IL-Strudel : Independence-Based Learning of Structured-Decomposable Probabilistic Circuit Ensembles

Abstract

Probabilistic Sentential Decision Diagrams (PSDD) are a class of highly tractable structured-decomposable probabilistic circuits. They provide for closed-form parameter learning and capture structure from data in the form of context-specific independences. When learning mixture models of such circuits, the soft EM algorithm is used for parameter learning which is highly sensitive to initialization. These independences properties can be a valuable source of prior information for component initialization. We hypothesize that if each component structure can capture independences on disjoint sub-supports of the data, the overall ensemble can get a boost in performance. Through this paper, we first develop a framework for connecting these independences to likelihood-based structure evaluation. Using this framework, we propose a novel algorithm to learn a stronger mixture model by providing an initialization strategy to enforce context-specific-independences at the component root levels. Our experimental results validate our approach as it beats previous approaches on 15 out of 20 datasets.

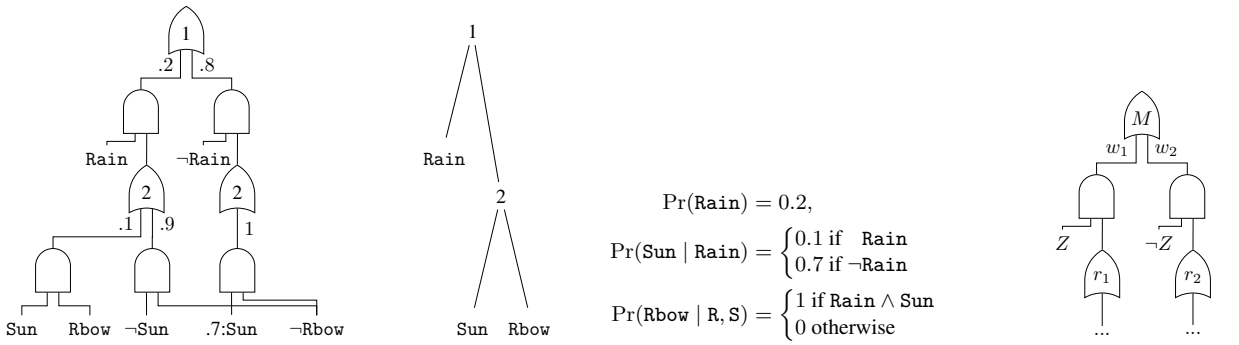
1 INTRODUCTION

There has been an emerging interest in the area of tractable probabilistic models. These provide for exact inference of various probabilistic queries in tractable time, which is of utmost importance in real-world domains such as healthcare. A variety of tractable probabilistic models (TPMs) have been proposed, each making different assumptions about the data distribution and supporting a different set of tractable queires. Recently, various forms of probabilistic circuits (PCs) have been chosen as the target representations for tractable learners [Poon and Domingos, 2011, Kisa et al., 2014, Rahman et al., 2014]. In this paper, we focus on a particular class of structured decomposable PCs called Probabilistic Sentential Decision Diagrams (PSDDs), as they support the most number of tractable queries [Shen et al., 2016, Bekker et al., 2015, Khosravi et al., Choi and Van den Broeck, 2018].

The structure of PSDDs can be learnt from data. LearnPSDD and Strudel [Liang et al., Dang et al., 2020] are two approaches for this. Though both algorithms can be extended to learn an ensemble model, they effectively focus on how to learn single circuit structures. On the other hand, ensemble models have demonstrated effectiveness in reducing the overall model bias compared to a single learner [Meila and Jordan, 2000, Schapire, 2003]. This effectiveness has also been shown in PC representations [Rahman and Gogate, 2016, Liang et al., Dang et al., 2020]. Parameter learning in mixture models is generally done using the Soft EM algorithm. Soft EM distributes the dataset examples across different components by assigning a weight to each component-example pair. Each component in a way becomes an expert on the examples it has a higher weight on, thus capturing a particular sub-support of the data distribution. However, Soft EM is highly sensitive to the initialization of its components. A bad initialization would lead to convergence to a poor local optimum.

PSDDs induce various context-specific-independences (CSIs), which are an important inductive bias imposed by their structure. When learning a single model, both LearnPSDD and Strudel start with an initial trivial structure and expand it based on local heuristics, which iteratively brings about a change in what context-specific-independences are captured by a circuit. Yet, neither explicitly utilise any similar inductive bias properties in the construction of the mixture model. We hypothesize that by initializing mixture components that are individually expanded over disjoint sub-supports of the data, one can learn a stronger mixture model. To confirm our hypothesis, we first provide a framework to evaluate a structure solely in terms of the CSIs embedded in it. Inspired by it, we propose

Submitted to the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021). **To be used for reviewing only**.



(a) PSDD (b) Vtree (c) Equivalent distribution (d) 2-PSDD Ensemble

Figure 1: Example PSDD and vtree for a set of conditional probabilities.

a novel algorithm to learn a strong mixture model, where each component tries to capture a diverse set of independences coherent with the data.

2 BACKGROUND AND NOTATION

We use upper-case letters to denote Boolean random variables. A lowercase complete instantiation **x** of variables **X** that satisfies sentence α is denoted $\mathbf{x} \models \alpha$.

Syntax and Semantics A probabilistic sentential decision diagram (PSDD) (Kisa et al. [2014]) is a circuit representation of a joint probability distribution over Boolean random variables. In a nutshell, a PSDD is a parameterized directed acyclic graph as shown in Figure 1a. Each inner node alternates between logical AND and OR nodes. A combination of an OR gate with its AND inputs is called a decision node. Every AND gate has two inputs. The left input is called as the prime p and right the sub s for that AND gate.

Each PSDD node encodes a probability distribution over the random variables it is defined over. These distributions can be obtained recursively. An AND node represents a distribution that can be factorized as a product of two distributions over disjoint sets of random variables. An OR node represents a weighted sum of distributions of its children. This recursion terminates at the leaf nodes, which represent univariate distributions over their literals. The PSDD in Figure 1a encodes equivalent distribution as Figure 1c.

As mentioned earlier, the inputs to an AND node must be defined over disjoint sets of random variables. We call this property *decomposability*. It is uniformly enforced throughout a PSDD by a *variable tree (vtree)*, a complete binary tree where the leaves represent boolean random variables. The internal nodes in a $vtree$ paritition variables into those appearing in their left and right children nodes. Each PSDD node must correspond to one $vtree$ node, and we say that the PSDD node is $normalized$ for the given $vtree$ node. Figure 1a assigns a label to each decision node based on the $vtree$ node it is normalized for in Firgure 1b. Every decision node n that is normalized for a vtree node v has its primes and subs ranging over the variables in the left and right subtrees of v respectively. The disjoint sets of random variables in the inputs to each AND node, thus conform to the structure imposed by the $vtree$, with the primes and subs defined over the variables under the vtree nodes they are normalized for. This uniform property across the whole circuit is known as *structured decomposability*.

Each PSDD node's induced distribution has an intricate support where it defines a non-zero probability. This support is denoted as the base $[n]$ of n. In order to reach a node n through a path, the bases of all encountered AND gates along the path must be satisfied. The conjunction of all such bases forms a sub-context of n. Disjunction of all the sub-contexts of n forms the context γ_n of n.[1]

Independences PSDDs encode independences that are context-specific. Context-specific-independences (Boutilier et al. [1996]) denote independences between variables subject to conditioning on a propositional sentence. A PSDD circuit captures structure in the data through the CSIs it encodes intrinsically. There are multiple types of independences engraved in a PSDD structure. We refer to Kisa et al. [2014] for a detailed analysis and highlight one particular type that we will be using for the rest of the work.

Proposition 1. (Prime-Sub Independence) *For a PSDD rooted at r, for every decision node n with prime and sub variables **X** and **Y** and their every element (p, s)[2]:*

$$\begin{aligned} &Pr_r(\mathbf{xy}|[p] \wedge [s] \wedge \gamma_n) \\ &= Pr_n(\mathbf{xy}|[p] \wedge [s]) = Pr_n(\mathbf{x}|[p] \wedge [s]) \cdot Pr_n(\mathbf{y}|[p] \wedge [s]) \\ &= Pr_r(\mathbf{x}|[p] \wedge [s] \wedge \gamma_n) \cdot Pr_r(\mathbf{y}|[p] \wedge [s] \wedge \gamma_n) \end{aligned}$$

[1] Original PSDD semantics define contexts with respect to the bases of primes, instead of the bases of the whole AND gates.

[2] Original PSDD semantics do not include the subs in the above independence. We use modified semantics in the context of Probabilistic Circuits. More details can be found in the appendix.

In other words, conditioned on the context of an AND child, the prime and sub variables of a decision node are independent. An example of context-specific prime-sub independence can be obtained with reference to Figure 1a. If we focus on the right most decision node normalized for $vtree$ node 2, it induces that Sun is independent from Rbow given $\neg$Rain $\wedge$ $\neg$Rbow. This is a straightforward example for the sake of illustration, as Rbow never occurs when $\neg$Rain.

Ensemble of PSDDs An ensemble of PSDDs M is a set of tuples $(r_i, w_i), i = 1...N$ where each component PSDD r_i has a component weight w_i, such that $\Sigma_{i=1}^{N} w_i = 1$. The ensemble encodes a distribution represented as :

$$Pr_M(\mathbf{x}) = \Sigma_{i=1}^{N} w_i Pr_{r_i}(\mathbf{x})$$

M can be interpreted as a single PSDD having a latent variable Z at the top. An example for a case of a 2-PSDD ensemble can be seen in Figure 1d. Due to the latent variables being un-observable, parameter learning in M is generally done using the soft EM algorithm. For fixed initial component structures, the EM algorithm iteratively updates the parameters and is guaranteed to reach locally optimal performance for log-likelihood.

3 ENSEMBLE LEARNING

A common practice in constructing ensembles is to assign different components to do well on different sub-supports of the data distribution. Intuitively, since soft EM tends to redistribute the data points among the different components, in the form of weights assigned to them. A good initial assignment can help the redistribution, leading to a better overall ensemble. In the context of PSDD mixture models, this raises the question: how to initialize these data points assignment such that we are more confident that each component will do well on a particular sub-support of the data distribution (under a reasonable training time)?

Not talking about generalization for now, an optimal PSDD mimics the training data distribution perfectly. This only happens when the structure-induced independences conform with the training data. This is formalized as the following.

Theorem 1. *A PSDD achieves the optimal (i.e., highest possible) training log-likelihood iff every induced prime-sub independence (defined in Proposition 1) holds empirically with respect to the training data.*[3]

The above theorem provides a framework for assessing the quality of a structure by only examining at its induced independences. This leads us to think whether we can partition the training data into disjoint sets such that each component's induced independences are more likely to hold with respect to the subsets of data points. Following on this though, we propose a novel algorithm to better initialize PSDD ensembles.

Initialization Strategy We hypothesize that if the initial components of an EM based mixture model are obtained by training them towards optimality on disjoint sub-supports of the data distribution, then the ensemble can get an overall boost in performance. We reason that if each component maximizes the log-likelihood on a disjoint sub-support, as a consequence of Theorem 1, it will start to pick independences specific to the domain it is trained on. Since all the sub-supports are disjoint, this can provide for a strong initialization strategy owing to the diversity of independences that can be captured.

Thus the first step for initialization of component structure boils down to finding disjoint sub-supports in the data distribution. A trivial appraoch to this will random chose between different partitions of the dataset examples. We provide for a more informative approach backed by theorem 1 to get disjoint sub-supports that aid component structures satisfy approximate root level prime sub independence.

We assume a given $vtree$ learned from data is shared across all the components. We then seek to find examples in the dataset such that on the dataset formed from these examples, the variables in the two sub-trees of the $vtree$ root are independent. This can be realised approximately by checking if the pairwise-mutual-information between the two variable sets is less than a threshold. Given an single example, the pairwise-mutual-information will always be zero between any two sets of variables. Thus we rather look to maximise the size of the set of examples such that the independence holds at the $vtree$ root level. This can be cast as a discrete optimization problem. We use a genetic algorithm with an appropriate fitness function to obtain such an example set. By repeating this process, we get mutually disjoint sets of examples, which are the disjoint domains for learning our component structures.

Any PSDD structure that is expanded over such a set of examples, has the prime sub independence holding at the root level, by the nature of the way these examples were obtained. This provides a great starting point than a randomly initialized set of examples to expand a structure, by explicitly making it satisfy prime sub independence at the root level. We expand a structure over each of the disjoint example sets to obtain the component structures for the mixture model.

4 EXPERIMENTS

We try to answer the following question through our experiments : Does using structures trained on disjoint domains of the dataset, such that the prime sub independence holds at the root level, really boost the overall performance of a mixture model?

[3] We defer its proof till the appendix.

Datasets	IL-Strudel	Strudel-EM	EM-LearnPSDD
NLTCS	**-6.03**	-6.07	-6.03
MSNBC	**-6.04**	-6.04	-6.04
KDD	**-2.12**	-2.14	-2.12
Plants	-13.30	**-13.22**	-13.79
Audio	**-40.22**	-41.2	-41.98
Jester	**-52.95**	-54.24	-53.47
Netflix	**-56.99**	-57.93	-58.41
Accidents	-29.86	**-29.05**	-33.64
Retail	-10.86	-10.83	**-10.81**
Pumsb-Star	-25.51	**-24.39**	-33.67
DNA	**-86.99**	-87.15	-92.67
Kosarek	**-10.61**	-10.7	-10.81
MSWeb	-9.78	**-9.74**	-9.97
Book	**-34.23**	-34.49	-34.97
EachMovie	**-51.78**	-53.72	-58.01
WebKB	**-153.38**	-154.83	-161.09
Reuters-52	**-85.60**	-86.35	-89.61
20NewsGrp.	**-152.16**	-153.87	-161.09
BBC	**-253.04**	-256.53	-253.19
AD	**-15.15**	-16.52	-31.78

Table 1: Comparision of test-set log likelihoods of our mixture-model (IL-Strudel) learned by mining contexts in the dataset with EM based ensembles of Strudel and LearnPSDD. Our reported numbers are the average over 5 runs. Bold values indicate the best performance among all mentioned approaches.

Setup For our independence-metric, we have used a bootstrapped version of pairwise-mutual-information, where the dataset is sampled with replacement 20 times and the pMI from all the datasets are averaged to obtain the final metric. Individual structures are expanded using the single learner from Strudel, with splitting heuristics "eFlow-vMI", and using cloning operations as highlighted in LearnPSDD, both with a depth of 1. For clones, the candidate with the maximum local log-likelihood change is preferred. EM is only used for parameter learning and the component structures are not altered. We set the number of EM-iterations to be 20 to ensure proper convergence. Grid search was performed over the number of mixture components as $\{5, 7, 10\}$ and the number of iterations for structure expansion as $\{30, 50, 150, 300\}$. The runs are repeated for 5 random seed values for each fixed set of hyperparameters, and the averaged results are reported. We use the open-source package Juice [Dang et al., 2021] for learning the structures of the components.

Results We have compared our mixture models to ones obtained from two PSDD learners , namely LearnPSDD and Strudel, when bagging is not used. Strudel initializes its mixture components by sharing a single structure across all components. The structure chosen is the best performing structure obtained by a single learner on a dataset. EM is only used for learning the mixture model parameters. EM-LearnPSDD includes both structure and parameter learning in different EM loops to build the ensemble. We use Strudel as our subrountine to learn component structures over disjoint sub-supports of the data distribution, such that prime sub independences approximately hold at the root level for each component structure.

Our experimental results show that our method (IL-Strudel) better or same as Strudel-EM and EM-LearnPSDD on 15 out of 20 datasets; see Table 1. Moreover, it beats them on 7 of the largest datasets in terms of variables including datasets like 20NewsGrp., BBC and AD. This shows the effectiveness of our initialization strategy in the context of PSDD mixture models. The improvement in performance on the larger datasets highlights the fact that a lot of potential exists when it comes to capturing context-specific-independence interactions between a large number of variables.

5 CONCLUSION

Through this paper, we successfully confirm the hypothesis posed at the beginning of the paper. We do this by first proving that maximizing log-likelihood of a structure on the training dataset is equivalent to finding a structure where the independences implied by the structure hold empirically. Using this, we propose a genetic algorithm based approach to find dataset examples such that structures expanded on such a dataset are a strong starting point for EM. Our experimental results show that using such structures learned on disjoint examples and combining them leads to a better overall ensemble model compared to random initialization and expansion.

Many important questions are still left to be answered. One would be to look at the use of 2-way mutual information instead of pairwise mutual-information for more robust measures. The current setup only factorizes the root node's distribution in the components. Exploring ways to do this for other levels as well could lead to more robust performance of structures. These can by taken up in future work.

References

Jessa Bekker, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck. Tractable learning for complex probability queries. In *NIPS*, December 2015.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, page 115–123, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 155860412X.

YooJung Choi and Guy Van den Broeck. On robust trim-

ming of bayesian network classifiers. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5002–5009. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/694.

Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In Manfred Jaeger and Thomas Dyhre Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 137–148. PMLR, 23–25 Sep 2020.

Meihua Dang, Pasha Khosravi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. Juice: A julia package for logic and probabilistic circuits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (Demo Track)*, Feb 2021.

Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams, 2014.

Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Marina Meila and Michael I Jordan. Learning with mixtures of trees. *JMLR*, 1:1–48, 2000.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 337–346, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

Tahrima Rahman and Vibhav Gogate. Learning ensembles of cutset networks. In *AAAI*, pages 3301–3307, 2016.

Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 630–645, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44851-9.

Robert E Schapire. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pages 149–171, 2003.

Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable operations for arithmetic circuits of probabilistic models. In *NIPS*, 2016.

A PROOFS

Semantics Before beginning the proofs, we highlight a modification in PSDD semantics we use for the rest of the proofs. The semantics we use are in the context of Probabilistic Circuits that have the *structured decomposability* and *determinism* properties, and do not specifically follow the semantics of PSDDs. Such semantics have been highlighted in [Dang et al., 2020]. Even with this change, *structured decomposability* and *determinism* are preserved in a structure. The definitions used from here on are presented below for the sake of clarity.

A decision node $(p_1, s_1, \theta_1) \dots (p_k, s_k, \theta_k)$ must be $deterministic$, which means that for any single world, it can have atmost one AND child assign a non-zero probability to it. In other words, the support sets of $[p_i] \wedge [s_i]; i = 1 \dots k$ must be disjoint.

We define the sub-context of a node to be the set of all AND nodes along a path to the node instead of the original PSDD semantics of primes. The context is still a disjunction of all the sub-contexts.

The distribution encoded by a decision node is defined as :

If n is a decision node $(p_1, s_1, \theta_1), \dots, (p_k, s_k, \theta_k)$ normalized for vtree node v, and v has left variables $\mathbf{X}$ and right variables $\mathbf{Y}$, then

$Pr_n(xy) = Pr_{p_i}(x) Pr_{s_i}(y) \theta_i$ for i where $xy \models [p_i] \wedge [s_i]$

Using the above modifications, it can be shown that the following porpositions hold as well :

If n is a decision node $(p_1, s_1, \theta_1), \dots, (p_k, s_k, \theta_k)$. Then we have $Pr_n([p_i] \wedge [s_i]) = \theta_i$.

Consider a PSDD r and n as one of its nodes. For a feasible context γ_n, $Pr_n(.) = Pr_r(.|\gamma_n)$.

The proofs for the above two propositions follows the same structure of the proofs as in the original PSDD paper and we refer to Kisa et al. [2014] for the details.

Notation We highlight the notation used in proofs.

$D(\alpha)$ denotes the number of examples in the training dataset that satisfy logical sentence α.

We assume the presence of a scope function ϕ that maps each PSDD node to its support set variables.

$D(.)$ denotes the total number of examples in the dataset.

$Pr_D(.)$ denotes the empirical data distribution. For a com-

plete instantiation of variables $\mathbf{v}$ in the dataset

$$Pr_D(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})}{\mathbf{D}(.)}$$

The no smoothing assumption for a PSDD r is used to denote the fact that the proabability under r for examples not in the data is zero.

Note : For all the remainder of the proof, we assume the base of the PSDD root to be a tautology and no smoothing. We also assume no sub is the constant False in the circuit.

Definition 1. *(Optimality)*

Define as PSDD n as optimal (with dataset D), if for each complete instantiation $\mathbf{v}$ of variables $\mathbf{V} = \phi(n)$ in the support set of n, $Pr_n(\mathbf{v}) = \mathbf{Pr_D}(\mathbf{v})$.

Definition 2. *(Faithfulness)*

A PSDD rooted at r is faithful if for every decision node n, we have that for every prime sub independence I(n) as defined in proposition 1, that is satisfied by node n, $I(n)$ empirically holds under dataset D.

Definition 3. *Let r be a PSDD root node. Let r be optimal with dataset D. We define the distribution of a sentence γ over $\phi(r)$, under $Pr_D(.)$ as : $Pr_D(\gamma) = \frac{D(\gamma)}{D(.)}$.*

Lemma 1. *For a PSDD r, let γ be a sentence over the variables $V = \phi(r)$. Let r be optimal with dataset D.*

Then, assuming no smoothing : $Pr_r(\gamma) = Pr_D(\gamma)$

Proof. $Pr_r(\gamma)$
$= \Sigma_{x|=\gamma} Pr_r(x)$ [x : complete instantiation of $\phi(r)$]
= $\Sigma_{x|=\gamma} Pr_D(x)$ [optimality]

Let x_D denote a complete instantiation of $\phi(r)$ that is present in dataset D and $x_{!D}$ denote a complete instantiation not present in dataset D.

Then we have :

$\Sigma_{x|=\gamma} Pr_D(x)$
$= \Sigma_{x_D|=\gamma} Pr_D(x_D) + \Sigma_{x_{!D}|=\gamma} Pr_D(x_{!D})$
$= \Sigma_{x_D|=\gamma} Pr_D(x_D)$
$= \frac{D(\gamma)}{D(.)} = Pr_D(\gamma)$ (Definition 3)

Thus we have that $Pr_r(\gamma) = Pr_D(\gamma)$ which concludes the proof. □

Lemma 2. *Let r be a PSDD. Let D be a dataset over $V = \phi(r)$. Let r be optimal with D. Let α and γ be sentences over V. Then, assuming no smoothing, we have :*

$$Pr_r(\alpha|\gamma) = Pr_D(\alpha|\gamma)$$

; where we define $Pr_D(\alpha|\gamma) = \frac{D(\alpha\wedge\gamma)}{D(\gamma)}$.

Proof. $Pr_r(\alpha|\gamma)$
$= \frac{\Sigma_{x|=\alpha\wedge\gamma} Pr_r(x)}{\Sigma_{x|=\gamma} Pr_r(x)}$
$= \frac{\Sigma_{x|=\alpha\wedge\gamma} Pr_D(x)}{\Sigma_{x|=\gamma} Pr_D(x)}$ [optimality]
$= \frac{\Sigma_{x|=\alpha\wedge\gamma} D(x)/D(.)}{\Sigma_{x|=\gamma} D(x)/D(.)}$
$= \frac{D(\alpha\wedge\gamma)}{D(\gamma)}$

Thus we have $Pr_D(\alpha|\gamma) = \frac{D(\alpha\wedge\gamma)}{D(\gamma)} = Pr_r(\alpha|\gamma)$ concluding the proof.

□

Lemma 3. *Let r be a PSDD optimal with dataset D. Let n be a decision node at some depth from r i.e. $n \neq r$. Let α be a sentence over $\phi(n)$. Then, assuming no smoothing, we have :*

$$Pr_n(\alpha) = Pr_r(\alpha|\gamma_n) = Pr_D(\alpha|\gamma_n)$$

; where γ_n is the context of given decision node n.

Proof. Using the properties of a PSDD (Kisa et al. [2014]), we have :

$Pr_n(\alpha) = Pr_r(\alpha|\gamma_n)$.

Combining this with Lemma-2, we have :

$$Pr_n(\alpha) = Pr_r(\alpha|\gamma_n) = Pr_D(\alpha|\gamma_n)$$

which concludes the proof. □

Proof of Theorem 1 With our introduced definitions, theorem 1 can alternately be rephreased as a PSDD is optimal iff it is faithful with the data. Thus, we break theorem 1 into two distinct theorems and prove each of them separately.

Theorem 2. *For a PSDD rooted at r, optimality implies faithfulness.*

Proof. We have that r is optimal with dataset D.

We just need to show that each prime sub independence is now satisfied under the data-distribution and we are done i.e. each decision node's induced independences are faithful to the dataset.

Let n be any arbitrary decision node. Let a be one AND child of n. Let $[a] = [p] \wedge [s]$. We denote $\gamma_a = \gamma_n \wedge [a]$ to represent the conjunction of the decision node's context and its prime base. Let $\mathbf{X} = \phi(p)$ and $\mathbf{Y} = \phi(s)$. Consider an instantiation **X=x, Y=y** in the support set of a.

The prime sub independence in proposition 1 states :

$Pr_r(\mathbf{xy}|\gamma_a)$
$= Pr_r(\mathbf{x}|\gamma_\mathbf{a})\mathbf{Pr_r}(\mathbf{y}|\gamma_\mathbf{a})$ (Prime Sub Independence)

$= Pr_D(\mathbf{x}|\gamma_a)Pr_D(\mathbf{y}|\gamma_\mathbf{a})$ (Lemma-2)
$= Pr_D(\mathbf{xy}|\gamma_a)$ (optimality of r).

The last step uses $Pr_r(\mathbf{xy}|\gamma_a) = Pr_D(\mathbf{xy}|\gamma_a)$

As n was arbitrary, this shows that the prime sub independence holds in the context of the data distribution for every decision node n, and hence any indepdence testing procedure should be able to validate it upto a level of significance. Thus r is faithful.

□

Theorem 3. *For a PSDD rooter at r, faithfulness implies optimality.*

Proof. We first define a relation between two PSDD nodes. A decision node m is said to be below n if the vtree node normalized for m is in the subtree of the vtree node normalizaed for n. The level of a vtree node is how high up it is in the vtree. The bottom-most level are the leaf vtree nodes. The highest level vtree node is the one corresponding to the PSDD root. The level of a PSDD node is the level of the vtree node it is normalized for.

We prove by induction over the levels of PSDD nodes.

<u>Base Case</u> :

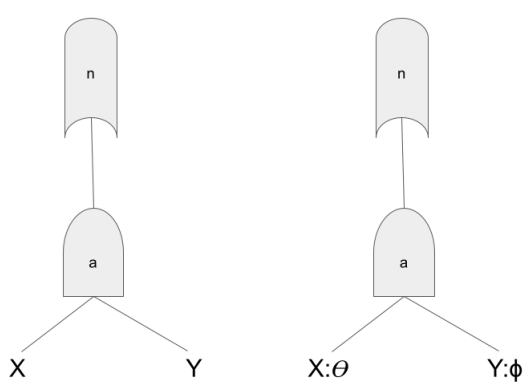


Figure 1.1 Figure 1.2

For our base case, we consider PSDD nodes normalized for vtree nodes one level above the leaves. Instances of such PSDD nodes can be seen in Figures 1.1 and 1.2.

Consider Fig. 1.1.

n may have multiple AND children. A single one is highlighted here as a. Let θ_a be the parameter corresponding to this AND child. This AND node has a support set for the instantiation $x = \top$ and $y = \top$. For these instantiations :

$Pr_r(xy|\gamma_n) = Pr_n(xy) = \theta_a Pr_X(x)Pr_Y(y)$ by definition.

Also, $Pr_X(x) = 1$ and $Pr_Y(y) = 1$.

This gives :

$$Pr_r(xy|\gamma_n) = \theta_a = \frac{D(X \wedge Y \wedge \gamma_n)}{D(\gamma_n)} = Pr_D(xy|\gamma_n)$$

This shows that for this (n, a) pair, the PSDD induced probability is the same as the one obtained from the dataset, for an instantiation in the support set of the AND node. Similarly, the cases for other AND nodes having different combinations of X and Y literals as their children can also be proven. Thus the inductive hypothesis is true for the PSDD node in Figure 1.1.

Consider Figure 1.2

We consider an initialization of variables as $x = \top$ and $y = \top$. The remaining cases can be proven in a similar manner. The distribution under n can be factorized as :

$$Pr_n(xy) = Pr_X(x)Pr_Y(y)$$

where $\theta_a = 1$ since there is only one AND child of n.

$Pr_X(x) = \theta$ and $Pr_Y(y) = \phi$.

By definition, we have :

$\theta = \frac{D(\gamma_n \wedge X)}{D(\gamma_n)} = Pr_D(x|\gamma_n)$

$\phi = \frac{D(\gamma_n \wedge Y)}{D(\gamma_n)} = Pr_D(y|\gamma_n)$

Thus, we have :

$$\begin{aligned} Pr_r(xy|\gamma_n) &= Pr_n(xy) = \theta\phi \\ &= Pr_D(x|\gamma_n)Pr_D(y|\gamma_n) \\ &= Pr_D(xy|\gamma_n) \end{aligned}$$

where the last step follows from faithfulness and the AND node represents a tautology as its base.

Thus the inductive hypothesis is true for the PSDD node in Figure 1.2. This concludes our proof showing that the hypothesis holding for the base case.

<u>Induction Case</u> :

We assume that the induction hypothesis holds for all PSDD nodes normalized for vtree nodes at level l. That means, for every PSDD node m at level l, we have :

$$Pr_r(\alpha|\gamma_m) = Pr_D(\alpha|\gamma_m)$$

for every logical sentence α over the support set of m.

Now consider a PSDD node n at level $l + 1$. We show that the inductive hypothesis holds for n. Let $\mathbf{X}$ and $\mathbf{Y}$ be the prime and sub variables corresponding to n. We consier an arbitrary instantiation $\mathbf{xy}$ of $\mathbf{X} \cup \mathbf{Y}$ in the support set of n.

Due to determinism, we have that only one AND child will satisfy the given instantiation. Let that child be a with

$[a] = [p] \wedge [s]$. Let the corresponding parameter associated be $\theta_a^{x,y}$.

Thus, we have :

$Pr_r(\mathbf{xy}|\gamma_n)$
$= \theta_a^{x,y} Pr_r(\mathbf{x}|\gamma_n \wedge [a]) Pr_r(\mathbf{y}|\gamma_n \wedge [a])$
$= \theta_a^{x,y} Pr_D(\mathbf{x}|\gamma_n \wedge [a]) Pr_D(\mathbf{y}|\gamma_n \wedge [a])$ (Induction Step)
$= \theta_a^{x,y} Pr_D(\mathbf{xy}|\gamma_n \wedge [a])$ (Faithfulness)
$= \frac{D(\gamma_n \wedge [a])}{D(\gamma_n)} \frac{D(x \wedge y \wedge \gamma_n \wedge [a])}{D(\gamma_n \wedge [a])}$ (Definition of the terms)
$= \frac{D(\mathbf{x} \wedge \mathbf{y} \wedge \gamma_n)}{D(\gamma_n)}$ (as $\mathbf{xy} \models [a]$)
= $\mathrm{Pr}_D(\mathbf{xy}|\gamma_n)$ (By definition)

Since we have shown the above hypothesis to hold for any arbitrary instantiation **X=x, Y=y** in the support set of a decision node n, we can conclude that it will hold for all instantiations in the support set. This completes the induction case and the proof.

□