Qualitative Global Sensitivity Analysis for Probabilistic Circuits

Julissa Giuliana Villanueva Llerena

Thesis presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
in partial fulfillment
of the requirements
for the degree of
Doctor of Science

Program: Ciência da Computação
Advisor: Prof. Dr. Denis Deratani Mauá

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

São Paulo
July, 2023

Qualitative Global Sensitivity Analysis for Probabilistic Circuits

Julissa Giuliana Villanueva Llerena

This version of the thesis includes the corrections and modifications suggested by the Examining Committee during the defense of the original version of the work, which took place on July 20, 2023.

A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

Examining Committee:

Prof. Dr. Denis Deratani Mauá (advisor) – IME-USP
Prof. Dr. Alessandro Antonucci – IDSIA-USI
Prof. Dr. Casio Polpo de Campos – TU Eindhoven
Prof. Dr. Ricardo Cerri – UFSCar
Prof. Dr. Antonio Vergari – U Edinburgh

Ficha catalográfica elaborada com dados inseridos pelo(a) autor(a)
Biblioteca Carlos Benjamin de Lyra
Instituto de Matemática e Estatística
Universidade de São Paulo

Villanueva Llerena, Julissa
Análise de Sensibilidade Global e Qualitativa para Circuitos Probabilísticos / Julissa Villanueva Llerena; orientador, Denis Deratani Mauá. - São Paulo, 2023.
95 p.: il.

Tese (Doutorado) - Programa de Pós-Graduação em Ciência da Computação / Instituto de Matemática e Estatística / Universidade de São Paulo.
Bibliografia
Versão corrigida

1. Circuitos Probabilisticos. 2. Circuitos Probabilisticos Credais. 3. Análise de Sensibilidade. 4. Dados Faltantes não Ignoráveis. I. Deratani Mauá, Denis. II. Título.

Bibliotecárias do Serviço de Informação e Biblioteca
Carlos Benjamin de Lyra do IME-USP, responsáveis pela
estrutura de catalogação da publicação de acordo com a AACR2:
Maria Lúcia Ribeiro CRB-8/2766; Stela do Nascimento Madruga CRB 8/7534.

You can do anything you want to, but you have to work at it.
— Anni Easley

Acknowledgments

First, I would like to express my deep gratitude to my advisor Denis for his guidance throughout this Ph.D. journey. His expertise and encouragement have been instrumental in shaping the trajectory of this research. I would also like to express my gratitude to the evaluation committee for their expertise, and valuable feedback on my dissertation. I would especially like to thank Prof. Alessandro for the helpful discussions and collaboration to improve my work.

I would like to express my sincere gratitude to my family, Ana, Daniel, Wil, Meni, Esme and Anto; without their unconditional support and trust in my abilities, this achievement would not have been possible. To my love, Marcos, your constant motivation, advice and understanding have been a source of inspiration to me. Thank you for always encouraging me and celebrating every milestone along the way.

I thank Viviane, Felipe, and all the members of LIAMF, where I had the privilege of conducting my research. Their contributions, both intellectual and personal, have been of great help throughout these years.

To all those whose names may not appear here, but who have contributed in some way to my growth and development, I thank you from the bottom of my heart.

Finally, and most importantly,I would like to thank Almighty God for giving me the guidance and opportunity to embark on this adventure.

Resumo

Julissa Giuliana Villanueva Llerena. **Análise de Sensibilidade Global e Qualitativa para Circuitos Probabilísticos**. Tese (Doutorado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Um Circuito Probabilístico (CP) é um modelo generativo expressivo que codifica uma distribuição de probabilidade através de uma estrutura de somas ponderadas, produtos e distribuições univariadas ou multivariadas. Sujeitos a algumas restrições, os CPs são tratáveis para varias classes de consultas. Os exemplos mais populares de CPs são Redes de Soma-Produto, Diagramas de Decisão Sentenciais Probabilísticos e Florestas Aleatórias Gerativas. Esses modelos têm mostrado desempenho competitivo em diversas tarefas de aprendizado de máquina. Apesar do relativo sucesso dos CPs, vários problemas podem afetar a qualidade de suas previsões. Neste trabalho, nos concentramos em duas questões relevantes. (i) CPs com um alto número de parâmetros e dados escassos podem produzir inferências não confiáveis e com excesso de confiança. (ii) Abordagens típicas tratam dados faltantes por marginalização ou heuristicamente, assumindo que o processo de falta é ignorável ou não informativo; no entanto, os dados geralmente estão ausentes de maneira não ignorável, o que introduz viés na previsão se não for tratado adequadamente. Para resolver essas questões, desenvolvemos dois algoritmos baseados em Circuitos Probabilísticos Credais, que são conjuntos de CPs obtidos pela perturbação simultânea de todos os parâmetros do modelo (com a estrutura do modelo fixa). Nosso primeiro algoritmo realiza uma análise de sensibilidade global qualitativa nos parâmetros do modelo, medindo a variabilidade das previsões para perturbações dos pesos do modelo. Para mitigar o segundo problema, propomos um procedimento para realizar inferência preditiva tratável sob dados ausentes não ignoráveis. Avaliamos nossos algoritmos em tarefas desafiadoras, como compleção de imagem, classificação multirótulo e classificação multiclasse.

Palavras-chave: Circuitos Probabilisticos. Circuitos Probabilisticos Credais. Análise de Sensibilidade. Dados Faltantes não Ignoráveis.

Abstract

Julissa Giuliana Villanueva Llerena. **Qualitative Global Sensitivity Analysis for Probabilistic Circuits**. Thesis (Doctorate). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

A Probabilistic Circuit (PC) is an expressive generative model that encodes a probability distribution through an structure of weighted sums, products and univariate or multivariate distributions. Subject to some restrictions, PCs are tractable for large classes of queries. The most popular examples of PCs are Sum-Product Networks, Probabilistic Sentential Decision Diagrams, and Generative Random Forests. These models have shown competitive performance in several machine learning tasks. Despite the relative success of PCs, several issues can affect the quality of their predictions. In this work, we focus on two relevant issues. (i) PCs with a high number of parameters and scarce data can produce unreliable and overconfident inferences. (ii) Typical approaches treat missing data either by marginalization or heuristically, assuming that the missingness process is ignorable or uninformative; however, data is often missing in a non-ignorable way, which introduces bias into the prediction if not handled properly. To address these issues, we developed two algorithms based on Credal Probabilistic Circuits, which are sets of PCs obtained by a simultaneously perturbing of all model parameters (with the model structure fixed). Our first algorithm performs a qualitative global sensitivity analysis on the model parameters, measuring the variability of the predictions to perturbations of the model weights. To mitigate the second issue, we propose a procedure to perform tractable predictive inference under non-ignorable missing data. We evaluate our algorithms on challenging tasks such as image completion, multi-label classification, and multi-class classification.

Keywords: Probabilistic Circuits. Credal Probabilistic Circuits. Sensitivity Analysis . Non-ignorable Missing Data.

Lista de abreviaturas

PC	Probabilistic Circuit
SPN	Sum-Product Network
GeF	Generative random Forest
PSDD	Probabilistic Sentential Decision Diagrams
MAP	Maximum-A-Posteriori
SSPN	Selective Sum-Product Network
DT	Decision Tree
MLE	Maximum likelihood Estimation
CPC	Credal Probabilistic Circuit
CSPN	Credal Sum-Product Network
SA	Sensitivity Analysis
CSSPN	Credal Selective Sum-Product Network
Acc	Accuracy
EM	Exact Match
HS	Hamming Score
SAcc	Set Accuracy
DAcc	Disconted Accuracy
MAR	Missing At Random
MCAR	Missing Completly At Random
MNAR	Missing Not-At Random
CIR	Conservative Inference Rule
TCI	Tractable Conservative Inference
CReM	Credal REsponse Model
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

List of Figures

List of Tables

Contents

Chapter 1

Introduction

By modeling a joint probability distribution, one can perform a variety of probability inferences such as computing marginal probabilities and finding maximum-a-posteriori (MAP) configurations (Koller and Friedman, 2009). Those inferences are particularly useful for classification tasks. For example, marginal inference can be used to perform multiclass classification by assigning objects to the most probable class label. MAP configurations can be used instead to associate objects to several relevant labels, performing multilabel classification.

Traditional probabilistic graphical models such as Bayesian Networks and Markov networks allow for the compact specification of high dimensional joint probability distributions (Koller and Friedman, 2009). Despite their intuitive graphical semantics, computing typical inferences with such models is NP-hard. Probabilistic Circuits (PC) are a more recently proposed class of graph-based probabilistic models (Poon and Domingos, 2011). Unlike traditional probabilistic graphical models, PCs allow several tractable inferences depending on which structural constraints are imposed. Sum-Product Networks (SPNs) (Poon and Domingos, 2011), Probabilistic Sentential Decision Diagrams (PSDDs) (Kisa *et al.*, 2014) and Generative Random Forests (GeFs) (Correia *et al.*, 2020) are notable examples of PCs.

PCs have obtained notable results in several supervised learning tasks, due to their ability to compactly represent multidimensional distributions and efficiently produce reliable inference (Poon and Domingos, 2011; Cheng *et al.*, 2014; Amer and Todorovic, 2016; Villanueva Llerena and Mauá, 2017; Shen *et al.*, 2017; Zheng *et al.*, 2018; Pronobis and Rao, 2017; Shao *et al.*, 2020; Peharz, Vergari, *et al.*, 2020).

Despite their relative success in supervised learning tasks, like many other machine learning models, PCs generalize poorly in regions of insufficient statistical support and/or in the presence of missing data, resulting in overconfident, unreliable predictions. Overconfident and incorrect predictions can be harmful, especially for critical applications such as autonomous vehicles or healthcare (Lambrou *et al.*, 2010).

These concerns can be mitigated by performing a *Sensitivity Analysis* (SA) of the model predictions to small changes in the parameters (Berger, 1985). SA can be performed locally or globally. The former considers the effect of perturbing a single parameter or a small

group of related parameters (Chan and Darwiche, 2002; Laskey, 1995). The latter is concerned with more general perturbations that may affect all parameters of the model simultaneously (Castillo *et al.*, 1997; Chan and Darwiche, 2004; Kjærulff and Gaag, 2000).

SA can also be performed in a quantitative or qualitative way. Quantitative SA measures the effect of perturbing the model parameters on the value of a particular prediction. A typical example is assessing the change in marginal posterior probabilities of a single variable when parameters are varied (Jiang *et al.*, 2018; Sensoy *et al.*, 2018; Malinin and Gales, 2018; Castillo *et al.*, 1997; Chan and Darwiche, 2004; Laskey, 1995; Kjærulff and Gaag, 2000; Chan and Darwiche, 2002). Unfortunately, computing the posterior for PCs is intractable (Rashwan *et al.*, 2016). Although, there are several algorithms to approximate Bayesian inference for SPNs (Rashwan *et al.*, 2016; Zhao *et al.*, 2016; Vergari, Molina, *et al.*, 2019), these techniques are very inefficient compared to the computational cost of a marginal inference in SPNs. In contrast, qualitative SA assesses whether decisions induced by inferences change as parameters are perturbed (Chan and Darwiche, 2006; Renooij and Van Der Gaag, 2008).

Credal Sum-Product Networks (CSPNs) are sets of SPNs obtained by simultaneous perturbation of model parameters (with the network structure fixed) (Mauá, Conaty, *et al.*, 2018). Thus, CSPNs are naturally used to perform SA in SPNs. Mauá et al. (2018) developed efficient algorithms to obtain upper and lower bounds of marginal inferences for a given CSPN. Those algorithms were then used to perform quantitative global SA in SPNs in classification tasks.

In this research, we develop new algorithms for qualitative global sensitivity analysis for PCs based on CSPNs. To meet the needs of many applications, we focus on analyses that can be obtained in a time comparable to the computation of the corresponding prediction (often linear in the size of the model).

Our first contribution was motivated by the difficulty of most classification models to identify overconfident MAP inferences in regions with insufficient or conflicting data. We developed a qualitative global sensitivity analysis algorithm for MAP inference in selective SPNs, i.e., PCs that compute MAP in linear time in the number of parameters. In particular, we devised a quadratic runtime procedure to decide whether a given MAP configuration remains optimal when the model parameters are subjected to small changes (Villanueva Llerena and Mauá, 2020; Mattei *et al.*, 2020).Our proposed algorithms have been evaluated by their performance on real-world tasks such as data imputation and multi-label classification. We empirically compared our qualitative SA for MAP inference algorithm with a standard baseline in discriminating robust from non-robust instances. The results show that the learned SPNs are often robust to small global perturbations in the parameters, and that quantitative global SA is at least as good as the baseline approach in distinguishing non-robust classifications.

Another challenge for machine learning models is coping with the presence of missing values. The standard approach to dealing with such data is to either impute or marginalize out the missing values. That approach is theoretically supported by a *missing at random* (MAR) assumption, which roughly states that the probability of the missing values does not depend on the variables with missing values themselves (Azur *et al.*, 2011; Khosravi,

Y. Choi, *et al.*, 2019; Rubin, 1976). While MAR is popular, it is often violated in practical scenarios when the probability of not observing a variable depends on other variables including its own value, a situation called non-ignorable missing data or missing not at random (MNAR). In such cases marginalization can still be used as a heuristic, at the risk of introducing excessive bias (Manski, 2005).

As a second contribution, we developed exact quadratic-time methods for the conservative treatment of non-ignorable missing data in Decision-Tree classifiers (Villanueva Llerena, Mauá, and Antonucci, 2021). By making use of GeFs (Correia *et al.*, 2020), which extend Decision-Trees to PCs, we are able to model the uncertainty introduced by the presence of MNAR data at prediction time and avoid making an unjustified MAR hypothesis. We develop two variants that differ in the assumptions about the response model and the level of informativeness of the analyses. A response model is a representation of the conditional probabilities of missing values, where the presence/absence of each variable is associated with a binary variable called the response, and then encodes the conditional dependence of each response given the respective variable values (Rubin, 1976).

The first algorithm we develop makes no assumption about the response model. It thus produces overly cautious non-informative predictions (e.g., the set of all classes). To make inferences more informative (and less conservative), we also propose an alternative algorithm that assumes the availability of a partially specified response model in the form of probability intervals of the probability of observing/measuring a variable conditional on its value. Such intervals can often be obtained from expert domain knowledge, or derived from specially curated training data (e.g., a small sample survey of users of a recommendation system). Either variant can be used to perform a qualitative SA of classifications of a target variable w.r.t. to the (full, unknown) specification of the response model (Villanueva Llerena and Mauá, 2022). We compare both approaches with respect to the accuracy of classifications made by marginalizing unobserved variables according. The results show that either approach is able to distinguish between instances where assuming MAR is harmful from those where is less harmful; also, the two variants exhibit a trade-off between informativess and accuracy.

1.1 Publications

We presented our algorithm for robust analysis of MAP inference at the 20th International Symposium on Imprecise Probabilities: Theories and Applications (ISIPTA 2019) (Villanueva Llerena and Mauá, 2019); an extended version of that work apepeared in the International Journal of Approximate Reasoning (Villanueva Llerena and Mauá, 2020). The robustness algorithms for PSDDs were first independently developed and presented at ISIPTA 2019 by Antonucci et al. (Antonucci, Facchini, *et al.*, 2019). We have collaborated on an extended version of that work that contains a deeper empirical analysis and also algorithms for MAP analysis (missing from the previous work) (Mattei *et al.*, 2020).

The conservative algorithm to quantify the effect of different imputations of the missing values of features in the classification using GeFs of a target discrete variable at prediction time was presented at the 16th European Conference on Symbolic and

Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2021) (Villanueva Llerena, Mauá, and Antonucci, 2021). Our last contribution, the tractable algorithm to perform exact inference at prediction time in the presence of non-ignorable missing data represented using a response model with GeFs was presented in the 10th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe 2022) (Villanueva Llerena and Mauá, 2022).

1.2 Organization

The rest of the manuscript is organized as follows. In Chapter 2, we introduce some basic knowledge about PCs, we focus in SPNs, PSDDs and GeFs. In Chapter 3 we introduce concepts of credal Probabilistic Circuits and the algorithms to compute upper and lower probabilities, as well as credal classification. Chapter 4 presents our first contribution, an approach for qualitative sensitivity analysis for MAP inference using Credal PCs, we also describe the experiments and show the experimental results of this proposed procedure. In chapter 5 we introduce basic concepts about missing data and present our two algorithms for conservative prediction in the presence of non-ignorable missing data in GeFs, with the corresponding experimental results and a comparison between these two approaches. Finally, we conclude the work and discuss possible improvements in Chapter 6.

Chapter 2

Probabilistic Circuits

Inspired by the need to better handle and study the complexity of inference in classical probabilistic graphical models (Darwiche, 2003; Chavira and Darwiche, 2005; Huang *et al.*, 2006; Lowd and Domingos, 2008), a rooted directed-acyclic graph architecture called Sum-Product Networks (SPNs) emerged at the expense of loss of interpretability (Poon and Domingos, 2011). Poon and Domingos noticed that SPNs and arithmetic circuits can be translated into each other, and thus have the same expressiveness (Poon and Domingos, 2011). Initial experiments with images suggested the capability of SPNs for machine learning tasks, which have been shown to match and often exceed the performance of probabilistic graphical models (e.g. Bayesian networks) (Poon and Domingos, 2011; Amer and Todorovic, 2016; Zheng *et al.*, 2018; Pronobis, Riccio, *et al.*, 2017; Pronobis and Rao, 2017; Cheng *et al.*, 2014).

SPNs have been a relevant starting point for several tractable models, that have improved the expressiveness and efficiency of some types of inference, that more recently called Probabilistic Circuits. Sum-Product Networks (SPNs) (Poon and Domingos, 2011), Probabilistic Sentential Decision Diagrams (PSDDs) (Kisa *et al.*, 2014), and Generative Random Forests (GeFs) (Correia *et al.*, 2020) are all special cases of Probabilistic Circuits. This chapter presents some relevant definitions and properties of Probabilistic Circuits: SPNs, PSDDs and GeFs; as well as inference and learning algorithms.

Notation

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 realizations of a collection of random variables $\mathbf{X}$ is denoted as $val(\mathbf{X}) = \times_{i=1}^{N} val(X_i)$, where $\times$ denotes the Cartesian product.

In this work, we assume that random variables take on a finite number of values. We write $\mathbf{x}$ to denote an element of $val(\mathbf{X})$, therefore we can associate to every random variable X_i a set of indicator functions $\{[\![X_i = x_i]\!] : x_i \in val(X_i)\}$, where the notation

$[\![X_i = x_i]\!]$ describes the function that returns 1 if X_i takes value x_i and 0 otherwise.

2.1 Probabilistic Circuits

A probabilistic circuit (PC) is a probabilistic model encoding a possibly unnormalized probability distribution. A PC $\mathtt{S}$ over random variables $\mathbf{X}$ is a rooted weighted acyclic directed graph whose internal nodes are associated to either sum or product operations, that we graphically represent by + and × respectively, and leaves associated to tractable distributions over $\mathbf{X}$. In this disertation, we consider discrete random variables using indicator functions as leaves, where each indicator appears in at least one leaf, for example leaf node $[\![X_i = x_i]\!]$ is associated with indicator function of variable X_i and value x_i.

The weights associated with arcs emanating from a sum node i are represented by $\mathbf{w}_i$ and the multiset of all weights of the network is denoted as $\mathbf{w}$, each arcs $i \longrightarrow j$ of the cirucit is associated with weights w_{ij}, such that arcs leaving product nodes are assigned weight one. When we want make explicit the dependence of a circuit $\mathtt{S}$ on the weights $\mathbf{w}$, we write $\mathtt{S}_{\mathbf{w}}$.

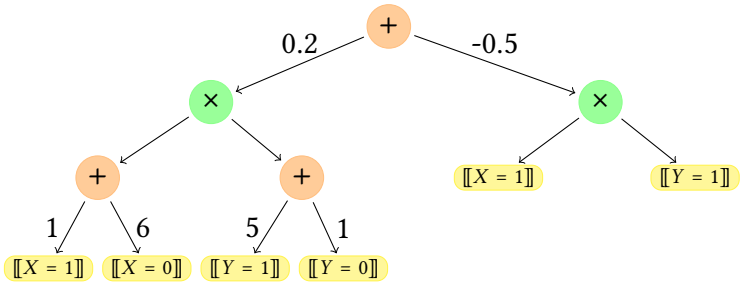


Figure 2.1: *Probabilistic Circuit over two binary variables.*

The *scope* of a PC is the set of random variables that appear at its indicator nodes (leaf nodes), is denoted as $sc(\mathtt{S})$. Figure 2.1 shows a PC whose scope is $\{X, Y\}$.

If i is a node in $\mathtt{S}$ we write $\mathtt{S}^i$ to denote the sub-PC rooted at i. The children of a node i (the nodes to which there is an arc from i) are denoted by $\mathrm{ch}(i)$.

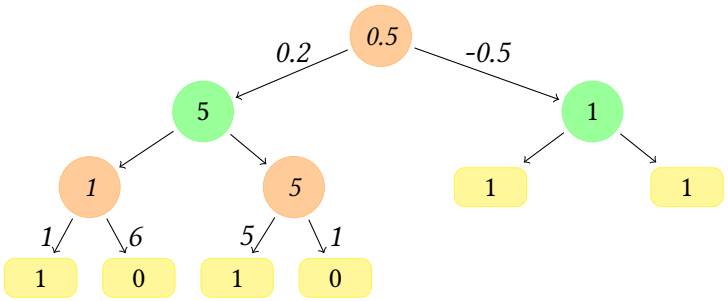
The *evaluation* of an PC, $\mathtt{S}$, at a complete instantiation of its scope, $\mathbf{x}$, produces a real-value, written $\mathtt{S}(\mathbf{x})$, defined inductively at each node i as:

$$\mathtt{S}^i(\mathbf{x}) = \begin{cases} 1, & \text{if } i \text{ is a leaf node consistent with } \mathbf{x}, \\ 0, & \text{if } i \text{ is a leaf node inconsistent with } \mathbf{x}, \\ \prod_{j \in \mathrm{ch}(i)} \mathtt{S}^j(\mathbf{x}), & \text{if } i \text{ is a product node}, \\ \sum_{j \in \mathrm{ch}(i)} w_{ij} \mathtt{S}^j(\mathbf{x}), & \text{if } i \text{ is a sum node}. \end{cases} \tag{2.1}$$

In Equation 2.1, we say that a leaf node i is consistent with a instantiation $\mathbf{x}$ if $\mathtt{S}^i$ is the indicator function $[\![X_j = k]\!]$ and $x_j = k$ in $\mathbf{x}$, or if X_j is not in the scope of $\mathtt{S}^i$, otherwise the node is said to be inconsistent. The evaluation of $\mathbf{x}$ is given by the value computed at the

circuit root, $\mathrm{S}(\mathbf{x}) = \mathrm{S}^r(\mathbf{x})$, where r is the circuit root, which can be computed in linear time by traversing the PC from the leaves towards the root, caching values if needed to avoid redundant computation.

Example 2.1.1. *Consider the PC in Figure 2.1, we want to evaluate this PC at the complete instantiation* $\mathbf{x} = \{X = 1, Y = 1\}$. *The following figure shows node values at the evaluation of* $\mathbf{x}$ *applying the procedure defined in Equation 2.1, the final result corresponds to the root value that is 0.5.*



2.2 Sum-Product Networks

A Sum-Product network is a PC whose weights are nonnegative and where the set of all nodes in the network satisfies the following properties to ensure that the network allows to compute marginal inference in linear time (we describe marginal inference for SPNs latter in the chapter):

Completeness: Any two children of a sum node have identical scope;

Decomposition: Any two children of a product node have disjoint scopes;

Additionally, we can say that a SPN is **normalized** if the sum of the weights of arcs leaving each sum node is one.

Sum-Product Trees are SPNs where each internal node has at most one parent. This property was defined in SPNs with categorical distributions as leaves (and not indicator variables) . If we consider SPNs with indicators as leaves, as we do in this work, then the indicators with multiple parents can be duplicated without changing the distribution represented. Note that in this type of structure, we can equate sum-product trees with tree-shaped SPNs. For example, Figure 2.2 shows a Sum-Product Tree with scope $\{X, Y\}$ that is complete, decomposable, and normalized.

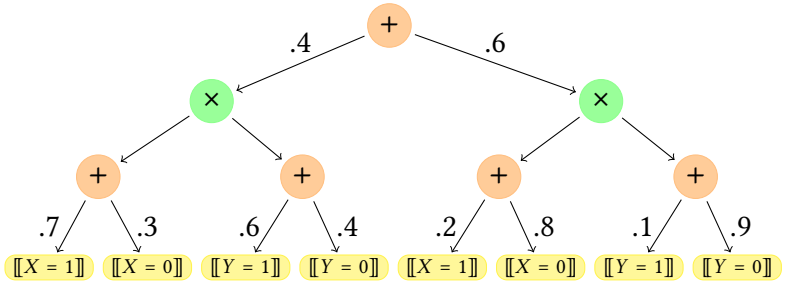


Figure 2.2: *Sum-Product Network over two binary variables.*

2.2.1 Probability of Evidence

Since, a decomposable and complete SPN defines a probability distribution $\mathbb{P}_{\mathcal{S}}(\mathbf{x})$, it is possible to perform probabilistic queries, such as the probability of the evidence, also known as marginal inference, which computes the probability of a partial observation.

Consider a partition into two subsets of $sc(\mathcal{S})$: $\mathbf{X}^{\text{obs}}, \mathbf{X}^{\text{miss}} \in \mathbf{X}$ where $\mathbf{X}^{\text{obs}} \cap \mathbf{X}^{\text{miss}} = \emptyset$, $\mathbf{x}^{\text{obs}} \in val(\mathbf{X}^{\text{obs}})$ is a partial instantiation of the scope of $\mathcal{S}$ called *evidence* and $\mathbf{x}^{\text{miss}} \in val(\mathbf{X}^{\text{miss}})$ represent an instantiation of the unobserved portion of variables. The respective marginal inference is:

$$\mathbb{P}_{\mathcal{S}}(\mathbf{x}^{\text{obs}}) = \sum_{\mathbf{x}^{\text{miss}} \in val(\mathbf{X}^{\text{miss}})} \mathbb{P}_{\mathcal{S}}(\mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) = \mathcal{S}(\mathbf{x}^{\text{obs}}). \tag{2.2}$$

We can calculate Equation 2.2 by extending the procedure in Equation 2.1 with the following case:

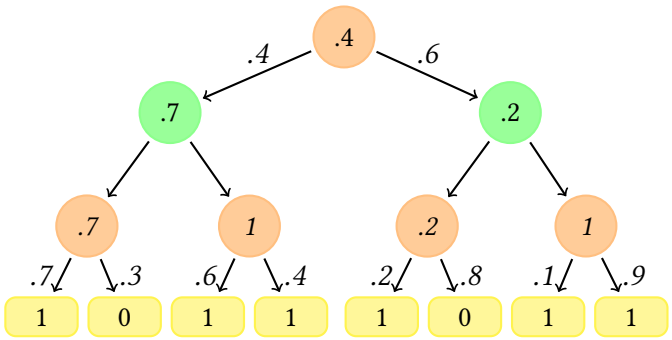
$$\mathcal{S}^i(\mathbf{x}^{\text{obs}}) = \begin{cases} 1, & \text{if } i \text{ is an indicator node whose scope } \mathbf{X}_i \text{ is in } \mathbf{X}^{\text{miss}}. \end{cases} \tag{2.3}$$

Given $\mathbf{X}^{\text{a}}, \mathbf{X}^{\text{b}} \in \mathbf{X}^{\text{obs}}$, conditional marginal probabilities $\mathbb{P}_{\mathcal{S}}(\mathbf{x}^{\text{a}}|\mathbf{x}^{\text{b}})$ can also be computed efficiently by computing numerator (i.e., $\mathbb{P}_{\mathcal{S}}(\mathbf{x}^{\text{a}})$) and denominator (i.e., $\mathbb{P}_{\mathcal{S}}(\mathbf{x}^{\text{b}})$) then outputting their ratio.

In (non-normalized) SPNs we can compute the normalization constant by setting $\mathcal{S}(\mathbf{x}) = 1$ for any leave, and then propagating values as Equation 2.1. Thus, linear time marginal inference is accomplished even without normalization, albeit with a constant increase in time complexity.

Completeness, Decomposability and tractable distributions as leaves guarantee the tractable computation of any marginal query (Y. Choi *et al.*, 2020). It is also possible to compute marginal queries in complete and non decomposable PCs by relaxing decomposability property (Peharz, Tschiatschek, *et al.*, 2015; Correia *et al.*, 2020), e.g. Generative Randon Forest, that we will review in section 2.3.3.

Example 2.2.1. *Consider the SPN in Figure 2.2, we want to evaluate this network at the evidence* $\mathbf{e} = \{X = 1\}$. *The following figure shows node values at the evaluation of* $\mathbf{x}^{obs}$ *applying the procedure defined in Equation 2.1, the final result corresponds to the root value that is 0.4.*



2.2.2 LearnSPN

LearnSPN is a popular algorithm for learning the structure and parameters of complete and decomposable Sum-Product Networks from data. It is a top-down, greedy algorithm that partitions the training data recursively (Gens and Domingos, 2013).

Algorithm 1 presents LearnSPN, where the variables I and C are parameters that limit the number of partitions in variable and instance sets, respectively, where N_k is the number of instances where variable X takes value k from the dataset, N_j is the total number of occurrences of variable X, and $|vals(X)|$ is the number of values that variable X can take.

Given a dataset $\mathbf{d}$ of i.i.d. instances $\mathbf{x} \in val(\mathbf{X})$, LearnSPN learns a tree-shaped SPN starting at the root and recursively partitioning the dataset to generate products and sum nodes. LearnSPN first attempts to partition the variables into independent subsets to learn a product node (see line 8 of Algorithm 1). A statistical test for independence can be performed on each pair of variables to generate approximately independent subsets, e.g. G-test . If this fails, then the instances are divided into similar subsets to learn a sum node, which can be achieved by a clustering algorithm. The weights associated with the arcs leaving a sum node represent the proportions of instances falling into the computed clusters (see line 12 of Algorithm 1). LearnSPN then proceeds with the recursive partitioning. The recursion stops when the processed data contains a single variable, then a sum node and the corresponding indicator nodes are introduced as children of the SPN and the corresponding weights are estimated from the data entries by counting the occurrences of their values while applying Laplace smoothing.

Vergari et al. (Vergari, Di Mauro, *et al.*, 2015) proposed some modifications to improve the structure of the SPNs obtained by LearnSPN algorithm. The first improvement consists in limiting the number of node children when performing the instance and variable splitting. In addition to an early stopping criterion when the number of instances falls below a given threshold, the data is modelled in a substructure as leaves, which in the simplest case can be a naive factorisation, introducing a product node with each variable distribution substructure as a child, i.e. a sum node with indicator nodes whose weights are estimated from the data. Another alternative is to model the data at the leaves as tractable models, such as Chow-Liu trees (Chow and Liu, 1968).

2.3 Selective Sum-Product Networks

Selective Sum-Product Networks (SSPNs) are a variant of Sum-Product Networks that have one more restriction than SPNs called selectivity (also known as determinism). This additional restriction allows linear computation of maximum a posteriori inference, and maximum-likelihood parameter learning (Peharz, Gens, and Domingos, 2014).

Before formally defining selectivity, we start by defining the support of a distribution and active nodes. The *support* of a distribution $\mathbb{P}$ is the set of instantiations $\mathbf{x}$ for which $\mathbb{P}(\mathbf{x}) > 0$. Given an instantiation $\mathbf{x}$, we say that a node i of an SPN $\mathbb{S}$ is *active* if $\mathbb{S}^i(\mathbf{x}) > 0$.

Algorithm 1: *LearnSPN*($\mathbf{d}$, m, α)

input : dataset: $\mathbf{d}$, minimum number of instances: m, laplace smooth parameter: α
output: An SPN $\mathtt{S}(\mathbf{X})$ learned from $\mathbf{d}$

$\mathbf{X} \leftarrow$ set variables of $\mathbf{d}$
$n \leftarrow$ number of instances in $\mathbf{d}$
if $|\mathbf{X}| = 1$ **then**
  $\mathtt{S} \leftarrow \sum_k \frac{N_k + \alpha}{N_j + \alpha|\text{vals}(X)|} [\![X = k]\!]$
else if $n < m$ **then**
  $\mathtt{S} \leftarrow tractableModel(\mathbf{d})$
else
  partition $\mathbf{X}$ into I approximately independent subsets of variables $\mathbf{X}_j$
  if *success* **then**
    $\mathtt{S} \leftarrow \prod_{j=1}^{I} LearnSPN(\mathbf{d}_j, m)$
  else
    partition $\mathbf{d}$ into C subsets of similar examples $\mathbf{d}_i$
    $\mathtt{S} \leftarrow \sum_{i=1}^{C} \frac{|\mathbf{d}_i|}{|\mathbf{d}|} LearnSPN(\mathbf{d}_i, m)$
  end
end
return $\mathtt{S}$

Selectivity requires that the supports of the distributions induced by the children of any sum node are disjoint. This implies for any instantiation $\mathbf{x}$ and sum node i that $\mathtt{S}^i(\mathbf{x}) = w_{ij}\mathtt{S}^j(\mathbf{x})$ for some $j \in \text{ch}(i)$. If the SPN is selective then at most one child of each sum node is active for any instantiarion of the scope of the network. For example, Figure 2.3 shows a selective SPN with scope $\{X, Y, Z\}$.

We say that a SPN is *structural selective* if the selectivity holds even if all the weights in the model are strictly positive, i.e., the structure of the SPN is selective regardless of the weights (Peharz, Gens, and Domingos, 2014). Since in this dissertation we consider the weights of an SPN to be greater than zero, selectivity and structural selectivity are equivalent. Figure 2.3 shows a structural selective SPN.

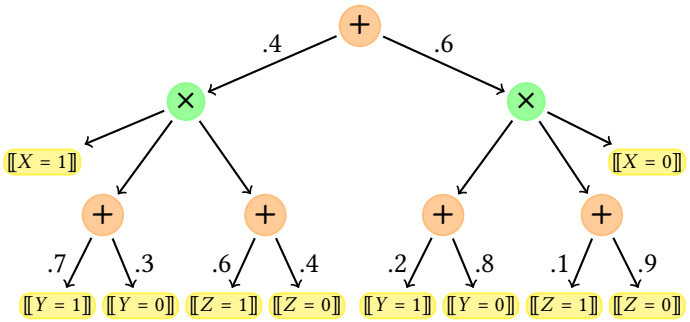


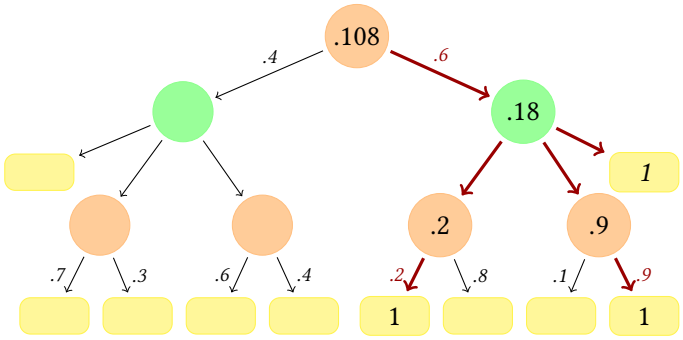
Figure 2.3: *Selective Sum-Product Network*

The *calculation tree* $\mathtt{T}_\mathtt{S}(\mathbf{x})$ of an selective SPN $\mathtt{S}$ is the subnetwork induced by the sequence of *active nodes* for the evaluation of instantiation $\mathbf{x}$ begining at the root of $\mathtt{S}$. Due

to the decomposition and selectivity properties, the calculation tree of a SSPN is a tree, hence justifying its name (Peharz, Gens, and Domingos, 2014, Lemma 1). We have from Equation 2.1 that:

$$\mathrm{T}_{\mathrm{S}}(\mathbf{x}) = \prod_{i \to j \in \mathrm{T}_{\mathrm{S}}(\mathbf{x})} w_{ij} \,. \tag{2.4}$$

Example 2.3.1. *Consider the Selective SPN in Figure 2.3, we want to obtain the Calculation Tree induced by the realization* $\mathbf{x} = \{X = 0, Y = 1, Z = 0\}$*, for that we use equation 2.1. In the figure below, the arcs associated with the calculation tree are highlighted in red.*



2.3.1 Maximum-A-Posteriori Inference

Recall that an SPN S induces a probability distribution $\mathbb{P}_{\mathrm{S}}$ over its scope. This allows the model to provide maximum-a-posteriori (MAP) inference , which consists of finding the most probable instantiation given some evidence, MAP has applications such as multi-label classification, which aims to find the most probable configuration of labels given the object features.

Given an SPN S with scope $\{\mathbf{X}^{\mathrm{obs}}, \mathbf{X}^{\mathrm{miss}}\} = \mathbf{X}$ and evidence $\mathbf{x}^{\mathrm{obs}} \in val(\mathbf{X}^{\mathrm{obs}})$, we define the set of MAP instantiations as:

$$\mathbf{x}^{*} \in \arg \max_{\mathbf{x}^{\mathrm{miss}} \in val(\mathbf{X}^{\mathrm{miss}})} \mathbb{P}_{\mathrm{S}}(\mathbf{x}^{\mathrm{miss}} | \mathbf{x}^{\mathrm{obs}}) = \arg \max_{\mathbf{x}^{\mathrm{miss}} \in val(\mathbf{X}^{\mathrm{miss}})} \mathrm{S}(\mathbf{x}^{\mathrm{miss}}, \mathbf{x}^{\mathrm{obs}}) \,. \tag{2.5}$$

Although MAP inference is NP-hard in SPNs even when restricted to binary variables and height-two networks (Conaty *et al.*, 2017; Mei *et al.*, 2017; Peharz, Gens, Pernkopf, *et al.*, 2017), MAP inference for selective SPNs is tractable in linear time in network size by the simple Max-Product algorithm (Peharz, Gens, and Domingos, 2014; Peharz, Gens, Pernkopf, *et al.*, 2017).

The Max-Product algorithm consists of replacing the sum operations in the calculation of the case of sum nodes in Equation 2.1 by maximizing on the values of the children,

caching values to achieve linear time. Then is described by the following recursion:

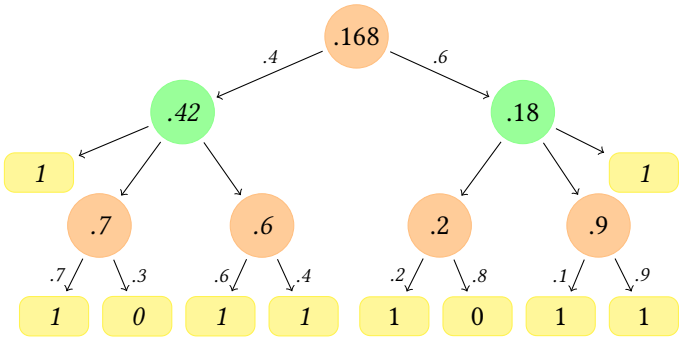
$$MAP^i(\mathbf{x}^{\text{obs}}) = \begin{cases} 1, & \text{if } i \text{ is a consistent leaf node,} \\ 0, & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_{j \in \text{ch}(i)} MAP^j(\mathbf{x}^{\text{obs}}), & \text{if } i \text{ is a product node,} \\ \max_{j \in \text{ch}(i)} w_{ij} MAP^j(\mathbf{x}^{\text{obs}}), & \text{if } i \text{ is a sum node.} \end{cases} \tag{2.6}$$

We say that a leaf node is consistent (w.r.t. a MAP inference) if its scope is in $\mathbf{X}^{\text{miss}}$, or if it is consistent by the definition used in 2.1.

The corresponding MAP instantiation is obtained by backtracking the solutions of the maximizations from the root toward the leaves.

Note that the Max Product algorithm maximizes over the children of the sum nodes, it actually maximizes over the SPN calculation trees; as we discussed in Section 2.3, for selective SPNs only one child is active, demonstrating the correctness of Max Product for selective SPNs.

Example 2.3.2. *Consider the SPN in Figure 2.3. We want to obtain MAP instantiation given evidence* $\mathbf{x}^{obs} = \{Y = 1\}$*, for that we use Equation 2.6.*



The result of Max-Product when backtracking from the root (i.e., it contains the arguments of the maximizations at sum nodes),is the MAP instantiation $\{X = 1, Z = 1\}$ *given the evidence* $Y = 1$*.*

2.3.2 Learning Selective Sum-Product Networks

One benefit of selectivity is that it allows us to learn the maximum-likelihood weights of a fixed-structure SSPN S given a dataset $\mathbf{d}$ of i.i.d. realizations $\mathbf{x} \in val(\mathbf{X})$ in closed-form as:

$$w_{ij} = \frac{N_j}{N_i}, \tag{2.7}$$

where $N_k = |\mathbf{x} \in \mathbf{d} : k \in \mathtt{T}_S(\mathbf{x})|$ is the number of calculation trees containing node k. In order to obtain more robust estimators, it is possible to smooth out the counts above, for example, by using the Multinomial-Dirichlet estimator $w_{ij} = (N_j + s_j)/(N_i + s)$ for $s = \sum_j s_j$.

A common choice is Laplace smoothing, which uses $s_j = 1$. Note that this is similar to obtain Bayesian MAP estimators assuming independent Dirichlet priors on the weights associated with a sum node.

Since the evaluation of the likelihood for a structure of an SSPN is tractable, it is possible to use it as a global scoring function to guide a structure search (Peharz, Gens, and Domingos, 2014). The LearnSelectiveSPN algorithm starts with an initial network, then performs a greedy hill-climbing search in the space of structures, applying at each step operations of splitting or merging substructures, selecting the structure that improves the score, until no score improvement is made. The **split** operation extends the structure by processing a product node i and two of its children, l and m, where the network rooted at l contains the indicator $[\![X_k = x_k]\!]$. This operation consists in replacing the arcs from i to l and m by a single arc connecting to a substructure representing possible independencies among the distributions encoded in the subnetworks rooted at l and m. This structure is rooted by a sum node with two product children, each of these product nodes has as a child a copy of the subnetwork rooted at m and differs in the other child, for the first product node is a copy of the subnetwork rooted at l dismissing the arcs connecting with indicator $[\![X_k = x_k]\!]$ and for the second product is a copy of the subnetwork rooted at l connecting only with indicator $[\![X_k = x_k]\!]$. The split operation can create chains of sum or product nodes that are collapsed into a single node. The **merge** operation can be seen as the reverse of the split operation.

2.3.3 Generative Random Forests

A Generative Random Forest (GeF) is an aggregation of Generative Decision Trees, so called because they are generative probabilistic models obtained from a Decision Tree Classifier.

Decision Trees (DTs) are popular models that can perform classification and regression tasks. In this work, we focus on DTs for classification. A DT classifier is a discriminative model that encodes a *conditional probability distribution*, $\mathbb{P}(y|\mathbf{x})$, that is used to predict the value $y \in val(Y)$ of a target variable Y depending on the feature variables $\mathbf{X}$. A DT is a tree whose internal nodes are associated with feature space partitions and leaves are associated with probabilities $\mathbb{P}(y|\mathbf{x}^\rho)$, where $\mathbf{x}^\rho$ is the subset of variables in the partition. Most commonly, partitions are represented by univariate decisions annotated on edges, such as, for example, $X \leq 2$ and $X > 2$.

A decision tree is built by recursively partitioning the feature space to learn decision nodes, in such a way that it optimizes the quality of the partition, by maximizing class impurity measures such as information gain. This partitioning is performed until a stopping condition is reached and then it learns one leaf per sub-region, where each leaf estimates $\mathbb{P}(y|\mathbf{x}^\rho)$ from the instances in the partition. The stopping condition could be the maximum depth or a minimum number of samples. Figure 2.5 shows a decision tree over a dataset with two feature variables $\{X, Z\}$ and a target Y. The numbers inside each node in the decision tree indicate the percentage of instances that fall in the corresponding partition out of those instances that are in the partition defined by the parent node.The edge labels represent the univariate decision for feature space partitions.

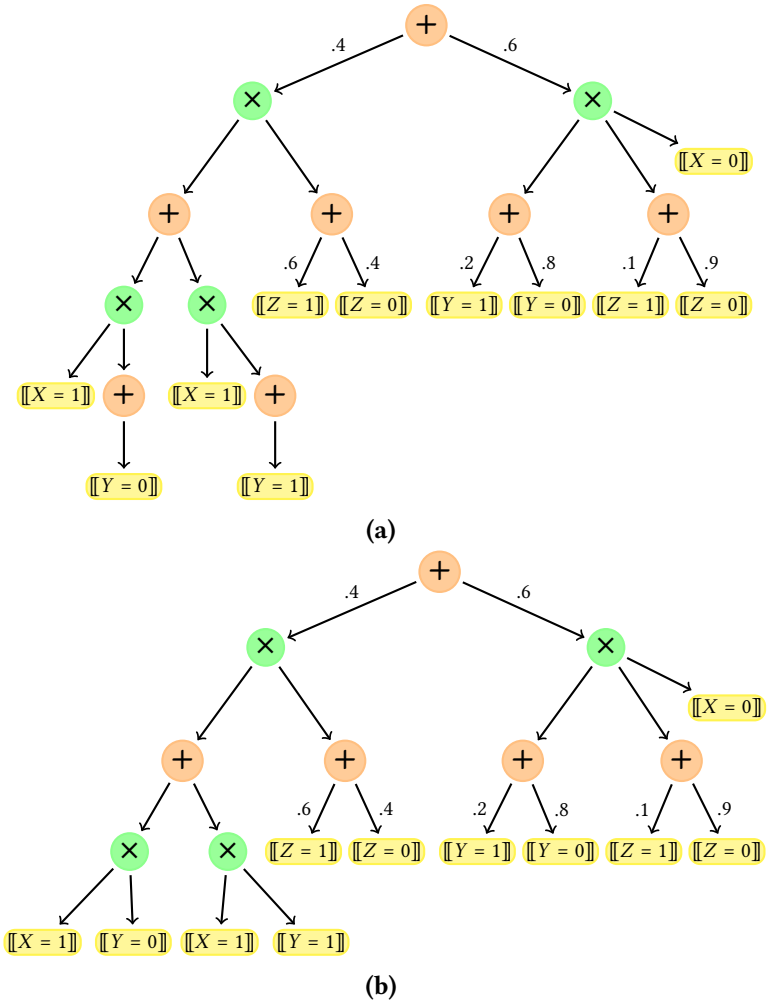


Figure 2.4: *Figure 2.4a shows split operation for the most left product node of SSPN showed in Figure 2.3, the children to split are* $[\![X = 1]\!]$ *and the left most sum child, the spliting occurs over the indicator* $[\![Y = 1]\!]$. *Figure 2.4b shows the collapsed nodes for the SSPN in Figure 2.4a*

Generative classifiers as Probabilistic Circuits have a significant advantage to deal with missing at random features when compared to discriminative classifiers, that consist in marginalizing over the unobserved features. Correia et al. proposed to build a Generative Decision Tree from a DT and data (Correia *et al.*, 2020). Their algorithm starts by converting each decision node of the DT classifier into a sum node and each leaf into a sub-PC whose support is the sub-region induced by the corresponding path of the DT. The weights associated with each outgoing arc in the GeDT are obtained from the percentages (transformed into probabilities) of training instances in the respective sub-region, for example the values at each decision node in Figure 2.5. Algorithm 2 presents the procedure for compiling a DT classifier into a GDT, whose inputs are the root of the DT, a DT classifier and the dataset used to learn the classifier.

In order to speed up computations in GDTs, we can pull up indicator nodes and add product nodes to connect them to the structure. Although this procedure violates decomposability, GeDTs still allow for linear time marginal inference in much the same way as decomposable PCs, due to the selectivity of the indicators. The sub-PCs at the leaves

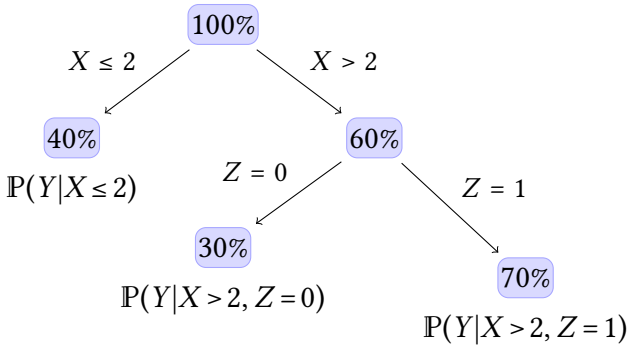


Figure 2.5: *A Decision Tree to classifying Y based on the values of X and Z, nodes show the percentage of instances in each sub-region.*

Algorithm 2: $ConvertDTintoGDT(i, \mathtt{D}, \mathbf{d})$

input : node of the DT: i, Decision Tree classifier: $\mathtt{D}$, dataset with instantiation of variables $\{\mathbf{X}, Y\}$: $\mathbf{d}$
output : An SPN $\mathtt{S}(\mathbf{X}, \mathbf{Y})$ compiled from $\mathtt{D}$ using $\mathbf{d}$

if $internalNode(i, \mathtt{D})$ **then**
  $\mathbf{d}^j \longleftarrow$ partition of $\mathbf{d}$ for child node j
  $\mathtt{S} \longleftarrow \sum_{j=1}^{ch(i)} \frac{|\mathbf{d}^j|}{|\mathbf{d}|} ConvertDTintoGDT(j, \mathtt{D}, \mathbf{d}^j)$
else
  $\mathtt{S} \longleftarrow tractableModel(\mathbf{d})$
end
return $\mathtt{S}$

can be learned with any structure learning algorithm for PCs or take simple forms such as fully factorized distributions. A particularly convenient form for the sub-PCs is to encode a distribution that factorizes as $\mathbb{P}(\mathbf{X}|\mathbf{x}^{\rho_1})\mathbb{P}(Y|\mathbf{x}^{\rho_2})$, where ρ_1 and ρ_2 denote the corresponding partition of the feature space. Such *class-factorized* GeDTs produce the same classifications under complete data as the original DT (Correia *et al.*, 2020). Figure 2.6 shows the GeDT extension of the DT shown in Figure 2.5, note the intentional correspondence between these proportions and the weights of the GeDT.

A GeDT is selective as long as its sub-PCs, i.e. the PCs representing the leaves of the DT, are also selective. Given a selective SPN $\mathtt{S}$, whose scope is defined by $\mathrm{sc}(\mathtt{S}) = \{Y, \mathbf{X}\}$, we say that $\mathtt{S}$ is *strong-selective* w.r.t. Y, if for all sum nodes i in $\mathtt{S}$ whose scope is not a singleton and $Y \in \mathrm{sc}(\mathtt{S}^i)$, there exists a single child of i, $j \in \mathrm{ch}(i)$, which satisfies $\mathtt{S}^j(\mathbf{x}, y^c) > 0$, for all $y^c \in \mathrm{val}(Y)$. Strong selectivity is trivial for nodes that do not contain Y in their scope and also for the singleton scope, which contains only Y. This condition implies that the PC remains selective even when Y is marginalized. Figure 2.6 shows an SPN that is strong selective w.r.t. Y.

Class factorized GeDTs with strictly positive weights imply *strong-selectivity* w.r.t. the target variable, which will be crucial to ensure the exactness of some of our proposed algorithms.

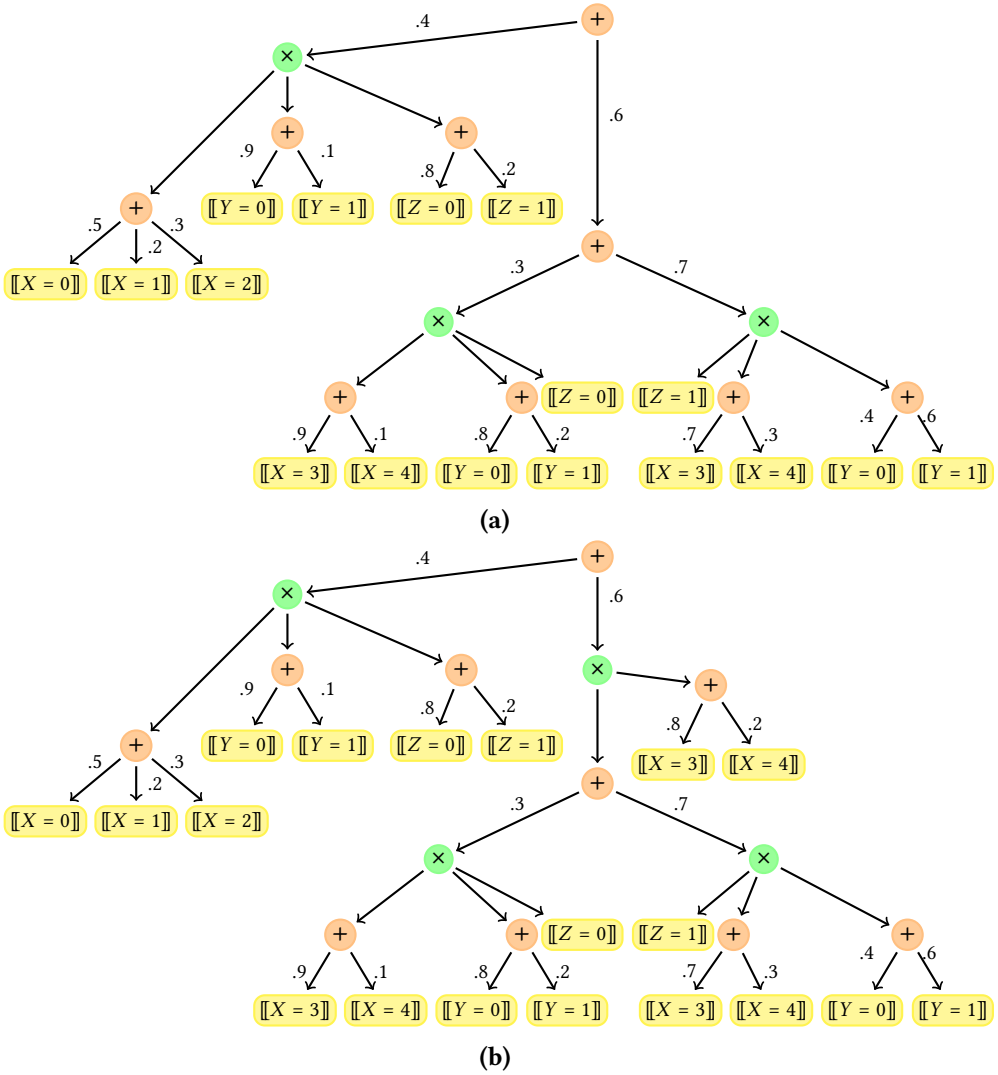


Figure 2.6: *Figure 2.6a shows the Generative Decision Tree with scope $X.Y.Z$ compiled from the the DT in Figure 2.5, which is strong-selective w.r.t. Y. Figure 2.6b shows a GeDT after pulling up indicators $[\![X = 3]\!]$ and $[\![X = 4]\!]$ of the GDTs showed in 2.6a .*

2.3.4 Probabilistic Sentential Decision Diagrams

Probabilistic Sentential Decision Diagrams (PSDDs) are PCs over binary variables, whose support can be read from the graph structure by reinterpreting nodes as logic gates, thus their structure can be seen as a logical circuit encoding a Boolean formula. Kisa et. al. introduce PSDD as a probabilistic extension of *sentential decision diagrams*, a class of logical circuits (Kisa *et al.*, 2014).

To represent a certain knowledge, a PSDD is defined as follows

- Constant i.e. $\{\bot, \top\}$ and literals e.g. $\{X, \neg X\}$ are leaf nodes.
- The internal nodes are either OR or AND gates, analogous to sum and product nodes in SPNs.
  - The AND (product) node i has exactly two children, called *prime* and *sub*, denoted as $i = (p, s)$, encoding $\langle p \rangle \wedge \langle s \rangle$.
  - The OR (sum) node i, denoted as $\{(p_k, s_k)\}_{k \in \text{ch}(i)}$, encodes $\bigvee_{k=\text{ch}(i)} \langle k \rangle$, which is an exclusive disjunction.

Given a PSDD $\mathbb{S}$, each sub-PSDD $\mathbb{S}^i$ rooted at node i also encodes a Boolean formula that is denoted by $\langle i \rangle$.

PSDDs are selective, complete and structured decomposable PCs. We say that a PSDD $\mathbb{S}$ is *structured decomposable* if it is decomposable and any pair of its product nodes i and j with the same scope are decomposed in the same way, that is:

$$(sc(\mathbb{S}^i) = sc(\mathbb{S}^j)) \longrightarrow \forall n \in ch(i) \, : \, \exists! m \in ch(j) / sc(\mathbb{S}^n) = sc(\mathbb{S}^m)$$

for some ordering of the scope of $\mathbb{S}$. Structural decomposability arises from the construction of the circuit where the scopes of product children are given by a vtree in learning time. Figure 2.7 shows a PSDD representing the formula $\phi = (X \wedge Y \wedge Z) \vee (\neg X \wedge Y \wedge \neg Z)$.

Let $\{\mathbf{X}, \mathbf{Z}\}$ be the scope of $\mathbb{S}$, where $\mathbf{X}$ are its left variables and $\mathbf{Z}$ are its right variables. For each sum node i, and each instantiation $\mathbf{xz}$ there is at most one prime p_k such that $\mathbb{S}^{p_k}(\mathbf{xy}) > 0$, this property is called *strong determinism*.

There are several algorithms to learn PCs using different approaches as divide-and-conquer and incremental methods. For example, in this chapter we focused in some of the most popular algorithms to learn SPNs, Selective SPNs and GeDts.

In contrast, the structure of a PSDD is determined by a *vtree*, which is a complete binary tree with leaves corresponding to the domain variables, where each variable occurs exactly once in a vtree leaf node (Darwiche, 2003). The choice of a particular structure can then be thought of as the choice of a particular vtree. Liang et al. developed the LearnPSDD algorithm (A. Choi and Darwiche, 2013; Liang *et al.*, 2017), whose strategy is similar to that of the LearnSSPN algorithm, that performs local modifications to the structure that improve the penalized log-likelihood while maintaining vtree constraints on the order for variables to encode independecies. The ensemble approach to learning PSDDs showed state-of-the-art performance in terms of test-set likelihood in the benchmarks (Liang

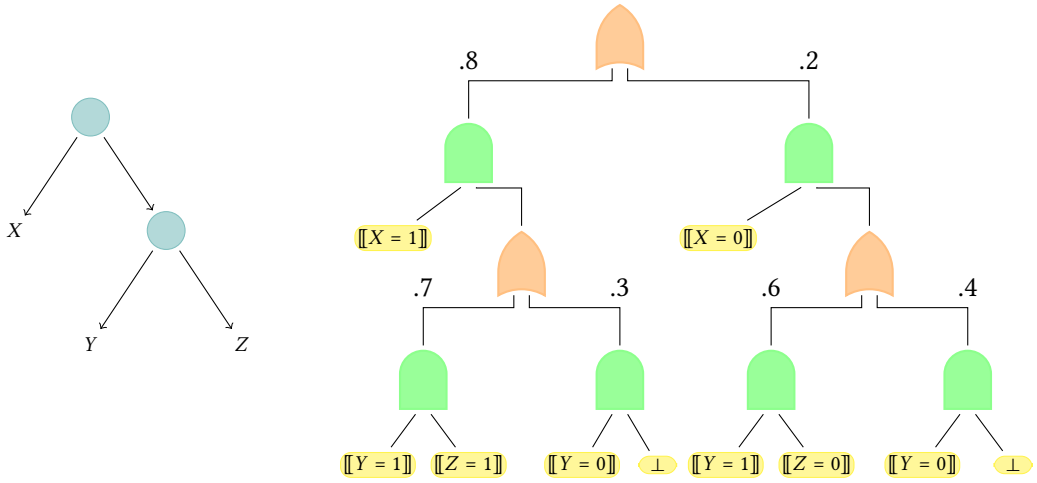


Figure 2.7: *A Probabilistic sentetial decision diagram representing the formula* $(X \wedge Y \wedge Z) \vee (\neg X \wedge Y \wedge \neg Z)$*(right) and its corresponding vtree (left).*

et al., 2017; Geh and Mauá, 2021). Note, however, that the use of ensembles makes MAP inference NP-hard (Conaty *et al.*, 2017).

Example 2.3.3. *Consider a simple machine learning task involving logical constraints over the model variables. The problem consists in the identification of the digit depicted by a seven-segment display (see following Figure), whose segments might occasionally fail to turn on. More specifically, given an input digit to be displayed, the control unit activates the corresponding set of segments in the display; each segment can however fail to be switched on independently with an identical probability.*

This setup can be described by fourteen Boolean variables: say that $\boldsymbol{X} := (X_1, \ldots, X_7)$ *and* $\boldsymbol{O} := (O_1, \ldots, O_7)$*, where the former refer to the states of the segments as decided by the control unit, and the latter are the real states of the segments as depicted in the display. Let us also assume that the true state of these Boolean variables corresponds to the segment on. Given digit, the corresponding configuration of* $\boldsymbol{X}$ *is provided by the formula* $\delta_j(\boldsymbol{X})$ *in the following Table.*

digit	$\delta_j(\boldsymbol{X})$
0	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge \neg X_7$
1	$\neg X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge \neg X_7$
2	$X_1 \wedge X_2 \wedge \neg X_3 \wedge X_4 \wedge X_5 \wedge \neg X_6 \wedge X_7$
3	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge X_7$
4	$\neg X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$
5	$X_1 \wedge \neg X_2 \wedge X_3 \wedge X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$
6	$X_1 \wedge \neg X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge X_7$
7	$X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge \neg X_7$
8	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge X_7$
9	$X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$

Then, for each $i = 1, \dots, 7$, if X_i is false, we also set O_i false, while if X_i is true, O_i might be false with a given failure probability p_f. Such mechanism obeys the formula:

$$\phi := \wedge_{i=1}^{7}(O_i \longrightarrow X_i) \wedge \left(\vee_{j=0}^{9}\delta_j(X_1, \dots, X_7)\right) .$$

We note that this scenario can be extended to more complex and realistic scenarios involving a large number of components/devices, whose interdependence is described as a logical function, and whose probability of failures are interconnected in a complicated way.

Since, this type of scenario involves modeling a joint probability distribution and logical constraints can be encoded by a PSDD, specifically in the case of the segment task can be modeled as a classification problem, given an observation $\boldsymbol{O}$ we want to predict the more likely status for each segment, $\boldsymbol{X_i}$, independently (Mattei et al., 2020).

Chapter 3

Credal Sum-Product Networks

In inductive learning, we can distinguish two essential phases: *training* and *prediction*. Training, also known as the learning phase, consists of adjusting the predictor on the basis of the training data. In the case of SPNs, we learn the parameters and structure that maximize the likelihood of the training data. Once the model is trained, it can then determine the labels associated with new instances comprised in a test set, which is called the prediction phase.

PCs have obtained impressive results in many machine learning tasks due to their ability to represent complicated multidimensional distributions (Poon and Domingos, 2011; Amer and Todorovic, 2016; Zheng *et al.*, 2018; Pronobis, Riccio, *et al.*, 2017; Pronobis and Rao, 2017; Cheng *et al.*, 2014; Geh and Mauá, 2019; Peharz, Vergari, *et al.*, 2020).

Despite the impressive results of several machine learning models on classification benchmarks, most of them, including PCs, can produce unreliable, overconfident predictions in the presence of conflicting data or in regions of insufficient statistical support. Overconfident, incorrect predictions can be harmful in critical applications such as medicine (Lambrou *et al.*, 2010).

A naive approach to identifying reliable predictions is to set a threshold in prediction scores. This approach may fail to identify overconfident incorrect predictions even in accurate models.

Credal Sum-Product Networks (CSPNs) are a set of SPNs that share the same structure, was proposed to perform global sensivity analyis of predictions (Mauá, Conaty, *et al.*, 2018). Performing a global *qualitative* sensitivity analysis allow the model distinguishes between robust and non-robust instances. Given an SPN, sensitivity analysis consists of comparing its prediction against similar inferences produced by the models built "around" the model by "perturbing" the parameters by adding small noisy constants that vary within a set (Mauá, Conaty, *et al.*, 2018).

In this chapter, we formally define Credal Sum-product networks, explaining their construction as well as some tractable inferences, that are the bases for our contributions.

3.1 Credal Sum-Product Networks

A *Credal Sum-Product Network* (CSPN) is a set of normalized SPNs encoding probability distributions where all of them share the same structure (Mauá, Conaty, *et al.*, 2018). A CSPN induces a set over the weights associated with outgoing edges of each sum node, where each parametrization represents a particular SPN encoding a probability distribution, that is, a set of probability distributions.

We denote a CSPN as $\{\mathtt{S}_{\mathbf{w}^*} : \mathbf{w}^* \in C\}$, where C is a subset of the configurations of weights (i.e., parametrizations) in $\mathtt{S}$. The set C is the Cartesian product of sets C_i, one for each sum node i in the network. In this work we consider only local C_i defined by intervals, that is, of the form:

$$C_i = \left\{ \underline{w}_{ij} \leq w_{ij} \leq \overline{w}_{ij}, \sum_{j \in \mathrm{ch}(i)} w_{ij} = 1 \right\}, \tag{3.1}$$

where $0 \leq \underline{w}_{ij} \leq \overline{w}_{ij} \leq 1$ are interval bounds specified by the user. For convenience, we assume that the intervals are *reachable*. This means that for each sum node i and child j, we have:

$$\overline{w}_{ij} \leq 1 - \sum_{j' \neq j} \underline{w}_{ij} \text{ and } \underline{w}_{ij} \geq 1 - \sum_{j' \neq j} \overline{w}_{ij}.$$

We asume reachability to enforce that there exists mass function w_{ij} attained at the bounds (Campos *et al.*, 1994).

A natural method to obtain a CSPN "around" an SPN $\mathtt{S}_{\mathbf{w}^*}$ consists in applying ϵ-contamination, where every sum node i from $\mathtt{S}$ structure satisfies the following condition:

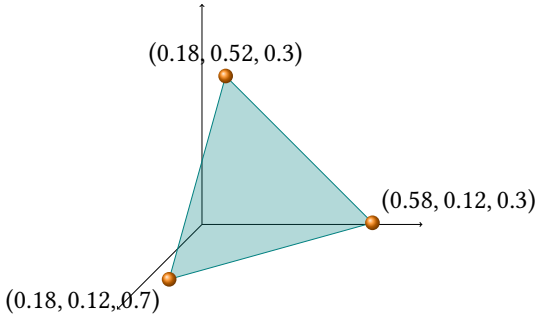
$$C_i = \left\{ (1-\epsilon_i) w^*_{ij} \leq w_{ij} \leq (1-\epsilon_i) w^*_{ij} + \epsilon_i, \sum_{j \in \mathrm{ch}(i)} w_{ij} = 1 \right\}, \tag{3.2}$$

where $\epsilon_i \in (0, 1)$.

Example 3.1.1. *Consider a sum node i with three childs j,k,l where the correspondent weights are:* $w_{ij} = 0.3$ $w_{ik} = 0.2$ *and* $w_{il} = 0.5$ *and* $w_{ij} + w_{ik} + w_{il} = 1$ *Applying Equation 3.2 with 0.4-contamination we obtain:*

$$C_i = \begin{Bmatrix} 0.18 \leq w_{ij} \leq 0.58, \\ 0.12 \leq w_{ik} \leq 0.52, \\ 0.3 \leq w_{il} \leq 0.7 \\ s.t.\ w_{ij} + w_{ik} + w_{il} = 1. \end{Bmatrix}$$

The follow figure shows the geometrical representation of the intervals over the weights associated with arcs emanate from i by 0.4-contamination.



We say that a CSPN is selective if for any choice of weights $\mathbf{w}^*$ the corresponding SPN $S_{\mathbf{w}}$ is selective. Figure 3.1 shows an example of a selective CSPN over three random variables, obtained by 0.01 – contamination of an SPN with weights $\{w_1 = 0.4, w_2 = 0.6, w_3 = 0.7, w_4 = 0.3, w_5 = 0.6, w_6 = 0.4, w_7 = 0.2, , w_8 = 0.8, w_9 = 0.1, w_10 = 0.9\}$.

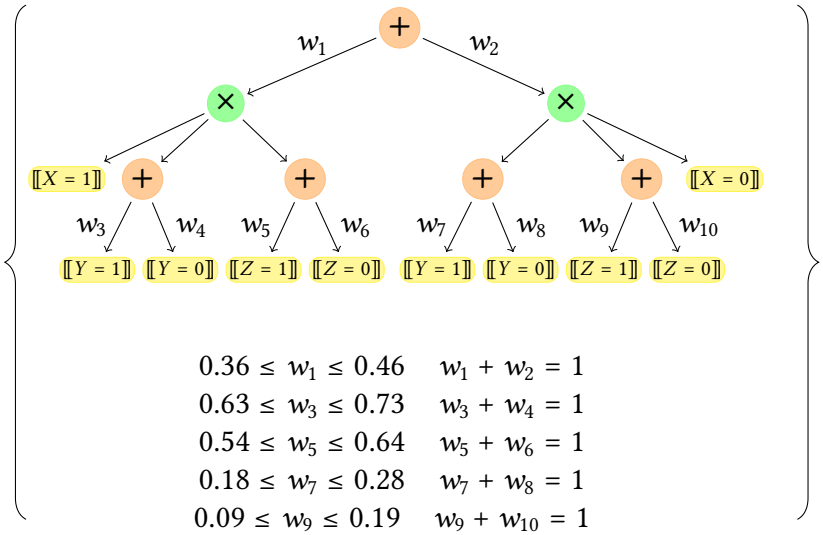


Figure 3.1: *Selective Credal Sum-Product Network obtained by* 0.1*-contamination of the SPN in Figure 2.3*

In Equation 3.2 each weight is perturbed by an ϵ_i regardless of its depth; however, if the (precise) SSPN weights was learned using (smoothed) maximum likelihood estimation, so that deeper nodes are learned from smaller datasets. Thus, we can vary the imprecision by the amount of data used to learn a weight w_{ij} by setting $\epsilon_i = s/(N_i + s)$, where N_i denotes the number of calculation trees containing node i when the model evaluates the training set instances, and $s > 0$ determines the amount of imprecision. This results in a CSPN where every sum node i satisfy:

$$C_i = \left\{ \frac{N_j}{N_i + s} \leq w_{ij} \leq \frac{N_j + s}{N_i + s}, \sum_{j \in \mathrm{ch}(i)} w_{ij} = 1 \right\}, \tag{3.3}$$

The equation above coincides with the Imprecise Dirichlet Model for a (latent) random variable representing the sum node i, whose value "selects" which edge is active. It also coincides with assuming a Multinomial-Dirichlet estimate of weights with the concentration parameters of the Dirichlet prior varying inside a probability simplex.

Analogous to SPN, a PSDD can be converted into a Credal Sentential Decision Diagram (CSDDs), Mattei et al. generated CSDDs whose set values parameters are obtained using imprecise dirichlet model. Similar to PSDDs, the intervals of a CSDD have a probabilistic interpretation as a set of conditional probabilities (Mattei *et al.*, 2020). The algorithms presented in this chapter and next ones for CSPNs are applicable also to CSDDs due to the structural equivalence between SPNs and PSDDs.

3.2 Probability of Evidence

Mauá et al. developed efficient algorithms for producing upper and lower bounds on the probability of the evidence. The computation of upper and lower probability of evidence can be performed in much the same way as the computation of marginal probabilities in SPNs, with a small overhead for solving local optimizations over the interval weights (Mauá, Conaty, *et al.*, 2018).

Given a CSPN $\{\mathrm{S}_{\mathbf{w}} : \mathbf{w} \in C\}$, we are particularly interested in the algorithm to compute the upper joint probability $\overline{\mathrm{P}_{\mathrm{S}}}(\mathbf{x}^{\mathrm{obs}}) = \max_{\mathbf{w}} \mathrm{S}_{\mathbf{w}}(\mathbf{x}^{\mathrm{obs}})$ of a given an evidence $\mathbf{x}^{\mathrm{obs}}$. The algorithm visits nodes in topological reverse order (i.e., from the bottom up), evaluating the following expression at each node i:

$$\overline{\mathrm{S}^i}(\mathbf{x}^{\mathrm{obs}}) = \begin{cases} 1, & \text{if } i \text{ is a leaf node consistent with } \mathbf{x}^{\mathrm{obs}}, \\ 0, & \text{if } i \text{ is a leaf node inconsistent with } \mathbf{x}^{\mathrm{obs}}, \\ \prod_j \overline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a product node}, \\ \max_{\mathbf{w}_i \in C_i} \sum_{j \in \mathrm{ch}(i)} w_{ij} \overline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a sum node}. \end{cases} \tag{3.4}$$

The desired value is obtained at the root r of the network, that is, $\overline{\mathrm{S}^r}(\mathbf{x}^{\mathrm{obs}}) = \max_{\mathbf{w}} \mathrm{S}_{\mathbf{w}}(\mathbf{x}^{\mathrm{obs}})$. Note that for selective credal SPNs and a complete instantiation, the sum in the last equation has at most one positive term, and hence reduces to:

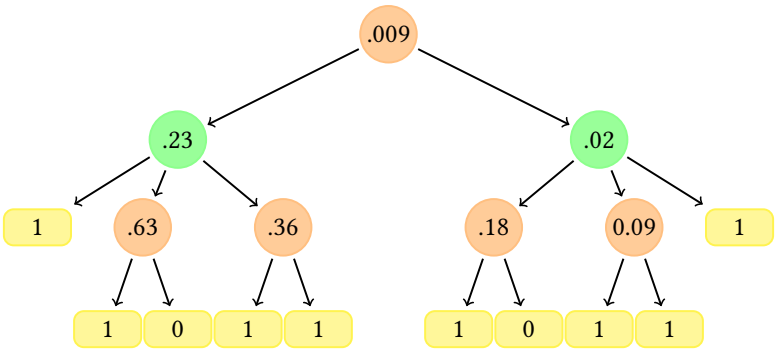
$$\overline{\mathrm{S}^i}(\mathbf{x}^{\mathrm{obs}}) = \begin{cases} 1, & \text{if } i \text{ is a leaf node consistent with } \mathbf{x}^{\mathrm{obs}}, \\ 0, & \text{if } i \text{ is a leaf node inconsistent with } \mathbf{x}^{\mathrm{obs}}, \\ \prod_j \overline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a product node}, \\ \overline{w_{ij}} \overline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a sum node and } j \text{ its active child}. \end{cases} \tag{3.5}$$

where the definition of *active child* is extended to $\overline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}) > 0$.

The computation of the lower probability of evidence is made analogously:

$$\underline{\mathrm{S}^i}(\mathbf{x}^{\mathrm{obs}}) = \begin{cases} 1, & \text{if } i \text{ is a leaf node consistent with } \mathbf{x}^{\mathrm{obs}}, \\ 0, & \text{if } i \text{ is a leaf node inconsistent with } \mathbf{x}^{\mathrm{obs}}, \\ \prod_j \underline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a product node}, \\ \min_{\mathbf{w}_i \in C_i} \sum_{j \in \mathrm{ch}(i)} w_{ij} \underline{\mathrm{S}^j}(\mathbf{x}^{\mathrm{obs}}), & \text{if } i \text{ is a sum node}. \end{cases} \tag{3.6}$$

Example 3.2.1. *Consider the CSPN in Figure 3.1 and the evidence* $\mathbf{x}^{obs} = \{Y = 1\}$*, to obtain the lower probability we compute Equation 3.6 at each node of the circuit as follows.*



As result of the computation, we obtain $\underline{\mathbb{P}}_{\mathbb{S}}(Y = 1) = 0.009$

3.3 Credal Classification

One of the most popular tasks that PCs perform is probabilistic classification, which consists in obtaining the most probable class associated with a given observation using the circuit. Since CSPNs define more than a single model, there are several possible maximizers (Troffaes, 2007). A very popular criterion for decision-making with imprecise probability models is *Credal Classification* based on the maximality principle (Zaffalon, 2002).

Given a class variable Y, an observation or evidence $\mathbf{X}^{\text{obs}} = \mathbf{x}^{\text{obs}}$, and a CSPN $\mathbb{S}_{\mathbf{w}}$ with scope $\mathbf{X}^{\text{obs}}$, Y, we say that an assignment y' for Y credally dominates another assignment y'' if:

$$\delta(y', y'') = \min_{\mathbf{w} \in C} [\mathbb{S}_{\mathbf{w}}(y', \mathbf{x}^{\text{obs}}) - \mathbb{S}_{\mathbf{w}}(y'', \mathbf{x}^{\text{obs}})] > 0. \tag{3.7}$$

Credal classification consists in computing the set of non-dominated classes for variable class Y, this procedure can be computed in polynomial time in CSPNs when each internal node has at most one parent, a number of classes is bounded (Mauá, Conaty, *et al.*, 2018). Their algorithm computes dominance as showed in equation 3.7 and can be described straightforwardly by a collection of equations depending on the type of node at which it operates as follows:

Sum nodes. If i is a sum node then the algorithm computes:

$$\delta_i(y', y'') = \min_{\mathbf{w}_i \in C_i} \sum_{j=\text{ch}(i)} \delta_j(y', y''). \tag{3.8}$$

Product nodes. If instead i is a product node, such that Y is in the scope of $\mathbb{S}^k$ (and no other), where $k \in \text{ch}(i)$, then the algorithm computes:

$$\delta_i(y', y'') = \delta_k(y', y'') \prod_{j=\text{ch}(i), j \neq k}^{n} \text{opt } \mathbb{S}^j(\mathbf{x}_j^{\text{obs}}), \tag{3.9}$$

where $\mathbf{x}_j^{\text{obs}}$ denotes the projection of $\mathbf{x}^{\text{obs}}$ into the scope of $\mathbb{S}^j$, and

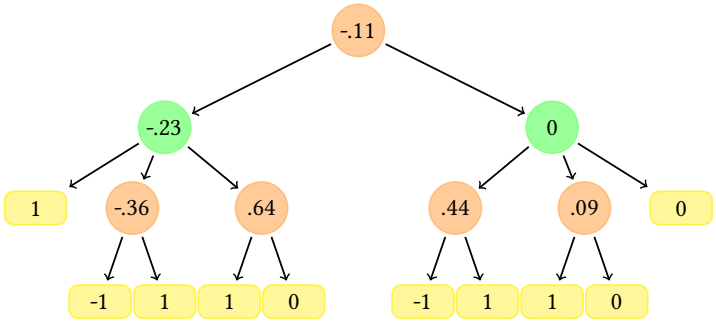
$$\text{opt} = \begin{cases} \max & \text{if } \delta_k(y', y'') > 0\,, \\ \min & \text{if } \delta_k(y', y'') \leq 0\,. \end{cases}$$

The first term denotes the recursive computation on the sub-SPN $\mathbb{S}^k$.

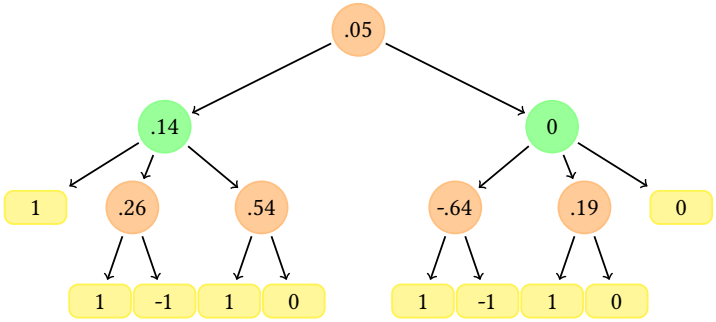
Leaves. Finally, if i is a leaf node representing an indicator variable then the algorithm computes:

$$\delta_i(y', y'') = \begin{cases} -1, & \text{if } \mathbb{S}^i \text{ is } [\![Y = y'']\!], \\ 1, & \text{if } \mathbb{S}^i \text{ is } [\![Y = y']\!], \\ 1, & \text{if } \mathbb{S}^i \text{ is consistent with } \mathbf{x}^{\text{obs}} \text{ and } Y \text{ is not in its scope}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

Example 3.3.1. *Consider the CSPN in Figure 3.1 and the evidence* $\mathbf{x}^{obs} = \{X = 1, Z = 1\}$*, we want credally classify the target variable* Y*. Firstly we show the computations using Equation 3.7 to assess dominance of 1 over 0 as follows.*



From the algorithm we obtain that $\delta(Y = 0, Y = 1) = -0.11 < 0$ *indicating that 0 not dominates class 1. Then we also want to determine if class 0 domains class 1 in the circuit.*



As the computation of $\delta(Y = 1, Y = 0) = 0.05$*, we know that class 1 domains 0, the credal prediction for class* Y *is the set of non dominate classes, that in this case have a unique ,* $Y = \{1\}$*.*

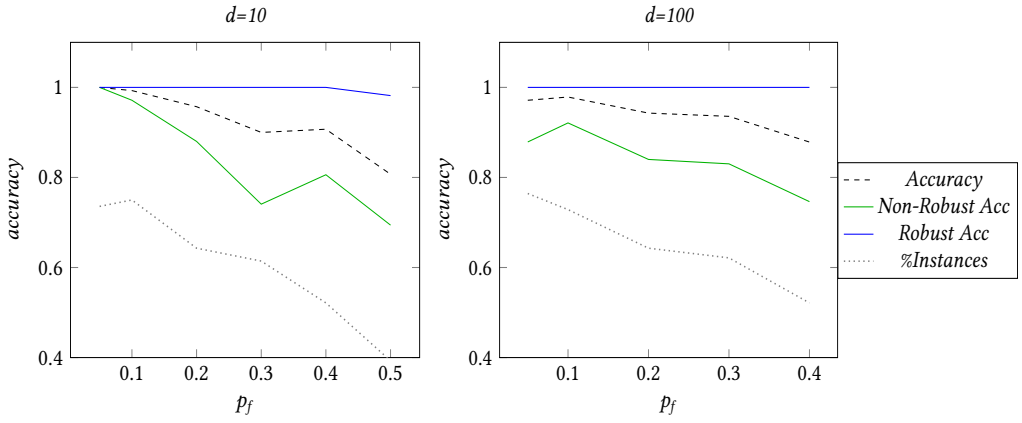
We consider such an inference *robust* if it remains the single maximizer under small perturbations of the model parameters, additionally the instances associated with robust

predictions are considered as easy-to-classify. CSPNs can then support global sensitivity analysis by evaluating the robustness of predictions. In general, credal classifiers are able to distinguish between robust and non-robust instances; this separation corresponds to the difference between the instances for which the output of the classifier is a single or a set of predictions, respectively (Mauá, Conaty, *et al.*, 2018; Villanueva Llerena and Mauá, 2020; Mattei *et al.*, 2020).

Credal classifiers can be used as reject option classifiers, which suspend prediction when the available information is insufficient to make a final (single) decision.

Example 3.3.2. *Consider the seven-segment problem, presented in Example 2.3.3 in Chapter 2. Given an input digit to display, the control unit activates the corresponding set of segments on the display. However, each segment may fail to switch independently with equal probability. The task was to predict the more likely status for each segment, $\boldsymbol{X_i}$, given an observation in the seven-segment display, $\boldsymbol{O}$. As we showed, this task was solved by learning PSDDs.*

Now we also want to identify robust predictions. To do this, we can perform a qualitative sensitivity analysis based on CSDDs obtained using Equation 3.3 with $s = 1$. The following figures show that CSDDs are able to discriminate robust inferences using different training sizes, d, and failure probabilities, p_f. Robust inferences maintain high accuracy (close to one) even with high perturbation levels, where the perturbation only affects the percentage of robust instances. The experiments used artificially generated data of segments (Mattei et al., 2020).



Chapter 4

Tractable Global Qualitative Sensitivity Analysis for Maximun A Posteriori Inference

The previous chapter reviewed concepts of CSPN that can be used for qualitative global sensitivity analysis for binary and multiclass classification. However, 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 MAP inference, such as image completion and multi-label classification tasks (Poon and Domingos, 2011; Villanueva Llerena and Mauá, 2017), where each instance can be associated with one or more labels, Figure 4.1 shows examples of multiclass and multi-label classification.

Figure 4.1: *Example of multiclass (left) and multi-label classification (right)*

In this chapter, we present our first contribution: an algorithm that performs a global qualitative sensitivity analysis of Maximum-A-Posteriori (MAP) inference for selective SPNs, by identifying robust inferences to global perturbations of the parameters. We present efficient algorithms and an empirical analysis on realistic problems involving missing data completion and multi-label classification. This chapter is organized as follows: First, we discuss qualitative global sensitivity analysis in SPNs, where we show that it is NP-hard in non-selective SPNs. Then we present optimistic MAP inference in selective CSPNs, which will be necessary for the procedure of our main contribution, then we present our approach to evaluate the robustness of MAP inferences in selective SPNs,

we describe our experiments concerning qualitative robustness analysis in missing data completion and multi-label classification.

4.1 Qualitative Global Sensitivity Analisys of MAP inferences in SPNs

Our proposed approach classifies the sensitivity of Maximum-A-Posteriori inference obtained from a Selective SPN to global perturbations of its parameters. We consider an instance $\mathbf{x}^* \in val(\mathbf{X}^{\text{miss}})$ as *robust* with respect to a credal sum-product network $\{\mathtt{S}_\mathbf{w} : \mathbf{w} \in C\}$ and evidence $\mathbf{x}^{\text{obs}}$ if $\mathbf{x}^* \in \mathbf{X}^{\text{miss}}$ is the single maximizer of each SPN $\mathtt{S}_\mathbf{w}$ (Villanueva Llerena and Mauá, 2019), that is, if:

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

Note that for SPNs (i.e., when C is a singleton) the problem reduces to deciding if there *not* exists an instantiation $\mathbf{x}^{\text{miss}} \neq \mathbf{x}^*$ such that $\mathtt{S}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) \geq \mathtt{S}(\mathbf{x}^*, \mathbf{x}^{\text{obs}})$. Bodlaender et al. (Bodlaender *et al.*, 2002) showed that the complementary problem (i.e., deciding if there *is* such an instatiantion) is NP-hard for Bayesian networks; one can adapt their proof using ideas from (Conaty *et al.*, 2017) to show that deciding if an arbitrary instance $\mathbf{x}^*$ is robust is coNP-hard. The following theorem strengthens that result for the case when $\mathbf{x}^*$ is a MAP configuration (and also provides a direct proof):

Theorem 4.1.1. *Deciding if an instance $\mathbf{x}^*$ is robust w.r.t. a CSPN $\{\mathtt{S}_\mathbf{w} : \mathbf{w} \in C\}$ is coNP-complete, even if $\mathbf{x}^*$ is a maximum probability configuration and C is a singleton.*

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

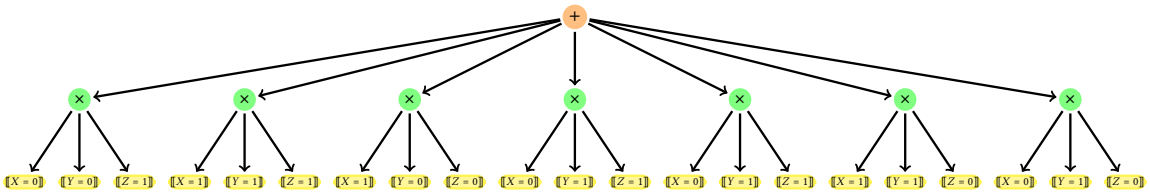


Figure 4.2: *An SPN encoding the satisfying assignments of the Boolean formula $\neg X \vee Y \vee \neg Z$.*

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 $\mathtt{S}_\psi$ that encodes each of the 7 satisfying assignments of that clause as a product of indicator variables, and such that $\mathtt{S}_\psi$ is selective and evaluates to 1/7 iff the corresponding assignment satisfies the clause. Figure 4.2 shows an example of such a network.

To extend the scope of the subnetwork to all the variables, create a network $\mathtt{S}'_\psi$, that is the product of $\mathtt{S}_\psi$ with a uniform distribution over the variables that do not appear in ψ. We thus have that $\mathtt{S}'_\psi(\mathbf{x}^{\text{miss}}) = \frac{1}{7}\frac{1}{2^{n-3}}$ if and only if $\mathbf{x}^{\text{miss}}$ satisfies ψ. Now connect all subnetworks $\mathtt{S}'_\psi$ via a sum node with edge weights $1/m$ where m is the number of clauses. Call this network $\mathtt{S}_\phi$. It follows that $\mathtt{S}_\phi(\mathbf{x}^{\text{miss}}) = \frac{1}{7}\frac{1}{2^{n-3}}$ iff ϕ is satisfied by the corresponding assignment $\mathbf{x}^{\text{miss}}$ and $\mathtt{S}_\phi(\mathbf{x}^{\text{miss}}) \leq \frac{1}{7}\frac{1}{2^{n-3}}\frac{m-1}{m}$ iff ϕ is not satisfied by $\mathbf{x}^{\text{miss}}$.

Now build a network $\mathtt{S}$ with a sum node that has $\mathtt{S}_\phi$ as one child, with an edge weight w, and subnetwork $\mathtt{S}_*$ that encodes the assignment $x_1^*, \ldots, x_n^*$ as the other child (with edge weight $1-w$); the latter subnetwork is simply a product node connected to indicator variables $[\![X_i^* = x_i^*]\!]$. First note that $\mathtt{S}(\mathbf{x}^*) = \frac{kw}{7m2^{n-3}} + (1-w)$: as $\mathtt{S}_\phi(\mathbf{x}^*) = \frac{k}{7m2^{n-3}}$ and $k = |\psi : \mathbf{x}^* \text{ satisfies } \psi|$ i.e. the number of clauses that $\mathbf{x}^*$ satisfies. Choose w to make $\mathtt{S}(\mathbf{x}^*) = \mathtt{S}(\mathbf{x})$ such that $\mathbf{x}$ satisfies ϕ, then $w = \frac{7m2^{n-3}}{m+7m2^{n-3}-k}$, and that can be computed in polynomial time on the size of the input. Thus, the instatiantion $\mathbf{x}^*$ is *not* robust (i.e., it is not the unique maximizer of $\mathtt{S}$) iff ϕ is satisfiable. This proves that deciding if a (MAP) instantiation is robust is coNP-hard. □

4.2 Optimistic Maximum A Posteriori inference in Credal Selective Sum-Product Networks

Recall that although MAP inference is NP-hard in SPNs even when restricted to binary variables and height-two networks (Conaty *et al.*, 2017; Mei *et al.*, 2017; Peharz, Gens, Pernkopf, *et al.*, 2017), the problem is solvable in linear time in the size of the network for Selective SPNs by the simple Max-Product algorithm as we reviewed in Chapter 2.

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 maximum joint probability over the set of selective SPNs in a selective CSPN. This will be used as a subroutine to classify robustness later. Thus consider a selective CSPN $\{\mathtt{S}_\mathbf{w} : \mathbf{w} \in C\}$ where C is given by the Cartesian product of sets C_i for each sum node i, whose scope is $\mathbf{X} = \{\mathbf{X}^{\text{miss}}, \mathbf{X}^{\text{obs}}\}$ and $\mathbf{X}^{\text{miss}} \cap \mathbf{X}^{\text{obs}} = \emptyset$. Given an evidence $\mathbf{x}^{\text{obs}} \in val(\mathbf{X}^{\text{obs}})$, we are interested in computing the value of MAP instantiation $\mathbf{x}^* \in \mathbf{X}^{\text{miss}}$ of the upper bound of the joint probability:

$$\max_{\mathbf{x}^{\text{miss}}} \max_{\mathbf{w} \in C} \mathtt{S}_\mathbf{w}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) \,. \tag{4.1}$$

We propose an algorithm that computes the above quantity by traversing the network in

reverse topological order, calculating at each node i the following:

$$\overline{M^i} = \begin{cases} 1, & \text{if } i \text{ is a consistent leaf node,} \\ 0, & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod\limits_{j \in \mathrm{ch}(i)} \overline{M^j}, & \text{if } i \text{ is a product node,} \\ \max\limits_{j \in \mathrm{ch}(i)} \overline{w_{ij}}\,\overline{M^j}, & \text{if } i \text{ is a sum node.} \end{cases} \tag{4.2}$$

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

Theorem 4.2.1. *Consider a selective CSPN $\{\mathtt{S}_{\mathbf{w}} : \mathbf{w} \in C\}$ with root r, where C is the Cartesian product of finitely-generated polytopes C_i, one for each sum node i. Then $\overline{M}^r$ computes $\max\limits_{\mathbf{x}^{miss}} \max\limits_{\mathbf{w}} \mathtt{S}_{\mathbf{w}}(\mathbf{x}^{miss}, \mathbf{x}^{obs})$ 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 correctness 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 leaves, 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 $\mathtt{S}^j(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) > 0$ for each instantiation $\mathbf{x}^{\text{miss}}$. Let $\mathcal{X}_j$ denote the set of instantiations $\mathbf{x}^{\text{miss}} \in val(\mathbf{X})$ for which $\mathtt{S}^j(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) > 0$. Then

$$\max_{\mathbf{x}^{\text{miss}} \in val(\mathbf{x}^{\text{miss}})} \max_{\mathbf{w}} \mathtt{S}^i_{\mathbf{w}}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) = \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{x}^{\text{miss}} \in \mathcal{X}_j} \max_{\mathbf{w}} w_{ij} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}).$$

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

$$\begin{aligned} \max_{\mathbf{x}^{\text{miss}}} \max_{\mathbf{w}} \mathtt{S}^i_{\mathbf{w}}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) &= \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{w}_{ij} \in C_{ij}} w_{ij} \max_{\mathbf{x}^{\text{miss}}} \max_{\mathbf{w}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) \\ &= \max_{j \in \mathrm{ch}(i)} \max_{\mathbf{w}_{ij} \in C_{ij}} w_{ij} M^j = M^i. \end{aligned}$$

If i is a product node, we have that

$$\begin{aligned} \max_{\mathbf{x}^{\text{miss}}} \max_{\mathbf{w}} \prod_{j \in \mathrm{ch}(i)} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\text{miss}}) &= \prod_{j \in \mathrm{ch}(i)} \max_{\mathbf{x}^{\text{miss}}} \max_{\mathbf{w}_j} \mathtt{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\text{miss}}, \mathbf{x}^{\text{obs}}) \\ &= \prod_{j \in \mathrm{ch}(i)} M^j = M^i. \end{aligned}$$

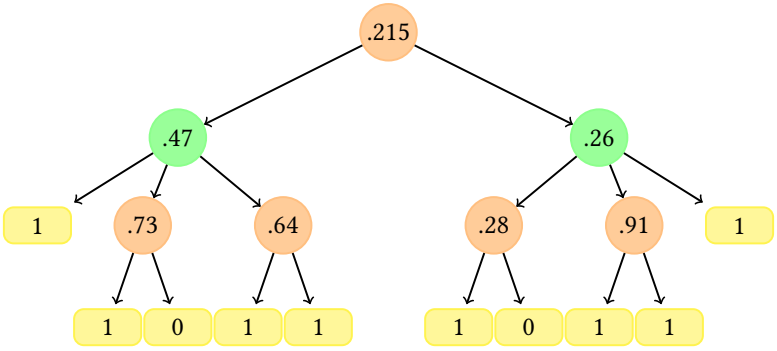
The weights $\mathbf{w}_j$ in the first equation can be optimized independently for each $j \in \mathrm{ch}(i)$, and the disjointedness of scopes for product nodes ensures that no weight is shared among the child sub-networks.

If we memorize 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 $O(|\mathbb{S}|)$. □

Theorems 4.1.1 and 4.2.1 justify our focus on selective SPNs, where deciding if an MAP instantiation is robust is tractable.

Example 4.2.1. *Consider the CSPN in Figure 3.1. We want to compute Equation 4.1 for maximazing variables* $\mathbf{x}^{miss} = \{X, Z\}$ *and evidence* $\mathbf{x}^{obs} = \{Y = 1\}$*, using the algorithm in Equation 4.2*



As the result of the algorithm, we obtain the upper probability of MAP instantiation at the root, that is 0.215.

4.3 Robustness of Maximum-A-Posteriori Inferences in Selective Sum-Product Networks

The following algorithm decides if a MAP instantiation, $\mathbf{x}^*$, is robust, by traversing the network from the leaves to the root, computing, for each node i:

$$V^i = \begin{cases} 1, & \text{if } i \text{ is a consistent leaf node,} \\ 0, & \text{if } i \text{ is an inconsistent leaf node,} \\ \prod_j V^j, & \text{if } i \text{ is a product node,} \\ \max\left\{ \max_{j \in \mathrm{ch}(i), j \neq k} U^j,\ V^k \right\}, & \text{if } i \text{ is a sum node,} \end{cases} \tag{4.3}$$

where k is the maximizing child of node i when MAP instantiation $\mathbf{x}^*, \mathbf{x}^{\mathrm{obs}}$ was computed in $\mathbb{S}$ and:

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

The values M^j and L^k are computed, respectively, using the algorithm previously described, and the algorithm for computing lower evidence probability.

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

Theorem 4.3.1. *Consider a tree-shaped selective CSPN* $\{\mathrm{S}_{\mathbf{w}} : \mathbf{w} \in C\}$ *with root r, where C is the Cartesian product of finitely-generated polytopes* C_i*, one for each sum node i. Then* V^r *computes*

$$\max_{\mathbf{x}^{miss}} \max_{\mathbf{w} \in C} \left(\frac{\mathrm{S}_{\mathbf{w}}(\mathbf{x}^{miss}, \mathbf{x}^{obs})}{\mathrm{S}_{\mathbf{w}}(\mathbf{x}^{*}, \mathbf{x}^{obs})} \right)$$

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

Proof. We prove that the algorithm is correct by induction in the height h of 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 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 $val(\mathbf{x}^{\mathrm{miss}})$ for which $\mathrm{S}^j(\mathbf{x}^{\mathrm{miss}}, \mathbf{x}^{\mathrm{obs}}) > 0$. We have that

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

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

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

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

$$\begin{aligned}
\max_{\mathbf{w}} \frac{w_{ij} \max_{\mathbf{x}^{\mathrm{miss}} \in \mathcal{X}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\mathrm{miss}}, \mathbf{x}^{\mathrm{obs}})}{w_{ik} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^{*}, \mathbf{x}^{\mathrm{obs}})} &= \max_{\mathbf{w}_i \in C_i} \frac{w_{ij} \max_{\mathbf{x}^{\mathrm{miss}}} \max_{\mathbf{w}_j} \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\mathrm{miss}}, \mathbf{x}^{\mathrm{obs}})}{w_{ik} \min_{\mathbf{w}_k} \mathrm{S}^k_{\mathbf{w}_k}(\mathbf{x}^{*}, \mathbf{x}^{\mathrm{obs}})} \\
&= \max_{\mathbf{w}_i \in C_i} \frac{w_{ij} M^j}{w_{ik} L^k(\mathbf{x}^{*}, \mathbf{x}^{\mathrm{obs}})} = U^j.
\end{aligned}$$

If i is a product node then

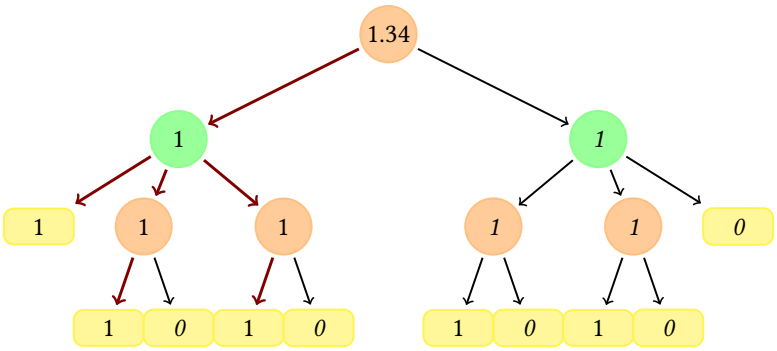
$$\begin{aligned}
\max_{\mathbf{x}^{\mathrm{miss}}} \max_{\mathbf{w}} \left(\frac{\prod_j \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\mathrm{miss}})}{\prod_j \mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^{*})} \right) &= \max_{\mathbf{x}^{\mathrm{miss}}} \max_{\mathbf{w}} \prod_{j \in \mathrm{ch}(i)} \frac{\mathrm{S}^j_{\mathbf{w}_j}(\mathbf{x}^{\mathrm{miss}},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

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 $O(|\mathrm{S}^k|)$. □

Note that the complexity of verifying robustness in tree-shaped selective CSPNs 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 produce MAP inference with little overhead. The complexity of deciding robustness for multiply-connected selective CSPNs remains an open question.

Example 4.3.1. *Consider the CSPN in Figure 3.1 and the evidence* $\mathbf{x}^{obs} = \{Y = 1\}$*, we want to perform robustness analysis of the MAP configuration* $\boldsymbol{x}^* = \{X = 1, Z = 1\}$*. The nodes are assigned the values* V^i *when they are computed. The highlighted edges represent the active nodes of the precise (original) selective SPN (i.e., the ones in* $\mathtt{T}_S(\mathbf{x}^*, \mathbf{x}^{obs})$*).*



The figure displays the values of V^i *for the corresponding nodes computed using Equation 4.3 (note that* V^i *is computed only for nodes in the circuit tree of* $\mathbf{x}^*, \mathbf{x}^{obs}$*). The MAP instance is considered not robust because* $V^r = U^r$*, where* $U^r = \frac{\overline{w}_{ij} M^j}{\underline{w}_{ik} L^k(\boldsymbol{x}^*, \mathbf{x}^{obs})}$ *as computed in examples 4.2.1 and 3.2.1 respectively,* $U^r = (0.64 * 0.26)/0.122 \approx 1.34 > 1$*.*

4.4 Experiments

We perform experiments on real-world datasets to show that our algorithm for global qualitative sensitivity analysis of MAP inferences in selective SPNs can discriminate between easy- and hard-to-classify instances, often more accurately than criteria based on (precise) probabilities induced by the model. We present experiments first with selective CSPNs and then with CSDDs, exploring the logical representation provided by such models.

4.4.1 Estimating MAP Robustness in Selective Sum-Product Networks

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 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 $\mathbf{X}^{\text{miss}}$ be the missing part and $\mathbf{x}^{\text{obs}}$ the non-missing part. The second task considers 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 a realization of features $\mathbf{x}^{\text{obs}}$.

Experimental Set-Up

For both tasks, we learn selective SPNs using the algorithm by Peharz et al. (Peharz, Gens, and Domingos, 2014). We then obtain CSPNs using the two methods described in Chapter 2, either by a *non-local* perturbation, where all the parameters are perturbed by the same value, using non-local perturbation as presented in Equation(3.2), or by *local* perturbation, which varies the imprecision by the amount of data used to learn each parameter using Equation(3.3), setting the level of imprecision to ϵ or s, respectively (more on this later). We use the CSPNs to classify each MAP inference drawn with the (precise) SPN as either robust or non-robust, and compare the performance of the robust and non-robust classifications.

We expect that the performance (e.g., accuracy) of instances classified as robust will be higher than that of instances classified as non-robust, 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 likely configuration consistent with the evidence). An instance is considered robust in this scheme if this difference is greater than a given threshold, we call this crieria as *probability difference*.

Completion

We start with the completion task. For this purpose, we learned selective SPNs from four well-known density estimation datasets(Davis and Domingos, 2010), available at https://github.com/arranger1044/DEBD. The selected datasets and their characteristics are listed in Table 4.1. These datasets are complete (i.e., have no missing values), are divided into training, validation and test parts, and had the variables been binarized. We build completion tasks using the test examples by running MAP inference on the first 50% 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 hyperparameters, and evaluate the methods, respectively.

Dataset	Training Examples	Test Examples	Evidence Vars.	Query Vars.
Jester	9000	4116	50	50
NLTCS	16181	3236	8	8
MSNBC	291326	58265	9	8

Table 4.1: *Characteristics of datasets used in completion tasks.*

We measure the performance of the completions either by Hamming Score (HS) or by the Exact Match (EM) metric. The Hamming score measures the percentage of correct value completions:

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

where $\mathbf{x}^*_i$ is the ith MAP instantiation, $\hat{\mathbf{x}}_i$ is the ith true completion, $x^*_{i,j}$ and $\hat{x}_{i,j}$ are single values of the MAP and the true completion, respectively; N is the number of instances

and L is the number of missing values (50% of the number of variables in this case) or labels. This metric can be misleadingly high if one value (say, 0) is much more frequent than the other values across a large number of variables (so simply predicting that value is accurate); it is also maximized by selecting for each variable, independently, the value that maximizes the conditional marginal probability given the evidence. Exact match addresses some of these issues, by calculating the percentage of perfect completions:

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

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. In combination, the two metrics help to evaluate completions from different perspectives.

Table 4.2 contains the performance metrics of the completions performed with (precise) SPN model. Note that the Exact Match value is very low for the Jester dataset; Hamming score on the hand is relatively high for all datasets, especially NLTCS.

Dataset	Exact Match	Hamming Score
Jester	0.003	0.69
NLTCS	0.37	0.81
MSNBC	0.25	0.78

Table 4.2: *Performance of completions with the (precise) SPN.*

We first perform experiments in which we select the values of ϵ, s and p for local, non local methods to obtain a CSPN and the probability difference threshold so that at least half of the instances in the test set are classified as robust (as a small increase of any of these parameters can "turn" the classification of several instances, we cannot adjust for a equal split of the instances). This scenario corresponds to a moderately conservative approach.

The performance of the completions (made with the precise SPN) in the robust and in the non-robust parts of the set are show in Table 4.3; ϵ, s and p stand, respectively, for local, non-local perturbations and the probability difference criteria. The best performing method for each dataset and partition (robust/non-robust) appears in bold.

Considering the Jester dataset, the instances deemed robust by the CSPN-based analyses had a much higher exact match than the instances deemed non-robust (which had zero score). The probability difference criterion on the other hand considered all instances as robust.

Regarding Hamming score, the set of deemed robust instances obtained with non-local perturbation scored lower than the set of deemed non-robust instances, although this difference was very small. The classifications with the local perturbation models faired much better; robust instances achieved 0.02 points higher score than non-robust instances. One possible explanation for the behavior of the methods is that, despite the fact that this dataset has less data compared to its dimension, most completions are relatively robust

Dataset	Method	%I	Exact Match			Hamming Score		
			Robust	¬*Robust*	Δ*EM*	*Robust*	¬*Robust*	Δ*HS*
Jester	ϵ = 0.01	50.02	**0.004**	0.001	0.003	0.67	0.71	-0.04
	s = 5	76.17	**0.005**	0.001	0.004	**0.71**	0.63	**0.02**
	p = 0	100	0.003	–	–	0.69	–	–
NLTCS	ϵ = 0.07	54.36	0.52	0.19	0.33	0.86	0.75	0.11
	s = 50	48.2	**0.56**	0.2	0.36	**0.88**	0.75	0.13
	p = 0.17	55.96	0.54	0.16	**0.38**	0.87	0.73	**0.14**
MSNBC	ϵ = 0.025	51.04	**0.35**	0.14	**0.21**	**0.84**	0.72	**0.12**
	s = 700	62.21	0.26	0.22	0.04	0.78	0.79	-0.01
	p = 0.13	91.69	0.25	0.17	0.08	0.78	0.77	0.01

Table 4.3: *Performance of completions when roughy 50% of instances are deemed robust; the symbol – is used to indicate that all instances have been deemed robust by the corresponding criterion.*

(to perturbations), and assuming that about half of the instances are non-robust is too conservative.

Considering the NLTCS dataset, all three methods performed similarly, with the accuracy of instances deemed robust w.r.t. CSPNs obtained by the local perturbation slightly outperforming the other methods in terms of both Hamming Score and Exact Match; and with probability difference slightly outperforming the other criteria in terms of difference between scores in robust/non-robust instances. We can explain the result as stating that most of the lack of robustness in completions is due to the existence of multiple high probability instantiations (these produce lower probability difference scores but are also most likely considered non-robust w.r.t. CSPNs).

For the MSBNC dataset, the CSPN-based approach using non-local perturbation is the better at discriminating accurate and inaccurate completions: the robust instances have higher scores, and the difference is also greater. Local perturbation based inferences show a small difference (with non-robust instances obtaining very slightly higher score) and do probability difference inferences (with robust instances have very slightly higher score).

We also performed a more conservative scenario where the values of ϵ, s and the probability difference threshold are set so that about 10% of the instances in the test set are deemed robust. The results are in Table 4.4.

The results are qualitatively similar to the previous scenario. In the Jester dataset, the criteria based on CSPNs do distinguish instances with higher Exact Match, although the increase is small with respect to the score of instances deemed non-robust. In the NLTCS dataset, all results obtain identical performance. In the MSNBC, probability difference discriminates better robust (accurate) and non-robust (inaccurate) instances.

To have a more complete picture of the ability that each method provides for discriminating robust and non-robust instances, we computed, for each instances the maximum value of imprecision (ϵ or s) such that the instance is considered robust. Call such value the robustness of that instance. Figure 4.3 shows the test-set performance versus the robustness of instances (solid lines) for both Exact Match and Hamming Score. We also

Dataset	Method	%I	Exact Match			Hamming Score		
			Robust	¬*Robust*	Δ*EM*	*Robust*	¬*Robust*	Δ*HS*
Jester	$\epsilon = 0.035$	12.17	**0.018**	0.001	**0.017**	0.71	0.69	0.02
	$s = 25$	10.84	**0.018**	0.001	**0.017**	0.71	0.69	0.02
	$p = 1.18e-6$	10.67	0.005	0.002	0.003	**0.81**	0.68	**0.13**
NLTCS	$\epsilon = 0.15$	25.06	0.74	0.25	0.49	0.93	0.77	0.16
	$s = 200$	25.06	0.74	0.25	0.49	0.93	0.77	0.16
	$p = 0.6$	25.06	0.74	0.25	0.49	0.93	0.77	0.16
MSNBC	$\epsilon = 0.085$	9.49	0.35	0.23	0.12	0.84	0.77	0.07
	$s = 2100$	8.94	0.39	0.23	0.16	0.85	0.78	0.07
	$p = 0.31$	16.08	**0.46**	0.2	**0.26**	**0.88**	0.77	**0.11**

Table 4.4: *Performance of completions when roughly 10% of instances are considered robust (for each method); the symbol – is used to indicate that all instances have been deemed robust by the corresponding criterion.*

plot the percentage of instances that have robustness value no less than that amount (dashed lines). For comparison, we plot in the last column the performance against the probability difference for each possible threshold (e.g. for a threshold of zero all instances are robust, and for a threshold of 1 no instance is robust).

We see that for all three approaches the performance increases monotonically with the robustness value, except when the the number of instances is very small (say, < 1% of the testset). Interestingly, the correlation between robustness and performance is positive when using non-local perturbation even for these extreme scenarios.

Although some results suggest a slight advantage of CSPN-based robustness analysis these results are still inconclusive; we conjecture that this can be explained by these datasets not being complex enough to verify a lack of robustness in the (smoothed) MLE estimates of weights

Multilabel Classification

We also performed a robustness analysis of multilabel classification on 9 benchmark datasets. These datasets have been widely used in Multilabel task using Probabilistic Circuits (Di Mauro *et al.*, 2016; Villanueva Llerena and Mauá, 2017) and are available at https://github.com/nicoladimauro/dcsn and http://meka.sourceforge.net.

Table 4.5 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 $LD = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} x^*_{ij}\hat{x}_{ij}}{R}$.

In addition to Exact Match, we also evaluate multilabel classifications by a metric commonly used in the multilabel classification literature (Dembczyński *et al.*, 2012; Di Mauro *et al.*, 2016), which rewards correctly predicted labels while discounting for incorrectly predicted ones. We name this metric *Accuracy* (Acc for short), and compute it as:

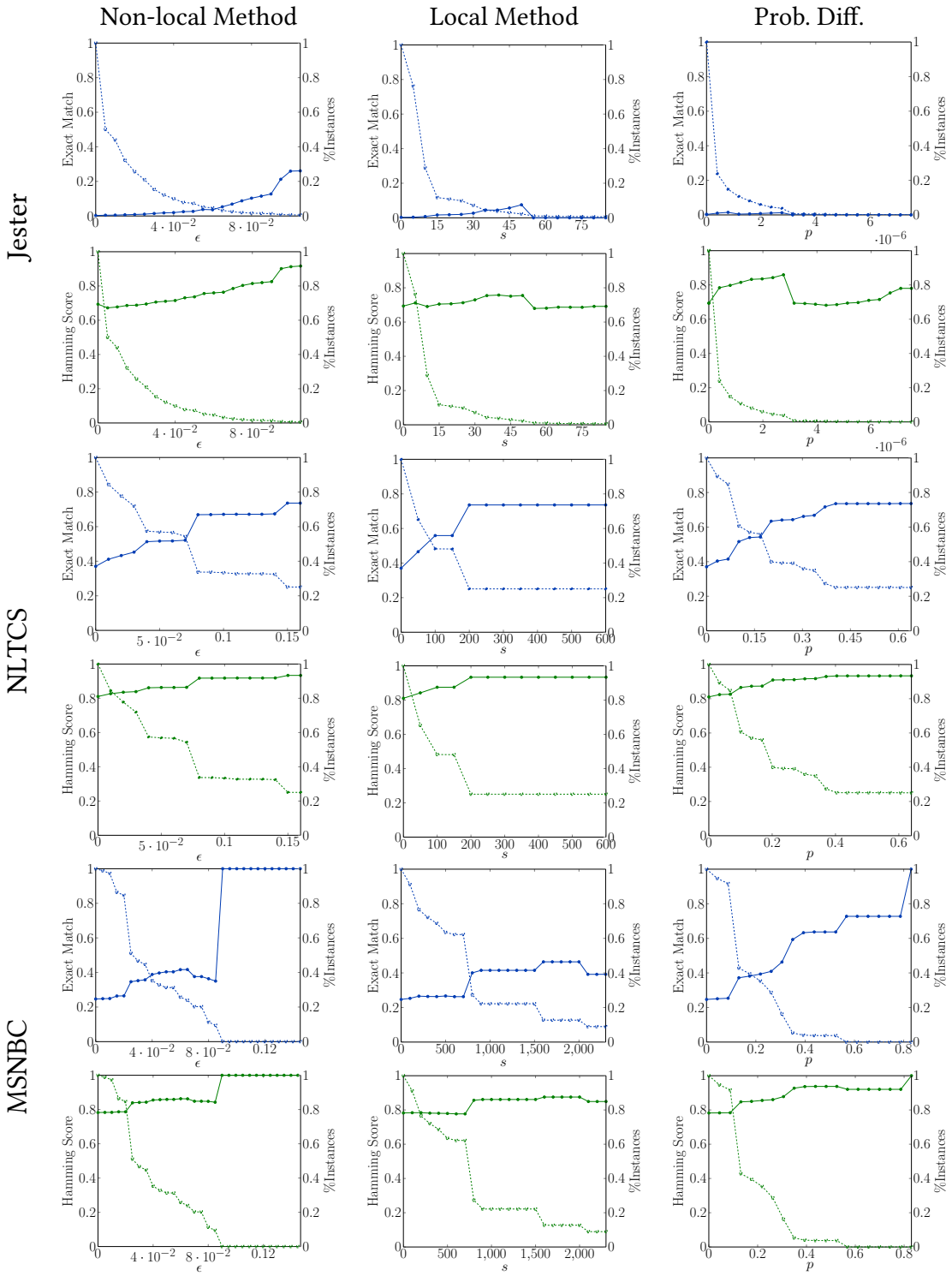


Figure 4.3: *Test-set performance vs. robustness vs. proportion of instances for completion tasks. Each two rows display results for a dataset. The curves in blue show results for Exact Match score, while the curves in green show results for the Hamming score, dashed curves show instance percentage.*

Dataset	#Training	#Test	Evidence Vars	L	LC	LD
Arts	5389	1496	500	26	1.65	0.06
Business	8073	2244	500	30	1.6	0.05
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

Table 4.5: *Multi-Label Datasets, number of training instances, number of test instances (N), number of evidence variables (M), number of labels L, label cardinality (LC) and label density (LD).*

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

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 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 values for ϵ, s and p so that roughly either 10% or 50% of instances are robust. The results for when about 50% of instances are robust are summarized in Table 4.6.

The results show that all three methods perform, on average, very similarly. Some notable exceptions appear for the Emotions, Flags and Health datasets. In the Emotions, Flags and Health datasets, robust instances according to the CSPN-based approach have signficantly higher accuracy that the robust instances according to probability difference, while only slightly higher exact match values. Probability difference outperforms the other methods with respect to accuracy on the Arts dataset, although the three methods fail to distinguish robust inferences. CSPN-based approaches perform at least as well as Probability Difference for almost all of the datasets, with the exception of the Arts dataset.

The results for about 10% of robust instances are shown in Table 4.7. The results are qualitatively the same, although CSPN-based approaches outperform probability difference on most datasets, in some cases with a very significant difference, such as Flags, Health and Scene datasets. CSPNs obtained by non-local perturbation now achieve the best performance on the Arts dataset, while CSPNs obtained by local perturbation still perform poorly, it is important to note that the percentage of instances considered by the former is smaller than by the latter.

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 improves on the ability to detect the most accurate instances, occasionally much more effectively than using the probability estimates of the model.

Dataset	Method	%I	Accuracy			Exact Match		
			Robust	¬*Robust*	Δ*Acc*	*Robust*	¬*Robust*	Δ*EM*
Arts	ϵ = 0.045	51.1	0.1	0.31	-0.21	0.07	0.23	-0.16
	s = 20	56	0.1	0.34	-0.24	0.08	0.24	-0.16
	p = 0.19	53.8	**0.17**	0.24	**-0.07**	**0.15**	0.16	**-0.01**
Business	ϵ = 0.13	63.5	0.76	0.6	0.16	0.62	0.43	0.19
	s = 240	56.6	**0.78**	0.61	**0.17**	**0.64**	0.42	**0.22**
	p = 0.39	57.5	0.77	0.62	0.15	0.63	0.43	0.2
Emotions	ϵ = 0.08	49.6	**0.54**	0.35	**0.19**	**0.24**	0.12	**0.12**
	s = 8	56.3	0.51	0.35	0.16	0.22	0.12	0.1
	p = 0.042	66.4	0.44	0.45	-0.01	0.15	0.23	-0.08
Flags	ϵ = 0.03	48.7	**0.64**	0.4	**0.24**	0.22	0.1	0.12
	s = 1	48.7	**0.64**	0.41	0.23	0.22	0.1	0.12
	p = 0.29	46	0.58	0.47	0.11	0.22	0.1	0.12
Health	ϵ = 0.045	50.8	**0.65**	0.46	**0.15**	**0.54**	0.29	**0.25**
	s = 40	53.2	0.61	0.49	0.12	0.51	0.31	0.2
	p = 0.32	52.9	0.62	0.48	0.14	0.52	0.3	0.22
Human	ϵ = 0.04	92.8	0.22	0.06	**0.16**	0.15	0.04	**0.11**
	s = 35	92.8	0.22	0.06	**0.16**	0.15	0.04	**0.11**
	p = 0.1	83.1	0.22	0.11	0.11	**0.16**	0.08	0.08
Plant	ϵ = 0.05	58.2	0.36	0.12	0.24	0.35	0.12	0.23
	s = 20	55.6	**0.37**	0.13	0.24	0.36	0.13	0.23
	p = 0.3	55.6	**0.37**	0.13	0.24	**0.37**	0.13	**0.24**
Scene	ϵ = 0.2	61.9	0.2	0.5	-0.3	0.2	0.35	-0.15
	s = 45	52.3	**0.25**	0.38	**-0.13**	**0.22**	0.28	**-0.06**
	p = 0.23	55.6	0.21	0.44	**-0.13**	0.21	0.31	-0.1
Yeast	ϵ = 0.007	96.7	0.43	0.18	**0.25**	0.1	0	0.1
	s = 1	97.7	0.43	0.19	0.24	0.1	0	0.1
	p = 0.08	97.3	0.43	0.2	0.23	0.1	0	0.1

Table 4.6: *Performance of multilabel classifications when about 50% of instances are robust.*

Dataset	Method	%I	Accuracy			Exact Match		
			Robust	¬*Robust*	Δ*Acc*	*Robust*	¬*Robust*	Δ*EM*
Arts	ϵ = 0.057	1.3	**0.89**	0.2	**0.69**	**0.84**	0.14	**0.7**
	s = 210	24.9	0	0.27	-0.27	0	0.2	-0.2
	p = 0.23	10.2	0.74	0.14	0.6	0.66	0.09	0.57
Business	ϵ = 0.13	63.5	0.76	0.6	**0.16**	0.62	0.43	**0.19**
	s = 520	43.2	**0.78**	0.65	0.13	**0.64**	0.48	0.16
	p = 0.41	46.6	0.77	0.65	0.12	0.63	0.47	0.16
Emotions	ϵ = 0.13	16	0.64	0.41	0.23	0.26	0.16	0.1
	s = 30	10.9	**0.69**	0.41	**0.28**	**0.31**	0.16	**0.15**
	p = 0.27	22.7	0.6	0.4	0.2	0.19	0.17	0.02
Flags	ϵ = 0.06	10.5	**0.92**	0.47	**0.45**	**0.5**	0.12	**0.38**
	s = 1	10.5	**0.92**	0.47	**0.45**	**0.5**	0.12	**0.38**
	p = 0.51	15.8	0.72	0.48	0.24	0.33	0.13	0.2
Health	ϵ = 0.05	6.2	**0.73**	0.55	**0.18**	**0.56**	0.41	0.15
	s = 160	38	0.66	0.5	0.11	**0.56**	0.33	**0.23**
	p = 0.34	48.1	0.64	0.48	0.16	0.53	0.31	0.22
Human	ϵ = 0.04	92.8	**0.22**	0.06	**0.16**	0.15	0.04	0.11
	s = 35	92.8	**0.22**	0.06	**0.16**	0.15	0.04	0.11
	p = 0.11	42.6	0.21	0.2	0.01	**0.16**	0.14	0.02
Plant	ϵ = 0.1	25	0.24	0.27	-0.03	0.22	0.27	-0.05
	s = 36	23.5	**0.25**	0.26	**-0.01**	0.24	0.26	-0.02
	p = 0.55	23.5	**0.25**	0.26	**-0.01**	0.24	0.26	-0.02
Scene	ϵ = 0.3	10	**0.75**	0.26	**0.49**	**0.75**	0.19	**0.56**
	s = 150	13.3	0.2	0.33	-0.13	0.2	0.26	-0.06
	p = 0.35	24.2	0.3	0.3	0	0.3	0.23	0.07
Yeast	ϵ = 0.11	5.8	**0.44**	0.42	**0.02**	0.07	0.1	-0.03
	s = 5	6.2	0.42	0.42	0	0.07	0.1	-0.03
	p0.085	1.1	0.37	0.42	-0.05	**0.2**	0.1	**0.1**

Table 4.7: *Performance of multilabel classifications when about 10% of instances are robust.*

4.4.2 Estimating prediction robustness on Credal Sentential Decision Diagrams

As an application for MAP robustness on CSDDs, we consider the simple task of identification of digits depicted by a seven-segment display, whose segments might fail to turn on given a failure probability p_f described in Example 2.3.3 in Chapter 3. Our problem is modeled by fourteen Boolean variables: $\boldsymbol{X} = (X_1, \ldots, X_7)$ and $\boldsymbol{O} = (O_1, \ldots, O_7)$, where the former refer to the states of the segments as decided by the control unit, and the latter are the real states of the segments as depicted in the display. The true state of these Boolean variables corresponds to the segment on, digits configuration $\boldsymbol{X}$ is provided by disjunctive formula

We use the algorithm proposed in (A. Choi and Darwiche, 2013) to build an SDD α normalized for a vtree such that, for each $i = 1, \ldots, 7$, the pair (X_i, O_i) corresponds to a pair of leaves with the same parent and with a so-called *balanced* shape. The resulting SDD

has a multiply connected structure, 128 nodes (82 of them decision nodes) and maximum number of elements for decision node equal to eight.

Given a training data set $\mathbf{d}$ of size N, generated according to the above described procedure, we can obtain from α a PSDD or a CSDD. In the second case, we use non-local perturbation method with the sample size $s = 1$ to learn the CSDD.

We pWe compare the performance of robust and non-robust MAP instances. In practice, we compute the MAP configuration of $\boldsymbol{X} = \boldsymbol{x}^*$ given $\boldsymbol{o}$ in the PSDD and use Algorithm 4.3 to check whether or not the configuration was robust. The corresponding accuracies of robust and non robust intances are reported in Figure 4.4 only for $N \geq 20$ as for lower training set size the amount of detected digits is very low in both cases. As expected, the CSDD is properly able to distinguish these two sets and keeps a level of accuracy very close to one even for high perturbation levels (the value of p_f only affects the percentage of robust instances).

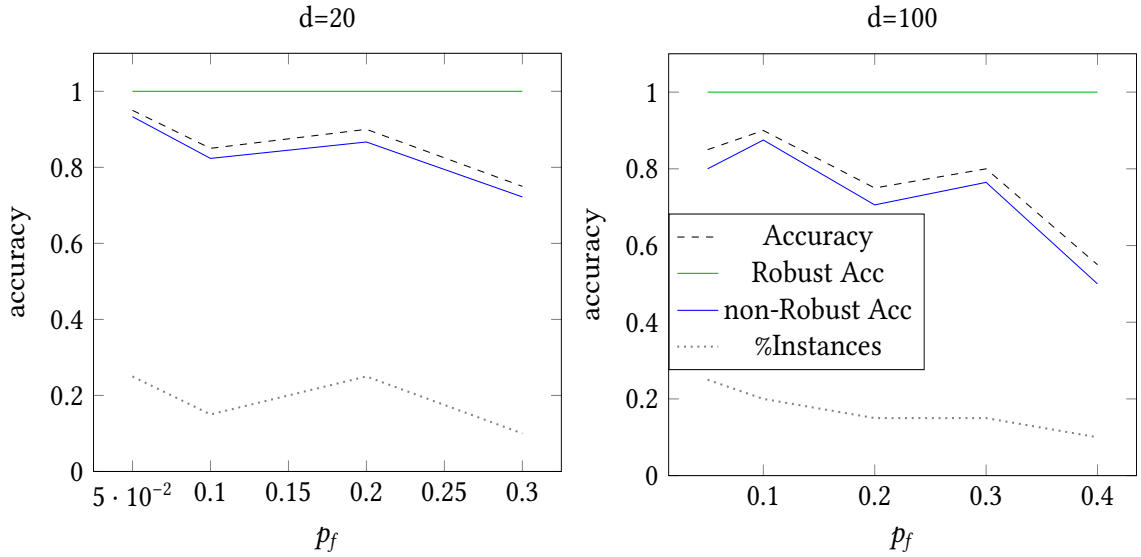


Figure 4.4: *PSDD robust vs. non-robust MAP instantions accuracies*

Chapter 5

Tractable Qualititative Sensitivity Analysis of Inference under Missing Not At Random Data

This chapter concerns prediction of data when some of the values are not observed. For example, respondents in a household survey may refuse to report income; in an industrial experiment some measures for sensors are missing because of mechanical breakdowns unrelated to the experiment. Commonly, the lack of response is essentially an additional point in the sample space of the variable being measured. An interesting example of data where the missing data mechanism is relevant is recommender systems. Recommender systems play an essential role in such highly rated sites as Amazon.com, YouTube and Netflix, most part of these platforms allow users to rate the items at their discretion, thus if a user does not rate a particular item, it also provides some relevant information about the user preferences. The data used to build recommendation systems is often a matrix with one row per user and one column per item, Figure 5.1 shows a toy example of a tabular dataset of user preferences for books with some missing preferences. Collaborative filtering methods use probabilistic models to predict the unseen ratings using the collaborations of user ratings, and then recommend items with the higher predicted ratings (Murphy, 2020; Marlin, Zemel, Sam, *et al.*, 2007).

Figure 5.1: *User preferences for books, an example of tabular data with missing values*

5.1 Missing Data

We formalize the problem of missing data as follows. Let $\mathbf{x}$ be a dataset with missing and observable values $\mathbf{x} = \{\mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}\}$, where x_j^i represents the value of variable X_j in the instance i, and $x_{j,v}$ is an indicator variable that represents that X_j takes on value v. Let $\mathbf{r}$ be a matrix of response-data indicators where $r_j^i = 1$ if x_j^i is observed and 0 otherwise. The joint distribution over the data and response indicators is given by:

$$\mathbb{P}(\mathbf{X}, \mathbf{R}) = \mathbb{P}_\theta(\mathbf{X})\mathbb{P}_\mu(\mathbf{R}|\mathbf{X}), \tag{5.1}$$

where $\mathbb{P}_\theta(\mathbf{X})$ is the *data model* (or sampling model), $\mathbb{P}_\mu(\mathbf{R}|\mathbf{X})$ is the *selection model* or *response model*, and θ and μ are parameters that specify the models.

Data is missing completely at random (MCAR) when the missing-data mechanism does not depend on any variable. In the case of missingness obtained at random (MAR), the missing data mechanism depend only on the observed values (Rubin, 1976), that is,

$$\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}) = \mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}). \tag{5.2}$$

We say that data is missing not at-random (MNAR) if the distribution of missing data mechanism does depend on observed and missing values, also called non-ignorable missing data (Rubin, 1976), that is, if:

$$\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}) \neq \mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}, \mathbf{x}'), \text{ for any } \mathbf{X}' \subsetneq \mathbf{X}^{\text{miss}}. \tag{5.3}$$

where $\mathbf{X}' \subsetneq \mathbf{X}^{\text{miss}}$ represent a subset of the missing variables.

Several methods have been proposed to deal with missing data most of then supported by the MAR assumption, such value imputation and marginalization (Azur *et al.*, 2011; Correia *et al.*, 2020). A notable approach to dealing with MAR missing data for Decision Trees at learning and prediction time is to compute expected predictions using PCs, taking advantage of the tractable computation of marginals (Khosravi, Vergari, *et al.*, 2020). In some realistic applications, such as recommender systems, the MAR hypothesis is violated. Therefore, using imputation or marginalization can lead to unreliable predictions.

In the rest of this chapter, we propose efficient methods for qualitative sensitivity analysis of inference under missing not at random data. If the MNAR mechanism is known at both training and prediction time, we can learn a single probabilistic circuit to represent both the model and the response model. In real-world problems, we rarely have access to the missingness mechanism. If the MNAR data is only available at prediction time, we can quantify the effect of different imputations of the missing values to decide which values are the most likely classification under some imputation. Alternatively, we could use a curated (but very small) dataset to estimate an imprecise response model, or rely on expert knowledge.

5.2 Prediction in the Presence of Missing Data

Suppose we are interested in using a probabilistic circuit, learned from complete training data (i.e., with no missing values), to predict the most likely value of a target variable $Y \in \mathbf{X}^{\text{miss}}$ given a configuration $\mathbf{x}^{\text{obs}}$ of some of the other variables. Assuming the unobserved values (including Y) are missing at random (MAR) we have:

$$\arg\max_y \mathbb{P}(y|\mathbf{x}^{\text{obs}}, \mathbf{r}) = \arg\max_y \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\theta(y, \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}\mathbf{x}^{\text{miss}}, y) \tag{5.4}$$

$$= \arg\max_y \mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}) \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\theta(y, \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) \tag{5.5}$$

$$= \arg\max_y \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\theta(y, \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}). \tag{5.6}$$

The last equation specifies a univariate MAP inference computation in a PC $\mathbb{S}$ that specifies the data model:

$$\arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}}) = \arg\max_y \mathbb{S}(y, \mathbf{x}^{\text{obs}}) \tag{5.7}$$

where $\mathbb{S}(y, \mathbf{x}^{\text{obs}}) = \sum_{\mathbf{x}^{\text{miss}}} \mathbb{S}(y, \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})$ is the marginal value of the circuit at y and $\mathbf{x}^{\text{obs}}$, which can be obtained in linear time as discussed.

When the missingness is not ignorable (MNAR) the inference in (5.7) can lead to erroneous and unreliable conclusions. For example, asssume that $\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}) = \prod \mathbb{P}_\mu(\mathbf{r}_i|\mathbf{x}_i)$. Then:

$$\arg\max_y \mathbb{P}(y|\mathbf{x}^{\text{obs}}, \mathbf{r}) = \arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}})\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}, y) \tag{5.8}$$

$$= \arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}}) \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\mu(\mathbf{r}|\mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}, y)\mathbb{P}_\theta(\mathbf{x}^{\text{miss}}|\mathbf{x}^{\text{obs}}, y) \tag{5.9}$$

$$= \arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}})\mathbb{P}_\mu(\mathbf{r}^{\text{obs}}|\mathbf{x}^{\text{obs}}) \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\mu(\mathbf{r}^{\text{miss}}|\mathbf{x}^{\text{miss}})\mathbb{P}_\theta(\mathbf{x}^{\text{miss}}|\mathbf{x}^{\text{obs}}, y) \tag{5.10}$$

$$= \arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}}) \sum_{\mathbf{x}^{\text{miss}}} \mathbb{P}_\mu(\mathbf{r}^{\text{miss}}|\mathbf{x}^{\text{miss}})\mathbb{P}_\theta(\mathbf{x}^{\text{miss}}|\mathbf{x}^{\text{obs}}, y) \tag{5.11}$$

$$= \arg\max_y \mathbb{P}_\theta(y|\mathbf{x}^{\text{obs}})\mathbb{P}(\mathbf{r}^{\text{miss}}|\mathbf{x}^{\text{obs}}, y). \tag{5.12}$$

where the sums are over the configurations $\mathbf{x}^{\text{miss}}$ that are consistent with y.

The difference between Equation 5.12 and 5.7 is the bias introduced by treating MNAR data as MAR.

5.3 Augmented Network with Response Model

As we discussed before, to perform inference with non-ignorable missing data, in addition to learn the data model to represent the relations between variables, it is important to determinate the response model to represent the missingness mechanism. The *response model* estimates the model over response indicators $\mathbf{R}$, which refers to each indicator

variable (Little and Rubin, 2014).

To perform prediction in the presence of MNAR data, we propose encoding the response model into the structure of a PC learned from complete training data. This training data must have the same distribution as the non-ignorable missing data. For example, popular sources of complete data for recommendation systems are user surveys.

Furthermore, we assume that the absence of a variable depends exclusively on its own value, $\mathbb{P}_\mu(\mathbf{r}|\mathbf{x}) = \prod_i \mathbb{P}_{\mu_i}(r_i|x_i)$. Note that this is true for MCAR data, as in the case $\mathbb{P}(r^i|x^i) = \mathbb{P}(r^i)$.

Starting with a fixed PC structure and parameters learned from complete data, we then incorporate sub-structures (PCs) to represent the response model. Replacing each leave indicator node $[\![X_j = v]\!]$ by a circuit rooted with a product node with two children, the first one is the original indicator $[\![X_j = v]\!]$ and the last one is a sum node with response indicators $[\![R_{j,v} = 0]\!]$ and $[\![R_{j,v} = 1]\!]$ as children, which weights $\mu_{j,v}$ and $1 - \mu_{j,v}$. Figure 5.2b shows the augmentation of the data model structure, shown in Figre 5.2a, by the response model of variable X.

Suppose we have available both a large MNAR sample $\mathbb{P}^{\text{MNAR}}$ and a small MAR sample $\mathbb{P}^{\text{MAR}}$, from which we want to estimate:

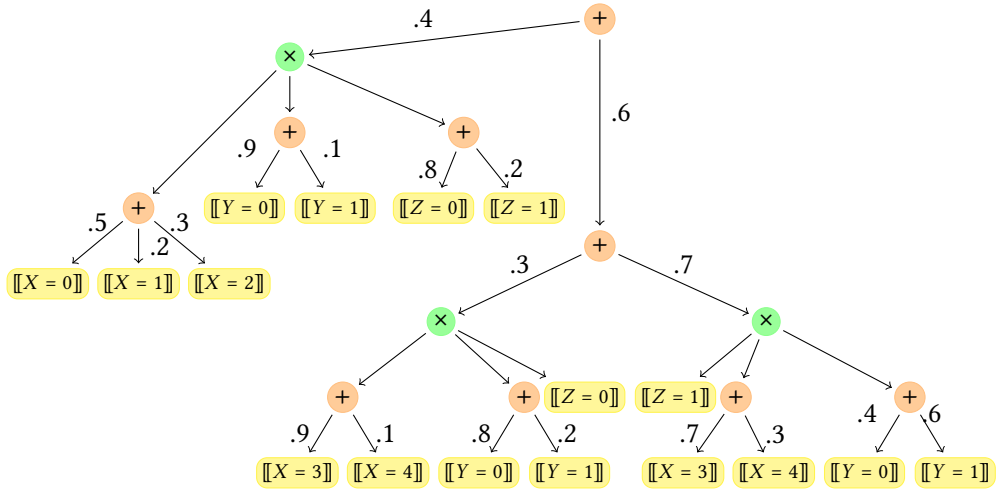
$$\mu_{j,v}\mathbb{P}_{\mu_j}(R_{j,v} = 1|X_j = v) = \frac{\mathbb{P}^{\text{MNAR}}(X_j = v, R_{j,v} = 1)}{\mathbb{P}^{\text{MAR}}(X_j = v)}. \tag{5.13}$$

The probabilities in the numerator can be estimated from the MNAR sample as relative frequencies as they mention only observable quantities, that is, $\mathbb{P}(X_j = v, R_{j,v} = 1) = \mathbb{P}^{\text{MNAR}}(X_j = v, R_{j,v} = 1)$. The denominator however includes cases where the value of X_j is missing not at random (i.e., $R_{j,v} = 0$) and thus a relative frequency estimator is biased and inconsistent. We can however compute it from the MAR sample (MCAR):

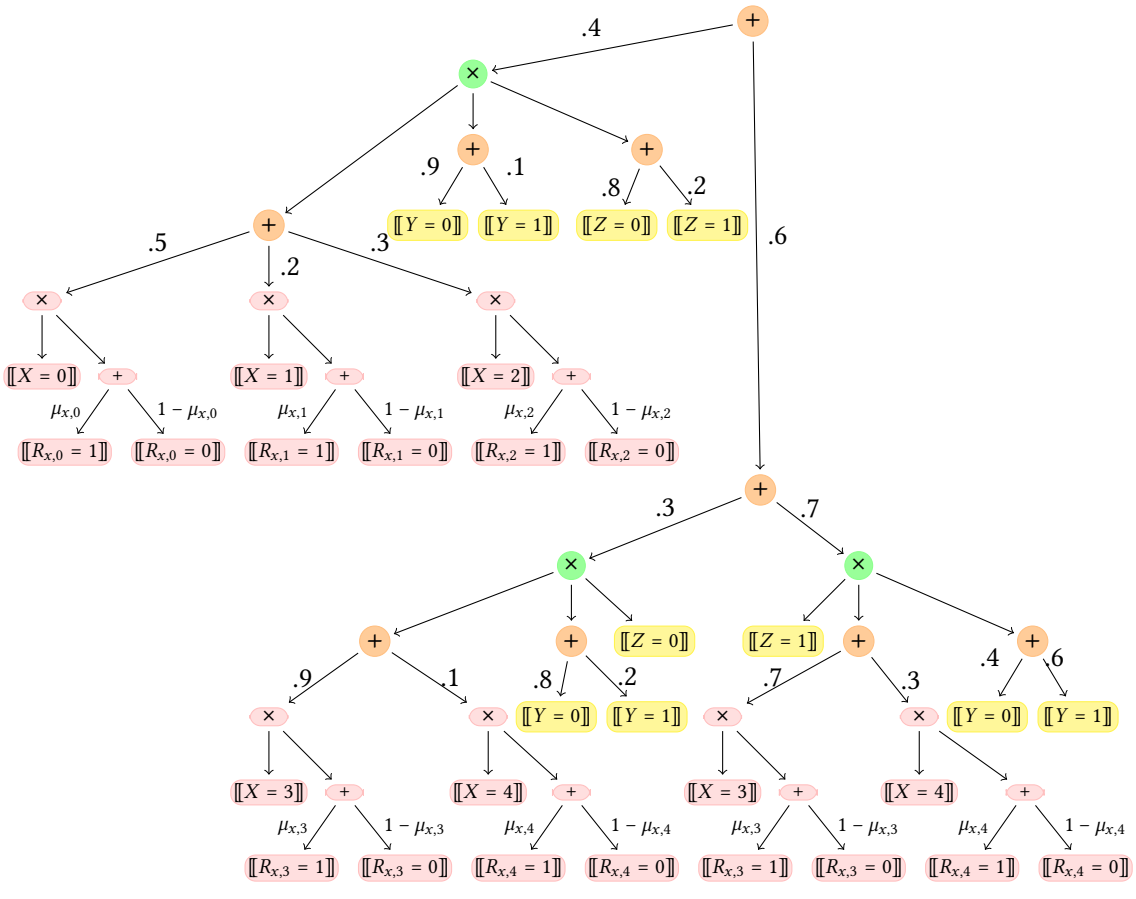
$$\mathbb{P}(X_j = v) = \mathbb{P}^{\text{MAR}}(X_j = v) = \mathbb{P}^{\text{MAR}}(X_j = v|R_{j,v} = 1)\,,$$

We can perform inference in the augmented model by marginalizing missing variables at data indicators and setting the corresponding values in response indicators to deal with the missing process, the indicator $[\![R_{j,v} = 1]\!]$ propagates 1 if $X_j = v$ is observable and takes value v, and $[\![R_{j,v} = 0]\!]$ for all $v \in val(X_j)$ propagates 1 if the value of X_j is missing.

Example 5.3.1. *Consider the model in Figure 5.2a as a PC learned from complete user ratings for three items. We want to predict the rating item Y, that can take two posible values, say we observe $Z = 0$ and X is missing due to the following MNAR process: $\mathbb{P}(R_{x,0} = 1|X = 1) = 0.1$, $\mathbb{P}(R_{x,1} = 1|X = 1) = 0.3$, $\mathbb{P}(R_{x,2} = 1|X = 2) = 0.5$, $\mathbb{P}(R_{x,3} = 1|X = 3) = 0.8$ and $\mathbb{P}(R_{x,4} = 1|X = 4) = 0.9$. We then encode the MNAR process using an augmented model, as shown in Figure 5.2b. We use this model to evaluate the two values of Y and classify the most likely one. Figure 5.3a shows the evaluation of the evidence $Z = 0$ for the evaluation $Y = 0$, that is 0.232, while if we evaluate the model for $Y = 1$ the value of the root is 0.0587, then the prediction obtained by our model is $Y = 0$.*



(a)



(b)

Figure 5.2: *Augmentation of the SPN in Figure 5.2a with response model discussed in the text (pink nodes). Figure 5.2b show augmentation for Response model representing the missing process of variable X. In this example the values for Y and Z can be complete or missing according to MCAR mechanisms.*

Now consider another possible scenario where X is generated by an MCAR mechanism instead of an MNAR mechanism, then we know that the missingness is completely at random and the each parameter for the response model for X takes the value 0.5, $\mu_{x,v} = 0.5$. Figure 5.3b shows the evaluation of the evidence, $Z = 0$, and the possible prediction $Y = 0$, which is 0.216; while the same procedure for $Y = 1$ gives 0.034, then the prediction obtained by this model is also $Y = 0$.

Although the two augmented models give the same prediction for the rating of item Y, the values obtained from the respective evaluations are different and can produce different predictions due to the response model, as we described in the earlier discussion in section 5.1. Note that using the augmented model for an MCAR missingness mechanism yields the same result as not augmenting the model with the response model.

5.3.1 Tractable Conservative Inference

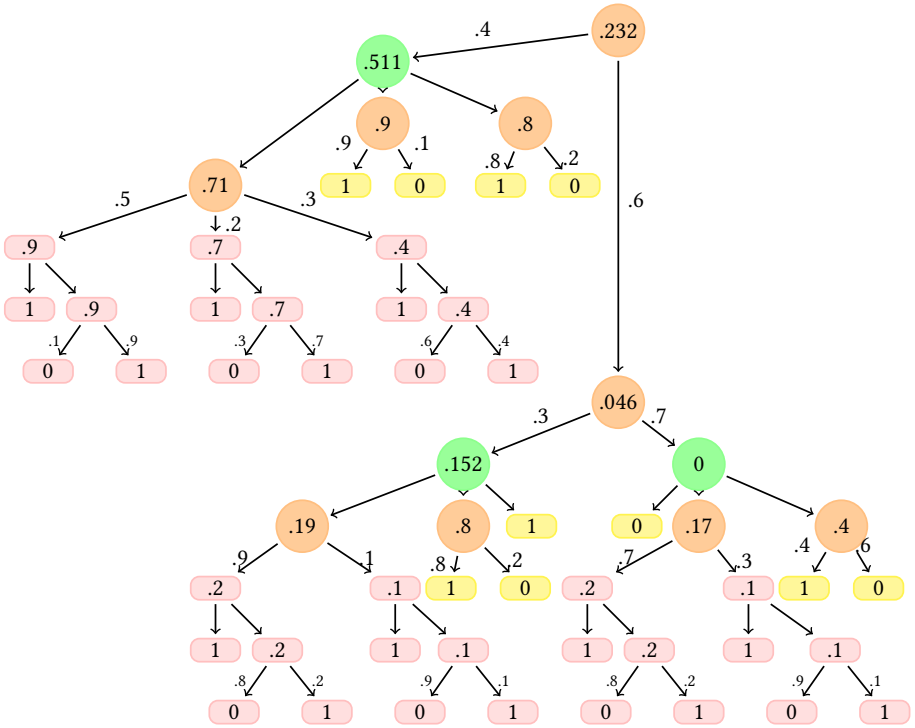
Tractable Conservative Inference (TCI) is based on *conservative inference rule* (CIR) proposes avoiding that potential bias by considering completions incomparable, leading to set-valued inferences (Zaffalon and Miranda, 2009). It is an abstract rule to update beliefs with non-ignorable missing data that can be applied in any situation or domain drawing reliable (but less informative) conclusions. CIR has been applied to traditional probabilistic models such as Bayesian networks (Antonucci and Piatti, 2009), where it suffers from intractability of inference. In fact, CIR inference in Bayesian networks has been proved to be equivalent to marginal inference in credal networks (Antonucci and Zaffalon, 2008), a task whose theoretical and practical complexity far exceeds that of marginal inference in Bayesian networks (Mauá, De Campos, *et al.*, 2014).

In this section, we analyze the reliable criteria of TCI and the tractable benefits of probabilistic circuits at prediction time, and GeFs in particular. To avoid erroneous assumptions, instead of computing marginal inference, Equation (5.7), we propose to estimate the robustness of a classification $Y = y'$ under non-ignorable missing data with respect to an alternative classification $Y = y''$ by

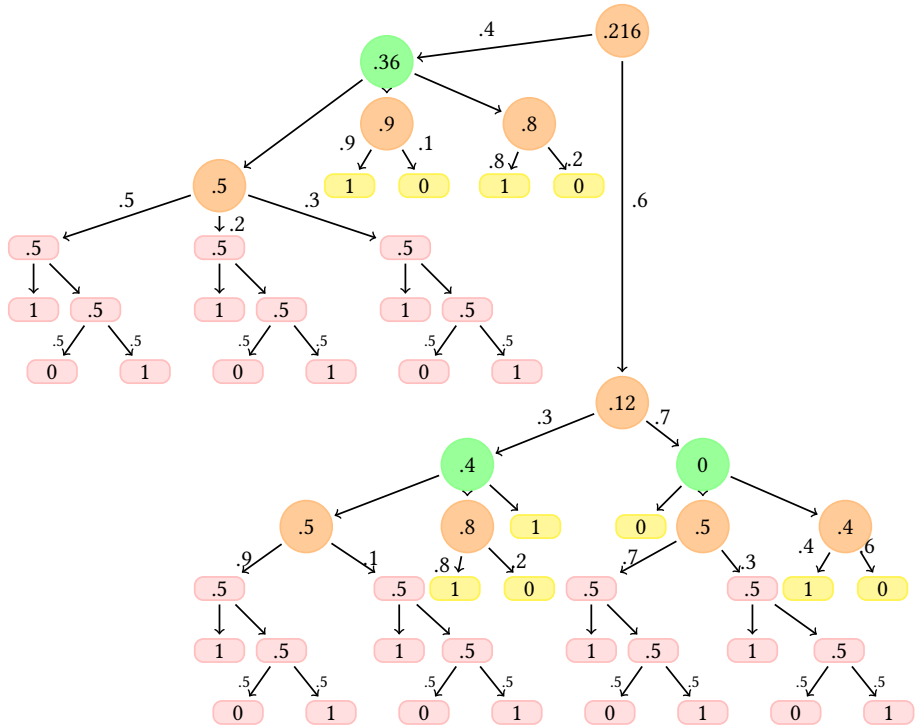
$$\begin{aligned}\min_{\mu}[\mathbb{P}(y', \mathbf{x}^{\text{obs}}, \mathbf{r}) - \mathbb{P}(y'', \mathbf{x}^{\mathbf{obs}}, \mathbf{r})] &= \min_{\mu} \sum_{\mathbf{x}^{\text{miss}}} [\mathbb{P}(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})\mathbb{P}_{\mu}(\mathbf{r}|\mathbf{x}^{\text{obs}}\mathbf{x}^{\text{miss}}, y') - \\ &\qquad \mathbb{P}(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})\mathbb{P}_{\mu}(\mathbf{r}|\mathbf{x}^{\text{obs}}\mathbf{x}^{\text{miss}}, y'')] \qquad (5.14)\\ &= \min_{\mu} \sum_{\mathbf{x}^{\text{miss}}} [\mathbb{P}(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})\mathbb{P}_{\mu}(r^{y'}|y') - \\ &\qquad \mathbb{P}(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})\mathbb{P}_{\mu}(r^{y''}|y'')] \qquad (5.15)\\ &\qquad \mathbb{P}_{\mu}(\mathbf{r}^{\text{miss}}|\mathbf{x}^{\text{miss}})\end{aligned}$$

When we assume a vacuous response model, the previous equation reduces as follows:

$$\begin{aligned}\min_{\mu}[\mathbb{P}(y', \mathbf{x}^{\text{obs}}, \mathbf{r}) - \mathbb{P}(y'', \mathbf{x}^{\mathbf{obs}}, \mathbf{r})] &= \min_{\mathbf{x}^{\text{miss}}}[\mathbb{P}(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) - \mathbb{P}(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})] \qquad (5.16)\\ &= \delta_{\mathbf{S},\mathbf{x}^{\text{obs}}}(y', y'') \qquad (5.17)\end{aligned}$$



(a)



(b)

Figure 5.3: *Two examples of evaluation of two Augmented models, Figure 5.3a given a MNAR missing mechanism with parameters:* $\mu_{x,0} = 0.1$, $\mu_{x,1} = 0.3$, $\mu_{x,2} = 0.6$, $\mu_{x,3} = 0.8$ *and* $\mu_{x,4} = 0.9$, *Figure 5.3b for a MCAR missing mechanism, that is* $\mu_{x,v} = 0.5$.

A decision analyst might want to suspend the classification on the basis of the value in (5.16), thus producing more conservative conclusions.

For example, if $\delta_{\mathrm{S},\mathbf{x}^{\mathrm{obs}}}(y', y'') > 0$, then any imputation of the values of $\mathbf{x}^{\mathrm{miss}}$ still leads to a classification y' that is more probable (as far as the model estimates) than y''; we thus say that y' dominates y''.

The conservative inference rule prescribes that the only conclusion supported by non-ignorable missing data is to return the set of non-dominated values (Zaffalon and Miranda, 2009):

$$\left\{ y \,:\, \max_{y'} \delta_{\mathrm{S},\mathbf{x}^{\mathrm{obs}}}(y', y) \leq 0 \right\} . \tag{5.18}$$

This is akin to classification with a rejection option, but possibly more informative (and arguably more principled).

Even though evaluation takes linear time in PCs, a brute-force approach to computing Equation (5.16) requires evaluating $\mathtt{S}(y, \mathbf{x}^{\mathrm{obs}}, \mathbf{x}^{\mathrm{miss}})$ for each $\mathbf{x}^{\mathrm{miss}}$. This is unfeasible when the number of possible completions is high. The next result shows that computing such a value is coNP-hard even in selective PCs, ruling out the existence of an efficient exact procedure (under common assumptions of complexity theory).

Theorem 5.3.1. *Given a smooth, decomposable and selective PC* $\mathtt{S}$ *over random variables* Y, $\mathbf{X}^{obs}$ *and* $\mathbf{X}^{miss}$, *target values* y' *and* y'', *a partial observation* $\mathbf{x}^{obs}$, *and a (rational) threshold* ρ, *deciding if* $\delta_{\mathrm{S},\mathbf{x}^{obs}}(y', y'') > \rho$ *is coNP-complete.*

Proof. Membership in coNP is trivial: given a configuration $\mathbf{x}^{\mathrm{miss}}$ we can compute $\mathtt{S}(y', \mathbf{x}^{\mathrm{obs}}, \mathbf{x}^{\mathrm{miss}})$ and $\mathtt{S}(y'', \mathbf{x}^{\mathrm{obs}}, \mathbf{x}^{\mathrm{miss}})$ in linear time and decide the sign of its difference in constant time. Hence we have a polynomial certificate that the problem is *not* in the language.

We show hardness by reduction from the *subset sum problem*: Given positive integers $z_1, \ldots, z_n$, decide:

$$\exists u \in \{0,1\}^n \,:\, \sum_{i\in[n]} v_i u_i = 1\,, \quad \text{where } v_i = \frac{2z_i}{\sum_{i\in[n]} z_i}\,. \tag{5.19}$$

To solve that problem, build a tree-shaped selective PC as shown in Figure 5.4. Note that the PC is **not strong selective w.r.t.** Y, where U_i are binary variables, $P_1(u) = \frac{e^{-2\sum_i v_i u_i}}{\prod_i (1+e^{-2v_i})}$ and $P_2(u) = \frac{e^{-\sum_i v_i u_i}}{\prod_i (1+e^{-v_i})}$.

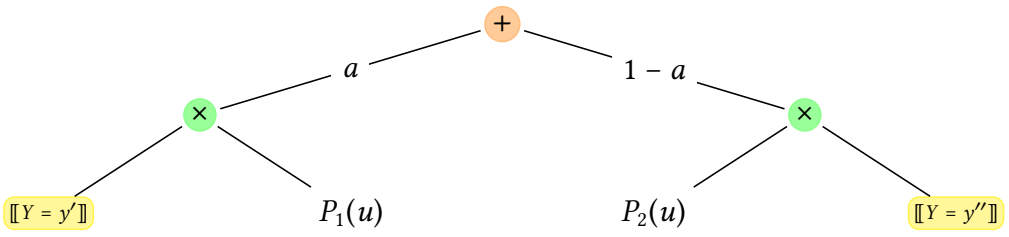


Figure 5.4: *Sketch of the PC gadget used to prove coNP-hardness of dominance.*

Use the PC to compute:

$$\delta(y', y'') = \min_u \left[\frac{a}{t_1} \exp\left(-2 \sum_i v_i u_i \right) - \frac{(1-a)}{t_2} \exp\left(- \sum_i v_i u_i \right) \right] .$$

where $t_1 = \prod_i (1 + \exp(-2v_i))$ and $t_2 = \prod_i (1 + \exp(-v_i))$.

If we call $x := \exp(-\sum_i v_i u_i)$, the above expression is the minimum for positive x of $f(x) := \frac{ax^2}{t_1} - \frac{(1-a)x}{t_2}$. Function f is a strictly convex function minimized at $x = \frac{t_1(1-a)}{t_2 2a}$. Selecting a such that $t_1(1-a)/(2at_2) = e^{-1}$ makes the minimum occur at $\sum_i v_i u_i = 1$. Thus, there is a solution to (5.19) if and only if $\delta(y', y'') \leq \frac{e^{-2}}{2t_2 e^{-2} + t_1}$. This proof is not quite valid because the distributions $P_1(u)$ and $P_2(u)$ use non-rational numbers. However, we can use the same strategy as used to prove Theorem 5 in (Mauá, Conaty, *et al.*, 2018) and exploit the rational gap between yes and no instances of the original problem to encode a rational approximation of P_1 and P_2 of polynomial size.

□

We now provide a linear-time algorithm for computing $\delta^r(y', y'')$ in tree-shaped strong selective PCs, which include class-factorized GeDTs. For the sake of readability, we drop the dependence on $\mathbf{x}^{\text{obs}}$ in the following. The algorithm can be described straightforwardly by a collection of recursive equations depending on the type of node at which it operates. The recursive formulation also provides a proof of its correctness under the above assumptions.

If node i is a **sum node** , then the algorithm computes:

$$\delta^i(y', y'') = \min_{j \in \text{ch}(i)} w_{ij} \min_{\mathbf{x}^{\text{miss}}} [\mathtt{S}_j(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) - \mathtt{S}_j(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})] = \min_{j \in \text{ch}(i)} w_{ij} \delta^j(y', y'') . \quad (5.20)$$

The correctness of the operation follows from the strong-selectivity of the circuit. This property ensures that for any realization $(y, \mathbf{x})$ at most one sub-PC $\mathtt{S}_j$ evaluates to a nonnegative value $\mathtt{S}_j(y, \mathbf{x}) > 0$, and also ensures that either $\mathtt{S}$ encodes a distribution over Y (i.e., its scope is the singleton $\{Y\}$) or the nonnegative child for $\mathtt{S}_j(y', \mathbf{x})$ and $\mathtt{S}_j(y'', \mathbf{x})$ is the same. Note that the above expression is negative whenever there is a child network $\mathtt{S}_j$ with $\mathtt{S}_j(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) = 0$ and $\mathtt{S}_j(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) > 0$ for some $\mathbf{x}^{\text{miss}}$.

If instead i is a **product node** such that Y is in the scope of $\mathtt{S}_k$ child (and no other), then the algorithm computes:

$$\delta^i(y', y'') = \delta^k(y', y'') \prod_{j \in \text{ch}(i), j \neq k} \text{opt}_{\mathbf{x}_j^{\text{miss}}} \mathtt{S}_j(\mathbf{x}_j^{\text{obs}}, \mathbf{x}_j^{\text{miss}}) , \quad (5.21)$$

where $\mathbf{x}_j^{\text{obs}}$ (resp., $\mathbf{x}_j^{\text{miss}}$) denotes the projection of $\mathbf{x}^{\text{obs}}$ (resp., $\mathbf{x}^{\text{miss}}$) into the scope of $\mathtt{S}_j$, and

$$\text{opt} = \begin{cases} \max & \text{if } \delta(y', y'') > 0 , \\ \min & \text{if } \delta(y', y'') \leq 0 . \end{cases}$$

The first term denotes the recursive computation on the sub-PC $\mathtt{S}_k$. The remaining terms $\text{opt}_{\mathbf{x}_j^{\text{miss}}} \mathtt{S}_j(\mathbf{x}_j^{\text{obs}}, \mathbf{x}_j^{\text{miss}})$ define an optimization of the configurations $\mathbf{x}_j^{\text{miss}}$ for the sub-PC $\mathtt{S}_j$;

this can be performed in linear time in selective PCs by bottom-up traversal, replacing sums with maximizations/minimizations (Peharz, Gens, and Domingos, 2014; Peharz, Gens, Pernkopf, *et al.*, 2017).

Finally, if i is a **leaf node** representing an indicator variable then the algorithm computes:

$$\delta^i(y', y'') = \begin{cases} 1 & \text{if } i \text{ is } [\![Y = y']\!], \\ -1 & \text{if } i \text{ is } [\![Y = y'']\!], \\ 1 & \text{if } i \text{ is consistent with } \mathbf{x}^{\text{obs}} \text{ or } \mathbf{x}^{\text{miss}}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.22}$$

We thus obtain the following result.

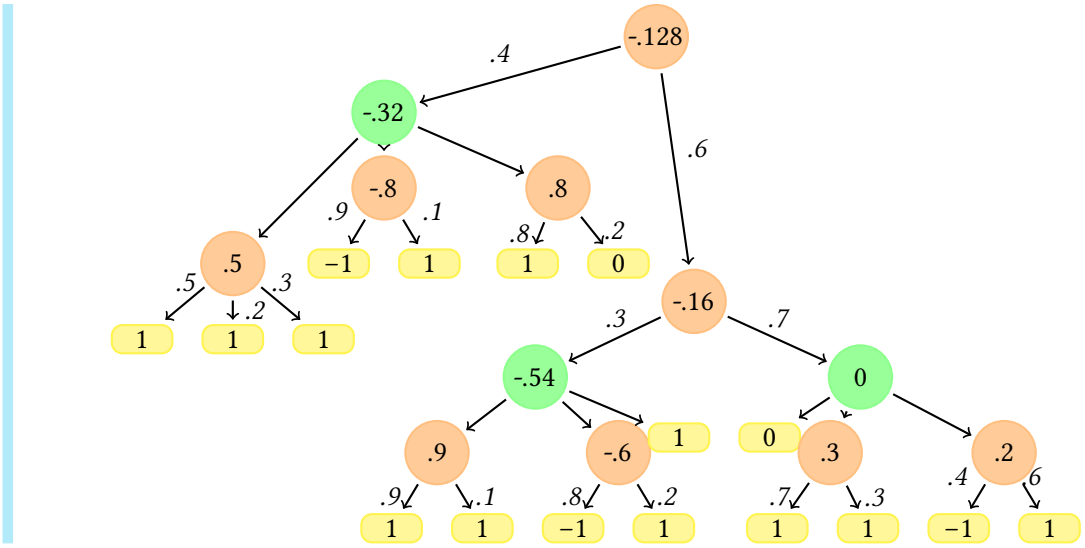
Theorem 5.3.2. *The algorithm obtained by Equations* (5.20), (5.21) *and* (5.22) *computes* $\delta_{\mathrm{S},\mathbf{x}^{obs}}(y', y'')$ *in strong selective tree-shaped PCs in linear time.*

For non-strong-selective networks, the equation for sum nodes is no longer valid, as in such circuits

$$\min_{\mathbf{x}^{\text{miss}}} \sum_{j \in \mathrm{ch}(i)} w_{ij}[\mathrm{S}_j(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) - \mathrm{S}_j(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})] \neq \min_{\mathbf{x}^{\text{miss}}} w_{ij} \min_{j \in \mathrm{ch}(i)} [\mathrm{S}_j(y', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}}) - \mathrm{S}_j(y'', \mathbf{x}^{\text{obs}}, \mathbf{x}^{\text{miss}})] .$$

The equations for products and leaves remain valid for non-selective circuits. Thus, we can use our algorithm as an effective heuristic for non-selective PCs or that violate strong-selectivity. This is the case for instance when we have a *partially ignorable missingness process* and we marginalize part of the missing variables by judging their missingness to satisfy MAR. Then, even for selective PCs, the algorithm described is not guaranteed to provide the correct outcome if some variables are marginalized. Yet our experiments show that it provides an effective heuristic, supporting the reasoning above.

Example 5.3.2. *Consider the model in Figure 5.2a, which is strongly selective with respect of* Y*. Given the observation* $Z = 0$ *and a missing value for* X *generated by an unknown MNAR process. We want to predict the value of* Y*, for that we can use TCI to obtain the set of non-dominated classes. As the result of the TCI algorithm we can predict* $Y = \{0\}$*, since class* 0 *is the single non-dominated value, the next figure shows the evaluation of* $\delta_{\mathrm{S}}(1, 0) = -0.128 < 0$*, similarly we also can obtain* $\delta_{\mathrm{S}}(0, 1) = 0.0216 > 0$.



5.4 Credal Response Model

Consider a PC $\mathtt{S}(\mathbf{X}, Y)$, possibly learned from some (complete or MAR incomplete) dataset of realizations of variables $\mathbf{X}$, Y, and representing our data model. Suppose we are interested in using our model to predict the value of a target variable Y given a configuration $\mathbf{x}$ of the variables such that some of its values are missing, and we do not want to assume MAR. Suppose further that we do not have access to the missingness mechanism. We can then augment our PC with the response model described and use it to draw inferences about Y. The only impediment is the the estimation of the parameters $\mu_{j,v}$ in Equation 5.13, as it relies on unseen and unavailable data. Very often, however, we can rely on expert domain knowledge to provide rough estimates for those parameters. Being only approximate, we are better subscribing for a partial specification in the form of probability intervals as described in Chapter 3.

If we have access to an MNAR sample of data (say, at prediction time), we can instead derive such bounds from Equation 5.13, by considering all possible values for $P(X_j = v|R_j = 1)$:

$$\mu_{j,v} \in \left[\frac{N_{j,v}}{N_{j,v} + N_{\text{miss}}}, 1\right] \tag{5.23}$$

where $N_{j,v}$ denotes the number of occurrences of $X_j = v$ in the MNAR dataset of size N with N_{miss} missing values for X_j.

In either case, the result is an augmented Credal Probabilistic Circuit, that we name CReM, where imprecision occurs only at edges connected to input indicator nodes (viz. those edges associated with response indicators). As the respective network structure is not tree-shaped, we cannot in principle use the algorithm of Credal Classification described in Chapter 3 to compute the dominance criterion.

Theorem 5.4.1. *Consider a complete, decomposable and strong selective w.r.t Y PC used to obtain a Partial Credal PC $\{\mathtt{S}_{\{\mathbf{w},\mu\}} : \mu \in \mathcal{C}\}$ representing our credal response model CReM with $sc(\mathtt{S}) = \{Y, \mathbf{X}^{obs}, \mathbf{X}^{miss}\}$ and root r where $\mathcal{C}$ is the Cartesian product of finitely-generated*

polytopes C_i, one for each sum node i. Then our proposed algorithm computes:

$$\delta_{\mathrm{S}^r,\mathbf{x}^{obs}}(y',y'') = \min_{\mu\in C}[\mathbb{P}_{\{\mathbf{w},\mu\}}(y',\mathbf{x}^{obs},\mathbf{x}^{miss}) - \mathbb{P}_{\{\mathbf{w},\mu\}}(y'',\mathbf{x}^{obs},\mathbf{x}^{miss})] \tag{5.24}$$

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

Proof. We now provide a linear-time algorithm for computing $\delta_{\mathrm{S}^r,\mathbf{x}^{\mathrm{obs}}}(y',y'')$ in a Credal PC obtained from a complete, decomposable and strong selective w.r.t Y PC. The algorithm can be described straightforwardly by a collection of recursive equations depending on the type of node at which it operates. The recursive formulation also provides a proof of its correctness under the above assumptions.

If node j is a **sum node of data model** S , the algorithm computes:

$$\delta_{\mathrm{S}_j}(y',y'') = \sum_{i\in ch(j)} w_i[\mathrm{S}_i(y',\mathbf{x}^{\mathrm{obs}}) - \mathrm{S}_i(y'',\mathbf{x}^{\mathrm{obs}})] = \sum_{i\in ch(j)} w_i\delta_{\mathrm{S}_i}(y',y'')\,. \tag{5.25}$$

The correctness of this operation follows from the *strong selectivity*, which allows us to optimize the weights of each subPC child independently by ensuring that there exists a unique *active child* (i.e. propagating a value other than 0) or that the scope of S is Y.

If j is a **credal sum node of response model** S with two children $[\![R_{X_j,v} = 1]\!]$ and $[\![R_{X_j,v} = 0]\!]$ and local weights μ_i , the algorithm computes:

$$\delta_{\mathrm{S}_j}(y',y'') = \min_{\mu\in C} \sum_{i\in ch(j)} \mu_i[\mathrm{S}_i(y',\mathbf{x}^{\mathrm{obs}}) - \mathrm{S}_i(y'',\mathbf{x}^{\mathrm{obs}})] = \sum_{i\in ch(j)} \min_{\mu_i\in C_i} \mu_i\delta_{\mathrm{S}_i}(y',y'')\,. \tag{5.26}$$

where $\min_{\mu_i\in C_i}\mu_i$ can be $\underline{\mu_{j,v}}$ or $1-\overline{\mu_{j,v}}$ depending on the value propagated by the leaves nodes $[\![R_{X_j,v} = 1]\!]$ and $[\![R_{X_j,v} = 0]\!]$ in the recursive call $\delta_{\mathrm{S}_i}(y',y'')$.

If j is a **product node** S such that Y is in the scope of S_k, $k \in ch(j)$, (and no other by Decomposability property), the algorithm computes:

$$\delta_{\mathrm{S}_j}(y',y'') = \min_{\mathbf{w}\in C}[\mathrm{S}_k(y',\mathbf{x}_k^{\mathrm{obs}}) - \mathrm{S}_k(y'',\mathbf{x}_k^{\mathrm{obs}})] = \delta_{\mathrm{S}_k}(y',y'') \prod_{i\in ch(j),i\neq k} \mathrm{opt}\ \mathrm{S}_i(\mathbf{x}_i^{\mathrm{obs}})\,, \tag{5.27}$$

where $\mathbf{x}_i^{\mathrm{obs}}$ denotes the projection of $\mathbf{x}^{\mathrm{obs}}$ into the scope of S_i, and opt = max if $\delta_{\mathrm{S}_k}(y',y'') < 0$ and opt = min if $\delta_{\mathrm{S}_k}(y',y'') \geq 0$. The first term in Eq. 5.27 denotes the recursive computation on the sub-PC S_k.

If j is a **leaf node** representing an indicator variable or indicator response then the algorithm computes:

$$\delta_{\mathrm{S}_j}(y',y'') = \begin{cases} -1 & \text{if } \mathrm{S}_j \text{ is } [\![Y = y'']\!], \\ 1 & \text{if } \mathrm{S}_j \text{ is } [\![Y = y']\!], \text{ or is consistent with } \mathbf{x}^{\mathrm{obs}}, \\ 1 & \text{if } \mathrm{S}_j \text{ is } [\![R_{X_i,v} = 0]\!] \text{ and } X_i \text{ is missing }, \\ 1 & \text{if } \mathrm{S}_j \text{ is } [\![R_{X_i,v} = 1]\!] \text{ and } X_i \text{ is observed at } v \text{ value }, \\ 0 & \text{otherwise.} \end{cases} \tag{5.28}$$

□

Example 5.4.1. *Consider the Augmented model of Figure 5.2b, which is strongly selective with respect of Y. Given the observation $Z = 1$ and and a missing value for X generated by an unknown MNAR process, suppose we obtain specialist information that can be encoded as intervals for response parameters as follows: $\mu_{x,0} \in [0, 0.3]$, $\mu_{x,1} \in [0.1, 0.4]$, $\mu_{x,2} \in [0.6, 0.7]$, $\mu_{x,3} \in [0.7, 0.8]$, $\mu_{x,4} \in [0.7, 0.9]$. We can model that information into the response model of the augmented circuit and then credally classify Y. Figure 5.5 shows the evaluation of credal dominance over the credal augmented model (that contains intervals only in the response model portions).*

As the result in the PC root of Figure 5.5 we obtain the value $\delta_{0,1} = 0.164$ and $\delta_{1,0} = -0.208$, thus the credal prediction is $\{0\}$

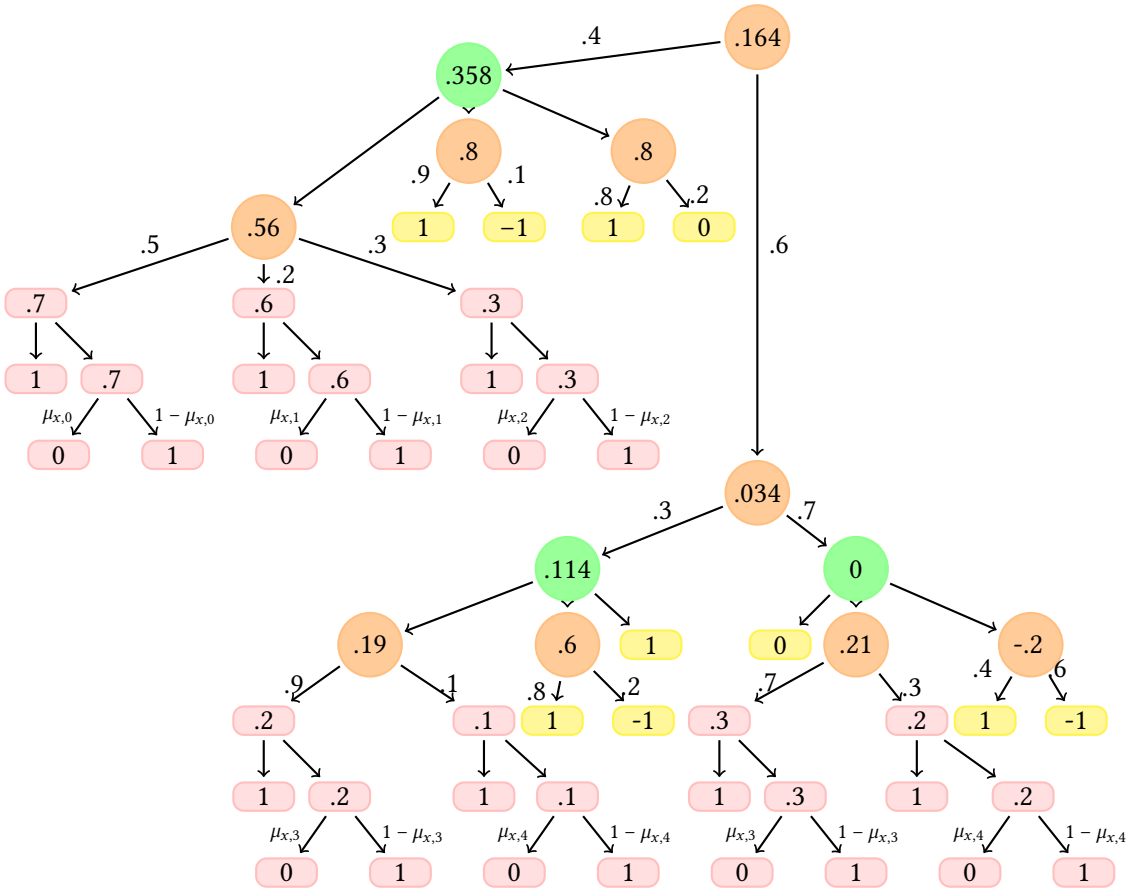


Figure 5.5: *Example of credal classification using intervals in the Response model*

5.5 Experiments

We empirically evaluate the ability of our proposed methods in assessing the robustness of classifications to non-ignorable missing feature values using the tractable conservative inference and Credal Response Model. To this end, we learn class-factorized GeFs from some well-known complete datasets for density estimation and classification

(Davis and Domingos, 2010; Goldberg *et al.*, 2001), using the algorithm in (Correia *et al.*, 2020).

5.5.1 Tractable Conservative Inference

The characteristics of the datasets are in Table 5.1. Missing test values are artificially simulated using a mix of MAR, MCAR and MNAR mechanisms. We consider a disjoint set of selected variables for each missing mechanism, and the MNAR mechanism is conditioned at most by two variables. The average number of (MAR, MCAR and MNAR) missing values per instance is denoted as AvM, and the average number of MNAR values per instance is denoted as AvMNAR.

Dataset	Variables	# Test Instances	AvM	AvMNAR	# Train Instances	Model Size
Audio	100	3,000	4.1	1.9	15,000	3,858
Dna	180	1,186	5.5	2.2	1,600	1,038
Netflix	100	3,000	6.7	3.0	15,000	3,524
Nltcs	16	3,236	1.4	0.4	16,181	568

Table 5.1: *Datasets characteristics.*

In Table 5.2 we report relevant performance metrics of our TCI predictions. The last column (*Acc*) shows the accuracy of classifications made by marginalizing all missing test values. Columns *RAcc* and $\neg$*RAcc* report the classification accuracy on the portions of instances that are robust and non-robust, respectively. A test instance is robust if the TCI inference (Eq. 5.18) returns only one non-dominated class value. For the rows tagged "marg", we marginalize MAR variables and optimize over the MNAR variables. For the other rows, we optimize over all missing values. Column *DR* shows the percentage of robust instances. By comparing *RAcc*, *Acc* and $\neg$*RAcc*, we observe the ability of TCI in discriminating between the easy-to-classify instances, corresponding to the robust ones, and the harder ones (non-robust instances), for which a set of classes is returned. Similar conclusions can be reached by inspecting the *SAcc* (Set Accuracy) column, which measures the percentage of (set-valued) classifications that contain the true class. Finally, the informative character of marginal classifications are captured by the discounted accuracy (*DAcc*), which penalizes "imprecise" classifications by weighting correct set-valued classifications by the reciprocal of their size (see (Zaffalon, Corani, *et al.*, 2012) for more details and motivation about the metric). A *DAcc* value higher than the corresponding *Acc* denotes that the classifier issues predictions that are on average more accurate than random classifications, hence being informative despite the false MAR assumption.

To analyze the approach on a more realistic missingness scenario, we learn GeFs from a binarized version of the complete version of the Jester dataset.[1] This is a complete dataset of user ratings on 10 items (variables), divided into 17,467 training instances (users) and 7,486 test instances. We build a binary classification task by predicting, for each user/instance, the rating of a distinguished item given the other items ratings. We fabricate MNAR values in both training and test sets by independently omitting a positive rating with either low

[1] http://eigentaste.berkeley.edu/dataset

Dataset	TCI			Marginalization		
	SAcc	*DAcc*	*DR*	*RAcc*	¬*RAcc*	*Acc*
Audio	0.879	0.707	65.6	0.807	0.679	0.763
Audio (marg)	0.863	0.708	69.1	0.798	0.686	
Dna	0.899	0.799	80.0	0.880	0.511	0.806
Dna (marg)	0.858	0.801	88.6	0.846	0.496	
Netflix	0.894	0.662	53.6	0.771	0.652	0.716
Netflix (marg)	0.873	0.665	58.3	0.760	0.655	
Nltcs	0.980	0.912	86.4	0.977	0.856	0.961
Nltcs (marg)	0.975	0.906	86.2	0.972	0.888	

Table 5.2: *Set Accuracy (SAcc), Discounted Accuracy (DAcc), percentage of robust instances (DR), and classification accuracy on robust (RAcc), non-robust (¬RAcc) and overall (Acc) instances when marginalizing missing values at prediction time.*

probability ($p = 0.05$) or high probability ($p = 0.5$). This simulates observed behaviour of users providing ratings in such systems (Marlin, Zemel, Roweis, *et al.*, 2011). Table 5.3 shows that learning an imprecise model results in better accuracy than the precise version that ignore missing values. Note that when learning a credal PC, we might produce set-valued classifications even when we marginalize (MAR) the missing test values. Figure 5.6 shows that for both missingness levels the measure in (5.16) can be used to detect easy-to-classify instances for the precise classifier that assumes MAR. Similar patterns are achieved in terms of (modified) discounted accuracy in Figure 5.7, where this approach is combined with a rejection option.

Model	Inference	p	Model + Inference			Precise + MAR		
			SAcc	*DAcc*	*DR*	*RAcc*	¬*RAcc*	*Acc*
Imprecise	MAR	0.05	0.689	0.664	95.1	0.596	0.540	
	TCI		0.716	0.661	88.9	0.600	0.540	0.593
Precise			0.644	0.602	91.5	0.598	0.536	
Imprecise	MAR	0.5	0.753	0.597	68.7	0.639	0.435	
	TCI		0.847	0.597	50.0	0.693	0.457	0.575
Precise			0.827	0.578	50.3	0.657	0.493	

Table 5.3: *Performance of models learned from Jester with two different missingness proportions p in the training and test set. Imprecise models are obtained as in Chapter 3, precise models are obtained after removal of instances with missing values.*

5.5.2 Credal Response Model

Now, we empirically compare Credal Response Model against Tractable Conservative Inference and marginalization of missing values. To this end, we learn class-factorized GeFs from some well-known complete datasets for density estimation and classification (Davis and Domingos, 2010; Goldberg *et al.*, 2001), using the algorithm in (Correia *et al.*, 2020). The characteristics of the datasets appear in Table 5.4. Missing test values are simulated using a MNAR mechanism, for each instance i and variable X_j we have

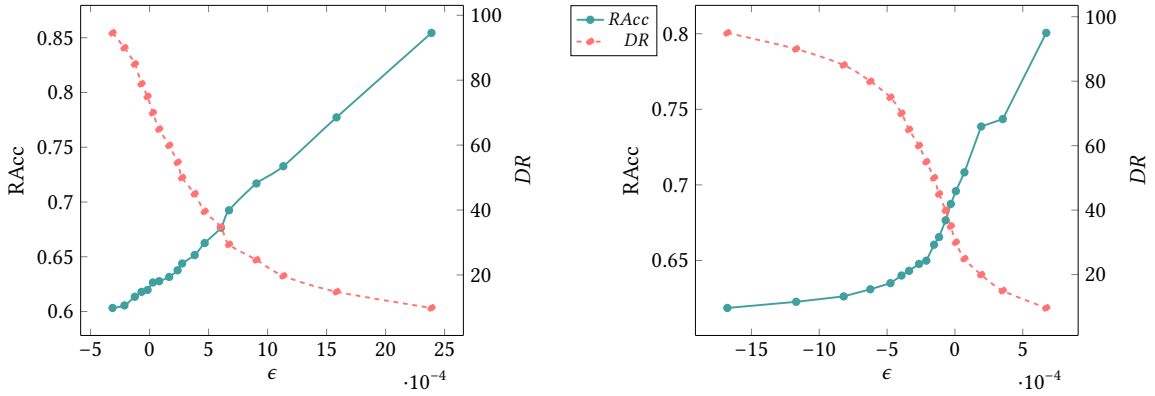


Figure 5.6: *Robust accuracy (RAcc) of the precise classifier (MAR) for the Jester dataset with low ($p = 0.05$, left) and high ($p = 0.5$, right) missingness levels. Condition $\delta_{\mathrm{M,o}}(y, \neg y) > \epsilon$, where $\delta_{\mathrm{M,o}}$ is defined as in Eq. (5.16) and y is the class returned by the classifier, is used to decide robustness. We also display %R by threshold ϵ.*

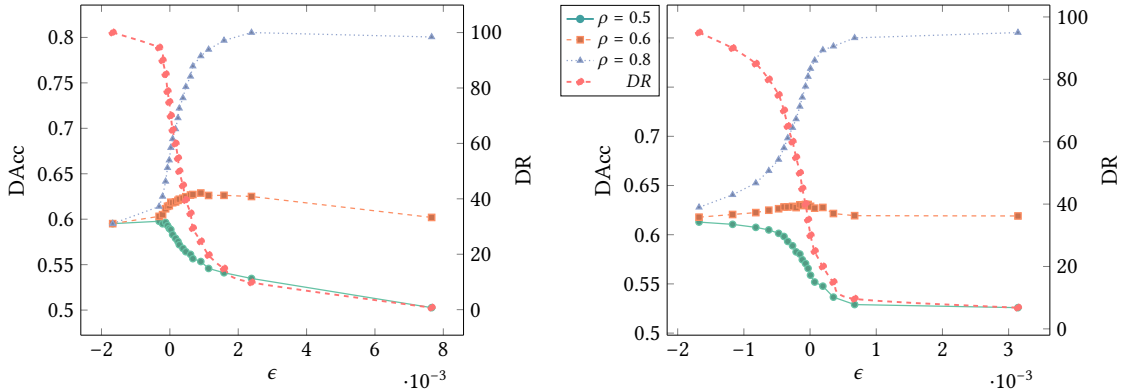


Figure 5.7: *Modified discounted accuracy of the (imprecise) classifier for the Jester dataset with low ($p = 0.05$, left) and high ($p = 0.5$, right) missingness levels. Robustness is decided as in Figure 5.6 for different values of the threshold ϵ. A value ρ is used instead of 0.5 to score imprecise classifications, which regulates preference for model uncertainty against aleatory uncertainty (see (Zaffalon, Corani, et al., 2012)).*

$\mathbb{P}(R_j^i = 0 | X_j^i = 1) = 0.2$ and $\mathbb{P}(R_j^i = 0 | X_j^i = 0) = 0$, wher $X_j \neq Y$. The average number of missing values per instance is denoted as AvM, and Y as number of classes.

Dataset	Variables	# Test Instances	AvM	# Train Instances	# Y
Dna	180	1,186	24.545	1,600	2
Jester	10	7,486	3.564	17,467	5
Insurance	27	2,400	1.303	5,600	3
Nltcs	16	3,236	1.077	16,181	2

Table 5.4: *Characteristics of the datasets*

In Table 5.5 we report relevant performance metrics of our Credal Response Model (CReM) predictions vs the Tractable Conservative Inference (TCI) proposed in (Villanueva Llerena, Mauá, and Antonucci, 2021), as well as the accuracy of the precise classifier

that marginalizes missing data. As in the previous section, set Accuracy (*SAcc*) measures the percentage of (indeterminate/non-robust or determinate/robust) classifications that contains the true class, Discounted Accuracy (*DAcc*) measures the percentage of (determinate or indeterminate) of classifications that contain the true class, weighted by the reciprocal of set size, *DR* shows the percentage of determinate classifications (the ones with a single maximal class) and Precise Accuracy (*RAcc*) measures accuracy among determinate classifications.

Dataset	Marg.	TCI				CReM			
	Acc	*SAcc*	*DAcc*	*DR*	*RAcc*	*SAcc*	*DAcc*	*DR*	*RAcc*
Dna	79.1	92.3	76.9	69.1	88.9	84.4	80.2	91.7	83
Jester	36.4	96.4	21.3	2.7	41.5	59.6	31.7	36.6	37.9
Insurance	78.2	85.2	77.4	84.1	84.5	79.9	78	95.7	84.3
Nltcs	93.9	98.0	90.9	85.7	97.7	95.1	93.3	96.4	94.9

Table 5.5: *Performance of TCI vs Credal Response Model vs Marginalization of missing values in terms of the metrics: accuracy (Acc), Set Accuracy (SAcc), Discounted Accuracy (DAcc), Determinacy Rate (DR), and accuracy of determinate classifications (RAcc). See text for explanation..*

According to the results, in comparison to TCI, CReM obtains smaller set and precise accuracies with a significantly higher determinacy rate, leading to an overall improved discounted accuracy.

The effect of such a trade-off is also shown in the left plot in Figure 5.8, which displays precise accuracy of classifications made either by CReM or TCI selecting the class with the highest value of δ, and sorted by that same value for dataset Nltcs (left) and Dna (rigth). We see here that CReM is less adequate than TCI at judging robustness of such instances.

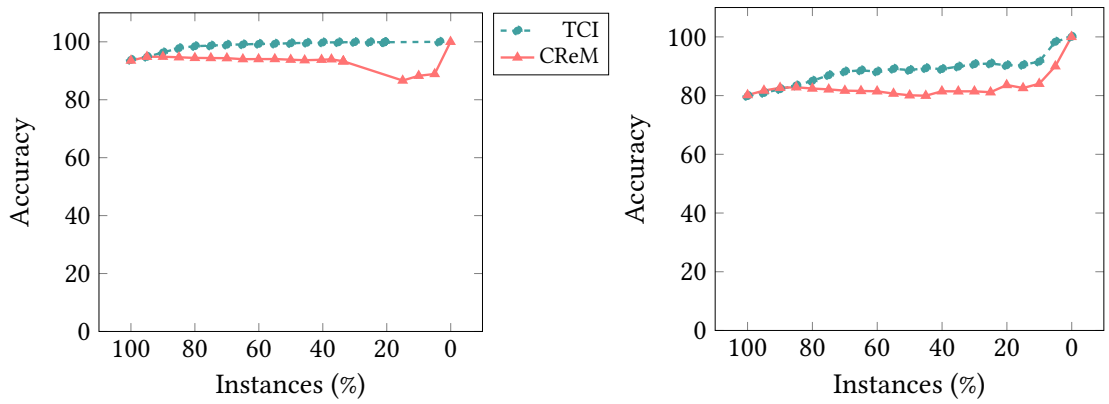


Figure 5.8: *Accuracy vs percentage (%Instances) of the dominant instances for TCI vs Credal Response model using Nltcs(left) and Dna (right) datsets, where robustness satisfy $\delta_{\mathrm{M}}(y', y'') > \delta$, δ_{M} is defined as in Eq. (3.7), and y' is the class with higher δ.*

Chapter 6

Conclusions and Future Work

Probabilistic circuits are probabilistic models that have shown very promising results in several tasks. Sum-Product Networks (SPNs), Selective Sum-Product Networks (SSPNs), Probabilistic Sentential Decision Diagrams (PSDDs) and Generative Random Forest (GeFS) are relevant examples of PCs.

Credal SPNs are sets of SPNs obtained by a simultaneous small perturbation of all model parameters, where all members share the same network structure.

In this work, we developed polynomial-time algorithms for qualitative global sensitivity analysis of inferences made by PCs using Credal PCs, Experimental evaluation showed that these algorithms outperform the baseline approaches in distinguishing robust (correct) and non-robust instances.

Our first contribution is a tractable procedure for classifying robust and non-robust MAP inferences 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 on two different tasks: missing value completion and performing multi-label classification. Our results show that the proposed algorithms are often better at separating robust from non-robust inferences than the standard approach based on the difference of probabilities, especially when data is scarce. We left as future work to perform experiments with other decision making criteria to decide dominance, such as maximality and interval dominance. Although we have carried extensive experimentation for qualitative SA of MAP inferences, 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.

As our second contribution, we developed two exact polynomial-time methods for handling non-ignorable missing data using probabilistic circuits. We show that our algorithms are exact if the model satisfies certain constraints, which is the case for GeFs. First, we develop an efficient cautious algorithm to identify robust classifications made by PCs for imputations of the non-ignorable portion of missing data at prediction time. Experiments on realistic data show that the approaches are effective in discriminating instances that are sensitive to the missingness process from those that are not. Finally, we developed a

tractable method for performing predictive inference under non-ignorable missing data with a partially specified response model of missingness. Experiments on realistic data showed that our method delivers less biased (probabilistic) classifications than approaches that assume missing at random and are more determinate than our previously proposed overcautious approach. We left as future work the treatment of other types of missing data (e.g., coarse and unreliable observations). The treatment of non-ignorable missing data at training time remains open.

Appendix A

Approximate Tractable Conservative Inference for Selective SPNs

As we reviewed in Chapter 5, the TCI is performed exactly in strong selective SPNs. In this chapter we present the results of TCI as an approximation for classisification in the presence of MNAR data in selective SPNs We learned selective SPNs from five well-known datasets for density estimation. The selected datasets and their characteristics appear in Table A.1, where the test sets are simulated using MAR, MCAR and MNAR mechanisms.

Dataset	Vars	Test	Avg.M	$\|Y\|$	Train	Model
Audio	100	3000	4.1	2	15000	3858
Dna	180	1186	5.5	2	1600	1038
Jester	100	4116	6	2	9000	3170
Netflix	100	3000	6.7	2	15000	3524
Nltcs	16	3236	1.4	2	16181	568

Table A.1: *Datasets characteristics to perform TCI in Selective SPNs, we use artificial generation of missing values using MAR, MCAR and MNAR.*

Table A.2 shows the results of the application of our tractable conservative approach both the training and the test set. When comparing the accuracies of Marginalization on the robust and non-robust instances ($RAcc$ and $\neg RAcc$), most of the times we observe more accurate predictions for the robust instances. This representing the ability of TCI in discriminating between the easy-to-classify instances, corresponding to the robust ones, and the *hard* ones for which the robustness of the TCI allows to return a set of classes, mostly including the true one, this corresponding to the high values for the Set Accuracy.

Dataset	TCI				Marginalization		
	SAcc	*DAcc*	*AvgSet*	*DR*	*RAcc*	¬*RAcc*	*Acc*
Audio	0.99	0.5	2	1.8	0.69	0.74	0.74
Audio Marg.	0.95	0.56	1.8	21.9	0.76	0.74	
Dna	0.99	0.5	2	0.59	0.86	0.77	0.77
Dna Marg.	0.98	0.54	1.9	10.9	0.84	0.76	
Jester	0.99	0.5	2	0.12	0.6	0.75	0.74
Jester Marg.	0.97	0.52	1.9	9.6	0.72	0.74	
Netflix	0.99	0.5	2	0.27	0.88	0.67	0.67
Netflix Marg.	0.98	0.51	1.9	5.3	0.69	0.67	
Nltcs	0.98	0.63	1.7	29.4	0.94	0.93	0.94
Nltcs Marg.	0.97	0.73	1.5	51.3	0.95	0.92	

Table A.2: *Performance of TCI versus Marginalization.*

References

[Amer and Todorovic 2016] Mohamed R Amer and Sinisa Todorovic. "Sum product networks for activity recognition". *IEEE transactions on pattern analysis and machine intelligence* (2016), pp. 800–813 (cit. on pp. 1, 5, 21).

[Antonucci, Facchini, *et al.* 2019] Alessandro Antonucci, Alessandro Facchini, and Lilith Mattei. "Credal sentential decision diagrams". In: *International Symposium on Imprecise Probabilities: Theories and Applications*. 2019, pp. 14–22 (cit. on p. 3).

[Antonucci and Piatti 2009] Alessandro Antonucci and A. Piatti. "Modeling unreliable observations in Bayesian networks by credal networks". In: *Proceedings of the Third International Conference on Scalable Uncertainty Management (SUM)*. 2009, pp. 28–39 (cit. on p. 50).

[Antonucci and Zaffalon 2008] Alessandro Antonucci and Marco Zaffalon. "Decision-theoretic specification of credal networks: a unified language for uncertain modeling with sets of Bayesian networks". *International Journal of Approximate Reasoning* 49.2 (2008), pp. 345–361 (cit. on p. 50).

[Azur *et al.* 2011] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. "Multiple imputation by chained equations: what is it and how does it work?" *International Journal of Methods in Psychiatric Research* 20.1 (2011), pp. 40–49 (cit. on pp. 2, 46).

[Berger 1985] James Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 1985 (cit. on p. 1).

[Bodlaender *et al.* 2002] 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*. 2002, pp. 675–679 (cit. on p. 30).

[Campos *et al.* 1994] Luis M. de Campos, Juan F. Huete, and Serafín Moral. "Probability interval: a tool for uncertain reasoning". *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems* 2.2 (1994), pp. 167–196 (cit. on p. 22).

[Castillo *et al.* 1997] 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* (1997), pp. 412–423 (cit. on p. 2).

[Chan and Darwiche 2002] Hei Chan and Adnan Darwiche. "When do numbers really matter?" *Journal of artificial intelligence research* (2002), pp. 265–287 (cit. on p. 2).

[Chan and Darwiche 2004] 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*. 2004, pp. 67–75 (cit. on p. 2).

[Chan and Darwiche 2006] Hei Chan and Adnan Darwiche. "On the robustness of most probable explanations". In: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*. 2006, pp. 63–71 (cit. on p. 2).

[Chavira and Darwiche 2005] Mark Chavira and Adnan Darwiche. "Compiling bayesian networks with local structure". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. 2005, pp. 1306–1312 (cit. on p. 5).

[Cheng *et al.* 2014] 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 (cit. on pp. 1, 5, 21).

[A. Choi and Darwiche 2013] Arthur Choi and Adnan Darwiche. "Dynamic minimization of sentential decision diagrams". In: *The 27th AAAI Conference on Artificial Intelligence*. 2013 (cit. on pp. 17, 43).

[Y. Choi *et al.* 2020] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. *Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models*. 2020 (cit. on p. 8).

[Chow and Liu 1968] C. Chow and C. Liu. "Approximating discrete probability distributions with dependence trees". *IEEE Transactions on Information Theory* 14.3 (1968), pp. 462–467 (cit. on p. 9).

[Conaty *et al.* 2017] 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*. 2017, pp. 322–331 (cit. on pp. 11, 18, 30, 31).

[Correia *et al.* 2020] Alvaro H. C. Correia, Robert Peharz, and Cassio Polpo de Campos. "Joints in random forests". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. 2020 (cit. on pp. 1, 3, 5, 8, 14, 15, 46, 58, 59).

[Darwiche 2003] Adnan Darwiche. "A differential approach to inference in Bayesian networks". *Journal of the ACM (JACM)* 50.3 (2003), pp. 280–305 (cit. on pp. 5, 17).

[Davis and Domingos 2010] Jesse Davis and Pedro Domingos. "Bottom-up learning of markov network structure". In: *Proceedings of the 27th International Conference on Machine Learning*. 2010, pp. 271–280 (cit. on pp. 36, 58, 59).

[Dembczyński *et al.* 2012] 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 (2012), pp. 5–45 (cit. on p. 39).

[Di Mauro *et al.* 2016] 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*. 2016, pp. 147–158 (cit. on p. 39).

[Geh and Mauá 2019] Renato Lui Geh and Denis Deratani Mauá. "End-to-end imitation learning of lane following policies using sum-product networks". In: *Proceedings of the Sixteenth National Meeting of Artificial and Computational Intelligence (Brazil)*. 2019 (cit. on p. 21).

[Geh and Mauá 2021] Renato Lui Geh and Denis Deratani Mauá. "Learning probabilistic sentential decision diagrams under logic constraints by sampling and averaging". In: *Uncertainty in Artificial Intelligence*. 2021, pp. 2039–2049 (cit. on p. 18).

[Gens and Domingos 2013] Robert Gens and Pedro Domingos. "Learning the structure of sum-product networks". In: *Proceedings of the International Conference on Machine Learning*. 2013, pp. 873–880 (cit. on p. 9).

[Goldberg *et al.* 2001] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. "Eigentaste: a constant time collaborative filtering algorithm". *information retrieval* 4.2 (2001), pp. 133–151 (cit. on pp. 58, 59).

[Huang *et al.* 2006] Jinbo Huang, Mark Chavira, and Adnan Darwiche. "Solving map exactly by searching on compiled arithmetic circuits". In: *Proceedings of the 21st National Conference on Artificial Intelligence*. 2006, pp. 1143–1148 (cit. on p. 5).

[Jiang *et al.* 2018] 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, pp. 5541–5552 (cit. on p. 2).

[Khosravi, Y. Choi, *et al.* 2019] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. "On tractable computation of expected predictions". In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019 (cit. on p. 2).

[Khosravi, Vergari, *et al.* 2020] Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. "Handling missing data in decision trees: a probabilistic approach". *arXiv preprint arXiv:2006.16341* (2020) (cit. on p. 46).

[Kisa *et al.* 2014] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. "Probabilistic sentential decision diagrams". In: *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*. 2014 (cit. on pp. 1, 5, 17).

[Kjærulff and Gaag 2000] Uffe Kjærulff and Linda C van der Gaag. "Making sensitivity analysis computationally efficient". In: *Proceedings of the 16th conference on Uncertainty in artificial intelligence*. 2000, pp. 317–325 (cit. on p. 2).

[Koller and Friedman 2009] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009 (cit. on p. 1).

[Lambrou *et al.* 2010] Antonis Lambrou, Harris Papadopoulos, and Alex Gammerman. "Reliable confidence measures for medical diagnosis with evolutionary algorithms". *IEEE Transactions on Information Technology in Biomedicine* 15 (2010), pp. 93–99 (cit. on pp. 1, 21).

[Laskey 1995] Kathryn Blackmond Laskey. "Sensitivity analysis for probability assessments in Bayesian networks". *IEEE Transactions on Systems, Man, and Cybernetics* (1995), pp. 901–909 (cit. on p. 2).

[Liang *et al.* 2017] 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 (cit. on p. 17).

[Little and Rubin 2014] Roderick Little and Donald Rubin. "Nonignorable Missing-Data Models" (2014), pp. 312–348. doi: 10.1002/9781119013563.ch15 (cit. on p. 48).

[Lowd and Domingos 2008] Daniel Lowd and Pedro Domingos. "Learning arithmetic circuits". In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. 2008, pp. 383–392 (cit. on p. 5).

[Malinin and Gales 2018] Andrey Malinin and Mark Gales. "Predictive uncertainty estimation via prior networks". In: *Proceedings of the 32nd International Conference on Neural Information Processing System*. 2018, pp. 7047–7058 (cit. on p. 2).

[Manski 2005] Charles Manski. "Partial identification with missing data: concepts and findings". *International Journal of Approximate Reasoning* 39.2-3 (2005), pp. 151–165 (cit. on p. 3).

[Marlin, Zemel, Roweis, *et al.* 2011] Benjamin Marlin, Richard Zemel, Sam Roweis, and Malcolm Slaney. "Recommender systems: missing data and statistical model estimation". In: *Proceedings of the 22nd International Joint Conference in Artificial Intelligence (IJCAI)*. 2011 (cit. on p. 59).

[Marlin, Zemel, Sam, *et al.* 2007] Benjamin Marlin, Richard Zemel, Roweis Sam, and Malcolm Slaney. "Collaborative filtering and the missing at random assumption". In: *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, UAI 2007*. 2007, pp. 267–275. isbn: 0974903930. eprint: 1206.5267 (cit. on p. 45).

[Mattei *et al.* 2020] Lilith Mattei, Alessandro Antonucci, Denis Deratani Mauá, Alessandro Facchini, and Julissa Villanueva Llerena. "Tractable inference in credal sentential decision diagrams". *International Journal of Approximate Reasoning* (2020) (cit. on pp. 2, 3, 19, 24, 27).

[Mauá, Conaty, *et al.* 2018] Denis Deratani Mauá, Diarmaid Conaty, Fabio Gagliardi Cozman, Katja Poppenhaeger, and Cassio Polpo de Campos. "Robustifying sum-product networks". *International Journal of Approximate Reasoning* (2018), pp. 163–180 (cit. on pp. 2, 21, 22, 24, 25, 27, 53).

[Mauá, De Campos, *et al.* 2014] Denis Deratani Mauá, Cassio Polpo De Campos, Alessio Benavoli, and Alessandro Antonucci. "Probabilistic inference in credal networks: new complexity results". *Journal of Artificial Intelligence Research* 50 (2014), pp. 603–637 (cit. on p. 50).

[Mei *et al.* 2017] Jun Mei, Yong Jiang, and Kewei Tu. "Maximum a posteriori inference in sum-product networks". In: *The 32nd AAAI Conference on Artificial Intelligence*. 2017, pp. 1923–1930 (cit. on pp. 11, 31).

[Murphy 2020] Kevin Murphy. *Probabilistic Machine Learning: An Introduction (Adaptive Computation and Machine Learning)*. 2020. isbn: 0262193981 (cit. on p. 45).

[Peharz, Gens, and Domingos 2014] Robert Peharz, Robert Gens, and Pedro Domingos. "Learning selective sum-product networks". In: *Proceedings of the Workshop on Learning Tractable Probabilistic Models*. 2014 (cit. on pp. 9–11, 13, 36, 54).

[Peharz, Gens, Pernkopf, *et al.* 2017] 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* (2017), pp. 2030–2044 (cit. on pp. 11, 31, 54).

[Peharz, Tschiatschek, *et al.* 2015] Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. "On theoretical properties of sum-product networks". In: *Artificial Intelligence and Statistics*. PMLR. 2015, pp. 744–752 (cit. on p. 8).

[Peharz, Vergari, *et al.* 2020] Robert Peharz, Antonio Vergari, *et al.* "Random sum-product networks: a simple and effective approach to probabilistic deep learning". In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference (UAI)*. 2020 (cit. on pp. 1, 21).

[Poon and Domingos 2011] Hoifung Poon and Pedro Domingos. "Sum-product networks: a new deep architecture". In: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*. 2011, pp. 337–346 (cit. on pp. 1, 5, 21, 29).

[Pronobis and Rao 2017] Andrzej Pronobis and Rajesh PN Rao. "Learning deep generative spatial models for mobile robots". In: *EEE/RSJ International Conference on Intelligent Robots and Systems*. 2017, pp. 755–762 (cit. on pp. 1, 5, 21).

[Pronobis, Riccio, *et al.* 2017] 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 (cit. on pp. 5, 21).

[Rashwan *et al.* 2016] Abdullah Rashwan, Han Zhao, and Pascal Poupart. "Online and distributed bayesian moment matching for parameter learning in sum-product networks". In: *Proceedings of the Artificial Intelligence and Statistics*. 2016, pp. 1469–1477 (cit. on p. 2).

[Renooij and Van Der Gaag 2008] Silja Renooij and Linda C Van Der Gaag. "Evidence and scenario sensitivities in naive bayesian classifiers". *International Journal of Approximate Reasoning* (2008), pp. 398–416 (cit. on p. 2).

[Rubin 1976] Donald Rubin. "Inference and missing data". *Biometrika* 63.3 (1976), pp. 581–592 (cit. on pp. 3, 46).

[Sensoy *et al.* 2018] Murat Sensoy, Melih Kandemir, and Lance Kaplan. "Evidential deep learning to quantify classification uncertainty". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3179–3189 (cit. on p. 2).

[Shao *et al.* 2020] Xiaoting Shao *et al.* "Conditional sum-product networks: imposing structure on deep probabilistic architectures". In: *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*. 2020 (cit. on p. 1).

[Shen *et al.* 2017] Yujia Shen, Arthur Choi, and Adnan Darwiche. "A tractable probabilistic model for subset selection". In: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. 2017 (cit. on p. 1).

[Troffaes 2007] Matthias Troffaes. "Decision making under uncertainty using imprecise probabilities". *International journal of approximate reasoning* 45.1 (2007), pp. 17–29 (cit. on p. 25).

[Vergari, Di Mauro, *et al.* 2015] 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*. 2015, pp. 343–358 (cit. on p. 9).

[Vergari, Molina, *et al.* 2019] Antonio Vergari, Alejandro Molina, *et al.* "Automatic bayesian density analysis". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 5207–5215 (cit. on p. 2).

[Villanueva Llerena and Mauá 2017] Julissa Villanueva Llerena and Denis Deratani Mauá. "On using sum-product networks for multi-label classification". In: *Proceedings of the 6th Brazilian Conference on Intelligent Systems*. BRACIS. 2017, pp. 25–30 (cit. on pp. 1, 29, 39).

[Villanueva Llerena and Mauá 2019] Julissa Villanueva Llerena and Denis Deratani Mauá. "Robust analysis of map inference in selective sum-product networks". In: *20th International Symposium on Imprecise Probabilities: Theories and Applications*. 2019, pp. 430–440 (cit. on pp. 3, 30).

[Villanueva Llerena and Mauá 2020] Julissa Villanueva Llerena and Denis Deratani Mauá. "Efficient algorithms for robust analysis of maximuma posteriori inference in selective sum-product networks". *International Journal of Approximate Reasoning* (2020) (cit. on pp. 2, 3, 27).

[Villanueva Llerena and Mauá 2022] Julissa Villanueva Llerena and Denis Deratani Mauá. "Tractable classification with non-ignorable missing data using generative random forests". In: *Proceedings of the X Symposium on Knowledge Discovery, Mining and Learning*. SBC, 2022 (cit. on pp. 3, 4).

[Villanueva Llerena, Mauá, and Antonucci 2021] Julissa Villanueva Llerena, Denis Deratani Mauá, and Alessandro Antonucci. "Cautious classification with data missing not at random using generative random forests". In: *European Conference on Symbolic and Quantitative Approaches with Uncertainty*. Springer. 2021, pp. 284–298 (cit. on pp. 3, 4, 60).

[Zaffalon 2002] Marco Zaffalon. "The naive credal classifier". *Journal of statistical planning and inference* 105.1 (2002), pp. 5–21 (cit. on p. 25).

[Zaffalon, Corani, *et al.* 2012] Marco Zaffalon, Giorgio Corani, and Denis Deratani Mauá. "Evaluating credal classifiers by utility-discounted predictive accuracy". *International Journal of Approximate Reasoning* (2012), pp. 1282–1301 (cit. on pp. 58, 60).

[Zaffalon and Miranda 2009] Marco Zaffalon and E. Miranda. "Conservative inference rule for uncertain reasoning under incompleteness". *Journal of Artificial Intelligence Research* 34 (2009), pp. 757–821 (cit. on pp. 50, 52).

[Zhao *et al.* 2016] Han Zhao, Tameem Adel, Geoff Gordon, and Brandon Amos. "Collapsed variational inference for sum-product networks". In: *International Conference on Machine Learning*. 2016, pp. 1310–1318 (cit. on p. 2).

[Zheng *et al.* 2018] 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 (cit. on pp. 1, 5, 21).

Index