Robustifying Sum-Product Networks

Denis Deratani Mauá

Institute of Mathematics and Statistics, Universidade de São Paulo, Brazil

Diarmaid Conaty

Centre for Data Science and Scalable Computing, Queen's University Belfast, U.K.

Fabio Gagliardi Cozman

Escola Politécnica, Universidade de São Paulo, Brazil

Katja Poppenhaeger

Astrophysics Research Centre, Queen's University Belfast, U.K.

Cassio Polpo de Campos

Centre for Data Science and Scalable Computing, Queen's University Belfast, U.K.
Dept. of Information and Computing Sciences, Utrecht University, The Netherlands

---

Abstract

Sum-product networks are a relatively new and increasingly popular family of probabilistic graphical models that allow for marginal inference with polynomial effort. They have been shown to achieve state-of-the-art performance in several tasks involving density estimation. Sum-product networks are typically learned from data; as such, inferences produced with them are prone to be unreliable and overconfident when data is scarce. In this work, we develop the credal sum-product networks, a generalization of sum-product networks that uses set-valued parameters. We present algorithms and complexity results for common inference tasks with this class of models. We also present an approach for assessing the reliability of classifications made with sum-product networks. We apply this approach on benchmark classification tasks as well as a new application in predicting the age of stars. Our experiments show that the use of credal sum-product networks allow us to distinguish between reliable and unreliable classifications with higher accuracy than standard approaches based on (precise) probability values.

Preprint submitted to IJAR *July 4, 2018*

Keywords: Sum-product networks, tractable probabilistic models, credal classification, sensitivity analysis, robust statistics.

1. Introduction

Probabilistic graphical models such as Bayesian networks and Markov networks allow for the compact specification of uncertain knowledge through a graphical language that represents variables as nodes and dependences as graph connectivity [17, 30]. Not only this graphical approach facilitates knowledge elicitation and communication, but is key to achieving efficient inference. For example, while marginal inference in Bayesian and Markov networks is #P-hard [54], networks of low treewidth (roughly meaning that their graphs resemble trees) admit polynomial-time inference [21, 34]. Moreover, the most popular approximate inference algorithms are based on passing messages through the graph structure, and their properties depend heavily on the graph topology [32, 62, 64]. In spite of that, many event-level independences are not properly captured by this graphical representation: these are variously called local structure [9], causal independence [67] and context-specific independence [6]. For example, consider a Boolean variable X_1 defined as the disjunction of Boolean variables X_2 and X_3. Knowing that $X_2 = 1$ (or, alternatively, that $X_3 = 1$) renders X_1 independent of X_3 (alternatively, of X_2). However, the standard representation as a Bayesian network or Markov network hides away this independence inside a conditional probability table or factor.[1]

Sum-Product Networks (SPNs) are a relatively new class of (precise) probabilistic graphical models that allow for the explicit representation of context-specific independence [48]. They have received increasing popularity in applications of machine learning due to their ability to represent complex and highly multidimensional distributions while enabling linear time marginal inference [2, 10, 43, 49, 50, 52, 57, 70] (but see [11] for a discussion on the complexity of most probable explanation inference). An SPN encodes an arithmetic circuit whose evaluation produces a marginal inference [15, 18, 53]. The internal nodes of an SPN perform (weighted) sums and multiplications, while the leaves represent variable assignments. The sum nodes can be in-

[1]Despite the fact that such event-level independences are "hidden" in the representation, they can still be harvested to speed up computations [9, 14, 28, 33, 55, 56, 60, 61].

terpreted as latent variables, while the product nodes can be interpreted as encoding (context-specific) probabilistic independences. Thus, SPNs can be seen as a class of (very complicated) mixtures of univariate distributions with tractable inference [27, 46, 69].

In spite of its relative success, and on par with standard probabilistic graphical models, SPNs learned from data can generalize poorly on regions with insufficient statistical support, and produce unreliable and overconfident conclusions.

Imprecise probability models extend precise probabilistic models to accommodate the representation of incomplete and indeterminate knowledge [3, 63]. 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 [12, 13]. This addition in representation power comes at an increased computational cost for inferences, and best exact and approximation algorithms can only be used in small settings [20, 40, 41].

In this work, we seek to robustify SPNs by developing 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.

The rest of this document is organized as follows. We begin by presenting some 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 [27, 53, 59], 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 7 that CSPNs are effective in assessing the reliability to classifications made with SPNs learned from data. We perform experiments using benchmark datasets from the UCI repository, as well as an application regarding

the prediction of the age of stars, where data is inherently scarce (due to its cost) and reliability of results are essential. Finally, we conclude the paper with a review of our contributions and some ideas for future work in Section 8. This work extends previous work presented at the International Symposium on Imprecise Probability: Theory and Applications [39] with a more gently introduction to SPNs, complete proofs of the theoretical results and more extensive experimental analysis.

2. Sum-Product Networks

Before formally defining SPNs, let us define some notation and terminology. We write integers in lower case (e.g., i, j, k), and sets of integers using capital calligraphic letters (e.g., $\mathcal{V}, \mathcal{E}$). A collection of random variables indexed by a set $\mathcal{V}$ is denoted by $X_{\mathcal{V}} = \{X_i : i \in \mathcal{V}\}$. When the index set is not important, we denote a collection of variables simply by X. There is no ambiguity since random variables are denoted always with a subscript (e.g., X_1, X_i). As usual, we write a realization of a collection of random variables as, for example, $X_{\mathcal{V}} = x_{\mathcal{V}}$ or $X_{\mathcal{V}} = z$. The set of all realizations of a collection of random variables $X_{\mathcal{V}}$ is denoted as $x_{\mathcal{V}} \in X_{\mathcal{V}}$, or simply as $x_{\mathcal{V}}$, when clear from the context (e.g., as in the range of sums).

In this work, we assume that random variables take only on a finite number of values. This allows us to associate every (finite-valued) random variable X_i taking values in $\{0, \ldots, |X_i| - 1\}$ with a set of *indicator variables* $\{\lambda_{i,j} : j = 0, \ldots, |X_i| - 1\}$, each taking on values 0 and 1. If X_i is binary, we also write x_i (resp., $\bar{x}_i$) to denote $\lambda_{i,1}$ (resp., $\lambda_{i,0}$). We denote an arbitrary specification of the indicator variables associated with random variables $X_{\mathcal{V}}$ as λ. For any realization $X_{\mathcal{V}} = x_{\mathcal{V}}$ we write λ^{x_v} to denote the configuration of indicator variables such that $\lambda_{i,x_i} = 1$ and $\lambda_{i,j} = 0$ for all $j \neq x_i$. For reasons that shall become clear later, when the realization mentions only a subset of all the variables, say $X_{\mathcal{E}} = e$ for $\mathcal{E} \subset \mathcal{V}$, we write λ^e to denote the configuration of indicator variables that assigns $\lambda_{i,j} = 0$ if $i \in \mathcal{E}$ and $e_i \neq j$ and $\lambda_{i,j} = 1$ otherwise. That is, λ^e is the configuration of indicator variables that is *consistent* with the realization and assigns 1 to indicator variables associated to unrealized random variables.

Any discrete probability measure $\mathbb{P}$ induced by random variables $X_{\mathcal{V}}$ can be represented as a multilinear polynomial on the corresponding indicator variables by $P(\lambda) = \sum_{x_{\mathcal{V}}} \mathbb{P}(X_{\mathcal{V}} = x_{\mathcal{V}}) \prod_{i \in \mathcal{V}} \lambda_{i,x_i}$. The probability of a realization $X = x$ of the variables can thus be recovered by evaluating the

polynomial P at λ^x. For example, a Bernoulli distribution can be written as $P(x, \bar{x}) = \mathbb{P}(X = 1)x + \mathbb{P}(X = 0)\bar{x}$. The probabilities of $X = 1$ and of $X = 0$ are given, respectively, by $P(1, 0)$ and $P(0, 1)$.

An SPN is a concise graphical representation of the multilinear polynomial specifying a (discrete) probability measure [16]. More formally, an SPN is a weighted, rooted and acyclic directed graph where internal nodes are labeled as either sum or product operations and leaves are associated with indicator variables. We assume that every indicator variable appears in at most one leaf node. Every arc from a sum node i to a child j is associated with a nonnegative weight w_{ij}; the remaining arcs have weight one. Given an SPN S and a node i, we denote S^i the SPN obtained by rooting the network at i, that is, by discarding any non-descendant of i (other than i itself). We call S^i the subnetwork rooted at i. If $\mathbf{w}$ are the weights of an SPN S and i is a node, we denote by $\mathbf{w}_i$ the weights in the subnetwork S^i rooted at i, and by w_i the vector of weights w_{ij} associated with arcs from i to its children j.

The *value* of an SPN S at a given configuration λ of its indicator variables, written $S(\lambda)$, is defined recursively in terms of its root node i. If i is a leaf node associated with indicator variable λ_{i,x_i} then $S(\lambda) = \lambda_{i,x_i}$. Else, if i is a product node, then $S(\lambda) = \prod_j S^j(\lambda)$, where j ranges over the children of i. Finally, if i is a sum node then $S(\lambda) = \sum_j w_{ij} S^j(\lambda)$, where again j ranges over the children of i. For example, the value of the SPN in Figure 1 at the configuration $\lambda = (x_1, x_2, \bar{x}_1, \bar{x}_2) = (1, 0, 0, 1)$ is 0.15.

The *scope* of an SPN with a single node (hence, a leaf associated with an indicator variable) is the respective random variable. The scope of an SPN with a root node which is not a leaf is the union of the scopes of the subnetworks rooted at every child of the root. Figure 1 shows an example of an SPN with scope $\{X_1, X_2\}$, where X_1 and X_2 are Boolean variables.

Every joint distribution over categorical random variables can be represented by an SPN. In order to ensure that any SPN computes a valid distribution and its marginals, we impose the following properties:[2]

Completeness: The scopes of children of a sum node are identical;

Decomposition: The scopes of children of a product node are disjoint;

[2]Poon and Domingos originally required only that SPNs satisfy completeness and consistency, a seemingly weaker condition than decomposition [48]. Perhaz et al. [47] later showed that any consistent SPN over discrete random variables can be transformed in an equivalent decomposable and normalized SPN with polynomial effort.

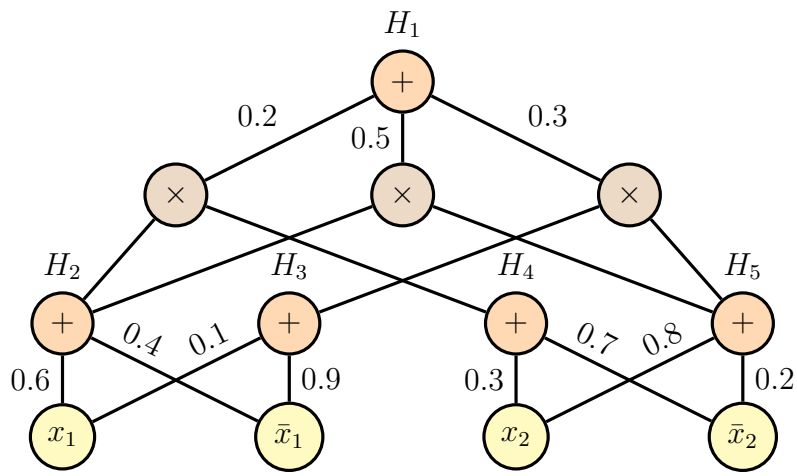


Figure 1: A sum-product network over Boolean random variables X_1 and X_2.

Normalization: The sum of the weights of arcs leaving a sum node is one.

Under the requirements above, every SPN specifies a probability measure $\mathbb{P}$ such that $\mathbb{P}(X = x) = S(\lambda^x)$. More importantly, a marginal probability can be computed from an SPN by setting all indicator variables of the summed out variables to one. That is, let $\mathcal{E} \subseteq \mathcal{V}$ and consider some *evidence* $X_{\mathcal{E}} = e$. Then $\mathbb{P}(X_{\mathcal{E}} = e)$ can be computed as $S(\lambda^e)$ [48]. In other words, it follows that $S(\lambda^e) = \sum_{x \sim e} S(\lambda^x)$, where the sum is performed over all realizations $X = x$ that agree with evidence e.

The evaluation of an SPN (i.e., the computation of its value) for a given configuration λ of the indicator variables can be performed by a bottom-up message propagation scheme where each node sends to its parent its value. The whole procedure takes linear time and space. Conditional probabilities can also be obtained in linear time either by evaluating the network at query and evidence (then dividing the result) or by applying Darwiche's differential approach, that propagates messages up and down through the network [16, 46]. Other inferences such as maximum-a-posteriori inference are however NP-hard to compute or even to approximate [11, 46].

The sum nodes in an SPN can be interpreted as hidden (latent) variables in a mixture model, and the product nodes can be seen as defining context-specific independences [46, 48]. The number of values of the hidden variable corresponding with a sum node is the number of outgoing arcs. For example, the SPN in Figure 1 can be interpreted as specifying a mixture distribution

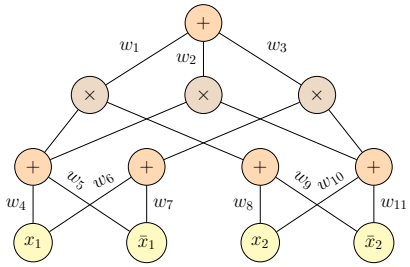
over observed variables X_1, X_2, and hidden variables H_1, H_2, H_3, H_4, H_5. According to the probability measure induced by the network, when conditioned on $H_1 = 1$, X_1 is (probabilistically) dependent on H_2 while independent of H_3, H_4, H_5 and X_2; similarly, conditional on $H_1 = 3$, X_1 is dependent on H_3 while independent of H_2, H_4, H_5 and X_2. This is an example of context-specific independence being represented by an SPN.

Alternatively, an SPN can be interpreted as a bilevel bipartite Bayesian network with an upper layer of latent variables $H_1, \ldots, H_m$ corresponding to sum nodes of the SPN, and a bottom layer of leaf variables $X_1, \ldots, X_n$ corresponding to (scopes of) indicator variables. There is an arc $H_j \to X_i$ if and only if X_i is in the scope of the sum node (associated with) H_j. Each variable H_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) [69]. Note that a variable X_i can have a large number of parents, so that obtaining this Bayesian network is usually impracticable.

A range of algorithms have been devised to "learn" SPNs from data [1, 22, 23, 27, 35, 44, 45, 51, 53, 59]. 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'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 [27]. 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. Alternatively, a fixed structure can be specified (e.g., a random structure), and the weights can be learned to optimize the data log-likelihood, for example, by gradient-based methods or by Expectation-Maximization [26, 46, 48].

3. Credal Sum-Product Networks

Recall that in this work we consider only SPNs that are complete, decomposable and normalized. Let $S_{\mathbf{w}}$ denote an SPN whose weights are $\mathbf{w}$. We can investigate the robustness of the network to perturbations in the parameters (or analogously, to the data from which the parameters were learned) by varying the weights $\mathbf{w}$ inside some fixed space, subject to the constraint that



$$
\left\{ \begin{array}{c}
(w_1, w_2, w_3) = 0.9(0.2, 0.5, 0.3) + 0.1v \\
v = (v_1, v_2, v_3) \geq 0, \quad v_1 + v_2 + v_3 = 1 \\
0.54 \leq w_4 \leq 0.64, \quad w_4 + w_5 = 1 \\
0.09 \leq w_6 \leq 0.19, \quad w_6 + w_7 = 1 \\
0.27 \leq w_8 \leq 0.37, \quad w_8 + w_9 = 1 \\
0.72 \leq w_{10} \leq 0.82, \quad w_{10} + w_{11} = 1
\end{array} \right\}
$$

Figure 2: A credal sum-product network over variables X_1 and X_2.

they still define a (normalized) SPN. To this aim, we define a *Credal Sum-Product Network* (CSPN) as 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 an 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 an SPN. Since each SPN induces a probability measure, the CSPN induces a *credal set*, that is, a (not necessarily convex) set of probability measures [36].

For any real value $0 \leq \epsilon \leq 1$, the ϵ-contamination of a vector u is given by

$$
\mathcal{C}_{u,\epsilon} = \left\{ (1-\epsilon)u + \epsilon v : v_j \geq 0, \sum_j v_j = 1 \right\} . \tag{1}
$$

For example, if $u = (u_1, 1 - u_1)$ is a point in the one dimensional simplex, then ϵ-contamination of u is given by $\mathcal{C}_{u,\epsilon} = \{(w_1, 1 - w_1) \geq 0 : (1-\epsilon)u_1 \leq w_1 \leq (1-\epsilon)u_1 + \epsilon\}$. The simplest form of obtaining a CSPN out of a SPN is by independently ϵ-contaminating each vector of local weights associated with sum nodes. Figure 2 shows a CSPN obtained by ϵ-contamination of the SPN in Figure 1, with $\epsilon = 0.1$.

Just as with SPNs, the sum nodes in an CSPN can be interpreted as latent variables, so that the whole model can be seen as a set of mixture models. Alternatively, we can interpret sum nodes as the latent variables in a bipartite credal network whose leaves are the observable variables. This network is obtained exactly as the Bayesian network for SPNs, except that conditional probability distributions are replaced by conditional credal sets. Note that credal networks obtained in this way form a very special case (and as before

the conversion is usually computationally expensive).

4. Likelihood

The simplest robustness analysis that one can perform with CSPNs is arguably to compute the minimum and maximum values obtained by an induced SPN for a given value λ of the indicator variables: $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$ and $\max_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$ subject to $\mathbf{w} \in \mathcal{C}$. For the sake of readability, we often omit the constraint on the weights when they are optimized; these should always be considered constrained in the appropriate space. When λ^e is consistent with some realization $X_{\mathcal{E}} = e$, this computation corresponds to computing the lower/upper likelihood of evidence $\text{opt}_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e)$ with $\text{opt} \in \{\min, \max\}$. In any case, the computation of minimum and maximum values 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 sum node. That is, visit nodes in reverse topological ordering, evaluating the corresponding expressions based on the type of node. So let $L^i(\lambda)$ be the value computed by the algorithm for node i with children (if any) indexed by j. Then,

$$L^i(\lambda) = \begin{cases} \lambda_{i,x} & \text{if } i \text{ is a leaf with indicator variable } \lambda_{i,x}, \\ \prod_j L^j(\lambda) & \text{if } i \text{ is a product node,} \\ \min_{w_i \in \mathcal{C}_i} \sum_j w_{ij} L^j(\lambda) & \text{if } i \text{ is a sum node.} \end{cases}$$

To see why the above procedure finds the correct value, first consider the simpler case of a tree-shaped CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ with root i. Since the structure is a tree, the subnetworks $S^1, \ldots, S^k$ rooted at the children of node i do not share any weights with each other. Hence, we have that

$$\min_{\mathbf{w} \in \mathcal{C}} S_{\mathbf{w}}(\lambda) = \min_{w_i} \sum_{j=1}^{k} w_{ij} \min_{\mathbf{w}_j} S^j_{\mathbf{w}_j}(\lambda)$$

when i is a sum node, and

$$\min_{\mathbf{w} \in \mathcal{C}} S_{\mathbf{w}}(\lambda) = \prod_{j=1}^{k} \min_{\mathbf{w}_j} S^j_{\mathbf{w}_j}(\lambda)$$

when i is a product node. In either case, the problem of computing the minimum or maximum of a value λ decomposes into the smaller equivalent

problems of computing $\min_{\mathbf{w}_j} S^j_{\mathbf{w}_j}(\lambda)$ for each child j of i. 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* $O(|S|C)$ *time, where* $|S|$ *is the number of nodes and arcs in the network, and* C *is an upper bound on the cost of solving a linear program of the form* $\min_{w_i} \sum_j c_{ij} w_{ij}$ *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). We will derive an algorithm that solves the problem by propagating messages upward in time $O(|S|C)$. Write $L(\lambda)$ to denote the value computed by this algorithm for a CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$, that is, the value of $L^r(\lambda)$ where r is the root of the network S. Start at the leaves; there are no weights associated, so these nodes simply propagate the value of the associated indicator variable in λ as in SPNs. Now consider an internal node i. If i is a product node, then propagate $L^i(\lambda) = \prod_j L^j(\lambda)$, where j ranges over the children of i. Otherwise, suppose i is a sum node, and propagate $\min_{w_i} \sum_j w_{ij} L^j(\lambda)$. Note that this denotes a linear program of the form $\min_{w_i} \sum_j c_{ij} w_{ij}$ subject to $w_i \in \mathcal{C}_i$, where $\mathcal{C}_i$ is a finitely-generated polytope.

To prove that the algorithm is correct, we first show that $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda) \geq L(\lambda)$ by induction on the height h of S. The base case for $h = 0$ is immediate. Assume that the result holds for networks of height $h \geq 0$ or smaller, and consider a network of height $h + 1$ whose root is i. If i is a sum node, then

$$\min_{\mathbf{w}} \sum_j w_{ij} S^j_{\mathbf{w}_j}(\lambda) \geq \min_{w_i} \sum_j w_{ij} \min_{\mathbf{w}_j} S^j_{\mathbf{w}_j}(\lambda) \geq \min_{w_i} \sum_j w_{ij} L^j(\lambda) = L^i(\lambda)\,,$$

where j ranges over the children of i. Similarly, we can show that if i is a product node then

$$\min_{\mathbf{w}} \prod_j S^j_{\mathbf{w}_j}(\lambda) \geq \prod_j \min_{\mathbf{w}_i} S^j_{\mathbf{w}_j}(\lambda) \geq \prod_j L^j(\lambda) = L^i(\lambda)\,.$$

The value computed by the algorithm is also an upper bound on $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$, since every sum node i selects weights $w_i \in \mathcal{C}_i$, and the propagated value is

the value of the corresponding SPN. Thus $L(\lambda) = \min_{\mathbf{w}} S_{\mathbf{w}}(\lambda)$. If we cache the values of each node during propagation, then computing the value of a node takes at most time $O(kC)$, where k is the number of nodes and arcs in the SPN rooted at that node. Hence the total cost of this computation takes time $O(|S|C)$. □

Since linear programs can be solved in polynomial time, the upper and lower bounds can be computed in time polynomial in the size of the input (which includes a description for the local polytopes). This leads to the following result:

Corollary 1. *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. If local polytopes* $\mathcal{C}_i$ *are specified by (finitely many) constraints of the form* $l_{ij} \leq w_{ij} \leq u_{ij}$ *for reals* $l_{ij} \leq u_{ij}$*, then the problem can be solved in time* $O(|S|^2 \log |S|)$*.*

Proof. When local polytopes take the form $l_{ij} \leq w_{ij} \leq u_{ij}$, then the local optimizations $\min_{w_i} \sum_j w_{ij} L^j(\lambda)$ are equivalent to fractional knapsack problems [31], which can be solved in time $O(k \log k)$, where k is the number of children of node i. The overall running time is thus $O(|S| \cdot |S| \log |S|)$. □

In fact, $O(|S|^2 \log |S|)$ is usually a very loose bound if the network has a small number of children per node.

5. Conditional Expectations

A more sophisticated analysis one can carry out with CSPNs is to obtain upper and lower bounds $\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e)$ and $\max_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e)$ on the expected value of some function f of X, conditional on evidence $X_{\mathcal{E}} = e$. Recall that each choice of the weights $\mathbf{w}$ of a CSPN $\{S_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ defines an SPN and hence induces a probability measure $\mathbb{P}_{\mathbf{w}}$. We can therefore use the CSPN to compute bounds on the conditional expectations of a function:

$$\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|X_{\mathcal{E}} = e) = \min_{\mathbf{w}} \sum_x f(x) \mathbb{P}_{\mathbf{w}}(X = x|X_{\mathcal{E}} = e)\,. \tag{2}$$

The equations above are only defined if $\min_{\mathbf{w}} \mathbb{P}(X_{\mathcal{E}} = e) > 0$. We will assume here that if this is not the case then the computation fails with some arbitrary value being returned. Note that as explained in Section 4, verifying whether $\min_{\mathbf{w}} \mathbb{P}(X_{\mathcal{E}} = e) = 0$ takes polynomial time. Note also that we

can focus on the computation of the lower expectation, as the upper expectation can be obtained from $\max_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(f|e) = -\min_{\mathbf{w}} \mathbb{E}_{\mathbf{w}}(-f|e)$. Provided that $\min_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e) > 0$, computing the lower conditional expectation in Equation (2) is equivalent to finding the unique value of μ that solves the equation:

$$\min_{\mathbf{w}} \sum_{x \sim e} [f(x) - \mu] S_{\mathbf{w}}(\lambda^x) = 0\,, \tag{3}$$

where the sum is performed only over assignments x that agree with the evidence e. It turns out that computing such type of inference is intractable (under the common assumptions in complexity theory):

Theorem 2. *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 6, 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 with respect to the network size).

While the general case is NP-hard, there is a polynomial-time algorithm to compute lower conditional expectations when the network obtained by discarding the leaves is a tree and $f : X_q \to \mathbb{Q}$ is a univariate function of a random variable X_q ($q \notin \mathcal{E}$). This type of network topology is particularly common, since it is generated by the most popular family of SPN structure learning algorithms [27, 53, 59]. The algorithm performs a binary search for the value of μ that solves Equation (3). For each step of the binary search, the algorithm traverses the network from the leaves towards the root, and computes the values $\underline{V}^i(\lambda)$ and $\overline{V}^i(\lambda)$ for each node i as follows. If i is a leaf node associated with indicator variable $\lambda_{i,j}$, then

$$\underline{V}^i(\lambda) = \overline{V}^i(\lambda) = \begin{cases} \lambda_{i,j} & \text{if } j \neq q\,, \\ f(j) - \mu & \text{if } j = q\,. \end{cases}$$

If i is a product node then

$$\underline{V}^i(\lambda) = \begin{cases} \prod_j \underline{V}^j(\lambda) & \text{if } X_q \text{ is not in the scope of } i\text{, or if } X_q \text{ is} \\ & \text{in the scope of both } i \text{ and its child } k \\ & \text{and } \underline{V}^k(\lambda) \geq 0\,, \\ \underline{V}^k(\lambda) \prod_{j \neq k} \overline{V}^j(\lambda) & \text{if } X_q \text{ is in the scope of } i \text{ and child } k, \\ & \text{and } \underline{V}^k(\lambda) < 0\,; \end{cases}$$

and

$$\overline{V}^i(\lambda) = \begin{cases} \prod_j \overline{V}^j(\lambda) & \text{if } X_q \text{ is not in the scope of } i\text{, or if } X_q \text{ is} \\ & \text{in the scope of both } i \text{ and its child } k \\ & \text{and } \overline{V}^k(\lambda) \geq 0\,, \\ \overline{V}^k(\lambda) \prod_{j \neq k} \underline{V}^j(\lambda) & \text{if } X_q \text{ is in the scope of } i \text{ and child } k, \\ & \text{and } \overline{V}^k(\lambda) < 0\,. \end{cases}$$

Finally, if i is a sum node then

$$\underline{V}^i(\lambda) = \min_{w_i \in \mathcal{C}_i} \sum_j w_{ij} \underline{V}^j(\lambda) \quad \text{and} \quad \overline{V}^i(\lambda) = \max_{w_i \in \mathcal{C}_i} \sum_j w_{ij} \overline{V}^j(\lambda)\,.$$

The soundness of this algorithm leads to the following result:

Theorem 3. *Computing lower and upper conditional expectations of a univariate (rational-valued) function in* CSPN*s takes at most polynomial time when each internal node has at most one parent.*

Proof. As before, let λ^e be the assignment of indicator variables that is consistent with evidence. By Theorem 1, we can efficiently compute $\max_{\mathbf{w}} S_{\mathbf{w}}(\lambda^e) = \max_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e)$ and $\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda^e) = \min_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e)$, and decide what to return in case any of these is zero. So assume that $\min_{\mathbf{w}} \mathbb{P}_{\mathbf{w}}(X_{\mathcal{E}} = e) > 0$, and let $g_\mu(x_q) = f(x_q) - \mu$. By Equation (3), the lower conditional expectation of f is the unique μ such that

$$\min_{\mathbf{w}} \sum_{x \sim e} g_\mu(x_q) S_{\mathbf{w}}(\lambda^x) = \min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e}) = 0\,,$$

where $\lambda^{x_q,e}$ denotes the configuration of the indicator variables that is consistent with both x_q and e. Since all numbers in the CSPN and in f are

(by assumption) rational, if we can compute $\min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e})$ efficiently, then we can also compute μ efficiently by performing a binary search in the interval $[\min_{x_q} f(x_q), \max_{x_q} f(x_q)]$. So consider a fixed rational μ, and a CSPN whose internal nodes have at most one parent. Call the root of the network 0, and let $1, \ldots, k$ denote the children of the root node. We prove the correctness of the algorithm by induction on the height of the network. That is, assume that for height $h \geq 0$, we have that

$$\underline{V}^0(\lambda^e) = \min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e}) \text{ and } \overline{V}^0(\lambda^e) = \max_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e}),$$

if X_q is in the scope of 0, else

$$\underline{V}^0(\lambda^e) = \min_{\mathbf{w}} S_{\mathbf{w}}(\lambda^e) \quad \text{and} \quad \overline{V}^0(\lambda^e) = \max_{\mathbf{w}} S_{\mathbf{w}}(\lambda^e).$$

Assume also (without loss of generality) that if X_q is in the scope of 0 and 0 is a product node, then only node 1 has also X_q in its scope. If 0 is a product node (of height $h+1$) then, because the scopes of children of product nodes are disjoint, we have that

$$\min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e}) = \min_{\mathbf{w}} \left(\sum_{x_q} g_\mu(x_q) S^1_{\mathbf{w}_1}(\lambda^{x_q,e}) \right) \prod_{j=2}^{k} S^j_{\mathbf{w}_j}(\lambda^e),$$

which equals

$$\left(\min_{w_1} \sum_{x_q} g_\mu(x_q) S^1_{\mathbf{w}_1}(\lambda^{x_q,e}) \right) \prod_{j=2}^{k} \min_{w_j} S^j_{\mathbf{w}_j}(\lambda^e) = \underline{V}^1(\lambda^e) \prod_{j=2}^{k} \underline{V}^j(\lambda^e)$$

when $\underline{V}^1(\lambda^e) = \min_{w_1} \sum_{x_q} g_\mu(x_q) S^1_{\mathbf{w}_1}(\lambda^{x_q,e}) \geq 0$ (where we have used the inductive hypothesis), and equals

$$\left(\min_{w_1} \sum_{x_q} g_\mu(x_q) S^1_{\mathbf{w}_1}(\lambda^{x_q,e}) \right) \prod_{j=2}^{k} \max_{w_j} S^j_{\mathbf{w}_j}(\lambda^e) = \underline{V}^1(\lambda^e) \prod_{j=2}^{k} \overline{V}^j(\lambda^e),$$

if $\underline{V}^1(\lambda^e) = \min_{w_1} \sum_{x_q} g_\mu(x_q) S^1_{\mathbf{w}_1}(\lambda^{x_q,e}) < 0$. If node 0 is a sum node (of height $h+1$) with X_q in its scope, then because the internal graph of the

CSPN is a tree,

$$\min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q,e}) = \min_{w_0} \sum_{j=1}^{k} w_{0j} \min_{\mathbf{w}_j} \sum_{x_q} g_\mu(x_q) S^j_{\mathbf{w}_j}(\lambda^{x_q,e}) \,,$$

which by the inductive hypothesis

$$= \min_{w_0} \sum_{j=1}^{k} w_{0j} \underline{V}^j(\lambda^e) \,.$$

If X_q is not in its scope, then the inductive step is trivial. The base case for $h = 0$ is obtained when 0 is a leaf node associated with indicator variable λ_{j,x_j}. Then,

$$\min_{\mathbf{w}} \sum_{x_q} g_\mu(x_q) S_{\mathbf{w}}(\lambda^{x_q}) = f(x_q) - \mu = \underline{V}^0(\lambda) \,,$$

if X_q is in the scope of 0 (i.e., $j = q$), and otherwise

$$\min_{\mathbf{w}} S_{\mathbf{w}}(\lambda^e) = \lambda^e_{j,x_j} = \underline{V}^0(\lambda^e) = \overline{V}^0(\lambda^e) \,.$$

All these computations can be performed in polynomial time by scheduling computations so that children are computed before their parents. □

Note that the values of $\overline{V}^i(\lambda)$ need only be computed for nodes that do not contain X_q in its scope, and that in this case they equal the values produced by the upper likelihood algorithm (left implicitly in Section 4).

6. Credal Classification

One of the most frequent uses of SPNs is in building probabilistic classifiers, that is, in estimating a probability distribution over class and attribute values, which can then be used to classify objects into classes by maximizing the class conditional probability. For example, Poon and Domingos [48] learned SPNs to predict the missing pixels of an image. Amer and Todorovic [2] used SPNs to classify video snippets according to actions being performed. Villanueva and Mauá [38] learned SPNs for multi-label classification (when an object can be assigned to several class labels).

In order to obtain more robust classifiers, we can replace SPNs with CSPNs. However, since CSPNs define not a single but a set of probability measures, there is no clear criterion to issue a classification. Many criteria have been devised for decision making with imprecise probability models; examples include interval dominance, maximality, e-admissibility and maximin [29]. Here we adopt a very popular criterion, based on the principle of maximality, often called *credal classification* in the context of probabilistic classifiers [65]. We leave the study of robustness analysis under other criteria for future work.

Given a distinguished set of class variables $X_{\mathcal{C}}$, evidence $X_{\mathcal{E}} = e$, and a set of probability measures $\mathcal{M}$, we say that an assignment c_1 for X_C *credally dominates* another assignment c_2 if [65]

$$\min_{\mathbb{P}\in\mathcal{M}}\Big[\mathbb{P}(X_{\mathcal{C}} = c_1, X_{\mathcal{E}} = e) - \mathbb{P}(X_{\mathcal{C}} = c_2, X_{\mathcal{E}} = e)\Big] > 0\,.$$

To put it differently, class c_1 credally dominates class c_2 if $\mathbb{P}(X_{\mathcal{C}} = c_1|X_{\mathcal{E}} = e) > \mathbb{P}(X_{\mathcal{C}} = c_2|X_{\mathcal{E}} = e)$ for all $\mathbb{P} \in \mathcal{M}$ where these conditional probabilities are defined.[3] Note that a class is defined as an assignment of a possibly multi-dimensional vector of class variables (so that in our definition the standard single-label classification is a special case).

In the setting of CSPNs, credal dominance amounts to establishing whether

$$\min_{\mathbf{w}}\Big[S_{\mathbf{w}}(\lambda^{c_1,e}) - S_{\mathbf{w}}(\lambda^{c_2,e})\Big] > 0\,, \tag{4}$$

for any two given classes c_1 and c_2, and evidence e. The following two results establish the complexity of this task:

Theorem 4. *Deciding if a class c_1 credally dominates a class c_2 is in coNP.*

Proof. To prove membership, consider the complementary problem of deciding if

$$\min_{\mathbf{w}}[S_{\mathbf{w}}(\lambda^{c_1,e}) - S_{\mathbf{w}}(\lambda^{c_2,e})] \leq 0\,.$$

If the inequality above is true, then there is a polynomial certificate $\mathbf{w}$ for which $S_{\mathbf{w}}(\lambda^{c_1,e}) - S_{\mathbf{w}}(\lambda^{c_2,e}) \leq 0$. Since we can compute this difference in polynomial time, the problem is in NP. Membership in coNP follows by noticing that class c_1 credally dominates c_2 if and only if the inequality above is false. □

[3] If $\min_{\mathbf{w}} \mathbb{P}(X_{\mathcal{E}} = e) = 0$, then no class credally dominates another class.

Theorem 5. *Deciding if a class c_1 credally dominates a class c_2 is coNP-hard.*

The proof of the above result is obtained by a polynomial-time reduction from the PARTITION problem: given a list of positive integers, determine if there is a partition into two sets with equal sum. The reduction consists in encoding any instance of PARTITION as the minimization of a multilinear function over the weights of the network, where each weight takes either value 0 or 1 corresponding to the inclusion of a certain integer in the first or in the second set of the partition. The multilinear function, obtained as the result of a credal dominance query, is convex with respect to the difference of the sums of the partitions, and attains its minimum value of zero if and only if the corresponding PARTITION problem has a solution. We postpone the full proof until the Appendix, as it is lengthy and rather technical.

Credal classification usually consists in finding all non-dominated classes. As the previous result shows, if the number of classes is unbounded (as in multilabel classification tasks), computing the set of non-dominated classes is coNP-hard. When the number of classes is small, we can perform credal classification by deciding Equation (4) for every pair of classes. The next result shows that when the number of classes is bounded and the network topology is constrained, credal classification can be achieved efficiently:

Theorem 6. *Credal classification (i.e., computing the set of non-dominated classes) can be done in polynomial time in* CSPN*s when each internal node has at most one parent and the number of classes in bounded.*

Proof. Credal classification can be cast as the decision of

$$\max_{\mathbf{w}} \sum_{\lambda} f(\lambda) S_{\mathbf{w}}(\lambda) > 0\,,$$

where $f(\lambda) = 1$ if $\lambda = \lambda^{c_1,e}$, $f(\lambda) = -1$ if $\lambda = \lambda^{c_2,e}$ and $f(\lambda) = 0$ otherwise. This equation describes the computation of an expectation, hence the result follows from Theorem 3. □

7. Assessing Robustness of Classifications

In many real applications of classifiers, practitioners are often interested in a confidence estimate of each classification being issued; this can be used for instance to motivate additional data gathering, to resort to alternative

methods, or simply to suspend judgment and prevent possible catastrophic failures. When using probabilistic classifiers, a commonly adopted approach is to measure the difference between the probability of the most probable and the second most probable class labels. Large differences are used to support a "confident classification", while small differences can support alternative behavior. This approach however cannot distinguish between aleatory uncertainty – that is, when the data supports the thesis that the respective instance is associated with more than a single class with high probability – and vagueness – that is, when there is no sufficient statistical support for issuing a classification.

Following the ideas from De Bock et al. [19], we now provide an alternative, arguably more principled, approach to measure the robustness of classifications made with SPN-based classifiers using CSPNs. So assume an SPN has been learned from data, and used to issue a classification based on the maximum probability class label. Given a value $\epsilon > 0$, we say that a classification is ϵ-robust if the respective class label is not credally dominated by any other class label in the CSPN obtained by ϵ-contamination of the SPN. We define the robustness of a classification as the largest value of ϵ for which the maximum probability class is robust.[4] Intuitively, robustness measures the amount of perturbation that would be necessary to cause the SPN to change its current classification. Note that the same value for robustness would be obtained if we used e-admissibility in lieu of credal dominance [19]. According to Theorem 6, when the number of classes is bounded and the internal graph of the SPN is a tree, we can determine whether the classification is robust in polynomial time; thus we can perform a linear search to find the robustness value in polynomial time.

In the rest of this section, we present empirical results showing that robustness provides a better estimate of classification accuracy than the difference of probabilities for SPN-based classifiers.

7.1. Benchmark Datasets

We first evaluate the ability of using robustness to predict the accuracy of classifications in a collection of benchmark datasets from the UCI Machine Learning Repository [37]. Table 1 contains the characteristics of the datasets used.

[4] De Bock et al. [19] have termed this value the critical perturbation threshold; we prefer the smaller and easier to remember name of robustness.

	Data			Sum Product Networks			
Dataset	Instances	Features	Classes	Nodes	Sum	Prod	Height
Cars	1728	6	4	74	7	8	8
Chess	28056	6	18	3416	187	186	19
Connect4	67557	42	3	42120	105	104	42
Flare	1389	12	3	141	22	21	8
Mushrooms	8416	22	2	1224	40	39	22
Nursery	12960	8	5	885	39	38	19

Table 1: Left: Characteristics of the datasets used in the experiments. Right: Characteristics of the learned SPNs using the corresponding datasets (these are median values over all the experiment repetitions): Number of nodes, sum nodes, product nodes and height.

For each dataset, we run the experiment as follows. We randomly select 10% of the data for testing and use the remaining 90% for training. An SPN is then learned by the LearnSPN algorithm [27] using the training data, and a robustness value is computed for each testing instance (this is performed by generating a CSPN by ϵ-contamination from increasingly larger values of ϵ into small discrete steps until the most probable class is not robust). Note that LearnSPN learns networks whose internal nodes have at most one parent, hence satisfying the conditions of Theorem 6. This procedure is repeated 30 times for each dataset and averages are presented. Only robustness values with at least 10 instances below that value are displayed (this is the reason why some curves do not start at zero).

To verify if robustness distinguishes between accurate and inaccurate predictions, we computed the mean and standard deviation of the distributions of the robustness values grouped according to correct and incorrect classifications. The results are shown in Table 2. We then performed a Wilcoxon rank-sum test to verify if the difference in the groups is statistically significant (i.e., whether the chance of a value of a group being greater than the value of the other group is 1/2 or not). All results are significant (i.e., very small p-values), except for the Flare dataset, where the differences in each group are approximately the same, and standard deviation is relatively high. As we will see from the following experiments, assessing reliability in this dataset is particularly difficult. For comparison, we also applied the same analysis for the difference between the probability of the most probable and second most probable classes. The results are in Table 3. All differences are

Dataset	**Correct** Mean ± Std.dev.	**Incorrect** Mean ± Std.dev.	**Rank-sum** P-value
Cars	0.08 ± 0.04	0.02 ± 0.02	≈ 0
Chess	0.01 ± 0.01	0.01 ± 0.01	≈ 0
Connect4	0.03 ± 0.02	0.02 ± 0.01	≈ 0
Flare	0.46 ± 0.07	0.48 ± 0.06	0.994
Mushrooms	0.21 ± 0.05	0.04 ± 0.08	$5 \cdot 10^{-5}$
Nursery	0.04 ± 0.02	0.02 ± 0.02	≈ 0

Table 2: Mean and standard deviation for the distribution of robustness values grouped by correctly and incorrectly classifications. The last-column informs the p-values for a Wilcoxon rank-sum test.

Dataset	**Correct** Mean ± Std.dev.	**Incorrect** Mean ± Std.dev.	**Rank-sum** P-value
Cars	0.67 ± 0.29	0.22 ± 0.16	≈ 0
Chess	0.16 ± 0.13	0.10 ± 0.10	≈ 0
Connect4	0.54 ± 0.21	0.35 ± 0.21	≈ 0
Flare	0.98 ± 0.02	0.96 ± 0.04	$8 \cdot 10^{-5}$
Mushrooms	0.98 ± 0.04	0.33 ± 0.21	$5 \cdot 10^{-7}$
Nursery	0.62 ± 0.25	0.28 ± 0.22	≈ 0

Table 3: Mean and standard deviation for the distribution of the difference between the probability of the most probable and the second most probable classes, grouped by correctly and incorrectly classifications. The last-column informs the p-values for a Wilcoxon rank-sum test.

statistically significant (even for the Flare dataset).

We then compared the ability to predict accuracy as a function of the robustness value. The curves in Figure 3 show the accuracy (no. of correctly classified instances/no. of instances) of the SPN for instances whose robustness is at most a given value (x-axis). The same idea is applied to create the curves of Figure 4, but instead of the robustness value produced by the CSPN, the difference between the probability of most probable and second most probable classes is used (so the curve shows the accuracy over the instances that have at most that given difference). The values for robustness and probability difference have been normalized (i.e., subtracted from the minimum and divided by its amplitude) to allow for comparison

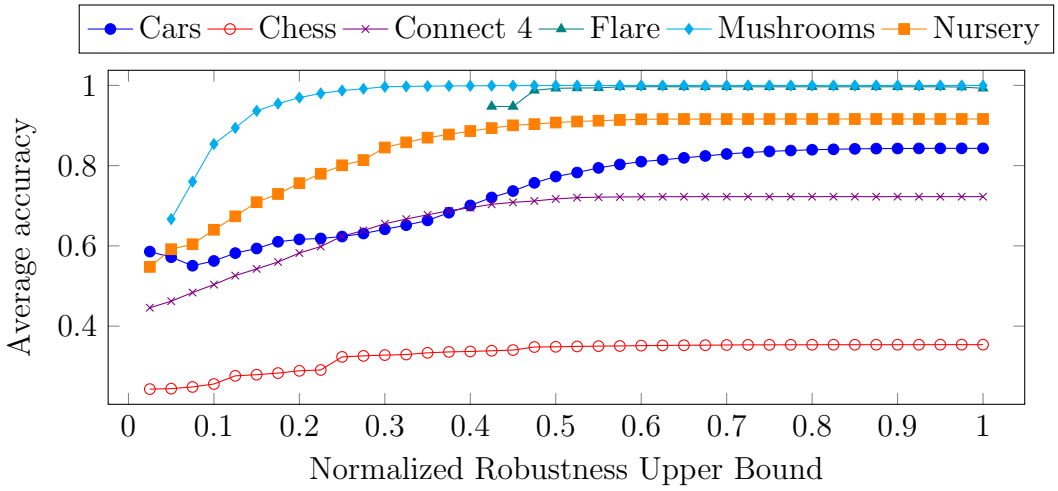


Figure 3: Relation of robustness and accuracy: each point shows the average accuracy of instances that have up to the given robustness.

across datasets. Arguably, the curves obtained with the robustness value are superior to those using the difference in probability in predicting accuracy.

We also computed the *determinacy* versus the threshold of each approach, that is, the proportion of classifications considered unreliable by each method. The results appear in Figures 5 and 6. Overall, robustness is more determinate than probability difference. This is particular so for the datasets Mushrooms and Flare, where using probability difference leads to consider nearly all instances as robust. Recall that robustness values were not able to distinguish accurate and inaccurate instance in the Flare dataset. Note that the correlation of probability difference and accuracy is worse exactly in these datasets. Robustness fairs much better than probability difference in the Mushrooms dataset according to both determinacy and correlation with accuracy. The Flare dataset is particularly challenging. Both methods are highly indeterminate and correlate weakly with accuracy.

It is difficult to use the previous results to directly compare the ability to assess the reliability of classifications of both approaches, since they use different scales. In order to have a direct and more quantitative measure of the relative performance of both approaches, we employed a betting scheme where either robustness or probability difference were used to decided whether or not to enter a bet that rewards the number of classes if the instance is correctly classified by the SPN-based classifier and zero otherwise.

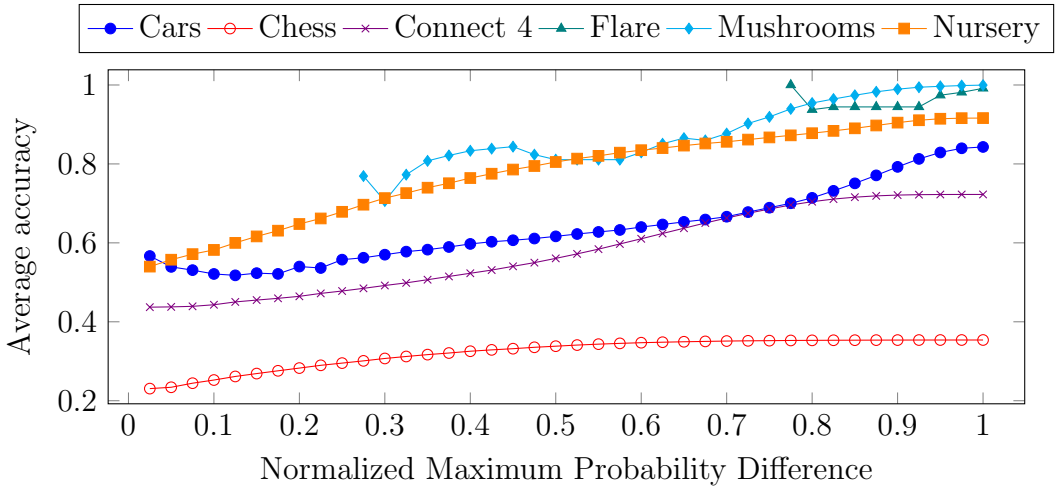


Figure 4: Relation between difference in probability of the best minus second best class and accuracy: each point shows the accuracy over all instances that have up to that given value in difference.

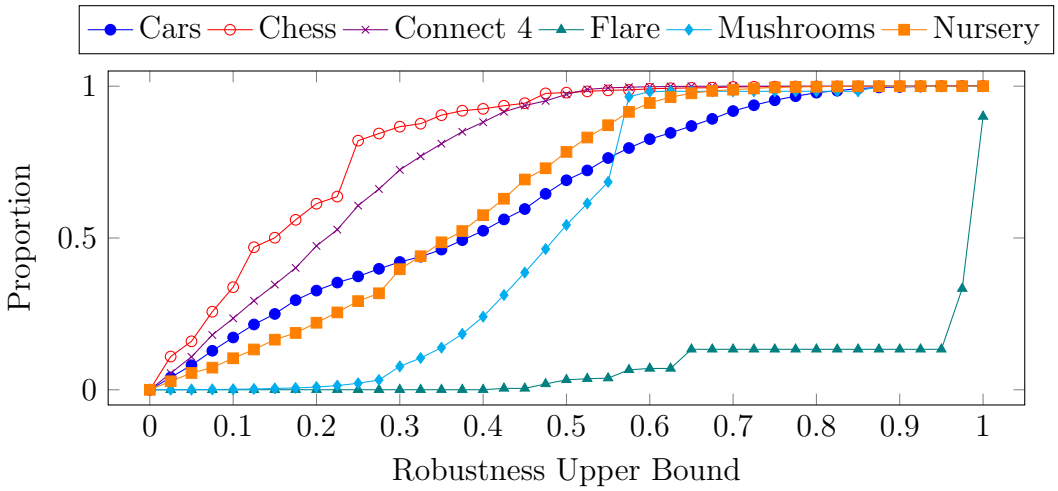


Figure 5: Proportion of instances with robustness less than given threshold.

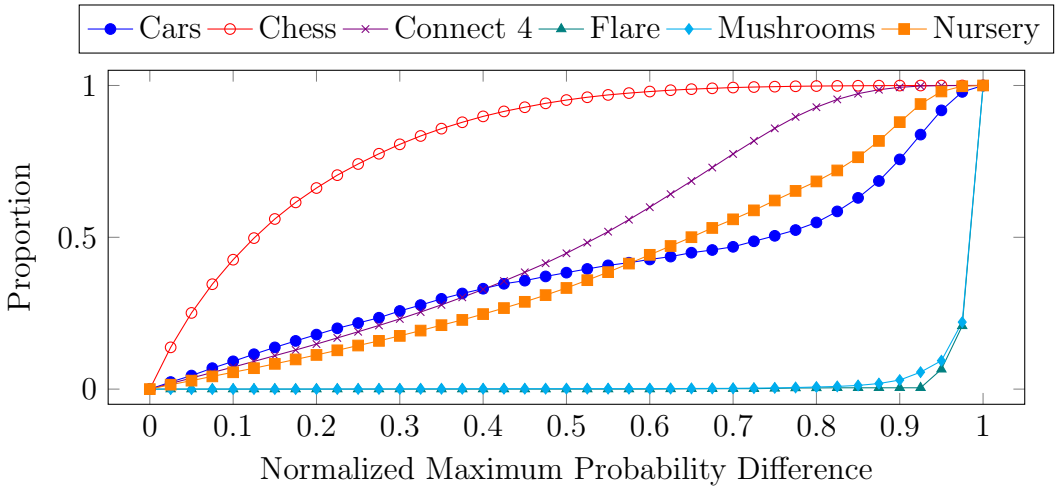


Figure 6: Proportion of instances with probability difference less than given threshold.

A similar scheme has been proposed by Zaffalon et al. [66] to evaluate credal classifiers (i.e., classifiers that can output multiple classes to indicate indeterminacy). We slightly deviate from their approach as our goal here is not to output a credal classification (i.e., a set of non-dominated classes), but to distinguish between reliable and unreliable classifications. Table 4 shows the performance of the different betting strategies. The total profit and number of bets accepted by each method is given in the respective columns Profit and Bets. Since we wanted to remain agnostic to the data, we set thresholds for betting based on the interpretation of each method. For the robustness-based strategy, we adopted a threshold of 5% corresponding to robustness in betting when the parameters are subject to an at most 5% perturbation. For probability difference, we adopted a threshold similar to the uniform class distribution (i.e., one over the number of classes). We have tested with slightly different thresholds (e.g., 1% and 10% for robustness and one over twice the number of classes for probability difference) and the results were qualitatively the same. For the sake of clarity of presentation we do not present the results for the alternative thresholds. For each dataset, the winning strategy (i.e., the one with higher profits) are highlighted in bold. We see from these results that using a threshold-based strategy based on robustness obtained higher or equal profits than either the threshold-based strategy using probability difference and the always bet strategy on all datasets but Nursery. For the Nursery dataset, the strategy based on probability difference obtained a

Dataset	Robustness > 0.05 Profit	Robustness > 0.05 Bets (%)	Prob. Diff. > 1/C Profit	Prob. Diff. > 1/C Bets (%)	Always Bet Profit
Cars	**98.1**	60.5	90.3	78.1	79.4
Chess	**353.7**	0.2	214.7	64.7	179.0
Connect4	**59.3**	11.9	47.0	73.2	38.9
Flare	**66.3**	100.0	**66.3**	100.0	**66.3**
Mushrooms	**33.4**	99.8	**33.4**	99.9	33.3
Nursery	125.2	28.8	**125.3**	88.4	119.4

Table 4: Results of the betting scheme comparison for different strategies. Profit is the total profit of each classifier, Bets is the proportion of bets accepted by each method, and C is the number of classes.

slightly higher profit with a much higher number of bets, while one might prefer to bet fewer to reach the (almost) same profit. In fact, the threshold-based strategy using probability difference accepted a much higher number of bets in nearly all datasets, suggesting that robustness is a more cautious approach.

We have also tested with a slightly different configuration, where gamblers can bet proportional to the probability of the most probable classification. The conclusions were essentially the same and are omitted here.

7.2. Ages of Stars

We also evaluate the use of CSPNs to establish robustness of classifications in the more involved task of predicting stellar age based on stellar spectra features.

Accurately determining the age of a star, for example by asteroseismology, is a time consuming process which takes typically more than a month of (usually) space-based observations [8]. Stellar age can instead be estimated more quickly but less accurately using more easily acquired data such as stellar high-resolution spectra. By establishing the reliability of age predictions based on stellar spectra, we can thus guide the collection of more costly data and improve the effectiveness of stellar age determination.

We repeat the same methodology used in the previous experiments with a dataset of 102 stars (instances) taken from a previous study [7]. The stellar spectra have been recorded via observations using the *NARVAL* [4] and *ESPaDOnS* [24] spectrographs as appropriate for each star, and the ages of the stars have been determined through the work of the Kepler mission [5].

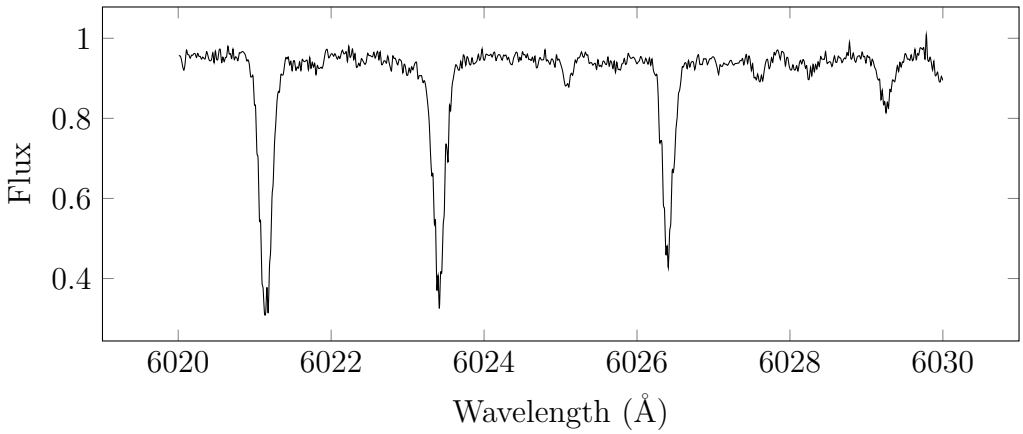


Figure 7: Absorption lines in stellar spectrum from 6020 to 6030 Ångström.

To simplify the task, we discretize the ages into young and old by the median age value (thus, the age labels are uniformly distributed in the dataset, that is, 50% of stars are labeled young/old).

After standard pre-processing to align the spectra, the software ARES (Automatic Routine for line Equivalent widths in stellar Spectra) [58] was used to locate and measure sets of commonly found absorption lines for each star. Absorption lines, visible as steep decreases in the spectrum, give information about trace elements in the stellar atmosphere and other properties such as the atmospheric temperature and pressure. Figure 7 depicts and example of a fragment of a stellar spectrum with three clear absorption lines.

We use the equivalent width or area of each absorption line found in a star's spectrum to produce a vector of 4947 features for each star [58]. The features were selected because of their importance – their wavelengths can be predicted from atomic physics. Each feature was then discretized into four quartiles (so our resulting dataset contains 4947 discrete features and a binary class).

The SPN learned from (the training dataset portion) had in median 17266 nodes, of which 38 were sum nodes and 39 were product nodes; the median height of the networks was 11. The SPN-based classifier achieves an accuracy of 67%, which is comparable to previous results using this dataset and different classifiers. The relatively low accuracy suggests that even distinguishing between young and old stars is already a difficult problem.

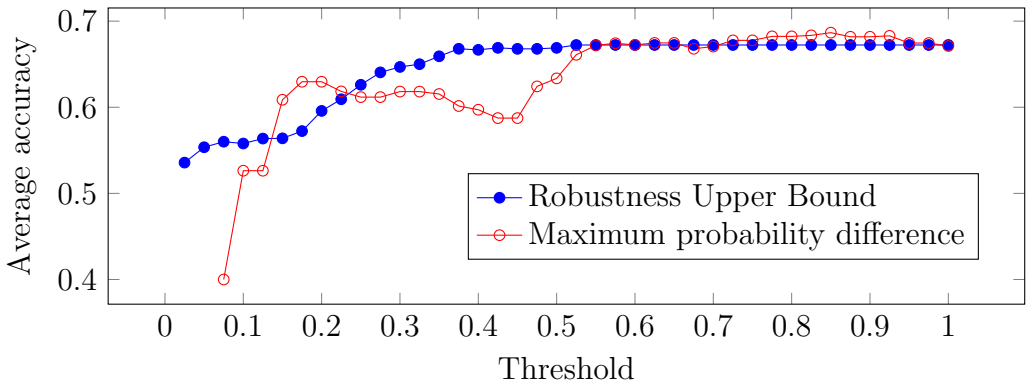


Figure 8: Accuracy against maximum robustness value and maximum difference between the probability of most probable and second most probable classes.

We compare robustness as defined previously and the difference in probability of the most probable and the second most probable classes in predicting accuracy of a classification (i.e., labeling a star as either young or old). The result is shown in Figure 8. As can be seen from the plots, there is a much stronger correlation of small robustness value and low accuracy than a small difference between best and second best probabilities. Higher values of robustness on the other end of the scale did not display a clear correlation with accuracy, and this also the case with the probability difference. This might be justified by the particular composition of dataset not having enough instances beyond a certain threshold to make a fair conclusion. As we see in the figure, the relation of robustness and accuracy is monotonic, while the relation of the difference in probabilities and accuracy is not, which also indicates that the former is a better estimate of classification accuracy than the latter. For this dataset, robustness values on instances that were correctly classified have mean 0.05 with standard deviation 0.04, while mean is 0.04 with the same standard deviation for incorrectly classified instances. In spite of that, the Wilcoxon rank-sum test produces a p-value of $7 \cdot 10^{-5}$, indicating a strong difference between the two groups. The same is not observed with the probability difference, which has mean 0.37 (standard deviation 0.19) for correctly classified instances and mean 0.34 (standard deviation 0.22) for incorrectly classified instances, and p-value 0.151 for the same test (hence not as significant a difference between correctly and incorrectly classified instances).

We also used the betting scheme to compare the different approaches. The

strategy to bet only if robustness is greater than 0.05 obtains a gain of 20.0 and participates in approximately 37.3% of the instances (gambles), while the strategy to bet only if the probability difference is higher than 0.5 obtains a gain of 11.0 and participates in 21.7% of the gambles. For comparison, taking all bets leads to a total gain of 11.8. Unlike the experiments with datasets from UCI, the robustness-based threshold for this data was able to obtain a higher gain participating in a higher number of bets. This might be caused by the high dimensionality of the problem relative to the dataset size (4947/150), suggesting that robustness is particularly more effective than probability difference in situations of severe scarceness of data (which is after all our main motivation here).

8. Conclusion

Sum-product networks are tractable probabilistic graphical models that have shown competitive results in many machine learning tasks. As with other probabilistic models, conclusions draw from sum-product networks are often sensitive to small perturbations in the numerical parameters, indicating lack of statistical support. 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 results 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 such models. We also showed in a realistic task of predicting star age how credal sum-product networks can improve classification accuracy.

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 the internal nodes have at most one parent and the test is over a single variable. In fact, we presented a polynomial-time 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.

Of course, there are other type of inferences that we have not addressed here. For example, one might be interested in obtaining bounds for the likelihood of a dataset induced by a credal sum-product network. This might be used to select more robust structures during learning from data. One might also be interested in obtaining the most likely configuration (given some evidence) under the most pessimistic scenario (i.e., following a maximin strategy).

We have only considered discrete random variables. Sum-product networks have already been extended to allow for continuous random variables [27]. In principle, we could extend the current framework to cope with continuous variables if we assumed that imprecision only appears on weights, so that precise densities are associated with the leaves. Exploiting such a strategy is left for the future.

Our experiments here, however promising, are still 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.

Finally, an alternative to the imprecise probabilistic approach taken here would be to adopt a full Bayesian approach, by integrating out the numerical parameters (weights) given a suitable prior distribution [68]. This creates an undesired dependency on the choice of prior distribution. In fact, one could argue that our approach consists in a sensitivity analysis of the Bayesian approach, one that is more robust to the specification of priors. It remains as future work an empirical comparison with a Bayesian approach.

Acknowledgments

This work was partially supported by the Brazilian National Research Council (CNPq) grants no. 308433/2014-9, 303920/2016-5 and 420669/2016-7 and the São Paulo Research Foundation (FAPESP) grants no. 2016/01055-1 and 2016/18841-0.

Appendix A. Proof of coNP-hardness of credal classification

We now present the proof of coNP-hardness of credal classification, stated in Section 6.

Proof of Theorem 5. We prove hardness by a reduction from the NP-hard problem PARTITION: Given a set of positive integers $z_1, \ldots, z_n$, decide if

there is a set $\mathcal{S} \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in \mathcal{S}} z_i = \sum_{i \notin \mathcal{S}} z_i$. First note that we can scale the integers to become rationals in the unit interval without affecting the complexity of the problem. So let $v_i = 2z_i/z$, where $z = \sum_i z_i$. Then set $\mathcal{S}$ solves the original problem if and only if $\sum_{i \in \mathcal{S}} v_i = 1$.

The strategy of the reduction is to encode an instance of PARTITION as the minimization of a multilinear function over the weights of the network, where each weight corresponds to a choice of selecting or not an integer z_i to be part of the set $\mathcal{S}$. The multilinear function, obtained as the result of a credal dominance query, is convex with respect to the quantity $\sum_{i \in \mathcal{S}} v_i$, and attains its minimum value of zero if and only if the corresponding PARTITION problem has a solution.

So given an instance of PARTITION, construct a CSPN as in Figure A.9. To avoid cluttering, we depict every leaf duplicated in the figure (so the network is not really a tree, but only its internal graph). The variables $X_1, \ldots, X_n$ denote binary class variables, and the variables $X_{n+1}, \ldots, X_{2n}$ denote binary evidence variables fixed at $X_{n+i} = 1$, for $i = 1, \ldots, n$. The network has a sum node as root, with two sub-networks. The left subnetwork S^{th} models a fully independent distribution over all variables, with a uniform distribution over $X_1, \ldots, X_n$ and a degenerate distribution over $X_{n+1}, \ldots, X_{2n}$. This subnetwork is used to impose an additive constant to the overall value of the network, for reasons that will become clear later. So focus on the right subnetwork S^0 for the moment. This network encodes the PARTITION problem as the product of subnetworks $S^1, \ldots, S^n$. Each one of these subnetworks S^i, with $i = 1, \ldots, n$, has scope $\{X_i, X_{n+i}\}$ and encodes the trade-off between adding item z_i to the set $\mathcal{S}$ or adding it to the complement of $\mathcal{S}$. The weights w_{i1} vary in $[0, 1]$ (with $w_{i2} = 1 - w_{i1}$) and represent such a choice: $w_{i1} = 1$ adds z_i to $\mathcal{S}$. The nodes labeled as e_i denote two-layer sub-networks with a sum node as root and indicators x_{n+i} and $\bar{x}_{n+i}$. The weights of these subnetworks are specified such that their value when evaluated with evidence $X_{n+i} = 1$ is the number associated with the edge from the product node to the respective node e_i in the figure (which is 2^{-2v_i}, 2^{-v_i} or 1). This way, when $z_i \in \mathcal{S}$ then the network S^i contributes with a value of $(2^{-2v_i} x_i + 2^{-v_i} \bar{x}_i)/2$ to the product in S^0; and when $z_i \notin \mathcal{S}$ then S^i contributes with $(x_i + \bar{x}_i)/2$.

Let c_1 be the class that assigns $X_i = 1$ for $i = 1, \ldots, n$, and c_2 be the class that assigns $X_i = 0$ for $i = 1, \ldots, n$. One can easily check that $S^{i_1}(\lambda^{c_1,e}) = 2^{-2v_i}/2$, $S^{i_1}(\lambda^{c_2,e}) = 2^{-v_i}/2$ and $S^{i_2}(\lambda^{c_1,e}) = S^{i_2}(\lambda^{c_2,e}) = 1/2$. Thus, we have

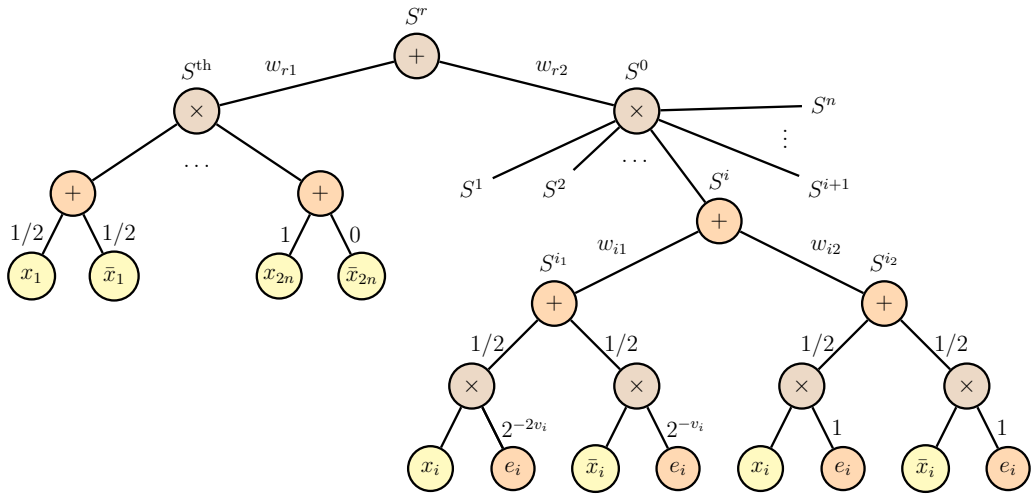


Figure A.9: Fragment of the sum-product network used to solve PARTITION. We duplicate leaves for the sake of clarity.

that

$$\min_{\mathbf{w}} \left(S^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e})\right) = \frac{1}{2^n} \min_{\mathbf{w}} \left(\prod_{i=1}^{n} \left[w_{i1} 2^{-2v_i} + (1 - w_{i1})\right] - \prod_{i=1}^{n} \left[w_{i1} 2^{-v_i} + (1 - w_{i1})\right] \right) .$$

The last term in the equation above defines a multilinear program over the weights; hence the solution lies at the boundary of the feasible set [25, Proposition 2.1]. In our case, this is achieved by setting each weight w_{i1} to either 0 or 1. Hence,

$$\min_{\mathbf{w}} \left(S^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e})\right) = \min_{\mathbf{w}} 2^{-n} [2^{-2\sum_i w_{i1} v_i} - 2^{-\sum_i w_{i1} v_i}]$$

Define $t = 2^{-\sum_i w_{i1} v_i}$. Then, the objective of the minimization on the right can be rewritten as the second-order polynomial on t:

$$2^{-n}(t^2 - t) = 2^{-n} t(t - 1) .$$

This polynomial is convex and achieves its minimum at $t = 1/2$, which corresponds to $\sum_i w_{i1} v_i = 1$, or equivalently, to $\sum_{i \in \mathcal{S}} v_i = 1$, where $\mathcal{S} = \{i :$

$w_{i1} = 1\}$ (see Figure A.10 for an example of such a polynomial). Thus, if such a set exists (i.e., the PARTITION problem is a yes-instance), then

$$\min_{\mathbf{w}} \left(S^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e}) \right) = -2^{-(n+2)} .$$

Now if no such set $\mathcal{S}$ exists, then $|\sum_{i\in\mathcal{S}} z_i - \sum_{i\notin\mathcal{S}} z_i| \geq 1$ for any set $\mathcal{S}$, because z_i are integers. Hence, in this case, the minimum is obtained at the $\mathbf{w}$ such that $|1 - \sum_i w_{i1} v_i| \geq 1/z$. It follows that

$$\begin{aligned}\min_{\mathbf{w}} \left(S^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e}) \right) \geq \\ 2^{-n} \min\{2^{-(1+1/z)}(2^{-(1+1/z)} - 1), 2^{-(1-1/z)}(2^{-(1-1/z)} - 1)\} \\ = 2^{-n} 2^{-(1-1/z)}(2^{-(1-1/z)} - 1) > -2^{-(n+2)} .\end{aligned}$$

The first inequality follows from the integer gap in the sum of the numbers. The equality can be obtained by analyzing the derivative of the difference of $2^{-(1-1/z)}(2^{-(1-1/z)} - 1)$ and $2^{-(1+1/z)}(2^{-(1+1/z)} - 1)$; one can check that this derivative is monotonically increasing, hence the difference is always positive.[5] Finally, the last inequality follows since $2^{-(1-1/z)}(2^{-(1-1/z)} - 1)$ is monotonically decreasing and achieves its minimum at $z \to \infty$. Therefore, deciding whether $\min_{\mathbf{w}} \left(S^0_{\mathbf{w}}(\lambda^{c1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e}) \right) \leq -2^{-(n+2)}$ solves PARTITION.

There are two issues to be fixed in order to turn this result into a proof of coNP-hardness of credal dominance. First, credal dominance is defined as verifying if the minimum difference between the values of two classes is non-positive, but our threshold is currently $-2^{-(n+2)}$. Second, the encoding of the weights 2^{-v_i} requires exponential space/time in the size of the encoding of PARTITION (which requires only the representation e.g. in binary if integers $z_1, \ldots, z_n$), so the reduction is not polynomial. We start by addressing the latter issue.

So approximate each number 2^{-v_i} by a rational $r_i \geq 2^{-v_i}$ such that $r_i - 2^{-v_i} \leq \epsilon$ and $r_i^2 - 2^{-2v_i} \leq \epsilon$, for some small $\epsilon > 0$ (we will explain how to specify such rationals in polynomial time later). Call this new network $\tilde{S}^r$,

[5] Let $h(x) = 2^{-(1-x)}(2^{-(1-x)} - 1) - 2^{-(1+x)}(2^{-(1+x)} - 1)$. Then $h'(x) = 2^{-1-2x}(-1 + 2^x)^2(1 + 2^x + 2^{2x}) \ln(2)$, which is positive for any value of $x \neq 0$. Also, we have that $h(1/2) > 0$. The argument follows by taking $x = 1/z \in [0, 1/2]$.

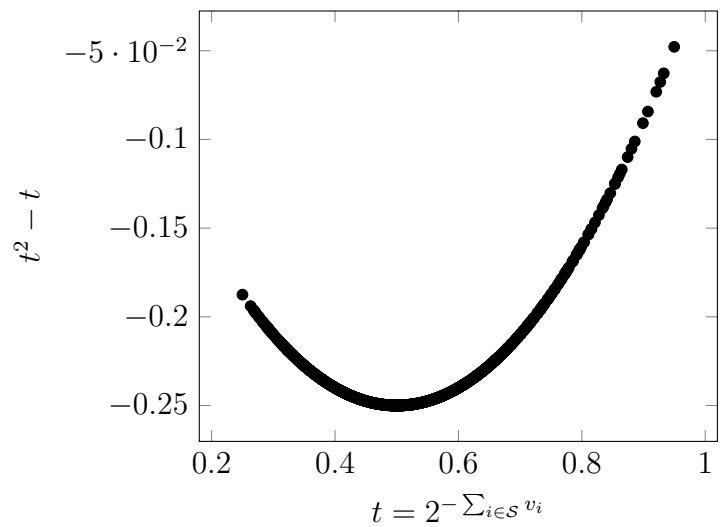


Figure A.10: The polynomial used in the proof of Theorem 5 for a random instance of PARTITION with 10 numbers. Each point plots the value of t for a subset $\mathcal{S}$ of the integers.

and its subnetworks similarly (e.g., $\tilde{S}^0, \tilde{S}^i$, $\tilde{S}^{i_1}$). We have that

$$\begin{aligned} 0 \leq \tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_1,e}) &= \frac{1}{2^n}\left(\prod_i r_i^{2w_{i1}} - 2^{-2\sum_i w_{i1}v_i}\right) \\ &\leq 2^{-n}\left(\prod_i [2^{-2w_{i1}v_i} + \epsilon] - 2^{-2\sum_i w_{i1}v_i}\right) \\ &= 2^{-n} \sum_{k \in \{0,1\}^n, k \neq \mathbf{1}} \prod_i 2^{-2k_i w_{i1} v_i} \epsilon^{1-k_i} < \epsilon\,. \end{aligned}$$

Similarly, one can show that

$$0 \leq \tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e}) < \epsilon\,.$$

Hence,

$$\begin{aligned} &\left|[\tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e})] - [S^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e})]\right| \\ &\qquad = \left|[\tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - S^0_{\mathbf{w}}(\lambda^{c_1,e})] - [\tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e}) - S^0_{\mathbf{w}}(\lambda^{c_2,e})]\right| < \epsilon\,. \end{aligned}$$

Thus, to decide PARTITION we need to select ϵ such that

$$2^{-n}2^{-(1-1/z)}(2^{-(1-1/z)} - 1) - \epsilon > -2^{-(n+2)} + \epsilon\,,$$

or, equivalently, that

$$\epsilon < 2^{-n-1}[2^{-(1-1/z)}(2^{-(1-1/z)} - 1) + 2^{-2}]\,.$$

The Taylor expansion of the term inside the square brackets around $1/z \to 0$ is $\sum_{i=1}^{\infty}(2^i - 2)\ln(2)^i/(4i!)z^{-i}$. Hence, by truncating the series at $i = 3$ and using $1/2 < \ln(2)$, we get

$$2^{-4}z^{-2} + 2^{-8}z^{-3} < 2^{-(1-1/z)}(2^{-(1-1/z)} - 1) + 2^{-2}\,.$$

Let b denote the size of a reasonable binary encoding of PARTITION (e.g., as the one described in the Footnote 5 in [42]). Assuming b encodes each number using at least $\lceil \log z_i \rceil$ bits, we have that $z < 2^b$ and $b > 2$ (note that $z \leq n \max_i z_i$). Thus,

$$2^{-8b} < 2^{-4-2b} + 2^{-8-3b} < 2^{-4}z^{-2} + 2^{-8}z^{-3}\,.$$

Thus, by selecting e.g. $\epsilon = 2^{-10b}$ we can decide PARTITION in polynomial time by checking whether $\min_{\mathbf{w}} \tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e}) \leq -2^{-(n+2)} + \epsilon$. The specification of values r_i can be done in polynomial time by Taylor expanding 2^{-v_i}, since the required bit precision ($10b$) is polynomial in the input size b.

It remains to solve the first issue, that is, to decide by comparing only the sign of the difference. To fix that, we can do a simple manipulation of the result of our network using node root S^r and the left sub-tree:

$$\begin{aligned}
\tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e}) > -2^{-(n+2)} + \epsilon \iff \\
w_{r1} + w_{r2}\left[\tilde{S}^0_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^0_{\mathbf{w}}(\lambda^{c_2,e})\right] > w_{r1} + w_{r2}(-2^{-(n+2)} + \epsilon) \iff \\
\tilde{S}^r_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^r_{\mathbf{w}}(\lambda^{c_2,e}) > w_{r1} + w_{r2}(-2^{-(n+2)} + \epsilon)
\end{aligned}$$

Thus, in order to check only the sign and since $w_{r1} = 1 - w_{r2}$, we need

$$1 - w_{r2} + w_{r2}(-2^{-(n+2)} + \epsilon) = 0 \iff w_{r2} = \frac{1}{1 - (-2^{-(n+2)} + \epsilon)}\,.$$

Hence, we can decide PARTITION by verifying if $\min_{\mathbf{w}} \tilde{S}^r_{\mathbf{w}}(\lambda^{c_1,e}) - \tilde{S}^r_{\mathbf{w}}(\lambda^{c_2,e}) > 0$, and returning "yes" if the latter is false and "true" otherwise. □

References

[1] T. Adel, D. Balduzzi, 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, 2015, pp. 32–41.

[2] M. R. Amer, S. Todorovic, Sum product networks for activity recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (4) (2016) 800–813.

[3] T. Augustin, F. P. A. Coolen, G. De Cooman, M. C. M. Troffaes, Introduction to Imprecise Probabilities, John Wiley and Sons, 2014.

[4] M. Aurière, Stellar polarimetry with NARVAL, EAS Publications Series 9 (2003) 105.

[5] G. Basri, W. J. Borucki, D. Koch, The Kepler mission: A wide-field transit search for terrestrial planets, New Astronomy Reviews 49 (7–9) (2005) 478–485.

[6] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence, 1996, pp. 115–123.

[7] H. Bruntt, S. Basu, B. Smalley, W. J. Chaplin, G. A. Verner, T. R. Bedding, Others, Accurate fundamental parameters and detailed abundance patterns from spectroscopy of 93 solar-type Kepler targets, Monthly Notices of the Royal Astronomical Society 423 (1) (2012) 122–131.

[8] W. J. Chaplin, S. Basu, D. Huber, A. Serenelli, L. Casagrande, V. Silva Aguirre, Others, Asteroseismic fundamental properties of solar-type stars observed by the NASA Kepler mission, The Astrophysical Journal Supplement 210 (1) (2014) 22.

[9] M. Chavira, A. Darwiche, Compiling Bayesian networks with local structure, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, 2005, pp. 1306–1312.

[10] W.-C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, K. M. A. Chai, Language modeling with sum-product networks, in: Proccedings of the 15th Annual Conference of the International Speech Communication Association, 2014, pp. 2098–2102.

[11] D. Conaty, D. D. Mauá, C. P. de Campos, Approximation complexity of maximum a posteriori inference in sum-product networks, in: Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, 2017, pp. 322–331.

[12] F. G. Cozman, Credal networks, Artificial Intelligence 120 (2) (2000) 199–233.

[13] F. G. Cozman, Graphical models for imprecise probabilities, International Journal of Approximate Reasoning 39 (2-3) (2005) 167–184.

[14] F. B. S. Dalmao, T. Pitassi, Value elimination: Bayesian inference via backtracking search, in: Proceedings of the Uncertainty in Artificial Intelligence, 2003, pp. 20–28.

[15] A. Darwiche, A differential approach to inference in Bayesian networks, in: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, 2000, pp. 123–132.

[16] A. Darwiche, A differential approach to inference in Bayesian networks, Journal of the ACM 50 (3) (2003) 280–305.

[17] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, 2009.

[18] A. Darwiche, G. M. Provan, Query DAGs: A practical paradigm for implementing belief-network inference, in: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, 1996, pp. 203–210.

[19] J. de Bock, A. Antonucci, C. P. de Campos, Global sensitivity analysis for MAP inference in graphical models, in: Neural Information Processing Systems, 2014, pp. 2690–2698.

[20] C. P. de Campos, F. G. Cozman, Inference in credal networks through integer programming, in: International Symposium on Imprecise Probability: Theories and Applications, 2007, pp. 145–154.

[21] R. Dechter, Bucket elimination: A unifying framework for reasoning, Artificial Intelligence 113 (1–2) (1999) 41–85.

[22] A. Dennis, D. Ventura, Learning the architecture of sum-product networks using clustering on variables, in: Advances in Neural Information Processing Systems 25, 2012, pp. 2042–2050.

[23] A. Dennis, D. Ventura, Greedy structure search for sum-product networks, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 932–938.

[24] J.-F. Donati, ESPaDOnS: An echelle spectropolarimetric device for the observation of stars at CFHT, in: Astronomical Society of the Pacific Conference Proceedings, San Francisco, 2003, p. 41.

[25] R. F. Drenick, Multilinear programming: Duality theories, Journal of Optimization Theory and Applications 81 (2) (1994) 421–422.

[26] R. Gens, P. Domingos, Discriminative learning of sum-product networks, in: Advances in Neural Information Processing Systems 25, 2012, pp. 3239–3247.

[27] R. Gens, P. Domingos, Learning the structure of sum-product networks, in: Proceedings of the Thirtieth International Conference on Machine Learning, 2013, pp. 873–880.

[28] D. Heckerman, A tractable inference algorithm for diagnosing multiple diseases, in: Proceedings of the Fifteeth Conference on Uncertainty in Artificial Intelligence, 1989, pp. 163–171.

[29] N. Huntley, R. Hable, M. C. M. Troffaes, Decision making, chap. 8, John Wiley and Sons, 2014, pp. 190–206.

[30] D. Koller, N. Friedman, Probabilistic Graphical Models, The MIT press, 2009.

[31] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, Algorithms and Combinatorics, Springer, 2012.

[32] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Transactions on Information Theory 47 (2) (2001) 498–519.

[33] D. Larkin, R. Dechter, Bayesian inference in the presence of determinism, in: Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, 2003.

[34] S. Lauritzen, D. Spiegelhalter, Local computation with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistics Society: Series B 50 (2) (1988) 157–224.

[35] S.-W. Lee, C. Watkins, B.-T. Zhang, Non-parametric Bayesian sum-product networks, in: ICML Workshop on Learning Tractable Probabilistic Models, vol. 32, 2014.

[36] I. Levi, The Enterprise of Knowledge, MIT Press, London, 1980.

[37] M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[38] J. V. Llerena, D. D. Mauá, On using sum-product networks for multi-label classification, in: Proceedings of the 6th Brazilian Conference on Intelligent Systems, 2017, pp. 25–30.

[39] D. D. Mauá, F. G. Cozman, D. Conaty, C. P. de Campos, Credal sum-product networks, in: Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications, 2017, pp. 205–216.

[40] D. D. Mauá, C. P. de Campos, A. Benavoli, A. Antonucci, Probabilistic inference in credal networks: New complexity results, Journal of Artificial Intelligence Research 50 (2014) 603–637.

[41] D. D. Mauá, C. P. de Campos, M. Zaffalon, Updating credal networks is approximable in polynomial time, International Journal of Approximate Reasoning 53(8) (2012) 1183–1199.

[42] D. D. Mauá, C. P. de Campos, M. Zaffalon, On the complexity of solving polytree-shaped limited memory influence diagrams with binary variables, Artificial Intelligence 205 (2013) 30–38.

[43] A. Nath, P. Domingos, Learning tractable probabilistic models for fault localization, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 1294–1301.

[44] R. Peharz, B. C. Geiger, F. Pernkopf, Greedy part-wise learning of sum-product networks, in: Machine Learining and Knowledge Discovery in Databases, vol. 8189 LNAI, 2013, pp. 612–627.

[45] R. Peharz, R. Gens, P. Domingos, Learning selective sum-product networks, in: ICML Workshop on Learning Tractable Probabilistic Models, vol. 32, 2014.

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

[47] R. Peharz, S. Tschiatschek, F. Pernkopf, P. Domingos, On theoretical properties of sum-product networks, in: Proceedings of the 18th International Con- ference on Artificial Intelligence and Statistics, 2015, pp. 744–752.

[48] H. Poon, P. Domingos, Sum-product networks: A new deep architecture, in: Proc. 27th Conf. on Uncertainty in Artif. Intell., 2011, pp. 337–346.

[49] A. Pronobis, R. P. N. Rao, Learning deep generative spatial models for mobile robots, in: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017, pp. 755–762.

[50] A. Pronobis, F. Riccio, R. P. N. Rao, Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments, in: ICAPS 2017 Workshop on Planning and Robotics, 2017.

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

[52] F. Rathke, M. Desana, C. Schnörr, Locally adaptive probabilistic models for global segmentation of pathological oct scans, in: Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention, 2017, pp. 177–184.

[53] A. Rooshenas, D. Lowd, Learning sum-product networks with direct and indirect variable interactions, in: Proceedings of the 31th International Conference on Machine Learning, 2014, pp. 710–718.

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

[55] T. Sang, P. Beame, H. Kautz, Performing Bayesian inference by weighted model counting, in: Proceedings of the 20th AAAI Conference, 2005, pp. 475–481.

[56] S. Sanner, D. A. McAllester, Affine algebraic decision diagrams and their application to structured probabilistic inference, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, 2005, pp. 1384–1390.

[57] B. M. Sguerra, F. G. Cozman, Image classification using sum-product networks for autonomous flight of micro aerial vehicles, in: Proceedings of the Fifth Brazilian Conference on Intelligent Systems, 2016, pp. 139–144.

[58] S. G. Sousa, N. C. Santos, G. Israelian, M. Mayor, M. J. P. F. G. Monteiro, A new code for automatic determination of equivalent widths: Automatic routine for line equivalent widths in stellar spectra (ARES), Astronomic and Astrophysics 469 (2007) 783–791.

[59] A. Vergari, N. Di Mauro, F. Esposito, Simplifying, regularizing and strengthening sum-product network structure learning, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2015, pp. 343–358.

[60] J. Vomlel, Exploiting functional dependence in Bayesian network inference, in: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, 2002, pp. 528–535.

[61] J. Vomlel, P. Tichavský, Probabilistic inference with noisy-threshold models based on a CP tensor decomposition, International Journal of Approximate Reasoning 55 (4).

[62] M. J. Wainwright, M. I. Jordan, Graphical models, exponential families, and variational inference, Foundations and Trends in Machine Learning 1 (2008) 1–305.

[63] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.

[64] J. S. Yedidia, W. T. Freeman, Y. Weiss, Constructing free-energy approximations and generalized belief propagation algorithms, IEEE Transactions on Information Theory 51 (7) (2005) 2282–2312.

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

[66] M. Zaffalon, G. Corani, D. D. Mauá, Evaluating credal classifiers by utility-discounted predictive accuracy, International Journal of Approximate Reasoning 53 (8) (2012) 1282–1301.

[67] N. L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, J. Artif. Intell. Res. 5 (1996) 301–328.

[68] H. Zhao, T. Adel, G. Gordon, B. Amos, Collapsed variational inference for sum-product networks, in: Proceedings of The 33rd International Conference on Machine Learning, vol. 48 of Proceedings of Machine Learning Research, 2016, pp. 1310–1318.

[69] H. Zhao, M. Melibari, P. Poupart, On the relationship between sum-product networks and Bayesian networks, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 116–124.

[70] K. Zheng, A. Pronobis, R. P. N. Rao, Learning graph-structured sum-product networks for probabilistic semantic maps, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 4547–4555.