Learning Sparse Markov Network Structure via Ensemble-of-Trees Models

Yuanqing Lin, Shenghuo Zhu
NEC Laboratories America
Cupertino, CA 95014

Daniel D. Lee[†], Ben Taskar[‡]
[†]Department of Electrical and Systems Engineering
[‡]Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA19104

Abstract

Learning the sparse structure of a general Markov network is a hard computational problem. One of the main difficulties is the computation of the generally intractable partition function. To circumvent this difficulty, we propose to learn the network structure using an *ensemble-of-trees* (ET) model. The ET model was first introduced by Meilă and Jaakkola (2006), and it represents a multivariate distribution using a mixture of all possible spanning trees. The advantage of the ET model is that, although it needs to sum over super-exponentially many trees, its partition function as well as data likelihood can be computed in a *closed form*. Furthermore, because the ET model tends to represent a Markov network using as small number of trees as possible, it provides a natural regularization for finding a *sparse* network structure. Our simulation results show that the proposed ET approach is able to accurately recover the true Markov network connectivity and outperform the state-of-art approaches for both discrete and continuous random variable networks when a small number of data samples is available. Furthermore, we also demonstrate the usage of the ET model for discovering the network of words from blog posts.

1 Introduction

Markov networks concisely represent complex multivariate distributions by exploiting the sparsity of direct dependencies: each node is directly connected to only a small subset of other nodes. For example, in a Markov network for modeling articulated objects, only neighboring parts are assumed to have direct interaction; in a social network, each individual directly interacts with only a small number of people in the network; in a gene network, a regulatory pathway involves only a small number of genes. Such sparse structures are not only essential for enabling effective learning and inference, but also may be the primary goal in knowledge discovery (for example, discovering the gene regulatory pathway). While the structures of most Markov networks are handcrafted, there has been increasing interest in learning the sparse structure of a Markov network from data.

One approach to Markov network structure learning is to constrain search to low-treewidth (thin) Markov networks to enable tractable exact inference. Chow-Liu algorithm (Chow & Liu, 1968) approximates a discrete probability distribution using a *single* spanning tree, and the optimal spanning tree can be learned from data in $O(NM^2)$ where M is the number of variables, and N is the number of data samples. Since the single spanning tree model is often too restrictive, Bach and Jordan (2001) proposed learning thin networks via greedy addition of cliques. Recently, several papers proposed more complex combinatorial search over the space of treewidth-bounded structures with strong approximation guarantees (Narasimhan & Bilmes, 2004; Srebro, 2001; Chechetka & Guestrin, 2007). However, these algorithms scale exponentially with treewidth and are not practical for many real-world domains where treewidth tends to be large.

For continuous Markov networks, if the variables are assumed to have a joint Gaussian distribution, the partition function has a closed-form regardless of the structure. Banerjee et al. (2006) formulated the Gaussian Markov network structure learning into a convex optimization problem with l_1-norm sparsity regularization, and the resulting optimization can be solved very efficiently (Friedman et al., 2007).

Since low-treewidth and Gaussian models are fairly limited, there are attempts at addressing general Markov networks. Abbeel et al. (2006) provided an efficient algorithm

Appearing in Proceedings of the 12^{th} International Confe-rence on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

with probability approximately correct (PAC) learnability guarantees for factor graphs. Lee et al. (2007) proposed to learn a sparse general log-linear model by optimizing an l_1-norm regularized log-likelihood. However, the difficulty there is that the partition function can be computed only approximately and becomes very hard to compute as the network becomes relatively dense. Wainwright et al. (2006) proposed a pseudo-likelihood based scheme for learning the sparse structure of a binary Markov network. To find the neighbor set of a specific node, their approach estimates an l_1-norm regularized binary logistic regression model of the node given the rest of the nodes and uses the non-zero weights as indication of edges in the network. The advantage of this approach is that it does not require computing partition function of the Markov network, and the l_1-norm logistic regression can be solved efficiently. However, this approach does not optimize global likelihood and its result may not even be consistent with respect to the edges selected by different nodes.

We propose to learn the structure of a general Markov network using an *ensemble of trees* (ET) model. The ET model was first introduced by Meilă and Jaakkola (2006), and it approximates a Markov network using a mixture of *all possible* (exponentially many) spanning trees of the network. The key feature of the ET model is that although it needs to sum over exponential many spanning trees, its partition function as well as data likelihood has a *closed form*. Surprisingly, we have not seen any work that uses the ET model for Markov network structure learning. In this paper, we simplify the ET model in a way such that it is suitable for learning the structure of both discrete and continuous Markov networks, and we develop an efficient algorithm for solving the resulting optimization problem. Besides having a tractable partition function, our proposed ET model has some other advantages in comparison to the existing methods for Markov network structure learning. First, it is versatile to different Markov networks: discrete or continuous, Gaussian or non-Gaussian, low or high tree width, and so on. Second, since the ET model tends to fit data using as small number of spanning trees as possible, it provides a natural regularization mechanism for finding a *sparse* representation of a Markov network. Hence there are no regularization parameters to set in our ET model learning in contrast to those methods that employ l_1-norm sparsity regularization. Our simulation results show that, when only a small number of data samples is available, the proposed ET approach provides significantly better performance in recovering the structures of Gaussian Markov networks and binary Markov networks, compared respectively to the l_1-norm Gaussian Markov network approach by Banerjee et al. (2006) and the l_1-norm regularized logistic regression approach by Wainwright et al. (2006). Our experimental results show that the ET approach is able to recover a meaningful word network from blog posts.

2 Ensemble-of-trees (ET) model for learning the sparse structure of a Markov network

We first introduce the notation that will be used throughout this paper. The $M \times 1$ vector $X = [X_1, X_2, ..., X_M]^T$ denotes the nodes of a Markov network, and M is the total number of nodes. We refer to nodes as X_u and X_v also as u and v, respectively. The edge between nodes u and v is denoted as e_{uv}. The vector $\mathbf{x}$ ($M \times 1$ vector) represents a realization of X. In a discrete Markov network, a node u takes r_u different values.

2.1 Previous work of ensemble of trees (ET) model

The *ensemble of trees* (ET) model was first introduced by Meilă and Jaakkola (2006). It is a mixture of trees probabilistic model, but the mixture is an ensemble of trees since the mixture is over *all possible* (and super-exponentially many) spanning trees of a Markov network. As such, the probability of a data sample $\mathbf{x}$ in the ET model is

$$P(\mathbf{x}) = \sum_{T} P(T) P(\mathbf{x}|T, \boldsymbol{\theta}_T) \tag{1}$$

where $P(T)$ denotes the probability of each spanning tree T, $\boldsymbol{\theta}_T$ are the parameters in the tree T, and the summation is over all possible spanning trees. While the ET model seems to be intractable at first glance, its data likelihood in Eq. 1 has a *closed form* when it is parameterized appropriately.

First, the ET model parameterizes the $P(T)$ in Eq. 1 as:

$$P(T) = \frac{1}{Z} \prod_{e_{uv} \in T} \beta_{uv}, \tag{2}$$

where $\boldsymbol{\beta} = \{\beta_{uv} = \beta_{vu} \geq 0\}_{u=1,2,...,M; v=u+1,u+2,...,M}$ are parameters and $Z = \sum_T \prod_{e_{uv} \in T} \beta_{uv}$ is the partition function that sums over all possible spanning trees. The matrix tree theorem provides a powerful tool for computing the partition function when the $\boldsymbol{\beta}$ are binary variables ($\beta_{uv} \in \{0, 1\}$). In fact, in the binary case, the partition function has a *closed form*

$$Z = \det[\mathbf{Q}(\boldsymbol{\beta})] \tag{3}$$

where $\mathbf{Q}(\boldsymbol{\beta})$ [$(M-1) \times (M-1)$ matrix] is the first $M-1$ rows and columns of the Laplacian matrix $\mathbf{L}(\boldsymbol{\beta})$ ($M \times M$), namely

$$L_{uv} = \begin{cases} -\beta_{uv} & \text{if} \quad u \neq v \\ \sum_k \beta_{uk} & \text{if} \quad u = v. \end{cases} \tag{4}$$

Meilă and Jaakkola (2006) showed that the Eq. 3 for the partition function also holds when $\boldsymbol{\beta}$ are not binary but nonnegative numbers. This result was referred as the generalized matrix tree theorem.

Second, the ET model assumes that the same edge in different trees shares the same θ parameters. In other words,

$\boldsymbol{\theta}_T = \boldsymbol{\theta}$ regardless of T. Since the parameters $\boldsymbol{\theta}$ are node distributions $\{\theta_u\}_{u=1,2,...,M}$ and edge distributions $\{\theta_{uv}\}_{u=1,2,...M;v=1,2,...M}$, the probability of $P(\mathbf{x}|T,\boldsymbol{\theta})$ can be written as

$$P(\mathbf{x}|T,\boldsymbol{\theta}) = \prod_{e_{uv}\in T} \frac{\theta_{uv}(x_u,x_v)}{\theta_u(x_u)\theta_v(x_v)} \prod_u \theta_u(x_u). \quad (5)$$

Note that we dropped the T index on $\boldsymbol{\theta}$ since $\boldsymbol{\theta}_T$ are the same for all trees.

With the $P(T)$ and $P(\mathbf{x}|T,\boldsymbol{\theta})$ in Eqs. 2 and 5, the data likelihood in Eq. 1 can be written as

$$\begin{aligned} P(\mathbf{x}) &= \sum_T \frac{1}{Z} \prod_{e_{uv}\in T} \beta_{uv} \frac{\theta_{uv}(x_u,x_v)}{\theta_u(x_u)\theta_v(x_v)} \prod_u \theta_u(x_u) \\ &= \frac{1}{Z} w_0(\mathbf{x}) \sum_T \prod_{e_{uv}\in T} \beta_{uv} w_{uv}(\mathbf{x}) \\ &= w_0(\mathbf{x}) \frac{\det[\mathbf{Q}(\boldsymbol{\beta}\otimes \mathbf{w}(\mathbf{x}))]}{\det[\mathbf{Q}(\boldsymbol{\beta})]} \end{aligned} \quad (6)$$

where $w_{uv}(\mathbf{x}) = \frac{\theta_{uv}(x_u,x_v)}{\theta_u(x_u)\theta_v(x_v)}$, $w_0(\mathbf{x}) = \prod_{u\in X}\theta_u(x_u)$, $\mathbf{w}(\mathbf{x})$ is the collection of $w_{uv}(x_u,x_v)$ using the same indexing as $\boldsymbol{\beta}$, $\otimes$ denotes element-wise multiplication, and the last equality is because $\sum_T \prod_{uv\in T} \beta_{uv} w_{uv}(\mathbf{x}) = \det[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}(\mathbf{x}))]$ due to the generalized matrix tree theorem in Eq. 3.

With the closed form for data likelihood, the parameters ($\boldsymbol{\beta}$, $\boldsymbol{\theta}$) in the ET model can be estimated in a maximum - likelihood (ML) fashion. For a discrete Markov network, given N observed data $\{\mathbf{x}^{(i)}\}_{i=1}^N$, Meilă and Jaakkola (2006) suggested to find ML estimations by solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta},\boldsymbol{\beta}} N \log\det[\mathbf{Q}(\boldsymbol{\beta})] - \sum_{i=1}^N \log\det[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})] \\ &\text{S.T.} \sum_{x_v=1}^{r_v} \theta_{uv}(x_u,x_v) = \theta_u(x_u), \quad \sum_{x_u=1}^{r_u} \theta_u(x_u) = 1, \\ &\quad u = 1,2,...,M;\ v = u+1, u+2, ..., M; \\ &\quad \boldsymbol{\theta} \geq 0 \end{aligned} \quad (7)$$

where $w_{uv}^{(i)} = \frac{\theta_{uv}(x_u^{(i)},x_v^{(i)})}{\theta_u(x_u^{(i)})\theta_v(x_v^{(i)})}$, and the constraints on $\boldsymbol{\theta}$ are necessary since they are node and edge distributions.

2.2 Ensemble of trees (ET) model for structure learning

In this paper, we exploit the ET model for Markov network structure learning. The ET model has two sets of parameters: one set is structural parameters $\boldsymbol{\beta}$, and the other set are the distribution parameters $\boldsymbol{\theta}$. Our primary interest is the set of structural parameters. Intuitively, the parameter β_{uv} can be understood as the connectivity between nodes u and v as in the matrix tree theorem where β_{uv} is either 1 or 0 indicating whether the nodes u and v are connected or not. More formally, it can be shown that β_{uv} is closely related to edge appearance probability $\rho_{uv} = \sum_T P(e_{uv} \in T)$ defined in (Wainwright et al., 2005) since

$$\rho_{uv} = 1 - \frac{\det[\mathbf{Q}(\tilde{\boldsymbol{\beta}})]}{det[\mathbf{Q}(\boldsymbol{\beta})]} = \beta_{uv}\mathbf{e}_{uv}^T[\mathbf{L}(\boldsymbol{\beta}) + \frac{1}{M^2}\mathbf{e}\mathbf{e}^T]^{-1}\mathbf{e}_{uv} \quad (8)$$

where $\tilde{\boldsymbol{\beta}}$ is equal to $\boldsymbol{\beta}$ except that $\tilde{\beta}_{uv} = 0$, $\mathbf{e}_{uv}$ is an $M\times 1$ vector with its u^{th} and v^{th} elements being -1 and the rest elements being zeros, and $\mathbf{e}$ is an $M \times 1$ vector with its elements being all ones. From Eq.8, we see that, if $\beta_{uv} = 0$, edge e_{uv} appears in trees with zero probability (though this does not mean u and v are conditionally independent in the ET model).

Compared to the previous approaches (Lee et al., 2007) for learning the structure of a general Markov network, the ET model has some advantages. First, the objective function in the ET model has a closed form and there is no need to worry about intractable partition functions. Second, the ET model tends to fit data using as small number of spanning trees as possible. For example, in the extreme case where there is only one data sample, the optimal solution is simply a single spanning tree. This behavior is explained by Eq. 7, where the $\det[\mathbf{Q}(\boldsymbol{\beta})]$ term represents the number of spanning trees given $\boldsymbol{\beta}$ and it serves as a regularization mechanism that penalizes complex networks. As such, when the structural parameters $\boldsymbol{\beta}$ are learned in the ET model, they are expected to be *sparse* in nature. Hence the ET model does not require setting l_1-norm sparsity regularization parameters in contrast to some existing methods.

Unfortunately, solving the optimization in Eq. 7 is still not easy. The optimization is nonconvex with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$. Moreover, the optimization has a large number of optimization variables [$M(M-1)/2$ for $\boldsymbol{\beta}$ and $(\frac{1}{2}\sum_{uv:u\neq v} r_u r_v + M)$ for $\boldsymbol{\theta}$] as well as a big number of constraints. This will be further complicated in the case of continuous Markov network where the summations in the constraints become integrations (Kirshner, 2008).

To simplify the ET model, we set the node parameters $\boldsymbol{\theta}$ with an plug-in estimators. First, since the parameters $\theta_u(x_u)$ $(u = 1,2,...,M)$ are node distributions and they have little to do with network structures, their optimal setting can be estimated directly from data. In a discrete Markov network, they can be computed by counting $\theta_u(x_u) = [\sum_{i=1}^N I(x_u^{(i)}, x_u)]/N,\ x_u = 1,2,...r_u$, where $I(\cdot,\cdot)$ is the indicator function and $x_u^{(i)}$ denotes the value of the u^{th} element in the i^{th} data sample $\mathbf{x}^{(i)}$; in a continuous Markov network, the node distributions $\theta_u(x_u)$ can be estimated by fitting a one-dimensional Gaussian to the data $\{x_u^{(i)}\}_{i=1}^N$. Second, our simplification of the ET model goes one more step further, that is, we also estimate the edge distributions $\theta_{uv}(x_u,x_v)$ directly from data without taking network structures into account. Similarly, in a discrete Markov network, $\theta_{uv}(x_u, x_v)$ is computed by counting; in a continuous Markov network, θ_{uv} is estimated by fitting the data $\{[x_u^{(i)}; x_v^{(i)}]\}_{i=1}^N$ with a two-dimensional Gaussian. The above choices dramatically simplify the ET model since the optimization in Eq. 7 is now only with respect to the structural parameters $\boldsymbol{\beta}$. Nevertheless, as we will demonstrate later in Section 3, the simplified ET model is still powerful in recovering the structure of Markov networks.

Another issue in the optimization in Eq. 7 is that the structural parameters $\boldsymbol{\beta}$ need to be constrained. First, according to the definition of the tree probability in Eq. 2, the elements in $\boldsymbol{\beta}$ must be nonnegative. The nonnegative constraints lead to sparse solutions of $\boldsymbol{\beta}$, similar to the nonnegativity regularization in the nonnegative matrix factorization and the support vector machine. Second, we also notice that the scaling of $\boldsymbol{\beta}$ does not change the objective function. This is because, if $\hat{\boldsymbol{\beta}} = c\boldsymbol{\beta}$ with c being a constant, $\log\det[\mathbf{Q}(\hat{\boldsymbol{\beta}})] - \log\det[\mathbf{Q}(\hat{\boldsymbol{\beta}} \otimes \mathbf{w}^{(i)})] = \log\{c^{M-1}\det[\mathbf{Q}(\boldsymbol{\beta})]\} - \log\{c^{M-1}\det[\mathbf{Q}(\boldsymbol{\beta} \otimes \mathbf{w}^{(i)})]\} = \log\det[\mathbf{Q}(\boldsymbol{\beta})] - \log\det[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})]$ where M is the number of nodes in the network. Therefore, to eliminate the scaling degeneracy in $\boldsymbol{\beta}$ solutions, we choose to impose a normalization constraint on $\boldsymbol{\beta}$, namely, $\sum_{uv,u\neq v} \beta_{uv} = 1$.

To summarize, the resulting optimization for the ET model for network structure learning is:

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} N\log\det[\mathbf{Q}(\boldsymbol{\beta})] - \sum_{i=1}^{N}\log\det[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})]\\
&s.t.\ \beta_{uv} \geq 0 \quad u,v = 1,\ldots,M,\ \ u\neq v\\
&\sum_{u,v:u\neq v}\beta_{uv} = 1. \qquad (9)
\end{aligned}$$

It is worth noting that, $\mathbf{Q}(\boldsymbol{\beta})$ needs to be symmetric positive definite, but that is automatically guaranteed by the way that $\mathbf{Q}(\boldsymbol{\beta})$ is constructed.

2.3 Optimization algorithm

Although we have greatly simplified the ET model and the simplification has led to a relatively simple optimization shown in Eq. 9, it is still not so easy to solve the resulting optimization because it is *nonconvex* (the objective is a difference of convex functions) and traditional methods for convex optimization may fail. For example, log-barrier interior point methods are powerful approaches for solving constrained convex optimization, but they are not suitable for nonconvex optimization since their early centering steps would generally shift solutions to the center of the domain of an optimization problem regardless of initial solutions. Consequently, there is no guarantee that the interior point method would yield a better solution than an initial solution due to the nonconvex nature of the optimization.

Our choice here is a projected gradient descent (PGD) algorithm (Bertsekas, 2003). As we shall show, the PGD algorithm is guaranteed to converge to a local minimum near an initial solution when its step sizes are selected properly.

The PGD algorithm is indeed efficient for the optimization that we have in Eq. 9. In general, a PGD algorithm consists of two steps in each update iteration. The first step is to move a solution along the opposite of gradient direction with a certain step size. For example, if the current solution estimate is $\boldsymbol{\beta}$ and the gradient is ∇_β, then the first step is to move to $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta} - \alpha\nabla_\beta$ where α is some nonnegative scalar representing step size. Then, the second step of the PGD algorithm is to find the point in the domain of optimization that is closest to $\hat{\boldsymbol{\beta}}$. This point is then a new $\boldsymbol{\beta}$ estimate. The second step is basically a projection step, and that is why the algorithm is named *projected* gradient descend. The PGD algorithm is particularly efficient when an optimization has only element-wise bound constraints. This is because, in this special case, the projection step is simply coordinate-wise truncation (Bertsekas, 2003). Our optimization in Eq. 9 indeed has element-wise nonnegative constraints except the unit l_1-norm constraint. Fortunately, since scaling of $\boldsymbol{\beta}$ does not change the objective function as explained in Section 2.2, the unit l_1-norm constraint can be treated separately. In other words, the unit l_1-norm constraint can be enforced by normalizing $\boldsymbol{\beta}$ after each PGD update, and the normalization should not change the convergence property of the PGD algorithm.

Since the optimization in Eq. 9 has a closed form, it is easy to compute its derivative with respect to $\boldsymbol{\beta}$. Let $f(\boldsymbol{\beta})$ be the objective function, then the gradient of $\boldsymbol{\beta}$ is

$$(\nabla_\beta f)_{uv} = \mathbf{e}_{uv}^T\{N[\mathbf{Q}(\boldsymbol{\beta})]^{-1} - \sum_{i=1}^{N} w_{uv}^{(i)}[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})]^{-1}\}\mathbf{e}_{uv} \qquad (10)$$

when $u \neq M$ and $v \neq M$ with $\mathbf{e}_{uv}$ being an $M \times 1$ vector whose u^{th} and v^{th} elements are -1 and other elements are zeros, and

$$(\nabla_\beta f)_{uv} = \{N[\mathbf{Q}(\boldsymbol{\beta})]^{-1} - \sum_{i=1}^{N} w_{uv}^{(i)}[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})]^{-1}\}_{uu} \qquad (11)$$

when $u < M$ and $v = M$, and $(\nabla_\beta f)_{vu} = (\nabla_\beta f)_{uv}$.

Given the derivative, the PGD update rule is

$$\boldsymbol{\beta}_{uv}^{new} = [\boldsymbol{\beta}_{uv}^{old} - \alpha(\nabla_\beta f)_{uv}]_+ \qquad (12)$$

where $(z)_+$ is a hinge function so that $(z)_+ = 0$ if $z < 0$, and $(z)_+ = z$ if $z \geq 0$. The parameter α is the step size determined by Armijo's rule (Bertsekas, 2003), namely, it is the largest value in the series $\{\gamma^m\}_{m=0}^{+\infty}$ that satisfies

$$f(\boldsymbol{\beta}^{new}) - f(\boldsymbol{\beta}^{old}) \leq \eta(\boldsymbol{\beta}^{new} - \boldsymbol{\beta}^{old})^T\nabla_\beta f \qquad (13)$$

where $\nabla_\beta f$ is the vectorization of $\{(\nabla_\beta f)_{uv}\}_{u,v=1,2,\ldots M,\ v\neq u}$, and typical settings for

γ and η are $\gamma = 0.5$ and $\eta = 0.01$. The above Armijo's rule guarantees that each PGD iteration significantly (no worse than linear) decreases the objective function until the PGD algorithm converges. This excludes early stopping in the PGD algorithm. Consequently, the algorithm is guaranteed to monotonically converge to a local optimizer. A more detailed convergence proof of the PGD algorithm can be found in Chapter 2.3 in (Bertsekas, 2003).

After each PGD update, our algorithm normalizes $\boldsymbol{\beta}$ to enforce the unit l_1-norm constraint in the optimization in Eq. 9. Since the normalization does not change the objective function, each iteration (a PGD update plus a normalization) still guarantees a significant decrease of the objective function. As such, the resulting algorithm is guaranteed to monotonically converge to a local optimizer.

Besides the main optimization algorithm that finds a local optimizer near an initial solution, how to find the initial solution is also an important issue in nonconvex optimization. The popular approach for choosing the initial solution is to take some random initial solutions and select the one that yields the best objective function after optimization. The downside of this approach is that it needs to solve the optimization many times. Here, we have another choice, that is, we derive the initial solution by solving a convex optimization whose objective function upper-bounds the objective function in Eq. 9. To do that, we notice that $\det[\mathbf{Q}(\boldsymbol{\beta})] = \prod_{u=1}^{M-1}(\lambda_{\mathbf{Q}})_u$ with $(\lambda_{\mathbf{Q}})_u$ representing the u^{th} eigenvalue of $\mathbf{Q}(\boldsymbol{\beta})$. Then $\prod_{u=1}^{M-1}(\lambda_{\mathbf{Q}})_u \leq (\frac{1}{M-1}\sum_u \lambda_{\mathbf{Q}})^{M-1} = \{\frac{1}{M-1}\text{Trace}[\mathbf{Q}(\boldsymbol{\beta})]\}^{M-1} \leq \{\frac{1}{M-1}\text{Trace}[\mathbf{L}(\boldsymbol{\beta})]\}^{M-1} = (\frac{2}{M-1})^{M-1}$ where the last equality is because the unit l_1-norm constrain on $\boldsymbol{\beta}$ and $L_{uu} = \sum_{k\neq u}\beta_{u,k}$ for any $u \in X$. Therefore, ignoring constant terms, we may solve the following optimization to attain an initial solution for initializing the optimization in Eq. 9:

$$\min_{\boldsymbol{\beta}} -\sum_{i=1}^{N}\log\det[\mathbf{Q}(\boldsymbol{\beta}\otimes\mathbf{w}^{(i)})] \tag{14}$$
$$S.T.\ \beta_{uv} \geq 0 \quad u = 1,2,...,M,\ v = 1,2,...,M,\ \ u \neq v$$
$$\sum_{u,v:u\neq v}\beta_{uv} = 1.$$

This is a convex optimization problem, and it can solved efficiently by many approaches; we use the PGD algorithm for this problem as well.

3 Results

In this section, we employ both simulated examples and real-world applications to test our proposed ET approach for Markov network structure learning. In simulations, we test the ET approach on both Gaussian Markov networks and binary Markov networks. For Gaussian Markov networks, we compare the ET approach with the l_1-norm regularized Gaussian Markov network approach (denoted as l_1GMN) by Banerjee et al. (2006); for binary Markov network, the ET approach is compared with the l_1-norm regularized logistic regression approach (denoted as l_1LR) by Wainwright et al. (2006). We also apply the ET model to a real-world application, which is to discover a word network from a number of blog posts.

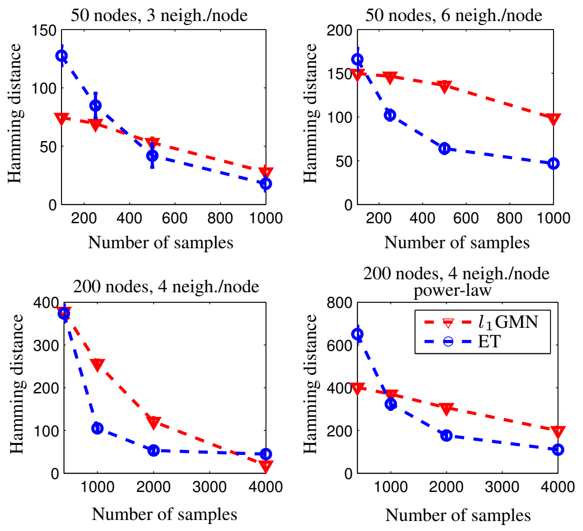

Figure 1: The comparison between our proposed ET approach and the l_1-norm regularized Gaussian Markov network approach (l_1GMN) (Banerjee et al., 2006) on recovering the sparse structures of simulated Gaussian Markov networks. The results are evaluated by their Hamming distances to their respective ground truth. We simulated four types of networks: **Upper left**, 50 nodes with averagely 3 neighbors per node; **Upper right**, 50 nodes with averagely 6 neighbors per node; **Lower left**, 200 nodes with averagely 4 neighbors per node; **Lower right**, 200 nodes, 4 neighbors per node on average, but the number of the neighbors to each node has a power law distribution. Each result is averaged over five independent trials, which have the same neighboring statistics but different Markov network structures.

3.1 Simulation results

3.1.1 Learning the sparse structure of a Gaussian Markov network

In this simulation, we demonstrate the performance of the proposed ET approach on finding the sparse structures of Gaussian Markov networks. We assumed the Gaussian Markov networks to have zero mean since the mean would not affect our analysis. A zero-mean Gaussian Markov network can be described as

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{M/2}\det(\Sigma)^{-1/2}}\exp\{-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\} \tag{15}$$

where M denotes the number of nodes in the network, Σ represents covariance matrix, and Σ^{-1} is the precision matrix describing the structure of the network (for example, if $(\Sigma^{-1})_{uv} = 0$, then nodes u and v are independent conditioned on the rest of nodes in the network).

We simulated Gaussian Markov networks with four types of sparse structures (or precision matrices): 1) 50 nodes with averagely 3 neighbors per node, 2) 50 nodes with averagely 6 neighbors per node, 3) 200 nodes with averagely 4 neighbors per node, and 4) 200 nodes with 4 neighbors per node on average but the number of neighbors to each node has a power law distribution. For each type of sparse structure, we randomly constructed 5 precision matrices (or inverse covariance matrices) for simulating Gaussian Markov networks. For each precision matrix, the positions of its nonzero elements were selected randomly with a constraint of its neighboring statistics. Its off-diagonal nonzero elements were sampled using $s(0.1 + 0.2|n|)$ where s was randomly drawn from $\{+1, -1\}$ and n was drawn from a zero-mean Gaussian with unit variance, and its diagonal elements were assigned to assure the positive definiteness of the precision matrix. Given the precision matrix for a zero-mean Gaussian distribution, it is easy to sample data from the distribution. For each 50-node Gaussian Markov network, we drew 1000 data samples; for each 200-node network, we drew 4000 data samples. Now, the goal here is to see how well the ET approach recovers the sparse structures of those precision matrices given different numbers of sampled data. We compare the performance of the ET approach with the l_1GMN approach (Banerjee et al., 2006). For the l_1GMN, its sparsity regularization parameter was determined by Eq.(3) in (Banerjee et al., 2006), namely, $\lambda(\alpha) = (\max_{u>v} \hat{\sigma}_u \hat{\sigma}_v) \frac{t_{N-2}[\alpha/(2M^2)]}{\sqrt{n-2+t^2_{N-2}[\alpha/(2M^2)]}}$, where $t_{n-2}[\alpha/(2M^2)]$ denotes the $(100-\alpha)\%$ point of Student's t-distribution for $N-2$ degrees of freedom, N is the number of samples used for structure learning, $\hat{\sigma}_u$ is the square root of the empirical variance of u, and α was set to be 0.05. It is worth noting that, to our experience, setting the regularization parameter in this way provided much better results than setting it using cross-validation.

Figure 1 illustrates the performance of the ET approach in recovering the structures of different Gaussian Markov networks in comparison with the l_1GMN approach. The performance is evaluated by Hamming distance, the number of disagreeing edges between an estimated network and the ground truth. The results show that, although the l_1GMN approach tends to yield accurate estimates when the number of samples is sufficiently big, the ET approach consistently out-performs the l_1GMN approach in different Markov network structures when only a medium number of data samples were used for structure learning.

Notably, the ET model is very general in the sense that it did not use the knowledge that the Markov networks were Gaussian. Therefore, we can expect the ET model to have strong ability to go beyond Gaussian Markov networks because the model only needs to estimate up to second-order joint distributions (for the marginals in Eq. 7), which can be estimated fairly accurately from a medium amount of data regardless of Gaussian or non-Gaussian distributions.

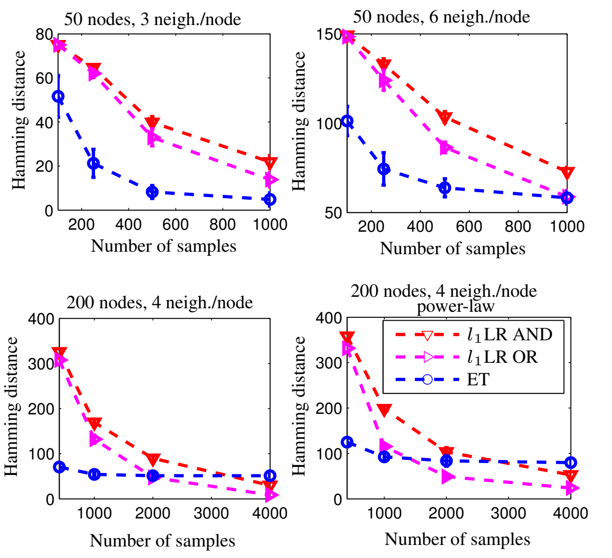

Figure 2: The performance comparison between our proposed ET approach and the l_1-norm regularized logistic regression (l_1LR) approach (Wainwright et al., 2006) for learning the sparse structures of simulated Binary Markov networks. In the l_1LR approach, its results (sparse weights) can be combined by using logic AND (shown by down-triangle dash lines) or logic OR (shown by right-triangle dot lines). The circle dash-dot lines describes the results of the ET approach. We simulated four types of networks: **Upper left**, 50 nodes with averagely 3 neighbors per node; **Upper right**, 50 nodes with averagely 6 neighbors per node; **Lower left**, 200 nodes with averagely 4 neighbors per node; **Lower right**, 200 nodes, 4 neighbors per node on average, but the number of the neighbors to each node has a power law distribution. Each result is averaged over five independent trials, which have the same neighboring statistics but different Markov network structures.

3.1.2 Learning the sparse structure of a binary Markov network

The ET model can also be applied for finding the sparse structure of a discrete Markov network. Here, we use binary Markov networks as examples and compare the ET approach with the l_1LR approach *et al.* (Wainwright et al., 2006) on network structure learning. We used the following form for simulating a binary Markov network:

$$P(\mathbf{x}) = \frac{1}{Z} \exp\{\sum_{uv} \theta_{uv} x_u x_v + \sum_u \psi_u x_u\} \qquad (16)$$

where $Z = \sum_{\mathbf{x}} \exp\{\sum_{uv} \theta_{uv} x_u x_v + \sum_u \psi_u x_u\}$ is the partition function, $\mathbf{x} \in \{1, -1\}^M$, $\boldsymbol{\psi}$ are the coefficients of linear terms, and $\boldsymbol{\theta}$ are coefficients of quadratic terms describing the structure of the network.

In simulating binary Markov networks, we employed the same sparse structures that we had simulated for Gaussian Markov networks. Given the sparse pattern of each structure, we sampled the nonzero elements of θ_{uv} using $\theta_{uv} = s(0.1 + 0.2n)$ where s was drawn randomly from

$\{-1, +1\}$ and n was uniformly distributed in $[0, 1]$. The linear coefficients $\boldsymbol{\psi}$ were drawn in a similar way except that they were all positive. Then given $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$, we employed Gibbs sampling to draw samples from the distribution. Similar as in the case of Gaussian Markov networks, we drew 1000 samples for each 50-node network and 4000 samples for each 200-node network. For the l_1LR approach, the l_1-norm sparsity regularization parameter (in Eq. 3 in (Wainwright et al., 2006)) was set to be $\lambda = \sqrt{\log(M)/N}$ as proposed in (Wainwright et al., 2006), where M is the number of nodes and N is the number of samples.

Figure 2 shows the performance comparison between our proposed ET approach and the l_1LR approach on recovering the sparse structures of different binary Markov networks. In the l_1LR approach, the solutions of individual logistic regressions may not be consistent to each other. For example, let's say, in one logistic regression where node i is seen as the output, node j is associated with a nonzero weight; however, in the logistic regression where node j is the output, note i may *not* be associated with a nonzero weight. As a result, the solutions of the l_1LR approach are combined by either logic AND or logic OR, and different logics often yield different structure estimates. Figure 2 illustrates that the ET approach is able to achieve good structure estimation with a relatively small number of data samples. In contrast, the l_1LR approach typically needs much more data to get a good structure estimation. This is because the l_1LR employs a *pseudo-likelihood* paradigm where the overall objective function is the sum of the log-likelihood of individual logistic regressions and a pseudo-likelihood approach typically is not efficient in terms of data samples. In contrast, the ET approach is a *maximum likelihood* estimation approach, albeit for a different model.

3.2 Finding word network from blog posts

We experimented with the ET model for learning a sparse word network from blog data. more recently Internet blogs have become a fastest-growing self-publishing media. Although the current search engines are able to take return popular webpages given some query, the information in blogosphere is scattered and knowledge mining becomes very challenging. An important task in blogosphere knowledge mining is to learn the relations among different subjects, described by words. The proposed ET model can be applied to learn a sparse network of those words that describes their joint distribution.

In our experiments, we had about 18,000 blog posts spanning from January to March, 2008. According to the *tf-idf* ranking of the vocabulary of the blogs, we selected 1000 words. Each post was represented by a 1000×1 binary vector whose i^{th} (i =1, ..., 1000) element was the indicator of whether the i^{th} word appeared in that post.

Table 1 shows a part of learned word network. Such a word network can be useful, for instance, for blog navigation. For example, if an Internet user is interested in looking at Apple company. Then, from the learned network, the user will be offered the several aspects to look at: iPhone (its product), Mac (its product), iPod (its product), iTunes (its product), store (place to get an Apple product), Nasdaq (about stock), Microsoft (its primary competitor), and so on. If the user is further interested in Apples' stock, he/she will be further directed to look at analyst (probably what analysts said about Apples' stock), the stock's expectation, trend and portfolio, as well as Apple's competitors like Microsoft and HP.

4 Discussion

We have described a ensemble-of-trees (ET) model for learning the sparse structure of a general Markov network. The advantage of the ET model is that its partition function and data likelihood have *closed forms*. Based on the original ET model introduced by Meilă and Jaakkola (2006), we propose a much simplified form. Our simplification not only is for easier optimization but also makes the ET model suitable for learning the structure of both continuous and discrete Markov networks. We also develop an efficient algorithm for solving the resulting optimization, which unfortunately is non-convex. To address the initialization issue for the non-convex optimization, we propose to solve a convex upper-bound optimization problem.

We tested the proposed ET approach for recovering the structures of Markov networks on both simulated examples and real-world applications. In simulated examples, when only a small number of data samples is available, the results showed that the ET approach outperforms the l_1GMN approach (Banerjee et al., 2006) on Gaussian Markov networks and the l_1LR approach (Wainwright et al., 2006) on binary Markov networks. Remarkably, in the case of Gaussian Markov networks, the ET model does *not* use the knowledge that the Markov networks were Gaussian. Our proposed ET model is indeed versatile to different Markov networks, Gaussian or non-Gaussian, continuous or discrete, and so on. In a real-world application, we showed that the ET model was able to learn a meaningful network of 1000 words from blog posts. Since it is impossible to show the whole learned 1000-word network in this paper, we showed a small sub-network and the whole network is available on our webpage.

As we have demonstrated, the ET approach is promising for learning the sparse structure of a general Markov network. Meanwhile, our ongoing and future work focuses on further improving the ET model. First, the ET model is still restrictive in representing relatively dense networks. We found in our simulations that the ET approach often yielded an overly sparse estimate of a dense network (with

Table 1: A part of learned sparse word network from blog posts. The subnetwork is centered at 'apple'. In each column, the word in the first row has connections to the rest of words in the same column, and they are ranked by their connectivity (described by $\boldsymbol{\beta}$ in the ET model) to the first word.

apple	**iphone**	**mac**	**ipod**	**itunes**	**store**	**nasdaq**
iphone	apple	apple	iphone	apple	shop	newsletter
mac	ipod	os	apple	download	retail	e-mail
microsoft	mobile	pc	touch	music	apple	analyst
ipod	device	windows	itunes	ipod	purchase	blackberry
itunes	phone	macbook	music	amazon	music	expectation
store	rumor	user	device	podcast	price	trend
pc	blackberry	hardware		song	iphone	download
user	smartphone	computer		dvd	library	letter
device	pc	leopard		stream	item	print
hardware	gps	desktop			selling	network
computer	user	safari			amazon	directory
macbook	nokia	pro			storage	responsibility
price	store	graphics			backup	hp
upgrade	wireless	vista			package	portfolio
mobile	gadget					demonstration
os	app					apple
leopard	australia					microsoft
nasdaq	carrier					exchange
safari	samsung					
maker	touch					
apps	launch					
smartphone						
competitor						

very low recalls on nonzero edges, though). We believe this is because the ET model tends to use as small number of spanning trees as possible due to the $\det[\mathbf{Q}(\boldsymbol{\beta})]$ regularization (representing the total number of spanning trees) in the optimization in Eq. 7. We can potentially relax the regularization to allow less sparse models. Second, the distributions represented by the ET models is very different than Markov networks, and yet the sparsity patterns seem to largely coincide. A challenge for future work is to establish a theoretical understanding of the relationship between the sparsity patterns of the two types of models.

References

Abbeel, P., Koller, D., & Ng, A. Y. (2006). Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, *7*, 1743–1788.

Bach, F., & Jordan, M. (2001). Thin junction trees. *Advances in Neural Information Processing Systems (NIPS)*.

Banerjee, O., Ghaoui, L. E., dAspremont, A., & Natsoulis, G. (2006). Convex optimization techniques for fitting sparse gaussian graphical models. *International Conference On Machine Learning (ICML)*.

Bertsekas, D. P. (2003). *Nonlinear programming*. Athena Scientific.

Chechetka, A., & Guestrin, C. (2007). Efficient principled learning of thin junction trees. *Advances in Neural Information Processing Systems (NIPS)*.

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

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *http://www-stat.stanford.edu/ tibs/glasso/index.html*.

Kirshner, S. (2008). Learning with tree-averaged densities and distributions. *Advances in Neural Information Processing Systems (NIPS)*.

Lee, S.-I., Ganapathi, V., & Koller, D. (2007). Efficient structure learning of markov networks using l1-regularization. *Advances in Neural Information Processing Systems (NIPS)*.

Meilă, M., & Jaakkola, T. (2006). Tractable bayesian learning of tree belief networks. *Statistics and Computing*, *16*, 77–92.

Narasimhan, M., & Bilmes, J. A. (2004). Pac-learning bounded tree-width graphical models. *Uncertainty in Artificial Intelligence (UAI)* (pp. 410–417).

Srebro, N. (2001). Maximum likelihood bounded tree-width markov networks. *Uncertainty in Artificial Intelligence (UAI)* (pp. 504–511).

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, *51*, 2313– 2335.

Wainwright, M. J., Ravikumar, P., & Lafferty, J. (2006). High-dimensional graphical model selection using ℓ_1-regularized logistic regression. *Advances in Neural Information Processing Systems (NIPS)*.