Latent Dependency Forest Models

Shanbo Chu, Yong Jiang and Kewei Tu

School of Information Science and Technology, ShanghaiTech University, Shanghai, China

Introduction

- Probabilistic Modeling
  - Compactly represents the joint probability distribution of random variables. It is one of the foundations of modern AI.
  - The most widely used approach for probabilistic modeling is probabilistic graphical models (PGMs), like Bayesian Network and Markov Network.
- Weakness of These Graphical Models
  - Learning PGMs (in particular, learning the model structures) is very difficult in general.
  - The dependencies between variables are fixed and therefore can not model context-specific independence (CSI).
  - For most of the above models, learning involves identification of a good model structure, which is typically a difficult discrete optimization problem.

Motivations

- Motivation
  - We formulate a LDFM as a first-order non-projective dependency grammar where the terminals are variable assignments and circumvent the structure learning step by including all possible grammar rules in the model and then learning their parameters.
- Comparisons with Other Models
  - Comparisons with **Sum-Product Networks (SPNs)** [*Poon and Domingos, 2011*]
    - **Similarities:** Both models model latent dependencies between random variables. Both models can model CSIs.
    - **Differences:** No latent variables in LDFMs. Easier learning.
  - Comparisons with **Mixture of Trees (MTs)** [*Meila and Jordan, 2001*]
    - **Similarities:** Both models assume the latent dependencies to form a forest structure.
    - **Differences:** Unlike MTs that restrict the possible dependency structure to a small number of forest structures, LDFMs consider all possible forest structures at the same time.

Latent Dependency Forest Models

- Basics
  - Let $\mathbf{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$ be a set of random variables and $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ be an assignment to the set of random variables. Given an assignment $\mathbf{x}$, we construct a complete directed graph $\boldsymbol{G}_\mathrm{x} = (\boldsymbol{V}_\mathrm{x}, \boldsymbol{E}_\mathrm{x})$ such that,
    - $\boldsymbol{V}_\mathrm{x} = \{\boldsymbol{x}_0 = \boldsymbol{root}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ where $\boldsymbol{x}_0$ is a dummy root node.
    - $\boldsymbol{E}_\mathrm{x} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) | i \neq j, 0 \leq i \leq n, 1 \leq j \leq n\}$

    We obtain a single dependency tree structure that is a directed spanning tree of the graph $\boldsymbol{G}_\mathrm{x}$ rooted at $\boldsymbol{x}_0$, where the weight of the graph is denoted as $\boldsymbol{w}_{ij}$. We denote this tree by $\boldsymbol{T} = (\boldsymbol{V}_T, \boldsymbol{E}_T)$, where $\boldsymbol{V}_T = \boldsymbol{V}_\mathrm{x}, \boldsymbol{E}_T \subseteq \boldsymbol{E}_\mathrm{x}$.
    We can compute the strength or weight of a spanning tree $\boldsymbol{T} = (\boldsymbol{V}_T, \boldsymbol{E}_T)$ as the product of the edge weights:

$$\boldsymbol{w}(T) = \prod_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \boldsymbol{E}_T} \boldsymbol{w}_{ij}$$

  - **Weight** of the assignment $\mathbf{x}$ is the sum over the weights of all possible dependency trees for $\mathbf{x}$.

$$\boldsymbol{Z}_\mathrm{x} = \sum_{T \in T(G_\mathrm{x})} \boldsymbol{w}(T) = \sum_{T \in T(G_\mathrm{x})} \prod_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \boldsymbol{E}_T} \boldsymbol{w}_{ij}$$

- LDFM, a generative model based on the above framework.

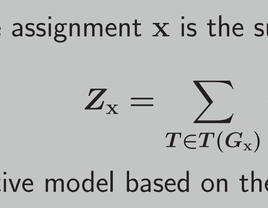
  An assignment $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ is generated recursively in a top-down manner.
  - **Firstly**, we generate a dependency tree with $\boldsymbol{n} + 1$ nodes uniformly at random. We label the root node as $\boldsymbol{x}_0$.
  - **Then**, starting from the root node, we recursively traverse the tree in pre-order; at each non-root node, we generate a $\langle \boldsymbol{variable}, \boldsymbol{value} \rangle$ pair conditioned on the $\langle \boldsymbol{variable}, \boldsymbol{value} \rangle$ pair of its parent node. The probability of generating an assignment $\mathbf{x}$ is:

$$\boldsymbol{p}(\mathrm{x}) = \boldsymbol{\beta} \boldsymbol{n}! \boldsymbol{Z}_\mathrm{x} \propto \boldsymbol{Z}_\mathrm{x}$$

  Matrix Tree Theorem (MTT) *.Let $\boldsymbol{G}$ be a graph with nodes $\boldsymbol{V} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and edges $\boldsymbol{E}$. Define (Laplacian) matrix $\boldsymbol{Q}$ as a $(\boldsymbol{n}+1) \times (\boldsymbol{n}+1)$ matrix indexed from $\boldsymbol{0}$ to $\boldsymbol{n}$. For all $\boldsymbol{i}$ and $\boldsymbol{j}$, define:*

$$\boldsymbol{Q}_{ij} = \begin{cases} \sum_{i' \neq j, (\boldsymbol{x}_{i'}, \boldsymbol{x}_j) \in \boldsymbol{E}} \boldsymbol{w}_{i'j} & \text{if } \boldsymbol{i} = \boldsymbol{j} \\ -\boldsymbol{w}_{ij} & \text{if } \boldsymbol{i} \neq \boldsymbol{j}, (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \boldsymbol{E} \end{cases}$$

  If the $\boldsymbol{i}$-th row and column are removed from $\boldsymbol{Q}$ to produce the matrix $\boldsymbol{Q}^i$, then the sum of the weights of all the directed spanning trees rooted at node $\boldsymbol{i}$ is equal to the determinant of $\boldsymbol{Q}^i$. Here, $\boldsymbol{Z}_x = \boldsymbol{det}(\boldsymbol{Q}^0)$



Dependency Forest	Weight
$X_1 = F \quad X_2 = T$	$w_1 = w_{x_1=F\|root} \times w_{x_2=T\|root}$
$X_1 = F \rightarrow X_2 = T$	$w_2 = w_{x_1=F\|root} \times w_{x_2=T\|x_1=F}$
$X_1 = F \leftarrow X_2 = T$	$w_3 = w_{x_1=F\|x_2=T} \times w_{x_2=T\|root}$

$$p(X_1 = F, X_2 = T) \propto w_1 + w_2 + w_3$$

Figure 1: An example of using LDFMs to compute the joint probability of $\boldsymbol{X}_1 = \boldsymbol{False}$ and $\boldsymbol{X}_2 = \boldsymbol{True}$. Left: all possible pairwise dependencies between the two variables and a root node; each dependency has a weight. Right: all three possible dependency forests and their weights. Bottom: computing the joint probability.

- The above process may also generate invalid assignments (Assignments that do not contain all random variables). We define the joint probability of a valid assignment $\mathbf{x}$ as

$$\phi(\mathrm{x}) = \frac{\boldsymbol{p}(\mathrm{x})}{\sum_{\mathrm{x} \in A} \boldsymbol{p}(\mathrm{x})} = \frac{\boldsymbol{Z}_\mathrm{x}}{\gamma}$$

Inference

In probabilistic inference, the set of random variables are divided into query, evidence, and hidden variables and we want to compute the conditional probabilities of the query variables given the evidence variables, which is shown to be ♯P-hard, We resort to Markov chain Monte Carlo (MCMC) for approximate inference.

- **Gibbs Sampling** resamples each of the query and hidden variables in turn according to its conditional distribution given the rest of the variables. Each step costs $\boldsymbol{O}(\boldsymbol{n}^3)$ time complexity.
- **Tree Sampling** simultaneously samples the latent dependency tree structure and the variable values. We name this method tree-augmented sampling. Each step costs $\boldsymbol{O}(\boldsymbol{n})$ time complexity.
  - We first randomly initialize the values of the query and hidden variables as well as the dependency tree structure.
  - At each MCMC step, we randomly pick a variable and simultaneously change its value and its parent node in the dependency tree (with the constraint that no loop is formed).
  - Suppose variable $\boldsymbol{X}_i$ is picked, then the proposal probability of value $\boldsymbol{x}_i$ and parent $\boldsymbol{X}_j$ is proportional to the dependency weight $\boldsymbol{w}_{x_i|x_j}$ where $\boldsymbol{x}_j$ is the value of $\boldsymbol{X}_j$ in the previous sample.

Learning

- Key Ideas
  - Avoid the difficult structure learning problem by assuming a complete LDFM structure.
  - Rely on parameter learning to specify the weights of all the dependencies.
- Objective Function: the log-likelihood of the model.

$$\sum_{\alpha=1}^{|D|} \log \boldsymbol{p}(\mathrm{x}_\alpha) = \sum_{\alpha=1}^{|D|} \log \boldsymbol{Z}_{\mathrm{x}_\alpha} + \mathbf{constant}$$

  where $\boldsymbol{D} = \{\mathrm{x}_\alpha\}_{\alpha=1}^{|D|}$ is the training dataset. Note that we compute the likelihood based on $\boldsymbol{p}(\mathrm{x})$, the probability of generating an assignment, instead of $\boldsymbol{\phi}(\mathrm{x})$, the probability of a valid assignment. This makes our learning algorithm tractable and also encourages the learned model to be more likely to produce valid assignments.
- Learning Algorithms
  - **E-step**, we compute the partition functions and edge expectations of the training samples. We can compute the edge expectations through matrix inversion. When $\boldsymbol{i}, \boldsymbol{j} > \boldsymbol{0}$,

$$\langle (\boldsymbol{x}_i, \boldsymbol{x}_j) \rangle_\mathrm{x} = \boldsymbol{w}_{x_j|x_i} \boldsymbol{Z}_\mathrm{x} [((\boldsymbol{Q}^0)^{-1})_{jj} - ((\boldsymbol{Q}^0)^{-1})_{ji}]$$

    When $\boldsymbol{i} = \boldsymbol{0}$ and $\boldsymbol{j} > \boldsymbol{0}$,

$$\langle (\boldsymbol{x}_0, \boldsymbol{x}_j) \rangle_\mathrm{x} = \boldsymbol{w}_{x_j|x_0} \boldsymbol{Z}_\mathrm{x} ((\boldsymbol{Q}^0)^{-1})_{jj}$$

  - **M-step**, we update the parameters $\boldsymbol{w}_{x_j|x_i}$ to maximize the log-likelihood subject to the constraints of $\sum_{j \neq i} \sum_{x_j} \boldsymbol{w}_{x_j|x_i} = \boldsymbol{1}$.

$$\boldsymbol{w}_{x_j|x_i} = \frac{\sum_{\alpha=1}^{|D|} \frac{1}{\boldsymbol{Z}_{\mathrm{x}_\alpha}} \langle (\boldsymbol{x}_i, \boldsymbol{x}_j) \rangle_{\mathrm{x}_\alpha}}{\sum_{\alpha=1}^{|D|} \frac{1}{\boldsymbol{Z}_{\mathrm{x}_\alpha}} \sum_{j' \neq i} \sum_{x_{j'}} \langle (\boldsymbol{x}_i, \boldsymbol{x}_{j'}) \rangle_{\mathrm{x}_\alpha}}$$

Experiments

- 9 benchmark BNs from the *bnlearn* repository. For each BN, we sampled two training sets of 5000 and 500 instances, one validation set of 1000 instances, and one testing set of 1000 instances. All the random variables are discrete.
- Comparisons by their accuracy in query answering on the test data, which is to compute the conditional log likelihood (CLL) $\boldsymbol{p}(\boldsymbol{Q} = \boldsymbol{q} | \boldsymbol{E} = \boldsymbol{e})$, where $\boldsymbol{q}$ and $\boldsymbol{e}$ are the values that $\boldsymbol{Q}$ and $\boldsymbol{E}$ take in the test data sample.
- Due to sample sparsity of sampling algorithms, we report the maximum of CLL and CMLL ($\sum_{\boldsymbol{X}_i \in \boldsymbol{Q}} \log \boldsymbol{p}(\boldsymbol{X}_i = \boldsymbol{x}_i | \boldsymbol{E} = \boldsymbol{e})$) values.
- We empirically showed that LDFMs are competitive with existing probabilistic models.

Table 1: The maximum of CLL and CMLL normalized by the number of query variables. The bold numbers mark the best performance. g-LDFM denotes the results of LDFM by using Gibbs sampling and t-LDFM denotes the results of LDFM by using tree-augmented sampling.

5000	40% Query, 30% Evidence						30% Query, 20% Evidence					
Dataset	g-LDFM	t-LDFM	BN	DN	SPN	MT	g-LDFM	t-LDFM	BN	DN	SPN	MT
Asia	-0.258	**-0.241**	-0.274	-0.268	-0.262	-0.262	-0.268	**-0.240**	-0.286	-0.276	-0.272	-0.272
Child	**-0.609**	-0.663	-0.721	-0.634	-0.630	-0.707	**-0.650**	-0.702	-0.744	-0.670	-0.688	-0.761
Alarm	-0.293	-0.357	-0.436	-0.317	**-0.277**	-0.343	-0.335	-0.396	-0.473	-0.375	**-0.328**	-0.379
Insurance	**-0.460**	-0.538	-0.565	-0.499	-0.476	-0.557	**-0.550**	-0.616	-0.660	-0.598	-0.581	-0.652
Sachs	**-0.605**	-0.620	-0.675	-0.610	-0.644	-0.647	-0.632	**-0.627**	-0.698	-0.649	-0.683	-0.681
Water	**-0.399**	-0.401	-0.474	-0.407	-0.415	-0.435	**-0.434**	-0.439	-0.496	-0.445	-0.457	-0.478
Win95pts	-0.166	-0.169	-0.229	-0.185	**-0.118**	-0.121	-0.163	-0.193	-0.251	-0.207	-0.147	**-0.146**
Hepar2	**-0.481**	-0.483	-0.509	-0.490	-0.489	-0.507	**-0.481**	-0.482	-0.513	-0.493	-0.497	-0.513
Hailfinder	-0.991	-1.091	-1.223	-1.089	**-0.941**	-1.241	-1.013	-1.096	-1.242	-1.110	**-0.979**	-1.268
500	40% Query, 30% Evidence						30% Query, 20% Evidence					
Dataset	g-LDFM	t-LDFM	BN	DN	SPN	MT	g-LDFM	t-LDFM	BN	DN	SPN	MT
Asia	-0.263	**-0.246**	-0.301	-0.266	-0.272	-0.264	-0.268	**-0.244**	-0.296	-0.280	-0.276	-0.273
Child	**-0.623**	-0.677	-0.801	-0.668	-0.757	-0.927	**-0.665**	-0.706	-0.813	-0.701	-0.804	-0.898
Alarm	**-0.328**	-0.368	-0.521	-0.359	-0.426	-0.526	**-0.370**	-0.403	-0.605	-0.408	-0.463	-0.510
Insurance	**-0.478**	-0.542	-0.665	-0.533	-0.596	-0.706	**-0.564**	-0.617	-0.751	-0.621	-0.698	-0.776
Sachs	**-0.627**	-0.634	-0.702	-0.653	-0.759	-0.733	**-0.644**	-0.654	-0.712	-0.678	-0.780	-0.723
Water	**-0.406**	-0.411	-0.482	-0.431	-0.511	-0.531	**-0.441**	-0.445	-0.507	-0.462	-0.541	-0.543
Win95pts	-0.162	-0.191	-0.271	-0.205	**-0.151**	-0.174	-0.188	-0.205	-0.311	-0.231	**-0.173**	-0.194
Hepar2	-0.491	**-0.488**	-0.539	-0.503	-0.531	-0.641	-0.490	**-0.485**	-0.535	-0.504	-0.535	-0.591
Hailfinder	**-1.024**	-1.10	-1.449	-1.127	-1.187	-2.244	**-1.040**	-1.098	-1.511	-1.135	-1.197	-1.938