Discriminative Bias for Learning Probabilistic Sentential Decision Diagrams

Laura I. Galindez Olascoaga[1], Wannes Meert[2], Nimish Shah[1], Guy Van den Broeck[3], and Marian Verhelst[1]

[1] Electrical Engineering Department, KU Leuven
[2] Computer Science Department, KU Leuven
[3] Computer Science Department, University of California, Los Angeles
laura.galindez@esat.kuleuven.be

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 demonstrated to be a well suited approach to address the challenges inherent to embedded applications, such as the need to handle uncertainty and missing data [9]. 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, and can thus be deployed in application scenarios where strict resource budget constraints must be met [10]. However, the models' robustness against missing data, enabled by learning them generatively, is often at odds with their discriminative capabilities, relevant to many embedded application scenarios. In this work, we address such a conflict by proposing a discriminative-generative tractable probabilistic model learning strategy, aiming at improved discriminative capabilities, while maintaining robustness against missing features.

We focus in particular on the Probabilistic Sentential Decision Diagram (PSDD) [15], a state-of-the-art tractable representation that encodes a joint probability distribution over a set of random variables. Previous work [10] has shown how to learn PSDDs that are guaranteed to be hardware efficient while

remaining robust to missing data and noise. This approach relies largely on the LearnPSDD algorithm [18], a generative algorithm that incrementally learns the structure of a PSDD from data. Moreover, it is shown that such robustness can be exploited to reduce resource usage, or to trade off resource usage with accuracy by dropping sensors or using less bits. And while the achieved accuracy is competitive when compared to Bayesian Network classifiers, discriminatively learned models perform consistently better than purely generative models, also in other works [19], as 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 still keeping the desired robustness and tractability.

In this work, we propose a hybrid discriminative-generative PSDD learning strategy, D-LearnPSDD, that capitalizes on the model's ability to encode domain knowledge as a logic formula and thus provides the PSDD with information about the discriminative relationship between the class and the feature variables. 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. We use standard 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.

PSDD. Probabilistic Sentential Decision Diagrams (PSDDs) are circuit representations of joint probability distributions over binary random variables [15], and were introduced as probabilistic extensions to sentential decision diagrams (SDDs) [5], which represent Boolean functions as logical circuits. The inner nodes of a PSDD alternate between AND gates with two inputs and OR gates with

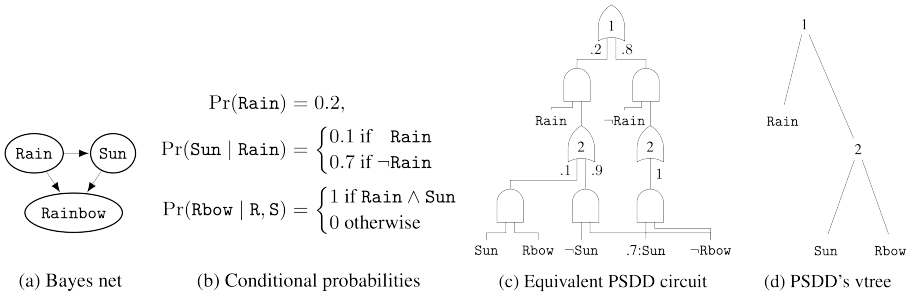

Fig. 1. A Bayesian network and its equivalent PSDD (taken from [18]).

arbitrary number of inputs; the root must be an OR node; and each leaf node encodes a distribution over a variable X (see Fig. 1.c). The combination of an OR gate with its AND gate inputs is referred to as a *decision* node, where the left input of the AND gate is referred to as *prime* (p), and the right referred to as *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 their input variables must be disjoint. This property is enforced by a variable tree, or *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. 1.d): 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*, meaning that only one of its inputs can be true.

Each node q represents a probability distribution, starting with the terminal nodes' univariate distributions. Each decision node q normalized for a vtree node with $\mathbf{X}$ and $\mathbf{Y}$ in its left and right subtrees respectively, represents a distribution over $\mathbf{XY}$ (see also Fig 1.a and 1.c):

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

This is possible because each decision node decomposes the distribution into independent distributions over $\mathbf{X}$ and $\mathbf{Y}$, guided by the vtree. Specifically, prime and sub variables are independent at PSDD node q given a prime *base* [15], written as $[q]$. This base is the support of the node's distribution, over which it defines a non-zero probability. The notation $[q]$ indicates that this is a logical sentence. The base of a node can be written as a logical sentence using the recursion $[q] = \bigvee_i [p_i] \wedge [s_i]$. Kisa et al. [15] show that prime and sub variables are independent in PSDD 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 [18] generatively learns a PSDD that maximizes the likelihood given a data set. LearnPSDD first uses the data to learn a *vtree*, a binary tree that minimizes the mutual information among all possible sets of variables. Next, given the vtree, it learns the PSDD structure by iteratively applying the operations Split and Clone on the nodes in the

PSDD [18]. 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 PSDD for classfication is that when the class variable is only weakly dependent on the features, the learner may choose to ignore that dependency making the model useless for classification.

3 A Discriminative Bias for PSDD learning

Generative learners such as LearnPSDD optimize the likelihood of the distribution given the training data rather than the conditional likelihood of the class variable C given a full set of feature variables $\mathbf{F}$. As a result, the zero-one loss or accuracy is often worse than simple models such as Naive Bayes (NB), and its close relative Tree Augmented Naive Bayes (TANB) [10]. Often NB and TANB perform surprisingly well on classification tasks although they encode a simple – or naive – structure that is learned generatively [8]. One of the main reasons why (TA)NB works well for classification even though it is generative is because it has a discriminative bias that directly encodes that all features are conditionally dependent on the class variable. In this work we use this insight to alter the LearnPSDD algorithm such that it learns a PSDD that has a discriminative bias, which improves classification performance while at the same time retaining the advantages from generative learning to be robust against missing data.

We introduce an extension to LearnPSDD, called D-LearnPSDD, which is 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 [18].

3.1 Discriminative bias

The classification task can be stated as a probabilistic query:

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

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, thus $C = 1$, 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 (following [18]). 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 [18]). First, the Clone operator requires a parent node, which is not available for the root node. Since the initial PSDD follows the template described above, whose only restriction is on the root node, there is no parent available to clone and the template thus remains intact. 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 template stays intact (see Figures 1(c) 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 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 $C=1$.

In the following examples, it is illustrated why both the specific vtree and template PSDD are required.

Example 1. This example shows a PSDD that can be generated by the generative LearnPSDD learner and that initializes the incremental learning procedure [18]. Fig. 2 shows a PSDD that encodes a fully factorized probability distribution normalized for the vtree in (a). Note that the vtree connects the class variable C to some of the feature variables, but doesn't for all (i.e. F_1). The learning algorithm might thus never learn conditional relations between certain features and the class.

Example 2. This example show a vtree and PSDD that is generated by D-LearnPSDD. Fig. 2.(e) shows the PSDD that explicitly conditions the feature set $\mathbf{F}$ on the class variable by following the template from Theorem 2. This PSDD is normalized for the vtree in (c), which has been forced to have the class variable C as the prime of the root node.

Example 3. Fig. 2.(d) shows why only setting the vtree in (c) 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 learner is not guaranteed to learn the relation between the class variable and the features. In this worst case scenario, the learned model could have an even worse performance than the case at 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 completely independent from all feature variables. While this might still result in a high likelihood, the classification accuracy will be low.

D-LearnPSDD as introduced for a binary classification task above can be generalized to an n-valued class variables as follows: (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 has thus nodes 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 here and start with a PSDD that encodes a distribution where all feature variables are independent given the class variable (see Figure 2.e). Next, the LearnPSDD algorithm will incrementally learn the relations between the feature variables by applying the Split and Clone operations following the approach in [18].

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 [18] with: $CMI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{x}} \sum_{\mathbf{y}} \sum_{\mathbf{z}} \Pr(\mathbf{xy}) \log \frac{\Pr(\mathbf{z}) \Pr(\mathbf{xyz})}{\Pr(\mathbf{xz}) \Pr(\mathbf{yz})}$.

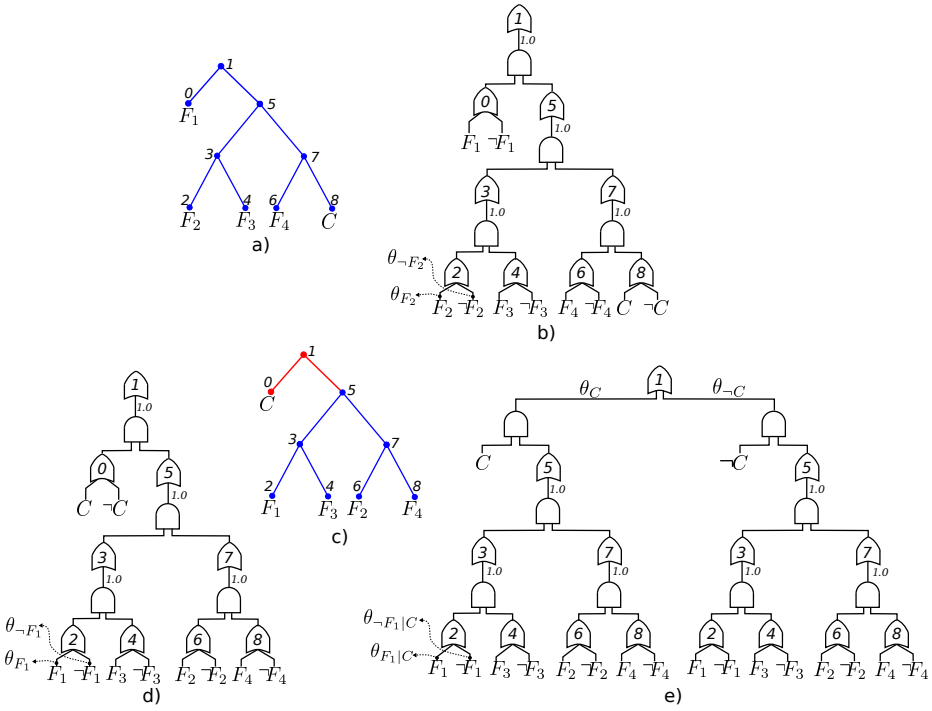

Fig. 2. Examples of vtrees and initial PSDDs.

4 Experiments

We evaluate the performance of D-LEARNPSDD and compare it with that of LEARNPSDD, two generative Bayesian classifiers (NB and TANB) and a discriminative classifier (logistic regression). In particular, we discuss the following research queries: (1) Section 4.2 examines whether the introduced discriminative bias improves classification performance on PSDDs. (2) Section 4.3 analyzes the

impact of the vtree and the imposed structural constraints on model tractability and performance. (3) Finally, Section 4.4 compares the robustness to missing values for all classification approaches.

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 [6], with exception of "Mofn" and "Corral" [16]. As preprocessing steps, we applied the discretization method described in [7], 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 |N| per dataset.

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

4.2 Evaluation of DG-LearnPSDD

Table 2 compares D-LearnPSDD, LearnPSDD, Naive Bayes (NB), Tree Augmented Naive Bayes (TANB) and logistic regression (LogReg)[4] in terms of accuracy via five fold cross validation [5]. For LearnPSDD, we incrementally learned a model on each fold until convergence on validation-data log-likelihood, following the methodology in [18]. 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 [6], a more general form of PSDDs [4], 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] NB, TANB and LogReg are learned using Weka with default settings.
[5] In each fold, we hold 10% of the data for validation.
[6] 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	$\mathbf{85.5 \pm 0.7}$

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 [18]. 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.

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

Figure 3(a) 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 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 Figure 3(b)). 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 likelihood and can therefore initially yield feature-value correlations that decrease the model's performance as a classifier.

Finally, Figure 3(b) demonstrates that learning a vtree that optimizes conditional mutual information results in an overall better performance versus 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.

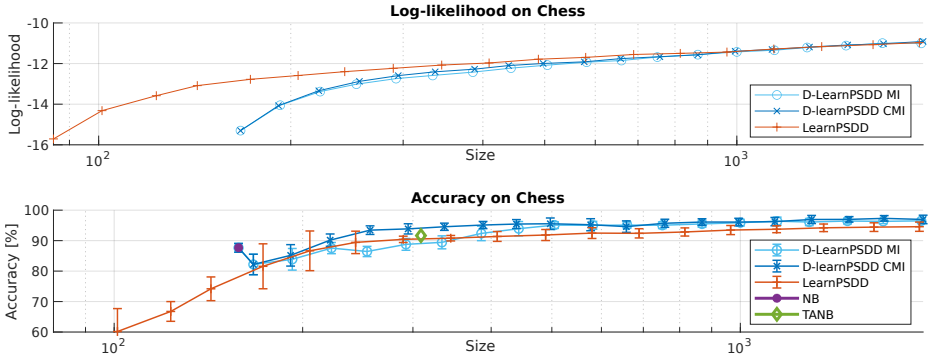

Fig. 3. Log-likelihood and accuracy vs. model size trade-off of the incremental PSDD learning approaches. MI and CMI denote mutual information and conditional mutual information vtree learning, respectively.

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.

5 Related work

There are a number of works that deal with the conflict between generative and discriminative model learning, some dating back decades [12]. Models that are based on the same general representation, Arithmetic Circuits and Sum-Product Networks, have been developed to support discriminative and generative learning of parameters [21, 11] and structure [22, 19]. They share the advantage of being tractable but expressive enough to achieve state-of-the-art performance.

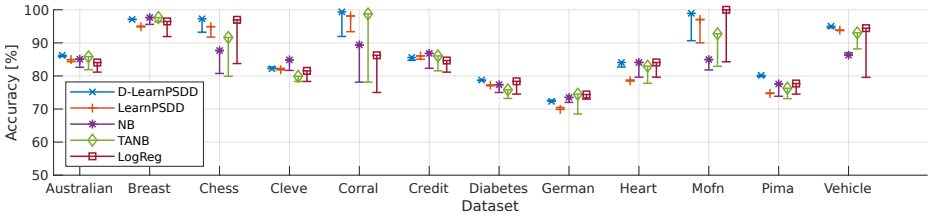

Fig. 4. Classification robustness per method.

Typically, different approaches are followed to learn models catered to either generative or discriminative tasks, but some methods exploit discriminative models' properties to deal with missing variables [20]. Other works that also constraint the structure of PSDDs have been proposed before, such as Choi et al. [3]. However, they learn separate structured PSDDs for each distribution of features given the class to improve accuracy and they only do parameter learning, not structure learning. There are also a number of methods that focus specifically on the interaction between discriminative and generative learning. Khosravi et al. [13] provides 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 [14]. More distant to this work is the line of hybrid discriminative and generative models [17], 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 while at the same time it ensures robustness against missing data 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.

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. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
5. Darwiche, A.: Sdd: A new canonical representation of propositional knowledge bases. In: International Joint Conference on Artificial Intelligence (2011)
6. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
7. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (1993)
8. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Journal of Machine Learning **29**(2), 131–163 (1997)
9. Galindez, L., Badami, K., Vlasselaer, J., Meert, W., Verhelst, M.: Dynamic sensor-frontend tuning for resource efficient embedded classification. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **8**(4), 858–872 (2018)
10. Galindez Olascoaga, L., Meert, W., Shah, N., Verhelst, M., Van den Broeck, G.: Towards hardware-aware tractable learning of probabilistic models. In: Proceedings of NeurIPS (2019)
11. Gens, R., Domingos, P.: Discriminative learning of sum-product networks. In: Advances in Neural Information Processing Systems (2012)
12. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in neural information processing systems (1999)
13. 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)
14. Khosravi, P., Liang, Y., Choi, Y., Van den Broeck, G.: What to expect of classifiers? reasoning about logistic regression with missing features. In: In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), (2019)
15. 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)
16. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial intelligence **97**(1-2), 273–324 (1997)
17. 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)
18. 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)
19. Liang, Y., Van den Broeck, G.: Learning logistic circuits. Proceedings of the Conference on Artificial Intelligence (AAAI) (2019)
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: Conference on Uncertainty in Artificial Intelligence (UAI) (2019)
21. Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. In: IEEE International Conference on Computer Vision Workshops (2011)
22. Rooshenas, A., Lowd, D.: Discriminative structure learning of arithmetic circuits. In: Artificial Intelligence and Statistics. pp. 1506–1514 (2016)