International Journal of Approximate Reasoning 126 (2020) 158–180

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Efficient algorithms for robustness analysis of maximum a posteriori inference in selective sum-product networks

Julissa Villanueva Llerena *, Denis Deratani Mauá

Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Brazil

ARTICLE INFO

Article history:
Received 30 November 2019
Received in revised form 21 July 2020
Accepted 28 July 2020
Available online 12 August 2020

Keywords:
Robust statistics
Sensitivity analysis
Sum-product networks
Tractable probabilistic models

ABSTRACT

Sum-Product Networks (SPN) are deep probabilistic models with demonstrated excellent performance in several machine learning tasks. As with many other probabilistic models, performing Maximum-A-Posteriori inference in SPNs is NP-hard. Selective SPNs are a subclass of SPNs that allow for efficient Maximum-A-Posteriori inference and closed-form parameter learning. Due to the high number of parameters, SPNs learned from data can produce unreliable and overconfident inferences, especially for instances with low statistical support. This issue can be partially mitigated by performing a robustness analysis of inferences with respect to small changes in the parameters. In this work, we address the problem of assessing the robustness of Maximum-A-Posteriori inferences produced with Selective SPNs to global perturbations of the parameters. We consider such an inference robust if it remains the single maximizer under small perturbations of the model parameters. We present efficient algorithms and an empirical analysis with realistic problems involving missing data completion and multilabel classification. The experiments show that our criteria are informative with respect to the inference accuracy, suggesting that it indeed discriminate robust and non-robust instances.

1. Introduction

Sum-Product Networks (SPNs) are computational graphs with a principled probabilistic semantics [1] (a.k.a. probabilistic circuits), that emerged from the desire to better understand and control the complexity of inference in probabilistic graphical models [2–5], at the expense of loss of interpretability. They have been shown to match and often surpass the performance of probabilistic graphical models (e.g. Bayesian networks) in many machine learning tasks [1,6–10], and to reach deep neural network performance in tasks where the latter excel [11].

SPNs are often referred to as *tractable probabilistic models* for their ability to produce linear-time predictive inference, that is, computing the conditional marginal probability of a joint configuration of some variables given some evidence. However, many tasks are better solved by performing *structured predictive inference*, that is, by finding a maximal probability joint configuration of the variables that is consistent with a given evidence [1,12]. This task is better known in the probabilistic graphical model literature as *Maximum-A-Posteriori* (MAP) inference [13].

MAP inference in SPNs is NP-hard [14], even to approximate, and even for shallow structures [15]. A notable exception is when SPNs are *selective*, that is, when the sub-networks of each sum node define distributions with disjoint supports. Selective SPNs admit MAP inference in linear time by a simple algorithm, as well as closed-form parameter learning [16,14].

* Corresponding author.
E-mail address: jgville@ime.usp.br (J. Villanueva Llerena).

https://doi.org/10.1016/j.ijar.2020.07.008
0888-613X/

While enforcing selectivity reduces the representation power of SPNs [17], empirical results have suggested that learned selective SPNs often obtain performance comparable to learned non-selective SPNs [16,18].

In spite of its relative success, and on par with other probabilistic models, SPNs learned from data generalize poorly on regions with insufficient statistical support, leading to unreliable, overconfident and prior-dependent conclusions. This issue can be mitigated by performing a *sensitivity analysis* of the model predictions to changes in the parameters [19]. Let us clarify the way we intend sensitivity analysis here, and in doing so, let us review related work.

Sensitivity analysis can be split into *quantitative* approaches and *qualitative* approaches. Quantitative approaches quantify the effect of perturbing the model parameters on the value of a particular inference. Qualitative approaches classify inferences into robust or non-robust, depending on how an inference changes according to perturbations in the model parameters. Most of the previous work in sensitivity analysis in probabilistic graphical models is quantitative, with particular focus on the computation of marginal posterior probabilities over a single variable [20–27]. Qualitative sensitivity analysis received much less attention, with some notable examples [28–30].

A second distinction can be made between *local* and *global* analyses. In the former one considers the effect of the perturbation of a single parameter, or a small group of related parameters. The latter type aims at more general perturbations possibly affecting all the parameters of the model simultaneously. Early work on algorithms for sensitivity analysis in probabilistic graphical models focused on local perturbations [27,25]; more recently, algorithms for global analysis have also been proposed [23,24,26].

Mauá et al. [28] proposed an approach to global sensitivity analysis in SPNs centered around *Credal SPNs*, which are sets of SPNs obtained by a simultaneous perturbation of all model parameters (with the network structure fixed). They developed efficient algorithms for producing upper and lower bounds on the probability of evidence, and for performing credal classification for small number of classes. Importantly, they showed that performing global qualitative sensitivity analysis of credal classification in these models is NP-hard, suggesting that a similar efficient procedure for MAP inference is likely out of reach. Antonucci et al. recently extended some of these algorithms to Credal Sentential Decision Diagrams, which are special types of selective Credal SPNs that allow for efficient handling of logical constraints [31,32].

In this work, we develop efficient algorithms for global qualitative analysis of MAP inference in selective SPNs. In particular, we devise a polynomial-time procedure to decide whether a given MAP configuration is robust with respect to a selective Credal SPN, that is, whether the configuration is the single maximizer of all SPNs in the set. We show that performing such a procedure takes coNP-complete effort in non-selective SPNs, further justifying the choice of these models. Experiments with real-world datasets show that this approach can discriminate easy- and hard-to-classify instances, often more accurately than criteria based on (precise) probabilities induced by the model.

This paper is organized as follows. We fix the notation and terminology regarding random variables in Section 2. Then, in Section 3, we introduce some basic knowledge about selective SPNs and MAP inference. In Section 4, we review Credal Sum-Product Networks. In Section 5, we present our approach to evaluate MAP inference robustness in selective SPNs. In Section 6, we describe our experiments concerning robustness analysis in missing data completion and multilabel classification. Finally, we conclude the paper and discuss possible improvements in Section 7.

2. Notation

We start with some notation and terminology. We write integers in lower case (e.g., i, j), sets of integers using capital calligraphic letters (e.g., $\mathcal{V}$) and random variables in capital letters (e.g., X_i). A collection of random variables $\{X_i, i \in \mathcal{V}\}$ is written as $\mathbf{X}_\mathcal{V}$, or simply as $\mathbf{X}$ when the index set is not important. The set of values that a random variable X_i assumes is denoted as $val(X_i)$, and the set of all instantiations of a collection of random variables $\mathbf{X}$ is denoted as $val(\mathbf{X}) = \times_{X\in\mathbf{X}} val(X)$, where $\times$ denotes the Cartesian product applied in some pre-determined ordering (that is not relevant for our purposes). We write $\mathbf{x}$ to denote an element of $val(\mathbf{X})$. In this work we assume that random variables take on a finite number of values, and we associate to every random variable X_i a set of *indicator variables* $\{\lambda_{i,k} : k = 0, \ldots, |val(X_i)| - 1\}$, where $\lambda_{i,k}$ denotes that X_i takes on its k-th value (for some arbitrary ordering of $val(X_i)$). We say that an indicator variable $\lambda_{i,k}$ is consistent with a configuration $\mathbf{x}$ if $x_i = k$, where $x_i \in val(X_i)$ is the value in $\mathbf{x}$ that corresponds to variable X_i. For example, let $\mathbf{X}_\mathcal{V}$ be a set of two binary variables $\{X_0, X_1\}$. The indicator variables associated are $\{\lambda_{0,0}, \lambda_{0,1}, \lambda_{1,0}, \lambda_{1,1}\}$. The configuration $(x_0, x_1) = (1, 0) \in val(\mathbf{X}_\mathcal{V})$ is consistent only with the indicator variables $\lambda_{0,1}$ and $\lambda_{1,0}$.

3. Selective sum-product networks

A (discrete) SPN $\mathbb{S}$ over random variables $\mathbf{X}$ is a rooted weighted acyclic directed graph whose leaves are in one-to-one correspondence with the indicator variables of $\mathbf{X}$, and internal nodes are associated to either sum or product operations (hence the name). The arcs $i \to j$ of the network are associated with non-negative weights w_{ij}, such that arcs leaving product nodes are assigned weight one (and usually omitted). Fig. 1 depicts a simple SPN over binary variables $\mathbf{X} = \{X_0, X_1, X_2\}$. The leftmost leaf node is associated to the indicator variable $\lambda_{0,1}$, which indicates whether X_0 takes on value 1. For the sake of clarity we duplicate some of the leaves in the figure (so the actual graph is not a tree), as for instance the node associated with $\lambda_{1,1}$ has two parents.

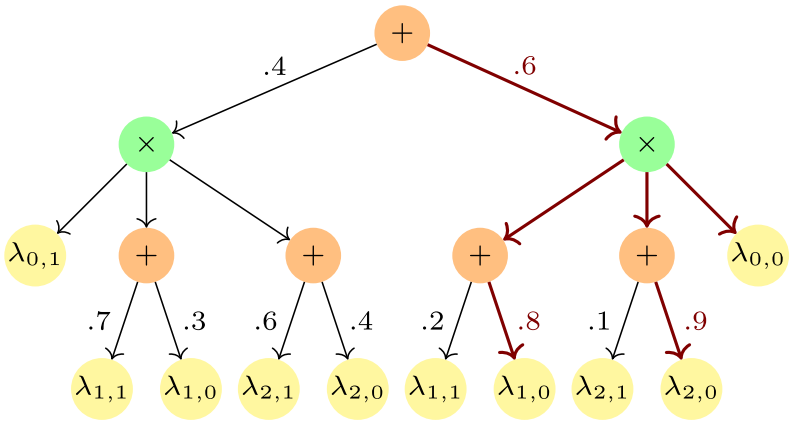


Fig. 1. Selective Sum-Product Network over three binary variables.

We denote the set of all the weights of a network as $\mathbf{w}$, and we often write $\mathtt{S}_{\mathbf{w}}$ to make explicit the dependence of network $\mathtt{S}$ on the weights $\mathbf{w}$. If i is a node in $\mathtt{S}$, we write $\mathtt{S}^i$ to denote the sub-SPN rooted at i. The children of a node i (the nodes to which there is an arc from i) are denoted by $\mathrm{ch}(i)$.

The *scope* of an SPN is the set of random variables associated with the indicator variables in the network. For example, the SPN in Fig. 1 has scope $\{X_0, X_1, X_2\}$, which is the same scope of the sub-SPNs rooted at the product nodes. The sub-SPN rooted at the leftmost leaf node has scope $\{X_0\}$, and the sub-SPN rooted at the leftmost sum node has scope $\{X_1\}$.

An SPN is a *Sum-Product Tree* if all internal nodes have at most one parent, except for sum nodes whose children are all leaves. Sum-Product Trees will be important for the algorithms we devise later on. Note that a tree-shaped SPN (i.e., an SPN where *any* node has at most one parent) is a Sum-Product Tree but the converse is not in general true. The SPN in Fig. 1 is a Sum-Product Tree which is not tree-shaped.

We say that an SPN $\mathtt{S}$ is *consistent* with an instantiation $\mathbf{x}$ if all of its leaves are associated with indicator variables that are consistent with $\mathbf{x}$. That is, if $\lambda_{i,k}$ is an indicator variable associated with a leaf then $\mathtt{S}$ is consistent with $\mathbf{x}$ only if $x_i = k$, where x_i is the value of X_i in $\mathbf{x}$.

The evaluation of an SPN $\mathtt{S}$ at an instantiation $\mathbf{x}$ produces a real-value, written $\mathtt{S}(\mathbf{x})$, defined inductively as:

- $\mathtt{S}(\mathbf{x}) = 1$ if $\mathtt{S}$ is a single-node network consistent with $\mathbf{x}$ (i.e., $\mathtt{S}$ is associated with $\lambda_{i,k}$ and $x_i = k$);
- $\mathtt{S}(\mathbf{x}) = 0$ if $\mathtt{S}$ is a single-node network *not* consistent with $\mathbf{x}$ (i.e., $\mathtt{S}$ is associated with $\lambda_{i,k}$ and $x_i \neq k$);
- $\mathtt{S}(\mathbf{x}) = \sum_{j \in \mathrm{ch}(i)} w_{ij} \mathtt{S}^j(\mathbf{x})$, if $\mathtt{S}$ is rooted at a sum node i; and
- $\mathtt{S}(\mathbf{x}) = \prod_{j \in \mathrm{ch}(i)} \mathtt{S}^j(\mathbf{x})$ if $\mathtt{S}$ is rooted at a product node i.

The value $\mathtt{S}(\mathbf{x})$ can be computed in linear time by traversing the network from the leaves towards the root, caching values if needed to avoid redundant computation. For example, the evaluation of the SPN in Fig. 1 at $X_0 = 1$, $X_1 = 1$ and $X_2 = 0$ is $0.4 \times (1 \times 0.7 \times 0.4) + 0.6 \times (0.2 \times 0.9 \times 0) = 0.112$.

Let $\mathbf{e}$ denote a partial instantiation (i.e., an assignment of values to some subset of the variables in the scope of $\mathtt{S}$), which we call, with some abuse of terminology, *evidence*. We say that a variable X is *mentioned* in $\mathbf{e}$ if $\mathbf{e}$ assigns some value for X. We can evaluate $\mathtt{S}$ at evidence $\mathbf{e}$ by extending the concept of consistency of a leaf node as follows: If X is not mentioned in $\mathbf{e}$ then any single-node network $\mathtt{S}$ with scope $\{X\}$ is consistent with $\mathbf{e}$. That means that all leaf nodes associated with indicator variables $\lambda_{i,k}$ where X_i is not mentioned in $\mathbf{e}$ will evaluate to 1. For example, the SPN in Fig. 1 evaluates to $\mathtt{S}(\mathbf{e}) = 0.6$ for $\mathbf{e}$ assigning $X_0 = 0$.

An SPN defines a probability distribution $\mathbb{P}_{\mathtt{S}}(\mathbf{x}) = \mathtt{S}(\mathbf{x})/Z$, where $Z = \sum_{\mathbf{x}} \mathtt{S}(\mathbf{x})$ is the normalization constant. We can thus compute *marginal inferences* of the form $\mathbb{P}_{\mathtt{S}}(\mathbf{e}) = \sum_{\mathbf{x} \sim \mathbf{e}} \mathtt{S}(\mathrm{x})/Z$, where the sum is over the (complete) instantiations that agree on the assignments set by the evidence. That computation is in general intractable as the sum involves an exponential number of terms. Poon and Domingos [1] showed that the following conditions ensure that marginal inference is tractable:

Completeness: The children of a sum node have identical scope.
Decomposition: The children of a product node have disjoint scopes.
Normalization: The sum of the weights of arcs leaving a sum node is one.

The SPN in Fig. 1 is complete, decomposable, and normalized. Completeness ensures that sum nodes define a mixture distribution of the (distributions defined by the) child sub-networks, which allow for efficient marginalization. Similarly, decomposition ensures that the scopes of the sub-SPNs at the children of product nodes are independent, and thus that their joint probability factorizes (which also enables efficient marginalization). Finally, normalization ensures that the nor-

malization constant $Z = 1$. This is not necessary for efficient marginal inference, as in (non-normalized) decomposable and complete SPNs we can compute Z by setting $\mathtt{S}(\mathbf{e}) = 1$ with an empty evidence $\mathbf{e}$. Normalization however allows us to ignore the normalization constant; moreover, any network can be efficiently manipulated so as to become normalized while defining the same probability distribution [33], and learned networks are often normalized. For normalized, complete and decomposable SPN $\mathtt{S}$, it follows that:

$$\mathbb{P}_{\mathtt{S}}(\mathbf{e}) = \sum_{\mathbf{x} \sim \mathbf{e}} \mathbb{P}_{\mathtt{S}}(\mathbf{x}) = \mathtt{S}(\mathbf{e})\,.$$

Hence, marginal inference can be computed in linear time in (normalized) complete and decomposable SPNs, by propagating values from the leaves towards the root (or by a recursive algorithm that memoizes computed values). Conditional marginal probabilities $\mathbb{P}_{\mathtt{S}}(\mathtt{q}|\mathtt{e})$ can also be computed efficiently by computing numerator (i.e., $\mathbb{P}_{\mathtt{S}}(\mathtt{q})$) and denominator (i.e., $\mathbb{P}_{\mathtt{S}}(\mathtt{e})$) then outputting their ratio. **For the rest of this paper, we assume that SPNs are complete, decomposable and normalized**.

The support of an SPN $\mathtt{S}$ is the set of instantiations that are evaluated to positive values: $\{\mathbf{x} \in val(\mathbf{X}) : \mathtt{S}(\mathtt{x}) > 0\}$. As we discuss later, in order to ensure that MAP inferences can be computed in linear time, we also impose the following additional property [16]:

Selectivity: For each sum node i, the support of any two sub-SPNs S^j and $S^{j'}$, with $j, j' \in \mathrm{ch}(i)$, $j \neq j'$, is disjoint.

Selectivity implies that for any instantiation $\mathbf{x}$ and sum node i that $\mathtt{S}^i(\mathbf{x}) = w_{ij}\mathtt{S}^j(\mathbf{x})$ for some $j \in \mathrm{ch}(i)$. Unlike completeness and decomposition, selectivity is a property of the evaluation of the model rather than of its graph. To our knowledge there is not efficient procedure to deciding whether a given SPN is selective. However, typical selective SPNs often satisfy certain structural properties that ensure that the network is selective irrespective of the choice of weights; such structures can be tested efficiently. For example, one way to build a selective SPN is by decomposing a variable: take an m-ary variable X_0 and m selective SPNs $\mathtt{S}^1, \ldots, \mathtt{S}^m$ whose scopes do not contain X_0, and create $\mathtt{S} = \sum_{j=1}^{m} \lambda_{0,j} \times \mathtt{S}^j$ (i.e., $\mathtt{S}$ is a sum node whose children are product nodes, each multiplying an indicator $\lambda_{0,j}$ and $\mathtt{S}^j$). The SPN in Fig. 1 is obtained in that way for variable X_0 (hence it is selective).

Given an instantiation $\mathbf{x}$ we say that a node i of an SPN $\mathtt{S}$ is *active* if $\mathtt{S}^i(\mathbf{x}) > 0$. If the network is selective then at most one child of each sum node is active for any complete instantiation. The *calculation tree* $\mathtt{T}_{\mathtt{S}}(\mathbf{x})$ of a selective SPN $\mathtt{S}$ is the SPN induced by the subgraph of *active nodes* for $\mathbf{x}$ that are connected to the root. Due to the decomposition property, the calculation tree of a selective SPN is a tree, hence justifying its name [16, Lemma 1]. In this case, we have that:

$$\mathtt{T}_{\mathtt{S}}(\mathbf{x}) = \prod_{i \to j \in \mathtt{T}_{\mathtt{S}}(\mathbf{x})} w_{ij}\,. \tag{1}$$

In Fig. 1, the sub-SPN induced by the highlighted arcs represent the calculation tree for $X_0 = X_1 = X_2 = 0$, and $\mathtt{T}(0,0,0) = 0.6 \times 0.8 \times 0.9 = 0.432$.

3.1. Learning selective sum-product networks

Given a DAG satisfying decomposability and completeness, finding a configuration for the weights that maximizes a score function (e.g., penalized loglikelihood or BIC) does not admit closed formula and one usually resorts to numerical methods (e.g. Expectation-Maximization or gradient descent) to find good approximations. This makes strategies for structural learning by search-and-score, which are popular for e.g. Bayesian networks, inefficient for SPNs. One of appealing properties of selective SPNs is that they allow for closed-form learning of the weights, thus enabling search-and-score approaches.

Given a dataset $\mathcal{D}$ of i.i.d. instantiations $\mathbf{x} \in val(\mathbf{X})$, Equation (1) allows us to use the Multinomial-Dirichlet Bayesian estimator for the weights of a fixed-structure selective SPN $\mathtt{S}$ by

$$w_{ij} = \frac{N_j + s_j}{N_i + s}\,, \tag{2}$$

where $N_k = |\mathbf{x} \in \mathcal{D} : k \in \mathtt{T}_{\mathtt{S}}(\mathbf{x})|$ is the number of calculation trees containing node k and $s = \sum_j s_j$. Note that the estimator above is equivalent to a smoothed Maximum Likelihood Estimator (MLE) of the distribution $w_i = (w_{ij})_{j \in \mathrm{ch}(i)}$.

Inspired by structure learning of standard probabilistic graphical models [13] and Equation (2), Peharz et al. [16] proposed learning selective SPNs by maximizing a penalized log-likelihood function that discounts the size of the graph from the data log-likelihood. They developed an algorithm that learns selective SPNs by performing a greedy hill-climbing search in the space of structures that at each step apply operations of splitting or merging nodes that respect selectivity. The split operation, for example, grows a network by decomposition a sum node at some variable, as we discussed earlier. Importantly, the selective SPNs learned with their algorithm are Sum-Product Trees.

More recently, Liang et al. [34] developed an algorithm for structure learning of Probabilistic Sentential Decision Diagrams (PSDDs). Their algorithm also performs local modifications to the network that improve the test-set penalized log-likelihood. Unlike the algorithm of Peharz et al. their algorithm for learning PSDDs learns structures where internal nodes may have

multiple parents. Their approach also builds mixtures of PSDDs (which are no longer selective), optimized via Expectation-Maximization. They showed state-of-the-art performance according to test-set likelihood in benchmarks when learning ensembles of PSDDs.

3.2. Maximum-a-posteriori inference

Recall that a normalized SPN $\mathcal{S}$ induces a probability measure $\mathbb{P}_{\mathcal{S}}$ over the domain of its scope. Thus, given an SPN $\mathcal{S}$ with scope $\mathbf{X} \cup \mathbf{E}$, $\mathbf{X} \cap \mathbf{E} = \emptyset$, and evidence $\mathbf{e} \in val(\mathbf{E})$, we define the set of MAP instantiations as:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x} \in val(\mathbf{X})} \mathbb{P}_{\mathcal{S}}(\mathbf{x}|\mathbf{e}) = \arg\max_{\mathbf{x} \in val(\mathbf{X})} \mathcal{S}(\mathbf{x}, \mathbf{e}) . \tag{3}$$

Although MAP inference is NP-hard in SPNs even when restricted to binary variables and height-two networks [15,35–37], the problem is solvable in linear time in the size of the network for Selective SPNs by the simple Max-Product algorithm [16,37], which we now review.

Fix a set $\mathbf{X}$ whose variables we want to maximize and some evidence $\mathbf{e}$. Recall that a leaf node is consistent with evidence $\mathbf{e}$ is either its scope is not mentioned in the evidence (which in this case implies that it is maximizing variable) or if its associated indicator variable is consistent with the evidence. The Max-Product algorithm consists in visiting the nodes in reverse topological order (i.e., from the leaves towards the root), and for each node i computing the value:

$$\text{MAP}^i = \begin{cases} 1 & \text{if } i \text{ is a consistent leaf node,} \\ 0 & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_{j \in \text{ch}(i)} \text{MAP}^j & \text{if } i \text{ is a product node,} \\ \max_{j \in \text{ch}(i)} w_{ij} \text{MAP}^j & \text{if } i \text{ is a sum node.} \end{cases} \tag{4}$$

A corresponding MAP instantiation is obtained by backtracking the solutions of the maximizations from the root toward the leaves, breaking ties arbitrarily. The correctness of the algorithms follows from Equation (1), as a MAP instantiation is obtained by maximizing over calculation trees. The calculation tree in Fig. 1 contains the arcs selected by the Max-Product algorithm when backtracking from the root (i.e., it contains the arguments of the maximizations at sum nodes) with $\mathbf{X} = \{X_0, X_1, X_2\}$. One can verify that the value of $\text{MAP}^r = 0.432$, where r is the root node of the SPN.

4. Credal sum-product networks

A *Credal Sum-Product Network* (CSPN) is a set of complete, decomposable and normalized SPNs which all share the same network structure [28]. A CSPN induces a *credal set* [38] over the scope of its networks, that is, a set of probability distributions. **In this work we consider only CSPNs with interval-valued weights**, that is, CSPNs of the form:

$$\left\{ \mathcal{S}_{\mathbf{w}} : \underline{w}_{ij} \leq w_{ij} \leq \overline{w}_{ij}, \sum_{j \in \text{ch}(i)} w_{ij} = 1, \text{ sum node } i \right\}, \tag{5}$$

where $0 \leq \underline{w}_{ij} \leq \overline{w}_{ij} \leq 1$. For convenience, we assume that the intervals are *reachable*, that is, that for each sum node i and child j, we have $\overline{w}_{ij} \leq 1 - \sum_{j' \neq j} \underline{w}_{ij}$ and $\overline{w}_{ij} \geq 1 - \sum_{j' \neq j} \underline{w}_{ij}$. This simply enforces that these intervals are tight, in the sense that there exist mass functions w_{ij} that are attained at the bounds [39]. While the algorithm we devise could be extended to cope with CSPNs where weights vary inside a local polytope, as we will soon discuss, interval-valued weight CSPNs are adequate for robustness analysis of SPNs and facilitate the presentation of the algorithms. We say that a CSPN is selective if all of its SPNs $\mathcal{S}_{\mathbf{w}}$ are selective. An example of a selective credal Sum-Product Tree is shown in Fig. 2.

Mauá et al. [28] proposed CSPNs to formalize the sensitivity analysis of SPNs, as well as offer greater flexibility and robustness for decision-making with such models. They devised efficient algorithms for basic inferences with such models, with a focus on single-variable inferences (e.g. conditional upper and lower expectations of a single variable and credal classification of a single variable). Notably, they developed a polynomial-time algorithm to compute the lower probability of a given evidence $\mathbf{e}$ induced by a CSPN $\{\mathcal{S}_{\mathbf{w}}, \mathbf{w} \in \mathcal{C}\}$:

$$\min_{\mathcal{S}} \mathbb{P}_{\mathcal{S}}(\mathbf{e}) = \min_{\mathbf{w}} \mathcal{S}_{\mathbf{w}}(\mathbf{e}) .$$

As the algorithm we devise later on makes use of such computations, we now revise their algorithm.

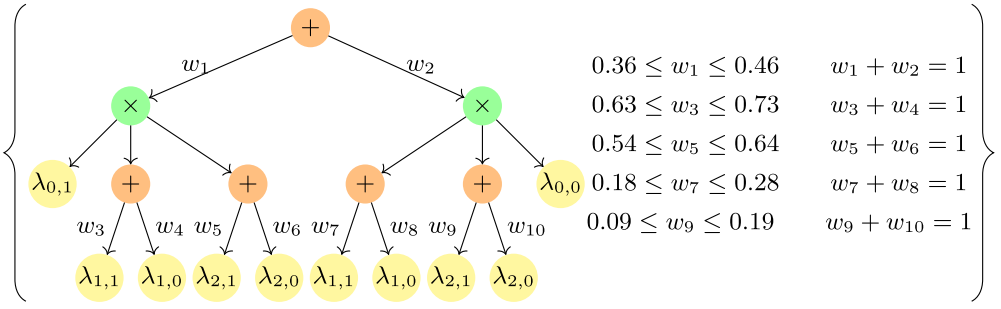


Fig. 2. Selective Credal Sum-Product Tree obtained by 0.1-contamination of the SPN in Fig. 1.

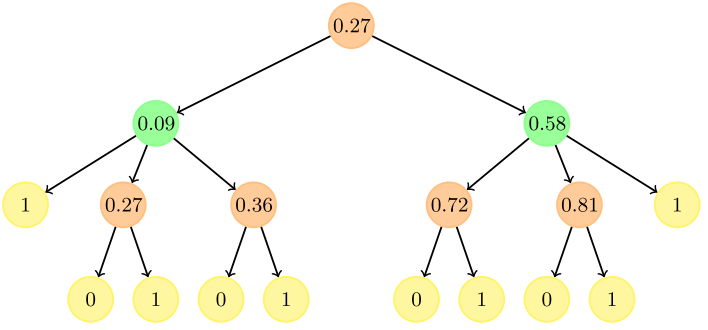


Fig. 3. Computation of the lower probability of $\mathbf{e} = \{X_1 = 0, X_2 = 0\}$ for the CSPN in Fig. 2.

The algorithm visits nodes in topological reverse order and evaluates the following expression at each node i:

$$L^i(\mathbf{e}) = \begin{cases} 1 & \text{if } i \text{ is a consistent leaf node,} \\ 0 & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_j L^j(\mathbf{e}) & \text{if } i \text{ is a product node,} \\ \min_{\mathbf{w}_i} \sum_{j \in \text{ch}(i)} w_{ij} L^j(\mathbf{e}) & \text{if } i \text{ is a sum node.} \end{cases} \tag{6}$$

The desired value is obtained at the root r of the network, that is, $L^r(\mathbf{e}) = \min_{\mathbf{w}} \mathrm{S}_{\mathbf{w}}(\mathbf{e})$. Since we assumed that CSPNs are defined by interval-valued weights $[\underline{w}_{ij}, \overline{w}_{ij}]$, the optimization in the computation of L^i for a sum node i (i.e., the last expression above) can be performed in $O(m \log m)$ time, where $m = |\text{ch}(i)|$, as it can be reduced to a continuous knapsack problem.[1] Moreover, for the special case of selective CSPNs with a complete instantiation $\mathbf{e}$, the sum reduces to a single term (as the other terms evaluate to zero), so we obtain $L^i(\mathbf{e}) = \max_{j \in \text{ch}(i)} \underline{w}_{ij} L^j(\mathbf{e})$, where the maximizer is the (single) active child $j \in \text{ch}(i)$. Hence, for selective CSPNs and complete evidence $\mathbf{e}$ the algorithm operates in time linear in the size of the network (which includes the bounds for each weight w_{ij}).

Fig. 3 shows the values of lower probability computed by the algorithm for the CSPN in Fig. 2 and evidence $\mathbf{e} = \{X_1 = 0, X_2 = 0\}$; each node i is assigned the value $L^i(\mathbf{e})$. As before, some leaves appear duplicated to avoid clutter. The algorithm outputs $L^r(\mathbf{e}) = 0.27216 \approx 0.27$.

4.1. Credal sum-product networks for robustness analysis

A common approach to analyze the robustness of an inference performed with a (precise) model is by comparing it against similar inferences produced by the models built "around" the analyzed model [40]. We can obtain a CSPN "around" a reference SPN $\mathrm{S}_{\tilde{\mathbf{w}}}$ by independently perturbing the weight mass functions $\tilde{\mathbf{w}}_i$ of each sum node i by a value $\epsilon_i \in (0, 1)$:

$$\left\{ \mathrm{S}_{\mathbf{w}} : (1 - \epsilon_i)\tilde{w}_{ij} \leq w_{ij} \leq (1 - \epsilon_i)\tilde{w}_{ij} + \epsilon_i, \sum_{j \in \text{ch}(i)} w_{ij} = 1, \text{ sum node } i \right\}. \tag{7}$$

[1] Alternatively, the result follows from the optimization being the Choquet integral of the 2-monotone capacity defined by the lower envelope of the credal set of $\mathbf{w}_i$.

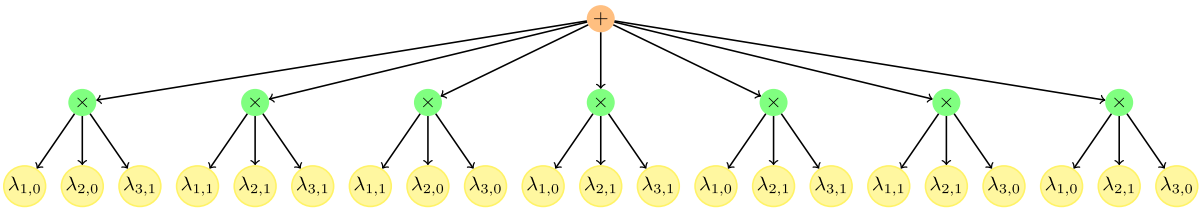


Fig. 4. An SPN encoding the satisfying assignments of the Boolean formula $\neg X_1 \vee X_2 \vee \neg X_3$.

If all $\epsilon_i = \epsilon$ in the equation above, we say that the CSPN was obtained by ϵ-contamination of the SPN $\mathrm{S}_{\tilde{\mathbf{w}}}$. According to Equation (2), the amount of data used to estimate a parameter w_{ij} in a selective SPN decreases with the depth of the node i. Yet CSPNs obtained by ϵ-contamination assign the same amount of imprecision ϵ to each mass function $\mathbf{w}_i$ associated to a sum node i irrespective of its depth. A principled way of allowing the imprecision to account for the amount of data used to learn a parameter w_{ij} is by using the Imprecise Dirichlet Model (IDM) [41], which amounts to assuming a Multinomial-Dirichlet estimate of weights with the concentration parameters $s_j \geq 0, s = \sum_j s_j$ of the Dirichlet prior varying inside a probability simplex:

$$w_{ij} \in \left[\frac{N_j}{N_i + s}, \frac{N_j + s}{N_i + s} \right], \tag{8}$$

where N_k denotes the number of computation trees that contain node k when evaluated at the (complete) instances of the dataset, and $s > 0$ determines the amount of imprecision. Note that the (smoothed) MLE (Eq. (2)) lies inside the above interval.

5. Robustness of maximum-a-posteriori inferences in sum-product networks

In this work, we consider a MAP configuration $\mathbf{x}^*$ *robust* with respect to a CSPN $\{\mathrm{S}_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ and evidence $\mathbf{e}$ if $\mathbf{x}^*$ is the single maximizer of each SPN $\mathrm{S}_{\mathbf{w}}$ in the credal set, that is, if:

$$\max_{\mathbf{x} \neq \mathbf{x}^*} \max_{\mathbf{w} \in \mathcal{C}} \left(\frac{\mathrm{S}_{\mathbf{w}}(\mathbf{x}, \mathbf{e})}{\mathrm{S}_{\mathbf{w}}(\mathbf{x}^*, \mathbf{e})} \right) < 1 . \tag{9}$$

Note that for SPNs (i.e., when $\mathcal{C}$ is a singleton) the problem reduces to deciding if there *not* exists an instantiation $\mathbf{x} \neq \mathbf{x}^*$ such that $\mathrm{S}(\mathbf{x}, \mathbf{e}) \geq \mathrm{S}(\mathbf{x}^*, \mathbf{e})$. Bodlaender et al. [42] showed that the complementary problem (i.e., deciding if there *is* such an instantiation) is NP-hard for Bayesian networks; one can adapt their proof using ideas from [15] to show that deciding if an arbitrary instance $\mathbf{x}^*$ is robust is NP-hard. The following theorem strengths that result for the case when $\mathbf{x}^*$ is a MAP configuration (and also provides a direct proof):

Theorem 1. *Deciding if a MAP configuration $\mathbf{x}^*$ is robust w.r.t. a CSPN $\{\mathrm{S}_{\mathbf{w}} : \mathbf{w} \in \mathcal{C}\}$ is coNP-complete, even if $\mathcal{C}$ is a singleton.*

Proof. Membership is trivial: an instance $\mathbf{x}$ and a configuration $\mathbf{w}$ for which $\mathrm{S}_{\mathbf{w}}(\mathbf{x}, \mathbf{e}) \geq \mathrm{S}_{\mathbf{w}}(\mathbf{x}^*, \mathbf{e})$ is a certificate that the problem instance is not in the language.

We show hardness by reducing a NP-hard problem to the complementary problem: deciding if an instantiation is *not* robust. Thus consider the NP-complete problem of deciding whether a 3-CNF Boolean formula ϕ over variables $X_1, \ldots, X_n$ is satisfiable. Without loss of generality, assume that ϕ has at least one positive literal and one negative literal (otherwise the formula is trivially satisfiable). Construct an assignment $x_1^*, \ldots, x_n^*$ that does *not* satisfy ϕ; such an assignment can be obtained by taking any clause, negating its literals (so as to not satisfy it) and then extending it with any assignment to the remaining variables. Assemble an SPN as follows. For each clause ψ in ϕ, construct a subnetwork S_ψ that encodes each of the 7 satisfying assignments of that clause as a product of indicator variables, and such that S_ψ is selective and evaluates to $1/7$ iff the corresponding assignment satisfies the clause. Fig. 4 shows an example of such a network. To extend the scope of the subnetwork to all the variables, create a product node with S_ψ as one child and indicator variables of the remaining variables at the configuration different than $\mathbf{x}^*$ (e.g., if X_i does not appear in ψ and $x_i^* = 1$ then connect the product node to $\lambda_{i,0}$). Call this network S'_ψ; we have that $\mathrm{S}'_\psi(\mathbf{x}) = 1/7$ if $\mathbf{x}$ satisfies ψ and $\mathbf{x} \neq \mathbf{x}^*$, otherwise $\mathrm{S}'_\psi(\mathbf{x}) = 0$. Now connect all subnetworks S'_ψ via a sum node with edge weights $1/m$ where m is the number of clauses. Call this network S_ϕ. It follows that $\mathrm{S}_\phi(\mathbf{x}) = 1/7$ iff ϕ is satisfied by the corresponding assignment $\mathbf{x}$. Now build a network S with a sum node that has S_ϕ as one child, with an edge weight $7/8$, and subnetwork S_* that encodes the assignment $x_1^*, \ldots, x_n^*$ as the other child (with edge weight $1/8$); the latter subnetwork is simply a product node connected to indicator variables λ_{i,x_i^*}. First note that $\mathrm{S}(\mathbf{x}^*) = 1/8$: as $\mathrm{S}'_\psi(\mathbf{x}^*) = 0$. Also, for any $\mathbf{x}$ whose corresponding assignment satisfies ϕ, we have that $\mathrm{S}(\mathbf{x} = 1/8)$. Thus, the instantiation $\mathbf{x}^*$ is *not* robust (i.e., it is not the unique maximizer of S) iff ϕ is satisfiable. This proves that deciding if a (MAP) instantiation is robust is coNP-hard. □

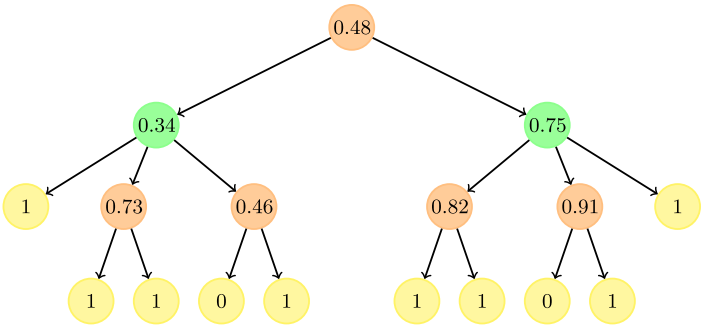


Fig. 5. Computation of the maximum upper probability of a configuration of $\mathbf{X} = \{X_0, X_1\}$ for evidence $\mathbf{e} = \{X_2 = 0\}$ and the CSPN in Fig. 2.

The previous result justifies our focus on selective CSPNs.

Before formulating an algorithm for deciding robustness of MAP inference in selective SPNs, let us consider the simpler problem of computing the MAP instantiation with the maximum upper joint probability over the set of SPNs in a CSPN. This will be used as a subroutine to verify robustness later. Thus consider a selective CSPN $\{\mathbb{S}_{\mathbf{w}} : \underline{\mathbf{w}}_i \leq \mathbf{w}_i \leq \overline{\mathbf{w}}_i\}$ and evidence $\mathbf{e}$ for all variables outside $\mathbf{X}$. We are interested in computing:

$$\max_{\mathbf{x}} \max_{\mathbf{w}} \mathbb{S}_{\mathbf{w}}(\mathbf{x}, \mathbf{e}) .$$

Call a leaf node of the network a *MAP leaf* if its scope is in $\mathbf{X}$, otherwise call it an *evidence leaf*. Recall that we call a leaf node consistent if it is a MAP leaf or if its an evidence leaf whose indicator variable is consistent with the evidence. We can compute the above maximum upper probability of a configuration by traversing the network in reverse topological order, calculating at each node i:

$$M^i = \begin{cases} 1 & \text{if } i \text{ is a consistent leaf node,} \\ 0 & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_{j \in \mathrm{ch}(i)} M^j & \text{if } i \text{ is a product node,} \\ \max_{j \in \mathrm{ch}(i)} \overline{w}_{ij} M^j & \text{if } i \text{ is a sum node.} \end{cases}$$

Note the algorithm is equivalent to performing Max-Product in the (non-normalized) SPN where weights are fixed at their upper bounds. Fig. 5 shows a computation of the values M^i for the CSPN in Fig. 2 and evidence $\mathbf{e} = \{X_2 = 0\}$.

The soundness and complexity of the algorithm are given by the following result.

Theorem 2. *Consider a selective CSPN $\{\mathbb{S}_{\boldsymbol{w}} : \boldsymbol{w} \in \mathcal{C}\}$ with root r. Then M^r computes $\max_{\boldsymbol{x}} \max_{\boldsymbol{w}} \mathbb{S}_{\boldsymbol{w}}(\boldsymbol{x})$ in time linear in the input size.*

Proof. We prove correctness by induction in the height h of $\mathbb{S}$. The base case for h=0 is immediate, as it consists of a leaf node which is maximized by setting the associated indicator to one, for the case of MAP leafs, or simply evaluated at the evidence. Assume that the inductive hypothesis is valid for networks of height $h \geq 0$ or smaller, and consider a network of height $h+1$ whose root is i. Recall that in a selective CSPN, at most one child j of a sum node satisfies $\mathbb{S}^j_{\mathbf{w}}(\mathbf{x}, \mathbf{e}) > 0$ for each instantiation $\mathbf{x}$ for any choice of $\mathbf{w}$. Hence, we can partition the set of instantiations $\mathbf{x} \in val(\mathbf{X})$ into the disjoint sets $\mathcal{X}_j$ for which $\max_{\mathbf{w}} \mathbb{S}^j_{\mathbf{w}}(\mathbf{x}, \mathbf{e}) > 0$. Thus if i is a sum node then

$$\begin{aligned} \max_{\mathbf{x} \in val(\mathbf{X})} \max_{\mathbf{w}} \mathbb{S}^i_{\mathbf{w}}(\mathbf{x}, \mathbf{e}) &= \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{x} \in \mathcal{X}_j} \max_{\mathbf{w}} w_{ij} \mathbb{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e}) \\ &= \max_{j \in \mathrm{ch}(i)} \max_{\underline{w}_{ij} \leq w_{ij} \leq \overline{w}_{ij}} w_{ij} \max_{\mathbf{x}} \max_{\mathbf{w}_j} \mathbb{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e}) \\ &= \max_{j \in \mathrm{ch}(i)} \overline{w}_{ij} M^j = M^i . \end{aligned}$$

Note that the weights $\mathbf{w}_j$ can be optimized independently because the maximization over $\mathbf{X}$ selects a circuit tree, where each edge/weight of the network appears at most once.

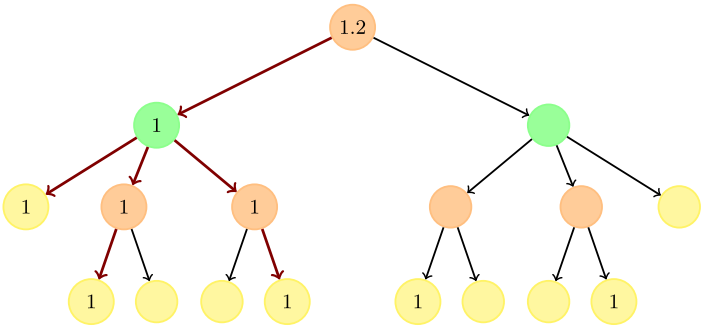


Fig. 6. Simulation of our algorithm for robustness analysis of the MAP configuration $\mathbf{x}^* = \{X_0 = 1\}$ for evidence $\mathbf{e} = \{X_1 = 1, X_2 = 0\}$ with respect to the CSPN in Fig. 2. The nodes are assigned the values V^i when they are computed. The highlighted edges represent the active nodes (i.e., the ones in $\mathtt{T}_{\mathtt{S}}(\mathbf{x}^*, \mathbf{e})$).

If i is a product node, we have that

$$\max_{\mathbf{x}} \max_{\mathbf{w}} \prod_{j \in \mathrm{ch}(i)} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}) = \prod_{j \in \mathrm{ch}(i)} \max_{\mathbf{x}} \max_{\mathbf{w}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e}) = \prod_{j \in \mathrm{ch}(i)} M^j = M^i .$$

The weights $\mathbf{w}_j$ in the first equation can be optimized independently for each $j \in \mathrm{ch}(i)$, as the disjointedness of scopes for product nodes ensures that no weight is shared among the child sub-networks.

If we process nodes in reverse topological order, then each value M^i is computed once in time linear in the number of children of i and in the size of the specification of the weight intervals. Therefore the total cost of this computation is linear in the input size. □

We are now ready to present our algorithm for deciding whether a MAP instantiation $\mathbf{x}^*$ is robust for evidence $\mathbf{e}$. We assume that $\max_{\mathbf{w}} \mathtt{S}_{\mathbf{w}}(\mathbf{x}^*, \mathbf{e}) > 0$, as otherwise the problem is trivially solved. As usual, the algorithm operates by traversing the network structure from the leaves towards the root. We assume that the values M^i and $L^i(\mathbf{x}^*, \mathbf{e})$ have been already computed for each node i (in practice we can perform these computations in parallel). First, compute for each sum node i in $\mathtt{T}_{\mathtt{S}}(\mathbf{x}^*, \mathbf{e})$ the value:

$$U^i = \max_{j \in \mathrm{ch}(i), j \neq k} \frac{\overline{w}_{ij} M^j}{\underline{w}_{ik} L^k(\mathbf{x}^*, \mathbf{e})} . \tag{10}$$

where k is the active child of i in $\mathtt{T}_{\mathtt{S}}(\mathbf{x}^*, \mathbf{e})$ (if $\underline{w}_{ik} L^k(\mathbf{x}^*, \mathbf{e}) = 0$ then set the value of $U^i = \infty$). Now compute for each node i in $\mathtt{T}_{\mathtt{S}}(\mathbf{x}^*, \mathbf{e})$:

$$V^i = \begin{cases} 1 & \text{if } i \text{ is a consistent leaf node,} \\ 0 & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_j V^j & \text{if } i \text{ is a product node,} \\ \max\{U^i, V^k\} & \text{if } i \text{ is a sum node with active child } k. \end{cases} \tag{11}$$

The diagram in Fig. 6 represents the simulation of the algorithm for $\mathbf{x}^* = \{X_0 = 1\}$ and evidence $\mathbf{e} = \{X_1 = 1, X_2 = 0\}$ with respect to the CSPN in Fig. 2. The figure displays the values of V^i for the corresponding nodes computed using Equation (11) (note that V^i is computed only for nodes in the circuit tree of $\mathbf{x}^*, \mathbf{e}$). The MAP instance is considered not robust because $V^r = U^r = (0.62 * 0.25)/(0.38 * 0.34) \approx 1.2 > 1$.

The following result states the soundness and the complexity of the algorithm for Sum-Product Trees.

Theorem 3. *Consider a selective CSPN $\{\mathtt{S}_{\boldsymbol{w}} : \boldsymbol{w} \in \mathcal{C}\}$ containing Sum-Product Trees with root r. Then V^r computes*

$$\max_{\boldsymbol{x}} \max_{\boldsymbol{w} \in \mathcal{C}} \left(\frac{\mathtt{S}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{e})}{\mathtt{S}_{\boldsymbol{w}}(\boldsymbol{x}^*, \boldsymbol{e})} \right)$$

in time linear in the size of input.

Proof. We prove that the algorithm is correct by induction in the height h of $\mathtt{S}$. The base case for $h = 0$ is immediate. So assume that the inductive hypothesis is valid for networks of height $h \geq 0$ or smaller, and consider a network of height

$h+1$ whose root is i. Assume that i is a sum node. The tree shape of the network implies that the weights $\mathbf{w}_j$ that appear in a sub-network S^j, $j \in \mathrm{ch}(i)$, do not appear in any other sub-network $\mathrm{S}^{j'}$, $j' \in \mathrm{ch}(i)$, $j' \neq j$. Denote by $\mathcal{X}_j$ the subset of $val(\mathbf{X})$ for which $\mathrm{S}^j(\mathbf{x}, \mathbf{e}) > 0$ as in the proof of Theorem 2. We have that

$$\max_{\mathbf{x}} \max_{\mathbf{w}} \frac{\mathrm{S}_{\mathbf{w}}(\mathbf{x}, \mathbf{e})}{\mathrm{S}_{\mathbf{w}}(\mathbf{x}^*, \mathbf{e})} = \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{w}} \frac{w_{ij} \max_{\mathbf{x} \in \mathcal{X}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{w_{ik} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})},$$

where k is the child of i for which $\mathrm{S}^k(\mathbf{x}^*, \mathbf{e}) > 0$. For $j = k$ it follows that

$$\max_{\mathbf{w}} \frac{w_{ij} \max_{\mathbf{x} \in \mathcal{X}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{w_{ik} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} = \max_{\mathbf{x}} \max_{\mathbf{w}_k} \frac{\mathrm{S}^k_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{\mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} = V^k.$$

For $j \neq k$, $\mathbf{w}_j$ and $\mathbf{w}_k$ can be optimized independently, as the network is a Sum-Product Tree. Hence,

$$\begin{aligned} \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{w}} \frac{w_{ij} \max_{\mathbf{x} \in \mathcal{X}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{w_{ik} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} &= \max_{j \in \mathrm{ch}(i)} \frac{\overline{w}_{ij} \max_{\mathbf{x}} \max_{\mathbf{w}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{\underline{w}_{ik} \min_{\mathbf{w}_k} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} \\ &= \max_{j \in \mathrm{ch}(i)} \frac{\overline{w}_{ij} M^j}{\underline{w}_{ik} L^k(\mathbf{x}^*, \mathbf{e})} = U^i. \end{aligned}$$

Assume now that i is a product node; then

$$\begin{aligned} \max_{\mathbf{x}} \max_{\mathbf{w}} \left(\frac{\prod_j \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x})}{\prod_j \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^*)} \right) &= \max_{\mathbf{x}} \max_{\mathbf{w}} \prod_{j \in \mathrm{ch}(i)} \frac{\mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{\mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^*, \mathbf{e})} \\ &= \prod_j \max_{\mathbf{x}} \max_{\mathbf{w}_j} \frac{\mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{\mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^*, \mathbf{e})} \\ &= \prod_j V^j = V^i. \end{aligned}$$

This proves soundness of the algorithm. To prove its complexity, first note that computing M^i and L^i takes linear effort. Also, computing V^i for a node i with the values of the children pre-computed and stored takes at most linear time in the number of children and in the representation of the intervals. Hence, the total cost of the computation is linear in the size of input. □

The complexity of verifying robustness in selective credal Sum-Product Trees is in the same order as the complexity of computing MAP inference in selective SPNs, so that the additional robustness check can be included in any application using SPNs to produced MAP inference with little overhead. The complexity of deciding robustness for multiply-connected selective CSPNs remains an open question.

A final note: we have defined robustness in Equation (9) as a maximization over instances $\mathbf{x} \neq \mathbf{x}^*$. The algorithm defined by Equations (11) however maximizes over the entire space $val(\mathbf{X})$, which includes $\mathbf{x}^*$. That means that the algorithm will always output a value $V^r \geq 1$. For the case of $V^r > 1$, we have that the maximization is attained at $\mathbf{x} \neq \mathbf{x}^*$, so *no* robustness is verified according to Equation (9) as well. When the algorithm outputs $V^r = 1$ we can still verify robustness by the same criterion by backtracking the subtrees used in the computation of V^i and verifying if $V^i \neq U^i$ in some node. That is, if $V^i = U^i$, we backtrack the edge selected in the maximization of U^i, otherwise we backtrack the edge k. If at any point we reach an inner node where $V^i > U^i$, then the MAP instantiation is robust. The pseudo-code in Fig. 7 shows the entire procedure for verifying robustness.

To see an example of robustness being decided when $V^r = 1$, consider the computation of V^r for $\mathbf{x}^* = \{X_0 = 0, X_1 = 0\}$ and evidence $\mathbf{e} = \{X_2 = 0\}$ with respect to the CSPN in Fig. 2. The diagram in Fig. 8 shows the values of V^i for each node, and highlights the edges in the circuit tree of the MAP instantiation. The MAP instance is considered robust because $V^r = 1$ and $U^r = (0.46 * 0.34)/(0.54 * 0.58) < 1$.

6. Experiments

We evaluate the ability of our proposed method to distinguish between robust and non-robust MAP inferences in two different tasks. The first task consists in learning a probabilistic model from complete data and then using that model to complete the missing values in the test data by maximizing the joint probability value of each instance (this is known as data imputation in the literature). We let $\mathbf{X}$ be the missing part and $\mathbf{e}$ the non-missing part. The second task considers

Require: CSPN $\{S_{\mathbf{w}}\}$, evidence $\mathbf{e}$, instantiation $\mathbf{x}^* \in \text{argmax}_{\mathbf{x} \in val(\mathbf{X})} S_{\tilde{\mathbf{w}}}$
Compute M^i and $L^i(\mathbf{x}^*, e)$ for each node i
for each node i in reverse topological order **do**
  if i is a leaf node **then**
    if i is consistent **then** $V^i \leftarrow 1$ **else** $V^i \leftarrow 0$
  else if i is a product node **then**
    $V^i \leftarrow \prod_{j \in \text{ch}(i)} V^j$
  else ▷ i is a sum node
    Let k be the active child of i in $S_{\tilde{\mathbf{w}}}(\mathbf{x}^*)$
    $U^i \leftarrow \max_{j \in \text{ch}(i), j \neq k} \frac{\overline{w}_{ij} M^j}{\underline{w}_{ik} L^k(\mathbf{x}^*, \mathbf{e})}$
    $V^i \leftarrow \max\{U^i, V^k\}$
  end if
end for
if $V^r > 1$ **then return** non-robust
else ▷ Backtrack solution
  $i \leftarrow r$
  while $V^i = U^i$ **do**
    $i \leftarrow \text{argmax}_{j \in \text{ch}(i), j \neq k} \frac{\overline{w}_{ij} M^j}{\underline{w}_{ik} L^k(\mathbf{x}^*, \mathbf{e})}$
  end while
  if i is a leaf **then**
    return non-robust
  else
    return robust
  end if
end if

Fig. 7. Verifying Robustness of MAP instantiations in SPNs.

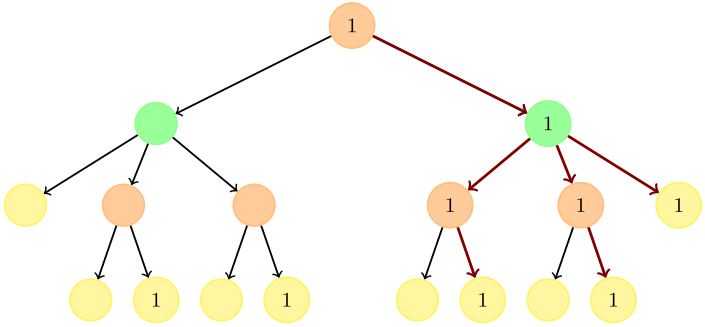


Fig. 8. Simulation of our algorithm for robustness analysis of the MAP configuration $\mathbf{x}^* = \{X_0 = 0, X_1 = 0\}$ for evidence $\mathbf{e} = \{X_2 = 0\}$ with respect to the CSPN in Fig. 2. The nodes are assigned the values V^i when they are computed.

multilabel classification, which extends standard classification by allowing each object be assigned multiple labels. This task can be solved by representing the presence/absence of labels as a binary vector $\mathbf{x}$, which can be predicted by a probabilistic model given features $\mathbf{e}$.

6.1. Experimental set-up

For either task, we learn selective Sum-Product Trees using the algorithm by Peharz et al. [16]. The weights of the networks are obtained by smoothed MLE (see Equation (2)) using a smoothing factor selected maximizing the validation set likelihood. We then obtain CSPNs either by ϵ-contamination or by the Imprecise Dirichlet Model over the smoothed counts by fixing the level of imprecision ϵ or s (more on that later). We use the CSPNs to classify each MAP inference drawn with the (precise) SPN into either robust or non-robust, and we compare the performance of robust and non-robust instances. We expect to observe the performance (e.g. classification accuracy) of deemed robust instances to be higher than that of deemed non-robust instances, indicating that our approach provides useful information about the robustness of MAP inferences. We also compare our methods against a baseline robust analysis that computes the difference between the probability of the MAP instantiation and the second-best MAP instantiation (i.e., the second most probable configuration consistent with evidence). An instance is considered robust in this scheme if the probability difference is greater than a given threshold. We obtain the second-best MAP by running Max-Product several times, each time fixing one variable at a value different than the selected value in the MAP instantiation. All three approaches rely on one parameter: the imprecision ϵ or s for the CSPN-based approaches and the difference in probability threshold for the SPN-based method that compares best and second-best MAP instances. To align the decisions of the three approaches in a meaningful and consistent way we select values for the parameter so that a fixed percentage of the test instances is classified as robust. We then compare

Table 1
Characteristics of datasets used in completion tasks: number of training instances, number of test instances, number of observed variables, number of missing values, and density of values set to one.

Dataset	Training	Test	#Evidence	#Query	Density
DNA	1600	1186	170	10	0.253
Jester	9000	4116	90	10	0.608
NLTCS	16181	3236	8	8	0.332
MSNBC	291326	58265	9	8	0.166

the approaches as we vary that percentage. Ideally, we expect an increase in performance as we decrease the percentage of robust instances (thus being more conservative); in practice this also increases the variance of the performance estimator as the sample size is reduced. Therefore, one should take caution when observing trends in the curves for different approaches, especially when test sets are small.

6.2. Completion

We start with the completion task. To this end, we learned selective SPNs from four well-known datasets for density estimation [43], available at https://github.com/arranger1044/DEBD. The selected datasets and their characteristics appear in Table 1. As one can see from the table, the collection of datasets is diverse, with dimensionality and data sizes widely differing. The datasets are complete (i.e., have no missing values), are separated into training, validation and test parts, and had the variables been binarized. Thus no pre-processing was performed. We built completion tasks using the test examples by running MAP inference in the first 10, 8 or 9 of the variables (as they appear in the file) with the remaining variables given as evidence. We use the available training/validation/test partition in the datasets to respectively learn, select parameters, and evaluate the methods. We measure the performance of the completions either by the Hamming Score (HS) or by the Exact Match (EM) metric, as defined next.

The Hamming Score measures the percentage of correct value completions:

$$\mathrm{HS} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathbb{I}(x^*_{i,j} = \hat{x}_{i,j})\,,$$

where $\mathbb{I}$ is the indicator function, $\mathbf{x}^*_i$ is the MAP instantiation of the i-th example, $\hat{\mathbf{x}}_i$ is the corresponding ground truth, N is the number of instances and L is the number of MAP variables (missing values in this case). This metric can be misleadingly high when the (marginal) distribution of values for most variables is very skewed. The density column in Table 1 shows the proportion of values ones in the dataset, computed as $\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(x_{ij} = 1)$, where N is the size of the dataset. Balanced datasets have density close to 0.5. Note how most of the datasets we use are unbalanced. A second issue is that HS is theoretically maximized by selecting for each variable, independently, the value that maximizes the conditional marginal probability given the evidence. When the MAP variables are highly correlated, a MAP instantiation can thus have relatively low marginal probability (hence low HS).

Exact match mitigates these issues by computing the percentage of perfect completions:

$$\mathrm{EM} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\mathbf{x}^*_i = \hat{\mathbf{x}}_i)\,.$$

Exact match is often too strict, as a completion that misassigns the value of a single variable and one that misassigns values for all variables are scored equally. This leads to a high variance estimator, especially for small datasets and for large number of MAP variables. In sum, the two metrics help evaluating the completions from different perspectives.

Table 2 contains the number of parameters (weights) and the performance of the learned SPNs. Comparing that table and Table 1, we note that the SPN learned from DNA dataset is relatively large compared to its training data set size (i.e., it has nearly as many parameters as training instances), while NLTCS is relatively small. From Table 2 we also see that the EM value is very low for the Jester dataset, despite the relatively high value of HS. Hamming score on the hand is relatively high for all datasets, especially NLTCS.

Recall that we align the decisions (i.e., whether each instance is deemed robust or non-robust) of the three approaches by selecting, for each approach, the corresponding parameter value that leads to a certain percentage of instances being considered robust in the test set. This way each method is evaluated (using either HS or EM) on a sample of equal size, thus facilitating the comparison. Also, this is an effective procedure for selecting the parameter values in practice. We note that while aligning the methods facilitates visualizing the results, it also introduces bias, as parameters are selected using the test-set. That means that a very different proportion of instances is deemed robust by a configuration of parameter and approach if the test-set is not representative of the population. We expect this effect to be small for the datasets in Table 1, as the test sets are relatively large. The plots of performance versus percentage for each dataset are shown in Fig. 9.

Table 2
Size and performance of completions of the (precise) SPNs learned from the full training dataset.

Dataset	#Param.	Training/#Param.	Exact Match	Hamming Score
DNA	1038	1.54	0.013	0.609
Jester	3170	2.84	0.069	0.680
NLTCS	568	28.5	0.370	0.810
MSNBC	2816	103.5	0.247	0.783

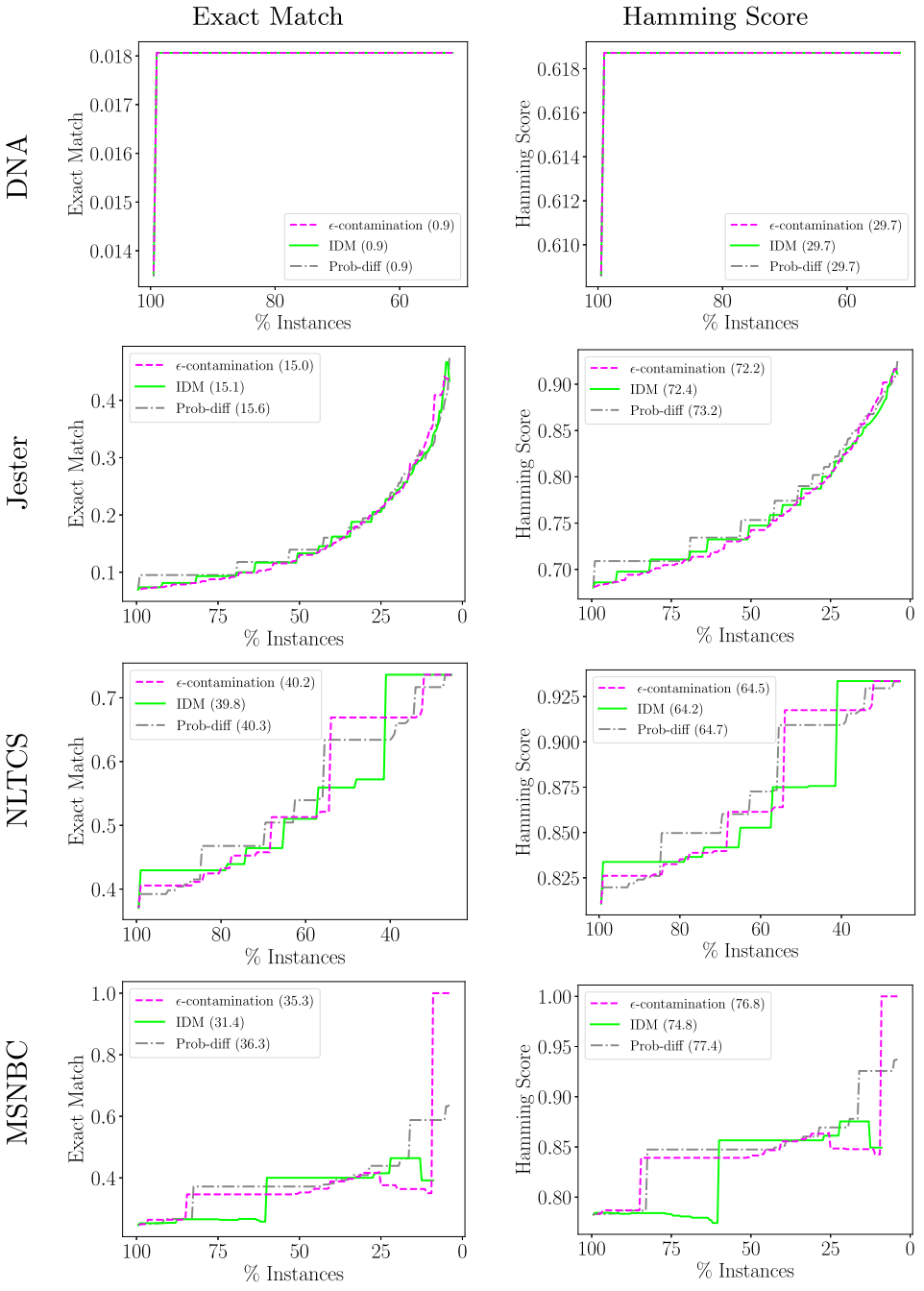


Fig. 9. Test-set performance vs. proportion of robust instances for the completion task on SPNs learned from the full training set, and the AUC in parentheses next to each method in the legend. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 3
Size and test-set performance of completions of the (precise) SPNs learned from a sample of 10% of the training instances.

Dataset	#Param.	Training/#Param.	Exact Match	Hamming Score
Jester	770	1.16	0.055	0.660
NLTCS	211.2	7.66	0.365	0.809
MSNBC	886.4	32.87	0.243	0.781

Table 4
Size and test-set performance of completions of the (precise) SPNs learned from a sample of 1% the training instances.

Dataset	#Param.	Training/#Param.	Exact Match	Hamming Score
Jester	312	0.29	0.016	0.613
NLTCS	70.8	2.27	0.319	0.791
MSNBC	106.2	27.43	0.178	0.793

The numbers next to the name of each method in the legends indicate the area under the corresponding curve (AUC), and measure the average performance of the approach.

As one can see from the plots, all three approaches are able to discriminate the more accurate instances, and the robustness parameters of each (ϵ, s or the probability difference threshold) are positively correlated with performance for any metric and dataset. Moreover, the increase in performance is monotone for all datasets but for MSNBC, where CSPN-based metrics oscillate, despite showing a positive trend. Recall that MSNBC has the highest ratio of training data and number of parameters, so one expects inferences with the corresponding SPN to be relatively more robust than with other datasets (especially, compared to DNA and Jester). No approach is strictly superior to any other: for any dataset and metric it is possible to select a percentage of robust/non-robust instance split that leads any of the approaches to outperform the others. On average, as measured by the areas under the curves, all methods perform similarly, with the baseline showing a slight advantage for all datasets, and IDM performing worse for all datasets/metric but for Jester/HM, where it ranks second. All methods perform perceptually identical on the DNA dataset; the performance of robust instances increases sharply as we decrease the percentage of robust instances (which is equivalent to increasing the values of parameters) until a plateau is reached where all methods become unable to further distinguish robust and non-robust instances. We conjecture that this is due to the structure of the corresponding learned SPN, which makes the value of the network heavily influenced by a few variables that are close to the root of the network, and very weakly affected by the other variables (so that a large number of instantiations obtain nearly identical probability values). Note that the two approaches (ours and the baseline) provide significantly different information: the baseline informs whether the MAP instantiation is considerably more probable than other instantiations for the distribution learned; our approaches inform whether that is so even when we severely perturb the model parameters (so they are more conservative). Combined, both approaches increase confidence in the model predictions more than each approach individually.

To investigate the effect of training data size in the quality of the robustness analysis, we repeated the experiments while subsampling the training sets. For each dataset, we generated 10 subsets containing a fixed percentage of the training dataset instances, learned an SPN from each training subset, and computed the median performance for all approaches (we also computed mean and standard deviation and the results were qualitatively similar). We used subsamples with 10% and 1% of the size of the original training dataset. Tables 3 and 4 show the characteristics and median performance of the learned networks for 10% and 1%, respectively, of training dataset size. The number of parameters and training data size/number of parameters ratio are averages over all the 10 SPNs learned with different subsamples of the training dataset for a fixed size. We observe that while the drop in the number of parameters follows the drop in the amount of training data, the ratio of training instances and parameters decreases by two or three times for most datasets. The sharpest decrease of that ratio is for the SPNs learned from subsamples of 1% of the Jester training datasets. This is followed by the sharpest decrease in performance for the EM metric in the completion task (a reduction of nearly a third). These results demonstrate the known ability of SPN structure learners to adapt the complexity of the model to the amount of available data, up to some minimum complexity given by the domain dimensionality. We also observe that the median performance of the learned SPNs from subsamples remains similar to the performance of the SPNs learned from the full training data when 10% of the data are used, and has a small decrease when 1% of the data are used. This suggests that these datasets are still relatively large and not very complex, and that SPNs are decent at capturing probabilistic relations even under small data scenarios.

We compare approaches as before (aligning median performances by percentage of robust instances on the test set). Figs. 10 and 11 show the results for subsamples with 10% and 1% of the training dataset, respectively. For any of the approaches, a small variation in the parameter caused all instances of the DNA test set to be deemed non-robust; we have thus omitted these results from the figures. The results with 10% of training data are qualitatively identical to the results with the full training dataset. The only notable difference is the more monotone curves for the CSPNs obtained by ϵ-contamination on the MSNBC datasets, which now follow very closely the curves of the baseline approach.

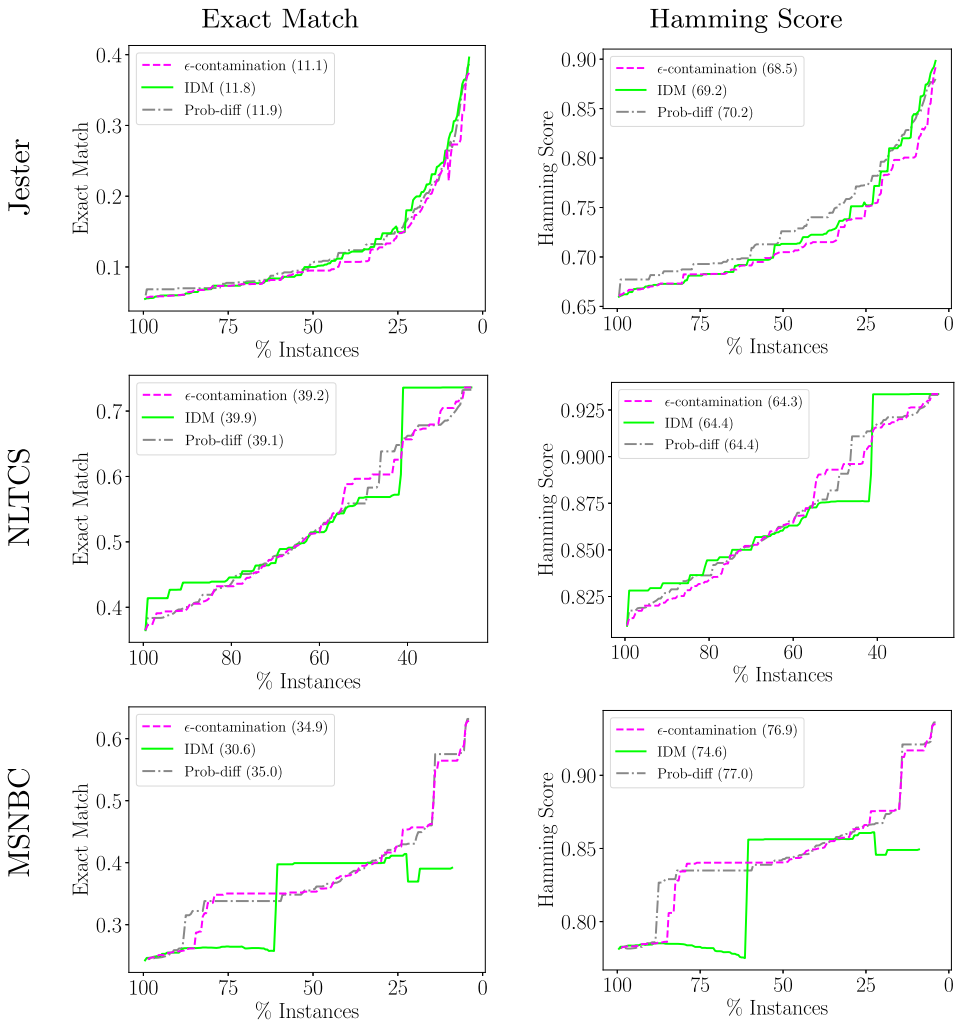


Fig. 10. Median test-set performance vs. proportion of robust instances for the completion task with SPNs learned from 10% of the training instances, and the AUC in parentheses next to each method in the legend.

The results with 1% of training data are more interesting discussing. Perturbation-based approaches are expected to be more robust to regimes of severe data scarcity compared to the baseline. This is observed for the datasets Jester and NLTCS, where the probability difference baseline performs much poorly at discriminating robust and non-robust instances. Note that we stop computing the performance for roughly less than 30% of robust instances on Jester and NLTCS, as the dataset becomes too small. The IDM-based approach outperforms the other approaches in the Jester dataset, and both CSPN-based approaches are superior to the baseline on NLTCS. In the MSNBC datasets, the ϵ-contamination approach was overall superior, especially with respect to EM. The average performance with respect to HS is comparable with the IDM-based method lagging slightly behind. In sum, these experiments suggest that CSPN-based approaches are competitive at identifying (non)robust inferences, and achieve a mild advantage in very small data scenarios.

6.3. Robustness of out-of-distribution instances

Due to various reasons, predictive models are often used in data whose distribution differs from that used in learning; one example is in detecting anomalies, concept drift or novelty. On top of that, instances might contain missing data. Imputing data by MAP inference has been shown to improve the accuracy of e.g. anomaly detectors [44]. Yet, one might expect that the accuracy of the learned model when applied to out-of-distribution instances is low; this might lead to statistically unsupported data imputations and consequently to non-robust decisions.

To evaluate the ability of CSPNs to discriminate between robust and non-robust completions under the presence of out-of-distribution instances, we learned an SPN from the Digits dataset [45], that contains pre-processed 8-by-8 binary images of handwritten digits (hence 64 binary variables). We selected that dataset as it has a small training dataset and allowed for the visual inspection and interpretation of the predictions. We use only images labeled as digit “five” (but we do not use

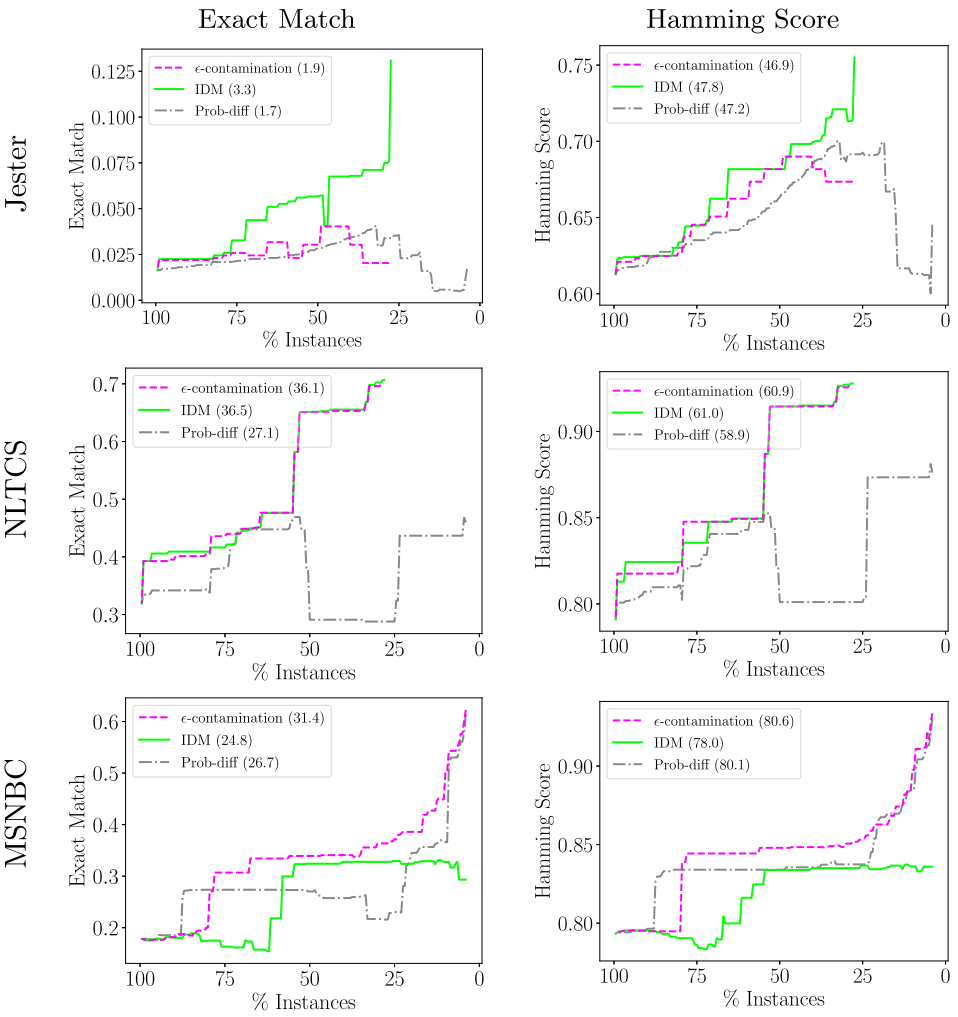


Fig. 11. Median test-set performance vs. proportion of robust instances for the completion task with SPNs learned from 1% of the training instances, and the AUC in parentheses next to each method in the legend.

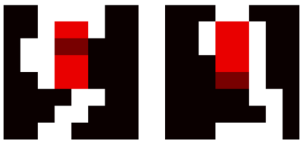


Fig. 12. Example of instances in the robustness analysis experiment with the Digits dataset; the pixels inside the red rectangles are the query variables, the remaining pixels are set as evidence.

any variable to indicate the label) as training set. To simulate the presence of out-of-distribution instances, we assembled a test set containing 30 instances of the digit "five" (not in the training dataset) and 30 instances from the other digits (in equal percentage). We did not use the class variable (as it would assume a single value in our training dataset). For each test instance, we compute the MAP prediction of the 4-by-2 pixel red region in Fig. 12 given all remaining pixels as evidence (we selected that region as it is a highly informative for discriminating digits). Table 5 shows the characteristics and performance of the SPNs learned from the datasets. The first row displays results on the test set containing only the 30 instances of digit five, and the second row displays results on the test set containing also the out-of-distribution instances. We see from these numbers that training data is scarce, and that the presence of out-of-distribution instances decreases accuracy.

As before, we compare the robustness metrics for the MAP predictions using CSPNs obtained with ϵ-contamination and the IDM against the baseline given by the difference between the most probable and the second most probable completions using the precise model.

Table 5
Size and performance of completions of the (precise) SPNs learned from the "five" Digit training dataset.

Dataset	#Param.	Training/#Param.	Exact Match	Hamming Score
Digit-5	190	0.695	0.200	0.800
All-Digits	190	0.695	0.133	0.725

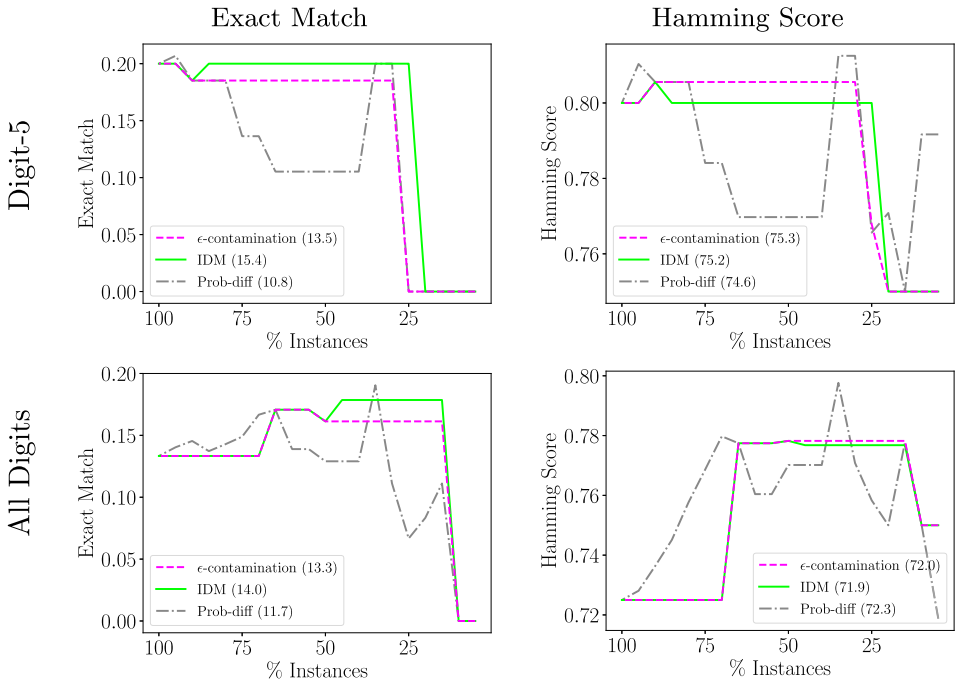


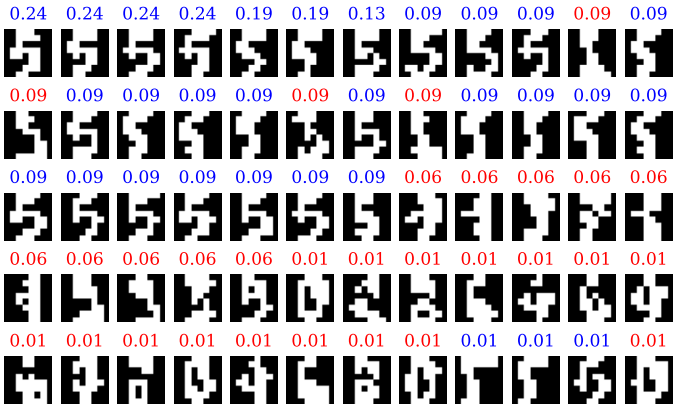
Fig. 13. Test-set performance vs. proportion of robust instances for the completion task on the Digits dataset, and the AUC in parentheses next to the legend. Top: test set contains only instances of the digit 5. Bottom: test set contains half of the instances of the digit 5 and half of the instances of other digits.

The results are shown in Fig. 13. The top row shows the performance when the test set contains only instances of digit five (i.e., no outliers), and the bottom row shows the performance when the test set contains half of out-of-distribution instances.

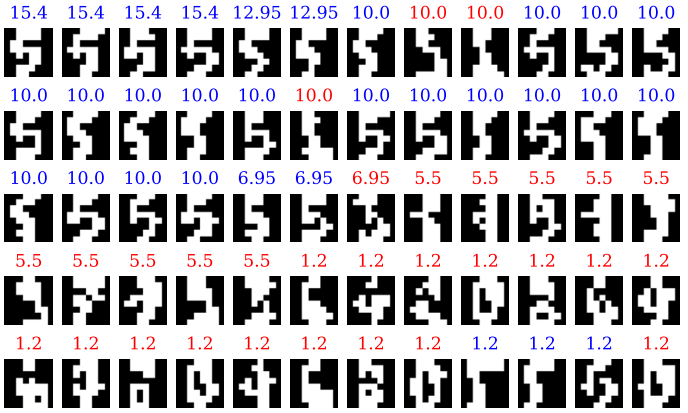
From the plots of the test set performance without out-of-distribution instances, we see that the baseline fails at ranking instances with higher accuracy as more robust than instances with smaller accuracy; that is, the performance (EM or HS) highly oscillates and correlates poorly with the percentage of robust instances. The CSPN-based methods do not show the same oscillation. While these methods also fail at discriminating the more accurate instances, they show non-decreasing performance except when the fraction of robust instances falls bellow 25%. One possible explanation is that robustness of instances indeed correlates poorly with their accuracy in this test set. If this is the case, then the results can be interpreted as showing that the baseline finds spurious correlations and actually hurts overall accuracy if decisions are suspended based on that criterion, while the CSPN-based methods show no change in accuracy if decision is suspended except for the more extreme scenarios. Yet, there is no method that outperforms the others for any choice of percentage of robust instances, although the advantage of the CSPN-based methods spans a large part of the decision thresholds, and on average (as indicated by the areas under the curves).

Regarding the test set with out-of-distribution instances, we see that the performance of the CSPN-based methods increases monotonically as we act more conservatively and more decisions are suspended. The baseline shows a similar and greater improvement for the first quarter (i.e., when the number of robust instances is above 75%) then starts to correlate poorly with accuracy. Again, there is no method that is always superior. The CSPN-based methods obtain a higher average performance with respect to EM, while average performance with respect to HS is very similar (with the baseline being slightly superior).

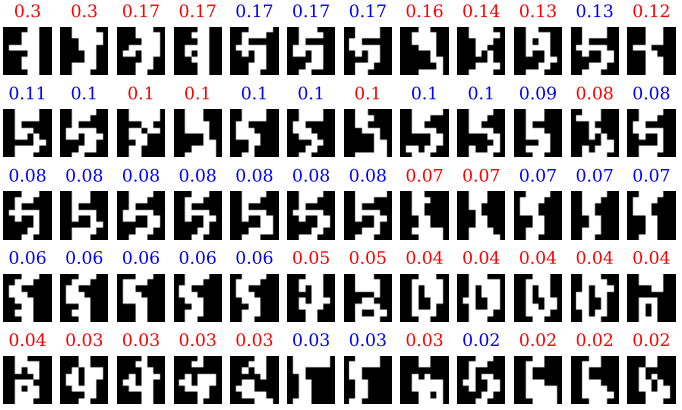
To better investigate how out-of-distribution instances affect the ability to discriminate robust and non-robust instances, we computed for each instance of the test set a *robustness index* given by the maximum value of the parameter under which that instances are deemed robust for each of the approaches. Fig. 14 shows the completed test digits ordered by decreasing value of (normalized) robustness index. Numbers in blue indicate instances of digit five, and numbers in red indicate out-of-distribution instances.



(a) ϵ-contamination



(b) IDM



(c) Probability difference

Fig. 14. Completed digits ordered by decreasing robustness; red numbers denote out-of-distribution instances.

We see from the figure that the robustness index obtained with CSPNs assigns for most of the completions of instances of digit five a higher score than for completions of out-of-distribution instances. The orderings produced with either ϵ-contamination or IDM are qualitatively identical, and assign low robustness index to instances of digit five which are arguably more dissimilar from typical instances. The baseline using the precise model is less successful at discriminating out-of-distribution instances. It assigns high scores to several completions of out-of-distribution instances, including the three highest scored instances. These results are certainly preliminary, and should be further investigated in the future.

Table 6
Characteristics of Multi-Label Datasets: number of training instances, number of test instances, number of evidence variables, number of labels (L), label cardinality (LC) and label density (LD).

Dataset	#Training	#Test	#Evidence	L	LC	LD
Arts	5389	1496	500	26	1.65	0.06
Business	8073	2244	500	30	1.6	0.05
CAL500	361	100	68	174	26.04	0.15
Emotions	426	119	72	6	1.87	0.31
Flags	140	38	19	7	3.39	0.48
Health	6626	1842	500	32	1.64	0.05
Human	2235	622	440	14	1.19	0.08
Plant	703	196	440	12	1.08	0.09
Scene	1734	480	294	6	1.07	0.18
Yeast	1739	484	103	14	4.24	0.30

6.4. Multilabel classification

In order to evaluate the approaches on tasks where the MAP variables are more meaningfully assigned, we performed a robustness analysis of multilabel classification on 10 benchmark datasets.[2] Table 6 shows general characteristics of the datasets used: number of training instances, number of test instances N, number of evidence variables M, number of labels L, label cardinality $LC = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} x^*_{ij}\hat{x}_{ij}$ and label density $LD = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} x^*_{ij}\hat{x}_{ij}}{R}$.

Multilabel classification consists assigning zero or more labels to an instance (as opposed to standard multiclass classification where each instance is assigned a single class label). As previously mentioned, we use a vector of binary variables $\mathbf{X}$ to represent the presence/absence of labels and run MAP inference to obtain a classification.

In addition to Exact Match, we also evaluate multilabel classifications by a metric commonly used in the multilabel classification literature [47,46], which rewards correctly predicted labels while discounting for incorrectly predicted ones. We name this metric *Accuracy* (Acc for short), and compute it as:

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} x^*_{ij}\hat{x}_{ij}}{\sum_{j=1}^{L}(x^*_{ij} + \hat{x}_{ij} - x^*_{ij}\hat{x}_{ij})},$$

where again N is the test dataset size, $\mathbf{x}^*$ is the MAP inference and $\hat{\mathbf{x}}$ the true labels. We do not use Hamming Score, as it does not distinguish between correct predictions of presence and absence of labels, and leads to overoptimistic measures when the number of labels is high and label density is low.

Similar to the previous task, we select values for ϵ, s and probability difference threshold so that roughly either 10% or 50% of instances are robust. The results for parameters selected so that 50% of instances are robust are summarized in Table 7. We omit the results of experiments with the CAL500 dataset, as all instances in this set contained more than one MAP instantiation and are therefore deemed non-robust by all methods. Numbers in boldface denote statistically significant superior performance according to a one-sided Welch t-test.

The results show that all three methods perform, on average, very similarly. Some notable exceptions appear for the Arts and Health datasets. In the Arts dataset, all methods perform worse on the set of robust instances, but the accuracy of robust instances is much lower for CSPN-based approaches. In the Health dataset, robust instances according to the ϵ-contamination approach have statistically significant higher accuracy that the robust instances according to probability difference. Probability difference outperforms the other methods with respect to accuracy also in the Human dataset, but the difference is very small; the same phenomenon occurs (with CSPN-based methods outperforming probability difference) in the remaining datasets.

To better understand the lack of robustness as measured by each approach, we also compute the proportion of nonrobust instances where the true label configuration attests the nonrobustness of the MAP configuration. For the CSPN-based approaches, this is the case when

$$\max_{\mathbf{w}\in\mathcal{C}}\left(\frac{\mathrm{S}_{\mathbf{w}}(\hat{\mathbf{x}},\mathbf{e})}{\mathrm{S}_{\mathbf{w}}(\mathbf{x}^*,\mathbf{e})}\right) \geq 1\,.$$

For the SPN-based approach, this happens if $\mathtt{S}(\mathbf{x}^*,\mathbf{e}) - \mathtt{S}(\hat{\mathbf{x}},\mathbf{e})$ is smaller than the given threshold. The results are given in the column $<$ GT of the Table. The numbers in parentheses next to the proportions give the corresponding rank (the highest

[2] These datasets were used to build SPN-based classifiers in previous work [46,12], and were obtained from https://github.com/nicoladimauro/dcsn and http://meka.sourceforge.net.

Table 7
Performance of multilabel classifications when about 50% of instances are robust.

Dataset		Accuracy			Exact Match			< GT (Rank)
		Robust	¬Robust	ΔAcc	Robust	¬Robust	ΔEM	
Arts	ϵ	0.1	0.31	-0.21	0.07	0.23	-0.16	0.294 (3)
	s	0.08	0.34	-0.26	0.07	0.24	-0.17	0.323 (2)
	p	**0.19**	0.33	**-0.14**	**0.15**	0.2	**-0.05**	0.424 (1)
Business	ϵ	0.76	0.6	0.16	0.61	0.43	0.17	0.217 (2)
	s	0.78	0.61	0.17	0.64	0.42	0.22	0.273 (1)
	p	0.77	0.65	0.12	0.63	0.47	0.16	0.205 (3)
Emotions	ϵ	0.54	0.35	0.19	0.24	0.12	0.12	0.15 (2)
	s	0.51	0.35	0.16	0.22	0.12	0.1	0.115 (3)
	p	0.44	0.42	0.02	0.18	0.27	-0.09	0.31 (1)
Flags	ϵ	0.64	0.4	0.24	0.22	0.1	0.12	0.05 (3)
	s	0.64	0.41	0.17	0.22	0.1	0.12	0.125 (2)
	p	0.58	0.44	0.14	0.21	0.06	0.15	0.33 (1)
Health	ϵ	**0.65**	0.46	**0.19**	0.54	0.29	0.25	0.268 (1)
	s	0.61	0.49	0.12	0.51	0.31	0.2	0.267 (2)
	p	0.63	0.48	0.15	0.52	0.3	0.22	0.232 (3)
Human	ϵ	0.21	0.06	0.15	0.15	0.04	0.11	0.467 (2)
	s	0.21	0.06	0.15	0.15	0.04	0.11	0.689 (1)
	p	0.22	0.06	0.16	0.16	0.04	0.12	0.467 (2)
Plant	ϵ	0.36	0.12	0.24	0.35	0.12	0.23	0.439 (1)
	s	0.37	0.13	0.24	0.36	0.13	0.23	0.356 (2)
	p	0.37	0.13	0.24	0.36	0.13	0.23	0.069 (3)
Scene	ϵ	0.2	0.5	-0.3	0.19	0.35	-0.06	0.388 (1)
	s	0.25	0.38	-0.13	0.22	0.28	-0.06	0.371 (2)
	p	0.21	0.45	-0.24	0.2	0.31	-0.11	0.352 (3)
Yeast	ϵ	0.43	0.18	0.25	0.1	0	0.1	0.512 (2)
	s	0.43	0.18	0.25	0.1	0	0.1	0.65 (1)
	p	0.42	0.17	0.25	0.1	0	0.1	0.382 (3)

value is assigned the smaller rank). Arguably, we would like to maximize the cases where the ground truth (GT) labeling "dominate" the MAP inference for the nonrobust cases. According to the results, no method clearly outperforms the others. The mean ranks for these proportions for the ϵ-contamination, IDM and probability difference approaches are, respectively, 1.89, 1.75 and 2.22. That shows that CSPN-based approaches provide, in a sense, more justified decisions.

The results for about 10% of robust instances are shown in Table 8. The results are qualitatively the same. CSPNs obtained by ϵ-contamination now achieve the best performance in the Arts dataset, but CSPNs obtained with IDM still perform poorly. In the Health and Scene datasets, using the CSPNs-based approaches to determine robust instances leads to statistically significantly higher accuracy among these instances, as well as a larger difference with respect to the deemed non-robust instances. Regarding the proportion of nonrobust instances which are dominated by the true label for each method (last column), we see that rankings for each dataset change with respect to the 50% analysis, but overall remain qualitatively similar for all methods. Note however the case of the Plant dataset, where none of the instances considered nonrobust by the probability difference were dominated by the true label (while this happened for most of 50% of the instances with the CSPN based approaches). The mean ranking for ϵ-contamination, IDM and probability difference are, respectively, 2.22, 1.55 and 2.125. Thus for this more conservative scenario, on average, the probability difference provides as justifiable decision as ϵ-contamination; the IDM based approach however remains the lowest mean rank.

6.5. Discussion

Overall, our results suggest that a CSPN-based robustness analysis can be carried out efficiently, and provide useful information for the analyst. It very often helps in detecting the most accurate instances, and is more robust to small training datasets and the presence of outliers in the test-set than using the probability estimates of the model. We also note that CSPN-based analysis provides an additional information, and of a different sort. They show to what extent predictions are (in)sensitive to perturbations of the parameters; such perturbations can model distribution changes that might arise when a model is used in different scenarios or across a long time period.

7. Conclusion

Sum-Product Networks (SPNs) are deep probabilistic models, that have shown very promising results in several machine learning tasks. Selective SPNs are a subclass of those models that allows for tractable MAP inference with a small decrease in model accuracy. In this work, we developed a polynomial-time algorithm for robust analysis of MAP inference in tree-shaped selective SPNs. We showed that performing the same task in non-selective SPNs is coNP-hard, further justifying our requirement of selectivity. We evaluated our algorithms in two different tasks: completing missing values and performing multilabel classification. Our results show that the proposed algorithm is often better at discrimination of robust and non-

Table 8
Performance of multilabel classifications when about 10% of instances are robust.

Dataset		Accuracy			Exact Match			< GT (Rank)
		Robust	¬Robust	ΔAcc	Robust	¬Robust	ΔEM	
Arts	ϵ	**0.88**	0.2	**0.68**	**0.83**	0.14	**0.69**	0.29 (3)
	s	0	0.27	-0.27	0	0.2	-0.2	0.418 (1)
	p	0.83	0.2	0.63	0.77	0.15	0.62	0.382 (2)
Business	ϵ	0.76	0.6	0.16	0.61	0.43	0.18	0.217 (2)
	s	0.78	0.65	0.13	0.64	0.48	0.16	0.293 (1)
	p	0.77	0.65	0.12	0.63	0.47	0.16	0.205 (3)
Emotions	ϵ	0.64	0.41	0.23	0.26	0.16	0.1	0.47 (2)
	s	0.69	0.41	0.28	0.31	0.16	0.15	0.613 (1)
	p	0.6	0.5	0.1	0.18	0.18	0	0.132 (3)
Flags	ϵ	0.92	0.47	0.45	0.5	0.12	0.38	0.05 (3)
	s	0.92	0.47	0.45	0.5	0.12	0.38	0.676 (1)
	p	0.92	0.47	0.45	0.5	0.12	0.38	0.095 (2)
Health	ϵ	**0.73**	0.55	**0.18**	0.56	0.41	0.15	0.159 (3)
	s	0.66	0.5	0.11	0.56	0.33	0.23	0.419 (1)
	p	0.61	0.47	0.14	0.5	0.31	0.19	0.197 (2)
Human	ϵ	0.21	0.06	0.15	0.15	0.04	0.11	0.467 (3)
	s	0.21	0.06	0.15	0.15	0.04	0.11	0.689 (2)
	p	0.22	0.06	0.16	0.16	0.04	0.12	0.724 (1)
Plant	ϵ	0.23	0.27	-0.03	0.22	0.27	-0.05	0.537 (1)
	s	0.25	0.26	-0.01	0.24	0.26	-0.02	0.533 (2)
	p	0.25	0.27	-0.02	0.24	0.26	-0.02	0 (3)
Scene	ϵ	**0.75**	0.26	**0.49**	**0.75**	0.19	**0.56**	0.431 (1)
	s	0.2	0.33	-0.13	0.2	0.28	-0.08	0.411 (2)
	p	0.25	0.3	-0.05	0.25	0.26	-0.01	0.234 (3)
Yeast	ϵ	0.43	0.18	0.25	0.1	0	0.1	0.201 (2)
	s	0.43	0.18	0.25	0.1	0	0.1	0.198 (3)
	p	0.42	0.17	0.25	0.1	0	0.1	0.231 (1)

robust inferences than the standard approach based on the difference of probabilities, especially when data is scarce or the test set contains outliers (i.e., out-of-distribution instances). It also shows that our approaches can be used alongside standard approaches to provide additional information to the analyst with little computational overhead. Even though we have carried extensive experimentation, our empirical analysis is somewhat preliminary, and even more experiments are needed in the future to better identify cases where our approaches excel. We also left open the complexity of robustness analysis in multiply-connected SPNs.

CRediT authorship contribution statement

Julissa Villanueva Llerena: Conceptualization, Formal analysis, Investigation, Software, Validation, Visualization, Writing - original draft. **Denis Deratani Mauá:** Conceptualization, Investigation, Methodology, Supervision, Validation, Visualization, Writing - review & editing.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

Acknowledgements

We thank Robert Peharz for kindly sharing the source code of the Selective SPN learning algorithm. This study was financed in part by the Brazilian Agency *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) - Finance Code 001. The second author also received financial support from CNPq grants No. 304012/2019-0 (PQ) and 420669/2016-7, and from the São Paulo Research Foundation (FAPESP) grant No. 2019/07665-4.

References

[1] H. Poon, P. Domingos, Sum-product networks: a new deep architecture, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, 2011, pp. 337–346.
[2] A. Darwiche, A differential approach to inference in Bayesian networks, J. ACM 50 (3) (2003) 280–305.
[3] M. Chavira, A. Darwiche, Compiling bayesian networks with local structure, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, 2005, pp. 1306–1312.
[4] J. Huang, M. Chavira, A. Darwiche, Solving map exactly by searching on compiled arithmetic circuits, in: Proceedings of the 21st National Conference on Artificial Intelligence, 2006, pp. 1143–1148.
[5] D. Lowd, P. Domingos, Learning arithmetic circuits, in: Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence, 2008, pp. 383–392.
[6] M.R. Amer, S. Todorovic, Sum product networks for activity recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2016) 800–813.
[7] K. Zheng, A. Pronobis, R.P. Rao, Learning graph-structured sum-product networks for probabilistic semantic maps, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 4547–4555.
[8] A. Pronobis, F. Riccio, R.P. Rao, Deep spatial affordance hierarchy: spatial knowledge representation for planning in large-scale environments, in: ICAPS 2017 Workshop on Planning and Robotics, 2017, pp. 1–9.
[9] A. Pronobis, R.P. Rao, Learning deep generative spatial models for mobile robots, in: EEE/RSJ International Conference on Intelligent Robots and Systems, 2017, pp. 755–762.
[10] W.-C. Cheng, S. Kok, H.V. Pham, H.L. Chieu, K.M.A. Chai, Language modeling with sum-product networks, in: 15th Annual Conference of the International Speech Communication Association, 2014, pp. 2098–2102.
[11] R. Peharz, A. Vergari, K. Stelzner, A. Molina, M. Trapp, K. Kersting, Z. Ghahramani, Probabilistic deep learning using random sum-product networks, CoRR, arXiv:1806.01910 [abs].
[12] J. Villanueva, D.D. Mauá, On using sum-products networks for multi-label classification, in: Proceedings of the 6th Brazilian Conference on Intelligent Systems, 2017, pp. 25–30.
[13] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[14] R. Peharz, R. Gens, F. Pernkopf, P. Domingos, On the latent variable interpretation in sum-product networks, J. Trans. Pattern Anal. Mach. Intell. (2016) 1–14.
[15] D. Conaty, D.D. Mauá, C.P. de Campos, Approximation complexity of maximum a posteriori in sum-product networks, in: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, 2017, pp. 322–331.
[16] R. Peharz, R. Gens, P. Domingos, Learning selective sum-product networks, in: Workshop on Learning Tractable Probabilistic Models, 2014.
[17] A. Choi, A. Darwiche, On relaxing determinism in arithmetic circuits, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 825–833.
[18] T. Rahman, S. Jin, V. Gogate, Look ma, no latent variables: accurate cutset networks via compilation, in: Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 97, 2019, pp. 5311–5320.
[19] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer-Verlag, 1985.
[20] H. Jiang, B. Kim, M. Guan, M. Gupta, To trust or not to trust a classifier, in: Advances in Neural Information Processing Systems, 2018, pp. 5541–5552.
[21] M. Sensoy, M. Kandemir, L. Kaplan, Evidential deep learning to quantify classification uncertainty, in: Advances in Neural Information Processing Systems, 2018, pp. 3179–3189.
[22] A. Malinin, M. Gales, Predictive uncertainty estimation via prior networks, in: Proceedings of the 32nd International Conference on Neural Information Processing System, 2018, pp. 7047–7058.
[23] E. Castillo, J.M. Gutiérrez, A.S. Hadi, Sensitivity analysis in discrete Bayesian networks, IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum. (1997) 412–423.
[24] H. Chan, A. Darwiche, Sensitivity analysis in bayesian networks: from single to multiple parameters, in: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, 2004, pp. 67–75.
[25] K.B. Laskey, Sensitivity analysis for probability assessments in Bayesian networks, IEEE Trans. Syst. Man Cybern. (1995) 901–909.
[26] U. Kjærulff, L.C. van der Gaag, Making sensitivity analysis computationally efficient, in: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, 2000, pp. 317–325.
[27] H. Chan, A. Darwiche, When do numbers really matter?, J. Artif. Intell. Res. (2002) 265–287.
[28] D.D. Mauá, D. Conaty, F.G. Cozman, K. Poppenhaeger, C.P. de Campos, Robustifying sum-product networks, Int. J. Approx. Reason. (2018) 163–180.
[29] H. Chan, A. Darwiche, On the robustness of most probable explanations, in: Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence, 2006, pp. 63–71.
[30] S. Renooij, L.C. Van Der Gaag, Evidence and scenario sensitivities in naive bayesian classifiers, Int. J. Approx. Reason. (2008) 398–416.
[31] A. Antonucci, A. Facchini, L. Mattei, Credal sentential decision diagrams, in: Proceedings of the Eleventh International Symposium on Imprecise Probabilities: Theories and Applications, in: Proceedings of Machine Learning Research, vol. 103, 2019, pp. 14–22.
[32] L. Mattei, A. Antonucci, D.D. Mauá, A. Facchini, J. Villanueva Llerena, Tractable inference in credal sentential decision diagrams, Int. J. Approx. Reason. https://doi.org/10.1016/j.ijar.2020.06.005.
[33] R. Peharz, S. Tschiatschek, F. Pernkopf, P. Domingos, On theoretical properties of sum-product networks, in: Artificial Intelligence and Statistics, 2015, pp. 744–752.
[34] Y. Liang, J. Bekker, G.V. den Broeck, Learning the structure of probabilistic sentential decision diagrams, in: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, 2017.
[35] B.M. Sguerra, F.G. Cozman, Image classification using sum-product networks for autonomous flight of micro aerial vehicles, in: Proceedings of the 5th Brazilian Conference on Intelligent Systems, 2016, pp. 139–144.
[36] J. Mei, Y. Jiang, K. Tu, Maximum a posteriori inference in sum-product networks, in: The 32nd AAAI Conference on Artificial Intelligence, 2017, pp. 1923–1930.
[37] R. Peharz, R. Gens, F. Pernkopf, P. Domingos, On the latent variable interpretation in sum-product networks, IEEE Trans. Pattern Anal. Mach. Intell. (2017) 2030–2044.
[38] I. Levi, The Enterprise of Knowledge, MIT Press, London, 1980.
[39] L.M. de Campos, J.F. Huete, S. Moral, Probability interval: a tool for uncertain reasoning, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 2 (2) (1994) 167–196.
[40] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, 2nd Edition, Springer Series in Statistics, Springer, 1993.
[41] P. Walley, Inferences from multinomial data: learning about a bag of marbles, J. R. Stat. Soc. B 58 (1) (1996) 3–34.
[42] H. Bodlaender, F. van den Eijkhof, L. van der Gaag, On the complexity of the MPA problem in probabilistic networks, in: Proceedings of the 15th European Conference on Artificial Intelligence, 2002, pp. 675–679.
[43] J. Davis, P. Domingos, Bottom-up learning of Markov network structure, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 271–280.

[44] T. Zemicheal, T.G. Dietterich, Anomaly detection in the presence of missing values for weather data quality control, in: Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies, 2019, pp. 65–73.
[45] E. Alpaydin, C. Kaynak, Cascading classifiers, Kybernetika 34 (1998) 369–374.
[46] N. Di Mauro, A. Vergari, F. Esposito, Multi-label classification with cutset networks, in: Proceedings of the 8th International Conference on Probabilistic Graphical Models, 2016, pp. 147–158.
[47] K. Dembczyński, W. Waegeman, W. Cheng, E. Hüllermeier, On label dependence and loss minimization in multi-label classification, Int. J. Mach. Learn. 88 (1–2) (2012) 5–45.