On Theoretical Properties of Sum-Product Networks

Robert Peharz[†] **Sebastian Tschiatschek**[†] **Franz Pernkopf**[†] **Pedro Domingos**[‡]

[†]Signal Processing and Speech Communication Lab, Graz University of Technology

[‡]Dept. of Computer Science & Engineering, University of Washington

Abstract

Sum-product networks (SPNs) are a promising avenue for probabilistic modeling and have been successfully applied to various tasks. However, some theoretic properties about SPNs are not yet well understood. In this paper we fill some gaps in the theoretic foundation of SPNs. First, we show that the weights of any complete and consistent SPN can be transformed into locally normalized weights without changing the SPN distribution. Second, we show that consistent SPNs cannot model distributions significantly (exponentially) more compactly than decomposable SPNs. As a third contribution, we extend the inference mechanisms known for SPNs with finite states to generalized SPNs with arbitrary input distributions.

1 INTRODUCTION

Sum-product networks (SPNs) are a promising type of probabilistic model and showed impressive results for image completion [Poon and Domingos, 2011, Dennis and Ventura, 2012, Peharz et al., 2013], computer vision [Amer and Todorovic, 2012], classification [Gens and Domingos, 2012] and speech/language modeling [Peharz et al., 2014, Cheng et al., 2014]. The potentially deep structure of SPNs allows them to capture complex interaction of the model variables, while at the same time still guaranteeing efficient inference. Inference cost is directly related to the network size, opening the door for inference-aware learning. However, some aspects of SPNs are not yet well understood and there are many open questions worth investigating. In this paper, we revisit the foundations of SPNs and present new theoretic insights, consisting of three main contributions.

The first contribution is concerned with normalized weights: SPNs have been defined to have *nonnegative* parameters (sum weights), where most work so far has additionally assumed locally normalized parameters, i.e. that for each sum node the associated weights sum up to 1. This assures that the SPN distribution is readily correctly normalized. Up to now it is not clear if this somehow restricts our modeling abilities, i.e. are there "fewer" locally normalized SPNs than unnormalized SPNs? The answer is no, and furthermore we provide an algorithm which transforms any unnormalized parameter set into a locally normalized one, *without changing the modeled distribution*. Thus, from a representational point of view we can always assume locally normalized parameters.

As a second contribution we investigate the two notions of *consistency* and *decomposability*, either of which is sufficient to make inference in SPNs efficient, given that the SPN is also *complete*. Consistency is the more general condition, i.e. any decomposable SPN is also consistent, but not vice versa. So far, only decomposable SPNs have been considered in practice, since decomposability is easier to ensure. An interesting question is: How much do we lose when we go for decomposability? Can we model our distribution exponentially more concise, using consistency? The answer we give is negative. We show that any distribution represented by a consistent SPN can also be represented by a decomposable SPN, using only polynomially more arithmetic operations. Furthermore, we show that consistent SPNs are *not amenable* to the differential approach [Darwiche, 2003], i.e. simultaneously obtaining all marginal posteriors given evidence. This fact was not mentioned in the literature so far. Decomposable SPNs, on the other hand, represent network polynomials and *can* be used in the differential approach.

SPNs were originally defined for finitely many states using indicator variables as leaves [Poon and Domin-

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. Copyright 2015 by the authors.

gos, 2011]. However, SPNs can be generalized and equipped with arbitrary distributions as inputs. As a third contribution, we extend the inference mechanisms known for finite state SPNs to these generalized SPNs. We show that marginalization in generalized SPNs reduces to marginalization over the input distributions. Furthermore, we propose a generalized differential approach, which yields the marginal posteriors of all variables given some evidence.

Throughout the paper, we use the following notation. Random variables (RVs) are denoted as X, Y and Z. The set of values an RV X can assume is $\mathbf{val}(X)$, where corresponding lower-case letters denote their values, e.g. x is an element of $\mathbf{val}(X)$. Sets of random variables are denoted by boldface letters, e.g. $\mathbf{X} = \{X_1, \ldots, X_N\}$. We define $\mathbf{val}(\mathbf{X})$ as the set of compound values, i.e. the Cartesian product $\mathbf{val}(\mathbf{X}) = \times_{n=1}^{N} \mathbf{val}(X_n)$, and use $\mathbf{x}$ for elements of $\mathbf{val}(\mathbf{X})$. For $\mathbf{Y} \subseteq \mathbf{X}$ and $X \in \mathbf{X}$, $\mathbf{x}[\mathbf{Y}]$ and $\mathbf{x}[X]$ denote the projections of $\mathbf{x}$ onto $\mathbf{Y}$ and onto X, respectively. The elements of $\mathbf{val}(\mathbf{X})$ represent *complete* evidence, assigning each RV in $\mathbf{X}$ a value. *Partial* evidence about X is represented as *subsets* $\mathcal{X} \subseteq \mathbf{val}(X)$, which is an element of the sigma-algebra $\mathcal{A}_X$ induced by RV X. For discrete RVs, we assume $\mathcal{A}_X = 2^{\mathbf{val}(X)}$, i.e. the power-set of $\mathbf{val}(X)$. For continuous RVs, we use $\mathcal{A}_X = \{\mathcal{X} \in \boldsymbol{\mathcal{B}} \mid \mathcal{X} \subseteq \mathbf{val}(X)\}$, where $\boldsymbol{\mathcal{B}}$ are the Borel-sets over $\mathbb{R}$. For sets of RVs $\mathbf{X} = \{X_1, \ldots, X_N\}$, 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 as $\boldsymbol{\mathcal{X}}$. When $\mathbf{Y} \subseteq \mathbf{X}$, $X \in \mathbf{X}$ and $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$, we define $\boldsymbol{\mathcal{X}}[\mathbf{Y}] := \{\mathbf{x}[\mathbf{Y}] \mid \mathbf{x} \in \boldsymbol{\mathcal{X}}\}$ and $\boldsymbol{\mathcal{X}}[X] = \{\mathbf{x}[X] \mid \mathbf{x} \in \boldsymbol{\mathcal{X}}\}$. An example for partial evidence is provided in the supplementary paper.

The paper is organized as follows: In Section 3 we discuss SPNs over random variables with finitely many states. Inference for generalized SPNs is discussed in Section 4. Section 5 concludes the paper. In the main paper, we provide proofs only for the main results. Proofs for all results can be found in the supplementary paper.

2 RELATED WORK

Darwiche [2003] introduced *network polynomials* (NPs) for Bayesian networks over RVs $\mathbf{X}$ with finitely many states. Poon and Domingos [2011] generalized them to unnormalized distributions, i.e. any nonnegative function $\Phi(\mathbf{x})$ where $\exists \mathbf{x}\colon \Phi(\mathbf{x}) > 0$. We introduce the so-called indicator variables (IVs) for each RV and each state, where $\lambda_{X=x} \in \mathbb{R}$ is the IV for X and state x. Let $\boldsymbol{\lambda}$ be a vector collecting all IVs of $\mathbf{X}$.

Definition 1 (Network Polynomial)**.** *Let Φ be an unnormalized probability distribution over RVs $\mathbf{X}$ with finitely many states and $\boldsymbol{\lambda}$ be their IVs. The* network polynomial f_Φ *of Φ is defined as*

$$f_\Phi(\boldsymbol{\lambda}) := \sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \Phi(\mathbf{x}) \prod_{X \in \mathbf{X}} \lambda_{X=\mathbf{x}[X]}. \tag{1}$$

The NP contains exponentially many term in the number of RVs. As shown by Darwiche [2003], it represents the distribution Φ in the following sense. Restrict the IVs to $\{0, 1\}$ and as a function of $\mathbf{x} \in \mathbf{val}(\mathbf{X})$:

$$\lambda_{X=x}(\mathbf{x}) = \begin{cases} 1 & \text{if } x = \mathbf{x}[X] \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Let $\boldsymbol{\lambda}(\mathbf{x})$ be the corresponding vector-valued function, collecting all $\lambda_{X=x}(\mathbf{x})$. When we input $\boldsymbol{\lambda}(\mathbf{x})$ to f_Φ, all but one of the terms evaluate to 0, i.e. $f_\Phi(\boldsymbol{\lambda}(\mathbf{x})) = \Phi(\mathbf{x})$. Now, let us extend (2) to a function of $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$:

$$\lambda_{X=x}(\boldsymbol{\mathcal{X}}) = \begin{cases} 1 & \text{if } x \in \boldsymbol{\mathcal{X}}[X] \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Let $\boldsymbol{\lambda}(\boldsymbol{\mathcal{X}})$ be the corresponding vector-valued function. It is easily verified that $f_\Phi(\boldsymbol{\lambda}(\boldsymbol{\mathcal{X}})) = \sum_{\mathbf{x} \in \boldsymbol{\mathcal{X}}} \Phi(\mathbf{x})$, i.e. *the NP returns the unnormalized probability measure for arbitrary sets in $\mathcal{H}_{\mathbf{X}}$*. Thus, the NP compactly describes marginalization over arbitrary domains of the RVs by simply setting the corresponding IVs to 1. In particular, when we set $\boldsymbol{\lambda} \equiv 1$, it returns the normalization constant of Φ. A direct implementation of the NP is not practical due the exponentially many terms. In [Darwiche, 2002, 2003, Lowd and Domingos, 2008, Lowd and Rooshenas, 2013] exponentially more compact representations were learned using *arithmetic circuits*.

It is important to note that, although the IVs are called *indicator* variables and set to values out of $\{0, 1\}$ by functions (2) and (3), they are in fact *real-valued* variables – this difference is essential when we take *derivatives* w.r.t. the IVs. These derivatives are used in the so-called *differential approach to inference* [Darwiche, 2003]. Taking the first derivative with respect to some $\lambda_{X=x}$ yields

$$\frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\boldsymbol{\mathcal{X}})) = \Phi(x, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]), \tag{4}$$

since the derivatives of sum-terms in (1) with $\mathbf{x}[X] \neq x$ are 0, and the derivatives of sum-terms with $\mathbf{x}[X] = x$ are $\Phi(\mathbf{x}) \prod_{X' \in \mathbf{X} \setminus X} \lambda_{X'=\mathbf{x}[X']}$. Thus, the derivative in (4), which is indeed the standard derivative known from calculus, evaluates Φ for *modified evidence* $x, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]$ and is proportional to the *marginal posterior* $\Phi(x \mid \mathbf{X} \setminus X)$. The marginal posteriors are especially useful for approximate MAP solvers [Park, 2002,

Park and Darwiche, 2004]. Given a compact representation of the NP, e.g. using an arithmetic circuit, *all* derivatives of the form (4) can be computed *simultaneously* using back-propagation, i.e. using a single upward and downward pass in the circuit.

3 FINITE STATE SPNS

For some node N in an *acyclic directed graph*, we denote the set of parents and children as $\mathbf{pa}(\mathsf{N})$ and $\mathbf{ch}(\mathsf{N})$, respectively. The descendants $\mathbf{desc}(\mathsf{N})$ are recursively defined as the set containing N and the children of any descendant. We now define SPNs over RVs with finitely many states [Poon and Domingos, 2011]:

Definition 2 (Sum-Product Network)**.** *Let $\mathbf{X}$ be a set of RVs with finitely many states and $\boldsymbol{\lambda}$ their IVs. A sum-product network $\mathcal{S} = (\mathcal{G}, \boldsymbol{w})$ over $\mathbf{X}$ is a rooted acyclic directed graph $\mathcal{G} = (V, E)$ and a set of nonnegative parameters $\boldsymbol{w}$. All* leaves *of $\mathcal{G}$ are IVs and all* internal *nodes are either sums or products. A sum node S computes a weighted sum $\mathsf{S}(\boldsymbol{\lambda}) = \sum_{\mathsf{C}\in\mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}}\,\mathsf{C}(\boldsymbol{\lambda})$, where the weight $w_{\mathsf{S},\mathsf{C}} \in \boldsymbol{w}$ is associated with the edge $(\mathsf{S},\mathsf{C}) \in E$. A product node computes $\mathsf{P}(\boldsymbol{\lambda}) = \prod_{\mathsf{C}\in\mathbf{ch}(\mathsf{P})} \mathsf{C}(\boldsymbol{\lambda})$. The output of $\mathcal{S}$ is the function computed by the root node and denoted as $\mathcal{S}(\boldsymbol{\lambda})$. The number of* additions *and* multiplications *performed by SPN $\mathcal{S}$ are denoted as $A_{\mathcal{S}}$ and $M_{\mathcal{S}}$, respectively, and given as $A_{\mathcal{S}} = \sum_{\mathsf{S}\in V} |\mathbf{ch}(\mathsf{S})|$, $M_{\mathcal{S}} = \sum_{\mathsf{P}\in V} (|\mathbf{ch}(\mathsf{P})| - 1) + \sum_{\mathsf{S}\in V} |\mathbf{ch}(\mathsf{S})|$.*

We use symbols λ, S, P, N, C, F to refer to nodes in SPNs, where λ denotes an IV, S always refers to a sum and P always to a product. N, C, F refer to nodes without specifying their type, where N can be any node, C and F are used to highlight their roles as children and parents of other nodes, respectively.

The *scope* of N, denoted as $\mathbf{sc}(\mathsf{N})$, is defined as

$$\mathbf{sc}(\mathsf{N}) = \begin{cases} \{X\} & \text{if } \mathsf{N} \text{ is some IV } \lambda_{X=x} \\ \bigcup_{\mathsf{C}\in\mathbf{ch}(\mathsf{N})} \mathbf{sc}(\mathsf{C}) & \text{otherwise.} \end{cases} \tag{5}$$

For any N in an SPN, we define the *sub-SPN* $\mathcal{S}_{\mathsf{N}} = (\mathcal{G}_{\mathsf{N}}, \boldsymbol{w}_{\mathsf{N}})$, resulting from the graph $\mathcal{G}_{\mathsf{N}}$ induced by $\mathbf{desc}(\mathsf{N})$ together with the corresponding weights $\boldsymbol{w}_{\mathsf{N}}$.

The nodes in an SPN are functions over the IVs $\boldsymbol{\lambda}$. Using (2) and (3), we define $\mathsf{N}(\mathbf{x}) := \mathsf{N}(\boldsymbol{\lambda}(\mathbf{x}))$ and $\mathsf{N}(\boldsymbol{\mathcal{X}}) := \mathsf{N}(\boldsymbol{\lambda}(\boldsymbol{\mathcal{X}}))$. To avoid pathological cases, we assume that for each N there exist $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ such that $\mathsf{N}(\mathbf{x}) > 0$. The SPN *distribution* is defined as [Poon and Domingos, 2011]:

Definition 3 (SPN Distribution)**.** *Let $\mathcal{S}$ be an SPN over $\mathbf{X}$. The* distribution *represented by $\mathcal{S}$ is*

$$P_{\mathcal{S}}(\mathbf{x}) := \frac{\mathcal{S}(\mathbf{x})}{\sum_{\mathbf{x}'\in\mathbf{val}(\mathbf{X})} \mathcal{S}(\mathbf{x}')}. \tag{6}$$

Using sub-SPNs, we associate to each node N a distribution $P_{\mathsf{N}} := P_{\mathcal{S}_{\mathsf{N}}}$ over $\mathbf{sc}(\mathsf{N})$.

Inference in structurally unconstrained SPNs is generally intractable. Therefore, Poon and Domingos [2011] introduce the notion of *validity*:

Definition 4 (Valid SPN)**.** *An SPN $\mathcal{S}$ over $\mathbf{X}$ is* valid *if for each $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$*

$$\sum_{\mathbf{x}\in\boldsymbol{\mathcal{X}}} P_{\mathcal{S}}(\mathbf{x}) = \frac{\mathcal{S}(\boldsymbol{\mathcal{X}})}{\sum_{\mathbf{x}'\in\mathbf{val}(\mathbf{X})} \mathcal{S}(\mathbf{x}')}. \tag{7}$$

A valid SPN performs marginalization in similar manner as an NP, cf. Section 2. Two conditions are given for guaranteeing validity of an SPN, *completeness* and *consistency*, which are defined as follows:

Definition 5 (Completeness)**.** *A sum node S in SPN $\mathcal{S}$ is* complete *if $\mathbf{sc}(\mathsf{C}') = \mathbf{sc}(\mathsf{C}'')$, $\forall \mathsf{C}', \mathsf{C}'' \in \mathbf{ch}(\mathsf{S})$. $\mathcal{S}$ is* complete *if every sum node in $\mathcal{S}$ is complete.*

Definition 6 (Consistency)**.** *A product node P in SPN $\mathcal{S}$ is* consistent *if $\forall \mathsf{C}', \mathsf{C}'' \in \mathbf{ch}(\mathsf{P}), \mathsf{C}' \neq \mathsf{C}''$, it holds that $\lambda_{X=x} \in \mathbf{desc}(\mathsf{C}') \Rightarrow \forall x' \neq x : \lambda_{X=x'} \notin \mathbf{desc}(\mathsf{C}'')$. $\mathcal{S}$ is* consistent *if every product node in $\mathcal{S}$ is consistent.*

Thus, for complete and consistent SPNs, we can perform arbitrary marginalization tasks of the form (7) with computational cost linear in the network size. A more restrictive condition implying consistency is *decomposability*:

Definition 7 (Decomposability)**.** *A product node P in SPN $\mathcal{S}$ is* decomposable *if $\forall \mathsf{C}', \mathsf{C}'' \in \mathbf{ch}(\mathsf{P}), \mathsf{C}' \neq \mathsf{C}''$, it holds that $\mathbf{sc}(\mathsf{C}') \cap \mathbf{sc}(\mathsf{C}'') = \emptyset$. $\mathcal{S}$ is* decomposable *if every product node in $\mathcal{S}$ is decomposable.*

Decomposability requires that the scopes of a product node's children do not overlap, while the notion of consistency is somewhat harder to grasp. The following definition and proposition provide an equivalent condition to consistency.

Definition 8 (Shared RVs)**.** *The* shared RVs *of some product node P are defined as*

$$\mathbf{Y} = \bigcup_{\substack{\mathsf{C}',\mathsf{C}''\in\mathbf{ch}(\mathsf{P}) \\ \mathsf{C}'\neq\mathsf{C}''}} \mathbf{sc}(\mathsf{C}') \cap \mathbf{sc}(\mathsf{C}''), \tag{8}$$

i.e. the part of $\mathbf{sc}(\mathsf{P})$ shared by at least two children.

Proposition 1. *Let P be a product node and $\mathbf{Y}$ be its shared RVs. P is consistent iff for each $Y \in \mathbf{Y}$ there exists a* unique *$y^* \in \mathbf{val}(Y)$ with $\lambda_{Y=y^*} \in \mathbf{desc}(\mathsf{P})$.*

We call y^* the *consistent state of the shared RV Y*, and collect all y^* in $\mathbf{y}^* \in \mathbf{val}(\mathbf{Y})$. The following theorem shows that the distribution represented by a consistent product is *deterministic with respect to the shared RVs.*

Furthermore, the distributions represented by descendants of this product are deterministic with respect to the RVs which overlap with $\mathbf{Y}$.

Theorem 1. *Let $\mathcal{S}$ be a complete and consistent SPN and P be a non-decomposable product in $\mathcal{S}$, $\mathbf{Y}$ be the shared RVs of P and $\mathbf{y}^*$ the consistent state of $\mathbf{Y}$. For $\mathsf{N} \in \mathbf{desc}(\mathsf{P})$ define $\mathbf{Y}^{\mathsf{N}} := \mathbf{Y} \cap \mathbf{sc}(\mathsf{N})$ and $\mathbf{X}^{\mathsf{N}} := \mathbf{sc}(\mathsf{N}) \setminus \mathbf{Y}^{\mathsf{N}}$. Then for all $\mathsf{N} \in \mathbf{desc}(\mathsf{P})$, and all $\mathbf{x} \in \mathbf{val}(\mathbf{sc}(\mathsf{N}))$:*

$$P_{\mathsf{N}}(\mathbf{x}) = \mathbb{1}\big(\mathbf{x}[\mathbf{Y}^{\mathsf{N}}] = \mathbf{y}^*[\mathbf{Y}^{\mathsf{N}}]\big)\ P_{\mathsf{N}}\left(\mathbf{x}[\mathbf{X}^{\mathsf{N}}]\right), \quad (9)$$

where $\mathbb{1}$ is the indicator function.

For the proof we require two lemmas.

Lemma 1. *Let N be a node in some complete and consistent SPN over $\mathbf{X}$, $X \in \mathbf{sc}(\mathsf{N})$ and $x \in \mathbf{val}(X)$. When $\lambda_{X=x} \notin \mathbf{desc}(\mathsf{N})$, then $\forall \mathbf{x} \in \mathbf{val}(\mathbf{X})$ with $\mathbf{x}[X] = x$ we have $\mathsf{N}(\mathbf{x}) = 0$.*

Lemma 2. *Let P be a probability mass function (PMF) over $\mathbf{X}$ and $\mathbf{Y} \subseteq \mathbf{X}$, $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$ such that there exists a $\mathbf{y}^* \in \mathbf{val}(\mathbf{Y})$ with $P(\mathbf{z}, \mathbf{y}) = 0$ when $\mathbf{y} \neq \mathbf{y}^*$. Then we have $P(\mathbf{z}, \mathbf{y}) = \mathbb{1}(\mathbf{y} = \mathbf{y}^*)\, P(\mathbf{z})$.*

Proof of Theorem 1. From Proposition 1 we know that $\forall Y \in \mathbf{Y}\colon \lambda_{Y=\mathbf{y}^*[Y]} \in \mathbf{desc}(\mathsf{P})$ and $\forall y \neq \mathbf{y}^*[Y]\colon \lambda_{Y=y} \notin \mathbf{desc}(\mathsf{P})$. Consequently, for any $\mathsf{N} \in \mathbf{desc}(\mathsf{P})$ we have for all $Y \in \mathbf{Y}^{\mathsf{N}}$ that $\lambda_{Y=\mathbf{y}^*[Y]} \in \mathbf{desc}(\mathsf{N})$ and $\forall y \neq \mathbf{y}^*[Y]\colon \lambda_{Y=y} \notin \mathbf{desc}(\mathsf{N})$. With Lemma 1 it follows that for all $\mathbf{x} \in \mathbf{val}(\mathbf{sc}(\mathsf{N}))$ with $\mathbf{x}[\mathbf{Y}^{\mathsf{N}}] \neq \mathbf{y}^*[\mathbf{Y}^{\mathsf{N}}]$, we have $P_{\mathsf{N}}(\mathbf{x}) = 0$. Theorem 1 follows with Lemma 2. □

Corollary 1. *Let P, $\mathbf{Y}$, $\mathbf{y}^*$ be as in Theorem 1. For $\mathsf{C} \in \mathbf{ch}(\mathsf{P})$, let $\mathbf{X}^{\mathsf{C}} := \mathbf{sc}(\mathsf{C}) \setminus \mathbf{Y}$, i.e. the part of $\mathbf{sc}(\mathsf{C})$ which belongs exclusively to C. Then*

$$P_{\mathsf{P}}(\mathbf{x}) = \mathbb{1}(\mathbf{x}[\mathbf{Y}] = \mathbf{y}^*) \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}\left(\mathbf{x}[\mathbf{X}^{\mathsf{C}}]\right). \quad (10)$$

A *decomposable* product has the intuitive interpretation of a distribution assuming independence among the scopes of its children. Corollary 1 shows that a consistent product assumes independence among the shared RVs $\mathbf{Y}$ and the RVs which are exclusive to P's children. Independence of $\mathbf{Y}$ stems from the fact that $\mathbf{Y}$ is deterministically set to the consistent state $\mathbf{y}^*$.

3.1 Locally Normalized SPNs

In this section we present our first main contribution. In valid SPNs, the normalization constant $\mathscr{Z}_{\mathcal{S}} = \sum_{\mathbf{x}} \mathcal{S}(\mathbf{x})$ can be determined efficiently by a single upwards pass, setting $\boldsymbol{\lambda} \equiv 1$. We call SPNs with $\mathscr{Z}_{\mathcal{S}} = 1$ *normalized* SPNs, for which we have $P_{\mathcal{S}}(\mathbf{x}) = \mathcal{S}(\mathbf{x})$. When the sum weights are normalized for each sum S, i.e. $\forall \mathsf{S}\colon \sum_{\mathsf{C} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}} = 1$, the SPN is automatically normalized, since i) IVs can be interpreted as distributions, ii) complete sum nodes with normalized weights are normalized if their children are normalized, and iii) consistent product nodes are normalized if their children are normalized, following from Corollary 1. We call such SPNs, whose weights are normalized for each sum, *locally normalized* SPNs. Clearly, any sub-SPN of a locally normalized SPNs is also locally normalized. Thus, for any complete, consistent and locally normalized SPN $\mathcal{S}$ and any N of $\mathcal{S}$, we have $\forall \mathbf{x} \in \mathbf{sc}(\mathsf{N})\colon P_{\mathsf{N}}(\mathbf{x}) = \mathcal{S}_{\mathsf{N}}(\mathbf{x})$. The question rises, if locally normalized SPNs are a weaker class of models than non-normalized SPNs, i.e. are there distributions which can be represented by an SPN with structure $\mathcal{G}$, but *not* by a locally normalized SPN sharing the same structure? The answer is no, as stated in the following theorem.

Theorem 2. *For each complete and consistent SPN $\mathcal{S}' = (\mathcal{G}', \boldsymbol{w}')$, there exists a complete, consistent and* locally normalized *SPN $\mathcal{S} = (\mathcal{G}, \boldsymbol{w})$ with $\mathcal{G}' = \mathcal{G}$, such that $\forall \mathsf{N} \in \mathcal{G}\colon \mathcal{S}_{\mathsf{N}} \equiv P_{\mathcal{S}'_{\mathsf{N}}}$.*

Algorithm 1 Locally Normalize SPN

```
Let N_1, ..., N_K be a topologically ordered list of all sum and product nodes
For all product nodes P initialize α_P ← 1
for k = 1 : K do
    if N_k is a sum node then
        α ← Σ_{C ∈ ch(N_k)} w_{N_k,C}
        ∀C ∈ ch(N_k): w_{N_k,C} ← w_{N_k,C} / α
    end if
    if N_k is a product node then
        α ← α_{N_k}
        α_{N_k} ← 1
    end if
    for F ∈ pa(N_k) do
        if F is a sum node then
            w_{F,N_k} ← α w_{F,N_k}
        end if
        if F is a product node then
            α_F ← α α_F
        end if
    end for
end for
```

Proof. Algorithm 1 finds locally normalized weights without changing the distribution of any node. For deriving the algorithm, we introduce a *nonnegative correction factor* α_{P} for each product node P, initialized to $\alpha_{\mathsf{P}} = 1$. We redefine the product node P as $\mathsf{P}(\boldsymbol{\lambda}) := \alpha_{\mathsf{P}} \mathsf{P}(\boldsymbol{\lambda})$. At the end of the algorithm, all α_{P} will be 1 and can effectively be removed.

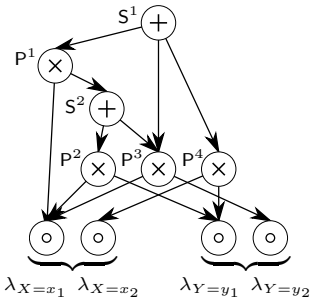

Figure 1: Example of a complete and consistent (but not decomposable) SPN over two binary RVs $\{X, Y\}$.

Let $\mathsf{N}'_1, \ldots, \mathsf{N}'_K$ be a topologically ordered list of all sums and products in the unnormalized SPN $\mathcal{S}'$, i.e. $\mathsf{N}'_k \notin \mathbf{desc}(\mathsf{N}'_l)$ if $k > l$. Let $\mathsf{N}_1, \ldots, \mathsf{N}_K$ be the corresponding list of $\mathcal{S}$, which will be the normalized version after the algorithm has terminated. We have the following loop invariant for the main loop. Given that at the k^{th} entrance of the main loop, i) $P_{\mathsf{N}'_l} = \mathcal{S}_{\mathsf{N}_l}$, for $1 \leq l < k$, and ii) $\mathcal{S}'_{\mathsf{N}'_m} = \mathcal{S}_{\mathsf{N}_m}$, for $k \leq m \leq K$, the same will hold for $k+1$ at the end of the loop.

Point i) holds since we normalize N_k during the main loop: All nodes prior in the topological order, and therefore all children of N_k are already normalized. If N_k is a sum, then it represents a mixture distribution after step 6. If N_k is a product, then it will be normalized after step 10, due to Corollary 1 and since we set $\alpha_{\mathsf{N}_k} = 1$. Point ii) holds since the modification of N_k can change any N_m, $m > k$, only via $\mathbf{pa}(\mathsf{N}_k)$. The change of N_k is compensated for all parents either in step 14 or step 17, depending on whether the parent is a sum or a product. From this loop invariance it follows by induction that all $\mathsf{N}_1, \ldots, \mathsf{N}_K$ compute the normalized distributions of $\mathsf{N}'_1, \ldots, \mathsf{N}'_K$ after the K^{th} iteration. □

Theorem 2 shows that from a representational point of view, we can always assume locally normalized SPNs. It might be advantageous during *learning* to assume the parameter space of non-normalized SPNs. However, after learning is completed we can use Algorithm 1 to find a set of locally normalized parameters.

3.2 Consistency vs. Decomposability

As already noted, consistency together with completeness guarantees validity of an SPN, i.e. we can perform efficient marginalization, similarly to NPs. However, a valid but non-decomposable SPN does in general not compute an NP, as the following example shows. The SPN depicted in Figure 1 is complete and consistent, but not decomposable, since P^1 is not decomposable. It computes the function

$$\begin{aligned}\mathcal{S}(\boldsymbol{\lambda}) = \; & w_{\mathsf{S}^1,\mathsf{P}^1}\, w_{\mathsf{S}^2,\mathsf{P}^2}\, \lambda^2_{X=x_1}\, \lambda_{Y=y_1} \\ & + w_{\mathsf{S}^1,\mathsf{P}^1}\, w_{\mathsf{S}^2,\mathsf{P}^3}\, \lambda^2_{X=x_1}\, \lambda_{Y=y_2} \\ & + w_{\mathsf{S}^1,\mathsf{P}^3}\, \lambda_{X=x_1}\, \lambda_{Y=y_2} \\ & + w_{\mathsf{S}^1,\mathsf{P}^4}\, \lambda_{X=x_2}\, \lambda_{Y=y_1},\end{aligned} \tag{11}$$

which is clearly no NP since $\lambda_{X=x_1}$ is raised to the power of 2 in two terms. This generally happens in non-decomposable SPNs and as a consequence, *Darwiche's differential approach is not applicable to consistent SPNs.* For example, consider the derivative

$$\begin{aligned}\frac{\partial \mathcal{S}}{\partial \lambda_{X=x_1}}(\boldsymbol{\lambda}) = \; & 2\, w_{\mathsf{S}^1,\mathsf{P}^1}\, w_{\mathsf{S}^2,\mathsf{P}^2}\, \lambda_{X=x_1}\, \lambda_{Y=y_1} \\ & + 2\, w_{\mathsf{S}^1,\mathsf{P}^1}\, w_{\mathsf{S}^2,\mathsf{P}^3}\, \lambda_{X=x_1}\, \lambda_{Y=y_2} \\ & + w_{\mathsf{S}^1,\mathsf{P}^3}\, \lambda_{Y=y_2},\end{aligned} \tag{12}$$

which does *not* have the probabilistic interpretation of evaluation of modified evidence, such as the derivatives of an NP, cf. (4). A complete and consistent, but non-decomposable SPN is valid and thus identical to some NP on the *domain of binary vectors*. However, when taking the *derivative*, intuitively speaking, we also consider a small ϵ-environment around the 0's and 1's, corrupting the differential approach.

However, complete and *decomposable* SPNs compute NPs and are thus amenable for the differential approach. This is stated in the following proposition.

Proposition 2. *A complete and decomposable SPN computes the NP of some unnormalized distribution.*

Therefore, when evaluations of modified evidence (4) are required, we should use decomposable SPNs instead of consistent SPNs. When these are not required, we might want to use the more general condition of consistency instead of decomposability, since we could possibly represent distributions more compactly, i.e. using smaller networks and fewer arithmetic operations. However, the following theorem shows that this potential saving is relatively modest.

Theorem 3. *Every complete and consistent SPN $\mathcal{S} = ((V, E), \boldsymbol{w})$ over $\mathbf{X}$ can be transformed into a complete and decomposable SPN $\mathcal{S}' = ((V', E'), \boldsymbol{w}')$ over $\mathbf{X}$ such that $P_{\mathcal{S}} \equiv P_{\mathcal{S}'}$, and where $|V'| \in \mathcal{O}(|V|^2)$, $A_{\mathcal{S}'} = A_{\mathcal{S}}$ and $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}}\, |\mathbf{X}|)$.*

Proof. Due to Theorem 2 we assume w.l.o.g. that $\mathcal{S}'$ is locally normalized, and thus $P_{\mathcal{S}} \equiv \mathcal{S}$. Algorithm 2 transforms $\mathcal{S}$ into a complete and decomposable SPN, representing the same distribution. First it finds a topologically ordered list $\mathsf{N}_1, \ldots, \mathsf{N}_K$ of all sum and product nodes, i.e. $k > l \Rightarrow \mathsf{N}_k \notin \mathbf{desc}(\mathsf{N}_l)$. Then, in steps 2–7, it considers all sum nodes S and all children

$\mathsf{C} \in \mathbf{ch}(\mathsf{S})$; if the child C has further parents except S, a newly generated product node $\mathsf{P}^{\mathsf{S},\mathsf{C}}$ is interconnected between S and C, i.e. $\mathsf{P}^{\mathsf{S},\mathsf{C}}$ is connected as child of S with weight $w_{\mathsf{S},\mathsf{C}}$, C is disconnected from S and connected as child of $\mathsf{P}^{\mathsf{S},\mathsf{C}}$. To $\mathsf{P}^{\mathsf{S},\mathsf{C}}$ we refer as *link* between S and C. Note that the link has *only* S as parent, i.e. the link represents a *private* copy of child C for sum node S. Clearly, after step 7, the SPN still computes the same function.

In each iteration of the main loop 8–26, the algorithm finds the lowest non-decomposable product node $\mathsf{N}_k = \mathsf{P}$ w.r.t. the topological ordering. We distinguish two cases: $\mathbf{sc}(\mathsf{P}) = \mathbf{Y}$ and $\mathbf{sc}(\mathsf{P}) \neq \mathbf{Y} \Leftrightarrow \mathbf{Y} \subset \mathbf{sc}(\mathsf{P})$.

In the first case, we know from Corollary 1 that $\mathsf{P}(\mathbf{y}) = \mathbb{1}(\mathbf{y} = \mathbf{y}^*)$, which is equivalent to the decomposable product $\prod_{Y \in \mathbf{Y}} \lambda_{Y=\mathbf{y}^*[Y]}$ replacing P, i.e. this new product is connected as child of all parents of P, and P itself is deleted. Deletion of P might render some nodes unreachable; however, these unreachable nodes do not "influence" the root node and will be safely deleted in step 27.

In the second case, when $\mathbf{Y} \subset \mathbf{sc}(\mathsf{P})$, the algorithm first finds the set $\mathbf{N}^d$ of all sum and product descendants of P. It also finds the subset $\mathbf{N}^o$ of $\mathbf{N}^d$, containing all nodes whose scope overlaps with $\mathbf{Y}$, but is no subset of $\mathbf{Y}$. Clearly, P is contained in $\mathbf{N}^o$. The basic strategy is to "cut" $\mathbf{Y}$ from the scope of P, i.e. that $\mathbf{Y}$ is marginalized, rendering P decomposable. Then, by re-connecting all indicators $\lambda_{Y=\mathbf{y}^*[Y]}$ to P in step 24, P computes the same distribution as before due to Corollary 1, but is rendered *decomposable* now. Steps 21–23 cut $\mathbf{Y}$ from *all* nodes in $\mathbf{N}^o$, in particular from P, but leave all sub-SPNs rooted at any node in $\mathbf{N}^d \setminus \mathbf{N}^o$ unchanged. This can be shown by induction over a topological ordering of $\mathbf{N}^o$. Due to space reasons, we omit this sub-proof here, but provide it in the supplementary. Although we achieve our primary goal to render P decomposable, steps 21–23 also cut $\mathbf{Y}$ from any other node in $\mathsf{N} \in \mathbf{N}^o$, which would modify the SPN output via N's parents *outside* of $\mathbf{N}^d$, i.e. via $\mathbf{F} = \mathbf{pa}(\mathsf{N}) \setminus \mathbf{N}^d$. Note that all nodes in $\mathbf{F}$ must be products. To see this, assume that $\mathbf{F}$ contains a sum S. This would imply that N is a link, which can reached from P only via its single parent S. This implies $\mathsf{S} \in \mathbf{N}^d$, a contradiction. By Theorem 1, the distribution of N is deterministic w.r.t. $\mathbf{sc}(\mathsf{N}) \cap \mathbf{Y}$. Steps 21–23 cut $\mathbf{Y}$ from N (see supplementary), which could change the distribution of the nodes in $\mathbf{F}$. Thus, in step 19 the IVs corresponding to $\mathbf{Y} \cap \mathbf{sc}(\mathsf{N})$ are connected to all $\mathbf{F}$, such that they still "see" the same distribution after steps 21–23. It is easy to see that if some $\mathsf{F} \in \mathbf{F}$ was decomposable beforehand, it will also be decomposable after step 23, i.e. steps 15–24 do not render other products non-decomposable. Thus, in each iteration, one non-decomposable product is rendered decomposable, without changing the SPN distribution.

Algorithm 2 Transform to decomposable SPN

1: Let $\mathbf{N} = \mathsf{N}_1, \ldots, \mathsf{N}_K$ be a topologically ordered list of all sums and products
2: **for** all sum nodes S and all $\mathsf{C} \in \mathbf{ch}(\mathsf{S})$ **do**
3: **if** $\mathbf{pa}(\mathsf{C}) > 1$ **then**
4: Generate a new product node $\mathsf{P}^{\mathsf{S},\mathsf{C}}$
5: Interconnect $\mathsf{P}^{\mathsf{S},\mathsf{C}}$ between S and C
6: **end if**
7: **end for**
8: **while** exist non-decomposable products in $\mathbf{N}$ **do**
9: $\mathsf{P} \leftarrow \mathsf{N}_{\min\{k' \mid \mathsf{N}_{k'} \text{ is a non-decomposable product}\}}$
10: $\mathbf{Y} \leftarrow$ shared RVs of P
11: $\mathbf{y}^* \leftarrow$ consistent state of $\mathbf{Y}$
12: **if** $\mathbf{sc}(\mathsf{P}) = \mathbf{Y}$ **then**
13: Replace P by $\prod_{Y \in \mathbf{Y}} \lambda_{Y=\mathbf{y}^*[Y]}$
14: **else**
15: $\mathbf{N}^d \leftarrow$ sums and products in $\mathbf{desc}(\mathsf{P})$
16: $\mathbf{N}^o \leftarrow \{\mathsf{N} \in \mathbf{N}^d : \mathbf{sc}(\mathsf{N}) \not\subseteq \mathbf{Y}, \mathbf{sc}(\mathsf{N}) \cap \mathbf{Y} \neq \emptyset\}$
17: **for** $\mathsf{N} \in \mathbf{N}^o \setminus \{\mathsf{P}\}$ **do**
18: $\mathbf{F} \leftarrow \mathbf{pa}(\mathsf{N}) \setminus \mathbf{N}^d$
19: $\forall Y \in \mathbf{Y} \cap \mathbf{sc}(\mathsf{N})$: connect $\lambda_{Y=\mathbf{y}^*[Y]}$ as child of all $\mathbf{F}$
20: **end for**
21: **for** $\mathsf{P}^o \in \mathbf{N}^o$ **do**
22: Disconnect $\mathsf{C} \in \mathbf{ch}(\mathsf{P}^o)$ if $\mathbf{sc}(\mathsf{C}) \subseteq \mathbf{Y}$
23: **end for**
24: $\forall Y \in \mathbf{Y}$: connect $\lambda_{Y=\mathbf{y}^*[Y]}$ as child of P
25: **end if**
26: **end while**
27: Delete all unreachable sums and products

Since the only new introduced nodes are the links between sum nodes and their children, and the number of sum-edges is in $\mathcal{O}(|V|^2)$, we have $|V'| \in \mathcal{O}(|V|^2)$. The number of summations is the same in $\mathcal{S}$ and $\mathcal{S}'$, i.e. $A_{\mathcal{S}'} = A_{\mathcal{S}}$. Furthermore, introducing the links cannot introduce more than twice the number of multiplications, since we already require one multiplication per sum-edge. Thus, after step 7 we have $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}})$. Since the while-loop in Algorithm 2 cannot connect more than one IV per $X \in \mathbf{X}$ to each product node, we have $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}} |\mathbf{X}|)$. □

A graphical example of Algorithm 2 is given in the supplementary. We see that any complete and consistent SPN $\mathcal{S}$ can be transformed into a complete and decomposable SPN $\mathcal{S}'$ with the same number of addition operations, and whose number of multiplication operations grows at most linearly with $M_{\mathcal{S}} |\mathbf{X}|$. Any distribution which can be encoded by a consistent SPN using polynomially many arithmetic operations in $|\mathbf{X}|$, can also be polynomially encoded by a decomposable

SPN. Consequently, the class of consistent SPNs is *not exponentially* more compact than the class of decomposable SPNs.

Furthermore, Algorithm 2 shows that using consistent but non-decomposable products is actually *wasteful* for a particular sub-class of SPNs we denote as *sum-product trees*:

Definition 9 (Sum-Product Tree)**.** *A sum-product tree (SPT) is an SPN where each sum and product has at most one parent.*

Proposition 3. *Every complete and consistent, but non-decomposable SPT $\mathcal{S} = ((V, E), \boldsymbol{w})$ over $\mathbf{X}$ can be transformed into a complete and decomposable SPT $\mathcal{S}' = ((V', E'), \boldsymbol{w}')$ over $\mathbf{X}$ such that $P_{\mathcal{S}} \equiv P_{\mathcal{S}'}$, and where $|V'| \leq |V|$, $A_{\mathcal{S}'} \leq A_{\mathcal{S}}$ and $M_{\mathcal{S}'} < M_{\mathcal{S}}$.*

The proof and an example of applying Algorithm 2 to an SPT is given in the supplementary.

4 GENERALIZED SPNS

So far, we considered SPNs over finite states using IVs. However, as pointed out in [Gens and Domingos, 2013, Peharz et al., 2013, Rooshenas and Lowd, 2014], SPNs can be generalized by replacing the IVs with arbitrary distributions over arbitrary large scopes. From now on, we assume that each RV is either continuous or discrete, where the latter can also have countably infinitely many states (e.g. Poisson distributed). PMFs and PDFs can be mixed within the same distribution: For example, consider a binary RV X with $\mathbf{val}(X) = \{x_1, x_2\}$ and a continuous RV Y with $\mathbf{val}(Y) = \mathbb{R}$. The function

$$\begin{aligned}\mathsf{D}_{\{X,Y\}}(x, y) = \; & 0.4 \times \mathbb{1}(x = x_1) \times \mathcal{N}(y; \mu_1, \sigma_1) \\ & + 0.6 \times \mathbb{1}(x = x_2) \times \mathcal{N}(y; \mu_2, \sigma_2),\end{aligned} \tag{13}$$

is a PMF with respect to X and a PDF with respect to Y. We refer to $\mathsf{D}_{\mathbf{Y}}$ as distribution nodes or simply as distributions. We define generalized SPNs as in Section 3, but now using distributions as leaves. The scope is now defined as

$$\mathbf{sc}(\mathsf{N}) = \begin{cases} \mathbf{Y} & \text{if } \mathsf{N} \text{ is some } \mathsf{D}_{\mathbf{Y}} \\ \bigcup_{\mathsf{C} \in \mathbf{ch}(\mathsf{N})} \mathbf{sc}(\mathsf{C}) & \text{otherwise.} \end{cases} \tag{14}$$

We require that all sums are complete and all products are decomposable. Note that we do allow *heterogeneous* scopes of the input distributions, i.e. when we have a $\mathsf{D}_{\mathbf{Y}}$ with $Y \in \mathbf{Y}$, we can have another $\mathsf{D}_{\mathbf{Y}'}$ with $Y \in \mathbf{Y}'$ and $\mathbf{Y} \neq \mathbf{Y}'$. In the most extreme case we can have distributions $\mathsf{D}_{\mathbf{Y}}$ for *all* possible subsets of $\mathbf{Y} \subseteq \mathbf{X}$, as long as completeness and decomposability hold. It is easily verified that Theorem 2 also holds for generalized SPNs, so w.l.o.g. we assume locally normalized weights. In generalized SPNs, each node clearly represents a distribution over its scope: i) leaves are distributions per definition, ii) products are distributions assuming independence, and iii) sums are mixture distributions.

In Section 3, we saw that for finite state SPNs we can efficiently evaluate

1. $\mathcal{S}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ (probability of sample).
2. $\mathcal{S}(\boldsymbol{\mathcal{X}})$ for $\boldsymbol{\mathcal{X}} \in \mathcal{H}_{\mathbf{X}}$ (marginalization).
3. $\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X])$, simultaneously for all $X \in \mathbf{X}$ (modified evidence).

We want to have the same inference mechanisms for generalized SPNs. (1.) Evaluation of $\mathcal{S}(\mathbf{x})$ clearly works in the same way as for SPNs over finite states. (2.) Marginalization also works in a similar way, for which we need to compute

$$\mathcal{S}(\boldsymbol{\mathcal{X}}) = \int_{x_1 \in \mathcal{X}_1} \dots \int_{x_N \in \mathcal{X}_N} \mathcal{S}(x_1, \dots, x_N)\, dx_1 \dots dx_N, \tag{15}$$

where integrals have to be replaced by sums for discrete RVs. Given that integrals can be computed efficiently for the input distributions, the integral (15) can be evaluated easily, since we can pull the integrals over all sums and products down to the input distributions: At sum nodes, we can interchange the integrals with the sum, due to completeness. At product nodes, we can interchange the integrals with the product, due to decomposability, i.e. since the factors are functions over non-overlapping variable sets. Therefore, we simply perform marginalization at the input distributions (over the respective scopes), and evaluate sums and products in the usual way. Note that in the case of finitely many states and using IVs, this precisely reproduces the mapping (3).

(3.) For evaluation of modified evidence $\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X])$ we develop an extension of the differential approach. Let $\mathbf{D}^X$ be the set of input distributions which have X in their scope. We assume an arbitrary fixed ordering of $\mathbf{D}^X = \{\mathsf{D}^{X,1}, \dots, \mathsf{D}^{X,|\mathbf{D}^X|}\}$ and define $[\mathsf{D}]^X = k$ if $\mathsf{D} = \mathsf{D}^{X,k}$. Note that the sets $\mathbf{D}^X$ do overlap, since for $\mathsf{D}_{\mathbf{Y}}$ we have $\forall X \in \mathbf{Y} \colon \mathsf{D}_{\mathbf{Y}} \in \mathbf{D}^X$. For each X, we introduce a latent RV Z_X with $\mathbf{val}(Z_X) = \{1, \dots, |\mathbf{D}^X|\}$. We denote these RVs as *distribution selectors* (DS) and let $\mathbf{Z} = \{Z_X \mid X \in \mathbf{X}\}$ be the set of all DS. Now we replace each distribution node $\mathsf{D}_{\mathbf{Y}}$ in the SPN by a product node:

$$\mathsf{D}_{\mathbf{Y}} \to \mathsf{D}_{\mathbf{Y}} \times \prod_{X \in \mathbf{Y}} \lambda_{Z_X = [\mathsf{D}_{\mathbf{Y}}]^X}, \tag{16}$$

and denote the result as *gated* SPN $\mathcal{S}^g$. An example of this process is shown in Figure 2. It is easy to see that

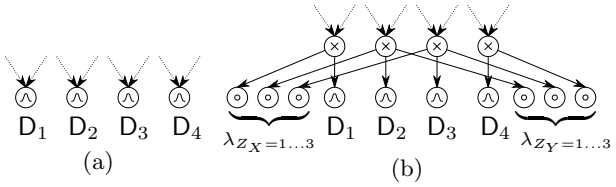

Figure 2: Construction of a gated SPN. We assume that $\mathbf{sc}(\mathsf{D}_1) = \{X\}$, $\mathbf{sc}(\mathsf{D}_2) = \mathbf{sc}(\mathsf{D}_3) = \{X, Y\}$ and $\mathbf{sc}(\mathsf{D}_4) = \{Y\}$. (a): Excerpt showing the distribution nodes over X and Y. (b): Same excerpt with IVs $\lambda_{Z_X=1}, \lambda_{Z_X=2}, \lambda_{Z_X=3}, \lambda_{Z_Y=1}, \lambda_{Z_Y=2}, \lambda_{Z_Y=3}$ of DSs.

the gated SPN is a complete and decomposable SPN and thus represents a distribution over $\mathbf{X} \cup \mathbf{Z}$. Furthermore, we have $\mathcal{S}(\mathbf{X}) = \mathcal{S}^g(\mathbf{X}) = \sum_{\mathbf{z} \in \mathbf{val}(\mathbf{Z})} \mathcal{S}^g(\mathbf{X}, \mathbf{z})$, since when the DS are marginalized, i.e. $\boldsymbol{\lambda}_{\mathbf{Z}} \equiv 1$, the introduced products simply copy the original distribution nodes. The following proposition establishes a conditional factorization property of the gated SPN.

Proposition 4. *Let $\mathcal{S}^g$ be a gated SPN. For any $X \in \mathbf{X}$ and $k \in \{1, \ldots, |\mathbf{D}^X|\}$, let $\mathbf{X}^D = \mathbf{sc}(\mathsf{D}^{X,k})$ and $\mathbf{X}^R = \mathbf{X} \setminus \mathbf{X}^D$. It holds that $\mathcal{S}^g(\mathbf{X}, Z_X = k, \mathbf{Z} \setminus Z_X) = \mathsf{D}^{X,k}(\mathbf{X}^D)\, \mathcal{S}^g(\mathbf{X}^R, Z_X = k, \mathbf{Z} \setminus Z_X)$.*

We define a modified notion of network polynomial, the *extended network polynomial* (ENP).

Definition 10 (Extended Network Polynomial)**.** *Let $P(\mathbf{X}, \mathbf{Z})$ be a probability distribution over RVs $\mathbf{X}$, $\mathbf{Z}$, where $\mathbf{Z}$ have finitely many states. The* extended network polynomial *f^e_P is defined as $f^e_P(\mathbf{X}, \boldsymbol{\lambda}) = \sum_{\mathbf{z} \in \mathbf{val}(\mathbf{Z})} P(\mathbf{X}, \mathbf{z}) \prod_{Z \in \mathbf{Z}} \lambda_{Z=\mathbf{z}[Z]}$.*

The ENP is an NP over only a part of the RVs, namely $\mathbf{Z}$. Similar as for the NP, we can use it to evaluate and marginalize over $\mathbf{Z}$ and apply the differential approach. Now, using a modification of the proof for Theorem 2, one can show that a gated SPN $\mathcal{S}^g$ over model RVs $\mathbf{X}$ and DS $\mathbf{Z}$ computes the ENP $f^e_{\mathcal{S}^g}(\mathbf{X}, \boldsymbol{\lambda})$.

We are now ready to derive the differential approach for generalized SPNs, i.e. to evaluate $\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X])$ for all X simultaneously. Using Proposition 4, we have

$$\begin{aligned}
\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]) &= \sum_{\mathbf{z}} \mathcal{S}^g(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X], \mathbf{z}) \\
&= \sum_{k=1}^{|\mathbf{D}^X|} \mathsf{D}^{X,k}(X, \boldsymbol{\mathcal{X}}[\mathbf{X}^D \setminus X]) \sum_{\mathbf{z}'} \mathcal{S}^g(\boldsymbol{\mathcal{X}}[\mathbf{X}^R], Z_X = k, \mathbf{z}') \\
&= \sum_{k=1}^{|\mathbf{D}^X|} \mathsf{D}^{X,k}(X, \boldsymbol{\mathcal{X}}[\mathbf{X}^D \setminus X])\, \mathcal{S}^g(\boldsymbol{\mathcal{X}}[\mathbf{X}^R], Z_X = k),
\end{aligned} \tag{17}$$

where the sets $\mathbf{X}^D$ and $\mathbf{X}^R$, depending on X and k, are defined as in Proposition 4 and $\mathbf{z}'$ runs over $\mathbf{val}(\mathbf{Z} \setminus Z_X)$. We see that $\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X])$ is computed by a linear combination of the input distributions in $\mathbf{D}^X$, evaluated for modified evidence $\boldsymbol{\mathcal{X}}[\mathbf{X}^D \setminus X]$ weighted by factors $\mathcal{S}^g(\boldsymbol{\mathcal{X}}[\mathbf{X}^R], Z_X = k)$. These factors are obtained using the differential approach. When $\mathbf{Z}$ is marginalized, i.e. when $\boldsymbol{\lambda}_{\mathbf{Z}} \equiv 1$, the differential approach tells us that

$$\mathcal{S}^g(\mathbf{X}, Z_X = k) = \frac{\partial \mathcal{S}^g}{\partial \lambda_{Z_X=k}} = \frac{\partial \mathcal{S}^g}{\partial \mathsf{P}}\, \mathsf{D}^{X,k}(\mathbf{X}^D), \tag{18}$$

where P is the parent product of $\lambda_{Z_X=k}$ and $\mathsf{D}^{X,k}$ in $\mathcal{S}^g$ (see Figure 2). Note that $\frac{\partial \mathcal{S}^g}{\partial \mathsf{P}}$ equals $\frac{\partial \mathcal{S}}{\partial \mathsf{D}^{X,k}}$, since the structure above $\mathsf{D}^{X,k}$ in $\mathcal{S}$ is identical to the structure in $\mathcal{S}^g$ above P. Comparing (17) and (18) we note that $\mathcal{S}^g(\boldsymbol{\mathcal{X}}[\mathbf{X}^R], Z_X = k) = \frac{\partial \mathcal{S}^g}{\partial \mathsf{P}} = \frac{\partial \mathcal{S}}{\partial \mathsf{D}^{X,k}}$, yielding

$$\mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]) = \sum_{k=1}^{|\mathbf{D}^X|} \mathsf{D}^{X,k}(X, \boldsymbol{\mathcal{X}}[\mathbf{X}^D \setminus X])\, \frac{\partial \mathcal{S}}{\partial \mathsf{D}^{X,k}}.$$

Inference scenario (3.) is summarized in Algorithm 3. Note, that although we used gated SPNs and ENPs for deriving this result, the required quantities can be computed in the original non-gated SPN. Note that Algorithm 3 reproduces the classical differential approach when considering SPNs over RVs with finitely many states and using IVs as distributed nodes.

Algorithm 3 Infer marginal posteriors

1: Evaluate input distributions for evidence $\boldsymbol{\mathcal{X}}$
2: Evaluate all sums and products (upwards pass)
3: For all nodes N, compute $\frac{\partial \mathcal{S}}{\partial \mathsf{N}}$ (backpropagation)
4: $\forall X, k$ let $f^k(X) = \mathsf{D}^{X,k}(X, \boldsymbol{\mathcal{X}}[\mathbf{sc}(\mathsf{D}^{X,k}) \setminus X])$
5: $\forall X : \mathcal{S}(X, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]) \leftarrow \sum_{k=1}^{|\mathbf{D}^X|} \frac{\partial \mathcal{S}}{\partial \mathsf{D}^{X,k}} f^k(X)$

5 CONCLUSION

In this paper, we aimed to develop a deeper understanding of SPNs and summarize our main results. We now summarize our main results. First, we do not lose any model power when we assume that the weights of SPNs are locally normalized, i.e. normalized for each sum node. Second, we do not lose much model power when we prefer the simpler condition of decomposability over consistency. Third, the inference scenarios known from network polynomials and Darwiche's differential approach can be applied in a similar way to general SPNs, i.e. SPNs with more or less arbitrary input distributions.

Acknowledgements

This work was supported by the Austrian Science Fund (project number P25244-N15).

References

M. R. Amer and S. Todorovic. Sum-Product Networks for Modeling Activities with Stochastic Structure. In *Proceedings of CVPR*, pages 1314 – 1321, 2012.

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.

A. Darwiche. A logical approach to factoring belief networks. In *Proceedings of KR*, pages 409–420, 2002.

A. Darwiche. A Differential Approach to Inference in Bayesian Networks. *ACM*, 50(3):280–305, 2003.

A. Dennis and D. Ventura. Learning the Architecture of Sum-Product Networks Using Clustering on Variables. In *Advances in Neural Information Processing Systems 25*, pages 2042–2050, 2012.

R. Gens and P. Domingos. Discriminative Learning of Sum-Product Networks. In *Proceedings of NIPS*, pages 3248–3256, 2012.

R. Gens and P. Domingos. Learning the Structure of Sum-Product Networks. *Proceedings of ICML*, pages 873–880, 2013.

D. Lowd and P. Domingos. Learning Arithmetic Circuits. In *Proceedings of UAI*, pages 383–392, 2008.

D. Lowd and A. Rooshenas. Learning Markov Networks with Arithmetic Circuits. *Proceedings of AISTATS*, pages 406–414, 2013.

J. D. Park. MAP Complexity Results and Approximation Methods. In *Proceedings of UAI*, pages 338–396, 2002.

J. D. Park and A. Darwiche. Complexity Results and Approximation Strategies for MAP Explanations . *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

R. Peharz, B. Geiger, and F. Pernkopf. Greedy Part-Wise Learning of Sum-Product Networks. In *ECML/PKDD*, volume 8189, pages 612–627. Springer Berlin, 2013.

R. Peharz, G. Kapeller, P. Mowlaee, and F. Pernkopf. Modeling Speech with Sum-Product Networks: Application to Bandwidth Extension. In *Proceedings of ICASSP*, 2014.

H. Poon and P. Domingos. Sum-Product Networks: A New Deep Architecture. In *Proceedings of UAI*, pages 337–346, 2011.

A. Rooshenas and D. Lowd. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. *ICML – JMLR W&CP*, 32:710–718, 2014.

Notes on the Revised Version

We would like to thank the anonymous reviewers and the meta-reviewer for their constructive comments. In what follows we respond to the reviewers' comments and note the changes in the revised paper.

Reviewer 2

The paper is difficult to read. I believe this is due to the complicated notation and lack of examples. Let me make some suggestions.

Consider Eq. 1. Lambda is subscripted by a series of three xs of different fonts. This is really difficult to interpret. For an assignment, I recommend using $X = x$ instead of X, x. For restricting a set to some subset, I recommend $\mathbf{x}[Y]$ instead of $\mathbf{x}_Y$.

We adopted the suggested notation.

Algorithm 2 is difficult to follow. Intuitively, I understand that what we are simply factoring the assignment $\mathbf{Y} = \mathbf{y}^*$ to a higher level in the tree. Steps 9-18 are quite technical and I couldnt follow the text that explains them. Can you give an example that shows a consistent tree followed by some intermediate trees corresponding to some steps of the algorithm with the final decomposable tree?

We modified Algorithm 2 and the proof of Theorem 3 to show more explicitly how the transformation of non-decomposable products into decomposable products takes place. In particular, all additional required nodes are introduced now in the beginning of the algorithm. We provide a graphical example illustrating Algorithm 2 in the supplementary, Figure 1.

I also have trouble seeing how the transformation from consistent to a decomposable tree leads to a blow up. Again, the transformation consists of factoring out the multiplication by $\mathbf{Y} = \mathbf{y}^*$ to a higher level in the tree, which should shrink it not make it bigger. Perhaps by introducing some examples it will become clear whether there is a blow up or not.

It is correct that in *trees* there is no blow-up. However, SPNs are not necessarily trees, i.e. all nodes can have *more* than one parent. The key difference to trees is that in non-trees, sub-SPNs can be re-used arbitrarily often. By factoring out $\mathbf{Y} = \mathbf{y}^*$ of the sub-SPN of the non-decomposable product, we would also modify other parts of the SPN via parents outside this sub-SPN. The example in the supplementary, Figure 1, illustrates this issue.

Additionally, we introduced Proposition 3, stating that in Sum-Product Trees non-decomposable products are indeed wasteful. We also give a graphical example for this Proposition in the supplementary, Figure 2.

The inference algorithm for generalized SPNs is described in words only and at some point it becomes difficult to follow. I recommend writing some pseudocode to clearly describe the algorithm.

We introduced pseudocode for inference in generalized SPNs in Algorithm 3.

Reviewer 4

Theorem 2: Is $\alpha_{\mathsf{P}} > 0$ (I assume so)? This should be called out explicitly.

In the proof of Theorem 2, we changed the sentence

"For deriving the algorithm, we introduce a *correction factor* α_{P} ..."

into

"For deriving the algorithm, we introduce a *nonnegative correction factor* α_{P} ..."

Algorithm 2: A figure would help greatly in parsing the algorithm.

We provide graphical examples illustrating Algorithm 2 in the supplementary, Figures 1 and 2. See also reply to Reviewer 2.

Section 4, part (2): The claim "this integral can be evaluated easily": the veracity of this claim is contingent on the choice of distributions used at the leaf nodes. This should be called out explicitly.

We changed the sentence

"This integral can be evaluated easily, ..."

into

"Given that integrals can be computed efficiently for the input distributions, the integral (15) can be evaluated easily, ..."

(p.1) "loose" → "lose"

Corrected.

Reviewer 5

I have to admit I don't have much confidence on whether I understood some key aspects: first, Equation (5) is the derivative obtained by taking lambda to be an arbitrary real number, I presume (discarding the

fact lambdas were defined to be binary).

We added some additional explanations about the IVs. Around Equation (4) we added

"It is important to note that, although the IVs are called *indicator* variables and set to values out of $\{0, 1\}$ by functions (2) and (3), they are in fact *real-valued* variables – this difference is essential when we take *derivatives* w.r.t. the IVs. These derivatives are used in the so-called *differential approach to inference* [Darwiche, 2003]. Taking the first derivative with respect to some $\lambda_{X=x}$ yields

$$\frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\boldsymbol{\mathcal{X}})) = \Phi(x, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]), \qquad (19)$$

since the derivatives of sum-terms in (1) with $\mathbf{x}[X] \neq x$ are 0, and the derivatives of sum-terms with $\mathbf{x}[X] = x$ are $\Phi(\mathbf{x}) \prod_{X' \in \mathbf{X} \setminus X} \lambda_{X'=\mathbf{x}[X']}$. Thus, the derivative in (4), which is indeed the standard derivative known from calculus, evaluates Φ for *modified evidence* $x, \boldsymbol{\mathcal{X}}[\mathbf{X} \setminus X]$ and is proportional to the *marginal posterior* $\Phi(x \mid \mathbf{X} \setminus X)$."

Much is made of Equation (15) as damning evidence against non-decomposable SPNs regarding the applicability of the differential approach: but I don't get it, as binary variables are idempotent regarding exponentiation - why should I care a binary variable is squared? It is possible I'm misunderstanding something, which might be related to the fact that the authors consider Equation (5) to be self-evident regardless of the fact that lambdas are defined to be binary. It might be obvious to those well-versed on SPNs and the differential approach, less so for people not directly working in this area.

We added two sentences for additional explanation:

"A complete and consistent, but non-decomposable SPN is valid and thus identical to some network polynomial on the domain of binary vectors. However, when taking the derivative, intuitively speaking, we also consider a small ϵ-environment around the 0's and 1's, corrupting the differential approach."

Theorem 3 is theoretically interesting indeed, although the authors aren't helping themselves by making light of the fact that a quadratic number of extra parameters is a massive increase by any practical measure.

Theorem 3 is indeed primarily a theoretic argument. As a famous analogy consider Savitch's theorem which shows that a deterministic Turing machine can decide the same languages as non-deterministic Turing machines using only *quadratically more space*, thus establishing that PSPACE = NPSPACE. Theorem 3 shows likewise that

"... any distribution which can be encoded using a consistent SPN using polynomially many arithmetic operations in $|\mathbf{X}|$, can also be polynomially encoded using a decomposable SPN."

Furthermore, we modified Algorithm 2 and the proof of Theorem 3, bounding the number of *arithmetic operations* when transforming a consistent into a decomposable SPN – the number of arithmetic operations is a more relevant complexity measure than the number of SPN nodes. In particular we show that the number of addition operations remains the same during Algorithm 2, i.e. $A_{\mathcal{S}'} = A_{\mathcal{S}}$. The increase of number of multiplication is bounded as $M_{\mathcal{S}'} = \mathcal{O}(M_{\mathcal{S}} |\mathbf{X}|)$, i.e. the number of multiplications grows at most linear times the number of RVs.

See also reply for Reviewer 2.

Metareviewer

But we voted to conditionally require that the authors connect better to Adnan's work and why a linear transformation cannot be performed.

See replies for the other reviewers, especially Reviewer 5.

Unfortunately, it is not trivial to show that a linear transformation from consistent to decomposable SPNs does *not* exist, and at the current state we cannot rule out this possibility. However, the modified version of Algorithm 2 shows that the number of multiplications grows at most linearly in the number of RVs. Using the main principle of Algorithm 2, *locally* refactoring the terms concerning the shared RVs Y, this bound can probably not be substantially improved. There could exist an algorithm using more *global* transformations. However, finding such algorithm, or rule out the possibility that it exists, is difficult.