Look Ma, No Latent Variables: Accurate Cutset Networks via Compilation

Tahrima Rahman [*1] **Shasha Jin** [*1] **Vibhav Gogate** [1]

Abstract

Tractable probabilistic models obviate the need for unreliable approximate inference approaches and as a result often yield accurate query answers in practice. However, most tractable models that achieve state-of-the-art generalization performance (measured using test set likelihood score) use latent variables. Such models admit poly-time marginal (MAR) inference but do not admit poly-time (full) maximum-a-posteriori (MAP) inference. To address this problem, in this paper, we propose a novel approach for inducing cutset networks, a well-known tractable, highly interpretable representation that does not use latent variables and admits linear time MAR as well as MAP inference. Our approach addresses a major limitation of existing techniques that learn cutset networks from data in that their accuracy is quite low as compared to latent variable models such as ensembles of cutset networks and sum-product networks. The key idea in our approach is to construct deep cutset networks by not only learning them from data but also compiling them from a more accurate latent tractable model. We show experimentally that our new approach yields more accurate MAP estimates as compared with existing approaches and significantly improves the test set log-likelihood score of cutset networks bringing them closer in terms of generalization performance to latent variable models.

1. Introduction

A fundamental shortcoming of probabilistic graphical models (PGMs) (Pearl, 1988) such as Bayesian and Markov networks is that probabilistic inference—the process used to answer queries—on most models used in practice is intractable. This is not surprising since even the most basic inference task of computing the marginal probability of a variable given observations or evidence—posterior marginal estimation—is #P-hard (Roth, 1996). To circumvent this issue, one can either use approximate inference in lieu of exponential time exact inference approaches or learn models such that exact inference is tractable, namely takes time that is polynomial in the number of parameters. Both approaches are widely used in practice and each has their own pros and cons. Tractable models are desirable because the user can always recover accurate, reliable answers according to the model. This is a major plus over the approximate inference approach (e.g., sampling, belief propagation, etc.) because the latter often exhibits high variability; different runs and algorithms often yield widely different query answers. However, tractable models typically have slightly worse test set log-likelihood scores, namely they generalize poorly, as compared to arbitrary latent PGMs. Therefore improving the accuracy/fit of tractable models is an active area of research (Rahman & Gogate, 2016a;b; Di Mauro et al., 2016; 2017; Rashwan et al., 2016; Liang et al., 2017).

An often overlooked and one of the earliest goals of PGM research is learning interpretable models (Pearl, 1988; Darwiche, 2009). By interpretable models, we mean PGMs whose random variables, dependencies (structure) and parameters are interpretable. To this end, sparse Bayesian networks having no latent variables are interpretable but Markov networks are not because parameters of the latter are not interpretable. Interpretability is desirable, especially in interactive settings (Kulesza et al., 2015) and explainable AI applications (Gunning, 2017) such as activity recognition and medical diagnosis because it helps explain the model, specifically its assumptions to the user. This allows the user to update the model if the assumptions are wrong based on his/her prior knowledge. When the interpretable model is also tractable, it helps explain why a model made a particular decision as opposed to a different one to the user. Thus, learning tractable interpretable models is an important research endeavor, especially for building high quality explainable AI systems (Gunning, 2017).

In this paper, we focus on a particular class of tractable, interpretable models called cutset networks (Rahman et al., 2014). At a high level, these networks represent how a cutset conditioning method (Pearl, 1988) that takes advantage of determinism (Gogate, 2009), dynamic orderings, context-

[*]Equal contribution [1]Department of Computer Science, The University of Texas at Dallas, United States. Correspondence to: Tahrima Rahman <tahrima.rahman@utdallas.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

specific independence (Boutilier et al., 1996) and similar probability values (Chavira & Darwiche, 2008) would approximate an intractable PGM. Graphically, a cutset network is an OR tree (Dechter & Mateescu, 2007) in which each node is labeled with a random variable and each edge is labeled with the conditional probability of the child given an assignment of values from the root to the parent. The leaves of the cutset network are tree Bayesian networks defined over the variables not present on the path between the root and the leaf and therefore can be learned efficiently using the Chow-Liu algorithm (Chow & Liu, 1968).

Although simple and efficient to learn, a major shortcoming of cutset networks is that their test set log-likelihood score is worse, often by a significant amount, as compared to arbitrary PGMs. To alleviate this shortcoming, (Rahman & Gogate, 2016a; Di Mauro et al., 2016) proposed to learn mixtures of cutset networks (i.e., introduce a latent variable) by using various approaches such as the expectation maximization algorithm, Bagging, random forests and Boosting. Latent variables are also employed to improve generalization performance by other, related tractable models such as sum-product networks (SPNs) (Poon & Domingos, 2011) and ensembles of probabilistic sentential decision diagrams (Kisa et al., 2014; Liang et al., 2017) . Unfortunately, despite improved generalization performance, in presence of latent variables (full) maximum-a-posteriori (MAP) inference is no longer tractable (while marginal inference (MAR) is still tractable). As a result, application designers have to use unreliable approximate inference approaches to compute MAP estimates on these models. In other words, existing tractable models use latent variables to trade the accuracy of MAP inference with the accuracy of MAR inference.

The main contribution of this paper is to propose a new method for improving the accuracy of cutset networks without using latent variables. We propose a two-step, anytime approach. In the first step, we learn a tractable model having latent variables from data. In the second step, we take a generic, heuristic algorithm for learning the structure and parameters of cutset networks and replace the empirical statistics computed from data alone by a convex combination of the empirical statistics and the statistics computed from a latent tractable model (compilation). The key reason for using latent tractable models instead of Markov and Bayesian networks is that sufficient statistics can be inferred efficiently and accurately in the former (by performing polytime exact inference). Although, our approach is simple, it is quite powerful. In particular, we prove that the accuracy of cutset networks constructed using our approach will increase over time. We demonstrate via experiments on 20 benchmark datasets that our new method achieves test set log-likelihood scores that are only slightly inferior to state-of-the-art latent models while substantially outperforming the latter as well as existing methods that induce tractable models having no latent variables for MAP inference.

2. Related Work

Our work is related to work in the classification community by (Towell & Shavlik, 1994; Craven, 1996) who proposed to learn accurate, interpretable decision trees by overfitting a decision tree to a highly accurate deep neural network. Our work is different in that we are interested in modeling probability distributions that can solve much harder tasks such as structured prediction and posterior marginal inference. Our work is also related to the work on approximate compilation approaches (Gogate & Dechter, 2008; Lowd & Domingos, 2010; Gogate & Dechter, 2012; Friedman & Van den Broeck, 2018) that seek to construct an approximate tractable model from an intractable graphical model. The difference is that the aforementioned approaches do not use data and often learn tractable models having latent variables. Another line of work that is related to our work is work on learning tractable models without latent variables directly from data (Rahman et al., 2014; Lowd & Domingos, 2008; Rahman & Gogate, 2016b). Our work is different in that we leverage more accurate latent models to improve the performance of non-latent models.

3. Notation and Background

We assume that all random variables used in this paper are Boolean or binary valued. Note that we make this assumption for simplicity of exposition and the techniques presented in this paper can be easily extended to multi-valued random variables. Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ denote the set of n Boolean variables where each variable $X_i \in \mathbf{X}$ takes values from the domain $\{0, 1\}$. Let $\mathbf{x}$ denote an assignment of values to all variables in $\mathbf{X}$. We will denote the assignment $X_i = 1$ by x_i and $X_i = 0$ by $\overline{x_i}$. Given a subset $\mathbf{U}$ of $\mathbf{X}$, we denote by $\mathbf{x_U}$ as the projection of $\mathbf{x}$ on $\mathbf{U}$.

3.1. Bayesian Networks

Bayesian networks (BNs) (Pearl, 1988; Darwiche, 2009) are often used in practice to represent and reason about uncertainty. At a high level, they can be understood as a compact graphical representation of a joint probability distribution over a large number of random variables. Formally, a Bayesian network, denoted by $\mathcal{B}$ is a triple $\langle \mathbf{X}, G, \mathbf{P} \rangle$ where $G = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph such that $\mathbf{V}$ has one node for each variable in $\mathbf{X}$, $\mathbf{E}$ is a set of directed edges, and $\mathbf{P}$ is a set of conditional probability tables (CPTs). Each CPT $P_i \in \mathbf{P}$ is defined as $P_i(X_i|pa(X_i))$ where $pa(X_i)$ is the set of parents of X_i in G. A Bayesian network represents the following probability distribution $P_{\mathcal{B}}(\mathbf{x}) = \prod_{i=1}^{n} P_i(\mathbf{x}_{\{X_i\}}|\mathbf{x}_{pa(X_i)})$. Since the size of each CPT is exponential in the number of its parents, in practice,

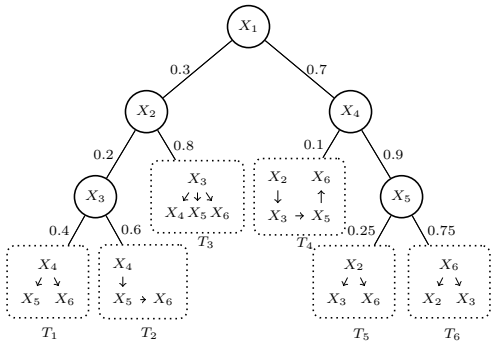


Figure 1. An example cutset network defined over the set of variables $\{X_1, X_2, X_3, X_4, X_5, X_6\}$.

for computational reasons, we assume that the number of parents is bounded by a constant.

The two key tasks in Bayesian networks are learning the structure and parameters from data and inference. We focus on two typical inference tasks in this paper: marginal inference (MAR) given evidence and full maximum-a-posteriori (MAP) inference.[1] It is well known that most tasks of interest in practice can be reduced to either MAR or MAP. Let the variables be partitioned into three (possibly empty) sets: evidence (or observed), non-evidence (or unobserved) and latent (or hidden). MAR is defined as finding the marginal probability distribution over each or a small subset of non-evidence variables given evidence while MAP is defined as finding the most probable assignment to all non-evidence (unobserved) variables given evidence. Both tasks are at least NP-hard and are computationally infeasible in practice. We say that a Bayesian network, and a probabilistic model in general, is MAR-tractable (MAP-tractable) when the MAR (MAP) task can be solved in time that scales polynomially with the number of variables. Examples of tractable Bayesian networks include tree Bayesian networks (Chow-Liu trees (Chow & Liu, 1968)), bounded treewidth Bayesian networks (Elidan & Gould, 2008) and networks having polynomial-sized arithmetic circuits (Darwiche, 2000).

3.2. Cutset Networks

Cutset networks (CNs) combine tree Bayesian networks with OR trees (probabilistic decision trees) (Dechter & Mateescu, 2007). Graphically, they can be depicted using a rooted OR tree with a tree Bayesian network at each leaf of the OR tree. Formally, a CN $\mathcal{C}$ is a pair $\langle \mathcal{O}, \mathcal{T} \rangle$ where $\mathcal{O}$ is an OR Tree having L leaves and $\mathcal{T}$ is a collection of L tree Bayesian networks attached to each of the L leaves. An OR tree is a rooted binary tree in which internal nodes are labeled with variables and each of the two edges emanating from a node represents conditioning of the variable with an appropriate value (either 0 or 1). We follow the convention that the left branch of a node labeled by X_i represents $\overline{x_i}$ while the right branch represents x_i. Each edge is labeled with the conditional probability of the variable taking the value given an assignment of values from the root to the parent node. $\mathcal{C}$ represents the following probability distribution

$$P_{\mathcal{C}}(\mathbf{x}) = \left(\prod_{(v_i,v_j) \in path_{\mathcal{O}}(\mathbf{x})} p_{i,j} \right) \left(T_{l(\mathbf{x})}(\mathbf{x}_{V(T_{l(\mathbf{x})})}) \right) \quad (1)$$

where $path_{\mathcal{O}}(\mathbf{x})$ is a unique path (sequence of edges) from the root to the leaf in $\mathcal{O}$ corresponding to the assignment $\mathbf{x}$, $p_{i,j}$ is the conditional probability attached to the edge between nodes v_i and v_j, $l(\mathbf{x})$ is the index of the leaf node along $path_{\mathcal{O}}(\mathbf{x})$, $T_{l(\mathbf{x})}$ is the tree Bayesian network in $\mathcal{T}$ at index $l(\mathbf{x})$ and $V(T_{l(\mathbf{x})})$ denotes the subset of variables over which $T_{l(\mathbf{x})}$ is defined.

Example 1. *Figure 1 shows an example cutset network defined over six variables* $\{X_1, \ldots, X_6\}$. *The probability of the assignment* $(0, 1, 1, 1, 0, 1)$ *to the six variables (where* X_1 *is assigned to* 0*,* X_2 *is assigned to* 1*, etc.) equals* $0.3 \times 0.8 \times P(X_3 = 1, X_4 = 1, X_5 = 0, X_6 = 1 | X_1 = 0, X_2 = 1)$ *where* 0.3 *equals the probability* $P(X_1 = 0)$ *(left branch of root node labeled by* X_1*),* 0.8 *equals the conditional probability* $P(X_2 = 1 | X_1 = 0)$ *(right branch of* X_2*) respectively and* $P(X_3 = 1, X_4 = 1, X_5 = 0, X_6 = 1 | X_1 = 0, X_2 = 1)$ *is computed from the tree Bayesian network* T_3 *(right child of the OR node labeled by* X_2*).*

A key feature of cutset networks is that they are both MAP-tractable and MAR-tractable (Rahman et al., 2014; Dechter & Mateescu, 2007) assuming that all variables in the OR tree as well as all tree Bayesian networks in $\mathcal{T}$ are known or observed (namely when the network has no latent variables). Other examples of tractable models that are both MAP- and MAR-tractable include bounded treewidth Bayesian networks (Bach & Jordan, 2002), Bayesian networks having polynomial sized arithmetic circuits (Lowd & Domingos, 2008) and probabilistic sentential decision diagrams (Liang et al., 2017). Another important feature of cutset networks is that their parameters have well-defined probabilistic semantics, i.e., they are interpretable and as a result enable smooth, reliable human-machine interaction (Gunning, 2017).

CNs can be learned from data by adapting typical top-down decision tree induction techniques for selecting a variable at each OR tree node and then using the Chow-Liu algorithm (Chow & Liu, 1968) for learning a tree Bayesian network at each leaf node when a pre-defined termination condition is satisfied. However, despite advances in learning algorithms (Vergari et al., 2015; Di Mauro et al., 2015), CNs are less accurate than arbitrary, intractable Bayesian networks.

To address this problem, (Di Mauro et al., 2015; Rahman & Gogate, 2016a;b) proposed using latent sum-product mixtures of cutset networks. In numerous empirical studies, it

[1] Full MAP inference is also called most probable explanation (MPE) inference in the Bayesian network literature.

has been shown that these networks achieve state-of-the-art test set log-likelihood scores. However, a key issue with these latent models is that they sacrifice MAP-tractability[2] while maintaining MAR-tractability.

4. Inducing Accurate Cutset Networks: A Novel Approach

The main goal of this paper is to improve the accuracy of cutset networks while maintaining both MAP and MAR tractability. Our main intuition is the following. Improving the model fit, measured using test set log-likelihood score, of MAR tractable models also substantially improves their MAR inference accuracy. In particular, numerous previous studies (Rooshenas & Lowd, 2014; Gens & Domingos, 2013) have shown that given two models, say M_1 (e.g., SPNs) and M_2 (e.g., BNs), such that: (1) M_1 is MAR-tractable; (2) M_2 is MAR-intractable; and (3) M_2 is slightly better than M_1 in terms of model fit, posterior marginal estimates obtained by performing exact inference on M_1 are far superior to those obtained by performing approximate inference on M_2 when both M_1 and M_2 are given the same amount of (reasonably bounded) time. We hope to achieve similar results for MAP inference, namely by improving the fit of MAP-tractable cutset networks (i.e., networks without latent variables), we hope to significantly improve the quality of their MAP estimates.

Next, we present our new method for learning accurate cutset networks. The key idea is to first learn a highly accurate tractable model with latent variables from data. We will denote this model by Q. We then combine statistics computed by performing exact marginal inference on Q with the ones empirically estimated from data to induce a cutset network. The main intuition behind this new approach is that we are unable to learn deep, accurate cutset networks in high dimensions from data alone because the variance increases with increasing depth. In particular, as we increase the depth d of a cutset network by one, the sufficient statistics used to estimate the remaining network at depth $d+1$ are not as reliable as the ones at depth d because the former are based on roughly half the number of training examples as compared to the latter. Therefore, to improve the accuracy of deep cutset networks we need to reduce the variance using an artifact other than the training data. In this paper, we propose to use Q, an auxiliary latent tractable model to achieve this objective.

Algorithm 1 describes the main steps in our approach. We call the algorithm LC-CN which stands for Learn/Compile Cutset Networks. The algorithm takes as input a dataset D defined over a collection of observed variables $\mathbf{X}$ and a tractable latent model representing a distribution Q (learned from D). The algorithm builds a cutset network via a top-down decision-tree style induction. At each recursive call, it outputs a Chow-Liu tree (steps 2-3) if the termination condition is met. Otherwise, it heuristically selects a variable X_i to condition on (steps 6-7) and then recurses on the 0 and 1 value assignments to X_i (steps 9-13).

Algorithm 1 LC-CN (D,Q)

Input : Training examples D defined over a set of variables $\mathbf{X}$ and a tractable latent model representing a distribution Q
Output : A Cutset network

1 **begin**
2 Compute pairwise marginal distribution $P(X_i, X_j)$ for all pairs (X_i, X_j) from Q and D (see Eq. (4)).
3 **if** *the termination condition is satisfied* **then**
4 **return** ChowLiuTree(P)
5 **else**
   `// Variable selection Heuristic`
6 Use P to compute $Score(X)$ for each variable X using Eq. (3).
7 X_i = variable with the highest Score.
8 Create a new internal node o labeled by X_i
9 Let l and r be the left and right child nodes of o respectively.
10 Label$(o, l) = P(\overline{x_i})$
11 l = LC-CN($D|\overline{x_i}, Q|\overline{x_i}$).
12 Label$(o, r) = P(x_i)$
13 r = LC-CN($D|x_i, Q|x_i$).
14 **return** o
15 **end**

Variable Selection Heuristic. Following previous work (Rahman et al., 2014; Di Mauro et al., 2015), we propose to use pairwise mutual information score to select the variable X_i in steps 6 and 7. This scoring function is based on the following intuition. Ideally, we should condition on a variable having strong dependencies with other variables, since conditioning on that variable would help us reach the termination condition faster. In other words, the conditioning operation is highly likely to yield a sparse, tree-like Bayesian network having fewer dependencies (edges). A popular approach for measuring dependence between two sets of variables is the mutual information score I:

$$I(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log \left(\frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})P(\mathbf{y})} \right) \quad (2)$$

However, estimating mutual information between a variable X_i and the remaining variables $\mathbf{X} \setminus \{X_i\}$ is computationally intractable because we need to sum over exponential number of combinations. Therefore, we approximate it using

[2] MAP inference in latent models is also called marginal MAP (MMAP) inference. MMAP is substantially harder because the latent variables need to be marginalized out before performing maximization (Liu & Ihler, 2013; Park & Darwiche, 2004).

the following, computationally efficient, pairwise mutual information score and select a variable having the largest score (breaking ties arbitrarily).

$$Score(X_i) = \sum_{j:j\neq i} I(X_i, X_j) \tag{3}$$

A key sub-step in computing $Score(X_i)$ is computing the marginal distribution $P(X_i, X_j)$ for all pairs of variables. When cutset networks are learned from data, we estimate $P(X_i, X_j)$ from data. Namely, counts of $(X_i = a, X_j = b)$ where $a, b \in \{0, 1\}$ in the data is the sufficient statistic. Since these counts will exhibit high variance as the depth increases, we propose to estimate $P(X_i, X_j)$ by combining the counts with estimates computed from the latent tractable model. More formally, we use

$$P(X_i, X_j) = \alpha Q(X_i, X_j) + (1 - \alpha)S(X_i, X_j) \tag{4}$$

where $\alpha \in [0, 1]$ is a constant (hyper-parameter), $Q(X_i, X_j)$ is computed from the distribution Q represented by the latent tractable model and $S(X_i, X_j)$ is the empirical distribution computed from the data counts. In general, computing $Q(X_i, X_j)$ is NP-hard. However, on tractable models, computing $Q(X_i, X_j)$ is poly-time and thus efficient. The hyper-parameter α controls the relative importance of S and Q. When $\alpha = 0$, P equals S and the latent model is not used. Similarly, when $\alpha = 1$, P equals Q and the data is ignored. In other words, the cutset network is *learned* from data when $\alpha = 0$ and *compiled* from Q when $\alpha = 1$. In practice, α can be set using the validation set.

Note that the Chow-Liu algorithm also uses pairwise mutual information to construct a tree Bayesian network. We propose to use $P(X_i, X_j)$ (see Eq. (4)) to compute pairwise mutual information for use in the Chow-Liu algorithm.

After selecting a variable X_i having the highest score, the algorithm conditions on it by creating an internal node o in the OR tree (step 8) and then recursively builds its left and right subtrees (steps 10–13). In the algorithm, the notation $D|x_i$ (similarly $D|\overline{x_i}$) denotes the dataset obtained by deleting all examples in which X_i equals 0 (similarly 1) and then removing the column for X_i. Also, the notation $Q|x_i$ (similarly $Q|\overline{x_i}$) denotes the conditional distribution obtained by setting X_i to 1 (similarly 0) in Q. Finally in step 14, the algorithm returns the new internal node o constructed in steps 6-13.

Termination Condition. Algorithm 1 can be easily modified to yield an anytime scheme by carefully setting the termination condition. In particular, we can perform an iterative deepening search by using a parameter and progressively increasing or decreasing the value of the parameter until a user-defined time bound is reached. One option is to use a bound on the maximum depth of the OR tree as a parameter and terminate the recursion when the bound is reached. A second option is to stop the recursion when the KL divergence between Q and P at a particular node is bounded by a small constant ϵ. In our experiments, we use the former approach.

Theoretical Properties of LC-CN. Next we formally describe the anytime properties of Algorithm 1 in Theorem 1. Specifically, the theorem shows that under the assumption that the latent tractable model (its distribution Q) is a more accurate approximation of the data generating distribution than the cutset network learned from data, the accuracy of the cutset network output by LC-CN can only increase or remain the same with increasing depth.

Theorem 1. *Let $\alpha = 1$ and R_d denote the distribution associated with the cutset network output by Algorithm LC-CN having maximum depth d, then $\mathcal{D}(Q, R_d) \geq \mathcal{D}(Q, R_j)$ where $j \geq d$ and $\mathcal{D}(Q, R)$ denotes the KL divergence between the distributions Q and R.*

Proof of Theorem 1 is presented in the supplementary material. Theorem 1 shows that learning deep cutset networks is a good idea because as the depth increases their performance will approach that of a superior latent model (Q).

We conclude this section by showing that each recursive step of LC-CN has only polynomial time complexity. In other words, LC-CN is a general-purpose, scalable algorithm for learning cutset networks.

Proposition 1. *The time complexity of Algorithm LC-CN is $O(n^2 \times \mathcal{V} \times (\mathcal{I} + N))$ where n is the number of variables, $\mathcal{V}$ is the number of nodes in the cutset network, $\mathcal{I}$ is the inference complexity to compute $Q(X_i, X_j)$, and N is the number of training examples.*

5. Experiments

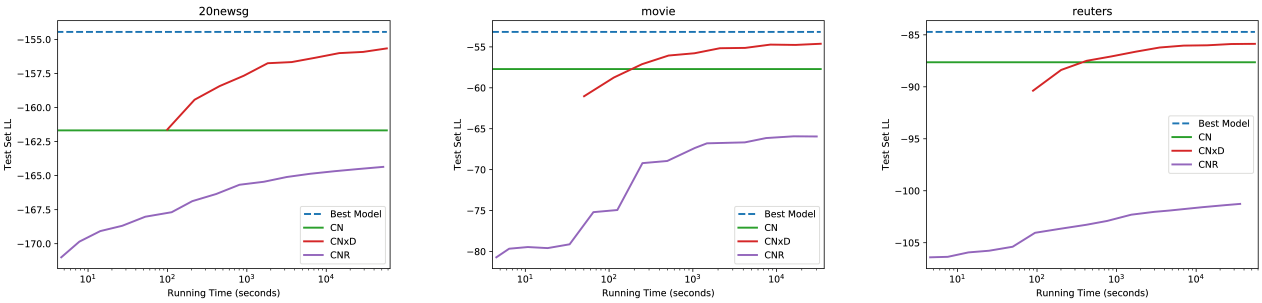
5.1. Setup

We evaluated the performance of cutset networks along two dimensions: (1) Model fit measured using the test set log-likelihood score and (2) MAP estimation quality measured using the log-likelihood of the MAP assignment. We used 20 benchmark datasets used in numerous prior studies (Rooshenas & Lowd, 2014; 2013; Gens & Domingos, 2013; Vergari et al., 2015) to evaluate our new algorithm. All the datasets are defined over binary valued variables with the number of variables ranging from 16 to 1556 (see Table 1).

We used mixtures of Chow-Liu trees (MTs) (Meila & Jordan, 2000) and bags of cutset networks (BCNs) as our choice of latent tractable models (Q) in Algorithm 1. Although any tractable model could have been used in Algorithm 1, for example SPNs, we chose MTs and BCNs because they admit faster inference and learning algorithms and on many datasets are as accurate as state-of-the-art methods. Moreover, since the complexity of each recursive call to our

Table 1. Average test set log-likelihood scores. Bold values indicate best scores obtained by CN, CNxD, CNR, ACBN, ACMN or PSDD.

Dataset	Dataset Characteristics				Test-set Log-Likelihood							
					No Latent Variables						Latent Variables	
	#Var	Train	Valid	Test	CNxD	CN	CNR	ACBN	ACMN	PSDD	MT	BCN
nltcs	16	16181	2157	3236	-6.01	-6.05	**-5.97**	-6.01	-6.00	-6.03	-6.01	-6.01
msnbc	17	291326	38843	58265	-6.07	-6.05	**-6.03**	-6.04	-6.04	-6.05	-6.04	-6.08
kdd	64	180092	19907	34955	**-2.15**	-2.19	-2.16	-2.16	-2.17	-2.16	-2.13	-2.14
plants	69	17412	2321	3482	**-12.73**	-13.25	-15.00	-12.85	-12.80	-14.96	-12.80	-12.32
audio	100	15000	2000	3000	-40.69	-41.97	-41.97	-41.13	**-40.32**	-42.53	-40.11	-40.09
jester	100	9000	1000	4116	-53.67	-55.26	-54.66	-54.43	**-53.31**	-57.57	-53.04	-52.69
netflix	100	15000	2000	3000	-57.48	-58.72	-59.15	-57.75	**-57.22**	-58.92	-56.67	-56.11
accidents	111	12758	1700	2551	-30.12	-30.66	-38.54	-27.15	**-27.11**	-34.13	-29.60	-29.88
retail	135	22041	2938	4408	**-10.84**	-10.98	-11.27	-10.87	-10.88	-11.13	-10.83	-10.84
pumsb*	163	12262	1635	2452	-23.57	-24.28	-36.16	-25.00	**-23.55**	-34.11	-23.64	-23.50
dna	180	1600	400	1186	-87.98	-87.50	-96.63	-80.23	**-80.03**	-89.11	-85.43	-82.17
kosarek	190	33375	4450	6675	**-10.74**	-11.07	-11.97	-10.92	-10.84	-10.99	-10.64	-10.76
msweb	294	29441	3270	5000	**-9.76**	-10.12	-11.12	-9.81	-9.77	-10.18	-9.83	-9.74
book	500	8700	1159	1739	**-35.31**	-37.51	-37.22	-36.02	-35.56	-35.90	-35.15	-34.94
movie	500	4524	1002	591	**-54.61**	-57.71	-65.95	-56.36	-55.80	-56.43	-54.02	-53.17
webkb	839	2803	558	838	**-155.77**	-161.58	-172.13	-159.85	-159.13	-163.42	-155.37	-155.10
reuters	889	6532	1028	1540	**-85.89**	-87.64	-101.16	-89.27	-90.23	-94.94	-85.76	-84.72
20newsg	910	11293	3764	3764	**-155.66**	-161.68	-164.34	-159.65	-161.13	-161.41	-154.45	-155.28
bbc	1058	1670	225	330	**-253.50**	-260.55	-271.98	-260.07	-257.10	-260.83	-259.15	-237.40
ad	1556	2461	327	491	**-15.40**	-16.14	-52.74	-16.47	-16.53	-30.49	-15.97	-15.34
Average					**-55.40**	-57.05	-62.81	-56.10	-55.78	-59.06	-55.33	-53.91



(a) 20newsg (b) movie (c) reuters

Figure 2. Average test set log-likelihood as a function of running time on three randomly chosen datasets. In the plots, "Best Model" indicates the best performing model between MT and BCN in terms of test set log-likelihood score.

algorithm is at least quadratic in the number of variables, inference time over the latent tractable models is a major bottleneck for learning deep cutset networks (see Theorem 1). As a result, MTs and BCNs offer the best alternative in terms of complexity and accuracy in single CPU settings.[3]

We learned MTs using the expectation-maximization algorithm (Meila & Jordan, 2000) and BCNs using the algorithm described in (Rahman & Gogate, 2016a). We varied the number of mixture components from 2 to 50 and bags from 2 to 40. We used a depth-bound of 5 in each bag for cutset networks. We used the validation set to choose the number of mixture components and bags for each dataset.

We evaluated three variants of cutset networks: (1) **CN**: networks learned from data using the algorithm described in (Rahman et al., 2014; Rahman, 2016) ; (2) **CNxD**: cutset networks learned using Algorithm 1 by exact inference over MTs or BCNs (Q) and data; and (3) **CNR**: randomly generated deep cutset networks, namely we use a random structure for both the OR tree and tree Bayesian network with BCNs used for parameter learning. The purpose of using CNRs is to evaluate whether structure learning is beneficial or not (or is parameter learning on a random deep cutset network sufficient for achieving state-of-the-art performance). CNRs are much faster to learn than CNxDs because the complexity of computing single variable marginals (parameter learning) is much lower than computing all pairwise marginals (required for structure learning in Algorithm 1).

[3]If parallel architectures or GPUs are used, complex architectures (e.g. SPNs) will be practically feasible in our algorithm and we leave this for future work.

Table 2. Average log-likelihood of the MAP completion of evidence over the test set (prediction accuracy) of CNxDs, CNs, CNRs, MTs and BCNs. Bold values indicate the best score obtained on a particular dataset and evidence settings by the model.

Datasets	20% Evidence					50% Evidence					80% Evidence				
	MAP Tractable			MAP Intractable		MAP Tractable			MAP Intractable		MAP Tractable			MAP Intractable	
	CNxD	CN	CNR	MT	BCN	CNxD	CN	CNR	MT	BCN	CNxD	CN	CNR	MT	BCN
nltcs	-3.32	-3.32	-3.22	-3.35	-3.36	-4.73	**-4.72**	-4.80	-4.80	-4.75	-5.58	-5.58	-5.51	-5.62	-5.59
msnbc	-3.56	-3.57	**-3.50**	-3.74	-3.57	-4.56	-4.63	**-4.43**	-4.65	-4.64	**-5.67**	-5.75	-5.68	-5.77	-5.74
kdd	-0.77	-0.77	-0.75	-0.77	-0.77	-1.31	-1.31	-1.39	-1.32	-1.32	-1.79	-1.79	-1.82	-1.79	-1.80
plants	**-7.54**	-8.02	-8.90	-8.52	-8.18	**-10.18**	-10.52	-11.61	-10.77	-10.49	**-11.63**	-11.83	-12.56	-11.88	-11.74
audio	**-19.29**	-19.49	-19.35	-19.72	-19.55	**-29.64**	-29.78	-30.30	-30.41	-30.14	**-35.92**	-35.97	-36.44	-36.31	-36.17
jester	-27.61	-28.73	**-27.65**	-28.72	-28.07	**-42.04**	-42.98	-42.45	-42.73	-42.88	**-48.39**	-48.81	-48.66	-48.60	-48.71
netflix	**-38.47**	-39.41	-42.01	-40.09	-40.16	**-47.01**	-47.61	-48.73	-49.13	-48.29	**-52.51**	-53.01	-54.36	-53.69	-53.30
accidents	-17.86	**-18.40**	-23.86	-19.83	-19.11	**-23.67**	-24.20	-29.63	-25.59	-25.48	**-27.47**	-27.98	-34.36	-28.71	-28.16
retail	**-4.34**	-4.44	-4.39	-4.47	-4.45	**-6.92**	-7.03	-7.15	-7.07	-7.10	**-8.78**	-8.89	-8.96	-8.90	-8.94
pumsb*	-15.81	**-16.06**	-23.55	-17.34	-16.42	**-19.79**	-20.16	-30.40	-20.96	-20.39	**-21.86**	-22.30	-28.74	-22.62	-22.38
dna	-68.32	**-74.57**	-76.14	-88.63	-81.02	**-74.25**	-78.33	-74.70	-82.50	-82.24	-70.25	-71.50	**-68.53**	-71.66	-71.65
kosarek	**-4.98**	-4.99	-5.41	-5.11	-5.12	**-7.28**	-7.37	-7.74	-7.39	-7.45	**-8.73**	-8.81	-9.03	-8.81	-8.05
msweb	-4.32	-4.48	**-4.19**	-4.51	-4.56	**-6.55**	-6.68	-7.16	-6.76	-6.73	**-8.16**	-8.27	-8.42	-8.33	-8.30
book	**-12.00**	-12.33	-12.30	-12.26	-12.38	**-21.18**	-21.85	-20.91	-21.68	-21.83	**-26.68**	-27.05	-27.54	-26.99	-27.31
movie	**-25.28**	-28.01	-29.38	-27.20	-23.76	**-38.14**	-41.38	-42.61	-40.18	-36.63	-46.33	-48.22	-48.82	-48.43	**-45.75**
webkb	**-58.22**	-58.91	-65.83	-60.65	-60.16	**-99.68**	-100.82	-106.47	-103.05	-102.68	**-115.27**	-116.20	-123.16	-116.93	-116.47
reuters	-31.86	**-32.41**	-41.89	-36.28	-34.07	**-54.38**	-54.97	-68.11	-56.46	-55.88	**-66.19**	-67.00	-69.25	-67.52	-67.09
20newsg	**-55.71**	-56.66	-59.57	-57.88	-58.81	**-96.45**	-98.19	-98.57	-98.14	-99.42	**-121.48**	-122.71	-126.85	-122.65	-122.96
bbc	**-104.08**	-105.08	-104.88	-106.48	-117.11	**-158.26**	-160.27	-165.78	-161.97	-166.17	**-177.10**	-178.42	-180.74	-180.17	-179.00
ad	**-9.74**	-9.96	-19.36	-10.81	-9.99	**-12.17**	-12.64	-25.58	-13.23	-12.62	**-13.19**	-13.67	-20.52	-13.96	-13.65
Average	**-25.65**	-26.48	-28.81	-27.82	-27.53	**-37.91**	-38.77	-41.43	-39.44	-39.36	**-43.65**	-44.19	-46.00	-44.47	-44.14

For all algorithms evaluated, we used a time bound of 48 hours and space bound of 4 GB. We used iterative deepening search for cutset networks, namely we start with a depth bound of 1 and progressively increase it until the time or space bound is reached. At each recursive call in Algorithm 1, we set the hyperparameter α (see Eq. (4)) using the validation set. We varied α from 0.0 to 1.0 in increments of 0.1. If the validation set or training set is empty at a particular recursive call (this happens frequently as the depth increases), we set α to 1, namely we use the latent model to compute the sufficient statistics.

5.2. Density Estimation

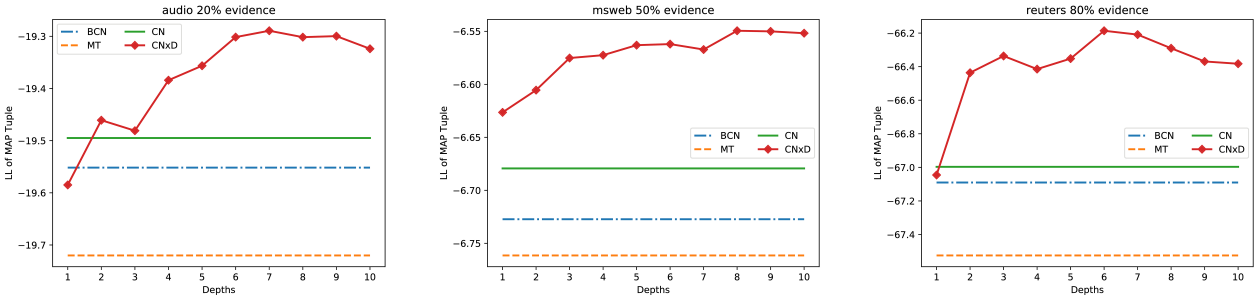
As a baseline, we also include results for three powerful tractable models that do not have latent variables: ACBNs (Lowd & Domingos, 2008), ACMNs (Rooshenas & Lowd, 2014) and PSDDs (Liang et al., 2017). We followed the prescription given in (Lowd & Domingos, 2008) and the libra toolkit (Lowd & Rooshenas, 2015) for learning ACBNs and ACMNs. For PSDDs, since the code is not publicly available, we report results from (Liang et al., 2017).

Table 1 shows the test set log-likelihood scores achieved by the individual models on each of the 20 datasets. We observe that CNxD performs better than CN on 18 out of the 20 datasets. CNxD also outperforms ACBNs, ACMNs and PSDDs on 16, 12 and 19 out of the 20 datasets respectively. Moreover, CNxDs are significantly better than CNRs (random deep cutset networks) especially as the dimensionality of the data increases. This demonstrates the power of our structure learning scheme.

Figure 2 shows the test set log-likelihood scores achieved by CNxD and CNR as a function of time for three datasets (The plots for the remaining datasets are presented in the supplementary material.). We see that as the depth (time) increases, the accuracy of CNxDs typically improves and approaches that of MTs and BCNs. CNs, on the other hand have much lower log-likelihood scores than MTs and BCNs. This shows that dependence on data restricts CNs; the latter performs a search over a much smaller, high variance hypothesis space resulting in shallow, inexpressive networks that generalize poorly on high-dimensional data. Combining estimations from latent models with data allows us to induce more accurate models.

5.3. Prediction Accuracy

We evaluate the prediction accuracy of cutset networks and latent tractable models by comparing the quality of their maximum-a-posteriori (MAP) estimates. To recap, in MAP estimation, we are interested in computing the most probable assignment to all the non-evidence (unobserved) variables given evidence. Since cutset networks are MAP-tractable, we can solve this task exactly and efficiently over them. In latent models (MTs and BCNs), this task is equivalent to a much harder marginal MAP inference task since the latent variable needs to be marginalized out before performing the maximization operation (Park & Darwiche, 2004) (MTs and



(a) 20% Evidence; audio (b) 50% Evidence; msweb (c) 80% Evidence; reuters

Figure 3. Average log-likelihood score of the MAP completion of evidence output by various algorithms as a function of the depth of the model on randomly chosen datasets and evidence percentages.

BCNs are MAR-tractable but not MAP-tractable). A popular approach that is used in practice (Liu & Ihler, 2013; Poon & Domingos, 2011) to approximate this intractable task in latent models is to replace the summation by maximization:

$$\max_{\boldsymbol{y}} \sum_{h} P(\boldsymbol{y}, h, \boldsymbol{e}) \approx \max_{\boldsymbol{y}} \max_{h} P(\boldsymbol{y}, h, \boldsymbol{e}) \qquad (5)$$

where $\boldsymbol{e}$ is the evidence, $\boldsymbol{y}$ is an assignment of values to all the non-evidence (query) variables and h denotes an assignment of a value to the latent variable.

In our experiments, we compare the quality of the MAP assignment output by this approximate marginal MAP inference algorithm on latent models with the one output by performing exact inference on cutset networks for each test example given evidence. We use the following experimental setup. Once models are learned, at test time, we randomly select $20\%, 50\%$ and 80% variables as evidence (observed) variables and the rest as query (unobserved) variables. We repeat this procedure over 5 runs and compute the average (the standard deviation was quite low).

We measure the quality of the MAP estimates using the following "oracle method." We treat BCNs learned by combining the training, validation and test set as an oracle. Let $\mathbf{E}$ and $\mathbf{Y}$ denote the set of evidence and query variables respectively such that $\mathbf{E} \cup \mathbf{Y} = \mathbf{X}$ and $\mathbf{E} \cap \mathbf{Y} = \emptyset$. For each test example $\mathbf{x}$, we set the evidence $\mathbf{x_E}$ and run an exact inference algorithm on each MAP-tractable model or the approximate inference technique described in Eq. (5) on each MAP intractable model to yield a MAP assignment $\mathbf{y}$ to the query variables. We define *MAP completion* of evidence as the full assignment $(\mathbf{x_E}, \mathbf{y})$ and compute the log-likelihood of $(\mathbf{x_E}, \mathbf{y})$ with respect to the distribution represented by the oracle. Under this measure, the network yielding the highest log-likelihood according to the oracle is the best one. We also evaluated F1 score and hamming loss of the MAP estimates and we report them in the supplement for lack of space. Note that unlike F1 score and hamming loss which score each variable independently, the "oracle method" models relationships between the query variables and is therefore more robust.

Table 2 and Figure 3 show the results. The plots in Figure 3 show the prediction accuracy of CNxDs as a function of the number parameters (depth) on three randomly chosen datasets (remaining plots are included in the supplement). We observe from Table 2 that CNxDs outperform CNs, CNRs, MTs and BCNs on a majority of datasets and evidence settings. CNs is the second best performing scheme while CNRs is the worst performing method. BCNs are slightly better than MTs. Our results clearly show the utility of ensuring MAP tractability as well as the utility of improving the fit of MAP tractable models. Specifically, MAP tractable models (CNs and CNxDs) are superior in terms of estimation quality to MAP intractable models (BCNs and MTs) and MAP tractable models having higher test set log-likelihood scores (CNxDs) yield better MAP estimates than models having lower test set log-likelihood scores (CNs).

6. Conclusion

In this paper, we presented a new approach for learning the structure of tractable, interpretable models called cutset networks. Unlike traditional data driven approaches for learning these models, we proposed to incorporate estimates computed from a highly accurate, latent tractable model. We showed via large scale experimental evaluation on high dimensional datasets that our unified approach that utilizes both the data and a superior latent model, learning from the former and compiling from the latter, significantly outperforms previous approaches on the MAP estimation task.

Future work includes using samples from latent Bayesian and Markov networks as Q in Eq. (4); applying our approach to arithmetic circuits, feature trees (Gogate et al., 2010), AND/OR graphs and probabilistic sentential decision diagrams; performing human subject studies to evaluate the interpretability of cutset networks; etc.

Acknowledgements

This work was supported in part by the DARPA Explainable Artificial Intelligence (XAI) Program under contract number N66001-17-2-4032, and by the National Science Foundation grants IIS-1652835 and IIS-1528037. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of DARPA, NSF or the US government.

References

Bach, F. R. and Jordan, M. I. Thin junction trees. In *Advances in Neural Information Processing Systems*, pp. 569–576, 2002.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-Specific Independence in Bayesian Networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 115–123, 1996.

Chavira, M. and Darwiche, A. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172 (6-7):772–799, 2008.

Chow, C. K. and Liu, C. N. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

Craven, M. W. *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, Department of Computer Sciences, University of Wisconsin – Madison, Madison, WI, 1996.

Darwiche, A. A differential approach to inference in bayesian networks. In *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pp. 123–132, 2000.

Darwiche, A. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

Dechter, R. and Mateescu, R. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171:73–106, 2007.

Di Mauro, N., Vergari, A., and Esposito, F. Learning accurate cutset networks by exploiting decomposability. In *Congress of the Italian Association for Artificial Intelligence*, pp. 221–232. Springer, 2015.

Di Mauro, N., Vergari, A., and Esposito, F. Multi-label classification with cutset networks. In *Conference on Probabilistic Graphical Models*, pp. 147–158, 2016.

Di Mauro, N., Vergari, A., Basile, T. M., and Esposito, F. Fast and accurate density estimation with extremely randomized cutset networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 203–219, 2017.

Elidan, G. and Gould, S. Learning bounded treewidth bayesian networks. *Journal of Machine Learning Research*, 9(Dec):2699–2731, 2008.

Friedman, T. and Van den Broeck, G. Approximate Knowledge Compilation by Online Collapsed Importance Sampling. In *Proceedings of the ICML Workshop on Tractable Probabilistic Models (TPM)*, 2018.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. In *International conference on machine learning*, pp. 873–880, 2013.

Gogate, V. *Sampling Algorithms for Probabilistic Graphical Models with Determinism*. PhD thesis, Computer Science, University of California, Irvine, USA, 2009.

Gogate, V. and Dechter, R. AND/OR importance sampling. In *24th Conference on Uncertainty in Artificial Intelligence*, pp. 212–219, 2008.

Gogate, V. and Dechter, R. Importance sampling-based estimation over AND/OR search spaces for graphical models. *Artificial Intelligence*, 184-185:38–77, 2012.

Gogate, V., Webb, W., and Domingos, P. Learning efficient Markov networks. In *Proceedings of the 24th conference on Neural Information Processing Systems*, pp. 748–756, 2010.

Gunning, D. Explainable artificial intelligence (xai), 2017. URL https://www.darpa.mil/program/explainable-artificial-intelligence.

Kisa, D., Van den Broeck, G., Choi, A., and Darwiche, A. Probabilistic sentential decision diagrams. In *14th International Conference on Principles of Knowledge Representation and Reasoning*, 2014.

Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pp. 126–137. ACM, 2015.

Liang, Y., Bekker, J., and Van den Broeck, G. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Liu, Q. and Ihler, A. Variational algorithms for marginal map. *The Journal of Machine Learning Research*, 14(1): 3165–3200, 2013.

Lowd, D. and Domingos, P. Approximate inference by compilation to arithmetic circuits. In *Advances in Neural Information Processing Systems*, pp. 1477–1485, 2010.

Lowd, D. and Domingos, P. M. Learning arithmetic circuits. In *24th Conference on Uncertainty in Artificial Intelligence*, pp. 383–392, 2008.

Lowd, D. and Rooshenas, A. The libra toolkit for probabilistic models. *The Journal of Machine Learning Research*, 16(1):2459–2463, 2015.

Meila, M. and Jordan, M. I. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1(Oct): 1–48, 2000.

Park, J. D. and Darwiche, A. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

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

Rahman, T. *Scalable learning approaches for Sum-Product Cutset networks*. PhD thesis, Department of Computer Science, The University of Texas at Dallas, Richardson, TX, 2016.

Rahman, T. and Gogate, V. Learning ensembles of cutset networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3301–3307, 2016a.

Rahman, T. and Gogate, V. Merging strategies for sum-product networks: From trees to graphs. In *Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 617–626, 2016b.

Rahman, T., Kothalkar, P., and Gogate, V. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 630–645. Springer, 2014.

Rashwan, A., Zhao, H., and Poupart, P. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Artificial Intelligence and Statistics*, pp. 1469–1477, 2016.

Rooshenas, A. and Lowd, D. Learning tractable graphical models using mixture of arithmetic circuits. In *AAAI (Late-Breaking Developments)*, 2013.

Rooshenas, A. and Lowd, D. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning*, pp. 710–718, 2014.

Roth, D. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

Towell, G. G. and Shavlik, J. W. Knowledge-Based Artificial Neural Networks. *Artificial Intelligence*, 70:119–165, 1994.

Vergari, A., Di Mauro, N., and Esposito, F. Simplifying, regularizing and strengthening sum-product network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 343–358. Springer, 2015.