Towards Compact Interpretable Models: Shrinking of Learned Probabilistic Sentential Decision Diagrams

Yitao Liang
Computer Science Department
University of California, Los Angeles
yliang@cs.ucla.edu

Guy Van den Broeck
Computer Science Department
University of California, Los Angeles
guyvdb@cs.ucla.edu

Abstract

Probabilistic sentential decision diagrams (PSDDs) were recently introduced as a tractable and interpretable representation of discrete probability distributions. PSDDs are tractable because they support a wide range of queries efficiently. They are interpretable because each parameter in the PSDD represents a conditional probability, as in Bayesian networks. This paper summarizes ongoing research that aims to answer two questions that are important to employ PSDDs as an explainable AI model. First, as a tractable and interpretable model, can PSDDs compete with more general machine learning models for density estimation? We answer this question positively, reporting state-of-the-art results on standard benchmarks. Second, can we effectively reduce the number of parameters in a learned PSDD to simplify its interpretation, without harming the quality of the learned model? For this task, we present an algorithm that merges PSDD substructures that are similar in KL-divergence, which we show can be done efficiently on PSDDs.

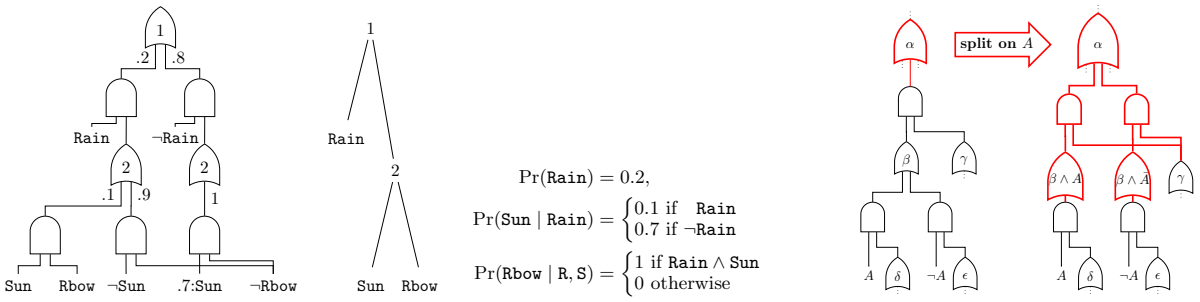
1 Introduction

Tractable learning aims to induce complex, yet tractable probability distributions from data (Domingos *et al.*, 2014; Mauro and Vergari, 2016). The learned tractable model serves as a certificate to the user that any query that arises can always be answered efficiently. Tractable learning initially targeted sparse graphical models (Meila and Jordan, 2000; Narasimhan and Bilmes, 2004; Chechetka and Guestrin, 2007). More recently, tractable circuit representations of probability distributions, such as *arithmetic circuits* (ACs) (Darwiche, 2003), have become the chosen target representation for these learners (Lowd and Domingos, 2008; Lowd and Rooshenas, 2013; Gens and Domingos, 2013; Dennis and Ventura, 2015; Bekker *et al.*, 2015), spurring innovation in arithmetic circuit dialects such as *sum-product networks* (SPNs) (Poon and Domingos, 2011; Peharz *et al.*, 2014) and *cutset networks* (Rahman *et al.*, 2014).

While closely related, these representations differ significantly in their properties, both in terms of their interpretability and their support for tractable queries. Our work considers the *probabilistic sentential decision diagram* (PSDD) (Kisa *et al.*, 2014a), which is perhaps the most powerful circuit proposed to date. Owing to their intricate structure, PSDDs stand out as being exceptionally interpretable: each PSDD parameter represents a conditional probability in the distribution, and the PSDD structure encodes an abundance of conditional independencies (Kisa *et al.*, 2014a). At the computational level, PSDDs support closed-form parameter learning, MAP inference, complex queries (Bekker *et al.*, 2015), and even efficient multiplication of distributions (Shen *et al.*, 2016), which are all exceedingly rare capabilities.

With these desirable properties, a key question is whether the PSDD representation can effectively be learned from data, and be competitive with other models for density estimation, such as Bayesian and Markov networks, or weaker types of tractable circuits. Liang *et al.* (2017) develop the first structure learning algorithm for PSDDs, called LearnPSDD. It uses local operations on the PSDD circuit that maintain the desired circuit properties, while steadily increasing model fit. LearnPSDD achieves state-of-the-art results on a large set of standard benchmarks. In this paper, we give a brief overview of LearnPSDD and its empirical performance.

A second question directly pertains to the explainability of PSDDs. While each PSDD parameter is individually interpretable as a conditional probability, LearnPSDD routinely learns circuits with tens of thousands of parameters, which hinders the interpretability of the model as a whole. Even though tractable learners trade off the likelihood of the model and its parameter count (a proxy for tractability), the learned model may be too large to interpret. To mitigate, we propose an algorithm to shrink PSDD circuits, reducing the number of parameters without affecting the likelihood. Our algorithm finds similar PSDD substructures, as measured by the KL-divergence between their distributions, and merges them. It is supported by an efficient algorithm to compute the KL-divergence between PSDDs.



(a) PSDD circuit (b) Vtree (c) Equivalent distribution (d) Minimal Split on A.

Figure 1: Examples of a PSDD, its vtree and distribution, and a split on an abstract PSDD. (After Liang *et al.* (2017).)

2 Background: PSDDs

Uppercase letters denote Boolean random variables. A lowercase complete instantiation $\mathbf{x}$ of variables $\mathbf{X}$ is a world, and $\mathbf{x} \models \alpha$ denotes that $\mathbf{x}$ satisfies sentence α.

Syntax and Semantics A probabilistic sentential decision diagram (PSDD) is a circuit representations of a joint probability distribution over binary variables. We refer to Kisa *et al.* (2014a) for a technical exposition and give a brief overview here. A PSDD is a parameterized directed acyclic graph, as depicted in Figure 1a. Each inner node is either a logical AND gate with two inputs, or a logical OR gate with an arbitrary number of inputs. The types of nodes alternate. Each terminal (input) node is a univariate distribution: X when X is always true, $\neg X$ when it is always false, or $(\theta : X)$ when it is true with probability θ. Each combination of an OR gate with its AND inputs is called a decision node. The left input to an AND gate is its prime (denoted p) and the right input is its sub (denoted s). The n wires in each decision node are annotated with a normalized probability distribution $\theta_1, \ldots, \theta_n$. Equivalently a decision node is represented as a set $\{(p_1, s_1, \theta_1), \ldots, (p_n, s_n, \theta_n)\}$.

Each PSDD node represents a probability distribution over its random variables. The inputs to an AND gate must represent *decomposable* distributions (i.e. over disjoint sets of variables). This is enforced uniformly throughout the circuit by a variable tree (vtree): a full, binary tree, whose leaves are labeled with variables. Intermediate vtree nodes partition variables into those appearing in the primes and subs of the corresponding PSDD decision nodes; see Figure 1b. Each PSDD node's distribution has an intricate support over which it defines a non-zero probability distribution. We refer to this set of worlds as the *base* of the PSDD node q, denoted $[q]$. For any single possible world and decision node, there is at most one prime input that assigns a non-zero probability to the world. That is, the support of each decision node's prime distributions has to be disjoint (a property called *determinism*). The probability of world $\mathbf{xy}$ according to decision node q factorizes recursively as

$$\Pr_q(\mathbf{xy}) = \theta_i \cdot \Pr_{p_i}(\mathbf{x}) \cdot \Pr s_i(\mathbf{y}) \text{ for } i \text{ s.t. } \mathbf{x} \models [p_i]$$

until it reduces to the univariate distributions at the terminals. Intuitively, each decision node branches based on which sentence $[p_i]$ is true, similar to how decision trees branch on the value of a single variable. We invite the reader to verify that the PSDD in Figure 1a represents the distribution shown in Figure 1c.

Interpretability From a top-down perspective, a PSDD repeatedly decomposes the distribution by conditioning it on the prime bases $[p_i]$. In each conditioned distribution, the prime and sub variables are independent. Independence given a logical sentence is called *context-specific independence* (Boutilier *et al.*, 1996). Moreover, to reach a node q through some path, all the primes on that path must be satisfied; they form the *subcontext* of the node. The disjunction of all sub-contexts forms the node's *context* γ_q. It gives us a way of precisely characterizing the parameter semantics of PSDD. Parameters θ_i of node q are conditional probabilities in root node r's overall distribution:

$$\theta_i = \Pr_r([p_i] \mid \gamma_q).$$

Inference and Parameter Learning PSDDs are a tractable representation: the probability of any assignment $\mathbf{x}$ can be computed in time linear in the PSDD size (its number of parameters), in a single bottom-up pass (Kisa *et al.*, 2014a). Second, PSDDs support efficient complex queries, such as count queries (Bekker *et al.*, 2015) and can be multiplied efficiently (Shen *et al.*, 2016). The maximum-likelihood estimates for the PSDD parameters are calculated in closed form by observing the fraction of complete examples flowing through the wire. That is, out of all the examples that agree with the node context γ_q, the fraction that also agrees with the prime base $[p_{q,i}]$ (Kisa *et al.*, 2014a).

3 PSDD Structure Learning

Liang *et al.* (2017) recently developed the first PSDD structure learning algorithm, called LearnPSDD. The objective of LearnPSDD is to obtain a compact PSDD that fits the data well. This section provides a high-level overview of that work (adapted from Liang *et al.* (2017)).

Operations Two local operations are proposed for LearnPSDD that change the PSDD structure: splitting and cloning. Splitting creates copies of an AND gate by constraining its prime and thereby changing its base. Cloning creates a structurally identical copy of a node but redirects some of the parents of the original node to the copy. A depth parameter d is used to specify the recursion depth to which these nodes are copied. When, $d = 0$, we call the operation minimal, and when d is infinity, we call the operation complete. Figure 1d depicts a minimal split. Node γ is still shared among the copies, as it exceeds depth d.

LearnPSDD Algorithm LearnPSDD incrementally improves the structure of an existing PSDD to better fit the data. In each iteration, the operation to execute is greedily chosen based on the best test-set likelihood improvement per size increment:

$$\text{score} = (\ln \mathcal{L}(r' \mid \mathcal{D}) - \ln \mathcal{L}(r \mid \mathcal{D}))/(\text{size}(r') - \text{size}(r))$$

where r is the original and r' the updated PSDD. The depth parameter d is fixed during learning. It is a critical parameter to tune in order to balance the learning speed and the tractability/explainability of the learned model.

Experiments After being extended to learn ensembles of PSDDs with bagging and EM, LearnPSDD achieves state-of-art results on standard benchmarks for density estimation (Liang *et al.*, 2017). An ensemble of PSDDs is equivalent to a single PSDD with a latent variable. LearnPSDD surpasses the state of the art[1] on 6 out of 20 datasets; see Table 1. This experiment shows that LearnPSDD performs competitively, despite the fact that PSDDs are a more interpretable, tractable, and restrictive representation than their alternatives.

4 Shrinking PSDDs

Both the interpretability and the tractability of a learned PSDD depend critically on its size. In LearnPSDD, this is largely a function of parameter d. This section reports ongoing work to control the size of the PSDD during learning by merging similar substructures with an algorithm called MergePSDD. This helps find the right trade-off between the number of parameters and the data fit, and eliminates the need to tune d.

Merge Operation Our merge operation takes as input two PSDD decision nodes that respect the same vtree and have the same base. It removes the larger node and redirects its parents to the remaining one; see Figure 2. The parameters of the modified substructure need to be re-estimated on the union of the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that flowed through the original nodes.

[1]Best-to-date is the best of ACMN (Lowd and Rooshenas, 2013), ID-SPN (Rooshenas and Lowd, 2014), SPN-SVD (Tameem Adel, 2015), ECNet (Rahman and Gogate, 2016a) and Merged L-SPN (Rahman and Gogate, 2016b).

Table 1: Comparison of test-data log-likelihood between LearnPSDD and the state of the art († denotes best).

Dataset	|Var|	LearnPSDD Ensemble	Best-to-Date
NLTCS	16	$-5.99^{\dagger}$	-6.00
MSNBC	17	$-6.04^{\dagger}$	$-6.04^{\dagger}$
KDD	64	$-2.11^{\dagger}$	-2.12
Plants	69	-13.02	$-11.99^{\dagger}$
Audio	100	-39.94	$-39.49^{\dagger}$
Jester	100	-51.29	$-41.11^{\dagger}$
Netflix	100	$-55.71^{\dagger}$	-55.84
Accidents	111	-30.16	$-24.87^{\dagger}$
Retail	135	$-10.72^{\dagger}$	-10.78
Pumsb-Star	163	-26.12	$-22.40^{\dagger}$
DNA	180	-88.01	$-80.03^{\dagger}$
Kosarek	190	$-10.52^{\dagger}$	-10.54
MSWeb	294	-9.89	$-9.22^{\dagger}$
Book	500	-34.97	$-30.18^{\dagger}$
EachMovie	500	-58.01	$-51.14^{\dagger}$
WebKB	839	-161.09	$-150.10^{\dagger}$
Reuters-52	889	-89.61	$-80.66^{\dagger}$
20NewsGrp.	910	-155.97	$-150.88^{\dagger}$
BBC	1058	-253.19	$-233.26^{\dagger}$
AD	1556	-31.78	$-14.36^{\dagger}$

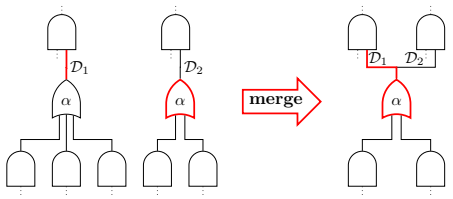


Figure 2: A merge operation. To-be-merged nodes have the same base α. The node with smaller size is retained.

Merge Heuristic MergePSDD incrementally improves the tractability of the learned model by repeatedly invoking the merge operation. Whereas each merge decreases the size of the learned model, it may increase or decrease the test-set likelihood. A heuristic is needed to select merges that are least likely to decrease the quality of the model. It is natural to only merge PSDDs that represent similar probability distributions. Similarity between distributions is commonly measured by their KL-divergence. However, KL-divergence cannot be directly applied to PSDDs: even if two PSDDs share the same set of variables, they may not share the same base (support), which invalidates the definition of KL-divergence. Therefore, we generalize the KL-divergence to the *Intersectional Divergence*, which is always defined between PSDDs that respect the same vtree.

Definition 1. (Intersectional Divergence D_I)
Given two PSDDs respecting the same vtree, m and n

$$D_I(m \parallel n) \stackrel{def}{=} \sum_{\mathbf{x} \models [m] \wedge [n]} \Pr_m(\mathbf{x}) \log \frac{\Pr_m(\mathbf{x})}{\Pr_n(\mathbf{x})}$$

This definition applies beyond PSDDs. Intuitively, it is the KL-divergence computed on the intersection of the supports of the two distributions. For nodes with the same base, intersectional and KL-divergence are equivalent. This is the way we use it in MergePSDD. Al-

Algorithm 1 `intersectional-divergence`(m, n)

input: PSDDs m and n that respect the same vtree.
output: Intersectional divergence $D_I(m, n)$
note: `pr-constraint`$(a, [b])$ is the probability of $[b]$ in PSDD a's induced distribution. See algorithm in Choi *et al.* (2015).
note: `cache` is loaded with divergences between terminals.
main:

```
if (m, n) ∈ in cache then return cache[(m, n)]
ρ ← 0
for each (p_i, s_i, θ_i) in decision node m do
    for each (r_j, t_j, β_j) in decision node n do
        ρ11 ← pr-constraint(s_i, [t_j])
        ρ12 ← pr-constraint(p_i, [r_j])
        ρ13 ← θ_i log θ_i/β_j
        ρ21 ← intersectional-divergence(p_i, r_j)
        ρ31 ← intersectional-divergence(s_i, t_j)
        ρ ← ρ + ρ11ρ12ρ13 + θ_iρ11ρ21 + θ_iρ12ρ31
cache[(m, n)] ← ρ
return ρ
```

gorithm 1 computes D_I efficiently (in quadratic time) using the following recursion.

Theorem 1. (D_I Calculation) Given a PSDD node $m = \{(p_1, s_1, \theta_1), (p_2, s_2, \theta_2) \dots\}$ and PSDD node $n = \{(r_1, t_1, \beta_1), (r_2, t_2, \beta_2)\}$ respecting the same vtree,

$$D_I(m \parallel n) = \sum_{i,j} \sum_{x \models [p_i] \wedge [r_j]} \sum_{y \models [s_i] \wedge [t_j]} \Pr_{p_i}(x) \Pr_{s_i}(y) \theta_i \left\{ \log \frac{\Pr_{p_i}(x)\theta_i}{\Pr_{r_j}(x)\beta_j} + \log \frac{\Pr_{s_i}(y)}{\Pr_{t_j}(y)} \right\}$$

$$= \sum_{i,j} \Pr_{s_i}([t_j]) \Pr_{p_i}([r_j]) D_{KL}(\theta_i \parallel \beta_j) + \theta_i \Pr_{s_i}([t_j]) D_I(p_i \parallel r_j) + \theta_i \Pr_{p_i}([r_j]) D_I(s_i \parallel t_j)$$

where $D_{KL}(\theta_i \parallel \beta_j) = \theta_i \log \frac{\theta_i}{\beta_j}$.

Merging Algorithm The MergePSDD algorithm starts from a (large) initial learned PSDD, for example obtained from LearnPSDD. It considers vtree nodes bottom-up. In each iteration, it finds all pairwise combinations of PSDD decision nodes that (i) respect the considered vtree node and (ii) have the same base. These pairs are candidates for a merge: the pair that yields the lowest intersectional divergence is chosen. The merge is first simulated and only permanently executed if the likelihood on validation data does not decrease.

Experiments We evaluate the effectiveness of MergePSDD with a focus on reduction in size. Experiments were conducted on a 16-core 2.6GHz Intel Xeon server with 256GB RAM. For each dataset in Table 2, two different PSDDs were obtained from LearnPSDD, using either greedy operations (complete split) or frugal operations (80% minimal operations and 20% depth-3 operations). Greedy LearnPSDD maximizes the learning speed, but PSDD size may be wasted. Frugal LearnPSDD better balances between size and learning speed. LearnPSDD was run until reaching the desired test-set likelihood, with a maximum of 24 hours. As expected, greedy LearnPSDD learns much larger models than frugal LearnPSDD; see Table 2.

MergePSDD was run on the models learned by greedy LearnPSDD for until all potential merge operations were exhausted, with a maximum of 6 hours. As shown in Figure 3 (comparing the left end of the brown line with the right end of the green line), MergePSDD effectively shrank the gap in size between the models learned by greedy LearnPSDD and frugal LearnPSDD. A full result on 6 datasets is reported in Table 2. It shows that MergePSDD is able to effectively reduce PSDD size, making the models more tractable and interpretable, without sacrificing too much model quality, by virtue of its KL-divergence heuristic.

Table 2: Number of parameters in PSDDs learned by LearnPSDD using frugal or greedy operations, and MergePSDD. LL is the desired test-set log-likelihood.

Dataset	Target LL	Frugal	Greedy	MergePSDD
NLTCS	−6.08	7491	31669	23471
MSNBC	−6.05	11074	17687	12943
KDD	−2.19	13814	29429	18921
Plants	−16.98	12021	12398	11574
Audio	−44.64	5804	5494	5389
Jester	−56.21	11774	16149	12349

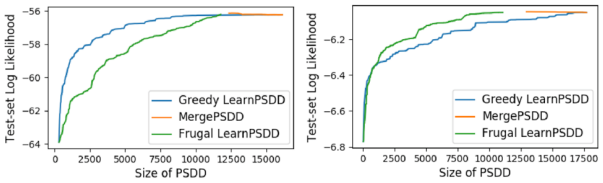


Figure 3: MergePSDD prunes away "unnecessary" PSDD structures while slightly improving performance.
Left: on dataset Jester. Right: on dataset MSNBC.

5 Conclusions

The two questions raised in this paper both received positive answers. First, LearnPSDD demonstrates the competitiveness of PSDDs in density estimation, even with structure learning from data instead of logical constraints (Kisa *et al.*, 2014b). Second, MergePSDD finds a better trade-off between learning speed and model size. Moreover, it was able to simplify PSDDs without a significant loss in quality. For future work, we hope to further decrease the size of PSDDs to any number required to make the model interpretable in practice.

Acknowledgements The authors thank Arthur Choi and Jessa Bekker for helpful discussions. This work is partially supported by NSF grants #IIS-1657613, #IIS-1633857 and DARPA XAI grant #N66001-17-2-4032.

References

Jessa Bekker, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck. Tractable learning for complex probability queries. In *Proceedings of NIPS*, pages 2242–2250, 2015.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of UAI*, pages 115–123, 1996.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Proceedings of NIPS*, pages 273–280, 2007.

Arthur Choi, Guy Van den Broeck, and Adnan Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Proceedings of IJCAI*, 2015.

A. Darwiche. A differential approach to inference in Bayesian networks. *JACM*, 50(3):280–305, 2003.

Aaron Dennis and Dan Ventura. Greedy structure search for sum-product networks. *Proceedings of IJCAI*, 2015.

P. Domingos, M. Niepert, and D. Lowd (Eds.). ICML workshop on learning tractable probabilistic models. 2014.

Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. In *Proceedings of ICML*, pages 873–880, 2013.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of KR*, pages 1–10, 2014.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams: Learning with massive logical constraints. In *ICML Workshop on Learning Tractable Probabilistic Models (LTPM)*, 2014.

Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of UAI*, 2017.

D. Lowd and P. Domingos. Learning arithmetic circuits. In *Proceedings of UAI*, pages 383–392, 2008.

Daniel Lowd and Amirmohammad Rooshenas. Learning markov networks with arithmetic circuits. In *Proceedings of AISTATS*, pages 406–414, 2013.

Nicola Di Mauro and Antonio Vergari. PGM tutorial on learning sum-product networks. 2016.

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

M. Narasimhan and J. Bilmes. PAC-learning bounded tree-width graphical models. In *Proceedings of UAI*, 2004.

Robert Peharz, Robert Gens, and Pedro Domingos. Learning selective sum-product networks. In *LTPM workshop*, 2014.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE, 2011.

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

Tahrima Rahman and Vibhav Gogate. Merging strategies for sum-product networks: From trees to graphs. In *Proceedings of UAI*, 2016.

T. Rahman, P. Kothalkar, and V. Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Proceedings of ECML PKDD*, pages 630–645, 2014.

Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of ICML*, pages 710–718, 2014.

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

Ali Ghodsi Tameem Adel, David Balduzzi. Learning the structure of sum-product networks via an svd-based algorithm. *Proceedings of UAI*, pages 32–41, 2015.