Discriminative bias for classification with robust probabilistic models

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

Abstract

Probabilistic Sentential Decision Diagrams (PSDD) are a type of probabilistic circuit that can be learned incrementally from data by iteratively optimizing the log-likelihood of the induced distribution. As generative models, PSDDs remain robust against missing features but are often outperformed by discriminatively trained models in classification tasks. We consider the recently published D-LEARNPSDD learner, which explicitly encodes the discriminative relation between class and feature variables to improve classification performance, while still reaping the robustness benefits of generative learning. In this work we further generalize the original contribution's theorems to consider multi-value classification scenarios, and we discuss the implementation techniques suitable for those scenarios.

1. Introduction

Current efforts in the field of Tractable Probabilistic Modeling have been able to successfully balance 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 (Choi et al., 2020) that make them amenable to application scenarios where strict resource budget constraints must be met (Galindez Olascoaga et al., 2019). But their robustness against missing data—from learning them generatively—is often at odds with their discriminative capabilities. This conflict is the focus of this work.

We look in particular at the PSDD (Kisa et al., 2014), a state-of-the-art tractable representation that encodes a joint probability distribution over a set of random variables. Previous work (Galindez Olascoaga et al., 2019) has shown how to learn resource-efficient PSDDs that remain robust to missing features and noise. While the achieved accuracy is competitive when compared to Bayesian network classifiers, discriminatively learned models perform consistently better than purely generative models (Liang & Van den Broeck, 2019) 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.

This work discusses the recently introduced D-LEARNPSDD (Galindez Olascoaga et al., 2020), a hybrid discriminative-generative PSDD learning strategy, that enforces the discriminative relationship between class and feature variables by means of a logic formula. Our approach shows to consistently outperform the purely generative case and is competitive compared to other classifiers, while remaining robust to missing features. We generalize this work to show how any multi-value classification scenario can benefit from the D-LEARNPSDD when initialized to the pertinent logical formulas and structural conditions.

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 $\mathbf{x}$.

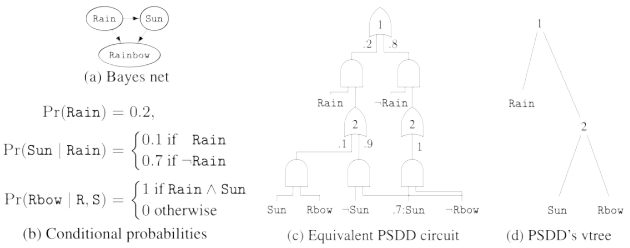

Figure 1. A Bayesian network and its equivalent PSDD (taken from (Liang et al., 2017)).

PSDD. Probabilistic Sentential Decision Diagrams (PSDDs) are circuit representations of joint probability distribu-

[1]KU Leuven, Belgium [2]University of California, Los Angeles. Correspondence to: Laura Galindez <laura.galindez@esat.kuleuven.be>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

tions over binary random variables (Kisa et al., 2014). They were introduced as probabilistic extensions to Sentential Decision Diagrams (SDDs) (Darwiche, 2011), 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. 1.c). 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 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 (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 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. 1.a and 1.c):

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

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* $[p]$ (Kisa et al., 2014):

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

The base of node q 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 $[q] = \bigvee_i [p_i] \wedge [s_i]$.

LearnPSDD. The LEARNPSDD algorithm (Liang et al., 2017) 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 (Liang et al., 2017). These operations keep the PSDD syntactically sound while improving likelihood of the distribution represented by the PSDD.

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 C given a full set of features $\mathbf{F}$. As a result, their accuracy is comparable or worse than that of simple models such as naive Bayes (NB), and Tree Augmented naive Bayes (TAN), which perform surprisingly well despite their naive structure (Friedman et al., 1997). One of the main reasons for their performance, despite being generative, is that (TA)NB models structurally enforce a discriminative bias that directly encodes the conditional dependence of all the features on the class variable.

This Section introduces the D-LEARNPSDD (Galindez Olascoaga et al., 2020), 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 state-of-the-art learning strategies (Liang et al., 2017).

3.1. D-LearnPSDD

The classification task can be stated as the following conditional probability query: $\Pr(C|\mathbf{F}) \sim \Pr(\mathbf{F}|C) \cdot \Pr(C)$. Our goal is to learn PSDDs whose joint probability distributions include the expression $\Pr(\mathbf{F}|C)$, like Bayesian network classifiers do (Friedman et al., 1997).

The first step in our strategy is to set the root-node's prime and sub variables to be C and $\mathbf{F}$, respectively. The root node can thus represent the following joint distribution (following the notation of Equations 1 and 2):

$$\Pr_{root}(C\mathbf{F}) = \sum_{i=0}^{n} \theta_i \Pr_{p_i}(C) \Pr_{s_i}(\mathbf{F}), \tag{3}$$

where n is the cardinality of the class variable. We need to ensure that the term $\Pr_{s_i}(\mathbf{F})$ represents the conditional relation of interest between $\mathbf{F}$ and C. To achieve this we require the following proposition:

Proposition 1. *Given (i) a vtree with 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 $[root] = \bigvee_{i:0...n}([p_i] \wedge [s_i])$; applying the Split and Clone operators will never change the root decision decomposition $[root] = \bigvee_{i:0...n}([p_i] \wedge [s_i])$.*

The proof of this proposition can be found in Appendix A. Note that for an n-value classification scenario, C is represented by multiple propositional variables $c_0, c_1, \ldots, c_n$ that correspond to the set $C = 0, C = 1, \ldots, C = n$, of which exactly one will be true at all times. The root-node's

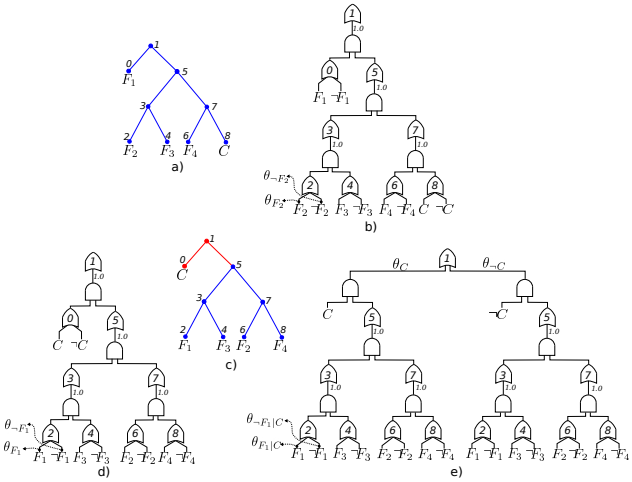

Figure 2. Examples of vtrees and initial PSDDs.

sub for the vtree in Proposition 1 is still the set of variables $\mathbf{F}$, but the prime is defined over the available propositional variables associated to the class (see Appendix C for an example). The base of the root-node is thus defined as:

$$[root] = \bigvee_{i=0\ldots n} ([c_i \bigwedge_{j:0\ldots n \wedge i \neq j} \neg c_j] \wedge [s_i]), \qquad (4)$$

which enforces the mutual exclusivity of the values of C. In the case of a binary classification scenario this can be simplified to $[root] = ([\neg c] \wedge [s_0]) \vee ([c] \wedge [s_1])$.

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

Proposition 2. *A PSDD of the form* $\bigvee_{i=0\ldots n}([c_i \bigwedge_{j:0\ldots n \wedge i \neq j} \neg c_j] \wedge [s_i])$ *with* s_i *any formula with propositional feature variables* $f_0, \ldots, f_n$, *directly expresses the distribution* $\Pr(\mathbf{F}|C)$.

Appendix B includes the proof. In addition to defining a suitable vtree, we must define an initial PSDD that is consistent with the logic formulas above, as illustrated by the following examples. These examples consider binary classification, but the concepts extend to the n-class scenario.

Example 1. Fig. 2.b shows a PSDD that encodes a fully factorized probability distribution normalized for the vtree in 2.a. Note that the vtree does not connect the class variable C to all feature variables (e.g. F_1) so D-LEARNPSDD is not guaranteed to find conditional relations between certain features and the class when initialized with this PSDD.

Example 2. Fig. 2.e shows a PSDD that explicitly conditions the feature variables on the class variables by normalizing for the vtree in 2.c and by following the logical formula from Proposition 2.

Example 3. When initializing on a PSDD that encodes a fully factorized formula (Fig. 2.d), the base of the root node is $[root] = [c \vee \neg c] \wedge [s_0]$. Eq. 1 evaluates to:

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

Thus, in this worst case scenario, the PSDD encodes a distribution that assumes the class to be independent from all feature variables, which may still lead to high log-likelihood but will have low classification accuracy.

3.2. Generative bias and vtree learning

Learning the distribution over the feature variables is a generative learning process and we achieve this by applying the Split and Clone operators in the same way as the original LEARNPSDD algorithm. To define the nodes corresponding to s_i with distributions $\Pr(\mathbf{F}|C=i)$ 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. 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 (Liang et al., 2017).

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 (Liang et al., 2017) with: $CMI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{x}} \sum_{\mathbf{y}} \sum_{\mathbf{z}} \Pr(\mathbf{xy}) \log \frac{\Pr(\mathbf{z}) \Pr(\mathbf{xyz})}{\Pr(\mathbf{xz}) \Pr(\mathbf{yz})}$.

4. Experiments

We compare the performance of D-LEARNPSDD, LEARNPSDD, two generative Bayesian classifiers (NB and TANB) and a discriminative classifier (logistic regression): Section 4.1 examines whether the introduced discriminative bias improves classification performance on PSDDs. Section 4.2 compares the robustness to missing values for all classification approaches.

We ran our experiments on the suite of standard benchmarks (Dua & Graff, 2017; Kohavi & John, 1997) listed in Table 1, which also summarizes the number of features $|\mathbf{F}|$, the number of classes $|C|$ and the available number of training samples $|\mathrm{N}|$. As pre-processing steps, we applied the discretization method described in (Fayyad & Irani, 1993), and we binarized all variables using a one-hot encoding.

Table 1. Five cross fold accuracy and size in number of parameters

Dataset	$\lvert\mathbf{F}\rvert$,$\lvert C\rvert$,$\lvert N\rvert$	D-LearnPSDD		LearnPSDD		NB		TANB		LogReg
		Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy
Australian	40, 2, 690	$\mathbf{86.2 \pm 3.6}$	367	84.9 ± 2.7	386	85.1 ± 3.1	161	85.8 ± 3.4	312	84.1 ± 3.4
Breast	28, 2, 683	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	39, 2, 3196	$\mathbf{97.3 \pm 1.4}$	2178	94.9 ± 1.6	2186	87.7 ± 1.4	158	91.7 ± 2.2	309	96.9 ± 0.7
Cleve	25, 2, 303	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, 2, 160	$\mathbf{99.4 \pm 1.4}$	39	98.1 ± 2.8	58	89.4 ± 5.2	26	98.8 ± 1.7	45	86.3 ± 6.7
Credit	42, 2, 653	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	11, 2, 768	$\mathbf{78.7 \pm 2.9}$	124	77.2 ± 3.3	144	77.4 ± 2.56	46	75.8 ± 3.5	86	78.4 ± 2.6
German	54, 2, 1000	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	17, 6, 214	$\mathbf{79.1 \pm 1.9}$	214	72.4 ± 6.2	321	70.0 ± 4.9	203	69.5 ± 5.2	318	73.0 ± 5.7
Heart	9, 2, 270	$\mathbf{84.1 \pm 4.3}$	51	78.5 ± 5.3	75	84.0 ± 3.8	38	83.0 ± 5.1	70	84.0 ± 4.7
Iris	12, 3, 150	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	10, 2, 1324	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	11, 2, 768	$\mathbf{80.2 \pm 0.3}$	108	74.7 ± 3.2	110	77.6 ± 3.0	46	76.3 ± 2.9	86	77.7 ± 2.9
Vehicle	57, 2, 846	$\mathbf{95.0 \pm 1.7}$	1186	93.9 ± 1.69	1560	86.3 ± 2.00	228	93.0 ± 0.8	442	94.5 ± 2.4
Waveform	109, 3, 5000	85.0 ± 1.0	3441	78.7 ± 5.6	2585	80.7 ± 1.9	657	83.1 ± 1.1	1296	$\mathbf{85.5 \pm 0.7}$

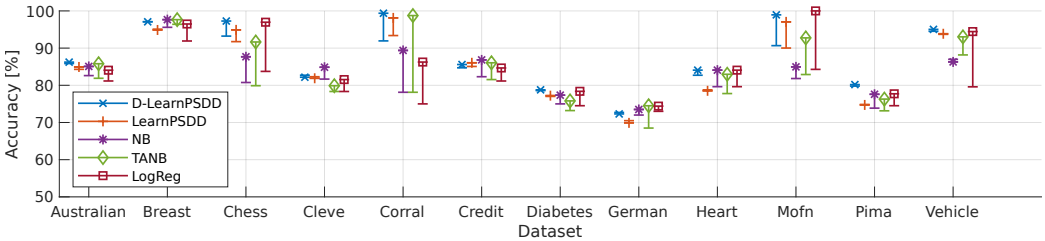

Figure 3. Robustness to missing features per method.

4.1. Evaluation of D-LEARNPSDD

Table 1 compares the classifiers in terms of accuracy via five fold cross validation and size in number of parameters. For LEARNPSDD, we incrementally learned a model on each fold until convergence on validation-data log-likelihood, following the methodology in (Liang et al., 2017). 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 ACs , a more general form of PSDDs (Darwiche, 2009). Finally, we compare all probabilistic classifier with a discriminative one, a multinomial logistic regression model.

Table 1 shows that the proposed D-LEARNPSDD outperforms 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 network classifiers in most benchmarks, as the learned PSDDs allow to encode complex relationships among sets of variables or local dependencies such as context specific independence (Boutilier et al., 1996), while remaining tractable.

4.2. 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. Fig. 3 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 1 (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. Conclusion

This paper discusses D-LEARNPSDD a hybrid discriminative-generative learning technique that improves classification performance by enforcing the conditional relation between the class and the features with logical constraints. It is proven that any multi-value classification scenario can benefit from the learner by constraining on the pertinent logical formulas and initialization settings. Meanwhile, robustness against missing features is kept by exploiting generative learning. Evaluation on a suite of benchmarks shows that the proposed technique outperforms purely generative PSDDs in terms of classification accuracy and the other baseline classifiers in terms of robustness.

References

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in bayesian networks. In *In Proceedings of the international conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

Choi, Y., Vergari, A., and Van den Broeck, G. Lecture notes: Probabilistic circuits: Representation and inference. 2020.

Darwiche, A. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

Darwiche, A. Sdd: A new canonical representation of propositional knowledge bases. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Fayyad, U. and Irani, K. Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Friedman, N., Geiger, D., and Goldszmidt, M. Bayesian network classifiers. *Journal of Machine Learning*, 29(2): 131–163, 1997.

Galindez Olascoaga, L. I., Meert, W., Shah, N., Verhelst, M., and Van den Broeck, G. Towards hardware-aware tractable learning of probabilistic models. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, December 2019.

Galindez Olascoaga, L. I., Meert, W., Shah, N., Van den Broeck, G., and Verhelst, M. Discriminative bias for learning probabilistic sentential decision diagrams. In *Proceedings of the Symposium on Intelligent Data Analysis (IDA)*, April 2020.

Kisa, D., Van den Broeck, G., Choi, A., and Darwiche, A. Probabilistic sentential decision diagrams. In *International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

Kohavi, R. and John, G. H. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

Liang, Y. and Van den Broeck, G. Learning logistic circuits. *In Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2019.

Liang, Y., Bekker, J., and Van den Broeck, G. Learning the structure of probabilistic sentential decision diagrams. In *In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

A. Proof of Proposition 1

Proof. The D-LEARNPSDD algorithm iteratively applies two operations: Split and Clone (following the algorithm in (Liang et al., 2017)). First, the Clone operation requires a parent node, which is not available for the root node. The nodes corresponding to the prime of the root node $c_0, \ldots, c_n$ can not be cloned either because the structure of the initial PSDD would not support the necessary copy and rerouting actions. Second, the Split operator splits AND nodes into multiple elements by constraining the prime with a partial assignment to the prime variable. These partial assignments are mutually exclusive and exhaustive to keep the decision node deterministic. In our case the logical formula at the root of the PSDD already encodes mutual exclusivity among the possible assignments of the Class (prime) variable. The root decision node is therefore already split into all the possible worlds available and the Split operator would be inconsequential. For these reasons, the root's base remains intact. □

B. Proof of Proposition 2

Proof. Applying this to Equation 1 results in:

$$\Pr_{root}(C\mathbf{F}) = \sum_i \theta_i \Pr_{c_i}(C)\Pr_{s_i}(\mathbf{F})$$
$$= \sum_i \theta_i \Pr_{c_i}(C|[c_i \bigwedge_{j:0\ldots n \wedge j \neq i} \neg c_j]) \cdot \Pr_{s_i}(\mathbf{F}|[c_i \bigwedge_{j:0\ldots n \wedge j \neq i} \neg c_j])$$
$$= \sum_i \theta_i \Pr_{c_i}(C = i) \cdot \Pr_{s_i}(\mathbf{F}|C = i)$$

The learned PSDD thus contains n nodes s_i with distribution $\Pr_{s_i}$ that directly represents $\Pr(\mathbf{F}|C = i)$. The PSDD thus encodes $\Pr(\mathbf{F}|C)$ directly because there are n possible value assignments of C. □

C. Vtree for the $n-$class scenario

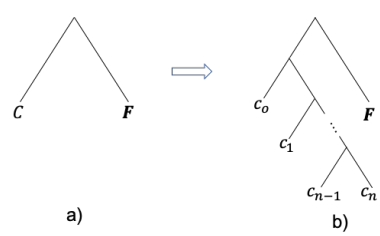

Figure 4. Vtree for an $n-$value classification problem. a) Variable C. b) Propositional variables for each value of C.