On the Latent Variable Interpretation in Sum-Product Networks

Robert Peharz, Robert Gens, Franz Pernkopf, *Senior Member, IEEE,* and Pedro Domingos

Abstract—One of the central themes in Sum-Product networks (SPNs) is the interpretation of sum nodes as marginalized latent variables (LVs). This interpretation yields an increased syntactic or semantic structure, allows the application of the EM algorithm and to efficiently perform MPE inference. In literature, the LV interpretation was justified by explicitly introducing the indicator variables corresponding to the LVs' states. However, as pointed out in this paper, this approach is in conflict with the completeness condition in SPNs and does not fully specify the probabilistic model. We propose a remedy for this problem by modifying the original approach for introducing the LVs, which we call SPN augmentation. We discuss conditional independencies in augmented SPNs, formally establish the probabilistic interpretation of the sum-weights and give an interpretation of augmented SPNs as Bayesian networks. Based on these results, we find a sound derivation of the EM algorithm for SPNs. Furthermore, the Viterbi-style algorithm for MPE proposed in literature was never proven to be correct. We show that this is indeed a correct algorithm, when applied to selective SPNs, and in particular when applied to augmented SPNs. Our theoretical results are confirmed in experiments on synthetic data and 103 real-world datasets.

Index Terms—Sum-Product Networks, Latent Variables, Mixture Models, Expectation-Maximization, MPE inference

1 Introduction

Sum-Product Networks are a promising type of probabilistic model, combining the domains of deep learning and graphical models [1], [2]. One of their main advantages is that many interesting inference scenarios are expressed as single forward and/or backward passes, i.e. these inference scenarios have a computational cost linear in the SPN's representation size. SPNs have shown convincing performance in applications such as image completion [1], [3], [4], computer vision [5], classification [6] and speech and language modeling [7], [8], [9]. Since their proposition [1], one of the central themes in SPNs has been their interpretation as hierarchically structured latent variable (LV) models. This is essentially the same approach as the LV interpretation in mixture models. Consider for example a Gaussian mixture model with K components over a set of random variables (RVs) $\mathbf{X}$:

$$p(\mathbf{X}) = \sum_{k=1}^{K} w_k \, \mathcal{N}(\mathbf{X} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{1}$$

where $\mathcal{N}(\cdot\,|\,\cdot)$ is the Gaussian PDF, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the means and covariances of the k^{th} component, and w_k are the mixture weights with $w_k \geqslant 0$, $\sum w_k = 1$. The GMM can be interpreted in two ways: i) It is a convex combination of PDFs and thus itself a PDF, or ii) it is a marginal distribution of a distribution $p(\mathbf{X}, Z)$ over $\mathbf{X}$ and a latent, marginalized variable Z, where $p(\mathbf{X} \,|\, Z = k) = \mathcal{N}(\mathbf{X} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and $p(Z = k) = w_k$. The second interpretation, the LV interpretation, yields a syntactically well-structured model. For example, following the LV interpretation, it is clear how to draw samples from $p(\mathbf{X})$ by using ancestral sampling. This structure can also be of semantic nature, for instance when Z represents a clustering of $\mathbf{X}$ or when Z is a class variable. Furthermore, the LV interpretation allows the application of the EM algorithm – which is essentially maximum-likelihood learning under missing data [10], [11] – and enables advanced Bayesian techniques [12], [13].

Mixture models can be seen as a special case of SPNs with a single sum node, which corresponds to a single LV. More generally, SPNs can have arbitrarily many sum nodes, each corresponding to its own LV, leading to a hierarchically structured model. In [1], the LV interpretation in SPNs was justified by explicitly introducing the LVs in the SPN model, using the so-called *indicator variables* corresponding to the LVs' states. However, as shown in this paper, this justification is actually too simplistic, since it is potentially in conflict with the *completeness* condition [1], leading to an incompletely specified model. As a remedy we propose the *augmentation* of an SPN, which additionally to the IVs also introduces the so-called *twin sum nodes*, in order to completely specify the LV model. We further investigate the independency structure of the LV model resulting from augmentation and find a parallel to the local independence assertions in Bayesian networks (BNs) [14], [15]. This allows us to define a BN representation of the augmented SPN. Using our BN interpretation and the differential approach [16], [17] in augmented SPNs, we give a sound derivation of the (soft) EM algorithm for SPNs.

Closely related to the LV interpretation is the inference scenario of finding the most-probable-explanation (MPE),

- *R. Peharz is with the Institute of Physiology (iDN), Medical University of Graz and BioTechMed–Graz.*
  E-mail: robert.peharz@gmail.com
- *R. Gens and P. Domingos are with the Department of Computer Science and Engineering, University of Washington.*
  E-mail: rcg@cs.washington.edu, pedrod@cs.washington.edu
- *F. Pernkopf is with the Signal Processing and Speech Communication Lab, Graz University of Technology.*
  E-mail: pernkopf@tugraz.at

Manuscript received November 12, 2015; revised June 20, 2016; accepted September 30, 2016.

i.e. finding a probability maximizing assignment for all RVs. Using results form [18], [19], we first point out that this problem is generally NP-hard for SPNs. In [1] it was proposed that an MPE solution can be found efficiently when maximizing over both model RVs (i.e. non-latent RVs) and LVs. The proposed algorithm replaces sum nodes by max nodes and recovers the solution by using Viterbi-style backtracking. However, it was not shown that this algorithm delivers a correct MPE solution. In this paper, we show that this algorithm is indeed correct, *when applied to selective SPNs* [20]. In particular, since augmented SPNs are selective, this algorithm obtains an MPE solution in augmented SPNs. However, when applied to *non-augmented* SPNs, the algorithm still returns an MPE solution of the augmented SPN, but implicitly assumes that the weights for all twin sums are deterministic, i.e. they are all 0 except a single 1. This leads to a phenomenon in MPE inference which we call *low-depth bias*, i.e. more shallow parts of the SPN are preferred during backtracking.

The main contribution in this paper is to provide a sound theoretical foundation for the LV interpretation in SPNs and related concepts, i.e. the EM algorithm and MPE inference. Our theoretical findings are confirmed in experiments on synthetic data and 103 real-world datasets.

The paper is organized as follows: In the remainder of this section we introduce notation, review SPNs and discuss related work. In Section 2 we propose the augmentation of SPNs, show its soundness as hierarchical LV model and give an interpretation as BN. Furthermore, we discuss independency properties in augmented SPNs and the interpretation of sum-weights as conditional probabilities. The EM algorithm for SPNs is derived in Section 3. In Section 4 we discuss MPE inference for SPNs. Experiments are presented in Section 5 and Section 6 concludes the paper. Proofs for our theoretical findings are deferred to the Appendix.

1.1 Background and Notation

RVs are denoted by upper-case letters W, X, Y and Z. The set of values of an RV X is denoted by $\mathbf{val}(X)$, where corresponding lower-case letters denote elements of $\mathbf{val}(X)$, e.g. x is an element of $\mathbf{val}(X)$. Sets of RVs are denoted by boldface letters $\mathbf{W}$, $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. For RV set $\mathbf{X} = \{X_1, \dots, X_N\}$, we define $\mathbf{val}(\mathbf{X}) = \times_{n=1}^{N} \mathbf{val}(X_n)$ and use corresponding lower-case boldface letters for elements of $\mathbf{val}(\mathbf{X})$, e.g. $\mathbf{x}$ is an element of $\mathbf{val}(\mathbf{X})$. For a subset $\mathbf{Y} \subseteq \mathbf{X}$, $\mathbf{x}[\mathbf{Y}]$ denotes the projection of $\mathbf{x}$ onto $\mathbf{Y}$.

The elements of $\mathbf{val}(\mathbf{X})$ can be interpreted as *complete* evidence, assigning each RV in $\mathbf{X}$ a fixed value. *Partial* evidence about X is represented as a subset $\mathcal{X} \subseteq \mathbf{val}(X)$, which is an element of the sigma-algebra $\mathcal{A}_X$ induced by RV X. For all RVs we use $\mathcal{A}_X = \{\mathcal{X} \in \boldsymbol{\mathcal{B}} \mid \mathcal{X} \subseteq \mathbf{val}(X)\}$, $\boldsymbol{\mathcal{B}}$ being the Borel-sets over $\mathbb{R}$. For discrete RVs, this choice yields the power-set $\mathcal{A}_X = 2^{\mathbf{val}(X)}$. For example, partial evidence $\mathcal{X} = \{1, 3, 5\}$ for a discrete RV X with $\mathbf{val}(X) = \{1, \dots, 6\}$ represents evidence that X takes one of the states 1, 3 or 5, and $\mathcal{Y} = [-\infty, \pi]$ for a real-valued RV Y represents evidence that Y takes a value smaller than π. Formally speaking, partial evidence is used to express the domain of *marginalization or maximization* for a particular RV.

For sets of RVs $\mathbf{X} = \{X_1, \dots, X_N\}$, we use the product sets $\mathcal{H}_{\mathbf{X}} := \{\times_{n=1}^{N} \mathcal{X}_n \mid \mathcal{X}_n \in \mathcal{A}_{X_n}\}$ to represent partial evidence about $\mathbf{X}$. Elements of $\mathcal{H}_{\mathbf{X}}$ are denoted using boldface notation, e.g. $\boldsymbol{\mathcal{X}}$. When $\mathbf{Y} \subseteq \mathbf{X}$ and $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$, we define $\boldsymbol{\mathcal{X}}[\mathbf{Y}] := \{\mathbf{x}[\mathbf{Y}] \mid \mathbf{x} \in \boldsymbol{\mathcal{X}}\}$. Furthermore, we use $\mathbf{e}$ to symbolize any combination of complete and partial evidence, i.e. for RVs $\mathbf{X}$ we have some complete evidence $\mathbf{x}'$ for $\mathbf{X}' \subseteq \mathbf{X}$ and some partial evidence $\boldsymbol{\mathcal{X}}'' \in \mathcal{H}_{\mathbf{X}''}$ for $\mathbf{X}'' = \mathbf{X} \setminus \mathbf{X}'$.

Given a node N in some directed graph $\mathcal{G}$, let $\mathbf{ch}(\mathsf{N})$ and $\mathbf{pa}(\mathsf{N})$ be the set of children and parents of N, respectively. Furthermore, let $\mathbf{desc}(\mathsf{N})$ be the set of descendants of N, recursively defined as the set containing N itself and any child of a descendant. Similarly, we define $\mathbf{anc}(\mathsf{N})$ as the ancestors of N, recursively defined as the set containing N itself and any parent of an ancestor. SPNs are defined as follows.

Definition 1 (Sum-Product Network)**.** *A Sum-Product network (SPN) $\mathcal{S}$ over a set of RVs $\mathbf{X}$ is a tuple $(\mathcal{G}, \boldsymbol{w})$ where $\mathcal{G}$ is a connected, rooted and acyclic directed graph, and $\boldsymbol{w}$ is a set of non-negative parameters. The graph $\mathcal{G}$ contains three types of nodes: distributions, sums and products. All* leaves *of $\mathcal{G}$ are distributions and all* internal *nodes are either sums or products. A distribution node (also called input distribution or simply distribution) $\mathsf{D}_{\mathbf{Y}} \colon \mathbf{val}(\mathbf{Y}) \mapsto [0, \infty]$ is a distribution function over a subset of RVs $\mathbf{Y} \subseteq \mathbf{X}$, i.e. either a PMF (discrete RVs), a PDF (continuous RVs), or a mixed distribution function (discrete and continuous RVs mixed). A sum node S computes a weighted sum of its children, i.e. $\mathsf{S} = \sum_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}}\, \mathsf{C}$, where $w_{\mathsf{S},\mathsf{C}}$ is a non-negative weight associated with edge $\mathsf{S} \rightarrow \mathsf{C}$, and $\boldsymbol{w}$ contains the weights for all outgoing sum-edges. A product node P computes the product over its children, i.e. $\mathsf{P} = \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}$. The sets $\mathbf{S}(\mathcal{S})$ and $\mathbf{P}(\mathcal{S})$ contain all sum nodes and all product nodes in $\mathcal{S}$, respectively.*

The size *$|\mathcal{S}|$ of the SPN is defined as the number of nodes and edges in $\mathcal{G}$. For any node N in $\mathcal{G}$, the* scope *of N is defined as*

$$\mathbf{sc}(\mathsf{N}) = \begin{cases} \mathbf{Y} & \text{if } \mathsf{N} \text{ is a distribution } \mathsf{D}_{\mathbf{Y}} \\ \bigcup_{\mathsf{C} \in \mathbf{ch}(\mathsf{N})} \mathbf{sc}(\mathsf{C}) & \text{otherwise.} \end{cases} \tag{2}$$

The function computed by $\mathcal{S}$ is the function computed by its root and denoted as $\mathcal{S}(\mathbf{x})$, where without loss of generality we assume that the scope of the root is $\mathbf{X}$.

We use symbols D, S, P, N, C and F for nodes in SPNs, where D denotes a distribution, S denotes a sum, and P denotes a product. Symbols N, C and F denote generic nodes, where C and F indicate a child or parent relationship to another node, respectively. The *distribution* $p_{\mathcal{S}}$ of an SPN $\mathcal{S}$ is defined as the normalized output of $\mathcal{S}$, i.e. $p_{\mathcal{S}}(\mathbf{x}) \propto \mathcal{S}(\mathbf{x})$. For each node N, we define the *sub-SPN* $\mathcal{S}_{\mathsf{N}}$ rooted at N as the SPN defined by the graph induced by the descendants of N and the corresponding parameters.

Inference in unconstrained SPNs is generally intractable. However, efficient inference in SPNs is enabled by two structural constraints, *completeness* and *decomposability* [1]. An SPN is *complete* if for all sums S it holds that

$$\forall \mathsf{C}', \mathsf{C}'' \in \mathbf{ch}(\mathsf{S}) \colon \mathbf{sc}(\mathsf{C}') = \mathbf{sc}(\mathsf{C}''). \tag{3}$$

An SPN is *decomposable* if for all products P it holds that

$$\forall \mathsf{C}', \mathsf{C}'' \in \mathbf{ch}(\mathsf{P}), \mathsf{C}' \neq \mathsf{C}'' \colon \mathbf{sc}(\mathsf{C}') \cap \mathbf{sc}(\mathsf{C}'') = \varnothing. \tag{4}$$

Furthermore, a sum node S is called *selective* [20] if for all choices of sum-weights $\boldsymbol{w}$ and all possible inputs $\mathbf{x}$ it holds that at most one child of S is non-zero. An SPN $\mathcal{S}$ is called selective if all its sum nodes are selective.

As shown in [17], [19], integrating $\mathcal{S}(\mathbf{x})$ over arbitrary sets $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$, i.e. *marginalization* over $\boldsymbol{\mathcal{X}}$, reduces to the corresponding integrals at the input distributions and evaluating sums and products in the usual way. This property is known as *validity* of the SPNs [1], and key for efficient inference. In this paper we only consider complete and decomposable SPNs. Without loss of generality [17], [21], we assume *locally normalized* sum-weights, i.e. for each sum node S we have $\sum_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}} = 1$, and thus $p_{\mathcal{S}} \equiv \mathcal{S}$, i.e. the SPN's normalization constant is 1.

For RVs with finitely many states, we will use so-called *indicator variables* (IVs) as input distributions [1]. For a finite-state RV X and state $x \in \mathbf{val}(X)$, we introduce the IV $\lambda_{X=x}(x') := \mathbb{1}(x = x')$, assigning all probability mass to x. A complete and decomposable SPN represents the *(extended) network polynomial* of $p_{\mathcal{S}}$, which can be used in the *differential approach to inference* [1], [16], [17]. Assume any evidence $\mathbf{e}$ which is evaluated in the SPN. The derivatives of the SPN function with respect to the IVs (by interpreting the IVs as real-valued variables, see [16], [17] for details) yield

$$\frac{\partial \mathcal{S}(\mathbf{e})}{\partial \lambda_{X=x}} = \mathcal{S}(X = x, \mathbf{e} \backslash X), \tag{5}$$

representing the inference scenario of *modified evidence*, i.e. evidence $\mathbf{e}$ is modified such that X is set to x. The computationally attractive feature of the differential approach is that (5) can be evaluated for *all* $X \in \mathbf{X}$ and *all* $x \in \mathbf{val}(X)$ simultaneously using a *single* back-propagation pass in the SPN, after evidence has been evaluated. Similarly, for the second (and higher) derivatives, we get

$$\frac{\partial^2 \mathcal{S}(\mathbf{e})}{\partial \lambda_{X=x} \lambda_{Y=y}} = \begin{cases} \mathcal{S}(X = x, Y = y, \mathbf{e} \backslash \{X, Y\}) & \text{if } X \neq Y \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Furthermore, the differential approach can be generalized to SPNs with arbitrary input distributions, i.e. SPNs over RVs with countably infinite or uncountably many states (cf. [17] for details).

1.2 Related Work

SPNs are related to negation normal forms (NNFs), a potential deep network representation of propositional theories [22], [23], [24]. Like in SPNs, structural constraints in NNFs enable certain polynomial-time queries in the represented theory. In particular, the notions of smoothness, decomposability and determinism in NNFs translate to the notions of completeness, decomposability and selectivity in SPNs, respectively. The work on NNFs led to the concept of network polynomials as a multilinear representation of BNs over finitely many states [16], [25]. BNs were cast into an intermediate d-DNNF (deterministic decomposable NNF) representation in order to generate an arithmetic circuit (ACs), representing the BNs network polynomial. ACs, when restricted to sums and products, are equivalent to SPNs but have a slightly different syntax. In [26], ACs were learned by optimizing an objective trading off the log-likelihood on the training set and the inference cost of the AC, measured as the worst-case number of arithmetic operations required for inference (i.e. the number of edges in the AC). The learned models still represent BNs with context-specific independencies [27]. A similar approach learning Markov networks represented by ACs is followed in [28]. SPNs were the first time proposed in [1], where the represented distribution was not defined via a background graphical model any more, but directly as the normalized output of the network. In this work, SPNs were applied to image data, where a generic architecture reminiscent to convolutional neural networks was proposed. Structure learning algorithms not restricted to the image domain were proposed in [2], [3], [4], [29], [30], [31]. Discriminative learning of SPNs, optimizing conditional likelihood, was proposed in [6]. Furthermore, there is a growing body of literature on theoretical aspects of SPNs and their relationship to other types of probabilistic models. In [32] two families of functions were identified which are efficiently representable by deep, but not by shallow SPNs, where an SPN is considered as shallow if it has no more than three layers. In [17] it was shown that SPNs can w.l.o.g. be assumed to be locally normalized and that the notion of consistency does not allow exponentially more compact models than decomposability. These results were independently found in [21]. Furthermore, in [17], a sound derivation of inference mechanisms for generalized SPNs was given, i.e. SPNs over RVs with (uncountably) infinitely many states. In [21], a BN representation of SPNs was found, where LVs associated with sum nodes and the model RVs are organized in a two layer bipartite structure. The actual SPN structure is captured in structured conditional probability tables (CPTs) using algebraic decision diagrams. Recently, the notion of SPNs was generalized to sum-product functions over arbitrary semirings [33]. This yields a general unifying framework for learning and inference, subsuming, among others, SPNs for probabilistic modeling, NNFs for logical propositions and function representations for integration and optimization.

2 Latent Variable Interpretation

As pointed out in [1], each sum node in an SPN can be interpreted as a marginalized LV, similar as in the GMM example in Section 1. For each sum node S, one postulates a discrete LV Z whose states correspond to the children of S. For each state, an IV and a product is introduced, such that the children are switched on/off by the corresponding IVs, as illustrated in Fig. 1.[1] When all IVs in Fig. 1b are set to 1, S still computes the same value as in Fig. 1a. Since setting all IVs of Z to 1 corresponds to marginalizing Z, the sum S should be interpreted as a latent, marginalized RV.

However, when we regard a larger structural context in Fig. 1b, we recognize that this justification is actually too simplistic. Explicitly introducing the IVs renders the ancestor S' incomplete, when S is no descendant of N, and Z is thus not in the scope of N. Note that setting all IVs to 1 in an *incomplete* SPN generally does *not* correspond to

1. In graphical representations of SPNs, IVs are depicted as nodes containing a small circle, general distributions as nodes containing a Gaussian-like PDF, and sum and products as nodes with $+$ and $\times$ symbols. Empty nodes are of arbitrary type.

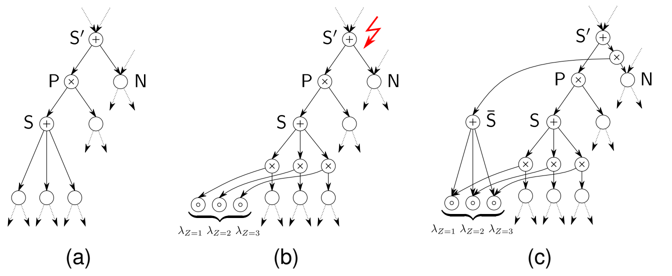


Fig. 1. Problems occurring when IVs of LVs are introduced. (a): Excerpt of SPN containing a sum S, corresponding to LV Z. (b): Introducing IVs for Z renders S$'$ incomplete, assuming that S $\notin$ $\mathbf{desc}$(N). (c): Remedy by extending SPN further, introducing twin sum node $\bar{\mathsf{S}}$.

marginalization. Furthermore, note that also S$'$ corresponds to an LV, say Z'. While we know the probability distribution of Z if Z' is in the state corresponding to P, namely the weights of S, we do not know this distribution when Z' is in the state corresponding to N. Intuitively, we recognize that the state of Z is irrelevant in this case, since it does not influence the resulting distribution over the model RVs $\mathbf{X}$. Nevertheless, the probabilistic model is not completely specified, which is unsatisfying.

A remedy for these problems is shown in Fig. 1c. We introduce the twin sum node $\bar{\mathsf{S}}$ whose children are the IVs corresponding to Z. The twin $\bar{\mathsf{S}}$ is connected as child of an additional product node, which is interconnected between S$'$ and N. Since this new product node has scope $\mathbf{sc}(\mathsf{N}) \cup \{Z\}$, S$'$ is rendered complete now. Furthermore, if Z' takes the state corresponding to N (or actually the state corresponding to the new product node), we now have a specified conditional distribution for Z, namely the weights of the twin sum node. Clearly, given that all IVs of Z are set to 1, the network depicted in Fig. 1c still computes the same function as the network in Fig. 1a (or Fig. 1b), since $\bar{\mathsf{S}}$ constantly outputs 1, as long as we use normalized weights for it. Which weights should be used for the twin sum node $\bar{\mathsf{S}}$? Basically, we can assume arbitrary normalized weights, which will cause $\bar{\mathsf{S}}$ to constantly output 1, where, however, a natural choice would be to use uniform weights for $\bar{\mathsf{S}}$ (maximizing the entropy of the resulting LV model). Although the choice of weights is not crucial for *evaluating evidence* in the SPN, it plays a role in *MPE inference*, see Section 4. For now, let us formalize the explicit introduction of LVs, denoted as *augmentation*.

2.1 Augmentation of SPNs

Let $\mathcal{S}$ be an SPN over $\mathbf{X}$. For each $\mathsf{S} \in \mathbf{S}(\mathcal{S})$ we assume an arbitrary but fixed ordering of its children $\mathbf{ch}(\mathsf{S}) = \{\mathsf{C}^1_\mathsf{S}, \dots, \mathsf{C}^{K_\mathsf{S}}_\mathsf{S}\}$, where $K_\mathsf{S} = |\mathbf{ch}(\mathsf{S})|$. Let Z_S be an RV on the same probability space as $\mathbf{X}$, with $\mathbf{val}(Z_\mathsf{S}) = \{1, \dots, K_\mathsf{S}\}$, where state k corresponds to child C^k_S. We call Z_S the *LV associated with* S. For sets of sum nodes $\mathbf{S}$ we define $\mathbf{Z}_\mathbf{S} = \{Z_\mathsf{S} \,|\, \mathsf{S} \in \mathbf{S}\}$. To distinguish $\mathbf{X}$ from the LVs, we will refer to the former as *model RVs*. For node N, we define the *sum ancestors/descendants* as

$$\mathbf{anc}_\mathbf{S}(\mathsf{N}) := \mathbf{anc}(\mathsf{N}) \cap \mathbf{S}(\mathcal{S}), \quad (7)$$

$$\mathbf{desc}_\mathbf{S}(\mathsf{N}) := \mathbf{desc}(\mathsf{N}) \cap \mathbf{S}(\mathcal{S}). \quad (8)$$

```
1:  procedure AUGMENTSPN($\mathcal{S}$)
2:      $\mathcal{S}' \leftarrow \mathcal{S}$
3:      $\forall \mathsf{S} \in \mathbf{S}(\mathcal{S}'),\ \forall k \in \{1,\dots,K_\mathsf{S}\}$:
            let $w_{\mathsf{S},k} = w_{\mathsf{S},\mathsf{C}^k_\mathsf{S}}$, $\bar{w}_{\mathsf{S},k} = \bar{w}_{\mathsf{S},\mathsf{C}^k_\mathsf{S}}$
4:      for $\mathsf{S} \in \mathbf{S}(\mathcal{S}')$ do
5:          for $k = 1 \dots K_\mathsf{S}$ do
6:              Introduce a new product node $\mathsf{P}^k_\mathsf{S}$ in $\mathbf{S}(\mathcal{S}')$
7:              Disconnect $\mathsf{C}^k_\mathsf{S}$ from S
8:              Connect $\mathsf{C}^k_\mathsf{S}$ as child of $\mathsf{P}^k_\mathsf{S}$
9:              Connect $\mathsf{P}^k_\mathsf{S}$ as child of S with weight $w_{\mathsf{S},k}$
10:         end for
11:     end for
12:     for $\mathsf{S} \in \mathbf{S}(\mathcal{S}')$ do
13:         for $k \in \{1,\dots,K_\mathsf{S}\}$ do
14:             Connect new IV $\lambda_{Z_\mathsf{S}=k}$ as child of $\mathsf{P}^k_\mathsf{S}$
15:         end for
16:         if $\mathbf{S}^c(\mathsf{S}) \neq \emptyset$ then
17:             Introduce a twin sum node $\bar{\mathsf{S}}$ in $\mathcal{S}'$
18:             $\forall k \in \{1,\dots,K_\mathsf{S}\}$: connect $\lambda_{Z_\mathsf{S}=k}$ as child of $\bar{\mathsf{S}}$,
                    and let $w_{\bar{\mathsf{S}},\lambda_{Z_\mathsf{S}=k}} = \bar{w}_{\mathsf{S},k}$
19:             for $\mathsf{S}^c \in \mathbf{S}^c(\mathsf{S})$ do
20:                 for $k \in \{k \mid \mathsf{S} \notin \mathbf{desc}(\mathsf{P}^k_{\mathsf{S}^c})\}$ do
21:                     Connect $\bar{\mathsf{S}}$ as child of $\mathsf{P}^k_{\mathsf{S}^c}$
22:                 end for
23:             end for
24:         end if
25:     end for
26:     return $\mathcal{S}'$
27: end procedure
```

Fig. 2. Pseudo-code for augmentation of an SPN.

For each sum node S we define the *conditioning sums* as

$$\mathbf{S}^c(\mathsf{S}) := \{\mathsf{S}^c \in \mathbf{anc}_\mathbf{S}(\mathsf{S}) \setminus \{\mathsf{S}\} \,|\, \exists \mathsf{C} \in \mathbf{ch}(\mathsf{S}^c) \colon \mathsf{S} \notin \mathbf{desc}(\mathsf{C})\}. \quad (9)$$

Furthermore, we assume a set of locally normalized *twin-weights* $\bar{\boldsymbol{w}}$, containing a twin-weight $\bar{w}_{\mathsf{S},\mathsf{C}}$ for each weight $w_{\mathsf{S},\mathsf{C}}$ in the SPN. We are now ready to define the *augmentation* of an SPN.

Definition 2 (Augmentation of SPN)**.** *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$, $\bar{\boldsymbol{w}}$ be a set of twin-weights and $\mathcal{S}'$ be the result of algorithm* AUGMENTSPN*, shown in Fig. 2. $\mathcal{S}'$ is called the* augmented *SPN of $\mathcal{S}$, denoted as $\mathcal{S}' =: \mathbf{aug}(\mathcal{S})$. Within the context of $\mathcal{S}'$, C^k_S is called the kth* former child *of* S*. The introduced product node P^k_S is called* link *of* S*, C^k_S and $\lambda_{Z_\mathsf{S}=k}$, respectively. The sum node $\bar{\mathsf{S}}$, if introduced, is called the* twin *sum node of* S*. With respect to $\mathcal{S}'$, we denote $\mathcal{S}$ as the* original *SPN.*

In steps 4–11 of AUGMENTSPN we introduce the links P^k_S which are interconnected between sum node S and its kth child. Each link P^k_S has a single parent, namely S, and simply copies the former child C^k_S. In steps 13–15, we introduce IVs corresponding to the associated LV Z_S, as proposed in [1]. As we saw in Fig. 1 and the discussion above, this can render other sum nodes incomplete. These sums are clearly the conditioning sums $\mathbf{S}^c(\mathsf{S})$. Thus, when necessary, we introduce a twin sum node in steps 17–23, to treat this problem. The following proposition states the soundness of augmentation.

Proposition 1. *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$, $\mathcal{S}' = \mathbf{aug}(\mathcal{S})$ and $\mathbf{Z} := \mathbf{Z}_{\mathsf{S}(\mathcal{S})}$. Then $\mathcal{S}'$ is a complete and decomposable SPN over $\mathbf{X} \cup \mathbf{Z}$ with $\mathcal{S}'(\mathbf{X}) \equiv \mathcal{S}(\mathbf{X})$.*

Proposition 1 states that the marginal distribution over $\mathbf{X}$ in the augmented SPN is the same distribution as represented by the original SPN, while being a *completely specified probabilistic model* over $\mathbf{X}$ *and* $\mathbf{Z}$. Thus, augmentation provides a sound way to generalize the LV interpretation from mixture models to more general SPNs. An example of augmentation is shown in Fig. 3.

Note that we understand the augmentation mainly as a *theoretical* tool to establish and work with the LV interpretation in SPNs. In most cases, it will be neither necessary nor advisable to *explicitly* construct the augmented SPN.

An interesting question is how the sizes of the original SPN and the augmented SPN relate to each other. A lower bound is $|\mathcal{S}'| \in \Omega(|\mathcal{S}|)$, holding e.g. for SPNs with a single sum node. An asymptotic upper bound is $|\mathcal{S}'| \in \mathcal{O}(|\mathcal{S}|^2)$. To see this, note that the introduction of links, IVs and twin sums cause at most a linear increase of the SPN's size. The number of edges introduced when connecting twins to the links of conditioning sums is bounded by $|\mathcal{S}|^2$, since the number of twins and links are both bounded by $|\mathcal{S}|$. Therefore, we have $|\mathcal{S}'| \in \mathcal{O}(|\mathcal{S}|^2)$. This asymptotic upper bound is indeed achieved by certain types of SPNs: Consider e.g. a chain consisting of K sum nodes and $K+1$ distribution nodes. For $k < K$ the k^{th} sum is the parent of the $(k+1)^{\text{th}}$ sum and the k^{th} distribution, and the K^{th} sum is the parent of the last two distributions. For the k^{th} sum, all preceding sums are conditioning sums, yielding $k-1$ introduced edges. In total this gives $\sum_{k=2}^{K}(k-1) = \frac{K\,(K-1)}{2} = \frac{K^2-K}{2}$ edges, i.e. in this case $|\mathcal{S}'|$ indeed grows quadratically in $|\mathcal{S}|$.

2.2 Conditional Independencies in Augmented SPNs and Probabilistic Interpretation of Sum-Weights

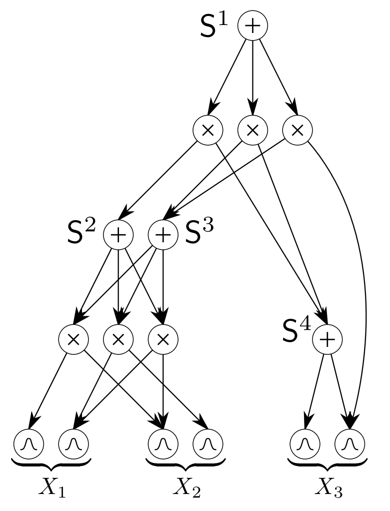
It is helpful to introduce the notion of configured SPNs, which takes a similar role as conditioning in the literature on DNNFs [22], [23], [24].

Definition 3 (Configured SPN)**.** *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$, $\mathbf{Y} \subseteq \mathbf{Z}_{\mathsf{S}(\mathcal{S})}$ and $\mathbf{y} \in \mathbf{val}(\mathbf{Y})$. The* configured SPN *$\mathcal{S}^{\mathbf{y}}$ is obtained by deleting the IVs $\lambda_{Y=y}$ and their corresponding link for each $Y \in \mathbf{Y}$, $y \neq \mathbf{y}[Y]$ from $\mathbf{aug}(\mathcal{S})$, and further deleting all nodes which are rendered unreachable from the root.*

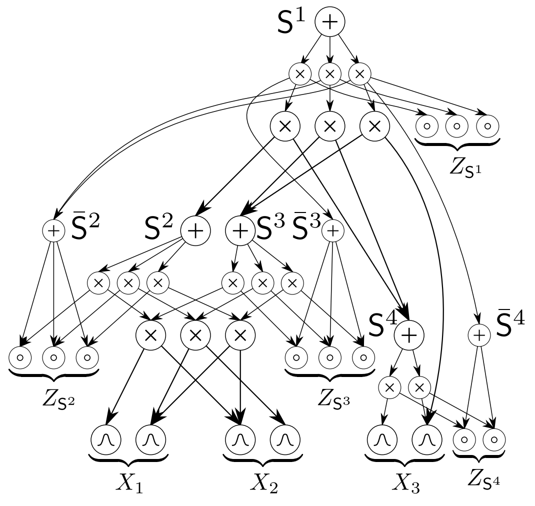
Intuitively, the configured SPN isolates the computational structure selected by $\mathbf{y}$. All sum edges which "survive" in the configured SPN are equipped with the same weights as in the augmented SPN. Therefore, a configured SPN is in general not locally normalized. We note the following properties of configured SPNs.

Proposition 2. *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$, $\mathbf{Y} \subseteq \mathbf{Z}_{\mathsf{S}(\mathcal{S})}$ and $\mathbf{Z} = \mathbf{Z}_{\mathsf{S}(\mathcal{S})} \setminus \mathbf{Y}$. Let $\mathbf{y} \in \mathbf{val}(\mathbf{Y})$ and let $\mathcal{S}' = \mathbf{aug}(\mathcal{S})$. It holds that*

1) *Each node in $\mathcal{S}^{\mathbf{y}}$ has the same scope as its corresponding node in $\mathcal{S}'$.*
2) *$\mathcal{S}^{\mathbf{y}}$ is a complete and decomposable SPN over $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$.*
3) *For any node N in $\mathcal{S}^{\mathbf{y}}$ with $\mathbf{sc}(\mathsf{N}) \cap \mathbf{Y} = \emptyset$, we have that $\mathcal{S}^{\mathbf{y}}_{\mathsf{N}} = \mathcal{S}'_{\mathsf{N}}$.*



(a)



(b)

Fig. 3. Augmentation of an SPN. (a): Example SPN over $\mathbf{X} = \{X_1, X_2, X_3\}$, containing sum nodes S^1, S^2, S^3 and S^4. (b): Augmented SPN, containing IVs corresponding to Z_{S^1}, Z_{S^2}, Z_{S^3}, Z_{S^4}, links and twin sum nodes $\bar{\mathsf{S}}^2$, $\bar{\mathsf{S}}^3$, $\bar{\mathsf{S}}^4$. For nodes introduced by augmentation, smaller circles are used.

4) *For $\mathbf{y}' \in \mathbf{val}(\mathbf{Y})$ it holds that*

$$\mathcal{S}^{\mathbf{y}}(\mathbf{X}, \mathbf{Z}, \mathbf{y}') = \begin{cases} \mathcal{S}'(\mathbf{X}, \mathbf{Z}, \mathbf{y}') & \text{if } \mathbf{y}' = \mathbf{y} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The next theorem shows certain conditional independencies in the augmented SPN. For ease of discussion, we make the following definitions.

Definition 4. *Let S be a sum node in an SPN and Z_{S} its associated LV. All other RVs (model RVs and LVs) are divided into three sets:*

- Parents $\mathbf{Z}_p$*, which are all LVs "above" S, i.e. $\mathbf{Z}_p = \mathbf{Z}_{\mathbf{ancs}_{\mathsf{S}}(\mathsf{S})} \setminus Z_{\mathsf{S}}$.*
- Children $\mathbf{Y}_c$*, which are all model RVs and LVs "below" S, i.e. $\mathbf{Y}_c = \mathbf{sc}(\mathsf{S}) \cup \mathbf{Z}_{\mathbf{descs}_{\mathsf{S}}(\mathsf{S})} \setminus Z_{\mathsf{S}}$.*
- Non-descendants $\mathbf{Y}_n$*, which are the remaining RVs, i.e. $\mathbf{Y}_n = (\mathbf{X} \cup \mathbf{Z}_{\mathsf{S}(\mathcal{S})}) \setminus (\mathbf{Z}_p \cup \mathbf{Y}_c \cup Z_{\mathsf{S}})$.*

We will show that the *parents*, *children* and *non-descendants* play the likewise role as for independencies in

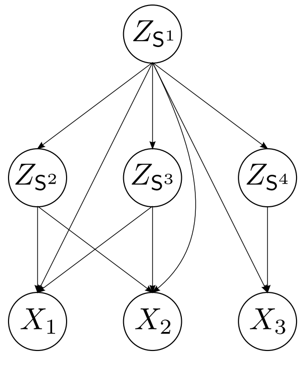


Fig. 4. Dependency structure of augmented SPN from Fig. 3, represented as BN.

BNs [14], [15], i.e. Z_{S} *is independent of* $\mathbf{Y}_n$ *given* $\mathbf{Z}_p$. We will further show that the sum-weights of S are the conditional distribution of Z_{S}, conditioned on the event that "$\mathbf{Z}_p$ select a path to S". One problem in the original LV interpretation [1] was, that no conditional distribution of Z_{S} was specified for the complementary event. Here, we will show that the twin-weights are precisely this conditional distribution. This requires that the event "$\mathbf{Z}_p$ select a path to the twin $\bar{\mathsf{S}}$" is indeed the complementary event to "$\mathbf{Z}_p$ select a path to S". This is shown in following lemma.

Lemma 1. *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$, let* S *be a sum node in $\mathcal{S}$ and $\mathbf{Z}_p$ be the parents of Z_{S}. For any $\mathbf{z} \in \mathbf{val}(\mathbf{Z}_p)$, the configured SPN $\mathcal{S}^{\mathbf{z}}$ contains either* S *or its twin* $\bar{\mathsf{S}}$, *but not both.*

We are now ready to state the our theorem concerning conditional independencies in augmented SPNs.

Theorem 1. *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$ and $\mathcal{S}' = \mathbf{aug}(\mathcal{S})$. Let* S *be an arbitrary sum in $\mathcal{S}$ and $w_k = w_{\mathsf{S},\mathsf{C}^k_{\mathsf{S}}}$, $\bar{w}_k = \bar{w}_{\mathsf{S},\mathsf{C}^k_{\mathsf{S}}}$, $k = 1, \ldots, K_{\mathsf{S}}$. With respect to* S, *let $\mathbf{Z}_p$ be the parents, $\mathbf{Y}_c$ be the children and $\mathbf{Y}_n$ be the non-descendants, respectively. Then there exists a two-partition of $\mathbf{val}(\mathbf{Z}_p)$, i.e. $\mathcal{Z}, \bar{\mathcal{Z}} \colon \mathcal{Z} \cup \bar{\mathcal{Z}} = \mathbf{val}(\mathbf{Z}_p)$, $\mathcal{Z} \cap \bar{\mathcal{Z}} = \emptyset$, such that*

$$\forall \mathbf{z} \in \mathcal{Z} \colon \mathcal{S}'(Z_{\mathsf{S}} = k, \mathbf{Y}_n, \mathbf{z}) = w_k \mathcal{S}'(\mathbf{Y}_n, \mathbf{z}), \textit{and} \tag{11}$$

$$\forall \mathbf{z} \in \bar{\mathcal{Z}} \colon \mathcal{S}'(Z_{\mathsf{S}} = k, \mathbf{Y}_n, \mathbf{z}) = \bar{w}_k \mathcal{S}'(\mathbf{Y}_n, \mathbf{z}). \tag{12}$$

From Theorem 1 it follows that the weights and twin-weights of a sum node S can be interpreted as *conditional probability tables* (CPTs) of Z_{S}, conditioned on $\mathbf{Z}_p$ and that Z_{S} is conditionally independent of $\mathbf{Y}_n$ given $\mathbf{Z}_p$, i.e.

$$\mathcal{S}'(Z_{\mathsf{S}} = k \,|\, \mathbf{Y}_n, \mathbf{z}) = \mathcal{S}'(Z_{\mathsf{S}} = k \,|\, \mathbf{z}) = \begin{cases} w_k & \text{if } \mathbf{z} \in \mathcal{Z} \\ \bar{w}_k & \text{if } \mathbf{z} \in \bar{\mathcal{Z}}. \end{cases} \tag{13}$$

Using this result, we can define a BN representing the augmented SPN as follows: For each sum node S, connect $\mathbf{Z}_p$ as parents of Z_{S}, and all RVs $\mathbf{sc}(\mathsf{S})$ as children of Z_{S}. By doing this for each LV, we obtain our BN representation of the augmented SPN, serving as a useful tool to understand SPNs in the context of probabilistic graphical models. An example of the BN interpretation is shown in Fig. 4.

Note that the BN representation by Zhao et al. [21] can be recovered from the BN representation of augmented SPNs. They proposed a BN representation of SPNs using a bipartite structure, where an LV is a parent of a model RV if it is contained in the scope of the corresponding sum node. The model RVs and LVs are unconnected among each other, respectively. When we constrain the twin-weights to be equal to the sum-weights, we can see in (13) that Z_{S} becomes independent of $\mathbf{Z}_p$. This special choice of twin weights effectively removes all edges between LVs, recovering the BN structure in [21]. In the next section, we use the augmented SPN and the BN interpretation to derive the EM algorithm for SPNs.

3 EM Algorithm

The EM algorithm is a general scheme for maximum likelihood learning, when for some RVs complete evidence is missing [10], [11]. Thus, augmented SPNs are amenable for EM due to the LVs associated with sum nodes. Moreover, the twin-weights can be kept fixed, so that EM applied to augmented SPNs actually optimizes the weights of the original SPN. This approach was already pointed out in [1], where it was suggested that for evidence $\mathbf{e}$ and for any LV Z_{S}, the marginal posteriors should be given as $p(Z_{\mathsf{S}} = k \,|\, \mathbf{e}) \propto w_{\mathsf{S},\mathsf{C}^k_{\mathsf{S}}} \frac{\partial \mathcal{S}(\mathbf{e})}{\partial \mathsf{S}(\mathbf{e})}$, which should be used for EM updates. These updates, however, cannot be the correct ones, as they actually *leave the weights unchanged*. Here, using augmented SPNs, we formally derive the standard EM updates for sum-weights and the input distributions, when they are chosen from an exponential family.

3.1 Updates for Weights

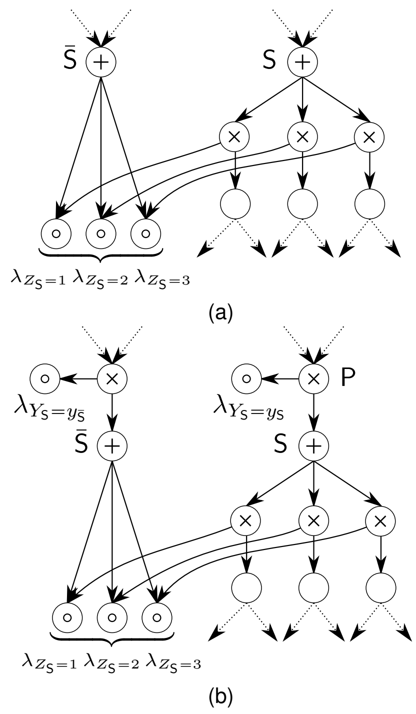
Assume a dataset $\mathcal{D} = \{\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(L)}\}$ of L i.i.d. samples, where each $\mathbf{e}^{(l)}$ is any combination of complete and partial evidence for the model RVs $\mathbf{X}$, cf. Section 1.1. Let $\mathbf{Z} = \mathbf{Z}_{\mathbf{S}(\mathcal{S})}$ be the set of all LVs and consider an arbitrary sum node S. Eq. (13) shows that the weights can be interpreted as conditional probabilities in our BN interpretation, where

$$\mathcal{S}'(Z_{\mathsf{S}} = k \,|\, \mathbf{Z}_p = \mathbf{z}) = \begin{cases} w_k & \text{if } \mathbf{z} \in \mathcal{Z} \\ \bar{w}_k & \text{if } \mathbf{z} \in \bar{\mathcal{Z}}. \end{cases} \tag{14}$$

As mentioned above, the twin-weights $\bar{w}_k$ are kept fixed. Using the well-known EM-updates in BNs over discrete RVs [10], [15], the updates for sum-weight w_k are given by summing over the expected statistics

$$\mathcal{S}'(Z_{\mathsf{S}} = k, \mathbf{Z}_p \in \mathcal{Z} \,|\, \mathbf{e}^{(l)}), \tag{15}$$

followed by renormalization. We make the event $\mathbf{Z}_p \in \mathcal{Z}$ explicit, by introducing a *switching parent* Y_{S} of Z_{S}: When the twin sum of S exists, Y_{S} assumes the two states $\mathbf{val}(Y_{\mathsf{S}}) = \{y_{\mathsf{S}}, y_{\bar{\mathsf{S}}}\}$, where $Y_{\mathsf{S}} = y_{\mathsf{S}} \Leftrightarrow \mathbf{Z}_p \in \mathcal{Z}$ and $Y_{\mathsf{S}} = y_{\bar{\mathsf{S}}} \Leftrightarrow \mathbf{Z}_p \in \bar{\mathcal{Z}}$. When the twin sum does not exist, Y_{S} just takes the single value $\mathbf{val}(Y_{\mathsf{S}}) = \{y_{\mathsf{S}}\}$. Clearly, when observed, Y_{S} renders Z_{S} independent from $\mathbf{Z}_p$. The switching parent can be explicitly introduced in the augmented SPN, as depicted in Fig. 5. Here we simply introduce two new IVs $\lambda_{Y_{\mathsf{S}}=y_{\mathsf{S}}}$ and $\lambda_{Y_{\mathsf{S}}=y_{\bar{\mathsf{S}}}}$, which switch on/off the output of S and $\bar{\mathsf{S}}$, respectively. It is easy to see that when these IV are constantly set to 1, i.e. when Y_{S} is marginalized, the augmented SPN performs exactly the same computations as before. It is furthermore easy to see that completeness and decomposability of the augmented SPN are maintained



$\bar{\mathsf{S}}$
S
$\lambda_{Z_\mathsf{S}=1}$ $\lambda_{Z_\mathsf{S}=2}$ $\lambda_{Z_\mathsf{S}=3}$
(a)
$\lambda_{Y_\mathsf{S}=y_{\bar{\mathsf{S}}}}$
$\lambda_{Y_\mathsf{S}=y_\mathsf{S}}$
P
(b)

Fig. 5. Explicitly introducing a switching parent Y_S in an augmented SPN. (a): Part of an augmented SPN containing a sum node with three children and its twin. (b): Explicitly introduced switching parent Y_S using IVs $\lambda_{Y_\mathsf{S}=y_\mathsf{S}}$ and $\lambda_{Y_\mathsf{S}=y_{\bar{\mathsf{S}}}}$.

when the switching parent is introduced. Using the switching parent, the required expected statistics (15) translate to

$$\mathcal{S}'(Z_\mathsf{S}=k, Y_\mathsf{S}=y_\mathsf{S} \,|\, \mathbf{e}^{(l)}). \tag{16}$$

To compute (16), we use the differential approach, [16], [17], [19], cf. also Section 1.1. First note that

$$\mathcal{S}'(Z_\mathsf{S}=k, Y_\mathsf{S}=y_\mathsf{S}, \mathbf{e}^{(l)}) = \frac{\partial^2 \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \lambda_{Y_\mathsf{S}=y_\mathsf{S}} \partial \lambda_{Z_\mathsf{S}=k}}. \tag{17}$$

The first derivative is given as

$$\frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \lambda_{Y_\mathsf{S}=y_\mathsf{S}}} = \frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \mathsf{P}} \mathsf{S}(\mathbf{e}^{(l)}) \tag{18}$$

$$= \frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \mathsf{P}} \sum_{k=1}^{K_\mathsf{S}} \lambda_{Z_\mathsf{S}=k}\, w_k\, \mathsf{C}_\mathsf{S}^k(\mathbf{e}^{(l)}), \tag{19}$$

where P is the common product parent of S and $\lambda_{Y_\mathsf{S}=y_\mathsf{S}}$ in the augmented SPN (see Fig. 5b). Differentiating (19) after $\lambda_{Z_\mathsf{S}=k}$ yields the second derivative

$$\frac{\partial^2 \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \lambda_{Y_\mathsf{S}=y_\mathsf{S}} \partial \lambda_{Z_\mathsf{S}=k}} = \frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \mathsf{P}}\, w_k\, \mathsf{C}_\mathsf{S}^k(\mathbf{e}^{(l)}), \tag{20}$$

delivering the required posteriors

$$\mathcal{S}'(Z_\mathsf{S}=k, Y_\mathsf{S}=y_\mathsf{S} \,|\, \mathbf{e}^{(l)}) = \frac{1}{\mathcal{S}'(\mathbf{e}^{(l)})} \frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \mathsf{P}}\, w_k\, \mathsf{C}_\mathsf{S}^k(\mathbf{e}^{(l)}). \tag{21}$$

We do not want to construct the augmented SPN explicitly, so we express (21) in terms of the original SPN. Since all LVs are marginalized, it holds that $\mathcal{S}'(\mathbf{e}^{(l)}) = \mathcal{S}(\mathbf{e}^{(l)})$ and $\frac{\partial \mathcal{S}'(\mathbf{e}^{(l)})}{\partial \mathsf{P}} = \frac{\partial \mathcal{S}(\mathbf{e}^{(l)})}{\partial \mathsf{S}}$, yielding

$$\mathcal{S}'(Z_\mathsf{S}=k, Y_\mathsf{S}=y_\mathsf{S} \,|\, \mathbf{e}^{(l)}) = \frac{1}{\mathcal{S}(\mathbf{e}^{(l)})} \frac{\partial \mathcal{S}(\mathbf{e}^{(l)})}{\partial \mathsf{S}}\, w_k\, \mathsf{C}_\mathsf{S}^k(\mathbf{e}^{(l)}), \tag{22}$$

delivering the required statistics for updating the sum-weights. We now turn to the updates of the input distributions.

3.2 Updates for Input Distributions

For simplicity, we derive updates for univariate input distributions, i.e. for all distributions $\mathsf{D}_\mathbf{Y}$ we have $|\mathbf{sc}(\mathsf{D}_\mathbf{Y})| = 1$. Similar updates can rather easily be derived also for multivariate input distributions. In [17], the so-called distribution selectors (DSs) were introduced to derive the differential approach for generalized SPNs. Similar as the switching parents for (twin) sum nodes, the DSs are RVs which render the respective model RVs independent from the remaining RVs. More formally, for each $X \in \mathbf{X}$, let $\mathbf{D}_X$ be the set of all input distributions which have scope $\{X\}$. Assume an arbitrary but fixed ordering of $\mathbf{D}_X$ and let $[\mathsf{D}_X]$ be the index of D_X in this ordering. Let the DS W_X be a discrete RV with $|\mathbf{D}_X|$ states. The so-called gated SPN $\mathcal{S}^g$ is obtained by replacing each distribution by the product node

$$\mathsf{D}_X \rightarrow \mathsf{D}_X \times \lambda_{W_X=[\mathsf{D}_X]}. \tag{23}$$

The introduced product is denoted as gate. As shown in [17], X is rendered independent from all other RVs in the SPN when conditioned on W_X. Moreover, D_X is the conditional distribution of X given $W_X = [\mathsf{D}_X]$. Therefore, each X and its DS W_X can be incorporated as a two RV family in our BN interpretation. When each input distribution D_X is chosen from an exponential family with natural parameters θ_{D_X}, the M-step is given by the expected sufficient statistics

$$\theta_{\mathsf{D}_X} \leftarrow \frac{\sum_l \mathcal{S}^g(W_X=k \,|\, \mathbf{e}^{(l)}) \int \mathsf{D}_X(x \,|\, \mathbf{e}^{(l)}) \theta_{\mathsf{D}_X}(x) \mathrm{d}x}{\sum_l \mathcal{S}^g(W_X=k \,|\, \mathbf{e}^{(l)})}, \tag{24}$$

where $k = [\mathsf{D}_X]$. When $\mathbf{e}^{(l)}$ contains complete evidence x' for X, then the integral $\int \mathsf{D}_X(x \,|\, \mathbf{e}^{(l)}) \theta_{\mathsf{D}_X}(x) \mathrm{d}x$ reduces to $\theta_{\mathsf{D}_X}(x')$. When $\mathbf{e}^{(l)}$ contains partial evidence $\mathcal{X}$, then

$$\int \mathsf{D}_X(x \,|\, \mathbf{e}^{(l)}) \theta_{\mathsf{D}_X}(x) \mathrm{d}x = \frac{\int_\mathcal{X} \mathsf{D}_X(x) \theta_{\mathsf{D}_X}(x) \mathrm{d}x}{\int_\mathcal{X} \mathsf{D}_X(x) \mathrm{d}x}. \tag{25}$$

Depending on X and the the type of D_X, evaluating (25) can be more or less demanding. A simple but practical case is when D_X is Gaussian and $\mathcal{X}$ is some interval, permitting a closed form solution for integrating the Gaussian's statistics $\theta(x) = (x, x^2)$, using truncated Gaussians [34].

To obtain the posteriors $\mathcal{S}^g(W_X = k \,|\, \mathbf{e}^{(l)})$ required in (24), we again use the differential approach. Note that

$$\mathcal{S}^g(W_X=k, \mathbf{e}^{(l)}) = \frac{\partial \mathcal{S}^g(\mathbf{e}^{(l)})}{\partial \lambda_{W_X=k}} = \frac{\partial \mathcal{S}^g(\mathbf{e}^{(l)})}{\partial \mathsf{P}} \mathsf{D}_X(\mathbf{e}^{(l)}), \tag{26}$$

where $k = [\mathsf{D}_X]$ and P is the gate of D_X, cf. (23). If we do not want to construct the gated SPN explicitly, we can use the identity $\frac{\partial \mathcal{S}^g(\mathbf{e}^{(l)})}{\partial \mathsf{P}} = \frac{\partial \mathcal{S}(\mathbf{e}^{(l)})}{\partial \mathsf{D}_X}$. Thus the required posteriors are given as

$$\mathcal{S}^g(W_X=k \,|\, \mathbf{e}^{(l)}) = \frac{1}{\mathcal{S}(\mathbf{e}^{(l)})} \frac{\partial \mathcal{S}(\mathbf{e}^{(l)})}{\partial \mathsf{D}_X} \mathsf{D}_X(\mathbf{e}^{(l)}). \tag{27}$$

```
1: procedure EXPECTATION-MAXIMIZATION($\mathcal{S}$)
2:     Initialize $\boldsymbol{w}$ and input distributions
3:     while not converged do
4:         $\forall \mathsf{S} \in \mathbf{S}(\mathcal{S}), \forall \mathsf{C} \in \mathbf{ch}(\mathsf{S})\colon n_{\mathsf{S},\mathsf{C}} \leftarrow 0$
5:         $\forall X \in \mathbf{X}, \forall \mathsf{D}_X \in \mathbf{D}_X\colon \theta_{\mathsf{D}_X} \leftarrow 0, n_{\mathsf{D}_X} \leftarrow 0$
6:         for $l = 1 \dots L$ do
7:             Input $\mathbf{e}^{(l)}$ to $\mathcal{S}$
8:             Evaluate $\mathcal{S}$ (upward-pass)
9:             Backprop $\mathcal{S}$ (backward-pass)
10:            for $\mathsf{S} \in \mathbf{S}(\mathcal{S}), \mathsf{C} \in \mathbf{ch}(\mathsf{S})$ do
11:                $n_{\mathsf{S},\mathsf{C}} \leftarrow n_{\mathsf{S},\mathsf{C}} + \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial \mathsf{S}} \mathsf{C}\, w_{\mathsf{S},\mathsf{C}}$
12:            end for
13:            for $X \in \mathbf{X}, \mathsf{D}_X \in \mathbf{D}_X$ do
14:                if $\mathbf{e}^{(l)}$ is complete w.r.t. $X$ then
15:                    $x \leftarrow$ complete evidence for $X$
16:                    $\theta \leftarrow \theta(x)$
17:                else
18:                    $\mathcal{X} \leftarrow$ partial evidence for $X$
19:                    $\theta \leftarrow \frac{\int_{\mathcal{X}} \mathsf{D}_X(x)\theta(x)\mathrm{d}x}{\int_{\mathcal{X}} \mathsf{D}_X(x)\mathrm{d}x}$
20:                end if
21:                $p \leftarrow \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial \mathsf{D}_X} \mathsf{D}_X$
22:                $\theta_{\mathsf{D}_X} \leftarrow \theta_{\mathsf{D}_X} + p\,\theta$
23:                $n_{\mathsf{D}_X} \leftarrow n_{\mathsf{D}_X} + p$
24:            end for
25:        end for
26:        $\forall \mathsf{S} \in \mathbf{S}(\mathcal{S}), \forall \mathsf{C} \in \mathbf{ch}(\mathsf{S})\colon\ w_{\mathsf{S},\mathsf{C}} \leftarrow \frac{n_{\mathsf{S},\mathsf{C}}}{\sum_{\mathsf{C}' \in \mathbf{ch}(\mathsf{S})} n_{\mathsf{S},\mathsf{C}'}}$
27:        $\forall X \in \mathbf{X}, \forall \mathsf{D}_X \in \mathbf{D}_X\colon$ set parameters to $\frac{\theta_{\mathsf{D}_X}}{n_{\mathsf{D}_X}}$
28:    end while
29:    return $\mathcal{S}$
30: end procedure
```

Fig. 6. Pseudo-code for EM algorithm in SPNs.

The EM algorithm for SPNs, both for sum-weights and input distributions, is summarized in Fig. 6. In Section 5.1 we empirically verify our derivation of EM and show that standard EM successfully trains SPNs when a suitable structure is at hand.

Note that recently Zhao and Poupart [35] derived a concave-convex procedure (CCCP) which yield the same sum-weight updates as the EM algorithm presented here and in [19]. This result is surprising, as EM and CCCP are rather different approaches in general.

4 Most Probable Explanation

In [1], [4], [7], SPNs were applied for reconstructing data using MPE inference. Given some distribution p over $\mathbf{X}$ and evidence $\mathbf{e}$, MPE can be formalized as finding $\arg\max_{\mathbf{x} \in \mathbf{e}} p(\mathbf{x})$, where we assume that p actually has a maximum in $\mathbf{e}$. MPE is a special case of MAP, defined as finding $\arg\max_{\mathbf{y} \in \mathbf{e}[\mathbf{Y}]} \int_{\mathbf{e}[\mathbf{Z}]} p(\mathbf{y}, \mathbf{z})\, \mathrm{d}\mathbf{z}$, for some two-partition of $\mathbf{X}$, i.e. $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z}, \mathbf{Y} \cap \mathbf{Z} = \emptyset$. Both MPE and MAP are generally NP-hard in BNs [36], [37], [38], and MAP is inherently harder than MPE [37], [38]. Using the result in [18], it follows that MAP inference is NP-hard also in SPNs. In particular, Theorem 5 in [18] shows that the decision version of MAP is NP-complete for a Naive Bayes model, when the class variable is marginalized. Naive Bayes is represented by the augmentation of an SPN with a single sum node, the LV representing the class variable. Therefore, MAP in SPNs is generally NP-hard. Since MAP in the augmented SPN representing the Naive Bayes model corresponds to MPE inference in the original SPN, i.e. a mixture model, it follows that also MPE inference is generally NP-hard in SPNs. A proof tailored to SPNs can be found in [19].

However, when considering the the sub-class of *selective* SPNs (cf. Section 1.1 and [20]), an MPE solution can be obtained using a Viterbi-style backtracking algorithm in *max-product networks*.

Definition 5 (Max-Product Network)**.** *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$. We define the* max-product network *(MPN) $\hat{\mathcal{S}}$, by replacing each distribution node* D *by a* maximizing *distribution node*

$$\hat{\mathsf{D}}\colon \mathcal{H}_{\mathbf{sc}(\mathsf{D})} \mapsto [0, \infty], \hat{\mathsf{D}}(\boldsymbol{\mathcal{Y}}) := \max_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \mathsf{D}(\mathbf{y}), \tag{28}$$

and each sum node S *by a* max node

$$\hat{\mathsf{S}} := \max_{\hat{\mathsf{C}} \in \mathbf{ch}(\hat{\mathsf{S}})} w_{\hat{\mathsf{S}},\hat{\mathsf{C}}} \hat{\mathsf{C}}. \tag{29}$$

A product node P in $\mathcal{S}$ corresponds to a product node $\hat{\mathsf{P}}$ in $\hat{\mathcal{S}}$.

Theorem 2. *Let $\mathcal{S}$ be a selective SPN over $\mathbf{X}$ and let $\hat{\mathcal{S}}$ the corresponding MPN. Let N be some node in $\mathcal{S}$ and $\hat{\mathsf{N}}$ its corresponding node in $\hat{\mathcal{S}}$. Then, for every $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{sc}(\mathsf{N})}$ we have $\hat{\mathsf{N}}(\boldsymbol{\mathcal{X}}) = \max_{\mathbf{x} \in \boldsymbol{\mathcal{X}}} \mathsf{N}(\mathbf{x})$.*

Theorem 2 shows that the MPN maximizes the probability in its corresponding selective SPN. The proof (see appendix) also shows how to actually *find* a maximizing assignment. For a product, a maximizing assignment is given by combining the maximizing assignments of its children. For a sum, a maximizing assignment is given by the maximizing assignment of a single child, whose weighted maximum is maximal among all children. Here the children's maxima are readily given by the upwards pass in the MPN. Thus, finding a maximizing assignment of any node in an selective SPN recursively reduces to finding maximizing assignments for the children of this node; this can be accomplished by a Viterbi-like backtracking procedure. This algorithm, denoted as MPESelective, is shown in Fig. 7. Here Q denotes a queue of nodes, where $Q \leftarrowtail \mathsf{N}$ and $\mathsf{N} \leftarrowtail Q$ denote the en-queue and de-queue operations, respectively. Note that Theorem 2 has already been derived for a special case, namely for arithmetic circuits representing network polynomials of BNs over discrete RVs [39].

A direct corollary of Theorem 2 is that MPE inference is tractable in augmented SPNs, since augmented SPNs are *selective SPNs over* $\mathbf{X}$ *and* $\mathbf{Z}$. This can easily be seen in AugmentSPN, as for any $\mathbf{z}$ and any sum S, exactly one IV of Z_{S} is set to 1, causing that at most one child of S (or $\bar{\mathsf{S}}$) can be non-zero. Therefore, we can use MPESelective in augmented SPNs, in order to find an MPE solution over *both* model RVs and LVs. Note that an MPE solution for the augmented SPN does in general *not* correspond to an MPE solution for the original SPN, when discarding the states of the LVs. However, this procedure is a frequently used approximation for models where MPE is tractable for both model RVs and LVs, but not for model RVs alone.

In [1], MPESelective was applied to *original* SPNs, not to *augmented* SPNs, but also with the goal to recover an

```
1: procedure MPESelective($\mathcal{S}$, $\mathbf{e}$)
2:     Initialize zero-vector $\mathbf{x}^*$ of length $|\mathbf{X}|$
3:     Evaluate $\mathbf{e}$ in corresponding MPN $\hat{\mathcal{S}}$ (upwards pass)
4:     $Q \leftharpoondown$ root node of MPN
5:     while $Q$ not empty do
6:         $\hat{\mathsf{N}} \leftharpoondown Q$
7:         if $\hat{\mathsf{N}}$ is a max node then
8:             $Q \leftharpoondown \underset{\hat{\mathsf{C}} \in \mathbf{ch}(\hat{\mathsf{N}})}{\arg\max} \left\{ w_{\hat{\mathsf{N}},\hat{\mathsf{C}}} \, \hat{\mathsf{C}} \right\}$
9:         else if $\hat{\mathsf{N}}$ is a product node then
10:            $\forall \hat{\mathsf{C}} \in \mathbf{ch}(\hat{\mathsf{N}}) : Q \leftharpoondown \hat{\mathsf{C}}$
11:        else if $\hat{\mathsf{N}}$ is a maximizing distribution node then
12:            $\mathsf{N} \leftarrow$ corresponding distribution node
13:            $\mathbf{x}^*[\mathbf{sc}(\mathsf{N})] = \underset{\mathbf{x} \in \mathbf{e}[\mathbf{sc}(\mathsf{N})]}{\arg\max} \; \mathsf{N}(\mathbf{x})$
14:        end if
15:    end while
16:    return $\mathbf{x}^*$
17: end procedure
```

Fig. 7. Pseudo-code for MPE inference in selective SPNs.

MPE solution over both model RVs and LVs. The states of the LVs were assigned during max-backtracking, as sum-children and LV states are in one-to-one correspondence. The states of the LVs whose sums are *not visited* during backtracking, are not assigned – again, this causes some confusion, since some LVs appear to be undefined in some contexts, cf. the illustrations in Section 2. However, since this algorithm was used as approximation for MPE over model RVs by discarding the states of the LVs, this situation was not paid any further attention.

Nevertheless, as we show here, applying MPESelective to original (non-selective) SPNs effectively "simulates" MPESelective in the corresponding augmented SPN. Thereby, however, *deterministic* twin-weights are implicitly assumed, i.e. twin-weights which are 0, except a single 1. To see this, let us modify MPESelective, such that it can be applied to an original SPN, but returning an MPE solution for the corresponding augmented SPN. First note that in the augmented MPN, every twin node simply outputs the maximal twin-weight among all children whose states are contained in evidence $\mathbf{e}$. For twin node $\bar{\mathsf{S}}$, let this maximal weight be denoted by $\hat{w}_{\bar{\mathsf{S}}}$. The effect of the twin nodes can now be simulated in the original SPN by replacing each weight $w_{\mathsf{S},\mathsf{C}}$ in the original SPN by $w_{\mathsf{S},\mathsf{C}} \times \tilde{w}_{\mathsf{S},\mathsf{C}}$. Here $\tilde{w}_{\mathsf{S},\mathsf{C}}$ is a correction factor and given as $\tilde{w}_{\mathsf{S},\mathsf{C}} = \prod_{\bar{\mathsf{S}}} \hat{w}_{\bar{\mathsf{S}}}$, where the product runs over all twins of those sums for which S is a conditioning sum. By using these corrected weights, each max node in the corresponding MPN gets the same input as in the MPN of the augmented SPN, i.e. the twin nodes are simulated. We can identify the maximizing states of those LVs whose sums are visited during backtracking, as in [1]. The states of the sums which are not visited are given by the child which correspond to the maximal twin-weight $\hat{w}_{\bar{\mathsf{S}}}$. Pseudo-code for this somewhat technical modification of MPESelective can be found in [19].

We see that the algorithm used in [1] is essentially equivalent to MPESelective in augmented SPNs when $\tilde{w}_{\mathsf{S},\mathsf{C}} = 1$ for all sum nodes, which implies that the twin-weights are deterministic. Therefore, although the LV model in [1] is not completely specified and it was not shown that the Viterbi-like algorithm recovers an MPE solution, it nevertheless corresponds to MPE inference in the augmented SPN for special twin-weights, i.e. deterministic weights.

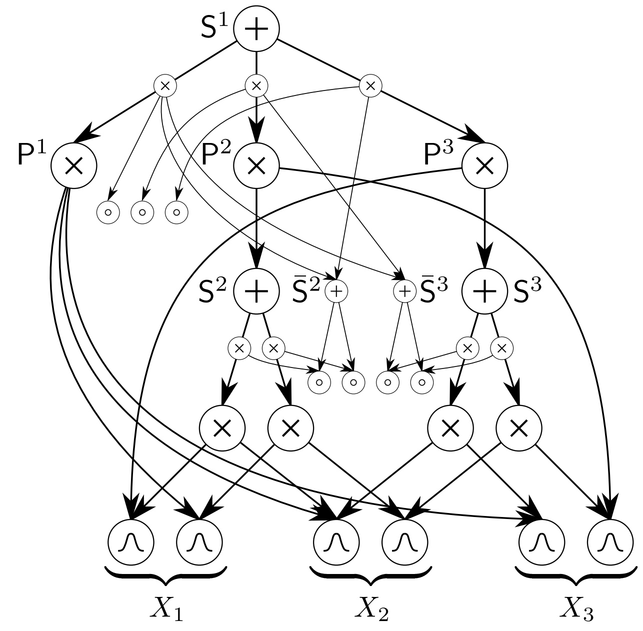


Fig. 8. Illustration of the low-depth bias using an SPN over RVs $\{X_1, X_2, X_3\}$. The structure introduced by augmentation is depicted by small nodes and edges. When deterministic twin-weights are used, the state of Z_{S^1} corresponding to P^1 is preferred over P^2 and P^3, since their probabilities are "dampened" by the weights of S^2 and S^3, respectively.

However, using deterministic twin-weights is a rather unnatural choice, since this prefers one arbitrary state over the others in cases where this LV is actually "rendered irrelevant". In this case, MPE inference also has a bias towards less structured sub-models, which we call *low-depth bias*. This is illustrated in Fig. 8, which shows an SPN over three RVs X_1, X_2, X_3. The augmented SPN has two twin sum nodes $\bar{\mathsf{S}}^2$ and $\bar{\mathsf{S}}^3$, corresponding to S^2 and S^3, respectively. When their twin-weights are deterministic, the selection of the state of Z_{S^1} is *biased* towards the state corresponding to P^1, which is a distribution assuming independence among X_1, X_2 and X_3. This comes from the fact, that the values of P^2 and P^3 are dampened by the weights of S^2 and S^3, respectively, which are generally smaller than 1. Therefore, when using deterministic weights for twin sum nodes, we introduce a bias towards the selection of sub-SPNs that are less deep and less structured. Using uniform weights for twin sum nodes is somewhat "fairer", since in this case P^1 gets dampened by $\bar{\mathsf{S}}^2$ and $\bar{\mathsf{S}}^3$, P^2 by S^2 and $\bar{\mathsf{S}}^3$, and P^3 by $\bar{\mathsf{S}}^2$ and S^3. Uniform weights are to some extend the opposite choice to deterministic twin-weights: the former represent the strongest possible dampening via twin-weights and therefore actually *penalize* less structured distributions. Investigating these effects further is subject to future work.

5 Experiments

5.1 Experiments with EM Algorithm

In [1], [40] SPNs were applied to image data, where a generic architecture reminiscent to convolutional neural networks was proposed. We refer to this architecture as PD architecture. Standard EM was not used in experiments for two reasons: First, explicitly constructing the proposed structure and to train it with standard EM is hardly possible with current hardware, since the number of nodes grows $\mathcal{O}(l^3)$, where l is the square-length of the modeled image domain in pixels [19]. Instead, a sparse hard EM algorithm was used, which virtualizes the PD structure, i.e. sum and products are generated on the fly (see [40] for details). Second, using standard EM seemed unsuited to train large and dense SPNs, either because it is trapped in local optima or due to the gradient vanishing phenomenon.

In our experiments,[2] we investigated three questions:

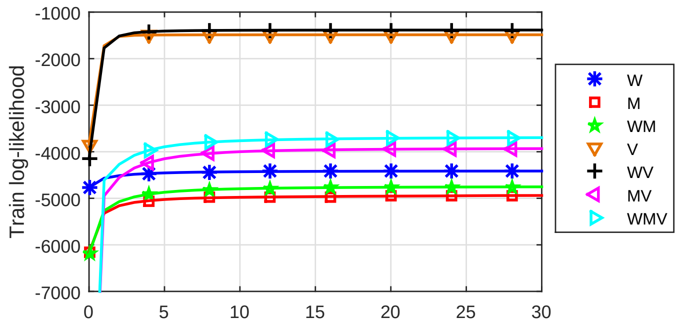
1) Is our derivation of EM correct, both for complete and missing data?
2) Can the result of hard EM [1] be improved by standard EM?
3) Given a suited sparse structure, does EM yield a good solution for parameters?

Question 1) is important since the original derivation contained an error. Questions 2) and 3) are concerned with the general applicability of EM for training SPN.

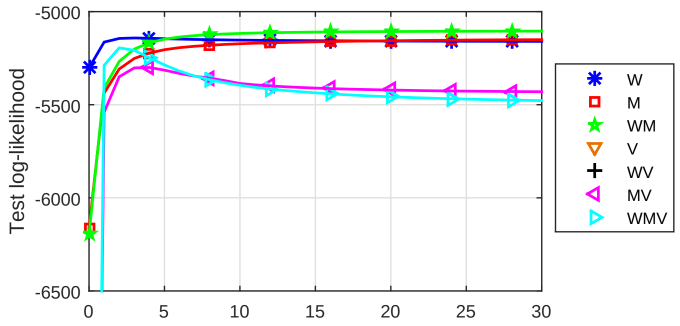
We used the same datasets and SPN structures as in [1], obtainable from [40]. The datasets comprise Caltech-101 (inclusive background class) [42] and the ORL face images [43], i.e. in total 103 datasets. The input distributions in these SPNs are single-dimensional Gaussians (4 for each pixel), where means were set to the averages of the 4-quantiles and variances were constantly 1. We ran EM (Fig. 6) for 30 iterations, with various settings:

- Update any combination of the three different types of parameters, i.e. sum-weights, Gaussian means and Gaussian variances. Each set of parameters types is encoded by a string of letters W (weights), M (means) and V (variances). (7 combinations)
- Use original parameters for initialization, obtained from [40], or use 3 random initialization, where sum-weights are drawn from a Dirichlet distribution with uniform $\alpha = 1$ hyper-parameter (i.e. uniform distribution on the standard simplex), Gaussian means are uniformly drawn from $[-1, 1]$ and Gaussian variances from $[0.01, 1]$. Only parameters which are actually updated are initialized randomly; otherwise the original parameters [1] are used and kept fixed. (4 combinations)
- Use complete data or missing training data, randomly discarding 33% or 66% of the observations, independently for each sample. (3 combinations)

Thus, in total we ran EM $7 \times 4 \times 3 \times 103 = 8652$ times, yielding 259560 EM-iterations. To avoid pathological solutions we used a lower bound of 0.01 for the Gaussian variances. In *no iteration* we observed a decreasing likelihood on the training set,[3] i.e. our derived EM algorithm showed monotonicity in our experiments. Moreover, as can be seen in Fig. 9a, the training log-likelihood actually increased over iterations. The curves for the missing data scenarios are similar. This gives affirmative evidence for question 1).



(a)



(b)

Fig. 9. Normalized log-likelihood over EM-iterations, averaged over all 103 datasets and 3 random initializations. (a): Training set. (b): Test set; Curves for V and WV are outside the displayed region, for better readability of the other curves. They start at approximately -8000 nats and decreased to approximately -11000 nats.

Fig. 9b shows the log-likelihood on the test set. Note that optimizing the parameter sets V and WV led to severe overfitting: while achieving extremely high likelihoods on the training set, they achieved extremely poor likelihoods on the test set. Also the parameter sets MV and WMV tend to overfit, although not as strong as V and WV.

Regarding question 2), we closer inspected the test log-likelihood when the original parameters are used for initialization, i.e. when the parameters obtained by [40] are post-trained using EM. Table 1 summarizes the results. When parameter sets not including Gaussian variances are optimized (i.e. W, M, and WM), the test log-likelihood increased most of the time, i.e. for 83.5% (M) to up to 92.23% (WM) of the datasets. Furthermore, having oracle knowledge about the ideal number of iterations (i.e. column best), the average log-likelihood increased by 0.58% (M) to up to 1.39% (WM) relative to the original parameters. Most of this improvement happens in the first iteration, yielding 0.52% (M) up to 1.05% (WM) improvement. These results indicate that the parameters obtained by [40] slightly underfit the given datasets. Similar as in Fig. 9, we see that parameter sets including the Gaussian variances (V,

2. Code available under [41].

3. Except for tiny occasional decreases (always $< 10^{-8}$) after EM had converged, which can be attributed to numerical artifacts.

TABLE 1
Changes in test log-likelihoods when original parameters are post-trained using EM. % inc.: percentage of datasets where log-likelihood increased in the first iteration. % all, % pos., % neg.: relative change of log-likelihood, averaged over all datasets, datasets with positive change, datasets with negative change, respectively.

		after 1st iteration			best		
	% inc.	% all.	% pos.	% neg.	% all	% pos.	% neg.
W	91.26	0.55	0.61	-0.03	0.87	0.96	-0.03
M	83.50	0.52	0.67	-0.21	0.58	0.73	-0.21
WM	92.23	1.06	1.18	-0.30	1.39	1.53	-0.30
V	39.81	-13.47	14.44	-31.93	-13.45	14.51	-31.93
WV	39.81	-13.41	14.79	-32.06	-13.33	14.98	-32.06
MV	38.83	-17.24	14.27	-37.25	-17.21	14.35	-37.25
WMV	38.83	-17.18	14.63	-37.37	-17.12	14.78	-37.37

TABLE 2
Log-likelihoods when sum-weights (W) are trained, using random initialization. % >: percentage of data sets, where log-likelihood is larger than for original parameters. % all, % pos., % neg.: relative log-likelihood w.r.t. original parameters, for all data sets, data sets where relative log-likelihood is positive/negative, respectively.

	after 1st iteration				best			
	%>	% all.	% pos.	% neg.	%>	% all	% pos.	% neg.
train	70.87	0.68	1.38	-1.00	100.00	3.97	3.97	-
test	41.75	-0.11	0.40	-0.48	67.96	0.46	0.76	-0.18

WV, MV, WMV) are prone to overfitting: more than 60% of the datasets decreased their test log-likelihood during EM. However, in the remaining 40% of the datasets, the test log-likelihood could be improved *substantially* by at least 14% on average.

We now turn to question 3). As pointed out above, a hard EM variant was used in [1], [40] which at the same time finds the effective SPN structure. Optimizing W using the 3 random initialization amounts to using the oracle structure obtained by [1], [40], discarding the learned parameters. For each dataset we selected the random initialization which yielded the highest likelihood on the training set in iteration 30. For this run, we compared the log-likelihoods with the log-likelihoods obtained by the original parameters. The results are summarized in Table 2. We see that on all data sets the log-likelihood on the training set is larger than for the original parameters. This is also the case for each individual random start (not just best one) – every random restart always yielded a higher training log-likelihood than the original parameters. Thus, by considering the actual optimization objective – the likelihood on the training set – EM successfully trains SPNs, given a suited oracle structure. Furthermore, as can be seen in Table 2, EM is also not more prone to overfitting than the algorithm in [1]: on 67.96% of the datasets, EM delivered a higher test log-likelihood than the original parameters, when using oracle knowledge about the ideal number of iterations (column best).

5.2 Experiments with MPE Inference

To illustrate correctness of MPESelective (Fig. 7) when applied to augmented SPNs, we generated SPNs using the PD architecture [1], arranging 4, 9 and 16 binary RVs in a 2×2, 3×3 and 4×4 grid, respectively. As inputs we used two indicator variables for each RV representing their two states. The sum-weights were drawn from a Dirichlet distribution

TABLE 3
Differences of log-likelihood to the ground-truth MPE solution found by exhaustive enumeration, averaged over 100 independent draws of sum-weights. Numbers in parentheses are the number of times where an MPE solution was found. Results for augmented SPNs using uniform twin-weights.

		MPEDet	MPEUni
4 RVs	$\alpha = 0.5$	0.00 (100)	0.00 (100)
	$\alpha = 1.0$	0.00 (100)	0.00 (100)
	$\alpha = 2.0$	0.00 (100)	0.00 (100)
9 RVs	$\alpha = 0.5$	-0.10 (70)	0.00 (100)
	$\alpha = 1.0$	-0.10 (68)	0.00 (100)
	$\alpha = 2.0$	-0.11 (62)	0.00 (100)
16 RVs	$\alpha = 0.5$	-0.63 (19)	0.00 (100)
	$\alpha = 1.0$	-0.85 (12)	0.00 (100)
	$\alpha = 2.0$	-0.82 (12)	0.00 (100)

TABLE 4
Similar as in Table 3. Results for augmented SPNs using deterministic twin-weights.

		MPEDet	MPEUni
4 RVs	$\alpha = 0.5$	0.00 (100)	0.00 (100)
	$\alpha = 1.0$	0.00 (100)	0.00 (100)
	$\alpha = 2.0$	0.00 (100)	0.00 (100)
9 RVs	$\alpha = 0.5$	0.00 (100)	-0.10 (70)
	$\alpha = 1.0$	0.00 (100)	-0.12 (68)
	$\alpha = 2.0$	0.00 (100)	-0.15 (62)
16 RVs	$\alpha = 0.5$	0.00 (100)	-0.89 (19)
	$\alpha = 1.0$	0.00 (100)	-1.11 (12)
	$\alpha = 2.0$	0.00 (100)	-1.01 (12)

with uniform α-parameters, where $\alpha \in \{0.5, 1, 2\}$. For all networks we drew 100 independent parameters sets. We ran MPESelective on the augmented SPN, once equipped with uniform twin-weights and once with deterministic twin-weights. For uniform twin-weights, we denote the result obtained by MPESelective as MPEUni. For deterministic twin-weights, we denote the result as MPEDet. As described in Section 4, MPEDet corresponds essentially to the result when MPESelective is applied to the original SPN [1]. For each assignment, the log-likelihoods were evaluated in the augmented SPN with deterministic weights, the augmented SPN with uniform weights and in the original SPN (discarding the states of the LVs). Additionally, we found ground truth MPE assignments in the two augmented SPNs and the original SPN using exhaustive enumeration. The results relative to the ground truth MPE solutions are shown in Tables 3, 4, and 5. As can be seen, MPEUni always finds an MPE solution in the augmented SPN with uniform twin-weights and MPEDet always finds an MPE solution in augmented SPNs with deterministic twin-weights. This gives empirical evidence for the correctness of MPESelective for MPE inference in augmented SPNs.

Furthermore, we wanted to investigate the quality of both algorithms when serving as approximation for MPE inference in the original SPNs. For the SPNs considered here, MPEDet delivered on average slightly better approximations than MPEUni. However, these results should be interpreted with caution, due to the rather similar nature of the distributions considered here. Closer investigating approximate MPE for (original) SPNs is an interesting direction and will be subject to future research.

TABLE 5
Similar as in Table 3. Results for original SPNs.

		MPEDET	MPEUNI
4 RVs	$\alpha = 0.5$	-0.06 (72)	-0.06 (72)
	$\alpha = 1.0$	-0.09 (59)	-0.09 (59)
	$\alpha = 2.0$	-0.10 (52)	-0.10 (52)
9 RVs	$\alpha = 0.5$	-0.31 (32)	-0.38 (27)
	$\alpha = 1.0$	-0.47 (12)	-0.48 (12)
	$\alpha = 2.0$	-0.40 (6)	-0.37 (7)
16 RVs	$\alpha = 0.5$	-0.76 (5)	-1.04 (4)
	$\alpha = 1.0$	-0.76 (3)	-1.18 (2)
	$\alpha = 2.0$	-0.67 (1)	-0.92 (0)

6 CONCLUSION

In this paper we revisited the interpretation of SPNs as hierarchically structured LV model. We pointed out that the original approach to explicitly incorporate LVs does not produce a sound probabilistic model. As a remedy we proposed the augmentation of SPNs and proved its soundness as LV model. Within augmented SPNs, we investigated the independency structure represented as BN, and showed that the sum-weights can be interpreted as structured CPTs within this BN. Using augmented SPNs, we derived the EM algorithm for sum-weights and single-dimensional input distributions from exponential families. While MPE-inference is generally NP-hard in SPNs, we showed that a Viterbi-style backtracking algorithm recovers an MPE solution in selective SPNs, and in particular in augmented SPNs. In experiments we give empirical evidence supporting our theoretical results. We furthermore showed that standard EM can successfully train generative SPNs, given a suitable network structure at hand.

APPENDIX A
PROOFS

A.1 Proof of Proposition 1

If $\mathcal{S}'$ is a complete and decomposable SPN over $\mathbf{X} \cup \mathbf{Z}$, then $\mathcal{S}'(\mathbf{X}) \equiv \mathcal{S}(\mathbf{X})$ is immediate: Computing $\mathcal{S}'(\mathbf{x})$ for any $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ is done by marginalizing $\mathbf{Z}$, i.e. setting all $\lambda_{Z_\mathsf{S}=k} = 1$. In this case, it is easy to see that none of the structural changes modifies the output of the SPN, i.e. the outputs of $\mathcal{S}$ and $\mathcal{S}'$ agree for each $\mathbf{x}$, i.e. $\mathcal{S}'(\mathbf{X}) \equiv \mathcal{S}(\mathbf{X})$.

It remains to show that $\mathcal{S}'$ is complete and decomposable, and that the root's scope is $\mathbf{X} \cup \mathbf{Z}$. Steps 4–11 in AUGMENTSPN introduce the links, representing "private copies" of the sum's children, and clearly leave the SPN complete and decomposable. In steps 13–15 the LV Z_S is introduced in the scope of S and thus in the scope of the root. Since this is done for all sum nodes, all $\mathbf{Z}$ are introduced in the root's scope. Steps 13–15 cannot render products non-decomposable, since this would imply that S is reachable by two distinct children of this product – a contradiction to the fact that the SPN was decomposable before. However, as shown in Fig. 1, steps 13–15 can render ancestor sums incomplete. These are treated in steps 17–23. The twin sum $\bar{\mathsf{S}}$, if introduced, is clearly complete and has scope $\{Z\}$. Furthermore, incompleteness of any conditioning sum S^c can only be caused by links not having Z_S in their scope. The scope of these links is augmented by Z_S in step 21. These links clearly remain decomposable and moreover, S^c is rendered complete now. □

A.2 Proof of Proposition 2

ad *1.)* When deleting the IVs and their links, the scopes of any (twin) sum remains the same, since it is complete and is left with one child. Thus also the scope of any ancestor remains the same.
ad *2.)* The graph of $\mathcal{S}^\mathbf{y}$ is rooted and acyclic, since the root cannot be a link and deleting nodes and edges cannot introduce cycles. When an IV $\lambda_{Y=y}$ is deleted, also the link $\mathsf{P}^y_{\mathsf{S}_Y}$ is deleted, so no internal nodes are left as leaves. The roots in $\mathcal{S}^\mathbf{y}$ and $\mathcal{S}'$ are the same, and by point *1.*, $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ is the scope of the root. $\mathcal{S}^\mathbf{y}$ is also complete and decomposable: Whenever an IV and its link are deleted, the corresponding sum node and twin sum node remain trivially complete, since they are left with a single child. Furthermore, completeness and decomposability of any ancestor of S_Y or $\bar{\mathsf{S}}_Y$ is left intact, since neither S_Y nor $\bar{\mathsf{S}}_Y$ changes its scope.
ad *3.)* According to point *1.*, the scope of N is the same in $\mathcal{S}'$ and $\mathcal{S}^\mathbf{y}$. Since $\mathbf{sc}(\mathsf{N}) \cap \mathbf{Y} = \varnothing$, the disconnected IVs and deleted links are no descendants of N, i.e. no descendants of N are disconnected during configuration. Since N is present in $\mathcal{S}^\mathbf{y}$, it must still be reachable from the root. Therefore also all descendants of N are reachable, i.e. $\mathcal{S}^\mathbf{y}_\mathsf{N} = \mathcal{S}'_\mathsf{N}$.
ad *4.)* When the input is fixed to $\mathbf{x}, \mathbf{z}, \mathbf{y}$, all IVs and links which are deleted from the configured SPN $\mathcal{S}^\mathbf{y}$ evaluate to zero in the augmented SPN $\mathcal{S}'$. The outputs of all sums and twin sums are therefore the same in $\mathcal{S}'$ and $\mathcal{S}^\mathbf{y}$. Therefore, also the output of all other nodes remains the same. This includes the root and therefore $\mathcal{S}^\mathbf{y}(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \mathcal{S}'(\mathbf{x}, \mathbf{z}, \mathbf{y})$, for any $\mathbf{x}, \mathbf{z}$.

When $\mathbf{y}' \neq \mathbf{y}$, then there must be a $Y \in \mathbf{Y}$ such that the IV $\lambda_{Y=\mathbf{y}'[Y]}$ has been deleted, i.e. $\lambda_{Y=\mathbf{y}'[Y]} \notin \mathbf{desc}(\mathsf{N})$, where N is the root of $\mathcal{S}^\mathbf{y}$. Using Lemma 1 in [17], it follows that $\mathcal{S}^\mathbf{y}(\mathbf{x}, \mathbf{z}, \mathbf{y}') = 0$. □

A.3 Proof of Lemma 1

$\mathcal{S}^\mathbf{z}$ must contain either S or $\bar{\mathsf{S}}$, since Z_S is in the scope of the root by Proposition 2. To show that *not both* are in $\mathcal{S}^\mathbf{z}$, let $\mathbf{\Pi}_k$ denote the set of paths of length k from the root to any node N with $Z_\mathsf{S} \in \mathbf{sc}(\mathsf{N})$. For $k > 1$, all paths in $\mathbf{\Pi}_k$ can be constructed by extending each path in $\mathbf{\Pi}_{k-1}$ with each child of this path's last node, if it has Z_S in its scope. Let K be the smallest number such that there is a path in $\mathbf{\Pi}_k$ containing S or $\bar{\mathsf{S}}$.

We show by induction, that $|\mathbf{\Pi}_k| = 1$, $k = 1, \dots, K$. Note that $\mathbf{\Pi}_1$ contains a single path (N), where N is the root, therefore the induction basis holds.

For the induction step, we show that given $|\mathbf{\Pi}_{k-1}| = 1$, then also $|\mathbf{\Pi}_k| = 1$. Let $(\mathsf{N}_1, \dots, \mathsf{N}_{k-1})$ be the single path in $\mathbf{\Pi}_{k-1}$. If N_{k-1} is a product node, then it has a single child C with $Z_\mathsf{S} \in \mathbf{sc}(\mathsf{C})$, due to decomposability. If N_{k-1} is a sum node, then it must be in $\mathbf{ancs}_\mathbf{S}(\mathsf{S}) \backslash \{\mathsf{S}\}$, and therefore has a single child in the configured SPN. Therefore, there is a single way to extend the path and therefore $|\mathbf{\Pi}_k| = 1, k = 1, \dots, K$. This single path does either lead to S or $\bar{\mathsf{S}}$. Since $\mathsf{S} \notin \mathbf{desc}(\bar{\mathsf{S}})$ and $\bar{\mathsf{S}} \notin \mathbf{desc}(\mathsf{S})$, $\mathcal{S}^\mathbf{z}$ contains a single path to one of them, but not to both. □

A.4 Proof of Theorem 1

By Lemma 1, for each $\mathbf{z} \in \mathbf{val}(\mathbf{Z}_p)$ the configured SPN $\mathcal{S}^{\mathbf{z}}$ contains either S or $\bar{\mathsf{S}}$, but not both. Let $\mathcal{Z}$ be the subset of $\mathbf{val}(\mathbf{Z}_p)$ such that S is in $\mathcal{S}^{\mathbf{z}}$ and $\bar{\mathcal{Z}}$ be the subset of $\mathbf{val}(\mathbf{Z}_p)$ such that $\bar{\mathsf{S}}$ is in $\mathcal{S}^{\mathbf{z}}$.

Fix $Z_{\mathsf{S}} = k$ and $\mathbf{z} \in \mathcal{Z}$. We want to compute $\mathcal{S}'(Z_{\mathsf{S}} = k, \mathbf{Y}_n, \mathbf{z})$, i.e. we marginalize $\mathbf{Y}_c$. According to Proposition 2 (*4.*), this equals $\mathcal{S}^{\mathbf{z}}(Z_{\mathsf{S}} = k, \mathbf{Y}_n, \mathbf{z})$. According to Proposition 2 (*3.*), the sub-SPN rooted at former child $\mathsf{C}^k_{\mathsf{S}}$ is the same in $\mathcal{S}'$ and $\mathcal{S}^{\mathbf{z}}$. Since $\mathcal{S}'$ is locally normalized, this sub-SPN is also locally normalized in $\mathcal{S}^{\mathbf{z}}$. Since the scope of the former child $\mathsf{C}^k_{\mathsf{S}}$ is a sub-set of $\mathbf{Y}_c$, which is marginalized, and $\lambda_{Z_{\mathsf{S}}=k} = 1$, the link $\mathsf{P}^k_{\mathsf{S}}$ outputs 1. Since $\lambda_{Z_{\mathsf{S}}=k'} = 0$ for $k' \neq k$, the sum S outputs w_k.

Now consider the set of nodes in $\mathcal{S}^{\mathbf{z}}$ which have Z_{S} in their scope, not including $\lambda_{Z_{\mathsf{S}}=k}$ and $\mathsf{P}^k_{\mathsf{S}}$. Clearly, since $\bar{\mathsf{S}}$ is not in $\mathcal{S}^{\mathbf{z}}$, this set must be $\mathbf{anc}(\mathsf{S})$. Let $\mathsf{N}_1, \dots, \mathsf{N}_L$ be a topologically ordered list of $\mathbf{anc}(\mathsf{S})$, where S is N_1 and N_L is the root. Let $\mathbf{Y}_{n,l} := \mathbf{sc}(\mathsf{N}_l) \cap \mathbf{Y}_n$ and $\mathbf{Z}_l := \mathbf{sc}(\mathsf{N}_l) \cap \mathbf{Z}_p$. We show by induction that for $l = 1, \dots, L$, we have

$$\mathsf{N}_l(Z_{\mathsf{S}} = k, \mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]) = w_k \, \mathsf{N}_l(\mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]). \tag{30}$$

Since $\mathbf{Y}_{n,1} = \emptyset$ and $\mathbf{Z}_1 = \emptyset$, and $\mathsf{N}_1(Z_{\mathsf{S}} = k) = w_k$, the induction basis holds. Assume that (30) holds for all $\mathsf{N}_1, \dots, \mathsf{N}_{l-1}$. If N_l is a sum, we have due to completeness

$$\mathsf{N}_l(Z_{\mathsf{S}} = k, \mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]) = \sum_{\mathsf{C} \in \mathbf{ch}(\mathsf{N}_l)} w_{\mathsf{N}_l,\mathsf{C}} \, w_k \, \mathsf{C}(\mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]) \tag{31}$$

$$= w_k \, \mathsf{N}_l(\mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]), \tag{32}$$

i.e. the induction step holds for sums. When N_l is a product, due to decomposability, it must have a single child with Z_{S} in its scope. Hence, this child must be a node $\mathsf{N}_m \in \mathbf{anc}(\mathsf{S})$ We have

$$\mathsf{N}_l(Z_{\mathsf{S}} = k, \mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]) \tag{33}$$

$$= w_k \, \mathsf{N}_m(\mathbf{Y}_{n,m}, \mathbf{z}[\mathbf{Z}_m]) \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{N}_l) \setminus \mathsf{N}_m} \mathsf{C}(\mathbf{Y}_{n,l} \cap \mathbf{sc}(\mathsf{C})) \tag{34}$$

$$= w_k \, \mathsf{N}_l(\mathbf{Y}_{n,l}, \mathbf{z}[\mathbf{Z}_l]), \tag{35}$$

i.e. the induction step holds for products. Therefore, by induction, (30) also holds for the root, and (11) follows.

Now we show (12). If the twin sum $\bar{\mathsf{S}}$ does not exist, $\bar{\mathcal{Z}}$ is empty and (12) holds trivially. Otherwise, fix the input to $Z_{\mathsf{S}} = k$ and $\mathbf{z} \in \bar{\mathcal{Z}}$. Clearly, $\bar{\mathsf{S}}$ outputs $\bar{w}_k$ and (12) can be shown in similar way as (11). □

A.5 Proof of Theorem 2

We prove the theorem using an inductive argument. The theorem clearly holds for any $\hat{\mathsf{D}}$ by definition. Consider a product $\hat{\mathsf{P}}$ and assume the theorem holds for all $\mathbf{ch}(\hat{\mathsf{P}})$. Then the theorem also holds for $\hat{\mathsf{P}}$, since

$$\hat{\mathsf{P}}(\mathcal{X}) = \prod_{\mathsf{C} \in \mathbf{ch}(\hat{\mathsf{P}})} \max_{\mathbf{x} \in \mathcal{X}} \mathsf{C}(\mathbf{x}) = \max_{\mathbf{x} \in \mathcal{X}} \prod_{\mathsf{C} \in \mathbf{ch}(\hat{\mathsf{P}})} \mathsf{C}(\mathbf{x}) = \max_{\mathbf{x} \in \mathcal{X}} \mathsf{P}(\mathbf{x}), \tag{36}$$

where the max and the product can be switched due to decomposability.

Now consider a max node $\hat{\mathsf{S}}$ and its corresponding sum node S. Let the *support* of an SPN-node N be the set $\mathrm{sup}_{\mathsf{N}} := \{\mathbf{x} \,|\, \mathsf{N}(\mathbf{x}) > 0\}$. Since S is selective, its support is partitioned by the supports of its children, i.e. $\mathrm{sup}_{\mathsf{S}} = \bigcup_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} \mathrm{sup}_{\mathsf{C}}$, $\mathrm{sup}_{\mathsf{C}'} \bigcap \mathrm{sup}_{\mathsf{C}''} = \emptyset$, for $\mathsf{C}' \neq \mathsf{C}''$. Assuming that the theorem holds for all $\mathbf{ch}(\hat{\mathsf{S}})$, we have

$$\hat{\mathsf{S}}(\mathcal{X}) = \max_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}} \max_{\mathbf{x} \in \mathcal{X}} \mathsf{C}(\mathbf{x}) \tag{37}$$

$$= \max_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}} \max_{\mathbf{x} \in \mathrm{sup}_{\mathsf{C}} \cap \mathcal{X}} \mathsf{C}(\mathbf{x}) \tag{38}$$

$$= \max_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} \max_{\mathbf{x} \in \mathrm{sup}_{\mathsf{C}} \cap \mathcal{X}} w_{\mathsf{S},\mathsf{C}} \, \mathsf{C}(\mathbf{x}) \tag{39}$$

$$= \max_{\mathbf{x} \in \mathrm{sup}_{\mathsf{S}} \cap \mathcal{X}} \mathsf{S}(\mathbf{x}) = \max_{\mathbf{x} \in \mathcal{X}} \mathsf{S}(\mathbf{x}). \tag{40}$$

In (38) we have a slight abuse of notation, as we actually should use suprema over the sets $\mathrm{sup}_{\mathsf{C}} \cap \mathcal{X}$ and define the supremum over the empty set as 0. In (39) we used the fact that the support of the sum node is partitioned by the supports of its children and that for selective sums we have $\mathsf{S} = w_{\mathsf{S},\mathsf{C}} \, \mathsf{C}$ whenever we have single child with $\mathsf{C} > 0$.

We see that the induction step also holds for $\hat{\mathsf{S}}$. Therefore, the theorem holds for all nodes. □

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the **Austrian Science Fund (FWF): P25244-N15** and **Austrian Science Fund (FWF): P27803-N15**. This research was partly funded by **ONR grant N00014-16-1-2697** and **AFRL contract FA8750-13-2-0019**.

References

[1] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," in *Proceedings of UAI*, 2011, pp. 337–346.
[2] R. Gens and R. Domingos, "Learning the structure of sum-product networks," in *Proceedings of ICML*, 2013, pp. 873–880.
[3] A. Dennis and D. Ventura, "Learning the architecture of sum-product networks using clustering on variables," in *Proceedings of NIPS*, 2012, pp. 2042–2050.
[4] R. Peharz, B. Geiger, and F. Pernkopf, "Greedy part-wise learning of sum-product networks," in *Proceedings of ECML/PKDD*, vol. 8189. Springer Berlin, 2013, pp. 612–627.
[5] M. Amer and S. Todorovic, "Sum-product networks for modeling activities with stochastic structure," in *Proceedings of CVPR*, 2012, pp. 1314–1321.
[6] R. Gens and P. Domingos, "Discriminative learning of sum-product networks," in *Proceedings of NIPS*, 2012, pp. 3248–3256.
[7] R. Peharz, G. Kapeller, P. Mowlaee, and F. Pernkopf, "Modeling speech with sum-product networks: Application to bandwidth extension," in *Proceedings of ICASSP*, 2014, pp. 3699–3703.
[8] W. C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, and K. M. A. Chai, "Language modeling with sum-product networks," in *Proceedings of Interspeech*, 2014, pp. 2098–2102.
[9] M. Zöhrer, R. Peharz, and F. Pernkopf, "Representation learning for single-channel source separation and bandwidth extension," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2398–2409, 2015.
[10] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.
[11] Z. Ghahramani and M. Jordan, "Supervised learning from incomplete data via an EM approach," in *Proceedings of NIPS*, 1994, pp. 120–127.
[12] S.-W. Lee, C. Watkins, and B. Zhang, "Non-parametric Bayesian sum-product networks," in *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.
[13] M. Trapp, R. Peharz, M. Skowron, T. Madl, F. Pernkopf, and R. Trappl, "Structure inference in sum-product networks using infinite sum-product trees," in *NIPS Workshop on Practical Bayesian Nonparametrics*, 2016.

[14] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.
[15] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
[16] A. Darwiche, "A differential approach to inference in Bayesian networks," *Journal of the ACM*, vol. 50, no. 3, pp. 280–305, 2003.
[17] R. Peharz, S. Tschiatschek, F. Pernkopf, and P. Domingos, "On theoretical properties of sum-product networks," in *Proceedings of AISTATS*, 2015, pp. 744–752.
[18] C. de Campos, "New complexity results for MAP in Bayesian networks," in *Proceedings of IJCAI*, 2011, pp. 2100–2106.
[19] R. Peharz, "Foundations of sum-product networks for probabilistic modeling," Ph.D. dissertation, Graz University of Technology, 2015.
[20] R. Peharz, R. Gens, and P. Domingos, "Learning selective sum-product networks," in *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.
[21] H. Zhao, M. Melibari, and P. Poupart, "On the relationship between sum-product networks and Bayesian networks," in *Proceedings of ICML*, 2015, pp. 116–124.
[22] A. Darwiche, "Compiling knowledge into decomposable negation normal form," in *Proceedings of IJCAI*, 1999, pp. 284–289.
[23] ——, "Decomposable negation normal form," *Journal of the ACM*, vol. 48, no. 4, pp. 608–647, 2001.
[24] A. Darwiche and P. Marquis, "A knowledge compilation map," *Journal of Artificial Intelligence Research*, vol. 17, no. 1, pp. 229–264, 2002.
[25] A. Darwiche, "A logical approach to factoring belief networks," in *Proceedings of KR*, 2002, pp. 409–420.
[26] D. Lowd and P. Domingos, "Learning arithmetic circuits," in *Proceedings of UAI*, 2008, pp. 383–392.
[27] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, "Context-specific independence in Bayesian networks," in *Proceedings of UAI*, 1996, pp. 115–123.
[28] D. Lowd and A. Rooshenas, "Learning Markov networks with arithmetic circuits," in *Proceedings of AISTATS*, 2013, pp. 406–414.
[29] A. Rooshenas and D. Lowd, "Learning sum-product networks with direct and indirect variable interactions," in *Proceedings of ICML*, 2014, pp. 710–718.
[30] T. Adel, D. Balduzzi, and A. Ghodsi, "Learning the structure of sum-product networks via an SVD-based algorithm," in *Proceedings of UAI*, 2015, pp. 32–41.
[31] A. Vergari, N. Di Mauro, and F. Esposito, "Simplifying, regularizing and strengthening sum-product network structure learning," in *Proceedings of ECML/PKDD*, 2015, pp. 343–358.
[32] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," in *Proceedings of NIPS*, 2011, pp. 666–674.
[33] A. Friesen and P. Domingos, "The sum-product theorem: A foundation for learning tractable models," in *Proceedings of ICML*, 2016, pp. 1909–1918.
[34] N. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*. Wiley, 1994.
[35] H. Zhao and P. Poupart, "A unified approach for learning the parameters of sum-product networks," http://arxiv.org/abs/1601.00318.
[36] H. Bodlaender, F. van den Eijkhof, and L. van der Gaag, "On the complexity of the MPA problem in probabilistic networks," in *Proceedings of ECAI*, 2002, pp. 675–679.
[37] J. D. Park and A. Darwiche, "Complexity results and approximation strategies for MAP explanations," *Journal of Artificial Intelligence Research*, vol. 21, no. 1, pp. 101–133, 2004.
[38] J. Kwisthout, "Most probable explanations in Bayesian networks: Complexity and tractability," *International Journal of Approximate Reasoning*, vol. 52, no. 9, pp. 1452–1469, 2011.
[39] A. Darwiche, *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2014.
[40] http://alchemy.cs.washington.edu/spn, (online).
[41] http://spn.cs.washington.edu/pubs.shtml https://www.spsc.tugraz.at/tools, (online).
[42] L. Fei-Fei, R. Fergus, and R. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.
[43] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, 1994, pp. 138–142.

Robert Peharz received his MSc degree in Computer Engineering and his Ph.D degree in Electrical Engineering from Graz University of Technology. His main research interest lies in machine learning, in particular probabilistic modeling, with applications to signal processing, speech and audio processing, and computer vision. Currently, he is with the research unit iDN (interdisciplinary developmental neuroscience) at the Medical University of Graz, applying machine learning techniques to detect early markers of neurological conditions in infants. He is funded by the BioTechMed-Graz cooperation, an interdisciplinary network of the 3 major universities in Graz with a focus on basic bio-medical research, technological development and medical applications.

Robert Gens received the S.B. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009 and the M.Sc. degree in computer science and engineering from the University of Washington, Seattle, WA, USA, in 2012. During the Summer of 2014, he was a Research Intern at Microsoft Research, Redmond, WA, USA. He is currently a Ph.D. student of computer science and engineering at the University of Washington, Seattle, WA, USA. He is supported by the 2014 Google Ph.D. Fellowship in Deep Learning. Mr. Gens was the recipient of an Outstanding Student Paper Award at the Neural Information Processing Systems conference in 2012.

Franz Pernkopf received his MSc (Dipl. Ing.) degree in Electrical Engineering at Graz University of Technology, Austria, in summer 1999. He earned a Ph.D degree from the University of Leoben, Austria, in 2002. In 2002 he was awarded the Erwin Schrödinger Fellowship. He was a Research Associate in the Department of Electrical Engineering at the University of Washington, Seattle, from 2004 to 2006. Currently, he is Associate Professor at the Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Austria. His research interests include machine learning, discriminative learning, graphical models, feature selection, finite mixture models, and image- and speech processing applications.

Pedro Domingos is a professor of computer science at the University of Washington and the author of The Master Algorithm. He is a winner of the SIGKDD Innovation Award, the highest honor in data science. He is a Fellow of the Association for the Advancement of Artificial Intelligence, and has received a Fulbright Scholarship, a Sloan Fellowship, the National Science Foundations CAREER Award, and numerous best paper awards. He received his Ph.D. from the University of California at Irvine and is the author or co-author of over 200 technical publications. He has held visiting positions at Stanford, Carnegie Mellon, and MIT. He co-founded the International Machine Learning Society in 2001. His research spans a wide variety of topics in machine learning, artificial intelligence, and data science, including scaling learning algorithms to big data, maximizing word of mouth in social networks, unifying logic and probability, and deep learning.