Discriminative Bias for Learning Probabilistic Sentential Decision Diagrams

Laura Isabel Galindez Olascoaga[1](✉), Wannes Meert[2], Nimish Shah[1], Guy Van den Broeck[3], and Marian Verhelst[1]

[1] Electrical Engineering Department, KU Leuven, Leuven, Belgium
{laura.galindez,nimish.shah,marian.verhelst}@esat.kuleuven.be
[2] Computer Science Department, KU Leuven, Leuven, Belgium
wannes.meert@cs.kuleuven.be
[3] Computer Science Department, University of California, Los Angeles, USA
guyvdb@cs.ucla.edu

Abstract. Methods that learn the structure of Probabilistic Sentential Decision Diagrams (PSDD) from data have achieved state-of-the-art performance in tractable learning tasks. These methods learn PSDDs incrementally by optimizing the likelihood of the induced probability distribution given available data and are thus robust against missing values, a relevant trait to address the challenges of embedded applications, such as failing sensors and resource constraints. However PSDDs are outperformed by discriminatively trained models in classification tasks. In this work, we introduce D-LearnPSDD, a learner that improves the classification performance of the LearnPSDD algorithm by introducing a discriminative bias that encodes the conditional relation between the class and feature variables.

Keywords: Probabilistic models · Tractable inference · PSDD

1 Introduction

Probabilistic machine learning models have shown to be a well suited approach to address the challenges inherent to embedded applications, such as the need to handle uncertainty and missing data [11]. Moreover, current efforts in the field of Tractable Probabilistic Modeling have been making great strides towards successfully balancing the trade-offs between model performance and inference efficiency: probabilistic circuits, such as Probabilistic Sentential Decision Diagrams (PSDDs), Sum-Product Networks (SPNs), Arithmetic Circuits (ACs) and Cutset Networks, posses myriad desirable properties [4] that make them amenable to application scenarios where strict resource budget constraints must be met [12]. But these models' robustness against missing data—from learning them generatively—is often at odds with their discriminative capabilities.

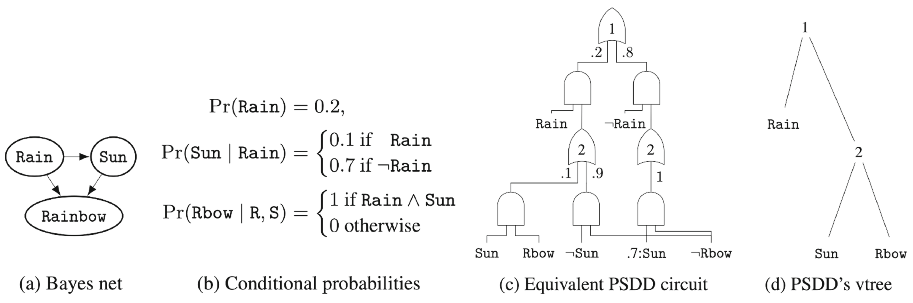
We address such a conflict by proposing a discriminative-generative probabilistic circuit learning strategy, which aims to improve the models' discriminative capabilities, while maintaining their robustness against missing features.

We focus in particular on the PSDD [17], a state-of-the-art tractable representation that encodes a joint probability distribution over a set of random variables. Previous work [12] has shown how to learn hardware-efficient PSDDs that remain robust to missing data and noise. This approach relies largely on the LearnPSDD algorithm [20], a generative algorithm that incrementally learns the structure of a PSDD from data. Moreover, it has been shown how to exploit such robustness to trade off resource usage with accuracy. And while the achieved accuracy is competitive when compared to Bayesian Network classifiers, discriminatively learned models perform consistently better than purely generative models [21] since the latter remain agnostic to the discriminative task they ought to perform. This begs the question of whether the discriminative performance of the PSDD could be improved while remaining robust and tractable.

In this work, we propose a hybrid discriminative-generative PSDD learning strategy, D-LearnPSDD, that enforces the discriminative relationship between class and feature variables by capitalizing on the model's ability to encode domain knowledge as a logic formula. We show that this approach consistently outperforms the purely generative PSDD and is competitive compared to other classifiers, while remaining robust to missing values at test time.

2 Background

Notation. Variables are denoted by upper case letters X and their instantiations by lower case letters x. Sets of variables are denoted in bold upper case $\mathbf{X}$ and their joint instantiations in bold lower case $\mathbf{x}$. For the classification task, the feature set is denoted by $\mathbf{F}$ while the class variable is denoted by C.



Rain
Sun
Rainbow
(a) Bayes net
Pr(Rain) = 0.2,
Pr(Sun | Rain) = {0.1 if Rain; 0.7 if ¬Rain
Pr(Rbow | R, S) = {1 if Rain ∧ Sun; 0 otherwise
(b) Conditional probabilities
1
.2
.8
Rain
¬Rain
2
.1
.9
2
1
Sun
Rbow
¬Sun
.7:Sun
¬Rbow
(c) Equivalent PSDD circuit
1
Rain
2
Sun
Rbow
(d) PSDD's vtree

Fig. 1. A Bayesian network and its equivalent PSDD (taken from [20]).

PSDD. Probabilistic Sentential Decision Diagrams (PSDDs) are circuit representations of joint probability distributions over binary random variables [17]. They were introduced as probabilistic extensions to Sentential Decision Diagrams (SDDs) [7], which represent Boolean functions as logical circuits. The inner nodes of a PSDD alternate between AND gates with two inputs and OR gates with arbitrary number of inputs; the root must be an OR node; and each leaf node encodes a distribution over a variable X (see Fig. 1c). The combination of an OR gate with its AND gate inputs is referred to as *decision* node, where the left input of the AND gate is called *prime* (p), and the right is called *sub* (s). Each of the n edges of a decision node are annotated with a normalized probability distribution $\theta_1, ..., \theta_n$.

PSDDs possess two important syntactic restrictions: (1) Each AND node must be *decomposable*, meaning that its input variables must be disjoint. This property is enforced by a *vtree*, a binary tree whose leaves are the random variables and which determines how will variables be arranged in primes and subs in the PSDD (see Fig. 1d): each internal vtree node is associated with the PSDD nodes at the same level, variables appearing in the left subtree $\mathbf{X}$ are the primes and the ones appearing in the right subtree $\mathbf{Y}$ are the subs. (2) Each decision node must be *deterministic*, thus only one of its inputs can be true.

Each PSDD node q represents a probability distribution. Terminal nodes encode a univariate distributions. Decision nodes, when normalized for a vtree node with $\mathbf{X}$ in its left subtree and $\mathbf{Y}$ in its right subtree, encode the following distribution over $\mathbf{XY}$ (see also Fig. 1a and c):

$$Pr_q(\mathbf{XY}) = \sum_i \theta_i Pr_{p_i}(\mathbf{X}) Pr_{s_i}(\mathbf{Y}) \tag{1}$$

Thus, each decision node decomposes the distribution into independent distributions over $\mathbf{X}$ and $\mathbf{Y}$. In general, prime and sub variables are independent at PSDD node q given the prime *base* $[q]$ [17]. This base is the support of the node's distribution, over which it defines a non-zero probability and it is written as a logical sentence using the recursion $[q] = \bigvee_i [p_i] \wedge [s_i]$. Kisa et al. [17] show that prime and sub variables are independent in PSDD node q given a prime base:

$$\begin{aligned} Pr_q(\mathbf{XY}|[p_i]) &= Pr_{p_i}(\mathbf{X}|[p_i]) Pr_{s_i}(\mathbf{Y}|[p_i]) \\ &= Pr_{p_i}(\mathbf{X}) Pr_{s_i}(\mathbf{Y}) \end{aligned} \tag{2}$$

This equation encodes *context specific independence* [2], where variables (or sets of variables) are independent given a logical sentence. The structural constraints of the PSDD are meant to exploit such independencies, leading to a representation that can answer a number of complex queries in polynomial time [1], which is not guaranteed when performing inference on Bayesian Networks, as they don't encode and therefore can't exploit such local structures.

LearnPSDD. The LearnPSDD algorithm [20] generatively learns a PSDD by maximizing log-likelihood given available data. The algorithm starts by learning a *vtree* that minimizes the mutual information among all possible sets of variables. This vtree is then used to guide the PSDD structure learning stage, which relies on the iterative application of the Split and Clone operations [20]. These operations keep the PSDD syntactically sound while improving likelihood of the distribution represented by the PSDD. A problem with LearnPSDD when using the resulting model for classification is that when the class variable is only weakly dependent on the features, the learner may choose to ignore that dependency, potentially rendering the model unfit for classification tasks.

3 A Discriminative Bias for PSDD Learning

Generative learners such as LearnPSDD optimize the likelihood of the distribution given available data rather than the conditional likelihood of the class variable C given a full set of feature variables $\mathbf{F}$. As a result, their accuracy is often worse than that of simple models such as Naive Bayes (NB), and its close relative Tree Augmented Naive Bayes (TANB) [12], which perform surprisingly well on classification tasks even though they encode a simple—or naive—structure [10]. One of the main reasons for their performance, despite being generative, is that (TA)NB models have a discriminative bias that directly encodes the conditional dependence of all the features on the class variable.

We introduce D-LearnPSDD, an extension to LearnPSDD based on the insight that the learned model should satisfy the "class conditional constraint" present in Bayesian Network classifiers. That is, all feature variables must be conditioned on the class variable. This enforces a structure that is beneficial for classification while still allowing to generatively learn a PSDD that encodes the distribution over all variables using a state-of-the-art learning strategy [20].

3.1 Discriminative Bias

The classification task can be stated as a probabilistic query:

$$\Pr(C|\mathbf{F}) \sim \Pr(\mathbf{F}|C) \cdot \Pr(C). \tag{3}$$

Our goal is to learn a PSDD whose root decision node directly represents the conditional probability distribution $\Pr(\mathbf{F}|C)$. This can be achieved by forcing the primes of the first line in Eq. 2 to be $[p_0] = [\neg c]$ and $[p_1] = [c]$, where $[c]$ states that the propositional variable c representing the class variable is true (i.e. $C = 1$), and similarly $[\neg c]$ represents $C = 0$. For now we assume the class is binary and will show later how to generalize to a multi-valued class variable. For the feature variables we can assume they are binary without loss of generality since a multi-valued variable can be converted to a set of binary variables via a one-hot encoding (see, for example [20]). To achieve our goal we first need the following proposition:

Proposition 1. *Given (i) a vtree with a single variable C as the prime and variables $\mathbf{F}$ as the sub of the root node, and (ii) an initial PSDD where the root decision node decomposes the distribution as $[root] = ([p_0] \wedge [s_0]) \vee ([p_1] \wedge [s_1])$; applying the Split and Clone operators will never change the root decision decomposition $[root] = ([p_0] \wedge [s_0]) \vee ([p_1] \wedge [s_1])$.*

Proof. The D-LearnPSDD algorithm iteratively applies two operations: Clone and Split (following the algorithm in [20]). First, the Clone operator requires a parent node, which is not available for the root node. Since the initial PSDD follows the logical formula described above, whose only restriction is on the root node, there is no parent available to clone and the root's base thus remains intact when applying the Clone operator. Second, the Split operator splits one of the subs to extend the sentence that is used to mutually exclusively and exhaustively define all children. Since the given vtree has only one variable, C, as the prime of the root node, there are no other variables available to add to the sub. The Split operator cant thus not be applied anymore and the root's base stays intact (see Figs. 1c and d).

We can now show that the resulting PSDD contains nodes that directly represent the distribution $\Pr(\mathbf{F}|C)$.

Proposition 2. *A PSDD of the form* $[root] = ([\neg c] \wedge [s_0]) \vee ([c] \wedge [s_1])$ *with* c *the propositional variable stating that the class variable is true, and* s_0 *and* s_1 *any formula with propositional feature variables* $f_0, \ldots, f_n$*, directly expresses the distribution* $\Pr(\mathbf{F}|C)$*.*

Proof. Applying this to Eq. 1 results in:

$$\begin{aligned}\Pr_q(C\mathbf{F}) &= \Pr_{\neg c}(C)\Pr_{s_0}(\mathbf{F}) + \Pr_c(C)\Pr_{s_1}(\mathbf{F})\\ &= \Pr_{\neg c}(C|[\neg c]) \cdot \Pr_{s_0}(\mathbf{F}|[\neg c]) + \Pr_c(C|[c]) \cdot \Pr_{s_1}(\mathbf{F}|[c])\\ &= \Pr_{\neg c}(C=0) \cdot \Pr_{s_0}(\mathbf{F}|C=0) + \Pr_c(C=1) \cdot \Pr_{s_1}(\mathbf{F}|C=1)\end{aligned}$$

The learned PSDD thus contains a node s_0 with distribution $\Pr_{s_0}$ that directly represents $\Pr(\mathbf{F}|C=0)$ and a node s_1 with distribution $\Pr_{s_1}$ that represents $\Pr(\mathbf{F}|C=1)$. The PSDD thus encodes $\Pr(\mathbf{F}|C)$ directly because the two possible value assignments of C are $C=0$ and $C=1$.

The following examples illustrate why both the specific vtree and initial PSDD are required.

Example 1. Figure 2b shows a PSDD that encodes a fully factorized probability distribution normalized for the vtree in Fig. 2a. The PSDD shown in this example initializes the incremental learning procedure of LearnPSDD [20]. Note that the vtree does not connect the class variable C to all feature variables (e.g. F_1). Therefore, when initializing the algorithm on this vtree-PSDD combination, there are no guarantees that the conditional relations between certain features and the class will be learned.

Example 2. Figure 2e shows a PSDD that explicitly conditions the feature variables on the class variables by normalizing for the vtree in Fig. 2c and by following the logical formula from Proposition 2. This biased PSDD is then used to initialize the D-LearnPSDD learner. Note that the vtree in Fig. 2c forces the prime of the root node to be the class variable C.

Example 3. Figure 2d shows, however, that only setting the vtree in Fig. 2c is not sufficient for the learner to condition the features on the class. When initializing on a PSDD that encodes a fully factorized formula, and then applying the Split and Clone operators, the relationship between the class variable and the features are not guaranteed to be learned. In this worst case scenario, the learned model could have an even worse performance than the case from Example 1. By applying Eq. 1 on the top split, we can give intuition why this is the case:

$$\begin{aligned}\Pr_q(C\mathbf{F}) &= \Pr_{p_0}(C|[c \vee \neg c]) \cdot \Pr_{s_0}(\mathbf{F}|[c \vee \neg c]) \\ &= (\Pr_{p_1}(C|[c]) + \Pr_{p_2}(C|[\neg c])) \cdot \Pr_{s_0}(\mathbf{F}|[c \vee \neg c]) \\ &= (\Pr_{p_1}(C = 1) + \Pr_{p_2}(C = 0)) \cdot \Pr_{s_0}(\mathbf{F})\end{aligned}$$

The PSDD thus encodes a distribution that assumes that the class variable is independent from all feature variables. While this model might still have a high likelihood, its classification accuracy will be low.

We have so far introduced the D-LearnPSDD for a binary classification task. However, it can be easily generalized to an n-valued classification scenario: (1) The class variable C will be represented by multiple propositional variables $c_0, c_1, \ldots, c_n$ that represent the set $C = 0, C = 1, \ldots, C = n$, of which exactly one will be true at all times. (2) The vtree in Proposition 1 now starts as a right-linear tree over $c_0, \ldots, c_n$. The $\mathbf{F}$ variables are the sub of the node that has c_n as prime. (3) The initial PSDD in Proposition 2 now has a root the form $[root] = \bigvee_{i=0\ldots n}([c_i \bigwedge_{j:0\ldots n \wedge i \neq j} \neg c_j] \wedge [s_i])$, which remains the same after applying Split and Clone. The root decision node now represents the distribution $\Pr_q(C\mathbf{F}) = \sum_{i:0\ldots n} \Pr_{c_i \bigwedge_{j\neq i} \neg c_j}(C = i) \cdot \Pr_{s_i}(\mathbf{F}|C = i)$ and therefore has nodes at the top of the tree that directly represent the discriminative bias.

3.2 Generative Bias

Learning the distribution over the feature variables is a generative learning process and we can achieve this by applying the Split and Clone operators in the same way as the original LearnPSDD algorithm. In the previous section we had not yet defined how should $\Pr(\mathbf{F}|C)$ from Proposition 2 be represented in the initial PSDD, we only explained how our constraint enforces it. So the question is how do we exactly define the nodes corresponding to s_0 and s_1 with distributions $\Pr(\mathbf{F}|C = 0)$ and $\Pr(\mathbf{F}|C = 1)$? We follow the intuition behind (TA)NB and start with a PSDD that encodes a distribution where all feature variables are independent given the class variable (see Fig. 2e). Next, the LearnPSDD algorithm will incrementally learn the relations between the feature variables by applying the Split and Clone operations following the approach in [20].

3.3 Obtaining the Vtree

In learnPSDD, the decision nodes decompose the distribution into independent distributions. Thus, the vtree is learned from data by maximizing the approximate pairwise mutual information, as this metric quantifies the level of independence between two sets of variables. For D-LearnPSDD we are interested in the level of conditional independence between sets of feature variables given the class variable. We thus obtain the vtree by optimizing for Conditional Mutual Information instead and replace mutual information in the approach in [20] with: $CMI(\mathbf{X},\mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{x}}\sum_{\mathbf{y}}\sum_{\mathbf{z}} \Pr(\mathbf{xy}) \log \frac{\Pr(\mathbf{z})\Pr(\mathbf{xyz})}{\Pr(\mathbf{xz})\Pr(\mathbf{yz})}$.

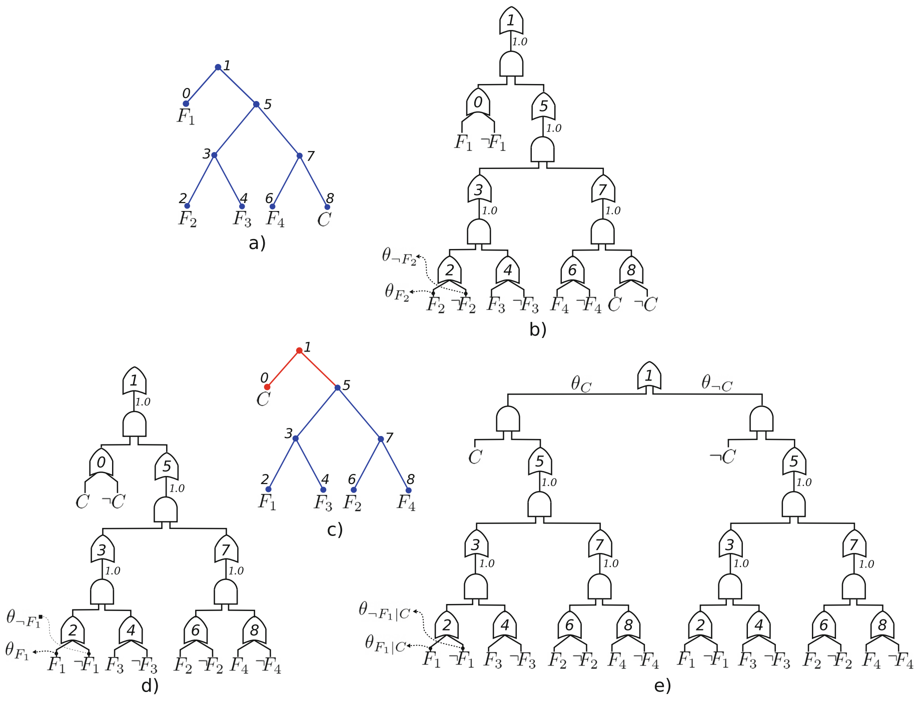


Fig. 2. Examples of vtrees and initial PSDDs.

4 Experiments

We compare the performance of D-LearnPSDD, LearnPSDD, two generative Bayesian classifiers (NB and TANB) and a discriminative classifier (logistic regression). In particular, we discuss the following research queries: (1) Sect. 4.2 examines whether the introduced discriminative bias improves classification performance on PSDDs. (2) Sect. 4.3 analyzes the impact of the vtree and the imposed structural constraints on model tractability and performance. (3) Finally, Sect. 4.4 compares the robustness to missing values for all classification approaches.

Table 1. Datasets

Dataset	$\|\mathbf{F}\|$	$\|C\|$	$\|N\|$
Australian	40	2	690
Breast	28	2	683
Chess	39	2	3196
Cleve	25	2	303
Corral	6	2	160
Credit	42	2	653
Diabetes	11	2	768
German	54	2	1000
Glass	17	6	214
Heart	9	2	270
Iris	12	3	150
Mofn	10	2	1324
Pima	11	2	768
Vehicle	57	2	846
Waveform	109	3	5000

4.1 Setup

We ran our experiments on the suite of 15 standard machine learning benchmarks listed in Table 1. All of the datasets come from the UCI machine learning repository [8], with exception of "Mofn" and "Corral" [18]. As pre-processing steps, we applied the discretization method described in [9], and we binarized all variables using a one-hot encoding. Moreover, we removed instances with missing values and features whose value was always equal to 0. Table 1 summarizes the number of binary features $|\mathbf{F}|$, the number of classes $|C|$ and the available number of training samples $|\mathrm{N}|$ per dataset.

4.2 Evaluation of DG-LearnPSDD

Table 2 compares D-LearnPSDD, LearnPSDD, Naive Bayes (NB), Tree Augmented Naive Bayes (TANB) and logistic regression (LogReg)[1] in terms of accuracy via five fold cross validation[2]. For LearnPSDD, we incrementally learned a model on each fold until convergence on validation-data log-likelihood, following the methodology in [20].

For D-LearnPSDD, we incrementally learned a model on each fold until likelihood converged but then selected the incremental model with the highest training set accuracy. For NB and TANB, we learned a model per fold and compiled them to Arithmetic Circuits[3], a more general form of PSDDs [6], which allows us to compare the size of these Bayes net classifiers and the PSDDs. Finally, we compare all probabilistic models with a discriminative classifier, a multinomial logistic regression model with a ridge estimator.

Table 2 shows that the proposed D-LearnPSDD clearly benefits from the introduced discriminative bias, outperforming LearnPSDD in all but two datasets, as the latter method is not guaranteed to learn significant relations between feature and class variables. Moreover, it outperforms Bayesian classifiers in most benchmarks, as the learned PSDDs are more expressive and allow to encode complex relationships among sets of variables or local dependencies such as context specific independence, while remaining tractable. Finally, note that the D-LearnPSDD is competitive in terms of accuracy with respect to logistic regression (LogReg) a purely discriminative classification approach.

4.3 Impact of the Vtree on Discriminative Performance

The structure and size of the learned PSDD is largely determined by the vtree it is normalized for. Naturally, the vtree also has an important role in determining the quality (in terms of log-likelihood) of the probability distribution encoded by the learned PSDD [20]. In this section, we study the impact that the choice of vtree and learning strategy has on the trade-offs between model tractability, quality and discriminative performance.

[1] NB, TANB and LogReg are learned using Weka with default settings.
[2] In each fold, we hold 10% of the data for validation.
[3] Using the ACE tool Available at http://reasoning.cs.ucla.edu/ace/.

Table 2. Five cross fold accuracy and size in number of parameters

Dataset	D-LearnPSDD		LearnPSDD		NB		TANB		LogReg
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy
Australian	$\mathbf{86.2 \pm 3.6}$	367	84.9 ± 2.7	386	85.1 ± 3.1	161	85.8 ± 3.4	312	84.1 ± 3.4
Breast	97.1 ± 0.9	291	94.9 ± 0.5	491	$\mathbf{97.7 \pm 1.2}$	114	97.7 ± 1.2	219	96.5 ± 1.6
Chess	$\mathbf{97.3 \pm 1.4}$	2178	94.9 ± 1.6	2186	87.7 ± 1.4	158	91.7 ± 2.2	309	96.9 ± 0.7
Cleve	82.2 ± 2.5	292	81.9 ± 3.2	184	$\mathbf{84.9 \pm 3.3}$	102	79.9 ± 2.2	196	81.5 ± 2.9
Corral 6	$\mathbf{99.4 \pm 1.4}$	39	98.1 ± 2.8	58	89.4 ± 5.2	26	98.8 ± 1.7	45	86.3 ± 6.7
Credit	85.6 ± 3.1	693	86.1 ± 3.6	611	$\mathbf{86.8 \pm 4.4}$	170	86.1 ± 3.9	326	84.7 ± 4.9
Diabetes	$\mathbf{78.7 \pm 2.9}$	124	77.2 ± 3.3	144	77.4 ± 2.56	46	75.8 ± 3.5	86	78.4 ± 2.6
German	72.3 ± 3.2	1185	69.9 ± 2.3	645	73.5 ± 2.7	218	$\mathbf{74.5 \pm 1.9}$	429	74.4 ± 2.3
Glass	$\mathbf{79.1 \pm 1.9}$	214	72.4 ± 6.2	321	70.0 ± 4.9	203	69.5 ± 5.2	318	73.0 ± 5.7
Heart	$\mathbf{84.1 \pm 4.3}$	51	78.5 ± 5.3	75	84.0 ± 3.8	38	83.0 ± 5.1	70	84.0 ± 4.7
Iris	90.0 ± 0.1	76	94.0 ± 3.7	158	$\mathbf{94.7 \pm 1.8}$	75	94.7 ± 1.8	131	94.7 ± 2.9
Mofn	98.9 ± 0.9	260	97.1 ± 2.4	260	85.0 ± 5.7	42	92.8 ± 2.6	78	$\mathbf{100.0 \pm 0}$
Pima	$\mathbf{80.2 \pm 0.3}$	108	74.7 ± 3.2	110	77.6 ± 3.0	46	76.3 ± 2.9	86	77.7 ± 2.9
Vehicle	$\mathbf{95.0 \pm 1.7}$	1186	93.9 ± 1.69	1560	86.3 ± 2.00	228	93.0 ± 0.8	442	94.5 ± 2.4
Waveform	85.0 ± 1.0	3441	78.7 ± 5.6	2585	80.7 ± 1.9	657	83.1 ± 1.1	1296	$\mathbf{85.5 \pm 0.7}$

Figure 3a shows test-set log-likelihood and Fig. 3b classification accuracy as a function of model size (in number of parameters) for the "Chess" dataset. We display average log-likelihood and accuracy over logarithmically distributed ranges of model size. This figure contrasts the results of three learning approaches: D-LearnPSDD when the vtree learning stage optimizes mutual information (MI, shown in light blue); when it optimizes conditional mutual information (CMI, shown in dark blue); and the traditional LearnPSDD (in orange).

Figure 3a shows that likelihood improves at a faster rate during the first iterations of LearnPSDD, but eventually settles to the same values as D-LearnPSDD because both optimize for log-likelihood. However, the discriminative bias guarantees that classification accuracy on the initial model will be at least as high as that of a Naive Bayes classifier (see Fig. 3b). Moreover, this results in consistently superior accuracy (for the CMI case) compared to the purely generative LearnPSDD approach as shown also in Table 2. The dip in accuracy during the second and third intervals are a consequence of the generative learning, which optimizes for log-likelihood and can therefore initially yield feature-value correlations that decrease the model's performance as a classifier.

Finally, Fig. 3b demonstrates that optimizing the vtree for conditional mutual information results in an overall better performance vs. accuracy trade-off when compared to optimizing for mutual information. Such a conditional mutual information objective function is consistent with the conditional independence constraint we impose on the structure of the PSDD and allows the model to consider the special status of the class variable in the discriminative task.

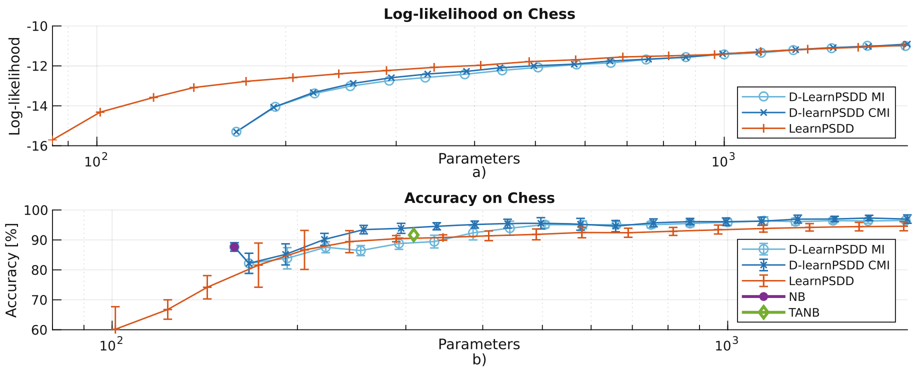


Fig. 3. Log-likelihood and accuracy vs. model size trade-off of the incremental PSDD learning approaches. MI and CMI denote mutual information and conditional mutual information vtree learning, respectively. (Color figure online)

4.4 Robustness to Missing Features

The generative models in this paper encode a joint probability distribution over all variables and therefore tend to be more robust against missing features than discriminative models, which only learn relations relevant to their discriminative task. In this experiment, we assessed this robustness aspect by simulating the random failure of 10% of the original feature set per benchmark and per fold in five-fold cross-validation. Figure 4 shows the average accuracy over 10 such feature failure trials in each of the 5 folds (flat markers) in relation to their full feature set accuracy reported in Table 2 (shaped markers). As expected, the performance of the discriminative classifier (LogReg) suffers the most during feature failure, while D-LearnPSDD and LearnPSDD are notably more robust than any other approach, with accuracy losses of no more than 8%. Note from the flat markers that the performance of D-LearnPSDD under feature failure is the best in all datasets but one.

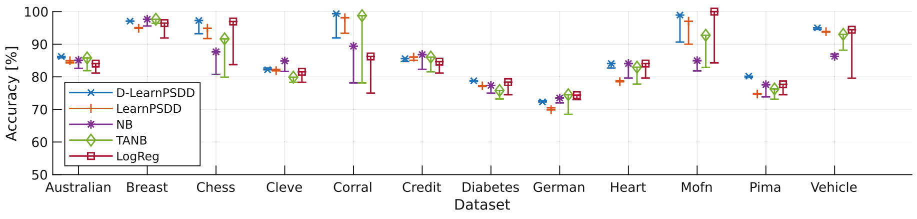


Fig. 4. Classification robustness per method.

5 Related Work

A number of works have dealt with the conflict between generative and discriminative model learning, some dating back decades [14]. There are multiple techniques that support learning of parameters [13,23] and structure [21,24] of probabilistic circuits. Typically, different approaches are followed to either learn generative or discriminative tasks, but some methods exploit discriminative models' properties to deal with missing variables [22]. Other works that also constraint the structure of PSDDs have been proposed before, such as Choi et al. [3]. However, they only do parameter learning, not structure learning: their approach to improve accuracy is to learn separate structured PSDDs for each distribution of features given the class and feed them to a NB classifier. In [5], Correira and de Campos propose a constrained SPN architecture that shows both computational efficiency and classification performance improvements. However, it focuses on decision robustness rather than robustness against missing values, essential to the application range discussed in this paper. There are also a number of methods that focus specifically on the interaction between discriminative and generative learning. In [15], Khosravi et al. provide a method to compute expected predictions of a discriminative model with respect to a probability distribution defined by an arbitrary generative model in a tractable manner. This combination allows to handle missing values using discriminative couterparts of generative classifiers [16]. More distant to this work is the line of hybrid discriminative and generative models [19], their focus is on semisupervised learning and deals with missing labels.

6 Conclusion

This paper introduces a PSDD learning technique that improves classification performance by introducing a discriminative bias. Meanwhile, robustness against missing data is kept by exploiting generative learning. The method capitalizes on PSDDs' domain knowledge encoding capabilities to enforce the conditional relation between the class and the features. We prove that this constraint is guaranteed to be enforced throughout the learning process and we show how not encoding such a relation might lead to poor classification performance. Evaluation on a suite of benchmarking datasets shows that the proposed technique outperforms purely generative PSDDs in terms of classification accuracy and the other baseline classifiers in terms of robustness.

Acknowledgements. This work was supported by the EU-ERC Project Re-SENSE grant ERC-2016-STG-71503; NSF grants IIS-1943641, IIS-1633857, CCF-1837129, DARPA XAI grant N66001-17-2-4032, gifts from Intel and Facebook Research, and the "Onderzoeksprogramma Artificiële Intelligentie Vlaanderen" programme from the Flemish Government.

References

1. Bekker, J., Davis, J., Choi, A., Darwiche, A., Van den Broeck, G.: Tractable learning for complex probability queries. In: Advances in Neural Information Processing Systems (2015)
2. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Proceedings of the International Conference on Uncertainty in Artificial Intelligence (1996)
3. Choi, A., Tavabi, N., Darwiche, A.: Structured features in naive bayes classification. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
4. Choi, Y., Vergari, A., Van den Broeck, G.: Lecture Notes: Probabilistic Circuits: Representation and Inference (2020). http://starai.cs.ucla.edu/papers/LecNoAAAI20.pdf
5. Correia, A.H.C., de Campos, C.P.: Towards scalable and robust sum-product networks. In: Ben Amor, N., Quost, B., Theobald, M. (eds.) SUM 2019. LNCS (LNAI), vol. 11940, pp. 409–422. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35514-2_31
6. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press, Cambridge (2009)
7. Darwiche, A.: SDD: a new canonical representation of propositional knowledge bases. In: International Joint Conference on Artificial Intelligence (2011)
8. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
9. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (1993)
10. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. J. Mach. Learn. **29**(2), 131–163 (1997)
11. Galindez, L., Badami, K., Vlasselaer, J., Meert, W., Verhelst, M.: Dynamic sensor-frontend tuning for resource efficient embedded classification. IEEE J. Emerg. Sel. Top. Circuits Syst. **8**(4), 858–872 (2018)
12. Galindez Olascoaga, L., Meert, W., Shah, N., Verhelst, M., Van den Broeck, G.: Towards hardware-aware tractable learning of probabilistic models. In: Advances in Neural Information Processing Systems, pp. 13726–13736 (2019)
13. Gens, R., Domingos, P.: Discriminative learning of sum-product networks. In: Advances in Neural Information Processing Systems (2012)
14. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in Neural Information Processing Systems (1999)
15. 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, pp. 11167–11178 (2019)
16. 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)
17. Kisa, D., den Broeck, G.V., Choi, A., Darwiche, A.: Probabilistic sentential decision diagrams. In: International Conference on the Principles of Knowledge Representation and Reasoning (2014)
18. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)

19. Lasserre, J.A., Bishop, C.M., Minka, T.P.: Principled hybrids of generative and discriminative models. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
20. Liang, Y., Bekker, J., Van den Broeck, G.: Learning the structure of probabilistic sentential decision diagrams. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI) (2017)
21. Liang, Y., Van den Broeck, G.: Learning logistic circuits. In: Proceedings of the Conference on Artificial Intelligence (AAAI) (2019)
22. Peharz, R., et al.: Random sum-product networks: a simple and effective approach to probabilistic deep learning. In: Conference on Uncertainty in Artificial Intelligence (UAI) (2019)
23. Poon, H., Domingos, P.: Sum-product networks: a new deep architecture. In: IEEE International Conference on Computer Vision Workshops (2011)
24. Rooshenas, A., Lowd, D.: Discriminative structure learning of arithmetic circuits. In: Artificial Intelligence and Statistics, pp. 1506–1514 (2016)