applications that provide clear motivation and contexts for the models so learned.

One class of models that has received considerable attention in the literature is the class of *latent tree models*, that is, graphical models Markov on trees, in which variables at some nodes represent the original (observed) variables of interest while others represent the latent variables. The appeal of such models for computational tractability is clear: with a tree-structured model describing the statistical relationships, inference—processing noisy observations of some or all of the original variables to compute the estimates of all variables—is straightforward and scalable. Although the class of tree-structured models, with or without latent variables, is a constrained one, there are interesting applications that provide strong motivation for the work presented here. In particular, a very active avenue of research in computer vision is the use of context—for example, the nature of a scene to aid the reliable recognition of objects (and at the same time to allow the recognition of particular objects to assist in recognizing the scene). For example, if one knows that an image is that of an office, then one might expect to find a desk, a monitor on that desk, and perhaps a computer mouse. Hence if one builds a model with a latent variable representing that context ("office") and uses simple, noisy detectors for different object types, one would expect that the detection of a desk would support the likelihood that one is looking at an office and through that enhance the reliability of detecting smaller objects (monitors, keyboards, mice, etc.). Work along these lines, including by some of the authors of this paper (Parikh and Chen, 2007; Choi et al., 2010), show the promise of using tree-based models of context.

This paper considers the problem of learning tree-structured latent models. If all variables are observed in the tree under consideration, then the well-known algorithm of Chow and Liu (1968) provides a tractable algorithm for performing maximum likelihood (ML) estimation of the tree structure. However, if not all variables are observed, that is, for *latent* tree models, then ML estimation is NP-hard (Roch, 2006). This has motivated a number of investigations of other tractable methods for learning such trees as well as theoretical guarantees on performance. Our work represents a contribution to this area of investigation.

There are three main contributions in our paper. Firstly, by adopting a statistical distance-based framework, we develop two new algorithms for the learning of latent trees—recursive grouping and CLGrouping, which apply equally well to discrete and Gaussian models. Secondly, we provide consistency guarantees (both structural and parametric) as well as very favorable computational and sample complexity characterizations for both of our algorithms. Thirdly, through extensive numerical experiments on both synthetic and real-world data, we demonstrate the superiority of our

approach for a wide variety of models ranging from ones with very large tree diameters (e.g., hidden Markov models (HMMs)) to star models and complete trees.[1]

Our first algorithm, which we refer to as *recursive grouping*, constructs a latent tree in a bottom-up fashion, grouping nodes into sibling groups that share the same parent node, recursively at each level of the resulting hierarchy (and allowing for some of the observed variables to play roles at arbitrary levels in the resulting hierarchy). Our second algorithm, *CLGrouping* first implements a global construction step, namely producing the Chow-Liu tree for the observed variables without any hidden nodes. This global step then provides guidance for groups of observed nodes that are likely to be topologically close to each other in the latent tree, thereby guiding subsequent recursive grouping or neighbor-joining (Saitou and Nei, 1987) computations. Each of these algorithms is consistent and has excellent sample and computational complexity.[2]

As Pearl (1988) points out, the identification of latent tree models has some built-in ambiguity, as there is an entire equivalence class of models in the sense that when all latent variables are marginalized out, each model in this class yields the same joint distribution over the observed variables. For example, we can take any such latent model and add another hidden variable as a leaf node connected to only one other (hidden or observed) node. Hence, much as one finds in fields such as state space dynamic systems (e.g., Luenberger, 1979, Section 8), there is a notion of minimality that is required here, and our results are stated in terms of consistent learning of such minimal latent models.

1.1 Related Work

The relevant literature on learning latent models is vast and in this section, we summarize the main lines of research in this area.

The classical *latent cluster models* (LCM) consider multivariate distributions in which there exists only *one* latent variable and each state of that variable corresponds to a cluster in the data (Lazarsfeld and Henry, 1968). Hierarchical latent class (HLC) models (Zhang and Kočka, 2004; Zhang, 2004; Chen et al., 2008) generalize these models by allowing multiple latent variables. HLC allows latent variables to have different number of states, but assume that all observed nodes are at the leaves of the tree. Their learning algorithm is based on a greedy approach of making one local move at a time (e.g., introducing one hidden node, or replacing an edge), which is computationally expensive and does not have consistency guarantees. A greedy learning algorithm for HLC called BIN is proposed in Harmeling and Williams (2010), which is computationally more efficient. In addition, Silva et al. (2006) considered the learning of directed latent models using so-called tetrad constraints, and there have also been attempts to tailor the learning of latent tree models in order to perform approximate inference accurately and efficiently downstream (Wang et al., 2008). In all these works, the latent variables can have different state spaces, but the observed nodes are required to be leaves of the tree. In contrast, we fix the state space of each hidden node, but allow the possibility that some observed nodes are internal nodes (non-leaves). This assumption leads to an identifiable model, and we provide algorithms with consistency guarantees which can recover the correct structure under mild conditions. In contrast, the works in Zhang and Kočka (2004);

---

1. A tree is called a *complete k*-ary tree (or *k*-complete tree), if all its internal nodes have degree *k* and there exists one node (commonly referred as the root node) that has the exactly same distance to all leaf nodes.

2. As we will see, depending on the true latent tree model, one or the other of these may be more efficient. Roughly speaking, for smaller diameter graphs (such as the star), recursive grouping is faster, and for larger diameter graphs (such as an HMM), CLgrouping is more efficient.

Zhang (2004); Chen et al. (2008); Harmeling and Williams (2010) do not provide such consistency guarantees.

Many authors also propose reconstructing latent trees using the expectation maximization (EM) algorithm (Elidan and Friedman, 2005; Kemp and Tenenbaum, 2008). However, as with all other EM-based methods, these approaches depend on the initialization and suffer from the possibility of being trapped in local optima and thus no consistency guarantees can be provided. At each iteration, a large number of candidate structures need to be evaluated, so these methods assume that all observed nodes are the leaves of the tree to reduce the number of candidate structures. Algorithms have been proposed (Hsu et al., 2009) with sample complexity guarantees for learning HMMs under the condition that the joint distribution of the observed variables generated by distinct hidden states are distinct.

Another related line of research is that of (hierarchical) clustering. See Jain et al. (1999), Balcan and Gupta (2010) and the references therein for extensive discussions. The primary objective of hierarchical clustering is to build a tree consisting of nested partitions of the observed data, where the leaves (typically) consist of single data points while the internal nodes represent coarser partitions. The difference from our work is that hierarchical clustering does not assume a probabilistic graphical model (Markov random field) on the data, but imposes constraints on the data points via a similarity matrix. We are interested in learning tree-structured graphical models with hidden variables.

The reconstruction of latent trees has been studied extensively by the *phylogenetic* community where sequences of extant species are available and the unknown phylogenetic tree is to be inferred from these sequences. See Durbin et al. (1999) for a thorough overview. Efficient algorithms with provable performance guarantees are available (Erdős et al., 1999; Daskalakis et al., 2006). However, the works in this area mostly assume that only the leaves are observed and each internal node (which is hidden) has the same degree except for the root. The most popular algorithm for constructing phylogenetic trees is the *neighbor-joining (NJ) method* by Saitou and Nei (1987). Like our recursive grouping algorithm, the input to the algorithm is a set of statistical distances between observed variables. The algorithm proceeds by recursively pairing two nodes that are the closest neighbors in the true latent tree and introducing a hidden node as the parent of the two nodes. For more details on NJ, the reader is referred to Durbin et al. (1999, Section 7.3).

Another popular class of reconstruction methods used in the phylogenetic community is the family of *quartet-based distance methods* (Bandelth and Dress, 1986; Erdős et al., 1999; Jiang et al., 2001).[3] Quartet-based methods first construct a set of quartets for all subsets of four observed nodes. Subsequently, these quartets are then combined to form a latent tree. However, when we only have access to the samples at the observed nodes, then it is not straightforward to construct a latent tree from a set of quartets since the quartets may be not be consistent.[4] In fact, it is known that the problem of determining a latent tree that agrees with the maximum number of quartets is NP-hard (Steel, 1992), but many heuristics have been proposed (Farris, 1972; Sattath and Tversky, 1977). Also, in practice, quartet-based methods are usually much less accurate than NJ (St. John et al., 2003), and hence, we only compare our proposed algorithms to NJ in our experiments. For further comparisons (the sample complexity and other aspects of) between the quartet methods and NJ, the reader is referred to Csűrös (2000) and St. John et al. (2003).

---

3. A *quartet* is simply an unrooted binary tree on a set of four observed nodes.

4. The term *consistent* here is not the same as the estimation-theoretic one. Here, we say that a set of quartets is *consistent* if there exists a latent tree such that all quartets agree with the tree.

Another distance-based algorithm was proposed in Pearl (1988, Section 8.3.3). This algorithm is very similar in spirit to quartet-based methods but instead of finding quartets for *all* subsets of four observed nodes, it finds *just enough* quartets to determine the location of each observed node in the tree. Although the algorithm is consistent, it performs poorly when only the samples of observed nodes are available (Pearl, 1988, Section 8.3.5).

The learning of phylogenetic trees is related to the emerging field of *network tomography* (Castro et al., 2004) in which one seeks to learn characteristics (such as structure) from data which are only available at the end points (e.g., sources and sinks) of the network. However, again observations are only available at the leaf nodes and usually the objective is to estimate the delay distributions corresponding to nodes linked by an edge (Tsang et al., 2003; Bhamidi et al., 2009). The modeling of the delay distributions is different from the learning of latent tree graphical models discussed in this paper.

1.2 Paper Organization

The rest of the paper is organized as follows. In Section 2, we introduce the notations and terminologies used in the paper. In Section 3, we introduce the notion of information distances which are used to reconstruct tree models. In the subsequent two sections, we make two assumptions: Firstly, the true distribution is a latent tree and secondly, perfect knowledge of information distance of observed variables is available. We introduce recursive grouping in Section 4. This is followed by our second algorithm CLGrouping in Section 5. In Section 6, we relax the assumption that the information distances are known and develop sample based algorithms and at the same time provide sample complexity guarantees for recursive grouping and CLGrouping. We also discuss extensions of our algorithms for the case when the underlying model is not a tree and our goal is to learn an approximation to it using a latent tree model. We demonstrate the empirical performance of our algorithms in Section 7 and conclude the paper in Section 8. The appendix includes proofs for the theorems presented in the paper.

2. Latent Tree Graphical Models

In this section, we provide some background and introduce the notion of minimal-tree extensions and consistency.

2.1 Undirected Graphs

Let $G = (W, E)$ be an undirected graph with vertex (or node) set $W = \{1, \ldots, M\}$ and edge set $E \subset \binom{W}{2}$. Let $\text{nbd}(i; G)$ and $\text{nbd}[i; G]$ be the set of neighbors of node i and the *closed neighborhood* of i respectively, that is, $\text{nbd}[i; G] := \text{nbd}(i; G) \cup \{i\}$. If an undirected graph does not include any loops, it is called a *tree*. A collection of disconnected trees is called a *forest*.[5] For a tree $T = (W, E)$, the set of leaf nodes (nodes with degree 1), the maximum degree, and the diameter are denoted by $\text{Leaf}(T)$, $\Delta(T)$, and $\text{diam}(T)$ respectively. The *path* between two nodes i and j in a tree $T = (W, E)$, which is unique, is the set of edges connecting i and j and is denoted as $\text{Path}((i, j); E)$. The *distance* between any two nodes i and j is the number of edges in $\text{Path}((i, j); E)$. In an undirected tree, we can choose a *root node* arbitrarily, and define the parent-child relationships with respect to the root:

---

5. Strictly speaking, a graph with no loops is called a forest, and it is called a tree only if every node is connected to each other.

for a pair neighboring nodes i and j, if i is closer to the root than j is, then i is called the *parent* of j, and j is called the *child* of i. Note that the root node does not have any parent, and for all other nodes in the tree, there exists exactly one parent. We use $C(i)$ to denote the set of child nodes. A set of nodes that share the same parent is called a *sibling* group. A *family* is the union of the siblings and the associated parent.

A *latent tree* is a tree with node set $W := V \cup H$, the union of a set of observed nodes V (with $m = |V|$), and a set of latent (or hidden) nodes H. The *effective depth* $\delta(T;V)$ (with respect to V) is the maximum distance of a hidden node to its closest observed node, that is,

$$\delta(T;V) := \max_{i \in H} \min_{j \in V} |\text{Path}((i,j);T)|. \tag{1}$$

2.2 Graphical Models

An *undirected graphical model* (Lauritzen, 1996) is a family of multivariate probability distributions that factorize according to a graph $G = (W, E)$. More precisely, let $\mathbf{X} = (X_1, \ldots, X_M)$ be a random vector, where each random variable X_i, which takes on values in an alphabet $\mathcal{X}$, corresponds to variable at node $i \in V$. The set of edges E encodes the set of conditional independencies in the model. The random vector $\mathbf{X}$ is said to be *Markov* on G if for every i, the random variable X_i is conditionally independent of all other variables given its neighbors, that is, if p is the joint distribution[6] of $\mathbf{X}$, then

$$p(x_i|x_{\text{nbd}(i;G)}) = p(x_i|x_{\setminus i}), \tag{2}$$

where $x_{\setminus i}$ denotes the set of all variables[7] excluding x_i. Equation (2) is known as the *local Markov property*.

In this paper, we consider both discrete and Gaussian graphical models. For discrete models, the alphabet $\mathcal{X} = \{1, \ldots, K\}$ is a finite set. For Gaussian graphical models, $\mathcal{X} = \mathbb{R}$ and furthermore, without loss of generality, we assume that the mean is known to be the zero vector and hence, the joint distribution

$$p(\mathbf{x}) = \frac{1}{\det(2\pi\Sigma)^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\right)$$

depends only on the covariance matrix Σ.

An important and tractable class of graphical models is the set of tree-structured graphical models, that is, multivariate probability distributions that are Markov on an undirected tree $T = (W, E)$. It is known from junction tree theory (Cowell et al., 1999) that the joint distribution p for such a model factorizes as

$$p(x_1, \ldots, x_M) = \prod_{i \in W} p(x_i) \prod_{(i,j) \in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)}. \tag{3}$$

That is, the sets of marginal $\{p(x_i) : i \in W\}$ and pairwise joints on the edges $\{p(x_i, x_j) : (i,j) \in E\}$ fully characterize the joint distribution of a tree-structured graphical model.

A special class of a discrete tree-structured graphical models is the set of *symmetric discrete distributions*. This class of models is characterized by the fact that the pairs of variables (X_i, X_j) on

---

6. We abuse the term *distribution* to mean a probability mass function in the discrete case (density with respect to the counting measure) and a probability density function (density with respect to the Lebesgue measure) in the continuous case.

7. We will use the terms node, vertex and variable interchangeably in the sequel.

all the edges $(i, j) \in E$ follow the conditional probability law:

$$p(x_i|x_j) = \begin{cases} 1 - (K-1)\theta_{ij}, & \text{if } x_i = x_j, \\ \theta_{ij}, & \text{otherwise,} \end{cases} \tag{4}$$

and the marginal distribution of *every* variable in the tree is uniform, that is, $p(x_i) = 1/K$ for all $x_i \in \mathcal{X}$ and for all $i \in V \cup H$. The parameter $\theta_{ij} \in (0, 1/K)$ in (4), which does not depend on the state values $x_i, x_j \in \mathcal{X}$ (but can be different for different pairs $(i, j) \in E$), is known as the *crossover probability*.

Let $\mathbf{x}^n := \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ be a set of n i.i.d. samples drawn from a graphical model (distribution) p, Markov on a latent tree $T_p = (W, E_p)$, where $W = V \cup H$. Each sample $\mathbf{x}^{(l)} \in \mathcal{X}^M$ is a length-M vector. In our setup, the learner only has access to samples drawn from the observed node set V, and we denote this set of sub-vectors containing only the elements in V, as $\mathbf{x}_V^n := \{\mathbf{x}_V^{(1)}, \ldots, \mathbf{x}_V^{(n)}\}$, where each observed sample $\mathbf{x}_V^{(l)} \in \mathcal{X}^m$ is a length-m vector. Our algorithms learn latent tree structures using the information distances (defined in Section 3) between pairs of observed variables, which can be estimated from samples.

We now comment on the above model assumptions. Note that we assume that the the hidden variables have the *same* domain as the observed ones (all of which also have a common domain). We do not view this as a serious modeling restriction since we develop efficient algorithms with strong theoretical guarantees, and these algorithms have very good performance on real-world data (see Section 7). Nonetheless, it may be possible to develop a unified framework to incorporate variables with different state spaces (i.e., both continuous and discrete) under a reproducing kernel Hilbert space (RKHS) framework along the lines of Song et al. (2010). We defer this to future work.

2.3 Minimal Tree Extensions

Our ultimate goal is to recover the graphical model p, that is, the latent tree structure and its parameters, given n i.i.d. samples of the observed variables $\mathbf{x}_V^n$. However, in general, there can be multiple latent tree models which result in the same observed statistics, that is, the same joint distribution p_V of the observed variables. We consider the class of tree models where it is possible to recover the latent tree model uniquely and provide necessary conditions for structure identifiability, that is, the identifiability of the edge set E.

Firstly, we limit ourselves to the scenario where *all* the random variables (both observed and latent) take values on a common alphabet $\mathcal{X}$. Thus, in the Gaussian case, each hidden and observed variable is a univariate Gaussian. In the discrete case, each variable takes on values in the same finite alphabet $\mathcal{X}$. Note that the model may not be identifiable if some of the hidden variables are allowed to have arbitrary alphabets. As an example, consider a discrete latent tree model with binary observed variables ($K = 2$). A latent tree with the simplest structure (fewest number of nodes) is a tree in which all m observed binary variables are connected to one hidden variable. If we allow the hidden variable to take on 2^m states, then the tree can describe all possible statistics among the m observed variables, that is, the joint distribution p_V can be arbitrary.[8]

A probability distribution $p_V(\mathbf{x}_V)$ is said to be *tree-decomposable* if it is the marginal (of variables in V) of a tree-structured graphical model $p(\mathbf{x}_V, \mathbf{x}_H)$. In this case, p (over variables in W) is said to be a *tree extension* of p_V (Pearl, 1988). A distribution p is said to have a *redundant* hidden node $h \in H$ if we can remove h and the marginal on the set of visible nodes V remains as p_V.

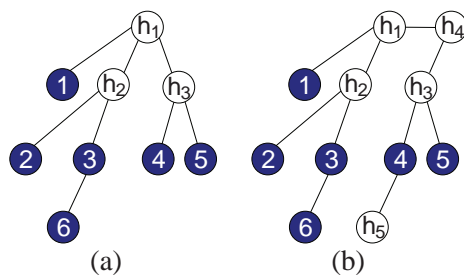8. This follows from a elementary parameter counting argument.

Figure 1: Examples of minimal latent trees. Shaded nodes are observed and unshaded nodes are hidden. (a) An identifiable tree. (b) A non-identifiable tree because h_4 and h_5 have degrees less than 3.

The following conditions ensure that a latent tree does not include a redundant hidden node (Pearl, 1988):

(C1) Each hidden variable has at least three neighbors (which can be either hidden or observed). Note that this ensures that all leaf nodes are observed (although not all observed nodes need to be leaves).

(C2) Any two variables connected by an edge in the tree model are neither perfectly dependent nor independent.

Figure 1(a) shows an example of a tree satisfying (C1). If (C2), which is a condition on parameters, is also satisfied, then the tree in Figure 1(a) is identifiable. The tree shown in Figure 1(b) does not satisfy (C1) because h_4 and h_5 have degrees less than 3. In fact, if we marginalize out the hidden variables h_4 and h_5, then the resulting model has the same tree structure as in Figure 1(a).

We assume throughout the paper that (C2) is satisfied for all probability distributions. Let $\mathcal{T}_{\geq 3}$ be the set of (latent) trees satisfying (C1). We refer to $\mathcal{T}_{\geq 3}$ as the set of *minimal (or identifiable) latent trees*. Minimal latent trees do not contain redundant hidden nodes. The distribution p (over W and Markov on some tree in $\mathcal{T}_{\geq 3}$) is said to be a *minimal tree extension* of p_V. As illustrated in Figure 1, using marginalization operations, any non-minimal latent tree distribution can be reduced to a minimal latent tree model.

Proposition 1 (Minimal Tree Extensions) (Pearl, 1988, Section 8.3)

(i) For every tree-decomposable distribution p_V, there exists a minimal tree extension p Markov on a tree $T \in \mathcal{T}_{\geq 3}$, which is unique up to the renaming of the variables or their values.

(ii) For Gaussian and binary distributions, if p_V is known exactly, then the minimal tree extension p can be recovered.

(iii) The structure of T is uniquely determined by the pairwise distributions of observed variables $p(x_i, x_j)$ for all $i, j \in V$.

2.4 Consistency

We now define the notion of consistency. In Section 6, we show that our latent tree learning algorithms are consistent.

Definition 2 (Consistency) *A* latent tree reconstruction algorithm $\mathcal{A}$ *is a map from the observed samples* $\mathbf{x}_V^n$ *to an estimated tree* $\widehat{T}^n$ *and an estimated tree-structured graphical model* $\widehat{p}^n$*. We say that a latent tree reconstruction algorithm* $\mathcal{A}$ *is* structurally consistent *if there exists a graph homomorphism[9] h such that*

$$\lim_{n\to\infty} \Pr(h(\widehat{T}^n) \neq T_p) = 0. \tag{5}$$

Furthermore, we say that $\mathcal{A}$ *is* risk consistent *if to every* $\varepsilon > 0$,

$$\lim_{n\to\infty} \Pr\left(D(p\,||\,\widehat{p}^n) > \varepsilon\right) = 0, \tag{6}$$

where $D(p\,||\,\widehat{p}^n)$ *is the KL-divergence (Cover and Thomas, 2006) between the true distribution p and the estimated distribution* $\widehat{p}^n$.

In the following sections, we design structurally and risk consistent algorithms for (minimal) Gaussian and symmetric discrete latent tree models, defined in (4). Our algorithms use pairwise distributions between the observed nodes. However, for general discrete models, pairwise distributions between observed nodes are, in general, not sufficient to recover the parameters (Chang and Hartigan, 1991). Therefore, we only prove structural consistency, as defined in (5), for general discrete latent tree models. For such distributions, we consider a two-step procedure for structure and parameter estimation: Firstly, we estimate the structure of the latent tree using the algorithms suggested in this paper. Subsequently, we use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to infer the parameters. Note that, as mentioned previously, risk consistency will not be guaranteed in this case.

3. Information Distances

The proposed algorithms in this paper receive as inputs the set of so-called (exact or estimated) *information distances*, which are functions of the pairwise distributions. These quantities are defined in Section 3.1 for the two classes of tree-structured graphical models discussed in this paper, namely the Gaussian and discrete graphical models. We also show that the information distances have a particularly simple form for symmetric discrete distributions. In Section 3.2, we use the information distances to infer the relationships between the observed variables such as j is a child of i or i and j are siblings.

3.1 Definitions of Information Distances

We define *information distances* for Gaussian and discrete distributions and show that these distances are additive for tree-structured graphical models. Recall that for two random variables X_i and X_j, the *correlation coefficient* is defined as

$$\rho_{ij} := \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var}(X_i)\text{Var}(X_j)}}. \tag{7}$$

---

9. A graph homomorphism is a mapping between graphs that respects their structure. More precisely, a *graph homomorphism h* from a graph $G = (W, E)$ to a graph $G' = (V', E')$, written $h : G \to G'$ is a mapping $h : V \to V'$ such that $(i, j) \in E$ implies that $(h(i), h(j)) \in E'$.

For Gaussian graphical models, the information distance associated with the pair of variables X_i and X_j is defined as:

$$d_{ij} := -\log|\rho_{ij}|. \tag{8}$$

Intuitively, if the information distance d_{ij} is large, then X_i and X_j are weakly correlated and vice-versa.

For discrete random variables, let $\mathbf{J}^{ij}$ denote the joint probability matrix between X_i and X_j (i.e., $J_{ab}^{ij} = p(x_i = a, x_j = b), a, b \in X$). Also let $\mathbf{M}^i$ be the diagonal marginal probability matrix of X_i (i.e., $M_{aa}^i = p(x_i = a)$). For discrete graphical models, the information distance associated with the pair of variables X_i and X_j is defined as Lake (1994):

$$d_{ij} := -\log \frac{|\det \mathbf{J}^{ij}|}{\sqrt{\det \mathbf{M}^i \det \mathbf{M}^j}}. \tag{9}$$

Note that for binary variables, that is, $K = 2$, the value of d_{ij} in (9) reduces to the expression in (8), that is, the information distance is a function of the correlation coefficient, defined in (7), just as in the Gaussian case.

For symmetric discrete distributions defined in (4), the information distance defined for discrete graphical models in (9) reduces to

$$d_{ij} := -(K-1)\log(1 - K\theta_{ij}). \tag{10}$$

Note that there is one-to-one correspondence between the information distances d_{ij} and the model parameters for Gaussian distributions (parametrized by the correlation coefficient ρ_{ij}) in (8) and the symmetric discrete distributions (parametrized by the crossover probability θ_{ij}) in (10). Thus, these two distributions are completely characterized by the information distances d_{ij}. On the other hand, this does not hold for general discrete distributions.

Moreover, if the underlying distribution is a symmetric discrete model or a Gaussian model, the information distance d_{ij} and the mutual information $I(X_i; X_j)$ (Cover and Thomas, 2006) are monotonic, and we will exploit this result in Section 5. For general distributions, this is not valid. See Section 5.5 for further discussions.

Equipped with these definitions of information distances, assumption (C2) in Section 2.3 can be rewritten as the following: There exists constants $0 < l, u < \infty$, such that

$$\text{(C2)} \qquad l \leq d_{ij} \leq u, \qquad \forall (i, j) \in E_p. \tag{11}$$

Proposition 3 (Additivity of Information Distances) *The information distances d_{ij} defined in (8), (9), and (10) are* additive tree metrics *(Erdős et al., 1999). In other words, if the joint probability distribution $p(\mathbf{x})$ is a tree-structured graphical model Markov on the tree $T_p = (W, E_p)$, then the information distances are additive on T_p:*

$$d_{kl} = \sum_{(i,j) \in \text{Path}((k,l); E_p)} d_{ij}, \quad \forall k, l \in W. \tag{12}$$

The property in (12) implies that if each pair of vertices $i, j \in W$ is assigned the weight d_{ij}, then T_p is a minimum spanning tree on W, denoted as $\text{MST}(W; \mathbf{D})$, where $\mathbf{D}$ is the information distance matrix with elements d_{ij} for all $i, j \in V$.
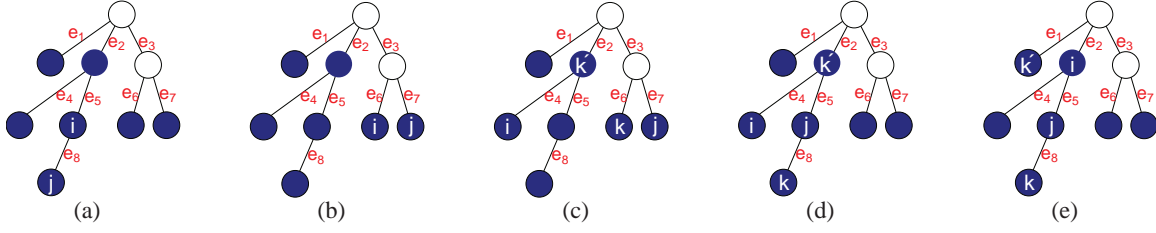
Figure 2: Examples for each case in TestNodeRelationships. For each edge, e_i represents the information distance associated with the edge. (a) Case 1: $\Phi_{ijk} = -e_8 = -d_{ij}$ for all $k \in V \setminus \{i, j\}$. (b) Case 2: $\Phi_{ijk} = e_6 - e_7 \neq d_{ij} = e_6 + e_7$ for all $k \in V \setminus \{i, j\}$ (c) Case 3a: $\Phi_{ijk} = e_4 + e_2 + e_3 - e_7 \neq \Phi_{ijk'} = e_4 - e_2 - e_3 - e_7$. (d) Case 3b: $\Phi_{ijk} = e_4 + e_5 \neq \Phi_{ijk'} = e_4 - e_5$. (e) Case 3c: $\Phi_{ijk} = e_5 \neq \Phi_{ijk'} = -e_5$.

It is straightforward to show that the information distances are additive for the Gaussian and symmetric discrete cases using the local Markov property of graphical models. For general discrete distributions with information distance as in (9), see Lake (1994) for the proof. In the rest of the paper, we map the parameters of Gaussian and discrete distributions to an information distance matrix $\mathbf{D} = [d_{ij}]$ to unify the analyses for both cases.

3.2 Testing Inter-Node Relationships

In this section, we use Proposition 3 to ascertain child-parent and sibling (cf., Section 2.1) relationships between the variables in a latent tree-structured graphical model. To do so, for any three variables $i, j, k \in V$, we define $\Phi_{ijk} := d_{ik} - d_{jk}$ to be the difference between the information distances d_{ik} and d_{jk}. The following lemma suggests a simple procedure to identify the set of relationships between the nodes.

Lemma 4 (Sibling Grouping) *For distances d_{ij} for all $i, j \in V$ on a tree $T \in \mathcal{T}_{\geq 3}$, the following two properties on $\Phi_{ijk} = d_{ik} - d_{jk}$ hold:*

(i) *$\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i, j\}$ if and only if i is a leaf node and j is its parent.*

(i) *$\Phi_{ijk} = -d_{ij}$ for all $k \in V \setminus \{i, j\}$ if and only if j is a leaf node and i is its parent.*

(ii) *$-d_{ij} < \Phi_{ijk} = \Phi_{ijk'} < d_{ij}$ for all $k, k' \in V \setminus \{i, j\}$ if and only if both i and j are leaf nodes and they have the same parent, that is, they belong to the same sibling group.*

The proof of the lemma uses Proposition 3 and is provided in Appendix A.1. Given Lemma 4, we can first determine all the values of Φ_{ijk} for triples $i, j, k \in V$. Now we can determine the relationship between nodes i and j as follows: Fix the pair of nodes $i, j \in V$ and consider all the other nodes $k \in V \setminus \{i, j\}$. Then, there are three cases for the set $\{\Phi_{ijk} : k \in V \setminus \{i, j\}\}$:

1. $\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i, j\}$. Then, i is a leaf node and j is a parent of i. Similarly, if $\Phi_{ijk} = -d_{ij}$ for all $k \in V \setminus \{i, j\}$, j is a leaf node and i is a parent of j.

2. Φ_{ijk} is constant for all $k \in V \setminus \{i, j\}$ but *not* equal to either d_{ij} or $-d_{ij}$. Then i and j are leaf nodes and they are siblings.

3. Φ_{ijk} is not equal for all $k \in V \setminus \{i, j\}$. Then, there are three cases: Either

    (a) Nodes i and j are not siblings nor have a parent-child relationship or,

    (b) Nodes i and j are siblings but at least one of them is not a leaf or,

    (c) Nodes i and j have a parent-child relationship but the child is not a leaf.

Thus, we have a simple test to determine the relationship between i and j and to ascertain whether i and j are leaf nodes. We call the above test TestNodeRelationships. See Figure 2 for examples. By running this test for all i and j, we can determine all the relationships among all pairs of observed variables.

In the following section, we describe a recursive algorithm that is based on the above TestNodeRelationships procedure to reconstruct the entire latent tree model assuming that the true model is a latent tree and that the true distance matrix $\mathbf{D} = [d_{ij}]$ are known. In Section 5, we provide improved algorithms for the learning of latent trees again assuming that $\mathbf{D}$ is known. Subsequently, in Section 6, we develop algorithms for the *consistent* reconstruction of latent trees when information distances are unknown and we have to estimate them from the samples $\mathbf{x}_V^n$. In addition, in Section 6.6 we discuss how to extend these algorithms for the case when p_V is not necessarily tree-decomposable, that is, the original graphical model is not assumed to be a latent tree.

4. Recursive Grouping Algorithm Given Information Distances

This section is devoted to the development of the first algorithm for reconstructing latent tree models, recursive grouping (RG). At a high level, RG is a recursive procedure in which at each step, TestNodeRelationships is used to identify nodes that belong to the same family. Subsequently, RG introduces a parent node if a family of nodes (i.e., a sibling group) does not contain an observed parent. This newly introduced parent node corresponds to a hidden node in the original unknown latent tree. Once such a parent (i.e., hidden) node h is introduced, the information distances from h to all other observed nodes can be computed.

The inputs to RG are the vertex set V and the matrix of information distances $\mathbf{D}$ corresponding to a latent tree. The algorithm proceeds by recursively grouping nodes and adding hidden variables. In each iteration, the algorithm acts on a so-called active set of nodes Y, and in the process constructs a new active set Y_{new} for the next iteration.[10] The steps are as follows:

1. Initialize by setting $Y := V$ to be the set of observed variables.

2. Compute $\Phi_{ijk} = d_{ik} - d_{jk}$ for all $i, j, k \in Y$.

3. Using the TestNodeRelationships procedure, define $\{\Pi_l\}_{l=1}^L$ to be the coarsest partition[11] of Y such that for every subset Π_l (with $|\Pi_l| \geq 2$), any two nodes in Π_l are either siblings which

---

10. Note that the current active set is also used (in Step 6) after the new active set has been defined. For clarity, we also introduce the quantity Y_{old} in Steps 5 and 6.

11. Recall that a *partition* P of a set Y is a collection of nonempty subsets $\{\Pi_l \subset Y\}_{l=1}^L$ such that $\cup_{l=1}^L \Pi_l = Y$ and $\Pi_l \cap \Pi_{l'} = \emptyset$ for all $l \neq l'$. A partition P is said to be *coarser than* another partition P' if every element of P' is a subset of some element of P.

are leaf nodes or they have a parent-child relationship[12] in which the child is a leaf.[13] Note that for some l, Π_l may consist of a single node. Begin to construct the new active set by adding nodes in these single-node partitions: $Y_{\text{new}} \leftarrow \bigcup_{l:|\Pi_l|=1} \Pi_l$.

4. For each $l = 1, \ldots, L$ with $|\Pi_l| \geq 2$, if Π_l contains a parent node u, update $Y_{\text{new}} \leftarrow Y_{\text{new}} \cup \{u\}$. Otherwise, introduce a new hidden node h, connect h (as a parent) to every node in Π_l, and set $Y_{\text{new}} \leftarrow Y_{\text{new}} \cup \{h\}$.

5. Update the active set: $Y_{\text{old}} \leftarrow Y$ and $Y \leftarrow Y_{\text{new}}$.

6. For each new hidden node $h \in Y$, compute the information distances d_{hl} for all $l \in Y$ using (13) and (14) described below.

7. If $|Y| \geq 3$, return to step 2. Otherwise, if $|Y| = 2$, connect the two remaining nodes in Y with an edge then stop. If instead $|Y| = 1$, do nothing and stop.

We now describe how to compute the information distances in Step 6 for each new hidden node $h \in Y$ and all other active nodes $l \in Y$. Let $i, j \in C(h)$ be two children of h, and let $k \in Y_{\text{old}} \setminus \{i, j\}$ be any other node in the previous active set. From Lemma 4 and Proposition 3, we have that $d_{ih} - d_{jh} = d_{ik} - d_{jk} = \Phi_{ijk}$ and $d_{ih} + d_{jh} = d_{ij}$, from which we can recover the information distances between a previously active node $i \in Y_{\text{old}}$ and its new hidden parent $h \in Y$ as follows:

$$d_{ih} = \frac{1}{2} \left(d_{ij} + \Phi_{ijk} \right). \tag{13}$$

For any other active node $l \in Y$, we can compute d_{hl} using a child node $i \in C(h)$ as follows:

$$d_{hl} = \begin{cases} d_{il} - d_{ih}, & \text{if } l \in Y_{\text{old}}, \\ d_{ik} - d_{ih} - d_{lk}, & \text{otherwise, where } k \in C(l). \end{cases} \tag{14}$$

Using Equations (13) and (14), we can infer all the information distances d_{hl} between a newly introduced hidden node h to all other active nodes $l \in Y$. Consequently, we have all the distances d_{ij} between all pairs of nodes in the active set Y. It can be shown that this algorithm recovers all minimal latent trees. The proof of the following theorem is provided in Appendix A.2.

Theorem 5 (Correctness and Computational Complexity of RG) *If $T_p \in \mathcal{T}_{\geq 3}$ and the matrix of information distances **D** (between nodes in V) is available, then RG outputs the true latent tree T_p correctly in time $O(\text{diam}(T_p)m^3)$.*

We now use a concrete example to illustrate the steps involved in RG. In Figure 3(a), the original unknown latent tree is shown. In this tree, nodes $1, \ldots, 6$ are the observed nodes and h_1, h_2, h_3 are the hidden nodes. We start by considering the set of observed nodes as active nodes $Y := V = \{1, \ldots, 6\}$. Once Φ_{ijk} are computed from the given distances d_{ij}, TestNodeRelationships is used to determine that Y is partitioned into four subsets: $\Pi_1 = \{1\}, \Pi_2 = \{2, 4\}, \Pi_3 = \{5, 6\}, \Pi_4 = \{3\}$. The subsets Π_1

---

12. In an undirected tree, the parent-child relationships can be defined with respect to a root node. In this case, the node in the final active set in Step 7 before the algorithm terminates (or one of the two final nodes if $|Y| = 2$) is selected as the root node.

13. Note that since we use the active set Y in the TestNodeRelationships procedure, the leaf nodes are defined with respect to Y, that is, a node is considered as a leaf node if it has only one neighbor in Y or in the set of nodes that have not yet been in an active set.
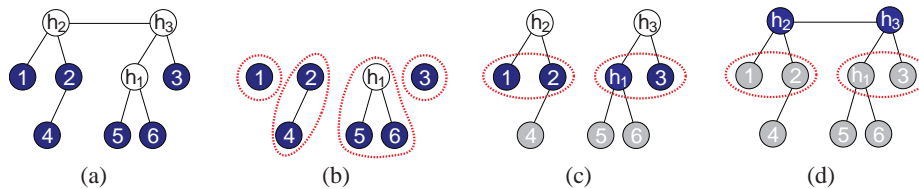
Figure 3: An illustrative example of RG. Solid nodes indicate the active set Y for each iteration. (a) Original latent tree. (b) Output after the first iteration of RG. Red dotted lines indicate the subsets Π_l in the partition of Y. (c) Output after the second iteration of RG. Note that h_3, which was introduced in the first iteration, is an active node for the second iteration. Nodes 4,5, and 6 do not belong to the current active set and are represented in grey. (d) Output after the third iteration of RG, which is same as the original latent tree.

and Π_4 contain only one node. The subset Π_3 contains two siblings that are leaf nodes. The subset Π_2 contains a parent node 2 and a child node 4, which is a leaf node. Since Π_3 does not contain a parent, we introduce a new hidden node h_1 and connect h_1 to 5 and 6 as shown in Figure 3(b). The information distances d_{5h_1} and d_{6h_1} can be computed using (13), for example, $d_{5h_1} = \frac{1}{2}(d_{56} + \Phi_{561})$. The new active set is the union of all nodes in the single-node subsets, a parent node, and a new hidden node $Y_{\text{new}} = \{1,2,3,h_1\}$. Distances among the pairs of nodes in Y_{new} can be computed using (14) (e.g., $d_{1h_1} = d_{15} - d_{5h_1}$). In the second iteration, we again use TestNodeRelationships to ascertain that Y can be partitioned into $\Pi_1 = \{1,2\}$ and $\Pi_2 = \{h_1,3\}$. These two subsets do not have parents so h_2 and h_3 are added to Π_1 and Π_2 respectively. Parent nodes h_2 and h_3 are connected to their children in Π_1 and Π_2 as shown in Figure 3(c). Finally, we are left with the active set as $Y = \{h_2,h_3\}$ and the algorithm terminates after h_2 and h_3 are connected by an edge. The hitherto unknown latent tree is fully reconstructed as shown in Figure 3(d).

A potential drawback of RG is that it involves multiple *local* operations, which may result in a high computational complexity. Indeed, from Theorem 5, the worst-case complexity is $O(m^4)$ which occurs when T_p, the true latent tree, is a hidden Markov model (HMM). This may be computationally prohibitive if m is large. In Section 5 we design an algorithm which uses a *global* pre-processing step to reduce the overall complexity substantially, especially for trees with large diameters (of which HMMs are extreme examples).

5. CLGrouping Algorithm Given Information Distances

In this section, we present CLGrouping, an algorithm for reconstructing latent trees more efficiently than RG. As in Section 4, in this section, we assume that **D** is known exactly; the extension to inexact knowledge of **D** is discussed in Section 6.5. CLGrouping is a two-step procedure, the first of which is a global pre-processing step that involves the construction of a so-called *Chow-Liu tree* (Chow and Liu, 1968) over the set of observed nodes V. This step identifies nodes that do not belong to the same sibling group. In the second step, we complete the recovery of the latent tree by applying a distance-based latent tree reconstruction algorithm (such as RG or NJ) repeatedly on smaller subsets of nodes. We review the Chow-Liu algorithm in Section 5.1, relate the Chow-Liu tree to the true latent tree in Section 5.2, derive a simple transformation of the Chow-Liu tree to

obtain the latent tree in Section 5.3 and propose CLGrouping in Section 5.4. For simplicity, we focus on the Gaussian distributions and the symmetric discrete distributions first, and discuss the extension to general discrete models in Section 5.5.

5.1 A Review of the Chow-Liu Algorithm

In this section, we review the Chow-Liu tree reconstruction procedure. To do so, define $\mathcal{T}(V)$ to be the set of trees with vertex set V and $\mathcal{P}(\mathcal{T}(V))$ to be the set of tree-structured graphical models whose graph has vertex set V, that is, every $q \in \mathcal{P}(\mathcal{T}(V))$ factorizes as in (3).

Given an arbitrary multivariate distribution $p_V(\mathbf{x}_V)$, Chow and Liu (1968) considered the following *KL-divergence minimization* problem:

$$p_{\text{CL}} := \underset{q \in \mathcal{P}(\mathcal{T}(V))}{\operatorname{argmin}} \ D(p_V \,||\, q). \tag{15}$$

That is, among all the tree-structured graphical models with vertex set V, the distribution p_{CL} is the closest one to p_V in terms of the KL-divergence. By using the factorization property in (3), we can easily verify that p_{CL} is Markov on the *Chow-Liu tree* $T_{\text{CL}} = (V, E_{\text{CL}})$ which is given by the optimization problem:[14]

$$T_{\text{CL}} = \underset{T \in \mathcal{T}(V)}{\operatorname{argmax}} \ \sum_{(i,j) \in T} I(X_i ; X_j). \tag{16}$$

In (16), $I(X_i ; X_j) = D(p(x_i, x_j) \,||\, p(x_i) \, p(x_j))$ is the *mutual information* (Cover and Thomas, 2006) between random variables X_i and X_j. The optimization in (16) is a max-weight spanning tree problem (Cormen et al., 2003) which can be solved efficiently in time $O(m^2 \log m)$ using either Kruskal's algorithm (Kruskal, 1956) or Prim's algorithm (Prim, 1957). The edge weights for the max-weight spanning tree are precisely the mutual information quantities between random variables. Note that once the optimal tree T_{CL} is formed, the parameters of p_{CL} in (15) are found by setting the pairwise distributions $p_{\text{CL}}(x_i, x_j)$ on the edges to $p_V(x_i, x_j)$, that is, $p_{\text{CL}}(x_i, x_j) = p_V(x_i, x_j)$ for all $(i, j) \in E_{\text{CL}}$. We now relate the Chow-Liu tree on the observed nodes and the information distance matrix $\mathbf{D}$.

Lemma 6 (Correspondence between T_{CL} and MST) *If p_V is a Gaussian distribution or a symmetric discrete distribution, then the Chow-Liu tree in* (16) *reduces to the minimum spanning tree (MST) where the edge weights are the information distances d_{ij}, that is,*

$$T_{\text{CL}} = \text{MST}(V; \mathbf{D}) := \underset{T \in \mathcal{T}(V)}{\operatorname{argmin}} \ \sum_{(i,j) \in T} d_{ij}. \tag{17}$$

Lemma 6, whose proof is omitted, follows because for Gaussian and symmetric discrete models, the mutual information[15] $I(X_i ; X_j)$ is a monotonically decreasing function of the information distance d_{ij}.[16] For other graphical models (e.g., non-symmetric discrete distributions), this relationship is not necessarily true. See Section 5.5 for a discussion. Note that when all nodes are observed (i.e., $W = V$), Lemma 6 reduces to Proposition 3.

---

14. In (16) and the rest of the paper, we adopt the following simplifying notation; If $T = (V, E)$ and if $(i, j) \in E$, we will also say that $(i, j) \in T$.

15. Note that, unlike information distances d_{ij}, the mutual information quantities $I(X_i ; X_j)$ do not form an additive metric on T_p.

16. For example, in the case of Gaussians, $I(X_i ; X_j) = -\frac{1}{2} \log(1 - \rho_{ij}^2)$ (Cover and Thomas, 2006).

5.2 Relationship between the Latent Tree and the Chow-Liu Tree (MST)

In this section, we relate $\text{MST}(V;\mathbf{D})$ in (17) to the original latent tree T_p. To relate the two trees, $\text{MST}(V;\mathbf{D})$ and T_p, we first introduce the notion of a surrogate node.

Definition 7 (Surrogate Node) *Given the latent tree $T_p = (W, E_p)$ and any node $i \in W$, the* surrogate node *of i with respect to V is defined as*

$$\text{Sg}(i; T_p, V) := \underset{j \in V}{\text{argmin}}\, d_{ij}.$$

Intuitively, the surrogate node of a hidden node $h \in H$ is an observed node $j \in V$ that is most strongly correlated to h. In other words, the information distance between h and j is the smallest. Note that if $i \in V$, then $\text{Sg}(i; T_p, V) = i$ since $d_{ii} = 0$. The map $\text{Sg}(i; T_p, V)$ is a many-to-one function, that is, several nodes may have the same surrogate node, and its inverse is the *inverse surrogate set of i* denoted as

$$\text{Sg}^{-1}(i; T_p, V) := \{h \in W : \text{Sg}(h; T_p, V) = i\}.$$

When the tree T_p and the observed vertex set V are understood from context, the surrogate node of h and the inverse surrogate set of i are abbreviated as $\text{Sg}(h)$ and $\text{Sg}^{-1}(i)$ respectively. We now relate the original latent tree $T_p = (W, E_p)$ to the Chow-Liu tree (also termed the MST) $\text{MST}(V;\mathbf{D})$ formed using the distance matrix $\mathbf{D}$.

Lemma 8 (Properties of the MST) *The MST in* (17) *and surrogate nodes satisfy the following properties:*

(i) The surrogate nodes of any two neighboring nodes in E_p are neighbors in the MST, that is, for all $i, j \in W$ with $\text{Sg}(i) \neq \text{Sg}(j)$,

$$(i, j) \in E_p \Rightarrow (\text{Sg}(i), \text{Sg}(j)) \in \text{MST}(V;\mathbf{D}). \tag{18}$$

(ii) If $j \in V$ and $h \in \text{Sg}^{-1}(j)$, then every node along the path connecting j and h belongs to the inverse surrogate set $\text{Sg}^{-1}(j)$.

(iii) The maximum degree of the MST satisfies

$$\Delta(\text{MST}(V;\mathbf{D})) \leq \Delta(T_p)^{1 + \frac{u}{l} \delta(T_p;V)}, \tag{19}$$

where $\delta(T_p;V)$ is the effective depth defined in (1) *and l, u are the bounds on the information distances on edges in T_p defined in* (11).

The proof of this result can be found in Appendix A.3. As a result of Lemma 8, the properties of $\text{MST}(V;\mathbf{D})$ can be expressed in terms of the original latent tree T_p. For example, in Figure 5(a), a latent tree is shown with its corresponding surrogacy relationships, and Figure 5(b) shows the corresponding MST over the observed nodes.

The properties in Lemma 8(i-ii) can also be regarded as *edge-contraction operations* (Robinson and Foulds, 1981) in the original latent tree to obtain the MST. More precisely, an edge-contraction operation on an edge $(j, h) \in V \times H$ in the latent tree T_p is defined as the "shrinking" of (j, h) to a single node whose label is the observed node j. Thus, the edge (j, h) is "contracted" to a single
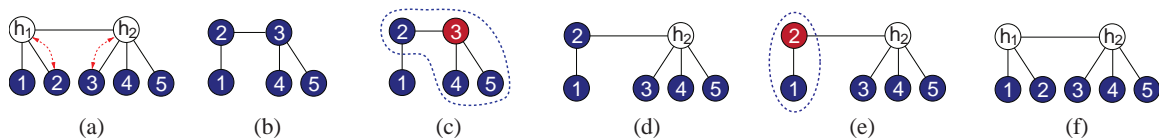
Figure 4: An illustration of CLBlind. The shaded nodes are the observed nodes and the rest are hidden nodes. The dotted lines denote surrogate mappings for the hidden nodes. (a) Original latent tree, which belongs to the class of blind latent graphical models, (b) Chow-Liu tree over the observed nodes, (c) Node 3 is the input to the blind transformation, (d) Output after the blind transformation, (e) Node 2 is the input to the blind transformation, (f) Output after the blind transformation, which is same as the original latent tree.

node j. By using Lemma 8(i-ii), we observe that the Chow-Liu tree $\mathrm{MST}(V;\mathbf{D})$ is formed by applying edge-contraction operations to each (j,h) pair for all $h \in \mathrm{Sg}^{-1}(j) \cap H$ sequentially until all pairs have been contracted to a single node j. For example, the MST in Figure 5(b) is obtained by contracting edges $(3,h_3)$, $(5,h_2)$, and then $(5,h_1)$ in the latent tree in Figure 5(a).

The properties in Lemma 8 can be used to design efficient algorithms based on transforming the MST to obtain the latent tree T_p. Note that the maximum degree of the MST, $\Delta(\mathrm{MST}(V;\mathbf{D}))$, is bounded by the maximum degree in the original latent tree. The quantity $\Delta(\mathrm{MST}(V;\mathbf{D}))$ determines the computational complexity of one of our proposed algorithms (CLGrouping) and it is small if the depth of the latent tree $\delta(T_p;V)$ is small (e.g., HMMs) and the information distances d_{ij} satisfy tight bounds (i.e., u/l is close to unity). The latter condition holds for (almost) homogeneous models in which all the information distances d_{ij} on the edges are almost equal.

5.3 Chow-Liu Blind Algorithm for a Subclass of Latent Trees

In this section, we present a simple and intuitive transformation of the Chow-Liu tree that produces the original latent tree. However, this algorithm, called Chow-Liu Blind (or CLBlind), is applicable only to a subset of latent trees called *blind latent tree-structured graphical models* $\mathcal{P}(\mathcal{T}_{\mathrm{blind}})$. Equipped with the intuition from CLBlind, we generalize it in Section 5.4 to design the CLGrouping algorithm that produces the correct latent tree structure from the MST *for all* minimal latent tree models.

If $p \in \mathcal{P}(\mathcal{T}_{\mathrm{blind}})$, then its structure $T_p = (W,E_p)$ and the distance matrix $\mathbf{D}$ satisfy the following properties:

(i) The true latent tree $T_p \in \mathcal{T}_{\geq 3}$ and all the internal nodes[17] are hidden, that is, $V = \mathrm{Leaf}(T_p)$.

(ii) The surrogate node of (i.e., the observed node with the strongest correlation with) each hidden node is one of its children, that is, $\mathrm{Sg}(h) \in \mathcal{C}(h)$ for all $h \in H$.

We now describe the CLBlind algorithm, which involves two main steps. Firstly, $\mathrm{MST}(V;\mathbf{D})$ is constructed using the distance matrix $\mathbf{D}$. Secondly, we apply the blind transformation of the Chow-Liu tree $\mathsf{BlindTransform}(\mathrm{MST}(V;\mathbf{D}))$, which proceeds as follows:

---

17. Recall that an internal node is one whose degree is greater than or equal to 2, that is, a non-leaf.

1. Identify the set of internal nodes in $\text{MST}(V;\mathbf{D})$. We perform an operation for each internal node as follows:

2. For internal node i, add a hidden node h to the tree.

3. Connect an edge between h and i (which now becomes a leaf node) and also connect edges between h and the neighbors of i in the *current* tree model.

4. Repeat steps 2 and 3 until all internal nodes have been operated on.

See Figure 4 for an illustration of CLBlind. We use the adjective *blind* to describe the transformation $\text{BlindTransform}(\text{MST}(V;\mathbf{D}))$ since it does not depend on the distance matrix $\mathbf{D}$ but uses *only* the structure of the MST. The following theorem whose proof can be found in Appendix A.4 states the correctness result for CLBlind.

Theorem 9 (Correctness and Computational Complexity of CLBlind) *If the distribution $p \in \mathcal{P}(\mathcal{T}_{\text{blind}})$ is a blind tree-structured graphical model Markov on T_p and the matrix of distances $\mathbf{D}$ is known, then CLBlind outputs the true latent tree T_p correctly in time $O(m^2 \log m)$.*

The first condition on $\mathcal{P}(\mathcal{T}_{\text{blind}})$ that all internal nodes are hidden is not uncommon in applications. For example, in phylogenetics, (DNA or amino acid) sequences of extant species at the leaves are observed, while the sequences of the extinct species are hidden (corresponding to the internal nodes), and the evolutionary (phylogenetic) tree is to be reconstructed. However, the second condition is more restrictive[18] since it implies that each hidden node is directly connected to at least one observed node and that it is closer (i.e., more correlated) to one of its observed children compared to any other observed node. If the first constraint is satisfied but not the second, then the blind transformation $\text{BlindTransform}(\text{MST}(V;\mathbf{D}))$ does not overestimate the number of hidden variables in the latent tree (the proof follows from Lemma 8 and is omitted).

Since the computational complexity of constructing the MST is $O(m^2 \log m)$ where $m = |V|$, and the blind transformation is at most linear in m, the overall computational complexity is $O(m^2 \log m)$. Thus, CLBlind is a computationally efficient procedure compared to RG, described in Section 4.

5.4 Chow-Liu Grouping Algorithm

Even though CLBlind is computationally efficient, it only succeeds in recovering latent trees for a restricted subclass of minimal latent trees. In this section, we propose an efficient algorithm, called CLGrouping that reconstructs *all* minimal latent trees. We also illustrate CLGrouping using an example. CLGrouping uses the properties of the MST as described in Lemma 8.

At a high-level, CLGrouping involves two distinct steps: Firstly, we construct the Chow-Liu tree $\text{MST}(V;\mathbf{D})$ over the set of observed nodes V. Secondly, we apply RG or NJ to reconstruct a latent subtree over the closed neighborhoods of every internal node in $\text{MST}(V;\mathbf{D})$. If RG (respectively NJ) is used, we term the algorithm CLRG (respectively CLNJ). In the rest of the section, we only describe CLRG for concreteness since CLNJ proceeds along similar lines. Formally, CLRG proceeds as follows:

1. Construct the Chow-Liu tree $\text{MST}(V;\mathbf{D})$ as in (17). Set $T = \text{MST}(V;\mathbf{D})$.

---

18. The second condition on $\mathcal{P}(\mathcal{T}_{\text{blind}})$ holds when the tree is (almost) homogeneous.
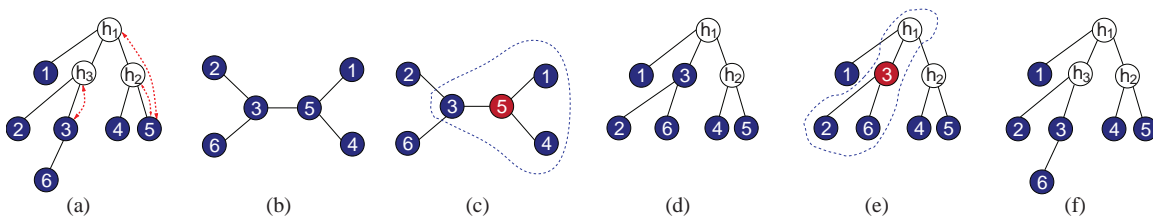
Figure 5: Illustration of CLRG. The shaded nodes are the observed nodes and the rest are hidden nodes. The dotted lines denote surrogate mappings for the hidden nodes so for example, node 3 is the surrogate of h_3. (a) The original latent tree, (b) The Chow-Liu tree (MST) over the observed nodes V, (c) The closed neighborhood of node 5 is the input to RG, (d) Output after the first RG procedure, (e) The closed neighborhood of node 3 is the input to the second iteration of RG, (f) Output after the second RG procedure, which is same as the original latent tree.

2. Identify the set of internal nodes in MST$(V;\mathbf{D})$.

3. For each internal node i, let nbd$[i;T]$ be its closed neighborhood in T and let $S = $ RG(nbd$[i;T],\mathbf{D}$) be the output of RG with nbd$[i;T]$ as the set of input nodes.

4. Replace the subtree over node set nbd$[i;T]$ in T with S. Denote the new tree as T.

5. Repeat steps 3 and 4 until all internal nodes have been operated on.

Note that the only difference between the algorithm we just described and CLNJ is Step 3 in which the subroutine NJ replaces RG. Also, observe in Step 3 that RG is only applied to a small subset of nodes which have been identified in Step 1 as possible neighbors in the true latent tree. This reduces the computational complexity of CLRG compared to RG, as seen in the following theorem whose proof is provided in Appendix A.5. Let $|J| := |V \setminus \text{Leaf}(\text{MST}(V;\mathbf{D}))| < m$ be the number of internal nodes in the MST.

Theorem 10 (Correctness and Computational Complexity of CLRG) *If the distribution $T_p \in \mathcal{T}_{\geq 3}$ is a minimal latent tree and the matrix of information distances $\mathbf{D}$ is available, then CLRG outputs the true latent tree T_p correctly in time $O(m^2 \log m + |J|\Delta^3(\text{MST}(V;\mathbf{D})))$.*

Thus, the computational complexity of CLRG is low when the latent tree T_p has a small maximum degree and a small effective depth (such as the HMM) because (19) implies that $\Delta(\text{MST}(V;\mathbf{D}))$ is also small. Indeed, we demonstrate in Section 7 that there is a significant speedup compared to applying RG over the entire observed node set V.

We now illustrate CLRG using the example shown in Figure 5. The original minimal latent tree $T_p = (W,E)$ is shown in Figure 5(a) with $W = \{1,2,\ldots,6,h_1,h_2,h_3\}$. The set of observed nodes is $V = \{1,\ldots,6\}$ and the set of hidden nodes is $H = \{h_1,h_2,h_3\}$. The Chow-Liu tree $T_{\text{CL}} = \text{MST}(V;\mathbf{D})$ formed using the information distance matrix $\mathbf{D}$ is shown in Figure 5(b). Since nodes 3 and 5 are the only internal nodes in MST$(V;\mathbf{D})$, two RG operations will be executed on the closed neighborhoods of each of these two nodes. In the first iteration, the closed neighborhood of node 5 is the input to

Latent variables	Distribution	$\text{MST}(V; \mathbf{D}) = T_{\text{CL}}$?	Structure	Parameter
Non-latent	Gaussian	✓	✓	✓
Non-latent	Symmetric Discrete	✓	✓	✓
Non-latent	General Discrete	×	✓	×
Latent	Gaussian	✓	✓	✓
Latent	Symmetric Discrete	✓	✓	✓
Latent	General Discrete	×	✓	×

Table 1: Comparison between various classes of distributions. In the last two columns, we state whether CLGrouping is consistent for learning either the structure or parameters of the model, namely whether CLGrouping is structurally consistent or risk consistent respectively (cf., Definition 2). Note that the first two cases reduce exactly to the algorithm proposed by Chow and Liu (1968) in which the edge weights are the mutual information quantities.

RG. This is shown in Figure 5(c) where $\text{nbd}[5; \text{MST}(V; \mathbf{D})] = \{1, 3, 4, 5\}$, which is then replaced by the output of RG to obtain the tree shown in Figure 5(d). In the next iteration, RG is applied to the closed neighborhood of node 3 in the current tree $\text{nbd}[3; T] = \{2, 3, 6, h_1\}$ as shown in Figure 5(e). Note that $\text{nbd}[3; T]$ includes $h_1 \in H$, which was introduced by RG in the previous iteration. The distance from h_1 to other nodes in $\text{nbd}[3; T]$ can be computed using the distance between h_1 and its surrogate node 5, which is part of the output of RG, for example, $d_{2h_1} = d_{25} - d_{5h_1}$. The closed neighborhood $\text{nbd}[3; T]$ is then replaced by the output of the second RG operation and the original latent tree T_p is obtained as shown in Figure 5(f).

Observe that the trees obtained at each iteration of CLRG can be related to the original latent tree in terms of edge-contraction operations (Robinson and Foulds, 1981), which were defined in Section 5.2. For example, the Chow-Liu tree in Figure 5(b) is obtained from the latent tree T_p in Figure 5(a) by sequentially contracting all edges connecting an observed node to its inverse surrogate set (cf., Lemma 8(ii)). Upon performing an iteration of RG, these contraction operations are inverted and new hidden nodes are introduced. For example, in Figure 5(d), the hidden nodes h_1, h_2 are introduced after performing RG on the closed neighborhood of node 5 on $\text{MST}(V; \mathbf{D})$. These newly introduced hidden nodes in fact, turn out to be the inverse surrogate set of node 5, that is, $\text{Sg}^{-1}(5) = \{5, h_1, h_2\}$. This is not merely a coincidence and we formally prove in Appendix A.5 that at each iteration, the set of hidden nodes introduced corresponds exactly to the inverse surrogate set of the internal node.

We conclude this section by emphasizing that CLGrouping (i.e., CLRG or CLNJ) has two primary advantages. Firstly, as demonstrated in Theorem 10, the structure of all minimal tree-structured graphical models can be recovered by CLGrouping in contrast to CLBlind. Secondly, it typically has much lower computational complexity compared to RG.

5.5 Extension to General Discrete Models

For general (i.e., not symmetric) discrete models, the mutual information $I(X_i; X_j)$ is in general not monotonic in the information distance d_{ij}, defined in (9).[19] As a result, Lemma 6 does not hold,

---

19. The mutual information, however, is monotonic in d_{ij} for asymmetric binary discrete models.

that is, the Chow-Liu tree T_{CL} is not necessarily the same as $\text{MST}(V; \mathbf{D})$. However, Lemma 8 does hold for all minimal latent tree models. Therefore, for general (non-symmetric) discrete models, we compute $\text{MST}(V; \mathbf{D})$ (instead of the Chow-Liu tree T_{CL} with edge weights $I(X_i; X_j)$), and apply RG or NJ to each internal node and its neighbors. This algorithm guarantees that the structure learned using CLGrouping is the same as T_p if the distance matrix $\mathbf{D}$ is available. These observations are summarized clearly in Table 1. Note that in *all* cases, the latent structure is recovered consistently.

6. Sample-Based Algorithms for Learning Latent Tree Structures

In Sections 4 and 5, we designed algorithms for the exact reconstruction of latent trees assuming that p_V is a tree-decomposable distribution and the matrix of information distances $\mathbf{D}$ is available. In most (if not all) machine learning problems, the pairwise distributions $p(x_i, x_j)$ are unavailable. Consequently, $\mathbf{D}$ is also unavailable so RG, NJ and CLGrouping as stated in Sections 4 and 5 are not directly applicable. In this section, we consider extending RG, NJ and CLGrouping to the case when only samples $\mathbf{x}_V^n$ are available. We show how to modify the previously proposed algorithms to accommodate ML estimated distances and we also provide sample complexity results for *relaxed* versions of RG and CLGrouping.

6.1 ML Estimation of Information Distances

The canonical method for deterministic parameter estimation is via maximum-likelihood (ML) (Serfling, 1980). We focus on Gaussian and symmetric discrete distributions in this section. The generalization to general discrete models is straightforward. For Gaussians graphical models, we use ML to estimate the entries of the covariance matrix,[20] that is,

$$\widehat{\Sigma}_{ij} = \frac{1}{n} \sum_{k=1}^{n} x_i^{(k)} x_j^{(k)}, \qquad \forall i, j \in V. \tag{20}$$

The ML estimate of the correlation coefficient is defined as $\widehat{\rho}_{ij} := \widehat{\Sigma}_{ij} / (\widehat{\Sigma}_{ii} \widehat{\Sigma}_{jj})^{1/2}$. The estimated information distance is then given by the analog of (8), that is, $\widehat{d}_{ij} = -\log|\widehat{\rho}_{ij}|$. For symmetric discrete distributions, we estimate the crossover probability θ_{ij} via ML as[21]

$$\widehat{\theta}_{ij} = \frac{1}{n} \sum_{k=1}^{n} \mathbb{I}\{x_i^{(k)} \neq x_j^{(k)}\}, \qquad \forall i, j \in V.$$

The estimated information distance is given by the analogue of (10), that is, $\widehat{d}_{ij} = -(K-1)\log(1 - K\widehat{\theta}_{ij})$. For both classes of models, it can easily be verified from the Central Limit Theorem and continuity arguments (Serfling, 1980) that $\widehat{d}_{ij} - d_{ij} = O_p(n^{-1/2})$, where n is the number of samples. This means that the estimates of the information distances are consistent with rate of convergence being $n^{-1/2}$. The $m \times m$ matrix of estimated information distances is denoted as $\widehat{\mathbf{D}} = [\widehat{d}_{ij}]$.

6.2 Post-processing Using Edge Contractions

For all sample-based algorithms discussed in this section, we apply a common post-processing step using edge-contraction operations. Recall from (11) that l is the minimum bound on the information

---

20. Recall that we assume that the mean of the true random vector $\mathbf{X}$ is known and equals to the zero vector so we do not need to subtract the empirical mean in (20).
21. We use $\mathbb{I}\{\cdot\}$ to denote the indicator function.

distances on edges. After learning the latent tree, if we find that there exists an edge $(i,h) \in W \times H$ with the estimated distance $\widehat{d_{ih}} < l$, then (i,h) is contracted to a single node whose label is i, that is, the hidden node h is removed and merged with node i. This edge contraction operation removes a hidden node if it is too close in information distances to another node. For Gaussian and binary variables, $\widehat{d_{ih}} = -\log|\widehat{\rho}_{ih}|$, so in our experiments, we use $l = -\log 0.9$ to contract an edge (i,h) if the correlation between the two nodes is higher than 0.9.

6.3 Relaxed Recursive Grouping (RG) Given Samples

We now show how to relax the canonical RG algorithm described in Section 4 to handle the case when only $\widehat{\mathbf{D}}$ is available. Recall that RG calls the TestNodeRelationships procedure recursively to ascertain child-parent and sibling relationships via equality tests $\Phi_{ijk} = d_{ik} - d_{jk}$ (cf., Section 3.2). These equality constraints are, in general, not satisfied with the estimated differences $\widehat{\Phi}_{ijk} := \widehat{d_{ik}} - \widehat{d_{jk}}$, which are computed based on the estimated distance in $\widehat{\mathbf{D}}$. Besides, not all estimated distances are equally accurate. Longer distance estimates (i.e., lower correlation estimates) are less accurate for a given number of samples.[22] As such, not all estimated distances can be used for testing inter-node relationships reliably. These observations motivate the following three modifications to the RG algorithm:

1. Consider using a smaller subset of nodes to test whether $\widehat{\Phi}_{ijk}$ is constant (across k).

2. Apply a threshold (inequality) test to the $\widehat{\Phi}_{ijk}$ values.

3. Improve on the robustness of the estimated distances $\widehat{d_{ih}}$ in (13) and (14) by averaging.

We now describe each of these modifications in greater detail. Firstly, in the relaxed RG algorithm, we only compute $\widehat{\Phi}_{ijk}$ for those estimated distances $\widehat{d_{ij}}$, $\widehat{d_{ik}}$ and $\widehat{d_{jk}}$ that are below a prescribed threshold $\tau > 0$ since longer distance estimates are unreliable. As such, for each pair of nodes (i,j) such that $\widehat{d_{ij}} < \tau$, associate the set

$$\mathcal{K}_{ij} := \left\{ k \in V \backslash \{i,j\} : \max\{\widehat{d_{ik}}, \widehat{d_{jk}}\} < \tau \right\}. \tag{21}$$

This is the subset of nodes in V whose estimated distances to i and j are less than τ. Compute $\widehat{\Phi}_{ijk}$ for all $k \in \mathcal{K}_{ij}$ only.

Secondly, instead of using equality tests in TestNodeRelationships to determine the relationship between nodes i and j, we relax this test and consider the statistic

$$\widehat{\Lambda}_{ij} := \max_{k \in \mathcal{K}_{ij}} \widehat{\Phi}_{ijk} - \min_{k \in \mathcal{K}_{ij}} \widehat{\Phi}_{ijk} \tag{22}$$

Intuitively, if $\widehat{\Lambda}_{ij}$ in (22) is close to zero, then nodes i and j are likely to be in the same family. Thus, declare that nodes $i, j \in V$ are in the same family if

$$\widehat{\Lambda}_{ij} < \varepsilon, \tag{23}$$

---

22. In fact, by using a large deviation result in Shen (2007, Theorem 1), we can formally show that a larger number of samples is required to get a good approximation of ρ_{ik} if it is small compared to when ρ_{ik} is large.

for another threshold $\varepsilon > 0$. Similarly, an observed node k is identified as a parent node if $|\widehat{d}_{ik} + \widehat{d}_{kj} - \widehat{d}_{ij}| < \varepsilon$ for all i and j in the same family. If such an observed node does not exists for a group of family, then a new hidden node is introduced as the parent node for the group.

Thirdly, in order to further improve on the quality of the distance estimate $\widehat{d}_{ih}$ of a newly introduced hidden node to observed nodes, we compute $\widehat{d}_{ih}$ using (13) with different pairs of $j \in C(h)$ and $k \in \mathcal{K}_{ij}$, and take the average as follows:

$$\widehat{d}_{ih} = \frac{1}{2(|C(h)|-1)} \left(\sum_{j \in C(h)} \widehat{d}_{ij} + \frac{1}{|\mathcal{K}_{ij}|} \sum_{k \in \mathcal{K}_{ij}} \widehat{\Phi}_{ijk} \right). \quad (24)$$

Similarly, for any other node $k \notin C(h)$, we compute $\widehat{d}_{kh}$ using all child nodes in $C(h)$ and $C(k)$ (if $C(k) \neq \emptyset$) as follows:

$$\widehat{d}_{kh} = \begin{cases} \frac{1}{|C(h)|} \sum_{i \in C(h)} (\widehat{d}_{ik} - \widehat{d}_{ih}), & \text{if } k \in V, \\ \frac{1}{|C(h)||C(k)|} \sum_{(i,j) \in C(h) \times C(k)} (\widehat{d}_{ij} - \widehat{d}_{ih} - \widehat{d}_{jk}), & \text{otherwise.} \end{cases} \quad (25)$$

It is easy to verify that if $\widehat{d}_{ih}$ and $\widehat{d}_{kh}$ are equal to d_{ih} and d_{kh} respectively, then (24) and (25) reduce to (13) and (14) respectively.

The following theorem shows that relaxed RG is consistent, and with appropriately chosen thresholds ε and τ, it has the sample complexity logarithmic in the number of observed variables. The proof follows from standard Chernoff bounds and is provided in Appendix A.6.

Theorem 11 (Consistency and Sample Complexity of Relaxed RG) *(i) Relaxed RG is structurally consistent for all $T_p \in \mathcal{T}_{\geq 3}$. In addition, it is risk consistent for Gaussian and symmetric discrete distributions. (ii) Assume that the effective depth is $\delta(T_p;V) = O(1)$ (i.e., constant in m) and relaxed RG is used to reconstruct the tree given $\widehat{\mathbf{D}}$. For every $\eta > 0$, there exists thresholds $\varepsilon, \tau > 0$ such that if*

$$n > C \log(m/\sqrt[3]{\eta}) \quad (26)$$

for some constant $C > 0$, the error probability for structure reconstruction in (5) is bounded above by η. If, in addition, p is a Gaussian or symmetric discrete distribution and $n > C' \log(m/\sqrt[3]{\eta})$, the error probability for distribution reconstruction in (6) is also bounded above by η. Thus, the sample complexity of relaxed RG, which is the number of samples required to achieve a desired level of accuracy, is logarithmic in m, the number of observed variables.

As we observe from (26), the sample complexity for RG is logarithmic in m for shallow trees (i.e., trees where the effective depth is constant). This is in contrast to NJ where the sample complexity is super-polynomial in the number of observed nodes for the HMM (St. John et al., 2003; Lacey and Chang, 2006).

6.3.1 RG WITH k-MEANS CLUSTERING

In practice, if the number of samples is limited, the distance estimates $\widehat{d}_{ij}$ are noisy and it is difficult to select the threshold ε in Theorem 11 to identify sibling nodes reliably. In our experiments, we employ a modified version of the k-means clustering algorithm to cluster a set of nodes with small $\widehat{\Lambda}_{ij}$, defined in (22), as a group of siblings. Recall that we test each $\widehat{\Lambda}_{ij}$ locally with a fixed threshold ε in (23). In contrast, the k-means algorithm provides a *global* scheme and circumvents the need to select the threshold ε. We adopt the *silhouette method* (Rousseeuw, 1987) with dissimilarity measure $\widehat{\Lambda}_{ij}$ to select optimal the number of clusters k.

6.4 Relaxed Neighbor-Joining Given Samples

In this section, we describe how NJ can be relaxed when the true distances are unavailable. We relax the NJ algorithm by using ML estimates of the distances $\widehat{d}_{ij}$ in place of unavailable distances d_{ij}. NJ typically assume that all observed nodes are at the leaves of the latent tree, so after learning the latent tree, we perform the post-processing step described in Section 6.2 to identify internal nodes that are observed.[23] The sample complexity of NJ is known to be $O(\exp(\text{diam}(T_p))\log m)$ (St. John et al., 2003) and thus does not scale well when the latent tree T_p has a large diameter. Comparisons between the sample complexities of other closely related latent tree learning algorithms are discussed in Atteson (1999), Erdős et al. (1999), Csűrös (2000) and St. John et al. (2003).

6.5 Relaxed CLGrouping Given Samples

In this section, we discuss how to modify CLGrouping (CLRG and CLNG) when we only have access to the estimated information distance $\widehat{\mathbf{D}}$. The relaxed version of CLGrouping differs from CLGrouping in two main aspects. Firstly, we replace the edge weights in the construction of the MST in (17) with the estimated information distances $\widehat{d}_{ij}$, that is,

$$\widehat{T}_{\text{CL}} = \text{MST}(V;\widehat{\mathbf{D}}) := \underset{T \in \mathcal{T}(V)}{\text{argmin}} \sum_{(i,j) \in T} \widehat{d}_{ij}. \tag{27}$$

The procedure in (27) can be shown to be equivalent to the learning of the ML tree structure given samples $\mathbf{x}_V^n$ if p_V is a Gaussian or symmetric discrete distribution.[24] It has also been shown that the error probability of structure learning $\Pr(\widehat{T}_{\text{CL}} \neq T_{\text{CL}})$ converges to zero exponentially fast in the number of samples n for both discrete and Gaussian data (Tan et al., 2010, 2011). Secondly, for CLRG (respectively CLNJ), we replace RG (respectively NJ) with the relaxed version of RG (respectively NJ). The sample complexity result of CLRG (and its proof) is similar to Theorem 11 and the proof is provided in Appendix A.7.

Theorem 12 (Consistency and Sample Complexity of Relaxed CLRG) *(i) Relaxed CLRG is structurally consistent for all $T_p \in \mathcal{T}_{\geq 3}$. In addition, it is risk consistent for Gaussian and symmetric discrete distributions. (ii) Assume that the effective depth is $\delta(T_p;V) = O(1)$ (i.e., constant in m). Then the sample complexity of relaxed CLRG is logarithmic in m.*

6.6 Regularized CLGrouping for Learning Latent Tree Approximations

For many practical applications, it is of interest to learn a latent tree that *approximates* the given empirical distribution. In general, introducing more hidden variables enables better fitting to the empirical distribution, but it increases the model complexity and may lead to overfitting. The Bayesian Information Criterion (Schwarz, 1978) provides a trade-off between model fitting and model complexity, and is defined as follows:

$$\text{BIC}(\widehat{T}) = \log p(\mathbf{x}_V^n;\widehat{T}) - \frac{\kappa(\widehat{T})}{2}\log n \tag{28}$$

---

23. The processing (contraction) of the internal nodes can be done in any order.
24. This follows from the observation that the ML search for the optimal structure is equivalent to the KL-divergence minimization problem in (15) with p_V replaced by $\widehat{p}_V$, the empirical distribution of $\mathbf{x}_V^n$.

where $\widehat{T}$ is a latent tree structure and $\kappa(\widehat{T})$ is the number of free parameters, which grows linearly with the number of hidden variables because $\widehat{T}$ is a tree. Here, we describe *regularized CLGrouping*, in which we use the BIC in (28) to specify a stopping criterion on the number of hidden variables added.

For each internal node and its neighbors in the Chow-Liu tree, we use relaxed NJ or RG to learn a latent subtree. Unlike in regular CLGrouping, before we integrate this subtree into our model, we compute its BIC score. Computing the BIC score requires estimating the maximum likelihood parameters for the models, so for general discrete distributions, we run the EM algorithm on the subtree to estimate the parameters.[25] After we compute the BIC scores for all subtrees corresponding to all internal nodes in the Chow-Liu tree, we choose the subtree that results in the highest BIC score and incorporate that subtree into the current tree model.

The BIC score can be computed efficiently on a tree model with a few hidden variables. Thus, for computational efficiency, each time a set of hidden nodes is added to the model, we generate samples of hidden nodes conditioned on the samples of observed nodes, and use these augmented samples to compute the BIC score approximately when we evaluate the next subtree to be integrated in the model.

If none of the subtrees increases the BIC score (i.e., the current tree has the highest BIC score), the procedure stops and outputs the estimated latent tree. Alternatively, if we wish to learn a latent tree with a given number of hidden nodes, we can used the BIC-based procedure mentioned in the previous paragraph to learn subtrees until the desired number of hidden nodes is introduced. Depending on whether we use NJ or RG as the subroutine, we denote the specific regularized CLGrouping algorithm as *regCLNJ* or *regCLRG*.

This approach of using an approximation of the BIC score has been commonly used to learn a graphical model with hidden variables (Elidan and Friedman, 2005; Zhang and Kočka, 2004). However, for these algorithms, the BIC score needs to be evaluated for a large subset of nodes, whereas in CLGrouping, the Chow-Liu tree among observed variables prunes out many subsets, so we need to evaluate BIC scores only for a small number of candidate subsets (the number of internal nodes in the Chow-Liu tree).

7. Experimental Results

In this section, we compare the performances of various latent tree learning algorithms. We first show simulation results on synthetic data sets with known latent tree structures to demonstrate the consistency of our algorithms. We also analyze the performance of these algorithms when we change the underlying latent tree structures. Then, we show that our algorithms can approximate arbitrary multivariate probability distributions with latent trees by applying them to two real-world data sets, a monthly stock returns example and the 20 newsgroups data set.

7.1 Simulations using Synthetic Data Sets

In order to analyze the performances of different tree reconstruction algorithms, we generate samples from known latent tree structures with varying sample sizes and apply reconstruction algorithms. We compare the neighbor-joining method (NJ) (Saitou and Nei, 1987) with recursive

---

25. Note that for Gaussian and symmetric discrete distributions, the model parameters can be recovered from information distances directly using (8) or (10).
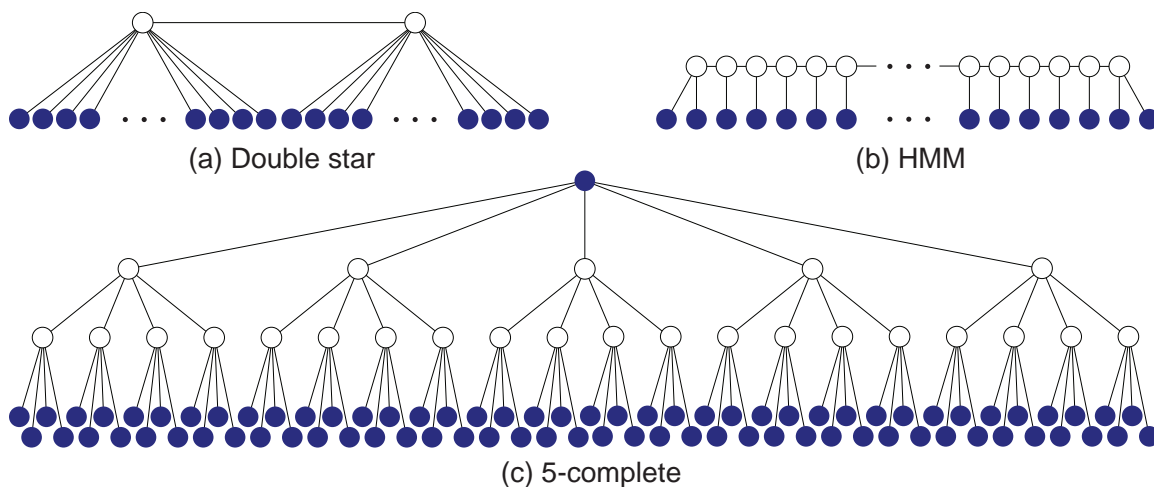
Figure 6: Latent tree structures used in our simulations.

grouping (RG), Chow-Liu Neighbor Joining (CLNJ), and Chow-Liu Recursive Grouping (CLRG). Since the algorithms are given only samples of observed variables, we use the sample-based algorithms described in Section 6. For all our experiments, we use the same edge-contraction threshold $\varepsilon' = -\log 0.9$ (see Sections 6.4 and 6.5), and set τ in Section 6.3 to grow logarithmically with the number of samples.

Figure 6 shows the three latent tree structures used in our simulations. The double-star has 2 hidden and 80 observed nodes, the HMM has 78 hidden and 80 observed nodes, and the 5-complete tree has 25 hidden and 81 observed nodes including the root node. For simplicity, we present simulation results only on Gaussian models but note that the behavior on discrete models is similar. All correlation coefficients on the edges ρ_{ij} were independently drawn from a uniform distribution supported on $[0.2, 0.8]$. The performance of each method is measured by averaging over 200 independent runs with different parameters. We use the following performance metrics to quantify the performance of each algorithm in Figure 7:

  (i) **Structure recovery error rate**: This is the proportion of times that the proposed algorithm fails to recover the true latent tree structure. Note that this is a very strict measure since even a single wrong hidden node or misplaced edge results in an error for the entire structure.

 (ii) **Robinson Foulds metric** (Robinson and Foulds, 1981): This popular phylogenetic tree-distortion metric computes the number of graph transformations (edge contraction or expansion) needed to be applied to the estimated graph in order to get the correct structure. This metric quantifies the difference in the structures of the estimated and true models.

(iii) **Error in the number of hidden variables**: We compute the average number of hidden variables introduced by each method and plot the absolute difference between the average estimated hidden variables and the number of hidden variables in the true structure.
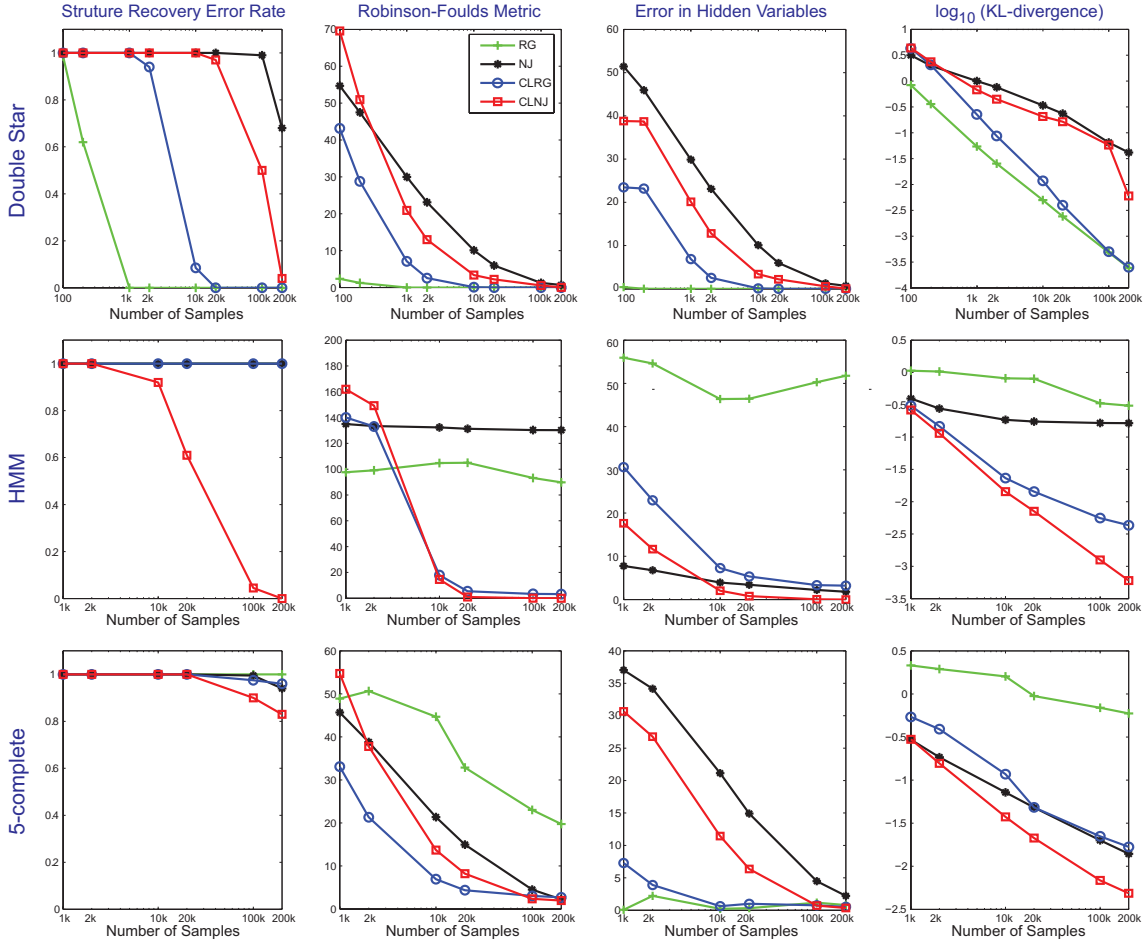
Figure 7: Performance of RG, NJ, CLRG, and CLNJ for the latent trees shown in Figure 6.

(iv) **KL-divergence** $D(p_V \,||\, \widehat{p}_V^n)$: This is a measure of the distance between the estimated and the true models over the set of observed nodes V.[26]

We first note that from the structural error rate plots that the double star is the easiest structure to recover and the 5-complete tree is the hardest. In general, given the same number of observed variables, a latent tree with more hidden variables or larger effective depth (see Section 2) is more difficult to recover.

For the double star, RG clearly outperforms all other methods. With only 1,000 samples, it recovers the true structure exactly in all 200 runs. On the other hand, CLGrouping performs significantly better than RG for the HMM. There are two reasons for such performance differences. Firstly, for Gaussian distributions, it was shown (Tan et al., 2010) that given the same number of variables and their samples, the Chow-Liu algorithm is most accurate for a chain and least accurate for a star. Since the Chow-Liu tree of a latent double star graph is close to a star, and the Chow-Liu

---

26. Note that this is not the same quantity as in (6) because if the number of hidden variables is estimated incorrectly, $D(p\,||\,\widehat{p}^n)$ is infinite so we plot $D(p_V\,||\,\widehat{p}_V^n)$ instead. However, for Gaussian and symmetric discrete distributions, $D(p\,||\,\widehat{p}^n)$ converges to zero in probability since the number of hidden variables is estimated correctly asymptotically.

	RG	NJ	CLRG	CLNJ
HMM	10.16	0.02	0.10	0.05
5-complete	7.91	0.02	0.26	0.06
Double star	1.43	0.01	0.76	0.20

Table 2: Average running time of each algorithm in seconds.

tree of a latent HMM is close to a chain, the Chow-Liu tree tend to be more accurate for the HMM than for the double star. Secondly, the internal nodes in the Chow-Liu tree of the HMM tend to have small degrees, so we can apply RG or NJ to a very small neighborhood, which results in a significant improvement in both accuracy and computational complexity.

Note that NJ is particularly poor at recovering the HMM structure. In fact, it has been shown that even if the number of samples grows polynomially with the number of observed variables (i.e., $n = O(m^B)$ for any $B > 0$), it is insufficient for NJ to recover HMM structures (Lacey and Chang, 2006). The 5-complete tree has two layers of hidden nodes, making it very difficult to recover the exact structure using any method. CLNJ has the best structure recovery error rate and KL divergence, while CLRG has the smallest Robinson-Foulds metric.

Table 2 shows the running time of each algorithm averaged over 200 runs and all sample sizes. All algorithms are implemented in MATLAB. As expected, we observe that CLRG is significantly faster than RG for HMM and 5-complete graphs. NJ is fastest, but CLNJ is also very efficient and leads to much more accurate reconstruction of latent trees.

Based on the simulation results, we conclude that for a latent tree with a few hidden variables, RG is most accurate, and for a latent tree with a large diameter, CLNJ performs the best. A latent tree with multiple layers of hidden variables is more difficult to recover correctly using any method, and CLNJ and CLRG outperform NJ and RG.

7.2 Monthly Stock Returns

In this and the next section, we test our algorithms on real-world data sets. The probability distributions that govern these data sets of course do not satisfy the assumptions required for consistent learning of the latent tree models. Nonetheless the experiments here demonstrate that our algorithms are also useful in *approximating* complex probability distributions by latent models in which the hidden variables have the same domain as the observed ones.

We apply our latent tree learning algorithms to model the dependency structure of monthly stock returns of 84 companies in the S&P 100 stock index.[27] We use the samples of the monthly returns from 1990 to 2007. As shown in Table 3 and Figure 8, CLNJ achieves the highest log-likelihood and BIC scores. NJ introduces more hidden variables than CLNJ and has lower log-likelihoods, which implies that starting from a Chow-Liu tree helps to get a better latent tree approximation. Figure 11 shows the latent tree structure learned using the CLNJ method. Each observed node is labeled with the ticker of the company. Note that related companies are closely located on the tree. Many hidden nodes can be interpreted as industries or divisions. For example, h1 has Verizon, Sprint, and T-mobile as descendants, and can be interpreted as the telecom industry, and h3 correspond to the technology division with companies such as Microsoft, Apple, and IBM as descendants. Nodes h26 and h27 group commercial banks together, and h25 has all retail stores as child nodes.

---

27. We disregard 16 companies that have been listed on S&P 100 only after 1990.

	Log-Likelihood	BIC	# Hidden	# Parameters	Time (secs)
CL	-13,321	-13,547	0	84	0.15
NJ	-12,400	-12,747	45	129	**0.02**
RG	-14,042	-14,300	12	96	21.15
CLNJ	**-11,990**	**-12,294**	29	113	0.24
CLRG	-12,879	-13,174	26	110	0.40

Table 3: Comparison of the log-likelihood, BIC, number of hidden variables introduced, number of parameters, and running time for the monthly stock returns example.
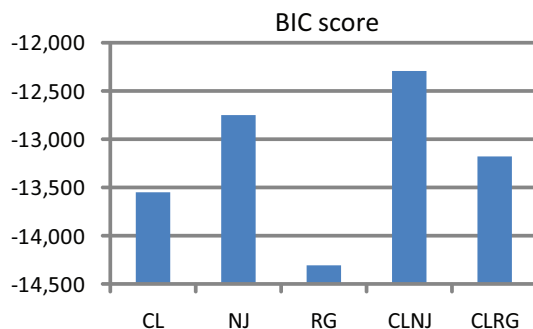
Figure 8: Plot of BIC scores for the monthly stock returns example.

7.3 20 Newsgroups with 100 Words

For our last experiment, we apply our latent tree learning algorithms to the 20 Newsgroups data set with 100 words.[28] The data set consists of 16,242 binary samples of 100 words, indicating whether each word appears in each posting or not. In addition to the Chow-Liu tree (CL), NJ, RG, CLNJ, and CLRG, we also compare the performances with the regCLNJ and regCLRG (described in Section 6.6), the latent cluster model (LCM) (Lazarsfeld and Henry, 1968), and BIN, which is a greedy algorithm for learning latent trees (Harmeling and Williams, 2010).

Table 4 shows the performance of different algorithms, and Figure 9 plots the BIC score. We use the MATLAB code (a small part of it is implemented in C) provided by Harmeling and Williams (2010)[29] to run LCM and BIN. Note that although LCM has only one hidden node, the hidden node has 16 states, resulting in many parameters. We also tried to run the algorithm by Chen et al. (2008), but their JAVA implementation on this data set did not complete even after several days. For NJ, RG, CLNJ, and CLRG, we learned the structures using only information distances (defined in (9)) and then used the EM algorithm to fit the parameters. For regCLNJ and regCLRG, the model parameters are learned during the structure learning procedure by running the EM algorithm locally, and once the structure learning is over, we refine the parameters by running the EM algorithm for the entire latent tree. All methods are implemented in MATLAB except the E-step of the EM algorithm, which is implemented in C++.

---

28. The data set can be found at `http://cs.nyu.edu/~roweis/data/20news_w100.mat`.
29. Code can be found at `http://people.kyb.tuebingen.mpg.de/harmeling/code/ltt-1.3.tar`.

	Log-Likelihood	BIC	Hidden	Params	Time (s)		
					Total	Structure	EM
CL	-238,713	-239,677	0	199	8.9	-	-
LCM	**-223,096**	**-230,925**	1	1,615	8,835.9	-	-
BIN	-232,042	-233,952	98	394	3,022.6	-	-
NJ	-230,575	-232,257	74	347	1,611.2	3.3	1,608.2
RG	-239,619	-240,875	30	259	927.1	30.8	896.4
CLNJ	-230,858	-232,540	74	347	1,479.6	2.7	1,476.8
CLRG	-231,279	-232,738	51	301	1,224.6	3.1	1,224.6
regCLNJ	-235,326	-236,553	27	253	630.8	449.7	181.1
regCLRG	-234,012	-235,229	26	251	606.9	493.0	113.9

Table 4: Comparison between various algorithms on the newsgroup set.
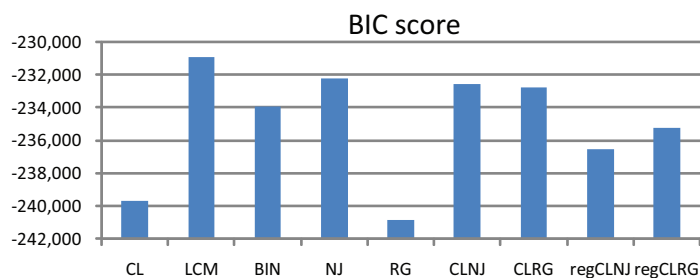
Figure 9: The BIC scores of various algorithms on the newsgroup set.

Despite having many parameters, the models learned via LCM have the best BIC score. However, it does not reveal any interesting structure and is computationally more expensive to learn. In addition, it may result in overfitting. In order to show this, we split the data set randomly and use half as the training set and the other half as the test set. Table 5 shows the performance of applying the latent trees learned from the training set to the test set, and Figure 10 shows the log-likelihood on the training and the test sets. For LCM, the test log-likelihood drops significantly compared to the training log-likelihood, indicating that LCM is overfitting the training data. NJ, CLNJ, and CLRG achieve high log-likelihood scores on the test set. Although regCLNJ and regCLRG do not result in a better BIC score, they introduce fewer hidden variables, which is desirable if we wish to learn a latent tree with small computational complexity, or if we wish to discover a few hidden variables that are meaningful in explaining the dependencies of observed variables.

Figure 12 shows the latent tree structure learned using regCLRG from the entire data set. Many hidden variables in the tree can be roughly interpreted as topics—h5 as sports, h9 as computer technology, h13 as medical, etc. Note that some words have multiple meanings and appear in different topics—for example, `program` can be used in the phrase "space program" as well as "computer program", and `win` may indicate the windows operating system or winning in sports games.

	Train		Test		Hidden	Params	Time (s)		
	Log-Like	BIC	Log-Like	BIC			Total	Struct	EM
CL	-119,013	-119,909	-120,107	-121,003	0	199	3.0	-	-
LCM	**-112,746**	-117,288	-116,884	-120,949	1	1,009	3,197.7	-	-
BIN	-117,172	-118,675	-117,957	-119,460	78	334	1,331.3	-	-
NJ	-115,319	**-116,908**	**-116,011**	-117,600	77	353	802.8	1.3	801.5
RG	-118,280	-119,248	-119,181	-120,149	8	215	137.6	7.6	130.0
CLNJ	-115,372	-116,987	-116,036	-117,652	80	359	648.0	1.5	646.5
CLRG	-115,565	-116,920	-116,199	**-117,554**	51	301	506.0	1.7	504.3
regCLNJ	-117,723	-118,924	-118,606	-119,808	34	267	425.5	251.3	174.2
regCLRG	-116,980	-118,119	-117,652	-118,791	27	253	285.7	236.5	49.2

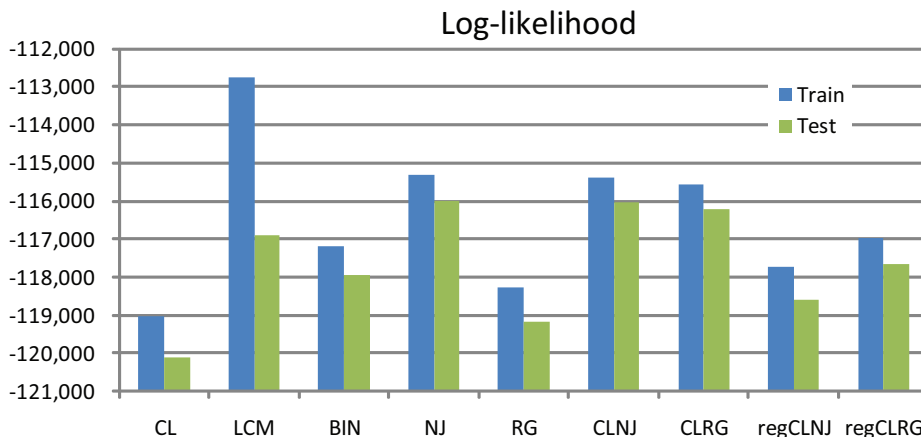Table 5: Comparison between various algorithms on the newsgroup data set with a train/test split.

Figure 10: Train and test log-likelihood scores of various algorithms on the newsgroup data set with a train/test split.

8. Discussion and Conclusion

In this paper, we proposed algorithms to learn a latent tree model from the information distances of observed variables. Our first algorithm, recursive grouping (RG), identifies sibling and parent-child relationships and introduces hidden nodes recursively. Our second algorithm, CLGrouping, maintains a tree in each iteration and adds hidden variables by locally applying latent-tree learning procedures such as recursive grouping. These algorithms are structurally consistent (and risk consistent as well in the case of Gaussian and discrete symmetric distributions), and have sample complexity logarithmic in the number of observed variables for constant depth trees.

Using simulations on synthetic data sets, we showed that RG performs well when the number of hidden variables is small, while CLGrouping performs significantly better than other algorithms when there are many hidden variables in the latent tree. We compared our algorithms to other EM-based approaches and the neighbor-joining method on real-world data sets, under both Gaussian and discrete data modeling. Our proposed algorithms show superior results in both accuracy (measured by KL-divergence and graph distance) and computational efficiency. In addition, we introduced regularized CLGrouping, which can learn a latent tree approximation by trading off model complexity (number of hidden nodes) with data fidelity. This is very relevant for
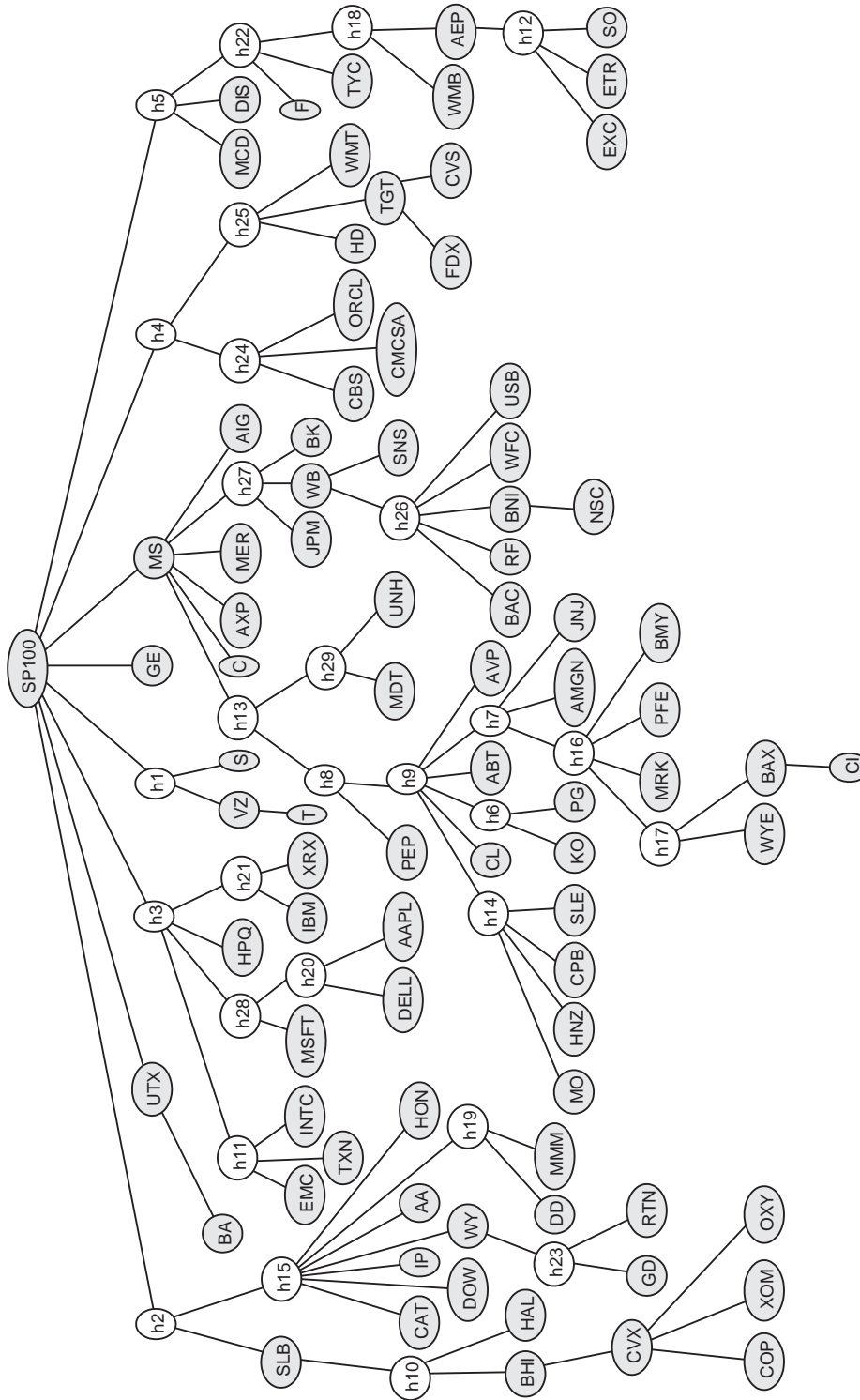
Figure 11: Tree structure learned from monthly stock returns using CLNJ.

Figure 12: Tree structure learned from 20 newsgroup data set using regCLRG.

practical implementation on real-world data sets. In future, we plan to develop a unified framework for learning latent trees where each random variable (node) may be continuous or discrete. The MATLAB implementation of our algorithms can be downloaded from the project webpage `http://people.csail.mit.edu/myungjin/latentTree.html`.

Acknowledgments

Appendix A. Proofs

In this appendix, we provide proofs for the theorems presented in the paper.

A.1 Proof of Lemma 4: Sibling Grouping

We prove statement (i) in Lemma 4 using (12) in Proposition 3. Statement (ii) follows along similar lines and its proof is omitted for brevity.

If : From the additive property of information distances in (12), if i is a leaf node and j is its parent, $d_{ik} = d_{ij} + d_{jk}$ and thus $\Phi_{ijk} = d_{ij}$ for all $k \neq i, j$.

Only If: Now assume that $\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i, j\}$. In order to prove that i is a leaf node and j is its parent, assume to the contrary, that i and j are not connected with an edge, then there exists a node $u \neq i, j$ on the path connecting i and j. If $u \in V$, then let $k = u$. Otherwise, let k be an observed node in the subtree away from i and j (see Figure 13(a)), which exists since $T_p \in \mathcal{T}_{\geq 3}$. By the additive property of information distances in (12) and the assumption that all distances are positive,

$$d_{ij} = d_{iu} + d_{uj} > d_{iu} - d_{uj} = d_{ik} - d_{kj} = \Phi_{ijk}$$

which is a contradiction. If i is not a leaf node in T_p, then there exist a node $u \neq i, j$ such that $(i, u) \in E_p$. Let $k = u$ if $u \in V$, otherwise, let k be an observed node in the subtree away from i and j (see Figure 13(b)). Then,

$$\Phi_{ijk} = d_{ik} - d_{jk} = -d_{ij} < d_{ij},$$

which is again a contradiction. Therefore, $(i, j) \in E_p$ and i is a leaf node. $\qquad \square$

A.2 Proof of Theorem 5: Correctness and Computational Complexity of RG

The correctness of RG follows from the following observations: Firstly, from Proposition 3, for all i, j in the active set Y, the information distances d_{ij} can be computed exactly with Equations (13) and (14). Secondly, at each iteration of RG, the sibling groups within Y are identified correctly using the information distances by Lemma 4. Since the new parent node added to a partition that does not
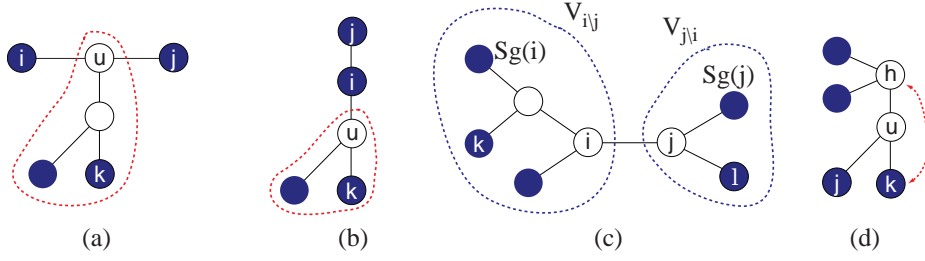
Figure 13: Shaded nodes indicate observed nodes and the rest indicate hidden nodes. (a),(b) Figures for Proof of Lemma 4. Dashed red line represent the subtrees away from i and j. (c) Figure for Proof of Lemma 8(i). (d) Figure for Proof of Lemma 8(iI)

contain an observed parent corresponds to a hidden node (in the original latent tree), a subforest of T_p is recovered at each iteration, and when $|Y| \leq 2$, and the entire latent tree is recovered.

The computational complexity follows from the fact there are a maximum of $O(m^3)$ differences $\Phi_{ijk} = d_{ik} - d_{jk}$ that we have to compute at each iteration of RG. Furthermore, there are at most $\mathrm{diam}(T_p)$ subsets in the coarsest partition (cf., step 3) of Y at the first iteration, and the number of subsets reduce at least by 2 from one iteration to the next due to the assumption that $T_p \in \mathcal{T}_{\geq 3}$. This proves the claim that the computational complexity is upper bounded by $O(\mathrm{diam}(T_p)m^3)$. $\qquad\square$

A.3 Proof of Lemma 8: Properties of the MST

(i) For an edge $(i, j) \in E_p$ such that $\mathrm{Sg}(i) \neq \mathrm{Sg}(j)$, let $V_{i \setminus j} \subset V$ and $V_{j \setminus i} \subset V$ denote observed nodes in the subtrees obtained by the removal of edge (i, j), where the former includes node i and excludes node j and vice versa (see Figure 13(c)). Using part (ii) of the lemma and the fact that $\mathrm{Sg}(i) \neq \mathrm{Sg}(j)$, it can be shown that $\mathrm{Sg}(i) \in V_{i \setminus j}$ and $\mathrm{Sg}(j) \in V_{j \setminus i}$. Since (i, j) lies on the unique path from k to l on T_p, for all observed nodes $k \in V_{i \setminus j}, l \in V_{j \setminus i}$, we have

$$d_{kl} = d_{ki} + d_{ij} + d_{jl} \geq d_{\mathrm{Sg}(i),i} + d_{ij} + d_{\mathrm{Sg}(j),j} = d_{\mathrm{Sg}(i),\mathrm{Sg}(j)},$$

where the inequality is from the definition of surrogacy and the final equality uses the fact that $\mathrm{Sg}(i) \neq \mathrm{Sg}(j)$. By using the property of the MST that $(\mathrm{Sg}(i), \mathrm{Sg}(j))$ is the shortest edge from $V_{i \setminus j}$ to $V_{j \setminus i}$, we have (18).

(ii) First assume that we have a tie-breaking rule consistent across all hidden nodes so that if $d_{uh} = d_{vh} = \min_{i \in V} d_{ih}$ and $d_{uh'} = d_{vh'} = \min_{i \in V} d_{ih'}$ then both h and h' choose the same surrogate node. Let $j \in V, h \in \mathrm{Sg}^{-1}(j)$, and let u be a node on the path connecting h and j (see Figure 13(d)). Assume that $\mathrm{Sg}(u) = k \neq j$. If $d_{uj} > d_{uk}$, then

$$d_{hj} = d_{hu} + d_{uj} > d_{hu} + d_{uk} = d_{hk},$$

which is a contradiction since $j = \mathrm{Sg}(h)$. If $d_{uj} = d_{uk}$, then $d_{hj} = d_{hk}$, which is again a contradiction to the consistent tie-breaking rule. Thus, the surrogate node of u is j.

(iii) First we claim that

$$|\mathrm{Sg}^{-1}(i)| \leq \Delta(T_p)^{\frac{u}{l}\delta(T_p;V)}. \tag{29}$$

To prove this claim, let γ be the longest (worst-case) graph distance of any hidden node $h \in H$ from its surrogate, that is,

$$\gamma := \max_{h \in H} |\text{Path}(h, \text{Sg}(h); T_p)|. \tag{30}$$

From the degree bound, for each $i \in V$, there are at most $\Delta(T_p)^\gamma$ hidden nodes that are within the graph distance of γ,[30] so

$$|\text{Sg}^{-1}(i)| \leq \Delta(T_p)^\gamma \tag{31}$$

for all $i \in V$. Let $d^* := \max_{h \in H} d_{h, \text{Sg}(h)}$ be the longest (worst-case) information distance between a hidden node and its surrogate. From the bounds on the information distances, $l\gamma \leq d^*$. In addition, for each $h \in H$, let $z(h) := \text{argmin}_{j \in V} |\text{Path}((h, j); T_p)|$ be the observed node that is closest to h in graph distance. Then, by definition of the effective depth, $d_{h, \text{Sg}(h)} \leq d_{h, z(h)} \leq u\delta$ for all $h \in H$, and we have $d^* \leq u\delta$. Since $l\gamma \leq d^* \leq u\delta$, we also have

$$\gamma \leq u\delta/l. \tag{32}$$

Combining this result with (31) establishes the claim in (29). Now consider

$$\Delta(\text{MST}(V; \mathbf{D})) \overset{(a)}{\leq} \Delta(T_p) \max_{i \in V} |\text{Sg}^{-1}(i)| \overset{(b)}{\leq} \Delta(T_p)^{1 + \frac{u}{l}\delta(T_p; V)}$$

where (a) is a result of the application of (18) and (b) results from (29). This completes the proof of the claim in (19) in Lemma 8. □

A.4 Proof of Theorem 9: Correctness and Computational Complexity of CLBlind

It suffices to show that the Chow-Liu tree $\text{MST}(V; \mathbf{d})$ is a transformation of the true latent tree T_p (with parameters such that $p \in \mathcal{P}(\mathcal{T}_{\text{blind}})$) as follows: contract the edge connecting each hidden variable h with its surrogate node $\text{Sg}(h)$ (one of its children and a leaf by assumption). Note that the blind transformation on the MST is merely the inverse mapping of the above. From (18), all the children of a hidden node h, except its surrogate $\text{Sg}(h)$, are neighbors of its surrogate node $\text{Sg}(h)$ in $\text{MST}(V; \mathbf{d})$. Moreover, these children of h which are not surrogates of any hidden nodes are leaf nodes in the MST. Similarly for two hidden nodes $h_1, h_2 \in H$ such that $(h_1, h_2) \in E_p$, $(\text{Sg}(h_1), \text{Sg}(h_2)) \in \text{MST}(V; \mathbf{d})$ from Lemma 8(i). Hence, CLBlind outputs the correct tree structure T_p. The computational complexity follows from the fact that the blind transformation is linear in the number of internal nodes, which is less than the number of observed nodes, and that learning the Chow-Liu tree takes $O(m^2 \log m)$ operations. □

A.5 Proof of Theorem 10: Correctness and Computational Complexity of CLRG

We first define some new notations.

Notation: Let $I := V \setminus \text{Leaf}(\text{MST}(V; \mathbf{d}))$ be the set of internal nodes. Let $v^r \in I$ be the internal node visited at iteration r, and let H^r be all hidden nodes in the inverse surrogate set $\text{Sg}^{-1}(v^r)$, that is, $H^r = \text{Sg}^{-1}(v^r) \setminus \{v^r\}$. Let $A^r := \text{nbd}[v^r; T^{r-1}]$, and hence A^r is the node set input to the recursive grouping routine at iteration r, and let $\text{RG}(A^r, \mathbf{d})$ be the output latent tree learned by recursive grouping. Define T^r as the tree output at the end of r iterations of CLGrouping. Let $V^r := \{v^{r+1}, v^{r+2}, \ldots, v^{|I|}\}$ be the set of internal nodes that have not yet been visited by CLGrouping

---

30. The maximum size of the inverse surrogate set in (30) is attained by a $\Delta(T_p)$-ary complete tree.
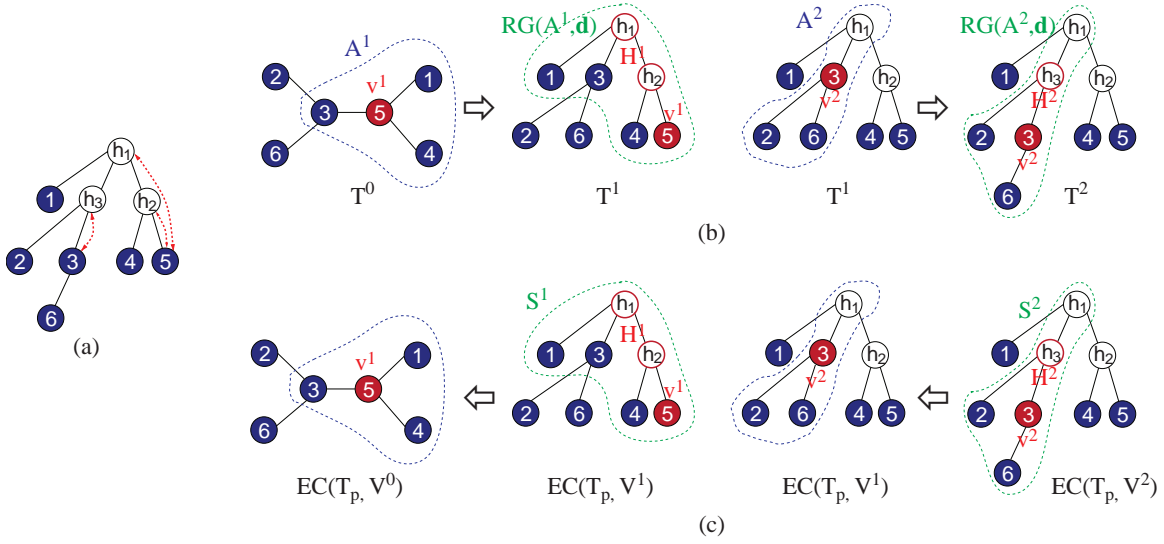
Figure 14: Figure for Proof of Theorem 10. (a) Original latent tree. (b) Illustration of CLGrouping. (c) Illustration of the trees constructed using edge contractions.

at the end of r iterations. Let $EC(T_p, V^r)$ be the tree constructed using edge contractions as follows: in the latent tree T_p, we contract edges corresponding to each node $u \in V^r$ and all hidden nodes in its inverse surrogate set $Sg^{-1}(u)$. Let S^r be a subtree of $EC(T_p, V^r)$ spanning v^r, H^r and their neighbors.

For example, in Figure 14, the original latent tree T_p is shown in Figure 14(a), and T^0, T^1, T^2 are shown in Figure 14(b). The set of internal nodes is $I = \{3, 5\}$. In the first iteration, $v^1 = 5$, $A^1 = \{1, 3, 4, 5\}$ and $H^1 = \{h_1, h_2\}$. In the second iteration, $v^2 = 3$, $A^2 = \{2, 3, 6, h_1\}$ and $H^1 = \{h_3\}$. $V^0 = \{3, 5\}$, $V^1 = \{3\}$, and $V^2 = \emptyset$, and in Figure 14(c), we show $EC(T_p, V^0)$, $EC(T_p, V^1)$, and $EC(T_p, V^2)$. In $EC(T_p, V^1)$, S^1 is the subtree spanning $5, h_1, h_2$ and their neighbors, that is, $\{1, 3, 4, 5, h_1, h_2\}$. In $EC(T_p, V^2)$, S^2 is the subtree spanning $3, h_3$ and their neighbors, that is, $\{2, 3, 6, h_1, h_3\}$. Note that $T^0 = EC(T_p, V^0)$, $T^1 = EC(T_p, V^1)$, and $T^2 = EC(T_p, V^2)$; we show below that this holds for all CLGrouping iterations in general.

We prove the theorem by induction on the iterations $r = 1, \ldots, |I|$ of the CLGrouping algorithm.

Induction Hypothesis: At the end of k iterations of CLGrouping, the tree obtained is

$$T^k = EC(T_p, V^k), \qquad \forall k = 0, 1, \ldots, |I|. \tag{33}$$

In words, the latent tree after k iterations of CLGrouping can be constructed by contracting each surrogate node in T_p that has not been visited by CLGrouping with its inverse surrogate set. Note that $V^{|I|} = \emptyset$ and $EC(T_p, V^{|I|})$ is equivalent to the original latent tree T_p. Thus, if the above induction in (33) holds, then the output of CLGrouping $T^{|I|}$ is the original latent tree.

Base Step $r = 0$: The claim in (33) holds since $V^0 = I$ and the input to the CLGrouping procedure is the Chow-Liu tree $MST(V; \mathbf{D})$, which is obtained by contracting all surrogate nodes and their inverse surrogate sets (see Section 5.2).

Induction Step: Assume (33) is true for $k = 1, \ldots, r - 1$. Now consider $k = r$.

We first compare the two latent trees $\text{EC}(T_p, V^r)$ and $\text{EC}(T_p, V^{r-1})$. By the definition of EC, if we contract edges with v^r and the hidden nodes in its inverse surrogate set H^r on the tree $\text{EC}(T_p, V^r)$, then we obtain $\text{EC}(T_p, V^{r-1})$, which is equivalent to T^{r-1} by the induction assumption. Note that as shown in Figure 14, this transformation is local to the subtree S^r: contracting v^r with H^r on $\text{EC}(T_p, V^r)$ transforms S^r into a star graph with v^r at its center and the hidden nodes H^r removed (contracted with v^r).

Recall that the CLGrouping procedure replaces the induced subtree of A^r in T^{r-1} (which is precisely the star graph mentioned above by the induction hypothesis) with $\text{RG}(A^r, \mathbf{d})$ to obtain T^r. Thus, to prove that $T^r = \text{EC}(T_p, V^r)$, we only need to show that RG reverses the edge-contraction operations on v^r and H^r, that is, the subtree $S^r = \text{RG}(A^r, \mathbf{d})$. We first show that $S^r \in \mathcal{T}_{\geq 3}$, that is, it is identifiable (minimal) when A^r is the set of visible nodes. This is because an edge contraction operation does not decrease the degree of any existing nodes. Since $T_p \in \mathcal{T}_{\geq 3}$, all hidden nodes in $\text{EC}(T_p, V^r)$ have degrees equal to or greater than 3, and since we are including all neighbors of H^r in the subtree S^r, we have $S^r \in \mathcal{T}_{\geq 3}$. By Theorem 5, RG reconstructs all latent trees in $\mathcal{T}_{\geq 3}$ and hence, $S^r = \text{RG}(A^r, \mathbf{d})$.

The computational complexity follows from the corresponding result in recursive grouping. The Chow-Liu tree can be constructed with $O(m^2 \log m)$ complexity. The recursive grouping procedure has complexity $\max_r |A^r|^3$ and $\max_r |A^r| \leq \Delta(\text{MST}(V; \widehat{d}))$. $\qquad\square$

A.6 Proof of Theorem 11: Consistency and Sample Complexity of Relaxed RG

(i) Structural consistency follows from Theorem 5 and the fact that the ML estimates of information distances $\widehat{d_{ij}}$ approach d_{ij} (in probability) for all $i, j \in V$ as the number of samples tends to infinity.

Risk consistency for Gaussian and symmetric discrete distributions follows from structural consistency. If the structure is correctly recovered, we can use the equations in (13) and (14) to infer the information distances. Since the distances are in one-to-one correspondence to the correlation coefficients and the crossover probability for Gaussian and symmetric discrete distribution respectively, the parameters are also consistent. This implies that the KL-divergence between p and $\widehat{p}^n$ tends to zero (in probability) as the number of samples n tends to infinity. This completes the proof.

(ii) The theorem follows by using the assumption that the effective depth $\delta = \delta(T_p; V)$ is constant. Recall that $\tau > 0$ is the threshold used in relaxed RG (see (21) in Section 6.3). Let the set of triples (i, j, k) whose pairwise information distances are less than τ apart be $\mathcal{J}$, that is, $(i, j, k) \in \mathcal{J}$ if and only if $\max\{d_{ij}, d_{jk}, d_{ki}\} < \tau$. Since we assume that the true information distances are uniformly bounded, there exist $\tau > 0$ and some sufficiently small $\lambda > 0$ so that if $|\widehat{\Phi}_{ijk} - \Phi_{ijk}| \leq \lambda$ for all $(i, j, k) \in \mathcal{J}$, then RG recovers the correct latent structure.

Define the error event $\mathcal{E}_{ijk} := \{|\widehat{\Phi}_{ijk} - \Phi_{ijk}| > \lambda\}$. We note that the probability of the event $\mathcal{E}_{ijk}$ decays exponentially fast, that is, there exists $J_{ijk} > 0$ such that for all $n \in \mathbb{N}$,

$$\Pr(\mathcal{E}_{ijk}) \leq \exp(-n J_{ijk}). \qquad (34)$$

The proof of (34) follows readily for Chernoff bounds (Hoeffding, 1958) and is omitted. The error probability associated to structure learning can be bounded as follows:

$$\Pr\left(h(\widehat{T}^n) \neq T_p\right) \overset{(a)}{\leq} \Pr\left(\bigcup_{(i,j,k)\in\mathcal{J}} \mathcal{E}_{ijk}\right) \overset{(b)}{\leq} \sum_{(i,j,k)\in\mathcal{J}} \Pr(\mathcal{E}_{ijk})$$

$$\leq m^3 \max_{(i,j,k)\in\mathcal{J}} \Pr(\mathcal{E}_{ijk}) \overset{(c)}{\leq} \exp(3\log m)\exp\left[-n \min_{(i,j,k)\in\mathcal{J}} J_{ijk}\right],$$

where (a) follows from the fact that if the event $\{h(\widehat{T}^n) \neq T_p\}$ occurs, then there is at least one sibling or parent-child relationship that is incorrect, which corresponds to the union of the events $\mathcal{E}_{ijk}$, that is, there exists a triple $(i,j,k) \in \mathcal{J}$ is such that $\widehat{\Phi}_{ijk}$ differs from Φ_{ijk} by more than λ. Inequality (b) follows from the union bound and (c) follows from (34).

Because the information distances are uniformly bounded, there also exists a constant $J_{\min} > 0$ (independent of m) such that $\min_{(i,j,k)\in\mathcal{J}} J_{ijk} \geq J_{\min}$ for all $m \in \mathbb{N}$. Hence for every $\eta > 0$, if the number of samples satisfies $n > 3(\log(m/\sqrt[3]{\eta}))/J_{\min}$, the error probability is bounded above by η. Let $C := 3/J_{\min}$ to complete the proof of the sample complexity result in (26). The proof for the logarithmic sample complexity of distribution reconstruction for Gaussian and symmetric discrete models follows from the logarithmic sample complexity result for structure learning and the fact that the information distances are in a one-to-one correspondence with the correlation coefficients (for Gaussian models) or crossover probabilities (for symmetric discrete models).

A.7 Proof of Theorem 12: Consistency and Sample Complexity of Relaxed CLRG

(i) Structural consistency of CLGrouping follows from structural consistency of RG (or NJ) and the consistency of the Chow-Liu algorithm. Risk consistency of CLGrouping for Gaussian or symmetric distributions follows from the structural consistency, and the proof is similar to the proof of Theorem 11(i).

(ii) The input to the CLGrouping procedure $\widehat{T}_{\mathrm{CL}}$ is the Chow-Liu tree and has $O(\log m)$ sample complexity (Tan et al., 2010, 2011), where m is the size of the tree. This is true for both discrete and Gaussian data. From Theorem 11, the recursive grouping procedure has $O(\log m)$ sample complexity (for appropriately chosen thresholds) when the input information distances are uniformly bounded. In any iteration of the CLGrouping, the information distances satisfy $d_{ij} \leq \gamma u$, where γ, defined in (30), is the worst-case graph distance of any hidden node from its surrogate. Since γ satisfies (32), $d_{ij} \leq u^2\delta/l$. If the effective depth $\delta = O(1)$ (as assumed), the distances $d_{ij} = O(1)$ and the sample complexity is $O(\log m)$. □

References

K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25(2):251–278, 1999.

M. F. Balcan and P. Gupta. Robust hierarchical clustering. In *Intl. Conf. on Learning Theory (COLT)*, 2010.

H.-J. Bandelth and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Adv. Appl. Math*, 7:309–43, 1986.

S. Bhamidi, R. Rajagopal, and S. Roch. Network delay inference from additive metrics. *To appear in Random Structures and Algorithms, Arxiv preprint math/0604367*, 2009.

R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: Recent developments. *Stat. Science*, 19:499–517, 2004.

J. T. Chang and J. A. Hartigan. Reconstruction of evolutionary trees from pairwise distributions on current species. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 254–257, 1991.

T. Chen, N. L. Zhang, and Y. Wang. Efficient model evaluation in the search based approach to latent structure discovery. In *4th European Workshop on Probabilistic Graphical Models*, 2008.

M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, June 2010.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, 3:462–467, 1968.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Science/Engineering/Math, 2nd edition, 2003.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, New York, 1999.

M. Csűrös. *Reconstructing Phylogenies in Markov Models of Sequence Evolution*. PhD thesis, Yale University, 2000.

C. Daskalakis, E. Mossel, and S. Roch. Optimal phylogenetic reconstruction. In *STOC '06: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 159–168, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge Univ. Press, 1999.

G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

P. L. Erdős, L. A. Székely, M. A. Steel, and T. J. Warnow. A few logs suffice to build (almost) all trees: Part ii. *Theoretical Computer Science*, 221:153–184, 1999.

J. Farris. Estimating phylogenetic trees from distance matrices. *Amer. Natur.*, 106(951):645–67, 1972.

S. Harmeling and C. K. I. Williams. Greedy learning of binary latent trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1958.

D. Hsu, S.M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Intl. Conf. on Learning Theory (COLT)*, 2009.

A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. *ACM Computing Reviews*, 1999.

T. Jiang, P. E. Kearney, and M. Li. A polynomial-time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM J. Comput.*, 30(6):194261, 2001.

C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Science*, 105(31):10687–10692, 2008.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), Feb 1956.

M. R. Lacey and J. T. Chang. A signal-to-noise analysis of phylogeny estimation by neighbor-joining: Insufficiency of polynomial length sequences. *Mathematical Biosciences*, 199:188–215, 2006.

J. A. Lake. Reconstructing evolutionary trees from dna and protein sequences: Parallnear distances. *Proceedings of the National Academy of Science*, 91:1455–1459, 1994.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

P. F. Lazarsfeld and N.W. Henry. *Latent Structure Analysis*. Boston: Houghton Mifflin, 1968.

D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. Wiley, 1979.

D. Parikh and T. H. Chen. Hierarchical Semantics of Objects (hSOs). In *ICCV*, pages 1–8, 2007.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible inference*. Morgan Kaufmann, 1988.

R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36, 1957.

D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53: 131–147, 1981.

S. Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(1), 2006.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20:53–65, 1987.

N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425, Jul 1987.

S. Sattath and A. Tversky. Additive similarity trees. *Psychometrika*, 42:319–45, 1977.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley-Interscience, Nov 1980.

S. Shen. Large deviation for the empirical correlation coefficient of two Gaussian random variables. *Acta Mathematica Scientia*, 27(4):821–828, Oct 2007.

R. Silva, R. Scheine, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, Feb 2006.

L. Song, B. Boots, S. M. Siddiqi, G. Gordon, and A. Smola. Hilbert space embeddings of hidden Markov models. In *Proc. of Intl. Conf. on Machine Learning*, 2010.

K. St. John, T. Warnow, B. M. E. Moret, and L. Vawter. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *J. Algorithms*, 48(1):173–193, 2003.

M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning Gaussian tree models: Analysis of error exponents and extremal structures. *IEEE Transactions on Signal Processing*, 58(5):2701–2714, May 2010.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning high-dimensional Markov forest distributions: Analysis of error rates. *Journal of Machine Learning Research (In Press)*, 2011.

Y. Tsang, M. Coates, and R. D. Nowak. Network delay tomography. *IEEE Trans. Signal Processing*, 51:21252136, 2003.

Y. Wang, N. L. Zhang, and T. Chen. Latent tree models and approximate inference in Bayesian networks. *Journal of Artificial Intelligence Research*, 32:879–900, Aug 2008.

N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

N. L. Zhang and T Kočka. Efficient learning of hierarchical latent class models. In *ICTAI*, 2004.