Cautious Classification with Data Missing Not at Random using Generative Random Forests*

Julissa Villanueva Llerena[1][0000−0003−1584−0427], Denis Deratani Mauá[1][0000−0003−2297−6349], and Alessandro Antonucci[2][0000−0001−7915−2768]

[1] Institute of Mathematics and Statistics, Universidade de São Paulo, Brazil
{jgville,ddm}@ime.usp.br
[2] Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland
alessandro@idsia.ch

Abstract. Missing data present a challenge for most machine learning approaches. When a generative probabilistic model of the data is available, an effective approach is to marginalize missing values out. Probabilistic circuits are expressive generative models that allow for efficient exact inference. However, data is often missing not at random, and marginalization can lead to overconfident and wrong conclusions. In this work, we develop an efficient algorithm for assessing the robustness of classifications made by probabilistic circuits to imputations of the non-ignorable portion of missing data at prediction time. We show that our algorithm is exact when the model satisfies certain constraints, which is the case for the recent proposed Generative Random Forests, that equip Random Forest Classifiers with a full probabilistic model of the data. We also show how to extend our approach to handle non-ignorable missing data at training time.

Keywords: probabilistic circuits · generative random forests · missing data · conservative inference rule.

1 Introduction

This work presents a new tractable algorithm for analyzing the effect of all potential imputations of non-ignorable missing values to a probabilistic classifier's response.

Missing data present a challenge in many machine learning tasks. The standard approach to inference with such data is to either impute or marginalize out the missing values [3, 7]. The latter option requires a complete statistical model of features and target variables, and efficient inference routines. Recently, Correia, Peharz and de Campos [5] proposed Generative Random Forest (GeFs), which extend standard random forest classifiers into complete statistical models with tractable marginalization of missing values. GeFs have shown superior performance to imputation approaches and other ad hoc heuristics used for random forests in classification tasks under missing data [5].

* Supported by CAPES Finance 001, CNPQ grant #304012/2019-0.

GeFs are actually part of a larger class of tractable probabilistic models, called Probabilistic Circuits, that allow for linear time marginalization [4, 11]. Sum-Product Networks [21], Probabilistic Sentential Decision Diagrams [9] and Cutset Networks [22] are other notable examples of Probabilistic Circuits. These models have obtained impressive results in several machine learning tasks due to their ability to efficiently represent and manipulate intricate multidimensional distributions [20, 21, 24–26, 31].

Imputation and marginalization are either theoretically supported by a *missing at random* assumption (MAR), that roughly considers that the probability of the missing values does not depend on the variables with missing values themselves [23]. This is not always a sensible choice [13]. For instance, in personalized recommendation, users have a strong bias towards rating items which they either strongly like or strongly dislike [14]. Automatically constructed knowledge-based systems offer another example, as they are most often populated exclusively with "positive" facts involving only a small fraction of the true facts [27]. In such cases, called *non-ignorable missing data* or MNAR (i.e., missing not at random), imputing or averaging over completions can lead to biased and inconsistent estimates and ultimately hurt performance. Importantly, it is not possible to statistically test whether or not the MAR assumption is satisfied, nor to learn from data the incompleteness process responsible for the missing values [17, 23].

In the presence of MNAR data, marginalization can still be used as a heuristic, at the risk of inducing excessive bias. To analyze such potential bias, we follow [30] and propose to quantify the effect in a probabilistic classifier's decision to all possible imputations of the MNAR data. The challenge that we overcome here is doing so in a computationally tractable way, making use of the machinery of Probabilistic Circuits. Such an analysis has been previously applied to traditional probabilistic models such as Bayesian networks [1], where it suffers from intractability of inference. In fact, one can show that for Bayesian networks the task is equivalent to performing marginal inference in credal networks [2], a task whose theoretical and practical complexity far exceeds that of marginal inference in Bayesian networks [15].

In this work, we devise a polynomial-time procedure to quantify the effect of different imputations of the missing values of features in the classification of a target discrete variable *at prediction time*. This is important because while training data can often be curated and missingness mechanisms investigated, the same is generally not true for missing data at prediction time. We assume the classifier is represented as a Probabilistic Circuit, and focus on the case of GeFs (although the algorithm we present is slightly more general). The procedure can be used to determine the set of maximal values for the target variable given an observation with data (assumed) MNAR, that is, to decide which values are the most probable classification under some imputation. We also discuss how to enable conservative inferences with non-ignorable data at learning time. Experiments show that our algorithm obtains reliable conclusions often more accurate than criteria that ignores or marginalizes missing variables.

2 Probabilistic Circuits and Generative Random Forests

We start by establishing some notation and terminology. We denote random variables by upper-case letters (e.g., X_i, X), and their values by lower case (e.g., x_i, x). Sets of random variables are written in boldface (e.g., $\mathbf{X}$), as well as their realization (e.g., $\mathbf{x}$). In this work we assume that random variables take on a finite number of values, denoted as $val(X)$ for random variable X. We associate every discrete random variable X with a set of indicator functions $\{[\![X = x]\!] : x \in val(X)\}$, where the notation $[\![X = x]\!]$ describes the function that returns 1 if X takes value x and 0 otherwise.

A Probabilistic Circuit (PC) $\mathtt{M}$ over a set of categorical random variables $\mathbf{X}$ is a rooted weighted acyclic directed graph whose leaves are associated with indicator functions $[\![X_i = x_i]\!]$ of variables in $\mathbf{X}$, and the internal nodes are associated to either sum or product operations. The arcs $i \to j$ leaving sum node i are associated with non-negative weights w_{ij}. We write $\mathtt{M}_i$ to denote the sub-PC rooted at node i . The *scope* of a PC is the set of random variables associated with the indicator variables at the leaves, and the scope of a node is the scope of the respective sub-PC. A PC represents a joint distribution of $\mathbf{X}$ by $P_{\mathtt{M}}(\mathbf{x}) = \mathtt{M}(\mathbf{x})/(\sum_{\mathbf{x}'} \mathtt{M}(\mathbf{x}'))$. The value $\mathtt{M}(\mathbf{x})$, called the evaluation of the circuit at $\mathbf{x}$, is defined inductively in the size of the circuit as: $\mathtt{M}(\mathbf{x}) = [\![X_i = x_i]\!](\mathbf{x}_i)$ if $\mathtt{M}$ is a leaf node $[\![X_i = x_i]\!]$; $\mathtt{M}(\mathbf{x}) = \sum_j w_{ij}\mathtt{M}_j(\mathbf{x})$ if $\mathtt{M}$ is a circuit rooted at a sum node i with children j; and $\mathtt{M}(\mathbf{x}) = \prod_j \mathtt{M}_j(\mathbf{x})$ if $\mathtt{M}$ is a circuit rooted at a product node with children j. For example, the evaluation of the PC on the right-hand side of Figure 1 at $X = 4$, $Y = 1$ and $Z = 1$ is $\mathtt{M}(\mathbf{x}) = 0.6 \times 0.7 \times 0.6 = 0.252$.

To ensure that marginal inference is computed in linear time in the size of the circuit, it suffices that the circuit satisfies the properties of smoothness and decomposability [9, 21]. Smoothness states that the scopes of any two children of a sum node are identical.[3] Decomposability states that the scopes of any two children of a product node are disjoint. For the rest of this paper, *we assume that PCs are smooth and decomposable.* The tractability of more complex probabilistic queries rely on additional properties. One such property is determinism: each sum node has at most one child that evaluates to non-zero at any (complete) realization of its scope.[4] Determinism ensures that maximum likelihood estimates for the weights can be obtained in closed-form under complete data; it also enables finding the most probable realization in linear time [18], a NP-hard task in non-deterministic PCs. The property is also necessary (but not sufficient) for advanced operations such as encoding constraints, computing entropy or KL-divergence, and producing expected predictions with respect to a compatible regressor [7–9].

Consider a Decision Tree mapping features $\mathbf{X}$ to a target variable Y. Generative Decision Trees (GeDT) are deterministic, decomposable and smooth PCs built from such a Decision Tree and data by converting each decision node into a sum node and each leaf into a sub-PC whose support is the partition induced by

[3] Smoothness is also called completeness in the context of Sum-Product Networks.
[4] Determinism is also called selectivity in the context of Sum-Product Networks.

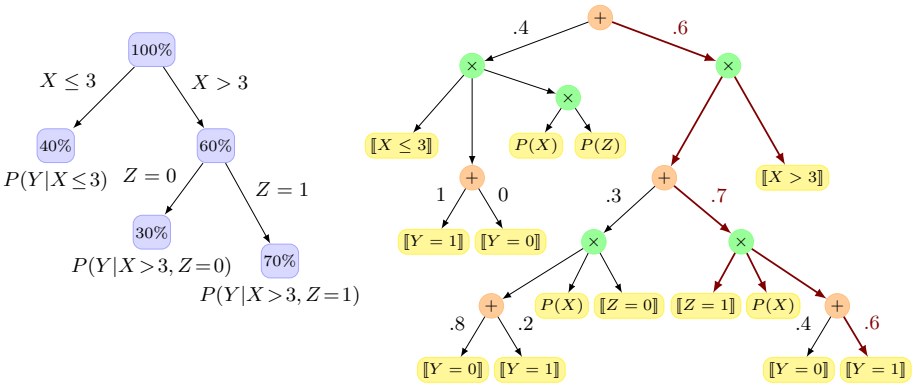


Fig. 1. A Decision Tree for classifying Y based on the values of X and Z (left) and its Generative Decision Tree extension (right).

the corresponding path of the Decision Tree. The sub-PCs at the leaves 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 $P(\mathbf{X}|\gamma)P(Y|\gamma)$, where γ denotes the corresponding partition of the feature space. Such *class-factorized* GeDTs produce the same classifications under complete data as the original Decision Tree [5]. As that property will be crucial to ensure exactness of the procedure we develop later, we generalize it to arbitrary PCs as follows. We say that a PC is class-factorized with respect to a target variable Y if for any leaf whose scope is $\{Y\}$, its parents also have scope $\{Y\}$. Intuitively, in a class-factorized PC the leaves with indicators for variable Y cannot be used to select between different sub-PCs as in the Proof of Theorem 1 we show later.

The structure of a GeDT can be modified by "pulling the indicators up" to speed up computations, that is, by adding product nodes and indicator leaves that encode the partitions of the decision tree nodes. While this procedure violates decomposability, it renders a PC whose evaluations produce identical result (and marginal inference is therefore still linear). Hereafter, we will refer to GeDT as the PC obtained from a Decision Tree after such an operation. Figure 1 shows a decision tree (on the left) and a class-factorized GeDT extension (on the right) obtained by pulling up indicators. The numbers inside each node in the decision tree indicate the percentage of the training data instances that fall in the corresponding partition out of those instances that are in the partition defined by the parent node. Those values (transformed in probabilities) are used to define the weight of the respective outgoing arc in the GeDT.

GeDTs allow for proper and efficient treatment of missing at random data by computing $P(Y|\mathbf{o})$, the conditional probability of the target variable given the observed features $\mathbf{o}$ (which might be a subset of all features). While Decision Trees are consistent (Bayes optimal) estimators only when used in fully observed data, GeDTs are consistent also for missing at random features [5]. A Generative

Random Forest (GeF) is the structure obtained as a mixture model where each component is the GeDT corresponding to a Decision Tree in a Random Forest Classifier. Marginal inference also takes therefore linear time in GeFs.

3 Tractable Conservative Inference

Consider a PC $\mathtt{M}(\mathbf{X})$, possibly learned from some (complete or MAR incomplete) dataset of realizations of variables $\mathbf{X}$. 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. Let $\mathbf{o}$ denote the observed part of $\mathbf{x}$ and $\mathbf{u}$ denote a possible completion for the unobserved values. Under the missing at random hypothesis, this is best performed by computing

$$\arg\max_y P(y|\mathbf{o}) = \arg\max_y \mathtt{M}(y,\mathbf{o})/\mathtt{M}(\mathbf{o})\,, \tag{1}$$

where $\mathtt{M}(y,\mathbf{o}) = \sum_{\mathbf{u}} \mathtt{M}(y,\mathbf{o},\mathbf{u})$ is the marginal value of the circuit at y and $\mathbf{o}$, which can be obtained in linear time as discussed, and $\mathtt{M}(\mathbf{o}) = \sum_y \mathtt{M}(y,\mathbf{o})$.

When the missingness process is non-ignorable, the inference in (1) can lead to erroneous and unreliable conclusions. As an example, consider a Boolean target Y and Boolean features O and U. Say we observe $O = o$, and assume that the value of U guides the prediction being $P(Y=1|o,u_1) = 0.7$ and $P(Y=1|o,u_2) = 0.4$, and that $P(U)$ is uniform. Also, the observation of U is missing due to the following MNAR process: when U is missing, value u_1 is nine times less likely than u_2. The Bayes-optimal classification is thus $Y = 0$ as $P(Y=1|o,U=\star) = 0.7\times 0.1 + 0.4\times 0.9 = 0.43 < 0.5$, yet the marginal classification is $Y = 1$ as $P(Y=1|o) = 0.55 > 0.5$.

Unlike the example above, we rarely have access to the missingness process. We can instead estimate the robustness of a classification $Y = y'$ under non-ignorable missing data with respect to an alternative classification $Y = y''$ by

$$\delta_{\mathtt{M},\mathbf{o}}(y',y'') = \min_{\mathbf{u}} [\mathtt{M}(y',\mathbf{o},\mathbf{u}) - \mathtt{M}(y'',\mathbf{o},\mathbf{u})]\,. \tag{2}$$

A decision analyst might want to suspend the classification on the basis of the value in (2), thus producing more conservative conclusions. For example, if $\delta_{\mathtt{M},\mathbf{o}}(y',y'') > 0$, then any imputation of the values of $\mathbf{u}$ 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''. Note that δ can be a function of $P(y|\mathbf{o})$ and $P(\mathbf{o})$, that is, it can be instance specific and account for other sources of information.

The *conservative inference rule* (CIR) prescribes that the only conclusion supported by non-ignorable missing data is to return the set of non-dominated values [30]:

$$\left\{ y : \max_{y'} \delta_{\mathtt{M},\mathbf{o}}(y',y) \leq 0 \right\}. \tag{3}$$

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 (2) requires evaluating $\mathtt{M}(y, \mathbf{o}, \mathbf{u})$ for each $\mathbf{u}$. 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 deterministic PCs, ruling out the existence of an efficient exact procedure (under common assumptions of complexity theory).

Theorem 1. *Given a smooth, decomposable and deterministic PC* $\mathtt{M}$ *over random variables* Y*,* $\mathbf{O}$ *and* $\mathbf{U}$*, target values* y' *and* y''*, a partial observation* $\mathbf{o}$*, and a (rational) threshold* ρ*, deciding if* $\delta_{\mathtt{M},\mathbf{o}}(y', y'') > \rho$ *is coNP-complete.*

The proof is in the appendix.

We now provide a linear-time algorithm for computing $\delta_{\mathtt{M},\mathbf{o}}(y', y'')$ in tree-shaped deterministic PCs that satisfy class-factorization, which include class-factorized GeDTs. For the sake of readability, we drop the dependence on $\mathbf{o}$ 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.

Sum nodes. If $\mathtt{M}$ is rooted at a sum node with children $\mathtt{M}_1, \ldots, \mathtt{M}_n$ and weights $w_1, \ldots, w_n$, then the algorithm computes:

$$\delta_{\mathtt{M}}(y', y'') = \min_{i=1}^{n} w_i \min_{\mathbf{u}} [\mathtt{M}_i(y', \mathbf{o}, \mathbf{u}) - \mathtt{M}_i(y'', \mathbf{o}, \mathbf{u})] = \min_{i=1}^{n} w_i \delta_{\mathtt{M}_i}(y', y'') \,. \tag{4}$$

The correctness of the operation follows from the determinism of the circuit and the class-factorization property. The former ensures that for any realization $(y, \mathbf{x})$ at most one sub-PC $\mathtt{M}_i$ evaluates to a nonnegative value $\mathtt{M}_i(y, \mathbf{x}) > 0$. The latter ensures that either $\mathtt{M}$ encodes a distribution over Y (i.e., its scope is the singleton $\{Y\}$) or the nonnegative child for $\mathtt{M}_i(y', \mathbf{x})$ and $\mathtt{M}_i(y'', \mathbf{x})$ is the same.

Product nodes. If instead $\mathtt{M}$ is a product node with children $\mathtt{M}_1, \ldots, \mathtt{M}_n$ such that Y is in the scope of $\mathtt{M}_1$ (and no other), then the algorithm computes:

$$\delta_{\mathtt{M}}(y', y'') = \underbrace{\min_{\mathbf{u}_1} [\mathtt{M}_1(y', \mathbf{o}_1, \mathbf{u}_1) - \mathtt{M}_1(y'', \mathbf{o}_1, \mathbf{u}_1)]}_{= \delta_{\mathtt{M}_1}(y', y'')} \prod_{i=2}^{n} \mathrm{opt}_{\mathbf{u}_i} \mathtt{M}_i(\mathbf{o}_i, \mathbf{u}_i) \,, \tag{5}$$

where $\mathbf{o}_i$ (resp., $\mathbf{u}_i$) denotes the projection of $\mathbf{o}$ (resp., $\mathbf{u}$) into the scope of $\mathtt{M}_i$, and

$$\mathrm{opt} = \begin{cases} \max & \text{if } \delta_{\mathtt{M}_1}(y', y'') > 0 \,, \\ \min & \text{if } \delta_{\mathtt{M}_1}(y', y'') \leq 0 \,. \end{cases}$$

The first term denotes the recursive computation on the sub-PC $\mathtt{M}_1$. The remaining terms $\mathrm{opt}_{\mathbf{u}_i} \mathtt{M}_i(\mathbf{o}_i, \mathbf{u}_i)$ define an optimization of the configurations $\mathbf{u}_i$ for the sub-PC $\mathtt{M}_i$; this can be performed in linear time in deterministic PCs by bottom-up traversal, replacing sums with maximizations/minimizations [18, 19].

Leaves. Finally, if $\mathtt{M}$ is a leaf node representing an indicator variable then the algorithm computes:

$$\delta_{\mathtt{M}}(y', y'') = \begin{cases} 1 & \text{if } \mathtt{M} \text{ is } [\![Y = y']\!], \\ -1 & \text{if } \mathtt{M} \text{ is } [\![Y = y'']\!], \\ 1 & \text{if } \mathtt{M} \text{ is consistent with } \mathbf{o} \text{ or } \mathbf{u}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

We thus obtain the following result.

Theorem 2. *The algorithm obtained by Equations* (4)*,* (5) *and* (6) *computes* $\delta_{\mathtt{M},\mathbf{o}}(y', y'')$ *in class-factorized tree-shaped deterministic PCs in linear time.*

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

$$\min_{\mathbf{u}} \sum_{i=1}^{n} w_i [\mathtt{M}_i(y', \mathbf{o}, \mathbf{u}) - \mathtt{M}_i(y'', \mathbf{o}, \mathbf{u})] \neq \min_{\mathbf{u}} w_i \min_{i=1}^{n} [\mathtt{M}_i(y', \mathbf{o}, \mathbf{u}) - \mathtt{M}_i(y'', \mathbf{o}, \mathbf{u})] \,.$$

The equations for products and leaves remain valid for non-deterministic circuits. Thus, we can use our algorithm as an effective heuristic for non-deterministic PCs or that violate class-factorization. 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 deterministic 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.

4 Non-Ignorable Training Data

In the previous section we considered PCs learned from complete (or MAR incomplete) datasets, while restricting the presence of MNAR data to prediction time. Yet, we might also have non-ignorable missingness in the training dataset and, following the ideas outlined by [28], apply the same conservative treatment to the learning of the PC. As a first step, in this work we consider that the structure of the circuit (i.e., its directed graph) is either specified in a data-free fashion (e.g., using region graphs or random structures), or learned by standard algorithms using only the complete portion of the data set [20, 21]. The latter is a sensible choice when the missing values do not significantly alter the (context-specific) independences in the data, but can affect the quantification of the weights in a significant form.

Say that a PC structure is uniformly deterministic, if any quantification assigning positive weights leads to a deterministic PC. Thus assume that we have a fixed uniformly deterministic PC structure that we need to quantify using incomplete data. For complete data, the maximum likelihood estimates of

the weights associated to a sum node can be obtained as

$$w_{ij} = \frac{N_{ij} + \alpha}{\sum_j N_{ij} + \alpha},$$

where N_{ij} counts the number of instances of the dataset for which the sub-PC $\mathtt{M}_j$ contributed to the PC value and α is a smoothing factor to counter the effect of small sample sizes.[5] Clearly, each possible completion of the non-ignorable missing values induces a different quantification of the weights using the formula above. This leads to the specification of a *credal* PC, whose weights are not specified by sharp numerical values, but only required to satisfy a finite number of linear constraints [16]. Standard inference in such models is therefore intended as the computation of the lower and upper bounds of the query with respect to all the possible specifications of the weights consistent with the constraints. A simple strategy is to obtain interval-valued weights:

$$0 \leq \underline{w}_{ij} \leq w_{ij} \leq \overline{w}_{ij} \leq 1, \quad \sum_{i \to j} w_{ij} = 1\,. \tag{7}$$

For deterministic PCs, the lower bound $\underline{w}_{ij}$ can be approximated as the count N_{ij} over instances which have no missing values for the scope of the corresponding node, and the upper bound $\overline{w}_{ij}$ is obtained by assuming that all completions will satisfy $\mathtt{M}_j > 0$ and hence contribute to the corresponding count N_{ij}. Note that those bounds are loose as they ignore the dependencies among different parameters. An alternative technique to learn non-deterministic credal PCs in the presence of incomplete data was recently proposed by [10]. We could also resort to their approach for a CIR-based training of credal PCs. We leave as future work adapting their results to learning deterministic credal PCs.

Assessing the robustness with interval-valued credal PCs amounts to computing:

$$\delta_{\mathtt{M}}(y', y'') = \min_{\mathbf{w}} \min_{\mathbf{u}} [\mathtt{M}_{\mathbf{w}}(y', \mathbf{o}, \mathbf{u}) - \mathtt{M}_{\mathbf{w}}(y'', \mathbf{o}, \mathbf{u})]\,, \tag{8}$$

where the notation $\mathtt{M}_{\mathbf{w}}$ denotes a PC quantified by weights $\mathbf{w}$.

The algorithm for deciding dominance can be easily adapted for handling class-factorized deterministic tree-shaped credal PCs. If $\mathtt{M}$ is rooted at a sum node with children $\mathtt{M}_1, \ldots, \mathtt{M}_n$ and weights $w_1, \ldots, w_n$, then the algorithm computes

$$\delta_{\mathtt{M}}(y', y'') = \min_{i=1}^{n} w_i \min_{\mathbf{w}, \mathbf{u}} [\mathtt{M}_i(y', \mathbf{o}, \mathbf{u}) - \mathtt{M}_i(y'', \mathbf{o}, \mathbf{u})]\,, \tag{9}$$

where $w_i = \underline{w}_i$ if the inner minimization is positive, and $w_i = \overline{w}_i$ if the inner minimization is negative. Similarly, if $\mathtt{M}$ is a product node with children $\mathtt{M}_1, \ldots, \mathtt{M}_n$ such that Y is in the scope of $\mathtt{M}_1$ (and no other), then the algorithm computes:

$$\delta_{\mathtt{M}}(y', y'') = \underbrace{\min_{\mathbf{w}_1, \mathbf{u}_1} [\mathtt{M}_1(y', \mathbf{o}_1, \mathbf{u}_1) - \mathtt{M}_1(y'', \mathbf{o}_1, \mathbf{u}_1)]}_{= \delta_{\mathtt{M}_1}(y', y'')} \prod_{i=2}^{n} \operatorname{opt}_i \mathtt{M}_i(\mathbf{o}_i, \mathbf{u}_i)\,, \tag{10}$$

[5] By *contributing* to the PC value, we mean that there is path from the root to $\mathtt{M}_j$ where each node evaluates to a positive value for the given instance.

where

$$\mathrm{opt}_i = \begin{cases} \max_{\mathbf{w}_i,\mathbf{u}_i} & \text{if } \delta_{\mathtt{M}_1}(y', y'') > 0\,, \\ \min_{\mathbf{w}_i,\mathbf{u}_i} & \text{if } \delta_{\mathtt{M}_1}(y', y'') \leq 0\,. \end{cases}$$

The sub-problems $\mathrm{opt}_i\, \mathtt{M}_i(\mathbf{o}_i, \mathbf{u}_i)$ can be computed in linear time by the algorithm described in [12]. The equations for the leaves remain unchanged. We have that:

Theorem 3. *The algorithm obtained by Equations* (9)*,* (10) *and* (6) *computes* $\delta_{\mathtt{M},\mathbf{o}}(y', y'')$ *in class-factorized tree-shaped deterministic credal PCs in linear time.*

5 Experiments

We empirically evaluate the ability of our proposed methods in assessing the robustness of classifications to non-ignorable missing feature values, by means of the index δ. To this end, we learn class-factorized GeFs from six well-known complete binary datasets for density estimation [6], using the algorithm in [5]. The characteristics of the datasets are in Table 1. Missing test values are simulated using a mix of MAR, MCAR and MNAR mechanisms. 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.

Table 1. Datasets characteristics.

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
Msnbc	17	5,624	1.6	0.5	291,326	2,816
Mushrooms	112	5,624	7.7	3.4	2,000	1,764
Netflix	100	3,000	6.7	3.0	15,000	3,524
Nltcs	16	3,236	1.4	0.4	16,181	568

In Table 2 we report relevant performance metrics of our CIR 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 CIR inference (Eq. 3) 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 $\%R$ shows the percentage of robust instances. By comparing *RAcc*, *Acc* and $\neg$*RAcc*, we observe the ability of CIR 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 [29] 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.

Table 2. Set Accuracy ($SAcc$), Discounted Accuracy ($DAcc$), percentage of robust instances ($\%R$), and classification accuracy on robust ($RAcc$), non-robust ($\neg RAcc$) and overall (Acc) instances when marginalizing missing values at prediction time.

Dataset	CIR			Marginalization		
	$SAcc$	$DAcc$	$\%R$	$RAcc$	$\neg RAcc$	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	0.763
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	0.806
Msnbc	0.978	0.956	95.5	0.978	0.925	0.976
Msnbc (marg)	0.979	0.932	90.6	0.978	0.956	0.976
Mushrooms	1.000	0.991	98.2	1.000	1.000	1.000
Mushrooms (marg)	1.000	0.991	98.2	1.000	1.000	1.000
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	0.716
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	0.961

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.[6] This is an 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 probability ($p = 0.05$) or high probability ($p = 0.5$). This simulates observed behaviour of users providing ratings in such systems [14]. Table 3 shows that learning an imprecise model relay 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 2 shows that for both missingness levels the measure in (2) 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 3, where this approach is combined with a rejection option.

[6] `http://eigentaste.berkeley.edu/dataset`

Table 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 Section 4, precise models are obtained after removal of instances with missing values.

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

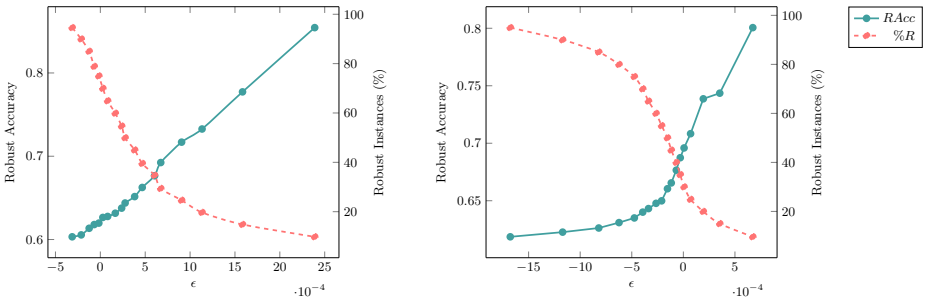


Fig. 2. Robust accuracy (*RAcc*) of the precise classifier (MAR) for the Jester dataset with low ($p = 0.05$, left) and high ($p = 0.5$, right) missingness levels. Condition $\delta_{\mathtt{M},\mathtt{o}}(y, \neg y) > \epsilon$, where $\delta_{\mathtt{M},\mathtt{o}}$ is defined as in Eq. (2) and y is the class returned by the classifier, is used to decide robustness. We also display $\%R$ by threshold ϵ.

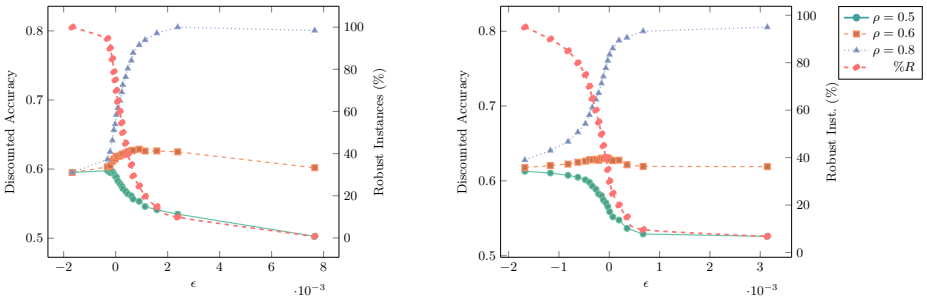


Fig. 3. Modified discounted accuracy of the (imprecise) classifier for the Jester dataset with low ($p = 0.05$, left) and high ($p = 0.5$, right) missingness levels. Robustness is decided as in Figure 2 for different values of the threshold ϵ. A value ρ is used instead of 0.5 to score imprecise classifications, which regulates preference for model uncertainty against aleatory uncertainty (see [29]).

6 Conclusion

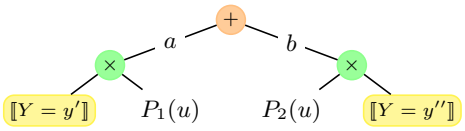
We developed an exact polynomial-time method for the conservative treatment of non-ignorable missing data using probabilistic circuits. Experiments with realistic data demonstrated that the approach is effective in discriminating instances which are sensitive to the missingness process from those that are not. Our approach to handling missing data at training time led us to consider *credal* circuits, which extend standard probabilistic circuits by locally associating sets of probabilities to sum nodes. Such extensions retain many of the tractable properties of probabilistic circuits, offering an interesting and more cautious alternative to marginal inference. We left as future work the treatment of other types of missing data (e.g., coarse and unreliable observations).

Proof of Theorem 1

Membership in coNP is trivial: given a configuration $\mathbf{u}$ we can compute $\mathtt{M}(y', \mathbf{o}, \mathbf{u})$ and $\mathtt{M}(y'', \mathbf{o}, \mathbf{u})$ 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

$$\exists u \in \{0,1\}^n : \sum_{i \in [n]} v_i u_i = 1\,, \quad \text{where } v_i = \frac{2z_i}{\sum_{i \in [n]} z_i}\,. \tag{11}$$



To solve that problem, build a tree-shaped deterministic PC as shown above, where U_i are binary variables, $P_1(u) = \prod_i e^{-2v_i u_i}$ and $P_2(u) = \prod_i e^{-v_i u_i}$. Note that the PC is not class-factorized. Use the PC to compute:

$$\delta(y', y'') = \min_u \left[a \exp\left(-2 \sum_i v_i u_i\right) - b \exp\left(-\sum_i v_i u_i\right) \right]\,.$$

If we call $x := \exp(-\sum_i v_i u_i)$, the above expression is the minimum for positive x of $f(x) := ax^2 - bx$. Function f is a strictly convex function minimized at $x = b/(2a)$. Selecting a and b such that $b/(2a) = e^{-1}$ makes the minimum occur at $\sum_i v_i u_i = 1$. Thus, there is a solution to (11) if and only if $\delta(y', y'') \leq -ae^{-2}$. 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 [16] 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. □

References

1. Antonucci, A., Piatti, A.: Modeling unreliable observations in Bayesian networks by credal networks. In: Proceedings of the Third International Conference on Scalable Uncertainty Management (SUM). pp. 28–39 (2009)
2. Antonucci, A., Zaffalon, M.: Decision-theoretic specification of credal networks: a unified language for uncertain modeling with sets of Bayesian networks. International Journal of Approximate Reasoning **49**(2), 345–361 (2008)
3. Azur, M.J., Stuart, E.A., Frangakis, C., Leaf, P.J.: Multiple imputation by chained equations: what is it and how does it work? International Journal of Methods in Psychiatric Research **20** (2011)
4. Choi, Y., Vergari, A., Van den Broeck, G.: Probabilistic circuits: A unifying framework for tractable probabilistic models (2020)
5. Correia, A.H.C., Peharz, R., de Campos, C.P.: Joints in random forests. In: Advances in Neural Information Processing Systems 33 (NeurIPS) (2020)
6. Davis, J., Domingos, P.: Bottom-up learning of Markov network structure. In: Proceedings of the 27th International Conference on Machine Learning (ICML). pp. 271–280 (2010)
7. Khosravi, P., Choi, Y., Liang, Y., Vergari, A., Van den Broeck, G.: On tractable computation of expected predictions. In: Advances in Neural Information Processing Systems 32 (NeurIPS) (2019)
8. Khosravi, P., Liang, Y., Choi, Y., Van den Broeck, G.: What to expect of classifiers? reasoning about logistic regression with missing features. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI) (2019)
9. Kisa, D., Van den Broeck, G., Choi, A., Darwiche, A.: Probabilistic sentential decision diagrams. In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (PKDD). pp. 1–10 (2014)
10. Levray, A., Belle, V.: Learning credal sum-product networks. In: Proceedings of the 2nd Conference on Automated Knowledge Base Construction (2020)
11. Liang, Y., Van den Broeck, G.: Learning logistic circuits. In: Proceedings of the 33rd Conference on Artificial Intelligence (AAAI) (2019)
12. Llerena, J.V., Mauá, D.D.: Efficient algorithms for robustness analysis of maximum a posteriori inference in selective sum-product networks. International Journal of Approximate Reasoning **126**, 158–180 (2020)
13. Manski, C.F.: Partial identification with missing data: concepts and findings. International Journal of Approximate Reasoning **39**(2-3), 151–165 (2005)
14. Marlin, B.M., Zemel, R.S., Roweis, S.T., Slaney, M.: Recommender systems: Missing data and statistical model estimation. In: Proceedings of the 22nd International Joint Conference in Artificial Intelligence (IJCAI) (2011)
15. Mauá, D.D., De Campos, C.P., Benavoli, A., Antonucci, A.: Probabilistic inference in credal networks: new complexity results. Journal of Artificial Intelligence Research **50**, 603–637 (2014)
16. Mauá, D.D., Conaty, D., Cozman, F.G., Poppenhaeger, K., de Campos, C.P.: Robustifying sum-product networks. International Journal of Approximate Reasoning **101**, 163–180 (2018)
17. Mohan, K., Pearl, J., Tian, J.: Graphical models for inference with missing data. In: Proceedings of Advances in neural information processing systems (NeurIPS). pp. 1277–1285 (2013)
18. Peharz, R., Gens, R., Domingos, P.: Learning selective sum-product networks. In: Proceedings of the Workshop on Learning Tractable Probabilistic Models (2014)

19. Peharz, R., Gens, R., Pernkopf, F., Domingos, P.: On the latent variable interpretation in sum-product networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(10), 2030–2044 (2017)
20. Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., Ghahramani, Z.: 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)
21. Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI). pp. 337–346 (2011)
22. Rahman, T., Kothalkar, P., Gogate, V.: Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD). pp. 630–645 (2014)
23. Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976)
24. Shao, X., Alejandro Molina, A.V., Stelzner, K., Peharz, R., Liebig, T., Kersting, K.: Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In: Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM) (2020)
25. Shen, Y., Choi, A., Darwiche, A.: A tractable probabilistic model for subset selection. In: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI) (2017)
26. Shen, Y., Goyanka, A., Darwiche, A., Choi, A.: Structured Bayesian networks: From inference to learning with routes. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI) (2019)
27. Shin, J., Wu, S., Wang, F., Sa, C.D., Zhang, C., Ré, C.: Incremental knowledge base construction using deepdive. In: Proceedings of the VLDB Endowment (2015)
28. Zaffalon, M.: Conservative rules for predictive inference with incomplete data. In: Proceedings of the 4th International Symposium on Imprecise Probabilities and Their Applications (ISIPTA). pp. 406–415 (2005)
29. Zaffalon, M., Corani, G., Mauá, D.: Evaluating credal classifiers by utility-discounted predictive accuracy. International Journal of Approximate Reasoning **53**(8), 1282–1301 (2012)
30. Zaffalon, M., Miranda, E.: Conservative inference rule for uncertain reasoning under incompleteness. Journal of Artificial Intelligence Research **34**, 757–821 (2009)
31. Zheng, K., Pronobis, A., Rao, R.P.N.: Learning graph-structured sum-product networks for probabilistic semantic maps. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI) (2018)