Collapsed Variational Inference for Sum-Product Networks

Han Zhao[†] HAN.ZHAO@CS.CMU.EDU
Tameem Adel[§] T.M.A.A.HESHAM@UVA.NL
Geoff Gordon[†] GGORDON@CS.CMU.EDU
Brandon Amos[†] BAMOS@CS.CMU.EDU

[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[§]Machine Learning Lab, University of Amsterdam, Amsterdam, the Netherlands; Radboud University

Abstract

Sum-Product Networks (SPNs) are probabilistic inference machines that admit exact inference in linear time in the size of the network. Existing parameter learning approaches for SPNs are largely based on the maximum likelihood principle and are subject to overfitting compared to more Bayesian approaches. Exact Bayesian posterior inference for SPNs is computationally intractable. Even approximation techniques such as standard variational inference and posterior sampling for SPNs are computationally infeasible even for networks of moderate size due to the large number of local latent variables per instance. In this work, we propose a novel deterministic collapsed variational inference algorithm for SPNs that is computationally efficient, easy to implement and at the same time allows us to incorporate prior information into the optimization formulation. Extensive experiments show a significant improvement in accuracy compared with a maximum likelihood based approach.

1. Introduction

Parameter learning is an important research problem in Sum-Product Networks (SPNs) (Poon & Domingos, 2011; Gens & Domingos, 2012). Due to recent progress, we can locally maximize the likelihood of SPNs, but those approaches overfit for complex models because they do not retain posterior uncertainty or allow flexible incorporation of prior information. SPNs are essentially mixture models with potentially exponentially many components (Zhao & Poupart, 2015). Exact posterior inference for SPNs is computationally intractable due to the exponential growth of the number of mixture components in the posterior distribution. Approximate Bayesian inference for SPNs is challenging, mostly because of the vast number of local latent variables and their intricate connections. Unlike traditional mixture models, e.g., LDA (Blei et al., 2003), Gaussian mixture models, etc., where the number of local latent variables is a small constant per observation, the number of local latent variables in SPNs is linearly proportional to the size of the network for each observable instance, often around thousands to millions. This property makes standard approximate Bayesian inference approaches, including mean-field variational inference (VI) (Wainwright & Jordan, 2008) and posterior sampling (Casella & Berger, 2002), computationally infeasible even for SPNs with moderate size. In SPNs, the memory overhead incurred by the number of local variational parameters or the number of local sampling points increases linearly with both the number of data points, D, and the size of the network, $|\mathcal{S}|$, and so can easily become prohibitively large.

In this work we propose a collapsed variational inference algorithm for SPNs that is robust to overfitting and can be naturally extended into a stochastic variant to scale to large data sets. We call our algorithm CVB-SPN. CVB-SPN is a deterministic approximate Bayesian inference algorithm that is computationally efficient and easy to implement while at the same time allowing us to incorporate prior information into the design. Like other variational techniques, CVB-SPN is based on optimization. Unlike other variational techniques, the number of parameters to be optimized is only linear in the network size ($O(|\mathcal{S}|)$) and is *independent* of the size of the training set, as opposed to $O(D|\mathcal{S}|)$ in ordinary VI. It is worth noting that, different from traditional collapsed variational approaches in the literature that marginalize out global latent variables (Teh et al., 2006; 2007), here we consider a complementary approach: instead of marginalizing out the global latent variables in order to spread out the interactions among many local latent variables, CVB-SPN takes advantage of the fast

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

exact inference in SPNs and *marginalizes out the local latent variables* in order to maintain a marginal variational posterior distribution directly on the global latent variables, i.e., the model parameters. The posterior mean of the variational distribution over model parameters can then be used as a Bayesian estimator. To the best of our knowledge, this is the first general Bayesian approach to learn the parameters of SPNs efficiently in both batch and stochastic settings.

At first glance, marginalizing out all the local latent variables in graphical models seems to be a bad idea for two reasons. First, by marginalizing out local latent variables we naively appear to incur computation exponential in the tree-width of the graph (Jordan et al., 1999; Wainwright & Jordan, 2008). Except for graphical models with special structures, for example, LDA, Gaussian mixture models, thin junction trees, etc., such exact computation is intractable by itself. Second, marginalization will in general invalidate the conjugacy between the prior distribution and the joint distribution over global and local latent variables, which further makes the expectation over the variational posterior intractable. Fortunately, as we will show in Sec. 3, the ability of SPNs to model context-specific independence helps to solve the first problem, and by using a change of variables CVB-SPN handles the second problem efficiently. Besides the reduced space complexity, we also show that the objective of CVB-SPN forms a strictly better lower bound to be optimized than the evidence lower bound (ELBO) in standard VI. To show the validity of CVB-SPN, we conduct extensive experiments and compare it with maximum likelihood based methods in both batch and stochastic settings.

To tackle the second problem described above, there is a strand of recent work on extending standard VI to general nonconjugate settings using stochastic sampling from the variational posterior (Blei et al., 2012; Kingma & Welling, 2013; Ranganath et al., 2014; Mnih & Gregor, 2014; Titsias & Lázaro-Gredilla, 2014; Titsias, 2015). However, control variates need to be designed in order to reduce the high variance incurred by insufficient samples. Furthermore, for each such sample from the variational posterior, those algorithms need to go through the whole training data set to compute the stochastic gradient, leading to a total computational cost $O(ND|\mathcal{S}|)$, where N is the sample size. This is often prohibitively expensive for SPNs.

2. Background

2.1. Sum-Product Networks

A sum-product network (SPN) is a graphical representation of a joint probability distribution over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. It is a rooted directed acyclic graph where the interior nodes are sums or products and the leaves are univariate distributions over X_i. Edges emanating from sum nodes are parameterized with positive weights. Each node in an SPN encodes an unnormalized marginal distribution over $\mathbf{X}$.

In more detail, let $\mathbf{x}$ be an instantiation of the random vector $\mathbf{X}$. We associate an unnormalized probability $V_k(\mathbf{x} \mid \mathbf{w})$ with each node k when the input to the network is $\mathbf{x}$ with network weights set to be $\mathbf{w}$:

$$V_k(\mathbf{x} \mid \mathbf{w}) = \begin{cases} p(X_i = \mathbf{x}_i) & k \text{ is a leaf node over } X_i \\ \prod_{j \in \mathrm{Ch}(k)} V_j(\mathbf{x} \mid \mathbf{w}) & k \text{ is a product node} \\ \sum_{j \in \mathrm{Ch}(k)} w_{kj} V_j(\mathbf{x} \mid \mathbf{w}) & k \text{ is a sum node} \end{cases} \tag{1}$$

where $\mathrm{Ch}(k)$ is the child list of node k in the graph and w_{kj} is the edge weight associated with sum node k and its child node j. The joint distribution encoded by an SPN is then defined by the graphical structure and the weights. The probability/density of a joint assignment $\mathbf{X} = \mathbf{x}$ is proportional to the value at the root of the SPN with input $\mathbf{x}$ divided by a normalization constant $V_{\text{root}}(\mathbf{1} \mid \mathbf{w})$:

$$p(\mathbf{x}) = \frac{V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{V_{\text{root}}(\mathbf{1} \mid \mathbf{w})} \tag{2}$$

where the normalization constant $V_{\text{root}}(\mathbf{1} \mid \mathbf{w})$ is obtained by setting V at all the leaf nodes to be 1 and then propagating toward the root according to Eq. 1. Intuitively, setting V at all the leaf nodes to be 1 corresponds to integrating/marginalizing out the random vector $\mathbf{X}$, which will ensure Eq. 2 defines a proper probability distribution. Eq. 2 can also be used to compute the marginal probability of a partial assignment $\mathbf{Y} = \mathbf{y}$: simply set V at leaf nodes whose corresponding random variable is not in $\mathbf{Y}$ to be 1 and other leaf nodes based on the assignment $\mathbf{Y} = \mathbf{y}$. Intuitively, this corresponds to integrating out variables outside of the partial assignment. We can compute conditional probabilities by evaluating two partial assignments:

$$p(\mathbf{Y} = \mathbf{y} \mid \mathbf{Z} = \mathbf{z}) = \frac{p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z})}{p(\mathbf{Z} = \mathbf{z})} = \frac{V_{\text{root}}(\mathbf{y}, \mathbf{z} \mid \mathbf{w})}{V_{\text{root}}(\mathbf{z} \mid \mathbf{w})}$$

Since joint, marginal and conditional queries can all be computed by two network passes, exact inference takes linear time with respect to the network size. Without loss of generality, in this paper we focus our attention on the case where all the random variables X_i are Boolean. However, direct extension to the continuous case is not hard; one can simply replace each univariate distribution at the leaves of the graph with the desired continuous distribution.

2.2. SPNs as Mixture Models

It has been recently shown that any complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X} = \{X_1, \ldots, X_n\}$ is equivalent to

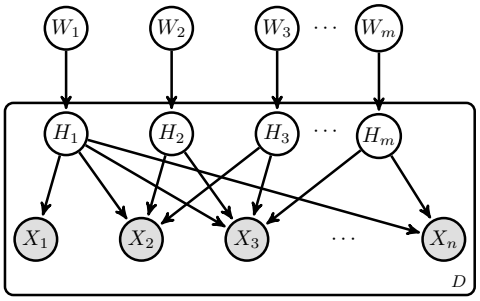

Figure 1. Graphical model representation of SPN $\mathcal{S}$. The box is a plate that represents replication over D training instances. $m = O(|\mathcal{S}|)$ corresponds to the number of sum nodes and n is the number of observable variables in $\mathcal{S}$. Typically, $m \gg n$. W, H, X correspond to global latent variables, local latent variables and observable variables, respectively.

a Bayes net with $O(n|\mathcal{S}|)$ size (Zhao et al., 2015). The insight behind the construction (Fig. 1) is that each internal sum node in $\mathcal{S}$ corresponds to a latent variable in the constructed Bayes net, and the Bayes net will be a bipartite graph (excluding the nodes that correspond to global latent variables) with one layer of local latent variables pointing to one layer of observable variables $\mathbf{X}$. An observable variable is a child of a local latent variable if and only if the observable variable appears as a descendant of the latent variable (sum node) in the original SPN. Equivalently, this shows that the SPN $\mathcal{S}$ can be interpreted as a Bayes net where the number of latent variables per instance is $O(|\mathcal{S}|)$. Although the Bayes net perspective provides an interesting connection between the two models, we can hardly work on the Bayes net in practice due to the large number of latent variables and high treewidth of the graph.

Another representation of an SPN is to decompose it into a sum of induced trees (Zhao & Poupart, 2015). This decomposition works as follows:

Definition 1 (Induced tree SPN)**.** Given a complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X} = \{X_1, \ldots, X_n\}$, $\mathcal{T} = (\mathcal{T}_V, \mathcal{T}_E)$ is called an *induced tree SPN* from $\mathcal{S}$ if

1. Root$(\mathcal{S}) \in \mathcal{T}_V$.
2. If $v \in \mathcal{T}_V$ is a sum node, then exactly one child of v in $\mathcal{S}$ is in $\mathcal{T}_V$, and the corresponding edge is in $\mathcal{T}_E$.
3. If $v \in \mathcal{T}_V$ is a product node, then all the children of v in $\mathcal{S}$ are in $\mathcal{T}_V$, and the corresponding edges are in $\mathcal{T}_E$.

Here $\mathcal{T}_V$ is the node set of $\mathcal{T}$ and $\mathcal{T}_E$ is the edge set of $\mathcal{T}$.

The name "induced tree SPN" is justified because it has been shown that Def. 1 produces mixture components that are trees whenever the original SPN is complete and decomposable. A key result based on the notion of induced trees that exhibits the distribution modeled by SPNs

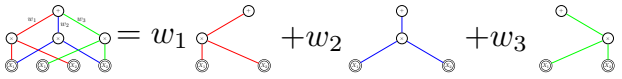

Figure 2. Induced tree decomposition of a complete and decomposable SPN into a mixture model. Each component is characterized by a product of univariate distributions over X_1 and X_2.

is (Zhao & Poupart, 2015):

Theorem 2. Let $\tau_{\mathcal{S}} = V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$. Then $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$ can be written as $\sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$, where $\mathcal{T}_t$ is the tth unique induced tree of $\mathcal{S}$ and $p_t(X_i)$ is a univariate distribution over X_i in $\mathcal{T}_t$ as a leaf node.

In Thm. 2, $\tau_{\mathcal{S}}$ equals the number of induced trees in $\mathcal{S}$. Fig. 2 gives an illustrative example of Thm. 2. Thm. 2 characterizes both the number of components and the form of each component in the mixture model, as well as their mixture weights.

2.3. Variational Bayes Inference

Standard variational Bayes inference is an optimization-based approach to approximate the full posterior distribution of a probabilistic model (Jordan et al., 1999). It works by constructing and maximizing an evidence lower bound (ELBO) of the log marginal likelihood function $\log p(\mathbf{x})$. Equivalently, one can minimize the Kullback-Leibler (KL) divergence between the true posterior distribution and the variational posterior distribution. To review, let $\Theta = \{\mathbf{H}, \mathbf{W}\}$ represent the set of latent variables in a probabilistic model (including both the local, $\mathbf{H}$, and the global, $\mathbf{W}$, latent variables) and let $\mathbf{X}$ represent the data. For example, in mixture models such as LDA, $\mathbf{H}$ corresponds to the topic assignments of the words and $\mathbf{W}$ corresponds to the topic-word distribution matrix. The joint probability distribution of Θ and $\mathbf{X}$ is $p(\mathbf{X}, \Theta \mid \boldsymbol{\alpha})$ where $\boldsymbol{\alpha}$ is the set of hyperparameters of the model. Standard VI approximates the true posterior $p(\Theta \mid \mathbf{X}, \boldsymbol{\alpha})$ with a variational distribution $q(\Theta \mid \boldsymbol{\beta})$ with a set of variational parameters $\boldsymbol{\beta}$. This reduces an inference problem into an optimization problem in which the objective function is given by the ELBO:

$$\log p(\mathbf{x} \mid \boldsymbol{\alpha}) \geq \mathbb{E}_q[\log p(\mathbf{x}, \Theta|\boldsymbol{\alpha})] + \mathbb{H}[q(\Theta|\boldsymbol{\beta})] =: \widehat{\mathcal{L}}(\boldsymbol{\beta})$$

The variational distribution $q(\Theta \mid \boldsymbol{\beta})$ is typically assumed to be fully factorized, with each variational parameter β_i governing one latent variable θ_i. The variational parameters are then optimized to maximize the ELBO, and the optimal variational posterior will be used as a surrogate to the true posterior distribution $p(\Theta \mid \mathbf{X}, \boldsymbol{\alpha})$. It is worth noting that in standard VI the number of variational parameters is linearly proportional to the number of latent variables, including both the global and local latent variables.

3. Collapsed Variational Bayesian Inference

3.1. Motivation

Standard VI methods are computationally infeasible for SPNs due to the large number of local latent variables for each training instance, as shown in Fig. 1. Let $\mathbf{W}$ denote the set of global latent variables (model parameters) and $\mathbf{H}_d$ denote the set of local latent variables, where d indexes over the training instances. In standard VI we need to maintain a set of variational parameters for each of the latent variables, i.e., $\mathbf{W}$ and $\{\mathbf{H}_d\}_{d=1}^D$. In the case of SPNs, the number of local latent variables is exactly the number of internal sum nodes in the network, which can be linearly proportional to the size of the network, $|\mathcal{S}|$. Together with the global latent variables, this leads to a total number of $O(D|\mathcal{S}| + |\mathcal{S}|)$ variational parameters to be maintained in standard VI. As we will see in the experiments, this is prohibitively expensive for SPNs that range from tens of thousands to millions of nodes.

To deal with the expensive computation and storage in standard VI, we develop CVB-SPN, a collapsed variational algorithm, which, instead of assuming independence among the local latent variables, models the dependence among them in an exact fashion. More specifically, we marginalize all the local latent variables out of the joint posterior distribution and maintain a marginal posterior distribution over the global latent variables directly. This approach would not be an option for many graphical models, but as we will see later, we can take advantage of fast exact inference in SPNs. On the other hand, we still variationally approximate the posterior distribution of the global variables (model parameters) using a mean-field approach. Note this basic assumption made in CVB-SPN is different from typical collapsed variational approaches developed for LDA and HDP (Teh et al., 2006; 2007), where global latent variables are integrated out and local latent variables are assumed to be mutually independent. As a result, CVB-SPN models the marginal posterior distribution over global latent variables only and hence the number of variational parameters to be optimized is $O(|\mathcal{S}|)$ compared with $O(D|\mathcal{S}|)$ in the standard case.

Intuitively, this is not an unreasonable assumption to make since compared with local latent variables, the global latent variables in Fig. 1 are further away from the influence of observations of $\mathbf{X}$ and they are mutually independent a priori. Besides the computational consideration, another motivation to collapse out the local latent variables is that no interpretation associated with the local latent variables in an SPN has so far proven to be useful in practice. Unlike other mixture models such as LDA, where local latent variables correspond to topic assignment of words, the sum nodes in SPNs do not share typical statistical interpretations that may be useful in real-world applications.

CVB-SPN makes fewer assumptions about the independence among random variables and has fewer variational variables to be optimized, but to achieve these benefits, we must overcome the following difficulties: the cost of exact marginalization over local latent variables and the non-conjugacy between the prior distribution and the likelihood function introduced by the marginalization. The first problem is elegantly handled by the property of SPN that exact inference over $\mathbf{X}$ is always tractable. In what follows we proceed to derive the CVB-SPN algorithm that efficiently solves the second problem.

3.2. Collapsed Variational Inference

Throughout the derivation we will assume that the weights w_{kj} associated with a sum node k are locally normalized, i.e., $\sum_{j\in \mathrm{Ch}(k)} w_{kj} = 1, \forall k$. This can be achieved by a bottom-up pass of the network in time $O(|\mathcal{S}|)$ without changing the joint probability distribution over $\mathbf{X}$; see Peharz et al. (2015) and Zhao et al. (2015) for more details. For SPNs with locally normalized weights $\mathbf{w}$, it can be verified that $V_{\text{root}}(\mathbf{1} \mid \mathbf{w}) = 1$, hence the marginal likelihood function $p(\mathbf{x} \mid \mathbf{w})$ that marginalizes out local latent variables $\mathbf{h}$ is given by $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$.

Since the weights associated with each sum node k are locally normalized, we can interpret each sum node as a multinomial random variable with one possible value for each child of the sum node. It follows that we can specify a Dirichlet prior $\mathrm{Dir}(w_k|\alpha_k)$ for each sum node k. Since all the global latent variables are d-separated before we get the observation $\mathbf{x}$, a prior distribution over all the weights can be factorized as:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{k=1}^{m} p(w_k|\alpha_k) = \prod_{k=1}^{m} \mathrm{Dir}(w_k|\alpha_k) \tag{3}$$

Based on Thm. 2, the true posterior distribution after a sequence of observations $\{\mathbf{x}_d\}_{d=1}^D$ is:

$$\begin{aligned} p(\mathbf{w}|\{\mathbf{x}_d\}_{d=1}^D, \boldsymbol{\alpha}) &\propto p(\mathbf{w}|\boldsymbol{\alpha}) \prod_{d=1}^{D} V_{\text{root}}(\mathbf{x}_d|\mathbf{w}) \\ &= \prod_{k=1}^{m} \mathrm{Dir}(w_k|\alpha_k) \prod_{d=1}^{D} \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(x_{di}) \end{aligned} \tag{4}$$

Intuitively, Eq. 4 indicates that the true posterior distribution of the model parameters is a mixture model where the number of components scales as $\tau_{\mathcal{S}}^D$ and each component is a product of m Dirichlets, one for each sum node. Here $\tau_{\mathcal{S}}$ corresponds to the number of induced trees in $\mathcal{S}$ (cf. Thm. 2). For discrete SPNs the leaf univariate distributions are simply point mass distributions, i.e., indicator variables. For continuous distributions such as Gaussians, we also need to specify the priors for the parameters of those leaf

distributions, but the analysis goes the same as the discrete case. Eq. 4 is intractable to compute exactly, hence we resort to collapsed variational inference. To simplify notation, we will assume that there is only one instance in the data set, i.e., $D = 1$. Extension of the following derivation to multiple training instances is straightforward. Consider the log marginal probability over observable variables that upper bounds the new ELBO $\mathcal{L}(\boldsymbol{\beta})$:

$$\begin{aligned}\log p(\mathbf{x} \mid \boldsymbol{\alpha}) &\geq \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x}, \mathbf{w}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w}|\boldsymbol{\beta})] \qquad (5)\\ &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h}, \mathbf{w}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w}|\boldsymbol{\beta})]\\ &=: \mathcal{L}(\boldsymbol{\beta})\end{aligned}$$

where $q(\mathbf{w}|\boldsymbol{\beta}) = \prod_{k=1}^{m} \mathrm{Dir}(w_k|\beta_k)$ is the factorized variational distribution over $\mathbf{w}$ to approximate the true marginal posterior distribution $p(\mathbf{w}|\mathbf{x}, \boldsymbol{\alpha}) = \sum_{\mathbf{h}} p(\mathbf{w}, \mathbf{h}|\mathbf{x}, \boldsymbol{\alpha})$. Note that here we are using $q(\mathbf{w})$ as opposed to $q(\mathbf{w})q(\mathbf{h})$ in standard VI. We argue that $\mathcal{L}(\boldsymbol{\beta})$ gives us a better lower bound to optimize than the one given by standard VI. To see this, let $\widehat{\mathcal{L}}(\boldsymbol{\beta})$ be the ELBO given by standard VI, i.e.,

$$\widehat{\mathcal{L}}(\boldsymbol{\beta}) = \mathbf{E}_{q(\mathbf{w})q(\mathbf{h})}[\log p(\mathbf{x}, \mathbf{w}, \mathbf{h}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})q(\mathbf{h})]$$

we have:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta}) &= \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x}, \mathbf{w}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})]\\ &= \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{p(\mathbf{h}|\mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})}]\right] + \mathbb{H}[q(\mathbf{w})]\\ &= \max_{q(\mathbf{h}|\mathbf{w})} \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{q(\mathbf{h}|\mathbf{w})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{q(\mathbf{h}|\mathbf{w})}]\right] + \mathbb{H}[q(\mathbf{w})]\\ &\geq \max_{q(\mathbf{h})} \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{q(\mathbf{h})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{q(\mathbf{h})}]\right] + \mathbb{H}[q(\mathbf{w})]\\ &\geq \mathbb{E}_{q(\mathbf{w})q(\mathbf{h})}[\log p(\mathbf{x}, \mathbf{w}, \mathbf{h}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})q(\mathbf{h})]\\ &= \widehat{\mathcal{L}}(\boldsymbol{\beta})\end{aligned}$$

The first equality holds by the definition of $\mathcal{L}(\boldsymbol{\beta})$ and the second equality holds since $\log p(\mathbf{x}, \mathbf{w}|\boldsymbol{\alpha}) = \log \frac{p(\mathbf{h},\mathbf{w},\mathbf{x}|\boldsymbol{\alpha})}{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}$ is constant w.r.t. $\mathbb{E}_{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}[\cdot]$. The third equality holds because the inner expectation is the maximum that can be achieved by the negative KL divergence between the approximate posterior $q(\mathbf{h}|\mathbf{w})$ and the true posterior $p(\mathbf{h}|\mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})$. The inequality in the fourth line is due to the fact that the maximization over $q(\mathbf{h}|\mathbf{w})$ is free of constraint hence the optimal posterior is given by the true conditional posterior $q^*(\mathbf{h}|\mathbf{w}) = p(\mathbf{h}|\mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})$, while the maximization over $q(\mathbf{h})$ in the fourth line needs to satisfy the independence assumption made in standard VI, i.e., $q(\mathbf{h}|\mathbf{w}) = q(\mathbf{h})$. Combining the inequality above with (5), we have

$$\log p(\mathbf{x} \mid \boldsymbol{\alpha}) \geq \mathcal{L}(\boldsymbol{\beta}) \geq \widehat{\mathcal{L}}(\boldsymbol{\beta}) \qquad (6)$$

which shows that the ELBO given by the collapsed variational inference is a better lower bound than the one given by standard VI. This conclusion is also consistent with the one obtained in collapsed variational inference for LDA and HDP (Teh et al., 2006; 2007) where the authors collapsed out the global latent variables instead of the local latent variables. It is straightforward to verify that the difference between $\log p(\mathbf{x}|\boldsymbol{\alpha})$ and $\mathcal{L}(\boldsymbol{\beta})$ is given by $\mathbb{KL}(q(\mathbf{w}|\boldsymbol{\beta}) \parallel p(\mathbf{w}|\mathbf{x}, \boldsymbol{\alpha}))$, i.e., the KL-divergence between the variational marginal posterior and the exact marginal posterior distribution. Substituting $q(\mathbf{w}|\boldsymbol{\beta})$ into the KL-divergence objective and simplifying it, we reach the following optimization objective:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \mathbb{KL}(q(\mathbf{w}|\boldsymbol{\beta}) \parallel p(\mathbf{w}|\boldsymbol{\alpha})) - \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x}|\mathbf{w})] \qquad (7)$$

where the first part can be interpreted as a regularization term that penalizes a variational posterior that is too far away from the prior, and the second part is a data-fitting term that requires the variational posterior to have a good fit to the training data set. The first part in (7) can be efficiently computed due to the factorization assumption of both the prior and the variational posterior. However, the second part does not admit an analytic form because after we marginalize out all the local latent variables, $q(\mathbf{w}|\boldsymbol{\beta})$ is no longer conjugate to the likelihood function $p(\mathbf{x}|\mathbf{w}) = V_{\text{root}}(\mathbf{x}|\mathbf{w})$. We address this problem in the next subsection.

3.3. Upper Bound by Logarithmic Transformation

The hardness of computing $\mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x}|\mathbf{w})]$ makes the direct optimization of (7) infeasible in practice. While nonconjugate inference with little analytic work is possible with recent innovations (Blei et al., 2012; Kingma & Welling, 2013; Ranganath et al., 2014; Mnih & Gregor, 2014; Titsias & Lázaro-Gredilla, 2014; Titsias, 2015), they fail to make use of a key analytic property of SPNs, i.e., easy marginalization, that allows for the optimal variational distributions of local variables to be used. In this section we show how to use a logarithmic transformation trick to develop an upper bound of the objective in (7) which leads to the efficient CVB-SPN algorithm.

The key observation is that the likelihood of $\mathbf{w}$ is a posynomial function (Boyd et al., 2007). To see this, as shown in Thm. 2, we have $p(\mathbf{x}|\mathbf{w}) = \sum_{t=1}^{\tau_S} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$. The product term with respect to $\mathbf{x}$, $\prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$, is guaranteed to be nonnegative and can be treated as a constant w.r.t. $\mathbf{w}$. Hence each component in $p(\mathbf{x}|\mathbf{w})$, in terms of $\mathbf{w}$, is a monomial function with positive multiplicative constant, and it follows that $p(\mathbf{x}|\mathbf{w})$ is a posynomial function of $\mathbf{w}$. A natural implication of this observation is that we can do a change of variables transformation to $\mathbf{w}$ such that $\log p(\mathbf{x}|\mathbf{w})$ becomes a convex function in terms of the transformed variables. More specifically, let $\mathbf{w}' = \log \mathbf{w}$, where $\log(\cdot)$ is taken elementwise. The log likelihood, now

expressed in terms of $\mathbf{w}'$, is

$$\begin{aligned}\log p(\mathbf{x} \mid \mathbf{w}) &= \log \left(\sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i) \right) \\ &= \log \left(\sum_{t=1}^{\tau_{\mathcal{S}}} \exp \left(c_t + \sum_{(k,j)\in\mathcal{T}_{tE}} w'_{kj} \right) \right) \\ &=: \log p(\mathbf{x} \mid \mathbf{w}') \end{aligned} \tag{8}$$

where c_t is defined as $c_t = \log \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$. For each unique tree $\mathcal{T}_t$, $c_t + \sum_{(k,j)\in\mathcal{T}_{tE}} w'_{kj}$ is an affine function of $\mathbf{w}'$. Since the log-sum-exp function is convex in its argument, $\log p(\mathbf{x} \mid \mathbf{w}')$ is convex in $\mathbf{w}'$. Such a change of variables trick is frequently applied in the geometric programming literature to transform a non-convex posynomial optimization problem into a convex programming problem (Boyd et al., 2007; Chiang, 2005).

The convexity of $\log p(\mathbf{x} \mid \mathbf{w}')$ helps us to develop a lower bound of $\mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x}|\mathbf{w})]$ that is efficient to compute. Let $q'(\mathbf{w}')$ be the corresponding variational distribution over $\mathbf{w}'$ induced from $q(\mathbf{w})$ by the bijective transformation between $\mathbf{w}$ and $\mathbf{w}'$. We have

$$\begin{aligned}\mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x} \mid \mathbf{w})] &= \int q(\mathbf{w}) \log p(\mathbf{x} \mid \mathbf{w}) \, d\mathbf{w} \\ &= \int q'(\mathbf{w}') \log p(\mathbf{x} \mid \mathbf{w}') \, d\mathbf{w}' \\ &\geq \log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}'])\end{aligned}$$

where $\log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}'])$ means the log-likelihood of $\mathbf{x}$ with the edge weights set to be $\exp(\mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}'])$. Since $q(\mathbf{w}) = \prod_{k=1}^{m} \mathrm{Dir}(w_k|\beta_k)$ is a product of Dirichlets, we can compute the weight $\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right)$ for each edge (k, j) as

$$\begin{aligned}\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}] &= \int q'(\mathbf{w}') w'_{kj} \, d\mathbf{w}' = \int q'(w'_k) w'_{kj} \, dw'_k \\ &= \int q(w_k) \log w_{kj} \, dw_k = \psi(\beta_{kj}) - \psi(\sum_{j'} \beta_{kj'})\end{aligned} \tag{9}$$

where $\psi(\cdot)$ is the digamma function. The equation above then implies the new edge weight can be computed by

$$\begin{aligned}\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right) &= \exp\left(\psi(\beta_{kj}) - \psi(\sum_{j'} \beta_{kj'}) \right) \\ &\approx \frac{\beta_{kj} - \frac{1}{2}}{\sum_{j'} \beta_{kj'} - \frac{1}{2}}\end{aligned} \tag{10}$$

where the approximation is by $\exp(\psi(x)) \approx x - \frac{1}{2}$ when $x > 1$. Note that the mean of the variational posterior is given by $\bar{w}_{kj} = \mathbb{E}_{q(\mathbf{w})}[w_{kj}] = \beta_{kj} / \sum_{j'} \beta_{kj'}$, which is close to $\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right)$ when $\beta_{kj} > 1$. Roughly speaking, this shows that the lower bound we obtained for $\mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x} \mid \mathbf{w})]$ by utilizing the logarithmic transformation is trying to optimize the variational parameters $\boldsymbol{\beta}$ such that the variational posterior mean has a good fit to the training data. This is exactly what we hope for since at the end we need to use the variational posterior mean as a Bayesian estimator of our model parameter. Combining all the analysis above, we formulate the following objective function to be minimized that is an upper bound of the objective function given in (7):

$$\mathbb{KL}(q(\mathbf{w}|\boldsymbol{\beta}) \parallel p(\mathbf{w}|\boldsymbol{\alpha})) - \log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}'|\boldsymbol{\beta})}[\mathbf{w}']) \tag{11}$$

Note that we will use the approximation given by Eq. 10 in the above optimization formulation and in Alg. 1. This is a non-convex optimization problem due to the digamma function involved. Nevertheless we can still achieve a local optimum with projected gradient descent in the experiments. We summarize our algorithm, CVB-SPN, in Alg. 1. For each instance, the gradient of the objective function in (11) with respect to $\boldsymbol{\beta}$ can be computed by a bottom-up and top-down pass of the network in time $O(|\mathcal{S}|)$. Please refer to (Zhao & Poupart, 2015; Peharz et al., 2016) for more details.

Algorithm 1 CVB-SPN

Input: Initial $\boldsymbol{\beta}$, prior hyperparameter $\boldsymbol{\alpha}$, training instances $\{\mathbf{x}_d\}_{d=1}^{D}$.
Output: Locally optimal $\boldsymbol{\beta}^*$.

1: **while** not converged **do**
2: Update $\mathbf{w} = \exp(\mathbb{E}_{q'(\mathbf{w}'|\boldsymbol{\beta})}[\mathbf{w}'])$ with Eq. 10.
3: Set $\nabla_{\boldsymbol{\beta}} = 0$.
4: **for** $d = 1$ to D **do**
5: Bottom-up evaluation of $\log p(\mathbf{x}_d|\mathbf{w})$.
6: Top-down differentiation of $\frac{\partial}{\partial \mathbf{w}} \log p(\mathbf{x}_d|\mathbf{w})$.
7: Update $\nabla_{\boldsymbol{\beta}}$ based on $\mathbf{x}_d$.
8: **end for**
9: Update $\nabla_{\boldsymbol{\beta}}$ based on $\mathbb{KL}(q(\mathbf{w}|\boldsymbol{\beta}) \parallel p(\mathbf{w}|\boldsymbol{\alpha}))$.
10: Update $\boldsymbol{\beta}$ with projected GD.
11: **end while**

3.4. Parallel and Stochastic Variants

Note that Alg. 1 is easily parallelizable by splitting training instances in the loop (Line 4–Line 8) across threads. It can also be extended to the stochastic setting where at each round we sample one instance or a mini-batch of instances $\{\mathbf{x}_{i_k}\}_{k=1}^{s}$ (s is the size of the mini-batch) from the training set. The stochastic formulation will be helpful when the size of the full training data set is too large to be stored in the main memory as a whole, or when the training instances are streaming so that at each time we only have access to one instance (Hoffman et al., 2013).

Table 1. Statistics of data sets and models. n is the number of observable random variables modeled by the network, $|\mathcal{S}|$ is the size of the network and p is the number of parameters to be estimated. $n \times D/p$ means the ratio of training instances times the number of variables to the number of parameters.

Data set	n	$\|\mathcal{S}\|$	p	Train	Valid	Test	$n \times D/p$
NLTCS	16	13,733	1,716	16,181	2,157	3,236	150.871
MSNBC	17	54,839	24,452	291,326	38,843	58,265	202.541
KDD 2k	64	48,279	14,292	180,092	19,907	34,955	806.457
Plants	69	132,959	58,853	17,412	2,321	3,482	20.414
Audio	100	739,525	196,103	15,000	2,000	3,000	7.649
Jester	100	314,013	180,750	9,000	1,000	4,116	4.979
Netflix	100	161,655	51,601	15,000	2,000	3,000	29.069
Accidents	111	204,501	74,804	12,758	1,700	2,551	18.931
Retail	135	56,931	22,113	22,041	2,938	4,408	134.560
Pumsb-star	163	140,339	63,173	12,262	1,635	2,452	31.638
DNA	180	108,021	52,121	1,600	400	1,186	5.526
Kosarak	190	203,321	53,204	33,375	4,450	6,675	119.187
MSWeb	294	68,853	20,346	29,441	3,270	5,000	425.423
Book	500	190,625	41,122	8,700	1,159	1,739	105.783
EachMovie	500	522,753	188,387	4,524	1,002	591	12.007
WebKB	839	1,439,751	879,893	2,803	558	838	2.673
Reuters-52	889	2,210,325	1,453,390	6,532	1,028	1,540	3.995
20 Newsgrp	910	14,561,965	8,295,407	11,293	3,764	3,764	1.239
BBC	1058	1,879,921	1,222,536	1,670	225	330	1.445
Ad	1556	4,133,421	1,380,676	2,461	327	491	2.774

4. Experiments

4.1. Experimental Setting

We conduct experiments on a set of 20 benchmark data sets to compare the performance of the proposed collapsed variational inference method with maximum likelihood estimation (Gens & Domingos, 2012). The 20 real-world data sets used in the experiments have been widely used (Rooshenas & Lowd, 2014) to assess the modeling performance of SPNs. All the features are binary. The 20 data sets also cover both low dimensional and high dimensional statistical estimation, hence, they enable a thorough experimental comparison. Detailed information about each data set as well as the SPN models is shown in Table 1. All the SPNs are built using LearnSPN (Gens & Domingos, 2013).

We evaluate and compare the performance of CVB-SPN with other parameter learning algorithms in both the batch and online learning settings. For a baseline, we compare to the state-of-the-art parameter learning algorithm for SPNs where the algorithm optimizes the training set log-likelihood directly in order to find a maximum likelihood estimator, which we will denote as MLE-SPN. We compare CVB-SPN and MLE-SPN in both batch and online cases. To have a fair comparison, we apply the same optimization method, i.e., projected gradient descent, to optimize the objective functions in CVB-SPN and MLE-SPN. CVB-SPN optimizes over the variational parameters of the posterior distribution based on (11) while MLE-SPN optimizes directly over the model parameters to maximize the log-likelihood of the training set. Since the optimization variables are constrained to be positive in both CVB-SPN and MLE-SPN, we need to project the parameters back onto the positive orthant after every iteration. We fix the projection margin ϵ to 0.01, i.e., $\mathbf{w} = \max\{\mathbf{w}, 0.01\}$ to avoid numerical issues. We implement both methods with backtracking line search to automatically adjust the learning rate at each iteration. In all experiments, the maximum number of iterations is fixed to 50 for both methods. We discard the model parameters returned by LearnSPN and use random weights as initial model parameters. CVB-SPN is more flexible to incorporate those model weights returned by LearnSPN as the hyperparameters for prior distributions. In the experiments we multiply the weights returned by LearnSPN by a positive scalar and treat them as the hyperparameters of the prior Dirichlet distributions. This is slightly better than using randomly initialized priors, but the differences are negligible on most data sets. MLE-SPN can also incorporate the model weights returned by LearnSPN by treating them as the hyperparameters of fixed prior Dirichlets. This corresponds to an MAP formulation. However in practice we find the MAP formulation performs no better than MLE-SPN, and on small data sets, MAP-SPN gives consistently worse results than MLE-SPN; so, we report only results for MLE-SPN. For both MLE-SPN and CVB-SPN we use a held-out validation set to pick the best solution during the optimization process. We report average log-likelihood on each data set for each method.

Both CVB-SPN and MLE-SPN are easily extended to the online setting where training instances are coming in a streaming fashion. In this case we also compare CVB-SPN and MLE-SPN to an online Bayesian moment matching

Table 2. Average log-likelihoods on test data. Highest average log-likelihoods are highlighted in bold. ↑ / ↓ are used to represent statistically better/worse results than (O)CVB-SPN respectively.

Data set	MLE-SPN	CVB-SPN	OBMM	OMLE-SPN	OCVB-SPN
NLTCS	↓ -6.44	**-6.08**	↑**-6.07**	↓-7.78	-6.12
MSNBC	↓-7.02	**-6.29**	-6.35	↓-6.94	**-6.34**
KDD 2k	↓-4.24	**-2.14**	**-2.14**	↓-27.99	-2.16
Plants	↓-28.78	**-12.86**	↑**-15.14**	↓-30.23	-16.03
Audio	↓-46.42	**-40.36**	↓-40.70	↓-48.90	**-40.58**
Jester	↓-59.55	**-54.26**	-53.86	↓-63.67	**-53.84**
Netflix	↓-64.88	**-60.69**	-57.99	↓-65.72	**-57.96**
Accidents	↓-50.14	**-29.55**	↓-42.66	↓-58.63	**-38.07**
Retail	↓-15.53	**-10.91**	↓-11.42	↓-82.42	**-11.31**
Pumsb-star	↓-80.61	**-25.93**	↓-45.27	↓-80.19	**-37.05**
DNA	↓-102.62	**-86.73**	↓-99.61	↓-96.84	**-91.52**
Kosarak	↓-47.16	**-10.70**	-11.22	↓-111.95	**-11.12**
MSWeb	↓-19.69	**-9.89**	↓-11.33	↓-140.86	**-10.73**
Book	↓-88.16	**-34.44**	↓-35.55	↓-299.02	**-34.77**
EachMovie	↓-97.15	**-52.63**	↑**-59.50**	↓-284.92	-64.75
WebKB	↓-199.15	**-161.46**	↑**-165.57**	↓-413.94	-169.31
Reuters-52	↓-218.97	**-85.45**	**-108.01**	↓-513.97	-108.04
20 Newsgrp	↓-260.69	**-155.61**	↓-158.01	↓-728.11	**-156.63**
BBC	↓-372.45	**-251.23**	↓-275.43	↓-517.36	**-272.56**
Ad	↓-311.87	**-19.00**	↓-63.81	↓-572.01	**-57.56**

method (OBMM) (Rashwan et al., 2016) that is designed for learning SPNs. OBMM is a purely online algorithm in the sense that it can only process one instance in each update in order to avoid the exponential blow-up in the number of mixture components. OBMM constructs an approximate posterior distribution after seeing each instance by matching the first and second moments of the approximate posterior to the exact posterior. In this experiment, we compare both CVB-SPN (OCVB-SPN) and MLE-SPN (OMLE-SPN) to OBMM where only one pass over the training set is allowed for learning and at each round only one instance is available to each of the algorithms.

4.2. Results

All experiments are run on a server with Intel Xeon CPU E5 2.00GHz. The running time ranges from 2 min to around 5 days depending on the size of the data set and the size of the network. All algorithms have roughly the same running time on the same data set as they scale linearly in the size of the training set and the size of the network. Table 2 shows the average joint log-likelihood scores of different parameter learning algorithms on 20 data sets. For each configuration, we use bold numbers to highlight the best score among the offline methods and among the online methods. We also use ↑ / ↓ to indicate whether the competitor methods achieve statistically significant better/worse results than CVB-SPN on the corresponding test data set under the Wilcoxon signed-rank test (Wilcoxon, 1950) with p-value ≤ 0.05. In the batch learning experiment, CVB-SPN consistently dominates MLE-SPN on every data set with a large margin. The same results can be observed in the online learning scenarios: both OCVB-SPN and OBMM significantly outperform OMLE-SPN on all the 20 experiments. These experiments demonstrate the effectiveness and robustness of Bayesian inference methods in parameter estimation on large graphical models like SPNs. In the online learning scenario, OCVB-SPN beats OBMM on 14 out of the 20 data sets, and achieves statistically better results on 10 out of the 14. On the other hand, OBMM obtains statistically better results than OCVB-SPN on 4 of the data sets. This is partly due to the fact that OCVB-SPN explicitly optimizes over an objective function that includes the evaluation of the variational posterior mean on the likelihood function, while OBMM only tries to match the first and second moments of the distribution after each update.

5. Conclusion

We develop a collapsed variational inference method, CVB-SPN, to learn the parameters of SPNs. CVB-SPN directly maintains a variational posterior distribution over the global latent variables by marginalizing out all the local latent variables. As a result, CVB-SPN is more memory efficient than standard VI. We also show that the collapsed ELBO in CVB-SPN is a better lower bound than the standard ELBO. We construct a logarithmic transformation trick to avoid the intractable computation of a high-dimensional expectation. We conduct experiments on 20 data sets to compare the proposed algorithm with state-of-the-art learning algorithms in both batch and online settings. The results demonstrate the effectiveness of CVB-SPN in both cases.

Acknowledgements

HZ and GG gratefully acknowledge support from ONR contract N000141512365. TA is partially funded by the Netherlands NWO, project 612.001.119. BA acknowledges support from the National Science Foundation (NSF) grant number CNS-1518865, the Intel Corporation, Google, Vodafone, NVIDIA, and the Conklin Kistler family fund.

References

Blei, David M, Ng, Andrew Y, and Jordan, Michael I. Latent Dirichlet Allocation. *the Journal of machine Learning Research*, 3:993–1022, 2003.

Blei, David M, Jordan, Michael I, and Paisley, John W. Variational Bayesian Inference with Stochastic Search. In *ICML*, 2012.

Boyd, Stephen, Kim, Seung-Jean, Vandenberghe, Lieven, and Hassibi, Arash. A Tutorial on Geometric Programming. *Optimization and Engineering*, 8(1):67–127, 2007.

Casella, George and Berger, Roger L. *Statistical Inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

Chiang, Mung. *Geometric Programming for Communication Systems*. Now Publishers Inc, 2005.

Gens, Robert and Domingos, Pedro. Discriminative Learning of Sum-Product Networks. pp. 3248–3256, 2012.

Gens, Robert and Domingos, Pedro. Learning the Structure of Sum-Product Networks. In *ICML*, 2013.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, and Saul, Lawrence K. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

Kingma, Diederik P and Welling, Max. Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Mnih, Andriy and Gregor, Karol. Neural Variational Inference and Learning in Belief Networks. In *ICML*, 2014.

Peharz, Robert, Tschiatschek, Sebastian, Pernkopf, Franz, and Domingos, Pedro. On Theoretical Properties of Sum-Product Networks. In *AISTATS*, 2015.

Peharz, Robert, Gens, Robert, Pernkopf, Franz, and Domingos, Pedro. On the Latent Variable Interpretation in Sum-Product Networks. *arXiv preprint arXiv:1601.06180*, 2016.

Poon, Hoifung and Domingos, Pedro. Sum-Product Networks: A New Deep Architecture. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, 2011.

Ranganath, Rajesh, Gerrish, Sean, and Blei, David. Black Box Variational Inference. In *AISTATS*, pp. 814–822, 2014.

Rashwan, Abdullah, Zhao, Han, and Poupart, Pascal. Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks. In *International Conference on Artificial Intelligence and Statistics*, 2016.

Rooshenas, Amirmohammad and Lowd, Daniel. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 710–718, 2014.

Teh, Yee W, Newman, David, and Welling, Max. A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. In *Advances in neural information processing systems*, 2006.

Teh, Yee W, Kurihara, Kenichi, and Welling, Max. Collapsed Variational Inference for HDP. In *Advances in neural information processing systems*, pp. 1481–1488, 2007.

Titsias, Michalis and Lázaro-Gredilla, Miguel. Doubly Stochastic Variational Bayes for Non-conjugate Inference. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Titsias, Michalis K. Local Expectation Gradients for Doubly Stochastic Variational Inference. *arXiv preprint arXiv:1503.01494*, 2015.

Wainwright, Martin J and Jordan, Michael I. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Wilcoxon, Frank. Some Rapid Approximate Statistical Procedures. *Annals of the New York Academy of Sciences*, pp. 808–814, 1950.

Zhao, Han and Poupart, Pascal. A Unified Approach for Learning the Parameters of Sum-Product Networks. *arXiv preprint arXiv:1601.00318*, 2015.

Zhao, Han, Melibari, Mazen, and Poupart, Pascal. On the Relationship between Sum-Product Networks and Bayesian Networks. In *ICML*, 2015.