Robust Analysis of MAP Inference in Selective Sum-Product Networks

Julissa Villanueva Llerena JGVILLE@IME.USP,BR
Denis Deratani Mauá DDM@IME.USP.BR
Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Brazil

Abstract

Sum-Product Networks (SPN) are deep probabilistic models that have shown to achieve state-of-the-art performance in several machine learning tasks. As with many other probabilistic models, performing Maximum-A-Posteriori (MAP) inference is NP-hard in SPNs. A notable exception is selective SPNs, that constrain the network so as to allow MAP inference to be performed in linear time. Due to the high number of parameters, SPNs learned from data can produce unreliable and overconfident inference; this phenomenon can be partially mitigated by performing a Robustness Analysis of the model predictions to changes in the parameters. In this work, we address the problem of assessing the robustness of MAP inferences produced with Selective SPNs to global perturbations of the parameters. We present efficient algorithms and an empirical analysis with realistic problems.

Keywords: Robust statistics, sensitivity analysis, sum-product networks, tractable probabilistic models.

1. Introduction

Sum-Product Networks (SPNs) are a class of deep probabilistic graphical models with an intuitive semantics given by context-specific independences [27, 23, 34, 14, 31]. An SPN represents a probability distribution over a set of random variables by a rooted directed-acyclic graph with tractable distributions as leaves, and weighted sums and product operations as inner nodes [27].

SPNs have obtained impressive results in many machine learning tasks due to their ability to represent complicated multidimensional distributions [27, 1, 37, 29, 28, 8, 26]. While SPNs allow for linear time predictive inference (i.e., computing the probability of a joint configuration of the variables), many tasks are more effectively solved by finding a maximal probability joint configuration of the variables that is consistent with a given evidence, a problem known as Maximum-A-Posteriori (MAP) inference [27, 35, 10]. This problem is known to be NP-hard even to approximate, and even for shallow structures [24, 10]. 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 and parallelizable greedy algorithm [23, 24]. While selective SPNs are less expressive than non-selective SPNs [9, Theorem 4], empirical results have shown that learned selective SPNs obtain performance comparable to learned non-selective SPNS [23].

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 [2]. 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. The former evaluate the effect of perturbation in the model parameters on the value of a particular inference. The latter concerns the classification of inferences into robust or not, depending on how an inference changes according to changes 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 [15, 32, 20, 4, 6, 18, 16, 5]. Qualitative sensitivity analysis received much less attention, with some notable examples [21, 7, 30]. A second distinction can be made between *local* and *global* analyses. The former considers the effect of the perturbation of a single parameter, or a small group of related parameters; the latter aims at more general perturbations possibly affecting all the parameters of the model simultaneously. Initial work on algorithms for sensitivity analysis in probabilistic graphical models focused on local perturbations [5, 18]; more recently, algorithms for global analyses have been proposed [4, 6, 16].

The only approach to global sensitivity analysis in SPNs is the recent work by Mauá et al. [21], who considered *Credal SPNs*, that is, sets of SPNs obtained by a simultaneous perturbation of all model parameters. They developed efficient algorithms for producing upper and lower bound on the probability of evidence, and for performing credal classification for small number of classes. Importantly, they showed that performing global quantitative sensitivity analysis of MAP inference in these models is NP-hard.

In this work we develop efficient algorithms for global quantitative analysis of MAP inference in selective SPNs. In particular, we devise a polynomial-time procedure to decide whether a given MAP configuration is admissible

with respect to a selective credal SPN, that is, whether the configuration is the single maximizer of all SPNs in the set. 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 6, we show the experimental results of our robustness analysis. Finally, we conclude the paper and discuss possible improvements in Section 7.

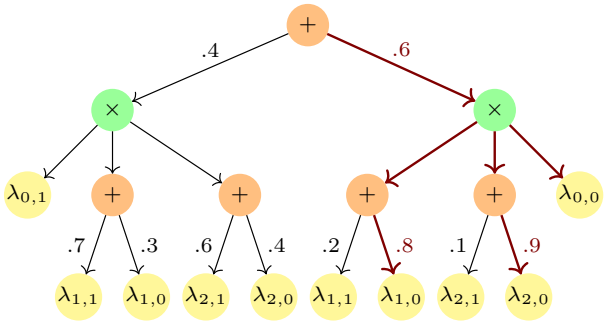

Figure 1: Selective Sum-Product Network over three binary variables.

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., $\mathscr{Y}$) and Random Variables (RV) in capital letters (e.g., X_i). A collection of RVs $X_i, i \in \mathscr{Y}$ is written as $\mathbf{X}_{\mathscr{Y}}$, or simply $\mathbf{X}$ when the index set is not important. The set of values that a RV X_i assumes is denoted as $val(X_i)$ and the set of all realizations of a collection of RVs $\mathbf{X}$ is denoted by $val(\mathbf{X}) = \times_{i=1}^{N} val(X_i)$, where $\times$ denotes the Cartesian product. An element of $val(\mathbf{X})$ is written as $\mathbf{x}$. In this work we assume that RVs take on a finite number of values; therefore we can represent every random variable X_i with a set of *indicator variables* $\{\lambda_{i,k} : k = 0, \ldots, m-1\}$, where $\lambda_{i,k}$ denotes that X_i takes on its k-th value (for some arbitrary ordering of $val(X_i)$). We denote an arbitrary specification of the indicator variables associated with RVs $\mathbf{X}_{\mathscr{Y}}$ as λ.

3. Selective Sum-Product Networks

A Sum-Product Network $\mathtt{S}$ is a rooted weighted acyclic directed graph with indicator variables as leaves, and sum and product operations as internal nodes. The arcs $i \to h$ of the network are associated with non-negative weights w_{ij} such that arcs leaving product nodes are assigned weight one (and usually omitted). The set of all weights is denoted $\mathbf{w}$. If i is a node in SPN $\mathtt{S}$ 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 $ch(i)$. The *scope* of an SPN is the set of RVs associated with the indicator variables in the network. Figure 1 shows an SPN with scope X_0, X_1, X_2.

The evaluation of an SPN $\mathtt{S}$ at a realization $\mathbf{x}$ of its scope is defined recursively: $\mathtt{S}(\mathbf{x}) = 1$ if $\mathtt{S}$ is an indicator node consistent with $\mathbf{x}$; $\mathtt{S}(\mathbf{x}) = 0$ if $\mathtt{S}$ is an indicator node not consistent with $\mathbf{x}$; $\mathtt{S}(\mathbf{x}) = \sum_{j \in ch(i)} w_{ij} \mathtt{S}^j(\mathbf{x})$, where i is the root; and $\mathtt{S}(\mathbf{x}) = \prod_{j \in ch(i)} \mathtt{S}^j(\mathbf{x})$. This value can be computed in linear time by traversing the network from the leaves towards the root (caching values). For example, the evaluation of the SPN in Figure1 at $X_0 = 1, X_1 = 1$ and $X_2 = 0$ is $\mathtt{S}(\mathbf{x}) = 0.4(1 \times 0.7 \times 0.4) + 0.6(0.2 \times 0.9 \times 0) = 0.112$.

In order to ensure that an SPN represents a valid distributions [27], we impose the following conditions:

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 Figure 1 is complete, decomposable, and normalized. For the rest of this paper, we assume that SPNs are complete, decomposable and normalized.

To ensure linear time MAP inference [23], we also impose the following additional property:

Selectivity: At most one child sub-network of any sum node evaluates to nonzero at any realization of its scope.

The SPN in Figure 1 is selective. An alternative way to state the above condition is to assume that the supports of the distributions induced by the children of any sum node are disjoint. This implies for any realization $\mathbf{x}$ and sum node i that $\mathtt{S}^i(\mathbf{x}) = w_{ij}\mathtt{S}^j(\mathbf{x})$ for $j \in ch(i)$.

Given a realization $\mathbf{x}$ 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 realization. The *calculation tree* $\mathtt{T}_{\mathtt{S}}(\mathbf{x})$ of an SPN $\mathtt{S}$ is the SPN induced by the sequence of *active nodes* for $\mathbf{x}$ rooted at the root of $\mathtt{S}$. Due to the decomposition property, the calculation tree of a selective SPN is a tree, hence justifying its name. 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 Figure 1, the sub-SPN induced by highlighted arcs represents 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 dataset $\mathscr{D}$ of i.i.d. realizations $\mathbf{x} \in val(\mathbf{X})$, Equation (1) allows us to obtain a closed-form maximum likelihood estimator for the weights of a fixed-structure selective SPN $\mathtt{S}$ by

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

where $N_k = |\mathbf{x} \in \mathscr{D} : k \in \mathtt{T}_\mathtt{S}(\mathbf{x})|$ is the number of calculation trees containing node k. We can obtain more robust estimators by smoothing the count above, for example, by using the estimator $w_{ij} = (N_j + s_j)/(N_j + s)$ for $s = \sum_j s_j$. A common choice is Laplace smoothing, which uses $s_j = 1$. Note that this is equivalent to obtain Bayesian estimators assuming independent Dirichlet priors on the weights associated with a sum node.

Inspired by structure learning of standard probabilistic graphical models [17] and Equation (2), Peharz et al. [23] proposed learning selective SPNs by maximizing a penalized log-likelihood function that discounts the size of the graph of the data log-likelihood. They developed an algorithm that learn selective SPNs by performing a greedy hill-climbing search in the space of structures.

More recently, Liang et al. [19] developed an algorithm to learning Probabilistic Setential Decision Diagrams, a similar type of probabilistic models which can efficiently be translated into selective sum-product networks. At each step, their algorithm performs local modifications to the network that improve the test-set penalized log-likelihood. They show state-of-the-art performance according to test-set likelihood in benchmarks when learning ensemble of PSDDs using bagging and an EM-like procedure. Note however that the use of ensembles makes MAP inference NP-hard [10].

3.2. Maximum-A-Posteriori Inference

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

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

Although MAP inference is NP-hard in SPN even when restricted to binary variables and height-two networks [10, 33, 22, 25], the problem is solvable in linear time in the size of the network for Selective SPNs by the simple Max-Product algorithm [23, 25]. This algorithm consists in replacing sum operations with maximizations (obtaining a so-called Max-Product network), then evaluating the network by traversing nodes from the leaves towards the root. We say that a leaf node is *consistent* if its scope is in $\mathbf{X}$, or if its scope is $E_j \in \mathbf{E}$ and the node evaluates to 1 at $\mathbf{e}$. The algorithm can be described by the following recursion (caching values to achieve linear time):

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

The corresponding MAP instantiation is obtained by backtracking the solutions of the maximizations from the root toward the leaves. The circuit tree in Figure 1 contains the arcs selected by Max-Product when backtracking from the root (i.e., it contains the arguments of the maximizations).

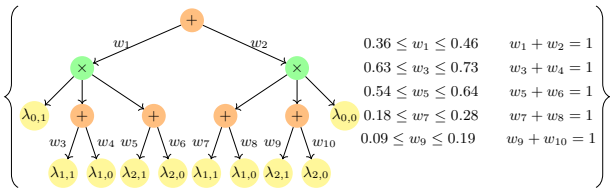

Figure 2: Credal Sum-Product Network obtained by 0.1-contamination of the Sum-Product Network in Figure 1.

4. Credal Sum-Product Networks

We denote by $\mathtt{S}_\mathbf{w}$ an SPN whose weights are $\mathbf{w}$. A *Credal SPN* (CSPN) is a set of (consistent, decomposable and normalized) SPNs $\{\mathtt{S}_\mathbf{w} : \mathbf{w} \in \mathscr{C}\}$ which all share the same network structure [21]. The space of weights $\mathscr{C}$ is usually taken as the Cartesian product of closed and convex sets of weights $\mathscr{C}_i$, one for each sum node i in the network. Since, the networks are normalized, a CSPN induces a credal set over the weights associated with outgoing edges of each sum node. An example of CSPN is shown in Figure 2.

Mauá et al. [21] developed a polynomial-time algorithm to compute the upper joint probability of a given instantiation induced by a given a Credal SPN $\mathtt{S}_\mathbf{w}$ and a realization $\mathbf{x}$ of its variables. The algorithm visits nodes in topological reverse ordering, evaluating the following expression at

each node i (we omit words "leaf" and "node" below):

$$L^i(\mathbf{x},\mathbf{e}) = \begin{cases} 1 & \text{if } i \text{ is consistent,} \\ 0 & \text{if } i \text{ is inconsistent,} \\ \prod_j L^j(\mathbf{x},\mathbf{e}) & \text{if } i \text{ is a product,} \\ \max_{\mathbf{w}_i \in \mathscr{C}_i} \sum_{j \in ch(i)} w_{ij} L^j(\mathbf{x},\mathbf{e}) & \text{if } i \text{ is a sum.} \end{cases} \tag{3}$$

The desired value is obtained at the root of the network, that is, $L^r(\mathbf{x},\mathbf{e}) = \max_{\mathbf{w}} \mathtt{S}_{\mathbf{w}}(\mathbf{x},\mathbf{e})$. Note that for selective credal SPNs, the sum in the last equation has at most one positive term.

4.1. Credal SPNs for Robustness Analysis

We can investigate the robustness of inferences drawn with a (precise) SPN by comparing the inferences produced by a CSPN built "around" the investigated SPN. A common approach is to obtain a CSPN by applying a perturbation to each weight in the network, while still respecting the normalization property. Given $\varepsilon \in (0,1)$, the ε-contamination of the vector $\mathbf{u}$ is the set:

$$\mathscr{C}_{\mathbf{u},\varepsilon} = \left\{ (1-\varepsilon)\mathbf{u} + \varepsilon \mathbf{v} : v_j \geq 0, \sum_j v_j = 1 \right\}. \tag{4}$$

We say that a CSPN is obtained by ε-contamination of an SPN $\mathtt{S}_{\mathbf{w}}$ if the weights associated with each sum node vary in the set obtained by the ε-contamination of the weights in $\mathtt{S}$:

$$\{\mathtt{S}_{\mathbf{w}} : \mathbf{w}_i \in \mathscr{C}_{\mathbf{w},\varepsilon}, i \text{ is a sum node}\}, \tag{5}$$

where $\mathbf{w}_i$ denotes the weights associated with sum node i.

Alternatively, we can obtain a CSPN by varying the priors used for estimating the parameters of the SPN. Recall from Equation (2) that the weights of a selective SPN can be estimated as a Multinomial-Dirichlet model with counts N_j, and prior strength s. Thus, we can obtain a local credal set for a sum node i given a count vector N as:

$$\mathscr{C}_{N,s} = \left\{ \mathbf{w}_i : w_{ij} = \frac{N_j + s \cdot v_i}{N_i + s}, v_j \geq 0, \sum_j v_j = 1 \right\}. \tag{6}$$

The equation above coincides with the Imprecise Dirichlet Model [36] for a (latent) RV representing the sum node i, whose value "selects" which edge is active.

5. Robustness of MAP Inferences in SPNs

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

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

Note that for SPNs (i.e., when $\mathscr{C}$ is a singleton) the problem reduces to deciding if there exists an instantiation $\mathbf{x}$ s.t. $\mathtt{S}(\mathbf{x},\mathbf{e}) > \mathtt{S}(\mathbf{x}^*,\mathbf{e})$, Bodlaender et al. [3] showed that this problem is NP-hard for Bayesian networks; one can adapt their proof using ideas from [10] to show that deciding if an instance $\mathbf{x}^*$ is robust is NP-hard.

Before formulating an algorithm for deciding robustness of MAP inference, let us consider the simpler problem of computing the MAP instantiation with maximum joint probability over the set of induced SPNs. This will be used as a subroutine to verify robustness later. Thus consider a credal SPN $\{\mathtt{S}_{\mathbf{w}} : \mathbf{w} \in \mathscr{C}\}$ where $\mathscr{C}$ is given by the Cartesian product of sets $\mathscr{C}_i$ for each sum node i, and some evidence $\mathbf{e}$. We are thus interested in computing

$$\max_{\mathbf{x}} \max_{\mathbf{w} \in \mathscr{C}} \mathtt{S}_{\mathbf{w}}(\mathbf{x},\mathbf{e}).$$

Call a leaf node of the network a *MAP leaf* if its scope is in $\mathbf{X}$, otherwise call it an *evidence leaf*. The following algorithm computes the above quantity by traversing the network, calculating at each node i:

$$M^i = \begin{cases} 1 & \text{if } i \text{ is consistent,} \\ 0 & \text{if } i \text{ is inconsistent,} \\ \prod_{j \in ch(i)} M^j & \text{if } i \text{ is a product node,} \\ \max_{j \in ch(i)} \max_{\mathbf{w}_i \in \mathscr{C}_i} w_{ij} M^j & \text{if } i \text{ is a sum node.} \end{cases}$$

The definition of consistent and inconsistent nodes applies to leaf nodes and is identical to the one used to define the Max-Product algorithm. The soundness and complexity of the algorithm above are given by the following result.

Theorem 1 *Consider a selective CSPN $\{\mathtt{S}_{\boldsymbol{w}} : \boldsymbol{w} \in \mathscr{C}\}$ with root r, where $\mathscr{C}$ is the Cartesian product of finitely-generated polytopes $\mathscr{C}_i$, one for each sum node i. Then M^r computes $\max_{\boldsymbol{x}} \max_{\boldsymbol{w}} \mathtt{S}_{\boldsymbol{w}}(\boldsymbol{x})$ in $O(|\mathtt{S}|)$ time, where $|\mathtt{S}|$ is the number of nodes and arcs in the model (assuming the local optimizations over the weights in M^i take linear time).*

Proof We prove corrrectness by induction in the height h of $\mathtt{S}$. 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 SPN, at most one child j of a sum node satisfies $\mathtt{S}^j(\mathbf{x},\mathbf{e}) > 0$ for each instantiation $\mathbf{x}$. Let $\mathscr{X}_j$ denote the set of instantiations $\mathbf{x} \in val(\mathbf{X})$ for which $\mathtt{S}^j(\mathbf{x},\mathbf{e}) > 0$. Then

$$\max_{\mathbf{x} \in val(\mathbf{X})} \max_{\mathbf{w}} \mathtt{S}^i_{\mathbf{w}}(\mathbf{x},\mathbf{e}) = \max_{j \in ch(i)} \max_{\mathbf{x} \in \mathscr{X}_j} \max_{\mathbf{w}} w_{ij} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x},\mathbf{e}).$$

Thus, if i is a sum node, it follows that

$$\max_{\mathbf{x}} \max_{\mathbf{w}} \mathtt{S}^i_{\mathbf{w}}(\mathbf{x},\mathbf{e}) = \max_{j \in ch(i)} \max_{\mathbf{w}_{ij} \in \mathscr{C}_{ij}} w_{ij} \max_{\mathbf{x}} \max_{\mathbf{w}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x},\mathbf{e})$$

$$= \max_{j \in ch(i)} \max_{\mathbf{w}_{ij} \in \mathscr{C}_{ij}} w_{ij} M^j = M^i .$$

If i is a product node, we have that

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

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

If we cache the values computed, then each node computes one value M^i in polynomial time in the number of edges of the network, considering that the local optimization over weights w_{ij} is polynomial (which is the case for most sensible credal SPNs). Therefore the total cost of this computation is $O(|\mathtt{S}|)$. ■

The following algorithm decides if a MAP instantiation is robust, by traversing the network from the leaves to the root, computing, for each node i:

$$V^i = \begin{cases} 1 & \text{if } i \text{ is consistent,} \\ 0 & \text{if } i \text{ is inconsistent,} \\ \prod_j V^j & \text{if } i \text{ is a product node,} \\ \max \left\{ \max_{j \in ch(i), j \neq k} U^j,\ V^k \right\} & \text{if } i \text{ is a sum node,} \end{cases}$$

where k is the active child of i for $\mathtt{S}$ at $\mathbf{x}^*, \mathbf{e}$, and

$$U^j = \max_{\mathbf{w}_i \in \mathscr{C}_i} \frac{w_{ij} M^j}{w_{ik} L^k(\mathbf{x}^*, \mathbf{e})} .$$

Where M^j and L^k was introduced at the previous section in equations 5 and 3 respectively.

The following result states the soundness and complexity of the algorithm for tree-shaped networks.

Theorem 2 *Consider a tree-shaped CSPN $\{\mathtt{S}_w : w \in \mathscr{C}\}$ with root r, where $\mathscr{C}$ is the Cartesian product of finitely-generated polytopes $\mathscr{C}_i$, one for each sum node i. Then V^r computes*

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

in $O(|\mathtt{S}|)$ time, where $|\mathtt{S}|$ is the number of nodes and arcs in the model (assuming the optimizations over w_i in U_j can be performed in linear time).

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 hypotheses 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 $\mathtt{S}^j$, $j \in ch(i)$, do not appear in any other sub-network $\mathtt{S}^{j'}$, $j' \in ch(i)$, $j' \neq j$. Denote by $\mathscr{X}_j$ the subset of $val(\mathbf{X})$ for which $\mathtt{S}^j(\mathbf{x}, \mathbf{e}) > 0$. We have that

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

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

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

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

$$\max_{\mathbf{w}} \frac{w_{ij} \max_{\mathbf{x} \in \mathscr{X}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{w_{ik} \mathtt{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} = \max_{\mathbf{w}_i \in \mathscr{C}_i} \frac{w_{ij} \max_{\mathbf{x}} \max_{\mathbf{w}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}, \mathbf{e})}{w_{ik} \min_{\mathbf{w}_k} \mathtt{S}^k_{\mathbf{w}_k}(\mathbf{x}^*, \mathbf{e})} = \max_{\mathbf{w}_i \in \mathscr{C}_i} \frac{w_{ij} M^j}{w_{ik} L^k(\mathbf{x}^*, \mathbf{e})} = U^j .$$

If i is a product node then

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

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. Hence, the total cost of the computation is $O(|\mathtt{S}^k|)$. ■

The complexity of deciding robustness for multiply-connected CSPNs remains an open question.

6. Experiments

We evaluate the ability of our proposed method to distinguish between robust and and non-robust MAP inferences in two different tasks. The first task requires learning a probabilistic model from complete data and then use that model to complete the missing values in the test data by maximizing the joint probability value of each instance. (this is also 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 multilabel classification, where objects are assignment multiple labels. This task can be solved by representing labels as a binary vector $\mathbf{X}$, which can be

Table 1: Characteristics of datasets used in completion tasks.

Dataset	Train	Test	Evidence	Query
DNA	1600	1186	90	90
Jester	9000	4116	50	50
NLTCS	16181	3236	8	8
MSNBC	291326	58265	9	8

predicted by a probabilistic model of relevance labels $\mathbf{X}$ given features $\mathbf{e}$.

We learn selective SPNs for both tasks using the algorithm by Peharz et al. [23]. We then obtain CSPNs by either ε-contamination (Equation (5)) or the Imprecise Dirichlet Model over the smoothed counts (Equation (6)). We use the CSPNs to determine the robustness of each MAP inference drawn with the (precise) SPN, and compare the performance of robust and non-robust classifications. We use a validation dataset to select values for ε and s that maximizes the performance of robust inferences (according to some metric, as discussed later). We also compare our methods against a baseline robust analysis that compare the difference between the probability of the MAP instatiation and the second-best MAP instatiantion (i.e., the second most probable configuration consistent with evidence). An instance is considered robust in this scheme if this difference is greater than a given threshold. We use a validation set to determine the best threshold for each dataset and task.

6.1. Completion

We learned selective SPNs from four well-known datasets for density estimation [11], available at https://github.com/arranger1044/DEBD. The selected datasets and their characteristics appear in Table 1. These datasets are complete and all variables have been binarized. We build completion tasks in the test sets by running MAP inference in the first 50% of the variables with the remaining variables given as evidence. We use the available training/validation/test partition in the datasets to learn, select parameters, and evaluate the methods, respectively.

We measure the performance of the completions by the Hamming Score (HS) and Exact Match (EM) metrics. Hamming score measures the percentage of correct value completions:

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

where $\mathbf{x}$ is the MAP instantiation, $\hat{\mathbf{x}}$ is the true completion, N is the number of instances and R is the number of missing values (50% of the number of variables in this case). Exact match computes the percentage of perfect completions:

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

For each metric,we select the values for $\varepsilon*$, $s*$ and the probability difference $p*$ by maximizing the performance of the robust instances on the validation set (according to some criterion). These values are kept fixed during evaluation in the test set.

The results are shown in Table 2, where $\varepsilon*$, $s*$ and $p*$ stand for ε-contamination, IDM and probability difference criterion, with the parameters and threshold selected using a validation set, as described. The best performing method for each dataset and partition is shown in boldface. We omit the results for the DNA dataset, as all instances had more than on MAP instantiation, making all inferences non-robust by any of the criteria.

According to the results, ε-contamination outperforms all other methods in terms of both Hamming Score and Exact Match. It also achieves the highest discrepancy between the performance of robust and non robust predictions. IDM is outperformed by the other methods in two datasets.

Figure 3 shows the test-set performance of robust instances in the test set as for different values of ε, s and different thresholds for probability difference. The curves with green triangle marks shown Exact Match, while the curves in orange square marks denote Hamming scores. We see that ε-contamination improves performance as the value of ε is increased, while the correlation of the threshold for probability difference and accuracy is less clear. IDM obtains intermediary results. All three approaches perform similarly at the highest values.

6.2. Multilabel Classification

We perform the robustness analysis of predictions on 10 benchmark of multilabel classification domains.[1] Table 3 shows general characteristics of the datasets used: domain, number of features M, number of instances N, number of labels L, label cardinality $LC = \frac{1}{N}\sum_{i=1}^{N}|\mathbf{x}_i|$ and label density $LD = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{x}_i|}{R}$, where $|\mathbf{x}|$ returns the number of ones in the instantiation $\mathbf{x}$.

In addition to Exact Match, we also evaluate multilabel classifications by a common measure [12, 13], which rewards relevant predictions while discounting for irrelevant predictions. We name this metric *Accuracy* (Acc), and compute it as:

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{x}^*_i \wedge \hat{\mathbf{x}}_i|}{|\mathbf{x}^*_i \vee \hat{\mathbf{x}}_i|}\,,$$

where again N is the test dataset size, $\mathbf{x}^*$ is the MAP inference and $\hat{\mathbf{x}}$ the true labels. The operations $\wedge$ (resp. $\vee$)

1. The datasets were also used in [13, 35] and are available at https://github.com/nicoladimauro/dcsn and http://meka.sourceforge.net.

Table 2: Results for completion tasks, *%I* column shows instances percentage, ΔEM and ΔHS is the metric difference between *Robust* and $\neg$*Robust* subsets.

Dataset		Exact Match					Hamming Score				
		Robust	*%I*	$\neg$*Robust*	*%I*	ΔEM	*Robust*	*%I*	$\neg$*Robust*	*%I*	ΔHS
Jester	$\varepsilon*$	**0.259**	0.66	0.001	99.34	**0.258**	0.66	**0.913**	99.34	0.692	**0.221**
	$s*$	0.076	2.26	.001	97.74	0.075	0.755	2.26	0.692	97.74	0.063
	$p*$	0.016	13.34	0.001	86.66	0.015	0.858	3.86	0.687	96.14	0.171
NLTCS	$\varepsilon*$	0.736	25.06	0.248	74.94	0.488	**0.934**	25.06	0.77	74.94	**0.164**
	$s*$	0.736	25.06	0.2487	74.94	0.488	**0.934**	25.06	0.77	74.94	**0.164**
	$p*$	0.736	25.12	0.248	74.88	0.488	0.933	25.12	0.77	74.88	0.163
MSNBC	$\varepsilon*$	**1**	0.01	0.247	99.99	**0.753**	**1**	0.01	0.783	99.99	**0.217**
	$s*$	0.464	12.69	0.215	87.31	0.249	0.875	12.69	0.769	87.31	0.106
	$p*$	0.727	0.02	0.247	99.98	0.48	0.92	0.02	0.783	99.98	0.137

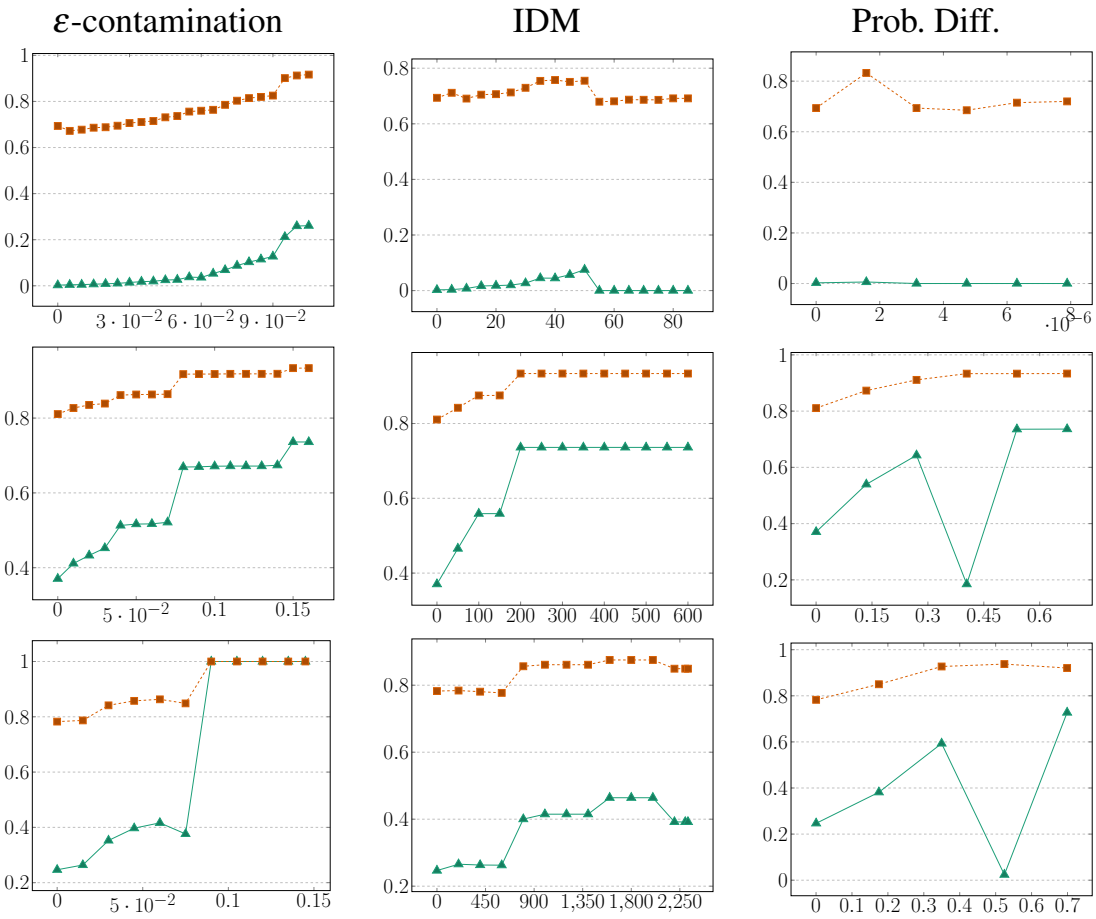

Figure 3: Test-set performance vs. threshold for completion tasks. Each row displays results for a dataset. From the top to bottom row: Jester, NLTCS, MSNBC. The curves with green triangle marks shown Exact Match, while the curves in orange square marks denote Hamming scores.

Table 3: Multi-Label Datasets, their domains, size of their feature vector (M), number of instances (N) , number of labels (L), label cardinality (LC) and label density (LD).

Dataset	Domain	M	N	L	LC	LD
Arts-Yahoo	Text	500	7484	26	1.65	0.06
Business-Yahoo	Text	500	11214	30	1.6	0.05
CAL500	Music	68	502	174	26.04	0.15
Emotions	Music	72	593	6	1.87	0.31
Flags	Images	19	194	7	3.39	0.48
Health-Yahoo	Text	500	9205	32	1.64	0.05
Human	Biological	440	3106	14	1.19	0.08
Plant	Biological	440	978	12	1.08	0.09
Scene	Images	294	2407	6	1.07	0.18
Yeast	Biological	103	2417	14	4.24	0.3

denote the vector obtained by coordinate-wise minimum (resp. maximum) of the instances. We do not use Hamming Score, as it does not distinguish between correct prediction of presence and absence of labels, and can lead to overoptimistic measures when the number of labels is high and label density is low.

Similar to the previous task, we select the best values $\varepsilon*$, $s*$ and $p*$ using the validation dataset, an then perform a robustness analysis for each multilabel prediction in the test dataset. The results are summarized in Table 4. All instances in the CAL500 dataset contained more that one MAP instantiation and are therefore deemed non-robust by all methods. We omit these results from the table.

One sees from these results that the CSPNs obtained by IDM achieve the highest accuracy in 5 of the 9 domains, followed by the CSPNs obtained with ε-contamination (3 out of 9). Probability difference outperforms the other methods with respect to accuracy only for Human, where the CSPN-based methods consider all instances robust. Yet, the difference in accuracy (or exact match scores) for robust and non-robust is very small, showing that the two sets perform indeed very similar (but perhaps it would be more sensible to classify all instances as non-robust). The IDM-based CSPNs perform particularly poorly in the Arts domain, where the accuracy of the robust portion is significantly inferior to the accuracy in the non-robust portion. We conjecture that this is due to some peculiarity of the learned network, but were not able to provide any deep explanation at this point.

Regarding Exact Match, we see that IDM-based CSPNS outperform other methods in 4/9 of the domains, while probability difference achieves the best performance in 3/9 of domains. CSPNs based on ε-contamination are outperformed in all but one domain (Arts), where IDM-based results exhibit the same strange behavior (robust instances obtain worse results than non-robust instances). One possible explanation for the best performance of ε-contamination is that it is more conservative, assigning more instances to the non-robust portion; this is particularly beneficial under exact match, which is more strict with mistakes.

7. Conclusion

Sum-Product Networks (SPNs) are deep probabilistic models that have shown very promising results in several tasks. Selective SPNs are a subclass of models that allows tractable MAP inference with only a small decrease in accuracy. In this work, we presented polynomial-time algorithm for robust analysis of MAP inference in tree-shaped selective SPNs. We evaluated our algorithms in two different tasks: completing missing values and performing multilabel classification. Our results show that the proposed algorithm is better at discrimination of robust and non-robust inferences than the standard approach based on the difference of probabilities. We left for the future the analysis of the complexity of robustness for multiply-connected SPNs, as well as a more extensive empirical evaluation.

Acknowledgments

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. 303920/2016-5 and 420669/2016-7.

References

[1] Mohamed R Amer and Sinisa Todorovic. Sum product networks for activity recognition. *IEEE transactions on pattern analysis and machine intelligence*, pages 800–813, 2016.

[2] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 1985.

[3] Hans Bodlaender, Frank van den Eijkhof, and Linda van der Gaag. On the complexity of the mpa problem in probabilistic networks. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 675–679, 2002.

[4] Enrique Castillo, José Manuel Gutiérrez, and Ali S Hadi. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, pages 412–423, 1997.

Table 4: Results for Multilabel classification, *%I* column shows instances percentage, ΔEM and ΔHS is the metric difference between *Robust* and $\neg$*Robust* subsets

Dataset		Accuracy					Exact Match				
		Robust	*%I*	$\neg$*Robust*	*%I*	ΔAcc	*Robust*	*%I*	$\neg$*Robust*	*%I*	ΔEM
Arts	$\varepsilon*$	**0.88**	1.2	0.196	98.8	0.634	**0.833**	1.2	0.143	98.8	**0.69**
	$s*$	0.107	51.94	0.351	48.06	-0.244	0.089	25.8	0.247	74.2	-0.158
	$p*$	0.81	6.95	0.159	93.05	**0.651**	0.75	6.95	0.107	93.05	0.643
Business	$\varepsilon*$	0.751	72.73	0.582	27.27	0.169	0.617	65.24	0.598	34.76	0.019
	$s*$	**0.781**	45.32	0.641	54.68	0.14	**0.642**	45.32	0.469	54.68	0.173
	$p*$	0.762	68.4	0.581	31.6	**0.181**	0.62	68.4	0.392	31.6	**0.228**
Emotions	$\varepsilon*$	0.595	17.65	0.41	82.35	0.185	0.238	17.65	0.163	82.35	0.075
	$s*$	**0.686**	10.92	0.413	89.08	**0.273**	**0.308**	10.92	0.16	89.08	**0.148**
	$p*$	0.574	28.57	0.391	71.43	0.183	0.176	28.57	0.176	71.43	0
Flags	$\varepsilon*$	0.917	10.53	0.468	89.47	0.449	0.5	57.89	0.118	42.11	0.382
	$s*$	0.917	10.53	0.468	89.47	0.449	0.5	47.37	0.1	52.63	**0.4**
	$p*$	0.917	10.53	0.468	89.47	0.449	0.5	10.53	0.118	89.47	0.382
Health	$\varepsilon*$	**0.667**	0.11	0.557	99.89	0.11	0.5	0.11	0.416	99.89	0.084
	$s*$	0.637	47.88	0.482	52.12	**0.155**	0.537	47.88	0.304	52.12	**0.233**
	$p*$	0.655	4.72	0.552	95.28	0.103	**0.552**	4.72	0.409	95.28	0.143
Human	$\varepsilon*$	0.203	100	-	0	-	0.146	100	-	0	-
	$s*$	0.203	100	-	0	-	0.146	100	-	0	-
	$p*$	**0.211**	42.6	0.198	57.4	0.013	**0.155**	42.6	0.14	57.4	0.015
Plant	$\varepsilon*$	0.331	37.76	0.217	62.24	0.114	0.324	37.76	0.213	62.24	0.111
	$s*$	**0.367**	47.96	0.162	52.04	**0.205**	**0.362**	47.96	0.157	52.04	**0.205**
	$p*$	0.345	36.22	0.212	63.76	0.132	0.338	36.22	0.208	63.76	0.13
Scene	$\varepsilon*$	0.857	5.83	0.277	94.17	0.58	0.857	5.83	0.212	94.17	0.645
	$s*$	**0.929**	2.92	0.293	0.636	97.08	**0.929**	2.92	0.293	97.08	0.636
	$p*$	0.923	5.42	0.276	94.58	**0.647**	0.923	5.42	0.211	94.58	**0.712**
Yeast	$\varepsilon*$	**0.436**	5.79	0.419	94.21	0.017	0.071	5.79	0.098	94.21	-0.027
	$s*$	**0.436**	5.79	0.419	94.21	0.017	0.071	5.79	0.098	94.21	-0.027
	$p*$	0.425	97.93	0.203	2.07	**0.222**	**0.099**	97.93	0	2.07	**0.099**

[5] Hei Chan and Adnan Darwiche. When do numbers really matter? *Journal of artificial intelligence research*, pages 265–287, 2002.

[6] Hei Chan and Adnan Darwiche. Sensitivity analysis in bayesian networks: From single to multiple parameters. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 67–75, 2004.

[7] Hei Chan and Adnan Darwiche. On the robustness of most probable explanations. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 63–71, 2006.

[8] Wei-Chen Cheng, Stanley Kok, Hoai Vu Pham, Hai Leong Chieu, and Kian Ming A Chai. Language modeling with sum-product networks. In *15th Annual Conference of the International Speech Communication Association*, 2014.

[9] Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In *Proceedings of the 34th International Conference on Machine Learning*, pages 825–833, 2017.

[10] Diarmaid Conaty, Denis Deratani Mauá, and Cassio Polpo de Campos. Approximation complexity of maximum a posteriori in sum-product networks. In *Proceedings of the 33rd conference on Uncertainty in artificial intelligence*, pages 322–331, 2017.

[11] Jesse Davis and Pedro Domingos. Bottom-up learning of markov network structure. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[12] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *International Journal of Machine Learning*, 88(1-2): 5–45, 2012.

[13] Nicola Di Mauro, Antonio Vergari, and Floriana Esposito. Multi-label classification with cutset networks. In *Proceedings of the 8th International Conference on Probabilistic Graphical Models*, pages 147–158, 2016.

[14] Robert Gens and Pedro M Domingos. Learning the structure of sum-product networks. In *Proceedings of the International Conference on Machine Learning*, pages 873–880, 2013.

[15] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. To trust or not to trust a classifier. In *Advances in Neural Information Processing Systems*, 2018.

[16] Uffe Kjærulff and Linda C van der Gaag. Making sensitivity analysis computationally efficient. In *Proceedings of the 16th conference on Uncertainty in artificial intelligence*, pages 317–325, 2000.

[17] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[18] Kathryn Blackmond Laskey. Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 901–909, 1995.

[19] Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[20] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Proceedings of the 32nd International Conference on Neural Information Processing System*, 2018.

[21] Denis Deratani Mauá, Diarmaid Conaty, Fabio Gagliardi Cozman, Katja Poppenhaeger, and Cassio Polpo de Campos. Robustifying sum-product networks. *International Journal of Approximate Reasoning*, pages 163–180, 2018.

[22] Jun Mei, Yong Jiang, and Kewei Tu. Maximum a posteriori inference in sum-product networks. In *The 32nd AAAI Conference on Artificial Intelligence*, pages 1923–1930, 2017.

[23] Robert Peharz, Robert Gens, and Pedro Domingos. Learning selective sum-product networks. In *Workshop on Learning Tractable Probabilistic Models*, 2014.

[24] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *Journal of Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2016.

[25] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, pages 2030–2044, 2017.

[26] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Probabilistic deep learning using random sum-product networks. *CoRR*, abs/1806.01910, 2018.

[27] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 337–346, 2011.

[28] Andrzej Pronobis and Rajesh PN Rao. Learning deep generative spatial models for mobile robots. In *EEE/RSJ International Conference on Intelligent Robots and Systems*, pages 755–762, 2017.

[29] Andrzej Pronobis, Francesco Riccio, and Rajesh PN Rao. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments. In *ICAPS 2017 Workshop on Planning and Robotics*, 2017.

[30] Silja Renooij and Linda C Van Der Gaag. Evidence and scenario sensitivities in naive bayesian classifiers. *International Journal of Approximate Reasoning*, pages 398–416, 2008.

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

[32] Murat Sensoy, Melih Kandemir, and Lance Kaplan. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, 2018.

[33] Bruno Massoni Sguerra and Fabio Gagliardi Cozman. Image classification using sum-product networks for autonomous flight of micro aerial vehicles. In *Proceedings of the 5th Brazilian Conference on Intelligent Systems*, pages 139–144, 2016.

[34] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358, 2015.

[35] Julissa Villanueva and Denis Deratani Mauá. On using sum-products networks for multi-label classification. In *Proceedings of the 6th Brazilian Conference on Intelligent Systems*, pages 25–30, 2017.

[36] Peter Walley. Inferences from multinomial data: learning about a bag of marbles. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1): 3–34, 1996.

[37] Kaiyu Zheng, Andrzej Pronobis, and Rajesh PN Rao. Learning graph-structured sum-product networks for probabilistic semantic maps. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.