Credal Sum-Product Networks

Denis Deratani Mauá DENIS.MAUA@USP.BR
Institute of Mathematics and Statistics, Universidade de São Paulo (Brazil)

Fabio Gagliardi Cozman FGCOZMAN@USP.BR
Escola Politécnica, Universidade de São Paulo (Brazil)

Diarmaid Conaty DCONATY01@QUB.AC.UK
Cassio Polpo de Campos C.DECAMPOS@QUB.AC.UK
Queen's University Belfast (United Kingdom)

Abstract

Sum-product networks are a relatively new and increasingly popular class of (precise) probabilistic graphical models that allow for marginal inference with polynomial effort. As with other probabilistic models, sum-product networks are often learned from data and used to perform classification. Hence, their results are prone to be unreliable and overconfident. In this work, we develop credal sum-product networks, an imprecise extension of sum-product networks. We present algorithms and complexity results for common inference tasks. We apply our algorithms on realistic classification task using images of digits and show that credal sum-product networks obtained by a perturbation of the parameters of learned sum-product networks are able to distinguish between reliable and unreliable classifications with high accuracy.

Keywords: Sum-product networks; tractable probabilistic models; credal classification.

1. Introduction

Probabilistic models are usually built so that they can be used to produce inferences, that is, to draw quantitative (probabilistic) conclusions about the domain of interest. Probabilistic graphical models such as Bayesian networks and Markov Networks (Koller and Friedman, 2009; Darwiche, 2009) allow complex uncertain knowledge to be modeled succinctly; however, producing inferences with them is notoriously hard (Cooper, 1990; Roth, 1996; Darwiche, 2009).

Sum-Product Networks (SPNs) are a relatively new class of (precise) probabilistic graphical models that allow marginal inference in linear time in their size (Poon and Domingos, 2011). They have received increasing popularity in applications of machine learning due to their ability to represent complex and highly multidimensional distributions (Poon and Domingos, 2011; Cheng et al., 2014; Nath and Domingos, 2016; Amer and Todorovic, 2016). An SPN encodes an arithmetic circuit whose evaluation produces a marginal inference (Darwiche, 2003). The internal nodes of a SPN perform (weighted) sums and multiplications, while the leaves represent variable assignments. The sum nodes can be interpreted as latent variables, while the product nodes can be interpreted as encoding context-sensitive probabilistic independences. Thus, SPNs can be seen as a class of complex mixture distributions with tractable inference (Zhao et al., 2015; Peharz et al., 2016).

Imprecise probability models extend precise probabilistic models to accommodate the representation of incomplete and indeterminate knowledge (Walley, 1991; Augustin et al., 2014). For example, (separately specified) credal networks extend Bayesian networks by allowing sets of conditional probability measures to be associated with nodes in lieu of conditional probability measures (Cozman, 2000, 2005).

In this work, we develop the *Credal Sum-Product Networks* (CSPNs), a class of imprecise probability models which extend SPNs to the imprecise case. A CSPN is simply an SPN where the weights associated with sum nodes (i.e., the numerical parameters of the model) are allowed to vary inside a closed and convex set. Among other things, CSPNs can be used to analyze the robustness of conclusions supported by SPNs.

We begin by presenting somee basic facts about SPNs in Section 2. Then in Section 3 we derive polynomial-time algorithms for computing upper and lower bounds on the marginal (unconditional) probability of an event; we also present a polynomial-time algorithm for computing upper and lower expectations when the structure is constrained so that every internal node has at most one parent. As many learning algorithms produce networks of this type (Gens and Domingos, 2013; Rooshenas and Lowd, 2014), this result is quite important and useful. We show that performing credal classification (i.e., verifying whether a class value dominates another value under maximality) is coNP-complete when the number of class values is unbounded. Since this task can be posed as the computation of a lower expectation, this result also shows hardness of computing expectation bounds on arbitrary (multivariate) functions. We show empirically in Section 4 that CSPNs are effective in assessing the reliability to classifications made with SPNs learned from data. Finally, we conclude the paper with a review of our contributions and some ideas for the future in Section 5.

2. Sum-Product Networks

We use capital letters to notate both random variables and random vectors, with the former usually being indexed by a subscript: e.g., X_i. If X is a random vector, we call the set composed of the random variables in X its *scope*. The scope of a function that takes a random vector X as argument is the scope of X. In this work, we consider only finite-valued random variables, and leave the extension to random variables with infinite domains as future work.

We associate every random variable X_i taking values in $\{0, \ldots, c_i - 1\}$ with a set of *indicator variables* $\{\lambda_{ij} : j = 0, \ldots, c_i - 1\}$, each taking on values 0 and 1. If X_i is binary, we write x_i (resp., $\bar{x}_i$) to denote λ_{i1} (resp., λ_{i0}). Any discrete multivariate distribution $P(X_\mathcal{V})$ can be written as a multilinear function of the corresponding indicator variables by $S(\lambda) = \sum_{x_\mathcal{V}} \mathbb{P}(X_\mathcal{V} = x_\mathcal{V}) \prod_{i \in \mathcal{V}} \lambda_{ix_i}$. For example, a Bernoulli distribution can be written as $S(x, \bar{x}) = \Pr(X = 1)x + \Pr(X = 0)\bar{x}$.

A SPN is a concise representation of the multilinear function representing a probability distribution. More formally, a SPN is a weighted rooted directed acyclic graph where internal nodes are associated to either sum or product operations and leaves are associated with indicator variables. Every arc from a sum node i to a child j is associated with a nonnegative weight w_{ij}, and every arc leaving a product node has weight one. The scope of a leaf node of the network is the respective random variable; the scope of an internal node is the union of the scopes of its children. If $\mathbf{w}$ are the weights of a subnetwork $S_\mathbf{w}$, we denote by $\mathbf{w}_i$ the weights in the subnetwork $S^i_{\mathbf{w}_i}$ rooted at node i, and by w_i the vector of weights w_{ij} associated with arcs from i to children j. Figure 1 shows an example of a SPN with scope $\{A, B\}$, where A and B are binary variables.

An SPN satisfies the following properties (Poon and Domingos, 2011; Peharz et al., 2015): (i) every indicator variable appears in at most one leaf node; (ii) the scope of any two children of a sum node are identical (completeness); (iii) the scopes of any two children of a product node are disjoint (decomposition); (iv) the sum of the weights associated with any sum node is one (normalization). Every discrete distribution can be represented by a SPN, and any SPN satisfying the those properties represents a valid distribution.

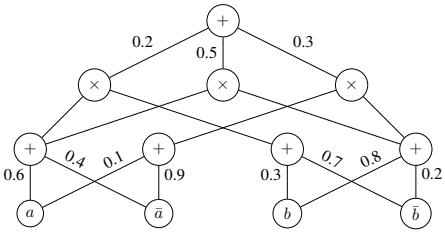

Figure 1: A sum-product network over binary random variables A and B.

The evaluation of a SPN for a given configuration λ of the indicator variables is performed from the leaves toward the root. The leaves (indicator variables) propagate up their corresponding value (either 0 or 1) in the configuration λ. Sum (resp., product) nodes propagate the weighted sum (resp., product) of the values of their children multiplied by the corresponding arc weights. For example, the value of the SPN $S_{\mathbf{w}}(a, \bar{a}, b, \bar{b})$ in Figure 1 at the point $\lambda = (1, 0, 0, 1)$ is 0.15 and corresponds to $\mathbb{P}(A = 1, B = 0)$, and for $\lambda = (1, 0, 1, 1)$ is 0.45 and corresponds to $\mathbb{P}(A = 1)$.

Let $\mathcal{E} \subseteq \{1, \ldots, n\}$ be an index set, and $X_{\mathcal{E}}$ be a random vector of scope $\{X_i : i \in \mathcal{E}\}$. The marginal probability of some evidence $\{X_i = e_i : i \in \mathcal{E}\}$ induced by a SPN S can be obtained by evaluating the network at λ that is consistent with the evidence, and assigns one to all other indicator variables (Poon and Domingos, 2011). That is, $\lambda_{ij} = 0$ if $i \in \mathcal{E}$ and $e_i \neq j$, and $\lambda_{ij} = 1$ otherwise. Thus marginal probabilities can be computed in time linear in the network size (the number of nodes, arcs and weights). For example, the marginal probability $\mathbb{P}(B = 0) = 0.3$ induced by the SPN in Figure 1 can be obtained by evaluating $S(a, \bar{a}, b, \bar{b})$ at $\lambda = (1, 1, 0, 1)$. Conditional probabilities can either be obtained by evaluating the network at query and evidence (then dividing the result) or by applying Darwiche's differential approach (Darwiche, 2003; Peharz et al., 2016).

A great deal of algorithms have been devised to "learn" SPNs from data (Dennis and Ventura, 2012; Gens and Domingos, 2013; Peharz et al., 2013, 2014; Lee et al., 2014; Rooshenas and Lowd, 2014; Dennis and Ventura, 2015; Adel et al., 2015; Rahman and Gogate, 2016). Most learning algorithms employ a greedy search on the space of SPNs augmenting an SPN in either a top-down or bottom-up fashion. For instance, Gens and Domingos (2013)'s algorithm starts with a single node representing the entire dataset, and recursively adds product and sum nodes that divide the dataset into smaller datasets until a stopping criterion is met. Product nodes are created using group-wise independence tests, while sum nodes are created performing clustering on the row instances. The weights associated with sum nodes are learned as the proportion of instances assigned to a cluster.

3. Credal Sum-Product Networks

Let $S_{\mathbf{w}}$ denote a SPN whose weights are $\mathbf{w}$. We can obtain an imprecise sum-product network by allowing the weights $\mathbf{w}$ to vary in some space, subject to the constraint that they still define a SPN. More formally, a *Credal Sum-Product Network* (CSPN) is a set $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$, where $\mathcal{C}$ is the Cartesian product of probability simplexes, and each probability simplex constrains only the weights associated with a single sum node. It is clear that a SPN is a CSPN where weights take values in a singleton $\mathcal{C}$, and that every choice of weights $\mathbf{w}$ inside $\mathcal{C}$ specifies a SPN. Since each SPN induces a probability measure, the CSPN induces a *credal set*, that is, a (not necessarily convex) set

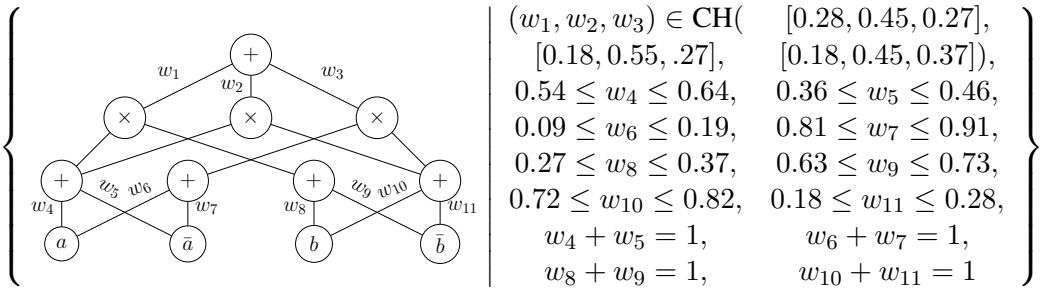

Figure 2: A credal sum-product network over variables A and B.

of probability measures (Levi, 1980). Figure 2 shows a CSPN obtained by ϵ-contamination of the SPN in Figure 1, with $\epsilon = 0.1$.

A SPN can be interpreted as a bilevel bipartite Bayesian network by identifying sum nodes with latent variables whose probability distributions are obtained from the corresponding weights (Zhao et al., 2015). The network has a layer of latent variables $Y_1, \ldots, Y_m$ corresponding to sum nodes of the network, and a layer of leaf variables $X_1, \ldots, X_n$ corresponding to (scopes of) indicator variables. There is an arc $Y_j \to X_i$ if and only if X_i is in the scope of the sum node (associated with) Y_j. Each variable Y_j has as many values as children, and its (unconditional) probabilities are specified as the associated weights. The (conditional) probabilities associated with a node X_i are specified as the weights entering the corresponding indicator variable (which depend on the value of the respective latent variables). Note that a variable X_i can have a large number of parents, so that obtaining this Bayesian network is often impracticable.

We can adapt a similar argument for CSPNs: sum nodes can be interpreted as latent variables in a credal network. This network is obtained exactly as the Bayesian network, except that conditional probability distributions are replaced by conditional credal sets.

3.1 Likelihood

The most trivial inference with CSPNs is to compute the minimum and maximum values obtained at a SPN for a given value λ of the indicator variables. This computation corresponds to computing the upper and lower likelihood of evidence, and can be performed in much the same way as the computation of marginal probabilities in SPNs, with the additional extra effort of solving a linear program at each node. To see this, consider a tree-shaped CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ with root r. Since the structure is a tree, the subnetworks $S^1, \ldots, S^k$ rooted at the children of a node i do not share any weights. Hence, we have that $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda) = \min_{w_i} \sum_j w_{ij} \min_{\mathbf{w}_j} S^j_{\mathbf{w}_j}(\lambda)$. Thus, the problem of computing the minimum or maximum of a value λ decomposes into smaller similar problems. A much similar argument applies to CSPNs with cycles; simply break the cycles by duplicating nodes until the structure is a tree, and perform optimizations from the leaves toward the root. Every duplicated network receives the same values from the (duplicated) children; thus the optimizations are the same whether we "tie" the weights of identical parts or not. A more formal argument is given next.

Theorem 1 *Consider a* CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$*, where* $\mathcal{C}$ *is the Cartesian product of finitely-generated polytopes* $\mathcal{C}_i$*, one for each sum node* i*. Computing* $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$ *and* $\max_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$ *takes* $O(sL)$ *time, where* s *is the number of nodes and arcs in the shared graphical structure and* L *is an upper bound on the cost of solving a linear program* $\min_{w_i} \sum_j c_{ij} w_{ij}$ *subject to* $w_i \in \mathcal{C}_i$*.*

Proof Consider the computation of $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$ (the case for $\max$ is analogous), and let $1, \ldots, k$ denote the sum nodes of the network. By construction, the optimization is over weight vectors $\mathbf{w} = (w_1, \ldots, w_k)$, where w_i denotes the weights associated with the sum node i, and vary in a finitely-generated polytope $\mathcal{C}_i$. Now start at the leaves. There are no weights associated, so these nodes simply propagate their values as in SPNs. Consider a sum node i, and assume that the weights of the subnetworks at its children have been optimized (and are hence fixed). The corresponding optimization is then $\min_{\mathbf{w}} \sum_p \mathbf{w}_p \sum_j w_{ij} S^j_{\mathbf{w}_j}(\lambda) + C_{\mathbf{w}}$, where the leftmost sum is over all paths from the root to i, the inner sum is over the children j of i, and $C_{\mathbf{w}}$ contains the subnetwork formed by nodes which are neither an ancestor nor a descendant of i (hence can be optimized independently of $\mathbf{w}_i$); this expression defines a linear program with (finitely many) linear constraints $w_i \in \mathcal{C}_i$. Solving this linear program takes time $O(L)$. The result follows by induction on the height of subnetworks. ■

The algorithm to compute the minimum or maximum values at a configuration λ visits nodes from leaves toward the root: at product or indicator nodes, it evaluates the corresponding expression as in SPNs; at a sum node, it builds the corresponding linear program and calls a solver. Since linear programs can be solved in polynomial time, the overall time is also polynomial in the size of the input (which includes a description for the local polytopes). This leads to the following:

Corollary 2 *Computing* $\min_w S_w(\lambda)$ *and* $\max_w S_w(\lambda)$ *takes at most polynomial time in* CSPN*s specified by finitely-generated polytopes.*

3.2 Conditional Expectations

Each choice of the weights $\mathbf{w}$ of a CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ defines a SPN and hence induces a probability measure $\mathbb{P}_{\mathbf{w}}$. We can thus use the CSPN to compute upper and lower conditional expectations:

$$\max_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e) \quad \text{and} \quad \min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e)\,,$$

where $X_{\mathcal{Q}}$ are known as target variables, $f : X_{\mathcal{Q}} \to \mathbb{Q}$ is a function to rational numbers and $X_{\mathcal{E}} = e$ is the evidence. We will focus on the lower expectation, since the upper expectation can be obtained from $\max_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|e) = -\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(-f|e)$. This inference is however intractable (under the common assumptions in complexity theory):

Theorem 3 *Assuming that* f *is encoded succinctly (e.g., sparsely by its non-zero terms only), computing lower/upper conditional expectations of* f *in* CSPN*s is NP-hard.*

We defer the proof to Section 3.3, where we address the case of credal classification (that can be posed as the computation of a conditional expectation). The requirement of a succinct representation for f is necessary because an exponentially large input would give too much power to any algorithm (since polynomial time in the input would allow exponential time computations).

While the general case is NP-hard, there are useful subcases with tractable inference. We now present an algorithm for the computation of lower and upper conditional expectations when the network obtained by discarding leaves is a tree and $f : X_{\mathcal{Q}} \to \mathbb{Q}$ is a univariate function. The algorithm is based on the generalized Bayes rule, and uses the fact that, for any real μ:

$$\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e) > \mu \iff \min_{\mathbf{w}} \sum_{q \in X_{\mathcal{Q}}} (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}}(X_{\mathcal{Q}} = q, X_{\mathcal{E}} = e) > 0\,, \qquad (1)$$

provided that $\max_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e) > 0$ (this can be checked in polynomial time, see Section 3.1). We can also check efficiently whether $\min_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e) = 0$, and decide what to do in such extreme scenarios. If we can decide Inequality (1) for any μ, then we can perform a binary search to find the value of $\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e)$.

Theorem 4 *Computing lower/upper conditional expectations of a variable in* CSPN*s takes at most polynomial time when each internal node has at most one parent.*

Proof Let λ_e be the assignment of indicator variables that is consistent with $X_{\mathcal{E}} = e$ and assigns 1 to variables not in $X_{\mathcal{E}}$. As shown before, we can efficiently compute $\max_{\mathbf{w}} S_{\mathbf{w}}(\lambda_e) = \max_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e)$ and $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda_e) = \min_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e)$. To compute a lower conditional expectation we might do a binary search to find μ such that

$$\min_{\mathbf{w}} \sum_{q \in X_{\mathcal{Q}}} (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}}(X_{\mathcal{Q}} = q, X_{\mathcal{E}} = e)\,.$$

To simplify notation we will write $\mathbb{P}_{\mathbf{w}}(q, e)$ to denote $\mathbb{P}_{\mathbf{w}}(X_{\mathcal{Q}} = q, X_{\mathcal{E}} = e)$. Now suppose that the CSPN has a product root node 0 with children $1, \ldots, k$ and (without loss of generality) only node 1 has $X_{\mathcal{Q}}$ in its scope. Then, because the scopes of children of product nodes are fully disjoint and the internal graph of the CSPN forms a tree, we have that

$$\min_{\mathbf{w}_0} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_0}(q, e_0) = \left(\min_{\mathbf{w}_1} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_1}(q, e_1) \right) \cdot \prod_{j=2}^{k} \mathbb{P}^*_{\mathbf{w}_j}(e_j)\,,$$

where e_j is the evidence for $\mathcal{E}_j$ within the scope of child j (note that $\mathcal{E}_1$ might be empty and e_1 would disappear), $\mathbb{P}^*_{\mathbf{w}_j}(e_j) = \max_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_j)$ in the case that $\min_{\mathbf{w}_1} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_1}(q, e_1) < 0$ and $\mathbb{P}^*_{\mathbf{w}_j}(e_j) = \min_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_j)$ otherwise. Hence, if we assume that $S^1_{\mathbf{w}_1}(\lambda) = \min_{\mathbf{w}_1} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_1}(q, e_1)$ and that $S^j_{\mathbf{w}_j}(\lambda) = \mathbb{P}^*_{\mathbf{w}_j}(e_j)$, then from the computation scheme of the CSPN for a sum node, it is clear that $S^0_{\mathbf{w}_0}(\lambda) = \min_{\mathbf{w}_0} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_0}(q, e_0)$. The assumption $S^j_{\mathbf{w}_j}(\lambda) = \mathbb{P}^*_{\mathbf{w}_j}(e_j)$ is satisfied by definition for all children j that are leaf nodes and do not contain $X_{\mathcal{Q}}$. Moreover, if the node 0 is a product node and does not have $X_{\mathcal{Q}}$ in its scope, then

$$\min_{\mathbf{w}_0} \mathbb{P}_{\mathbf{w}_0}(e_0) = \prod_{j=1}^{k} \min_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_j) \text{ and } \max_{\mathbf{w}_0} \mathbb{P}_{\mathbf{w}_0}(e_0) = \prod_{j=1}^{k} \max_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_j)\,,$$

and so it is immediate that $\min_{\mathbf{w}_0} S^0_{\mathbf{w}_0}(\lambda) = \min_{\mathbf{w}_0} \mathbb{P}_{\mathbf{w}_0}(e_0)$ if each $S^j_{\mathbf{w}_j}(\lambda) = \min_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_j)$ (analogous for the maximization).

If node 0 is a sum node with $X_{\mathcal{Q}}$ in its scope, then because the internal graph of the CSPN is a tree and expectations are linear, we have that

$$\min_{\mathbf{w}_0} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_0}(q, e_0) = \min_{w_0} \sum_{j=1}^{k} w_{0,j} \cdot \min_{\mathbf{w}_j} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_j}(q, e_0)\,,$$

where $w_0 = (w_{0,1}, \ldots, w_{0,k})$ varies in the corresponding polytope specifying the weights of the current node 0 (note that $\mathcal{E}_0$ might be empty and e_0 would disappear). Hence, if we assume that $S^j_{\mathbf{w}_j}(\lambda) = \min_{\mathbf{w}_j} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_j}(q, e_0)$, it is immediate from the local computation of the CSPN for a sum node that $S^0_{\mathbf{w}_0}(\lambda) = \min_{\mathbf{w}_0} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_0}(q, e_0)$. If node 0 is a sum node without $X_{\mathcal{Q}}$, then

$$\min_{\mathbf{w}_0} \mathbb{P}_{\mathbf{w}_0}(e_0) = \min_{w_0} \sum_{j=1}^{k} w_{0,j} \cdot \min_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_0)\,,$$

where $w_0 = (w_{0,1}, \ldots, w_{0,k})$ varies in the polytope specifying the weights of the current node 0. Again, $\min_{\mathbf{w}_0} S^0_{\mathbf{w}_0}(\lambda) = \min_{\mathbf{w}_0} \mathbb{P}_{\mathbf{w}_0}(e_0)$ if each $S^j_{\mathbf{w}_j}(\lambda) = \min_{\mathbf{w}_j} \mathbb{P}_{\mathbf{w}_j}(e_0)$ (analogous for the maximization). Finally, if node 0 is a leaf node with scope $X_{\mathcal{Q}}$, then

$$\min_{\mathbf{w}_0} \sum_q (f(q) - \mu) \cdot \mathbb{P}_{\mathbf{w}_0}(q) = f(q') - \mu\,,$$

where q' is the value of the variable $X_{\mathcal{Q}}$ associated to the $\lambda_{\mathcal{Q},q'}$ of the leaf node. Therefore, by using these expressions, we can perform the computation recursively and obtain the desired upper or lower conditional expectation in polynomial time. ■

3.3 Credal Classification

SPNs are most often constructed to perform probabilistic classification: to assign each object the assignment that maximizes the probability of a distinguished set of variables $X_{\mathcal{C}}$ given the realization of (a subset of) the remaining variables. Since CSPNs define more than a single SPN, there is more than one such possible maximizer. Many criteria have been devised for decision making with imprecise probability models. Here we adopt a very popular one, based on the principle of maximality, often called *credal classification* in the context of probabilistic classifiers.

Given distinguished variables $X_{\mathcal{C}}$, evidence $e = \{X_i = e_i : i \in \mathcal{E}\}$ on some variables, and a credal set $\mathcal{M}$, we say that an assignment c' for $\mathcal{X}_C$ *credally dominates* another assignment c'' if (Zaffalon, 2002)

$$\min_{\mathbb{P} \in \mathcal{M}} \big(\mathbb{P}(X_{\mathcal{C}} = c', X_{\mathcal{E}} = e) - \mathbb{P}(X_{\mathcal{C}} = c'', X_{\mathcal{E}} = e)\big) > 0\,.$$

There is a special case of $\mathbb{P}(X_{\mathcal{E}} = e) = 0$ to be treated— an advantage in CSPNs is that computing lower/upper marginals is efficient. According to the above definition, an assignment c' credally dominates class value c'' if $\mathbb{P}(X_{\mathcal{C}} = c' | X_{\mathcal{E}} = e) > \mathbb{P}(X_{\mathcal{C}} = c'' | X_{\mathcal{E}} = e)$ for all $\mathbb{P} \in \mathcal{M}$ where these conditional probabilities are defined. In the setting of CSPNs, credal dominance amounts to establishing whether $\min_{\mathbf{w}}(S_{\mathbf{w}}(\lambda_{c'}, \lambda_e) - S_{\mathbf{w}}(\lambda_{c''}, \lambda_e)) > 0$, where $\lambda_{c'}$ (resp., $\lambda_{c''}$) is the assignment of indicator variables associated with variables $X_{\mathcal{C}}$ consistent with c' (resp., c''), and λ_e is the

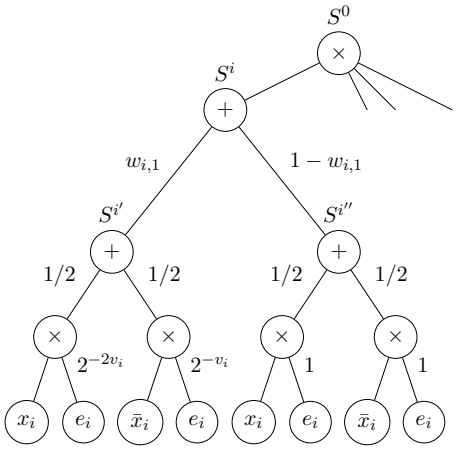

Figure 3: Fragment of the sum-product network used to solve PARTITION. We duplicate leaves for the sake of readability.

assignment of indicator variables consistent with evidence (if there are other variables, these have their indicator variables set to one to indicate their marginalization).

Lemma 5 *If we allow all weights* $\mathbf{w}$ *of an* SPN *an additive variation* $\varepsilon > 0$*, then the result of* $S(\lambda)$ *will vary (additively) at most* $O(s) \cdot \varepsilon$*, where* s *is the number of nodes and arcs in the graphical structure. If we allow all* $\mathbf{w}$ *a multiplicative variation* $\varepsilon > 0$*, then the result of* $S(\lambda)$ *will vary (multiplicatively) at most* $\varepsilon^{O(s)}$*.*

Proof Each sum node propagates an extra error of at most ε, while the product nodes propagate an extra error of at most $O(d) \cdot \varepsilon$, where d is its degree. So the result follows by induction. For the multiplicative error, we hve that sum nodes contribute at most a factor ε to the error, while product nodes may contribute a factor $\varepsilon^{O(d)}$. Hence the overall result follows. ■

Theorem 6 *Credal classification is coNP-complete.*

Proof Membership in coNP is trivial: Given $\mathbf{w}$, computing $S_{\mathbf{w}}(\lambda_{c'}, \lambda_e) - S_{\mathbf{w}}(\lambda_{c''}, \lambda_e)$ is a polynomial time task. Hence, there is a polynomial certificate $\mathbf{w}$ that confirms that $\min_{\mathbf{w}}(S_{\mathbf{w}}(\lambda_{c'}, \lambda_e) - S_{\mathbf{w}}(\lambda_{c''}, \lambda_e)) \leq 0$ if that is indeed the case, and since credal classification is the complement, membership follows.

Hardness follows by a reduction from the NP-hard problem PARTITION: Given a set of integers $z_1, \ldots, z_n$, decide if there is a set $\mathcal{S} \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in \mathcal{S}} z_i = Z/2$, where $Z = \sum_i z_i$. First note that we can scale integers to become rationals in the unit interval without affecting complexity: Let $v_i = 2z_i/Z$; then set $\mathcal{S}$ solves the original problem if and only if $\sum_{i \in \mathcal{S}} v_i = 1$.

Now, we build an CSPN over variables $X = (X_1, \ldots, X_n, X_{n+1}, \ldots, X_{2n})$ as in Figure 3, where the weights $w_{i,1}$ vary in $[0, 1]$, and let $\mathcal{C} = \{1, \ldots, n\}$ and $\mathcal{E} = \{n + 1, \ldots, 2n\}$. Since the variables X_i, $i \in \mathcal{E}$, have their value fixed by the evidence e (say $X_i = 1$), we only show the corresponding value in the figure. The product node S^0 has children $S^1, \ldots, S^n$. Note that for

$X_{\mathcal{C}} = c'$, $S^{i'}$ computes 2^{-2v_i} while $S^{i''}$ computes 1; and for $X_{\mathcal{C}} = c''$, $S^{i'}$ computes 2^{-v_i} and $S^{i''}$ computes 1. Because the weights are minimized at $\{0, 1\}$. Thus, we have that $S(\lambda_{c'}, \lambda_e) = 2^{-\sum_{i:w_{i,1}=1} 2v_i - n}$ and $S(\lambda_{c''}, \lambda_e) = 2^{-\sum_{i:w_{i,1}=1} v_i - n}$. Hence,

$$S_{\mathbf{w}}(\lambda_{c'}, \lambda_e) - S_{\mathbf{w}}(\lambda_{c''}, \lambda_e) = 2^{-n} \cdot (t^2 - t) = 2^{-n} \cdot t \cdot (t - 1), \text{ with } t = 2^{-\sum_{i:w_{i,1}=1} v_i}.$$

Now, deciding whether $\min_{\mathbf{w}}(S_{\mathbf{w}}(\lambda_{c'}, \lambda_e) - S_{\mathbf{w}}(\lambda_{c''}, \lambda_e)) \leq -2^{-n-2}$ solves the partition problem since t would be 2^{-1}. With a small change in the model, we can move the threshold -2^{-n-2} to zero, as required in the classification problem. Therefore, credal classification is coNP-complete. However, we have to deal with the specification of 2^{-v_i} in polynomial time. To do so, we find rational numbers which approximate them, and in view of Lemma 5, we can find accurate enough results to separate between yes and no instances of PARTITION. ■

Since credal classification can be casted as the computation of the lower expectation of a univariate function, we have from Theorem 4 that:

Theorem 7 *Credal classification with a single class variable can be done in polynomial time in* CSPN*s when each internal node has at most one parent.*

4. Experiments

We evaluate the ability of CSPNs in distinguishing between robust and non-robust classifications in a handwritten digit recognition task. The dataset consists of 700 digitalized images of handwritten Arabic numerals ranging from 0 to 9 (70 images per digit). Each image consists of 20×30 pixels taking on values 0 and 1, and we associate every pixel with a binary variable. To assess the effect of dataset size, we consider two splits in training/test data: 50%/50% and 20%/80%. For each split, we learn a SPN from the training set using the approach discussed by Poon and Domingos (2011), and use it to classify each instance in the test set. Then, for each test instance, we find the maximum value of ϵ such that the CSPN obtained by imposing a local ϵ-contamination to each of the sum nodes produces a single classification under maximality (which is equivalent to *E-admissibility* in this case). Call this value the *classification robustness*. We repeat this procedure 10 times using different random partitions of the data into train and test parts. The curves show the accuracy (no. of correctly classified instances/no. of instances) of the SPN for instances with robustness at most a given ϵ (x-axis). The results are compiled into Figure 4. We see that the higher the robustness the greater the accuracy.

Examples of misclassified instances are given in Figure 5. For comparison, we also analyze a different approach to measure robustness: we compute the difference between the probability of the most probable class and the second most probable class. As we see in the figure, this measure correlates poorly with the accuracy.

In order to give a more quantitative perspective of the robustness value, we present in Table 1 some descriptive statistics for correctly and wrongly classified instances, using either robustness measure. We see a much clearer separation of the robustness values between correctly and incorrectly classified instances using the CSPN approach instead of the "best minus second best" probability approach.

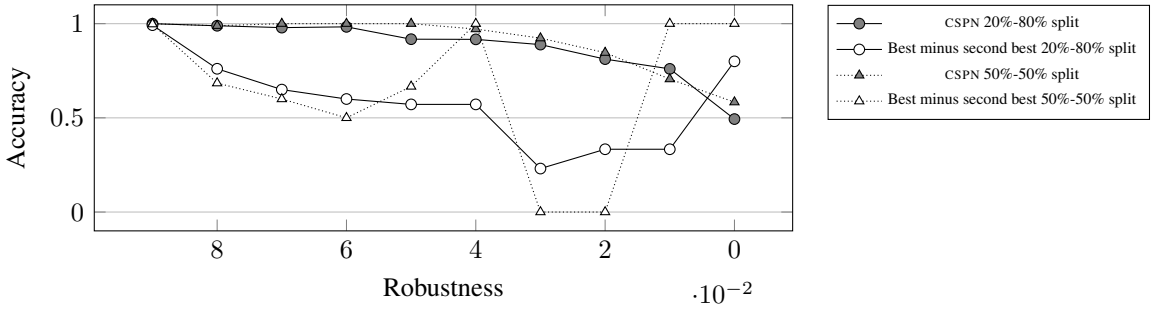

Figure 4: Average classification accuracy for instance below the given robustness (the x-axis shows the values times a constant 20 to be visually compatible with the probabilities), as explained in the text.

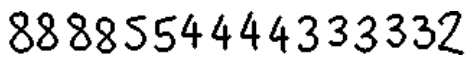

Figure 5: Examples of misclassified instances. Usually, number 8 is misclassified as 3, number 4 as number 1, and number 3 as 5 and 8. These classifications obtained low robustness values as given by the CSPN analysis (< 0.01), rightfully indicating the lack of statistical support.

5. Conclusion

Sum-product networks are tractable probabilistic graphical models that have shown competitive results in many machine learning tasks. In this work we developed the credal sum-product networks, a new class of imprecise probabilistic graphical models that extend sum-product networks to accommodate imprecision in the numerical parameters. We described algorithms and complexity for common inference tasks such as computing upper and lower bounds on the probability of evidence, computing conditional expectations and performing credal classification. We performed experiments that showed that credal sum-product networks can distinguish between reliable and unreliable classifications of sum-product networks, thus providing an important tool for the analysis of

Robustness	CSPN		Best minus second best	
Measure	Correct	Wrong	Correct	Wrong
1st quartile	0.0255	0.0012	0.0909	0.0627
median	0.0363	0.0029	0.0909	0.0880
3rd quartile	0.0461	0.0049	0.0909	0.0905
maximum	0.1524	0.0199	0.3333	0.3333
mean (std.dev.)	0.0369 $\pm$0.017	0.0043 $\pm$0.004	0.0976 $\pm$0.04	0.1042 $\pm$0.09

Table 1: Robustness values for split of 50% training and 50% testing, repeated 10 times. Overall classification accuracy of 99.31%.

such models. There are many open questions. We showed that verifying maximality is coNP-hard when the query involves multiple variables, but the problem admits an efficient solution if internal nodes have at most one parent and the test is over a single variable. In fact, we have showed a polynomial algorithm for computing conditional expectations in networks of that structure, which subsumes maximality. There remains to establish the complexity of verifying maximality and computing conditional expectations for single variables in general structures, and for multiple variables in tree-shaped networks. Our experiments here, however promising, are preliminary. In the future, we intend to perform a more thorough examination of the credal sum-product networks applied to robust analysis of sum-product networks.

Acknowledgments

This work was partially supported by CNPq (grants 308433/2014-9, 303920/2016-5) and FAPESP (grants 2016/01055-1). We greatly thank Renato Geh for making his source code and the handwritten digits dataset publicly available (at http://github.com/RenatoGeh/gospn).

References

T. Adel, D. Balduzzi, and A. Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 32–41, 2015.

M. R. Amer and S. Todorovic. Sum product networks for activity recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(4):800–813, 2016.

T. Augustin, F. P. A. Coolen, G. De Cooman, and M. C. M. Troffaes. *Introduction to Imprecise Probabilities*. 2014.

W.-C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, and K. M. A. Chai. Language modeling with sum-product networks. In *Proccedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH'14)*, 2014.

G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial intelligence*, 42(2–3):393–405, 1990.

F. G. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

F. G. Cozman. Graphical models for imprecise probabilities. In *International Journal of Approximate Reasoning*, volume 39, pages 167–184, 2005.

A. Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50 (3):280–305, 2003.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

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.

A. Dennis and D. Ventura. Greedy structure search for sum-product networks. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 932–938, 2015.

R. Gens and P. Domingos. Learning the structure of sum-product networks. In *Proc. 30th Int. Conf. on Mach. Learning*, pages 873–880, 2013.

D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT press, 2009.

S.-W. Lee, C. Watkins, and B.-T. Zhang. Non-Parametric Bayesian Sum-Product Networks, 2014.

I. Levi. *The Enterprise of Knowledge*. MIT Press, London, 1980.

A. Nath and P. Domingos. Learning tractable probabilistic models for fault localization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1294–1301, 2016.

R. Peharz, B. C. Geiger, and F. Pernkopf. Greedy part-wise learning of sum-product networks. In *Machine Learining and Knowledge Discovery in Databases*, volume 8189 LNAI, pages 612–627, 2013.

R. Peharz, R. Gens, and P. Domingos. Learning selective sum-product networks. In *ICML Workshop on Learning Tractable Probabilistic Models*, volume 32, 2014.

R. Peharz, S. Tschiatschek, F. Pernkopf, and P. Domingos. On theoretical properties of sum-product networks. In *Proceedings of the 18th International Con- ference on Artificial Intelligence and Statistics*, pages 744–752, 2015.

R. Peharz, R. Gens, F. Pernkopf, and P. Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2016.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proc. 27th Conf. on Uncertainty in Artif. Intell.*, pages 337–346, 2011.

T. Rahman and V. Gogate. Merging strategies for sum-product networks: From trees to graphs. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, pages 617–626, 2016.

A. Rooshenas and D. Lowd. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning*, pages 710–718, 2014.

D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

M. Zaffalon. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1):5–21, 2002.

H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and bayesian networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 116–124, 2015.