FSPN: A New Class of Probabilistic Graphical Model

Anonymous authors
Paper under double-blind review

Abstract

We introduce factorize-sum-split-product networks (FSPNs), a new class of probabilistic graphical models (PGMs). FSPNs are designed to overcome the drawbacks of existing PGMs in terms of estimation accuracy and inference efficiency. Specifically, Bayesian networks (BNs) have low inference speed and performance of tree-structured sum-product networks(SPNs) significantly degrades in presence of highly correlated variables. FSPNs absorb their advantages by *adaptively* modeling the joint distribution of variables according to their dependence degree, so that one can simultaneously attain the two desirable goals—*high estimation accuracy* and *fast inference speed*. We present efficient probability inference and structure learning algorithms for FSPNs, along with a theoretical analysis and extensive evaluation evidence. Our experimental results on synthetic and benchmark datasets indicate the superiority of FSPN over other PGMs.

1 Introduction

Probabilistic graphical models (PGMs), a rich framework for representing probability distributions, have commonly been used for building *accurate* and *tractable* models for high-dimensional complex data. In comparison with deep generative models containing many hidden layers, which are "black-box" approximators, PGMs without latent variables tend to be much more interpretable and faster in inference. They are more appliable in lots of tasks such as online causal inference (Pearl et al., 2009) and query selectivity estimation (Getoor & Taskar, 2001), which have strict requirements on both inference accuracy and speed of the deployed models. Therefore, PGMs have recently re-attracted considerable research interests in the ML community (Zheng et al., 2018; Scanagatta et al., 2019; París et al., 2020). Much efforts (Rooshenas & Lowd, 2014; Vergari et al., 2015; Desana & Schnörr, 2020; Rahman & Gogate, 2016; Shao et al., 2019; Sharir & Shashua, 2018; Choi et al.; Rahman et al., 2014; Darwiche, 2009; Boutilier et al., 2013) have been devoted to improving the accuracy and tractability (inference speed) of PGMs.

Challenges of PGMs. In the most well-known class of PGMs—Bayesian networks (BNs), a set of random variables is modeled as a directed acyclic graph (DAG) where each variable (node) is conditionally independent of others given its parents. BNs can accurately and compactly model data distributions. However, although sometimes tractable, marginal probability inference on BNs is generally intractable. Even worse, structure learning of BN is NP-hard (Chickering, 1996). Another class of PGMs—Markov random fields (MRFs), model the joint probability density function (PDF) as an undirected graph. However, marginal probability inference on MRFs is also difficult (Koller & Friedman, 2009; Murphy, 2012). Even computing the likelihood of a single data point is intractable.

To improve tractability, (Poon & Domingos, 2011) proposed a new class of PGMs, the sum-product networks (SPNs). SPNs are recursively defined as weighted sums or products of smaller SPNs on simpler distributions. The most widely used SPN has a tree-structure, whose inference time is linear w.r.t. the number of nodes. They have high expressive efficiency on weakly correlated variables (Martens & Medabalimi, 2014). However, for highly correlated variables, their joint PDF is difficult to split into smaller ones where variables are locally independent. The learned SPN has large size and poor generality. Prior works have tried to extend SPNs by incorporating BNs or MRFs into SPNs (Rooshenas & Lowd, 2014; Vergari et al., 2015; Desana & Schnörr, 2020) or learning directed acyclic graph (DAG)-structured SPNs (Rahman & Gogate, 2016; Dennis & Ventura, 2015). However, these models either slow down the inference speed or hard to learn. Some works (Shao et al., 2019; Sharir & Shashua, 2018) designs variations of SPNs to model conditional PDFs, but they are not suitable for evidence or marginal probability computation.

In summary, existing PGMs still have some drawbacks in terms of estimation accuracy or inference speed. Designing highly accurate and tractable PGMs remains a challenging task.

Our Contributions. The key reason of existing PGMs' drawbacks arise from that they only utilize a single approach to decompose the joint PDF. BNs rely on conditional factorization, which is accurate but difficult for inference. Tree-structured SPNs use locally independent factorization, which cannot work well in presence of highly correlated variables. This naturally leads us to the following question: *if we could combine the strength of the two factorization approaches, is it possible to design a new type of PGM that is both accurate in estimation and fast for inference?* To this end, we propose factorize-sum-split-product networks (FSPNs), a versatile PGM aiming at this goal.

The main idea of FSPN is to *adaptively* decompose the joint PDF of variables based on their dependence degree. Specifically, FSPN separates the set of highly correlated variables from the rest by conditional factorization without loss of precision and processes each part accordingly. For highly correlated variables, their values are *interdependent*, so a multivariate PDF using some dimension reduction techniques (McGee & Carleton, 1970; Wang, 2015) is more suitable to model them. For the remaining weakly correlated variables, local independence commonly exists, so that their joint PDF can be split into small regions where they are mutually independent. FSPN recursively applies the above operations to model the joint PDF in a compact tree structure. We show that, FSPN is a very general PGM, which subsumes tree-structured SPNs and discrete BNs as special cases.

FSPNs absorb existing PGMs' advantages while overcoming their drawbacks. First, expressive efficiency of FSPNs is high, as two factorization approaches are used for variables with different dependence degree in an adaptive manner. Second, FSPN models are tractable, as their inference time is near linear w.r.t. its number of nodes. Third, structure learning of FSPNs is efficient. A locally optimal structure can be easily and efficiently obtained. In our evaluation, FSPNs consistently outperform other models on datasets with varied variable dependence degree in terms of estimation accuracy, inference speed, model size and training cost. On a series of PGM benchmark datasets, FSPNs also achieve comparable performance w.r.t. the state-of-the-art models. In summary, our contributions are as follows:

- We propose FSPNs, a novel and general class of PGMs that simultaneously attain high estimation accuracy and fast inference speed (Section 3).
- We design an efficient inference algorithm for FSPNs, which runs in near linear time w.r.t. its node size (Section 4).
- We devise an efficient structure learning algorithm for FSPNs, which returns a locally maximum likelihood structure (Section 5).
- We conduct extensive experiments to demonstrate the superiority of FSPNs on both synthetic and benchmark datasets (Section 6).

2 Background and Related Work

In this section, we briefly review some background knowledge and related work. Let $X = \{X_1, \ldots, X_m\}$ be a set of m random variables and $D \in \mathbb{R}^{n \times m}$ be a training data matrix sampled from $\Pr(X)$. A PGM aims at building a compact generative model $\Pr_{\text{D}}(X)$ on X such that: 1) $\Pr_{\text{D}}(X)$ can be efficiently learned to accurately approximate the joint PDF $\Pr(X)$; and 2) marginal probabilities can be inferred efficiently using the model on $\Pr_{\text{D}}(X)$. Building a PGM that fulfills these two goals remains non-trivial. We review two well-known classes of PGMs as follows.

Bayesian Networks (BNs) represent the joint PDF $\Pr(X)$ of X as a DAG based on the conditional independence assumption. Such a representation is exact when the BN structure accurately captures the causal relations among variables. Therefore, it is an expressive and explainable PGM. However, BNs have significant drawbacks in terms of inference and structure learning efficiency. First, marginal probability inference for BNs has a high time complexity and is sometimes even intractable. Exact inference methods, such as variable elimination and belief propagation (Koller & Friedman, 2009), have exponential time complexity w.r.t. its node size. Approximate inference methods, such as sampling (Gelfand, 2000; Andrieu et al., 2003) and loopy belief propagation (Murphy et al., 2013), reduce the time cost but sacrifice estimation accuracy. Second, the structure learning problem for BN is NP-hard. Exact methods rely on either combinatorial search (Chickering & Heckerman, 1997) in a super-exponential space of all DAGs or numerical optimization (Zheng et al., 2018; Yu et al., 2019) with high time complexity on computing the DAG constraints. Some approximate methods speed up the learning process by heuristic rules (Scanagatta et al., 2015; Ramsey et al., 2017), which may make the learned BN structure inaccurate for probability estimation. Some prior works have focused on learning tractable BNs by compiling into a circuit (Lowd & Domingos, 2012) or bounding tree width (Scanagatta et al., 2016). On a Chow-Liu tree, inference time is $O(nd^{h+1})$ where n, d and h represents the number of nodes, domain size and tree width, respectively. In our evaluation, the inference time on such BNs is still slow even when $h = 1$.

Unlike BNs, Markov random field (MRFs) model the joint PDF of variables as an undirected graph. The marginal probability inference on MRFs involve computing the normalizing factor of the potential functions, the complexity of which is exponential w.r.t. the tree width of the underlying graph. Therefore, MRFs are not suitable for computing marginal probabilities and are thus mainly used for data generation and pattern recognition (Li, 2009). Another tractable PGM related to BNs are cutset networks (Rahman et al., 2014), which can be regarded as an ensemble of tractable BNs.

Sum-Product Networks (SPNs) model the joint PDF by recursively applying two operations, namely sum and product, to split the joint PDF into simpler PDFs to capture contextual independence. Specifically, a sum node represents a weighted sum of mixture models as $\Pr_{D'}(X') = \sum_j w_j \Pr_{D'_j}(X')$ where D'_j and w_j are the data and weight of the i-th child. A product node partitions variables X' into mutually independent subsets $S_1, S_2, \ldots, S_d$ such that $\Pr_{D'}(X') = \prod_i \Pr_{D'}(S_i)$. A leaf node commonly maintains a univariate distribution $\Pr_{D'}(X')$ of a single variable X'. The marginal probability can be computed by a bottom-up traversal on SPNs, whose time cost is linear w.r.t. the number of nodes.

In the literature, tree-structured SPNs can be efficiently learned by a number of methods (Gens & Domingos, 2012; 2013; Peharz et al., 2013; Trapp et al., 2019). Thus, they are most widely used. However, tree-structured SPNs can not work well in presence of highly correlated variables in X. In this situation, a single product node is unable to split these variables and SPN would repeatedly add sum nodes to split D' into very small volumes, i.e., $|D'| = 1$ in extreme. This large structure has low inference speed and poor generality, which degrades its estimation quality.

In order to overcome the drawbacks of tree-structured SPNs, a number of attempts have been made to incorporate BNs or MRFs into SPNs. (Desana & Schnörr, 2020) designs SPGMs, a hybrid model of BNs and SPNs, where sum and product nodes are applied and their children may be BNs modeling partial variables in X. However, inference time for SPGMs is quadratic w.r.t. the maximum domain size of variables in X with a large factor. (Rooshenas & Lowd, 2014) and (Vergari et al., 2015) proposed SPN-BTB and ID-SPN, which apply BNs and MRFs to model $\Pr_{D'}(X')$ as multivariate leaf nodes where variables in X' can not be easily modeled by sum and product operations. They are more compact than tree-structured SPNs. However, the embedded BNs or MRFs may slow down the inference process. In our evaluation, their inference time is longer than tree-structured SPNs.

More general form of SPN structures are DAGs. DAG-structured SPNs tend to be much more compact and efficient for inference than tree-structured SPNs, as they merge redundant sub-structures into a singleton unit in DAGs. However, learning an optimal DAG-structured SPN is NP-Hard (Rahman & Gogate, 2016). Existing solutions obtains the DAG structure by greedy search or heuristic rules (Rahman & Gogate, 2016; Dennis & Ventura, 2015) over tree-structured SPNs, whose performance gain may be limited.

SPNs can be extended to model conditional PDFs $\Pr(Y|X)$. (Shao et al., 2019) proposed conditional SPNs (CSPNs). Each leaf node in CSPNs models $\Pr(Y_i|X)$ on a singleton variable Y_i. CSPNs are mainly used for point data and cannot be directly used to compute marginal probabilities on X. (Sharir & Shashua, 2018) proposed sum-product-quotient networks (SPQNs), where the quotient operation can model the conditional PDF by dividing the PDFs of its children. However, until now, SPQNs are only a theoretical model, and no structure learning methods have been proposed.

There also exist other types of PGMs, such as probabilistic sentential decision diagrams (PSDDs) (Kisa et al., 2014). PSDDs can model the PDF in presence of massive, logical constraints but only work on binary variables.

Summary. To model a joint PDF, existing PGMs apply two factorization approaches. BNs and some types of SPNs use *conditional factorization* to capture conditional independence, which is accurate but inefficient for inference and structure learning. Tree-structured SPNs use *independent factoriza-*

tion to capture contextual independence, which enables fast inference but has poor performance in presence of highly correlated variables. None of them can comprehensively fulfill the desired goals of PGMs. To resolve these drawbacks, we combine their advantages to design a new type of PGMs, which attains both estimation accuracy and inference efficiency at the same time.

3 THE FSPN MODEL

In this section, we propose the factorize-split-sum-product network (FSPN), a new class of PGM.

Main Idea. FSPN adaptively applies both conditional and independent factorization approaches to decompose variables X with different dependence degree. Specifically, let H be a set of highly correlated variables in X, e.g., whose pairwise correlation values are above certain threshold. A tree-structured SPN can not accurately and efficiently model $\Pr(X)$ in presence of H. Therefore, we first decompose $\Pr(X)$ as $\Pr(X) = \Pr(W) \cdot \Pr(H|W)$ where $W = X \setminus H$ by the conditional factorization approach and model each distribution accordingly.

For $\Pr(H|W)$, each value x of W specifies a conditional PDF $\Pr(H|x)$. To compactly model $\Pr(H|W)$, we partition W into multiple ranges $R_1, R_2, \ldots, R_t$ such that for any x, x' in the same R_i, $\Pr(H|x) = \Pr(H|x')$ roughly holds. Then, we only need to maintain one PDF $\Pr_i(H)$ for each R_i. As variables in H are highly correlated, we can model $\Pr_i(H)$ as a multivariate PDF using sparse distribution modeling (Wang, 2015) and dimension reduction techniques such as PCA/ICA and piece-wise regression (McGee & Carleton, 1970). For $\Pr(W)$, we can recursively separate each set of highly correlated variables from W using the above method until the remaining variables $W' \subseteq W$ are weakly correlated. As local contextual independence highly likely holds on W', we can apply the independent factorization approach to further decompose $\Pr(W')$. Specifically, $\Pr(W')$ is split into multiple small regions where variables in W' are locally independent. Then, for each region, we maintain a univariate distribution $\Pr(X_j)$ for each $X_j \in W'$.

Formal Definition. Given a set of variables X and training data matrix D, let $\mathcal{F}$ denote the FSPN modeling the joint PDF $\Pr_{\text{D}}(X)$. Each node N in $\mathcal{F}$ either represents a PDF $\Pr_{\text{D}'}(S)$ of a variable subset $S \subseteq X$ on a data subset $D' \subseteq D$; or a conditional PDF $\Pr_{\text{D}'}(S|C)$ of S conditioned on variables $C \subseteq X \setminus S$ on D'. The root node of $\mathcal{F}$ with $S = X, D' = D$ exactly represents $\Pr_{\text{D}}(X)$. Each node N decomposes the distribution $\Pr_{\text{D}'}(S)$ or $\Pr_{\text{D}'}(S|C)$ according to the following operations:

• *Factorize*: Given the PDF $\Pr_{\text{D}'}(S)$, let $H \subseteq S$ be a set of highly correlated attributes and $W = S \setminus H$. We have $\Pr_{\text{D}'}(S) = \Pr_{\text{D}'}(W) \cdot \Pr_{\text{D}'}(H|W)$. The factorize node generates the left child and right child to process the PDF $\Pr_{\text{D}'}(W)$ and conditional PDF $\Pr_{\text{D}'}(H|W)$, respectively.

• *Sum*: Given the PDF $\Pr_{\text{D}'}(S)$, we can partition the data D' into subsets $D'_1, D'_2, \ldots, D'_k$. The sum node creates the i-th child to process the PDF $\Pr_{\text{D}'_i}(S)$ for each $1 \leq i \leq k$. We can regard $\Pr_{\text{D}'}(S)$ as mixture models, i.e., $\Pr_{\text{D}'}(S) = \sum_i w_i \Pr_{\text{D}'_i}(S)$ where w_i is the weight of the i-th child.

• *Product*: Given the PDF $\Pr_{\text{D}'}(S)$, assume that we can partition S into mutually independent subsets of variables $S_1, S_2, \ldots, S_d$ on D', i.e., $\Pr_{\text{D}'}(S) = \prod_j \Pr_{\text{D}'}(S_j)$. Then, the product node creates the j-th child to process PDF $\Pr_{\text{D}'}(S_j)$ for each $1 \leq j \leq d$.

• *Uni-Leaf*: Given the PDF $\Pr_{\text{D}'}(S)$, if $|S| = 1$, we create a leaf node with the univariate distribution $\Pr_{\text{D}'}(S)$.

• *Split*: Given the conditional PDF $\Pr_{\text{D}'}(S|C)$, we partition the data D' into regions $D'_1, D'_2, \ldots, D'_t$ in a grid manner according to the domain space of C. The ℓ-th child models the conditional PDF $\Pr_{\text{D}'_\ell}(S|C)$. For each value c of C, $\Pr_{\text{D}'}(S|c)$ is modeled on exactly one child as $\Pr_{\text{D}'_\ell}(S|c)$. Note that the different semantic of split and sum nodes. A split node separately models $\Pr_{\text{D}'}(S|C)$ for different values of variables in C while a sum node decomposes the large model $\Pr_{\text{D}'}(S)$ into smaller models on S.

• *Multi-Leaf*: Given the conditional PDF $\Pr_{\text{D}'}(S|C)$, if S is independent of C on D', $\Pr_{\text{D}'}(S|c)$ stays the same for any value c of C. At this time, we create a leaf node to represent the multivariate distribution $\Pr_{\text{D}'}(S)$.

The above operations are recursively applied in order to model the joint PDF. In this paper, we focus on tree-structured FSPNs. However, a more general structure for FSPNs could be a DAG. We leave the exploration of DAG-structured FSPN for future work. Factorize, sum, product and uni-leaf nodes represent PDFs while split and multi-leaf nodes represent conditional PDFs. Figure 1 illustrates an FSPN example of the four variables in the data table. The two highly correlated attributes

X_1 🟥	X_2 🟦	X_3 🟨	X_4 🟩
2	5.1	3.1	21
4	8.5	5.6	1.6
9	17.6	8	7.3
...	...	...	...
1	2.8	3.7	6.1
0	0.8	6.1	0.9

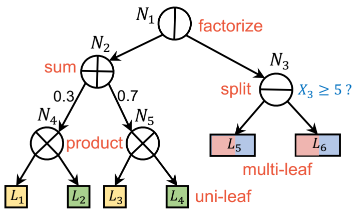


Figure 1: An data table of four variables and its corresponding FSPN.

X_1, X_2 are first separated from X_3, X_4 on node N_1. $\Pr(X_3, X_4)$ are decomposed by sum node (N_2) and product nodes (N_4, N_5). The uni-leaf nodes L_1, L_3 and L_2, L_4 models $\Pr(X_3)$ and $\Pr(X_4)$, respectively. $\Pr(X_1, X_2|X_3, X_4)$ are split into two regions by the value of X_3 on node N_3. On each region, X_1, X_2 are independent of X_3, X_4 and modeled as multi-leaf nodes L_5 and L_6.

Comparison with Other PGMs. For the remainder of this section, we compare our tree-structured FSPNs with other PGMs. First, we show that FSPNs subsume tree-structured SPNs and discrete BNs. Clearly, if we disable the factorize operation, an FSPN degenerates to the tree-structured SPN model; if we only apply the factorize, product, split, uni-leaf and multi-leaf operations, an FSPN could equally represent a discrete BN model. Due to space limits, we put the details on how a joint PDF represented by BN or tree-structured SP can be equivalently modeled by a FSPN in Appendix A. Based on the transformation process, we obtain the following proposition, which shows that the model size of an equivalent FSPN is bounded above by the model size of a tree-structured SPN or BN, so the expressive efficiency of FSPN is no worse than them.

Proposition 1 *Given a set of random variables X and data matrix D, if $\Pr_{\mathrm{D}}(X)$ can be represented by an SPN or a BN with model size M, then there exists an equivalent FSPN modeling $\Pr_{\mathrm{D}}(X)$ with model size no larger than M.*

Second, we show that FSPNs resolve the drawbacks of BNs and tree-structured SPNs. FSPNs combine the strengths of both conditional and independent factorization methods. Unlike tree-structured SPNs, FSPNs separately model highly correlated variables together as multi-leaf nodes, so an FSPN's structure tends to be more compact and allows more accurate probability estimates. Unlike the situation for BNs, the structure of FSPNs can be accurately and efficiently learned, and the probability inference on an FSPN is near linear time w.r.t. its number of nodes. We discuss the details of probability inference and structure learning for FSPNs in Section 4 and 5, respectively.

Third, unlike the case of SPGMs (Desana & Schnörr, 2020), FSPNs do not require an inference process on any BN sub-structures, thus more efficient. Unlike SPN-BTBs (Rooshenas & Lowd, 2014) and ID-SPNs (Vergari et al., 2015) with multi-leaf nodes on any possible subsets $X' \subseteq X$, the multi-leaf nodes in an FSPN only model highly correlated variables S. Notice that values in the joint PDF of S are very concentrated, so $\Pr(S)$ can be easily compressed and modeled in a lower dimension space. Whereas, for $\Pr(X')$, the storage cost for the exact PDF grows w.r.t. $|X'|$. Hence, SPN-BTBs and ID-SPNs use BNs or MRFs with low tree-width, which may degrade the estimation accuracy and inference speed. Furthermore, DAG-SPNs (Rahman & Gogate, 2016; Dennis & Ventura, 2015) could be subsumed by DAG-structured FSPNs.

Fourth, comparing FSPNs with CSPNs (Shao et al., 2019), we note the different goals behind modeling the conditional PDFs $\Pr(Y|X)$. CSPNs try to find local conditional independence between variables in Y and model $\Pr(Y_i|X)$ for each $Y_i \in Y$. Whereas, FSPNs try to find local independence between Y and X and model $\Pr(Y) = \Pr(Y|X)$ as multi-leaf nodes together. Unlike SPQNs (Sharir & Shashua, 2018), FSPNs use the factorize nodes to simplify the representation of joint PDF $\Pr(X, Y)$ using two simpler distributions $\Pr(X)$ and $\Pr(Y|X)$. Whereas, SPQNs model $\Pr(Y|X)$ using SPNs on $\Pr(X, Y)$ and $\Pr(X)$.

4 Probability Inference on FSPN

In this section, we describe the probability inference algorithm FSPN-Infer. In general, FSPN-Infer works in a recursive manner. It starts from the root node of the FSPN and computes the probability on different types of nodes accordingly. Specifically, for sum or product nodes, we accumulate the probabilities from children by weighted sum or multiplication, in a way similar to tree-structured SPNs. For each factorize node, we apply a divide-and-conquer process. Notice that,

```
Algorithm FSPN-INFER($\mathcal{F}, E$)
1:  let $N$ be the root node of $\mathcal{F}$
2:  if $N$ is uni-leaf node then
3:      return $\Pr_{D}(E)$ by the univariate PDF on $N$
4:  else if $N$ is a sum node then
5:      let $N_1, N_2, \ldots, N_t$ be the children of $N$ with weights $w_1, w_2, \ldots, w_t$
6:      $p_i \leftarrow$ FSPN-INFER($\mathcal{F}_{N_i}, E$) for each $1 \leq i \leq t$
7:      return $\sum_{i=1}^{t} w_i p_i$
8:  else if $N$ is a product node then
9:      let $N_1, N_2, \ldots, N_t$ be the children of $N$
10:     $p_i \leftarrow$ FSPN-INFER($\mathcal{F}_{N_i}, E$) for each $1 \leq i \leq t$
11:     return $\prod_{i=1}^{t} p_i$
12: else
13:     let $N_L$ be the left child on variables $W$ and $N_R$ be the right child
14:     let $L_1, L_2, \ldots, L_t$ be the multi-leaf nodes of $N_R$
15:     split $E$ into $E_1, E_2, \ldots, E_t$ by regions of $L_1, L_2, \ldots, L_t$
16:     get $p_i$ of $E_i$ on variables $X \setminus W$ from the multivariate PDF on $L_i$ for each $1 \leq i \leq t$
17:     $p_i \leftarrow$ FSPN-INFER($\mathcal{F}_{N_L}, E_i$) for each $1 \leq i \leq t$
18:     return $\sum_{i=1}^{t} p_i \cdot q_i$
```

each multi-leaf node in the right child of the factorize node specifies a range, within which the highly correlated variables are locally independent of the others. Hence, we first divide the range of the computed event into several parts by multi-leaf nodes. Then, for each part, the the probability of all highly correlated variables can be obtained directly from the multi-leaf node, and the probability of other variables could be recursively computed from the FSPN rooted at the left child of the factorize node. Finally, we multiply and sum them together to obtain the result.

Next, we formally describe the FSPN-INFER algorithm. Given the FSPN $\mathcal{F}$ modeling the PDF $\Pr_{D}(X)$, we can easily compute the marginal probability of an event of X. We can represent E in a canonical form as a hyper-rectangle: $X_1 \in [L_1, U_1], X_2 \in [L_2, U_2], \ldots, X_m \in [L_m, U_m]$, where each side of the closed interval can also be open. L_i and U_i represent the lower and upper bound of variable X_i, respectively. We have $L_i = -\infty$ or $U_i = \infty$ if E has no constraint on left or right side of X_i. If the constraint of E on a variable contains discontinuous intervals, we may split it into several events satisfying the above form. In this paper, we do not consider events with ranges that are not axis-aligned. FSPN-INFER takes the FSPN $\mathcal{F}$ and an event E as inputs, and outputs the probability $\Pr_{D}(E) = \Pr_{D}(X \in E)$. Let N be the root node of $\mathcal{F}$ (line 1). For any node N' in $\mathcal{F}$, let $\mathcal{F}_{N'}$ denote the FSPN rooted at N'. FSPN-INFER recursively computes $\Pr_{D}(E)$ by the following rules:

• In the base case (lines 2–3) where N is a uni-leaf node, we directly return the probability of E on the univariate PDF.

• If N is a sum (lines 4–7) or product node (lines 8–11), let $N_1, N_2, \ldots, N_t$ be all of its children. We can further call FSPN-INFER on $\mathcal{F}_{N_i}$ and E for each $1 \leq i \leq t$ to obtain the probability on the PDF represented by each child. Then, node N computes a weighted sum (for sum node) or multiplies (for product node) these probabilities together to obtain $\Pr_{D}(E)$.

• If N is a factorize node (lines 12–18), let N_L and N_R be its left child and right child. Assume N_L and N_R of N representing the PDF $\Pr_{D}(W)$ and the conditional PDF $\Pr_{D}(H|W)$, respectively. We have $\Pr_{D}(E) = \sum_{e \in E} \Pr_{D}(e_W) \cdot \Pr_{D}(e_H|e_W)$, where e_W and e_H represent the values of e on variables W and H, respectively. Let $L_1, L_2, \ldots, L_t$ be all multi-leaf nodes of N_R. Each L_i is defined on a sub-range of W and maintains the PDF $\Pr_i(H) = \Pr_{D}(H|w)$, which stays the same for all w in this sub-range. Each sub-range also forms a hyper-rectangle, which is ensured by our structure learning algorithm described in the next section. For different e_W, $\Pr_{D}(H|e_W)$ is represented by different PDFs on L_i. Therefore, we need to partition the range of E into $E_1, E_2, \ldots, E_t$ in terms of W to compute $\Pr_{D}(E)$. E_i represents the sub-range of E whose values of W fall in L_i, which could also be interpreted as a valid event since its range is also a hyper-rectangle. Then, we have

$$\Pr_{D}(E) = \sum_{e \in E} \Pr_{D}(e_W) \cdot \Pr_{D}(e_H|e_W) = \sum_{i=1}^{t} \sum_{e \in E_i} \Pr_{D}(e_H|e_W) \cdot \Pr_{D}(e_W) = \sum_{i=1}^{t} \left[\sum_{e_H \in E_i} \Pr_i(e_H) \cdot \sum_{e_W \in E_i} \Pr_{D}(e_W) \right]. \quad (1)$$

The probability $p_i = \sum_{e \in E_i} \Pr_i(e_H)$ of E_i on H could be directly obtained from the multi-leaf L_i. The probability $q_i = \sum_{e \in E_i} \Pr_{D}(e_W)$ of E_i on W can be obtained by calling FSPN-INFER on $\mathcal{F}_{N_L}$, the FSPN rooted at the left child of N, and E_i. Then we sum all $p_i \cdot q_i$ together to get $\Pr_{\mathrm{D}}(E)$. Similarly, for the evidence probability inference $\Pr_{\mathrm{D}}(Q|E=e)$ where Q, E are disjoint subsets of X. We can obtain $\Pr_D(Q, E=e)$ and $\Pr_D(E=e)$ on the FSPN to compute $\Pr_{\mathrm{D}}(Q|E=e)$.

We present a comprehensive example in Appendix B, which describes the probability inference process on an FSPN step by step. We now analyze the complexity of our inference algorithm. Computing the probability of any range on each leaf node can be done in $O(1)$ time (Gens & Domingos, 2013). Let f and l be the number of factorize and multi-leaf nodes in $\mathcal{F}$. The maximum number of ranges to be computed on each node is $O(l^f)$, so the inference time of FSPN is $O(l^f n)$. Actually, f tends to be a very small number (near $O(1)$) and the probability of multiple ranges could be computed in parallel. Therefore, we have the following proposition regarding the inference time cost of FSPN. In our evaluations, inference on FSPN is 1–3 orders of magnitude faster than BN and SPN. We reserve designing the FSPNs with theoretical bounds on l and f in the future work.

Proposition 2 *Given a FSPN $\mathcal{F}$ representing $\Pr_D(X)$ with n nodes and any event E of X, the marginal probability $\Pr_{\mathrm{D}}(E) = \Pr_{\mathrm{D}}(X \in E)$ can be obtained in near $O(n)$ time on $\mathcal{F}$.*

5 STRUCTURE LEARNING OF FSPN

In this section, we discuss the structure learning algorithm LEARN-FSPN of FSPN. LEARN-FSPN takes as inputs a data matrix D, two sets X and C of variables, and outputs the FSPN for $\Pr_{\mathrm{D}}(X|C)$. Initially, we can call LEARN-FSPN$(D, X, \emptyset)$ to build the FSPN on $\Pr_{\mathrm{D}}(X)$. LEARN-FSPN generally works in a top-down manner by recursively identifying different operations to decompose the joint PDF into small and tractable PDFs. Due to space limits, we put the pseudocode of LEARN-FSPN in Appendix C. The main steps of the algorithm are described as follows:

1. Separating highly and weakly correlated variables: when $C = \emptyset$, LEARN-FSPN detects whether there exists a set H of highly correlated attributes since the principle of FSPN is to separate them with others as early as possible. We find H by examining pairwise correlations and then group variables whose correlation value is larger than a threshold τ. If $H \neq \emptyset$, we add a factorize node to split $\Pr_{\mathrm{D}}(X)$. The left child and right child recursively call LEARN-FSPN to model $\Pr_{\mathrm{D}}(X \setminus H)$ and $\Pr_{\mathrm{D}}(H|X \setminus H)$, respectively.

2. Modeling weakly correlated variables: when $C=\emptyset$ and there do not exist highly correlated variables in X, we try to split $\Pr_{\mathrm{D}}(X)$ into small regions where variables in X are locally independent. Specifically, if X contains only one variable, we model the univariate distribution $\Pr_{\mathrm{D}}(X)$ using maximum likelihood estimation (MLE) of parameters. This can be done by applying a multinomial Dirichlet distribution for discrete variables or a Gaussian mixture model for continuous ones. Otherwise, we first try to partition X into mutually independent subsets using an independence test oracle. If X can be partitioned as $S_1, S_2, \ldots, S_k$, we add a product node and call LEARN-FSPN to model $\Pr_{\mathrm{D}}(S_i)$ on each child. If not, we apply an EM algorithm, such as k-means, to cluster instances D into $D_1, D_2, \ldots, D_t$, add a sum node and call LEARN-FSPN to model $\Pr_{\mathrm{D}_i}(X)$ on each child.

3. Modeling conditional PDFs: when $C \neq \emptyset$, it tries to model the conditional PDF $\Pr_{\mathrm{D}}(X|C)$. At this time, variables in X must be highly correlated. First, we test if X is independent of C on D by the oracle. If so, we can learn the MLE parameters of multivariate distribution $\Pr_D(X)$, such as the multi-dimensional Dirichlet distribution or Gaussian mixture model. If not, we use an EM algorithm, such as grid approximation of k-means, to partition the domain space of variables in C into multiple grids. Based on this, instances in D are split into $D'_1, D'_2, \ldots, D'_t$ where each D'_i represents all data points in a grid. Due to space limits, we put the details of the partition methods in Appendix C. Then, we add a split node and call LEARN-FSPN to model $\Pr_{\mathrm{D}'_i}(X|C)$ on each child.

Note that, LEARN-FSPN should be viewed as a framework rather than a specific algorithm, since we can choose different independent test oracles and clustering algorithms such as (Neal & Hinton, 1998). Moreover, any structure learning optimization (Bueff et al., 2018; Jaini et al., 2018; Kalra et al., 2018; Trapp et al., 2019) for SPNs can be also incorporated into LEARN-FSPN.

Next, we show that LEARN-FSPN returns an FSPN which locally maximizes the likelihood. The analysis proceeds in a bottom-up manner. First, on both uni-leaf and multi-leaf nodes, the parameters of distribution are learned using MLE. Second, the independence test oracle used in the product node factorize the joint PDF into a product of independent ones, which causes no likelihood loss. Third,

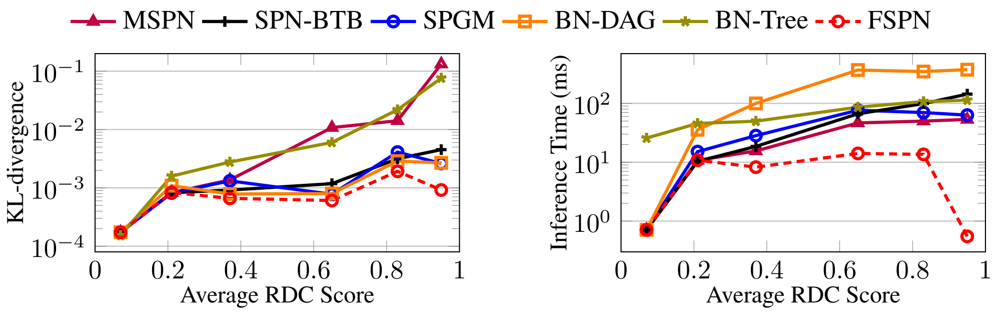


Figure 2: Evaluation results on synthetic data.

for the EM methods used in sum and split nodes, the nodes can locally maximize the likelihood if all children locally maximize the likelihood (Gens & Domingos, 2013). Fourth, the factorize node uses the exact probability conditional factorization, which causes no likelihood loss. Therefore, we have the following proposition.

Proposition 3 *Given a set of random variables X and data D,* LEARN-FSPN *can return a local MLE FSPN $\mathcal{F}$ of $\Pr_D(X)$ with independence test oracles and EM algorithms.*

6 Evaluation Results

In this section, we report the evaluation results on both synthetic and real-world benchmark data.

Results on Synthetic Data. First, we demonstrate the superiority of FSPNs over existing PGMs on six synthetic datasets with varied degree of dependence between variables. The dependence degree is evaluated as the average RDC score (Lopez-Paz et al., 2013) between variables, which ranges from 0.07 to 0.95. Each dataset contains 10^8 rows on 20 discrete variables with different domain size ranging from 5 to 100. We sampled 10^5 rows for training PGMs and 10^4 rows for tuning hyper-parameters. During testing, we randomly generate 10^2 evidence queries $\Pr(Q|E=e)$. We uniformly select 2–4 variables as Q, 6–10 variables as E and assign a value e of E. To measure inference accuracy we use the KL-divergence between the true distribution and the estimated one by the learned PGM.

We compare our FSPN with a variety of PGMs: MSPN is a tree-structured SPN learned using the method in (Molina et al., 2018) on mixed domains, which is shown to be better than the SPN learned by (Gens & Domingos, 2013). SPN-BTB (Rooshenas & Lowd, 2014) is a SPN structure with embedded BN as multivariate leaf nodes. The embedded BN is implemented by Chow-Liu tree whose tree width is bounded by 1. SPGM (Desana & Schnörr, 2020) is the model integrating BN and SPN. BN-DAG is a plain BN structure. BN-Tree is a BN implemented by Chow-Liu tree with bounded width of 1. The structure of BN-DAG and BN-Tree are learned by the Pomegranate package (Schreiber, 2018). Notice that, we do not find any open-source implementation of DAG-SPN (Rahman & Gogate, 2016). Other start-of-the-art variations of BNs and SPNs, described in Table 1 either work only on binary domains or do not support probability inference given evidence. Hence, we can not compare with them in our experiments.

For fairness, we apply an exhaustive grid search over the hyper-parameters of each model and report the best result. Figure 2 reports the evaluation results in terms of the two criteria: KL-divergence for inference accuracy and the inference time for efficiency. We clearly observe that:

• The estimation accuracy of FSPN is consistently better than all other models. In comparison with MSPN and BN-Tree, the KL-divergence decreases by up to $144\times$ and $82\times$, respectively. This is because the tree-structured SPN can not work well in presence of highly correlated variables, and the tree-structured BN sacrifices the model accuracy to improve its inference speed. In comparison with SPN-BTB, SPGM and BN-DAG, FSPN also decreases the KL-divergence. This verifies that FSPN can model the joint PDF more accurately.

• The inference time of FSPN is 1–3 orders of magnitude faster than others. In comparison with BN-DAG, SPN-BTB and BN-Tree, it improves the inference speed by up to $680\times$, $261\times$ and $206\times$, respectively. This verifies that the BN inference process is very time costly. Although bounding the

Table 1: Average test log-likelihoods on discrete datasets.

Dataset	# of vars	**FSPN (ours)**	LearnSPN	BayesSPN	SPGM	SPN-BTB	ID-SPN	MT	WinMine	ECNet	EM-PSDD
NLTCS	16	-6.05	-6.11	-6.02	**-5.99**	-6.01	-6.00	-6.01	-6.03	-6.00	-6.03
MSNBC	17	**-6.01**	-6.11	-6.03	-6.03	-6.03	-6.06	-6.07	-6.04	-6.04	-6.04
KDD	65	-2.14	-2.18	-2.13	-2.13	**-2.12**	**-2.12**	-2.13	-2.18	**-2.12**	**-2.12**
Plants	69	**-12.00**	-12.98	-12.94	-12.71	-12.09	-12.68	-12.95	-12.65	-12.78	-13.79
Audio	100	-40.02	-40.50	-39.79	-39.90	**-39.62**	-39.77	-40.08	-40.50	-39.73	-41.98
Jester	100	-52.39	-53.48	-52.86	-52.83	-53.60	-52.42	-53.08	**-51.07**	-52.57	-53.47
Netflix	100	-57.12	-57.33	-56.80	-56.42	-56.37	-56.36	-56.74	-57.02	**-56.32**	-58.41
Accidents	111	-26.99	-30.04	-33.89	-26.89	-28.35	-26.98	-29.63	**-26.32**	-29.96	-33.64
Retail	135	-10.83	-11.04	-10.83	-10.83	-10.86	-10.88	-10.83	-10.87	-10.82	**-10.81**
Pumsb-star	163	**-22.04**	-24.78	-31.96	-22.15	-22.66	-22.40	-23.71	-22.72	-24.18	-33.67
DNA	180	-80.97	-82.52	-92.84	**-79.88**	-80.07	-81.21	-85.14	-80.65	-85.82	-92.67
Kosarak	190	-10.66	-10.99	-10.77	**-10.57**	-10.58	-10.60	-10.62	-10.83	-10.58	-10.81
MSWeb	294	**-9.60**	-10.25	-9.89	-9.81	-9.61	-9.73	-9.85	-9.70	-9.79	-9.97
Book	500	**-33.81**	-35.89	-34.34	-34.18	-33.82	-34.14	-34.63	-36.41	-33.96	-34.97
EachMovie	500	-50.69	-52.49	-50.94	-54.08	**-50.41**	-51.51	-54.60	-54.37	-51.39	-58.01
WebKB	839	**-149.72**	-158.20	-157.33	-154.55	-149.85	-151.84	-156.86	-157.43	-153.22	-161.09
Reuters-52	889	-81.62	-85.07	-84.44	-85.24	**-81.59**	-83.35	-85.90	-87.55	-86.11	-89.61
20 Newsgrp	910	-155.30	-155.93	-151.95	-153.69	—	-151.47	-154.24	-158.95	**-151.29**	-161.09
BBC	1, 058	-252.81	-250.69	-254.69	-255.22	**-226.56**	-248.93	-261.84	-257.86	-250.58	-253.19
AD	1, 556	-15.46	-19.73	-63.80	-14.30	**-13.60**	-19.00	-16.02	-18.35	-16.68	-31.78

tree width of BN could speed up the inference, it is still much slower than FSPN. FSPN is also faster than MSPN since its structure is more compact.

We also evaluate the model size and training time of all models. Due to space limits, we put the results and analysis in Appendix D. In a nutshell, model size of learned FSPNs are up to two orders of magnitude smaller than others, and FSPNs' training time is several times faster than others except BN-Tree. In summary, this set of experiments validates the design choices of FSPNs, which provide both accurate results and fast inference.

Benchmark Testing. Next, we compare FSPN with the current state-of-the-art methods on 20 real-world benchmark datasets used in the literature (Gens & Domingos, 2013). Table 1 reports the test log-likelihood scores of FSPN and other PGMs. Specifically, LearnSPN is the tree-structured SPN from (Gens & Domingos, 2013). BayesSPN (Trapp et al., 2019) is an SPN with Bayesian structure learning. ID-SPN (Rooshenas & Lowd, 2014) use embedded MRF as leaf nodes to enhance the performance of SPN. MT stands for the mixture of tree models (Meila & Jordan, 2000). WinMine is one of the most sophisticated BN learning package (Chickering, 2002). ECNet (Rahman et al., 2014) and EM-PSDD (Liang et al., 2017) are the cutset network and PSDD with the best performance, respectively. To avoid the effects of hyper-parameters, we quote results of other PGMs from their original paper. We find that:

• Overall, FSPN outperforms LearnSPN, BayesSPN, MT, WinMine, ECNet and EM-PSDD. This is because LearnSPN and BayesSPN can not accurately model the joint PDF in presence of highly correlated variables. The expressiveness of MT is inherently low since its model complexity is not as high as others.

• The performance FSPN is comparable to SPGM, SPN-BTB and ID-SPN on the whole. It is slightly better than SPGM and ID-SPN but slightly worse than SPN-BTB. These models use embedded BNs or MRFs in their structure, so they are more accurate than other SPN models.

7 Conclusions

In this paper we propose FSPNs, a novel class of PGMs aiming at overcoming the drawbacks of existing PGMs. FSPN can adaptively model the joint distribution of variables with different dependence degree. It achieves high estimation accuracy and tractability at the same time. We design a near linear time marginal probability inference algorithm and a local MLE structure learning algorithm for FSPNs. Our extensive evaluation results on synthetic and benchmark datasets demonstrate that FSPNs attain superior performance. Based on these promising results, we affirmatively believe in that FSPNs may be a better alternative to existing PGMs in a wide range of ML applications. Moreover, we believe that there are many possible extensions of FSPNs worth researching in the future, such as supporting maximum a posterior inference, latent variable interpretation, DAG and Bayesian structure leaning of FSPNs and bounding their tree-width for tractability.

REFERENCES

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. *arXiv preprint arXiv:1302.3562*, 2013.

Andreas Bueff, Stefanie Speichert, and Vaishak Belle. Tractable querying and learning in hybrid domains via sum-product networks. *arXiv preprint arXiv:1807.05464*, 2018.

David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from Data*, pp. 121–130. Springer, 1996.

David Maxwell Chickering. The winmine toolkit. Technical report, Technical Report MSR-TR-2002-103, Microsoft, Redmond, WA, 2002.

David Maxwell Chickering and David Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models.

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

Aaron Dennis and Dan Ventura. Greedy structure search for sum-product networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Mattia Desana and Christoph Schnörr. Sum–product graphical models. *Machine Learning*, 109(1): 135–173, 2020.

Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pp. 3239–3247, 2012.

Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pp. 873–880, 2013.

Lise Getoor and Daphne Taskar, Ben andKoller. Selectivity estimation using probabilistic models. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pp. 461–472, 2001.

Priyank Jaini, Amur Ghose, and Pascal Poupart. Prometheus: directly learning acyclic directed graph structures for sum-product networks. In *International Conference on Probabilistic Graphical Models*, pp. 181–192, 2018.

Agastya Kalra, Abdullah Rashwan, Wei-Shou Hsu, Pascal Poupart, Prashant Doshi, and Georgios Trimponias. Online structure learning for feed-forward and recurrent sum-product networks. In *Advances in Neural Information Processing Systems*, pp. 6944–6954, 2018.

D Kisa, G Van den Broeck, and A Choi. Probabilistic sentential decision diagrams. In *International Conference on Principles of Knowledge Representation and Reasoning*, pp. 1–10, 2014.

Daphne Koller and Nir Friedman. *Probabilistic graphical models principles and techniques*. MIT Press, 2009.

Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

David Lopez-Paz, Philipp Hennig, and Bernhard Schölkopf. The randomized dependence coefficient. In *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

Daniel Lowd and Pedro Domingos. Learning arithmetic circuits. *arXiv preprint arXiv:1206.3271*, 2012.

James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

Victor E. McGee and Willard T. Carleton. Piecewise regression. *Regression, Journal of the American Statistical Association*, pp. 1109–1124, 1970.

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

Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pp. 355–368. Springer, 1998.

Iago París, Raquel Sánchez-Cauce, and Francisco Javier Díez. Sum-product networks: a survey. *arXiv preprint arXiv:2004.01167*, 2020.

Judea Pearl et al. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

Robert Peharz, Bernhard C. Geiger, and Franz Pernkopf. Greedy part-wise learning of sum-product networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 612–627, 2013.

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

Tahrima Rahman and Vibhav Gogate. Merging strategies for sum-product networks: from trees to graphs. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 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 *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3(2):121–129, 2017.

Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning*, 2014.

Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pp. 1864–1872, 2015.

Mauro Scanagatta, Giorgio Corani, Cassio P De Campos, and Marco Zaffalon. Learning treewidth-bounded bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pp. 1462–1470, 2016.

Mauro Scanagatta, Antonio Salmerón, and Fabio Stella. A survey on bayesian network structure learning from data. *Progress in Artificial Intelligence*, pp. 1–15, 2019.

Jacob Schreiber. Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, 18(164):1–6, 2018.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. *arXiv preprint arXiv:1905.08550*, 2019.

Or Sharir and Amnon Shashua. Sum-product-quotient networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 529–537. PMLR, 2018.

Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pp. 6347–6358, 2019.

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. 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.

Junhui Wang. Joint estimation of sparse multivariate regression and conditional graphical models. *Statistica Sinica*, pp. 831–851, 2015.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pp. 7154–7163, 2019.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pp. 9472–9483, 2018.

A GENERALITY OF FSPN

We present the details on how FSPNs subsume tree-structured SPNs, as well as discrete BNs, where all variables are discretized and all (conditional) probability distributions are stored in a tabular form.

• Given a set of variables X and data D, if $\Pr_D(X)$ could be represented by a tree-structured SPN $\mathcal{S}$, we can easily construct an FSPN $\mathcal{F}$ that equally represent $\Pr_D(X)$. Specifically, we disable the factorize operation in FSPN by setting the factorization threshold to ∞, and follow the same steps of $\mathcal{S}$ to construct $\mathcal{F}$. Then, the FSPN $\mathcal{F}$ is exactly the same of $\mathcal{S}$. Obviously, their model size is the same.

```
Algorithm BN-TO-FSPN($\mathcal{B}, N$)
1:  if $\mathcal{B}$ contains more than one connected component $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t$ then
2:      set $N$ to be a product node with children $N_1, N_2, \ldots, N_t$
3:      call BN-TO-FSPN($\mathcal{B}_i, N_i$) for each $i$
4:  else
5:      let $X_i$ be a node in $\mathcal{B}$ containing no out-neighbor
6:      if $X_i$ has no in-neighbor in $\mathcal{B}$ then
7:          set $N$ to be a uni-leaf node representing $\Pr_D(X_i)$
8:      else
9:          set $N$ to be a factorize node with left child $N_{\text{L}}$ and right child $N_{\text{R}}$
10:         set $N_{\text{R}}$ to be a split node
11:         for each value $y$ of $X_{\text{pa}(i)}$ in the CPT of $X_i$ do
12:             add a multi-leaf node $N_y$ as child of $N_{\text{R}}$
13:             let $D_y \leftarrow \{d \in D | X_{\text{pa}(i)}$ of $d$ is $y\}$
14:             let $N_y$ represent $\Pr_{D_y}(X_i)$
15:         remove $X_i$ from $\mathcal{B}$ to be $\mathcal{B}'$
16:         call BN-TO-FSPN($\mathcal{B}', N_{\text{L}}$)
```

• Given a set of variables X and data D, if $\Pr_{\text{D}}(X)$ can be represented by a discrete BN $\mathcal{B}$, we can also build an FSPN $\mathcal{F}$ that equally represent $\Pr_{\text{D}}(X)$. Without ambiguity, we also use $\mathcal{B}$ to refer to its DAG structure. We present the procedures in the BN-TO-FSPN algorithm. It takes as inputs a discrete BN $\mathcal{B}$ and a node N in $\mathcal{F}$ and outputs F_{N} representing the PDF of $\mathcal{B}$. We initialize $\mathcal{F}$ with a root node N. Then, BN-TO-FSPN works in a recursive manner by executing the following steps:

① (lines 1–3) If $\mathcal{B}$ contains more than one connected component $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t$, the variables in each are mutually independent. Therefore, we set N to be a product node with children $N_1, N_2, \ldots, N_t$ into $\mathcal{F}$, and call BN-TO-FSPN on $\mathcal{B}_i$ and node N_i for each i.

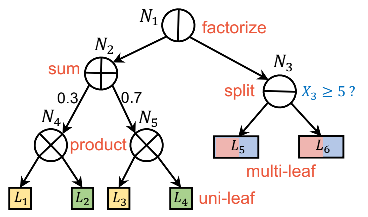
② (lines 5–7) If $\mathcal{B}$ contains only one connected component, let X_i be a node (variable) in $\mathcal{B}$ that has no out-neighbor. If X_i also has no in-neighbor (parent) in $\mathcal{B}$, it maintains the PDF $\Pr_{\text{D}}(X_i)$. At that time, we set N to be a uni-leaf representing the univariate distribution $\Pr_{\text{D}}(X_i)$.

③ (lines 9–16) If the parent set $X_{\text{pa}(i)}$ of X_i is not empty, X_i has a conditional probability table (CPT) defining $\Pr_{\text{D}}(X_i | X_{\text{pa}(i)}) = \Pr_{\text{D}}(X_i | X \setminus \{X_i\})$. At this time, we set N to be a factorize node with the left child representing $\Pr_{\text{D}}(X \setminus \{X_i\})$ and right child N_{R} representing $\Pr_{\text{D}}(X_i | X \setminus \{X_i\})$. For the right child N_{R}, we set it to be a split node. For each entry y of $X_{\text{pa}(i)}$ in the CPT of X_i, we add a leaf L_y of N_{R} containing all data D_y in D whose value on $X_{\text{pa}(i)}$ equals y. On each leaf L_y, by the first-order Markov property of BN, X_i is conditionally independent of variables $X \setminus \{X_i\} \setminus X_{\text{pa}(i)}$ given its parents $X_{\text{pa}(i)}$. Therefore, we can simplify the PDF represented by L_y as $\Pr_{\text{D}}(X_i|y) = \Pr_{\text{D}_y}(X_i)$. Therefore, N_{R} characterizes the CPT of $\Pr_{\text{D}}(X_i | X_{\text{pa}(i)}) = \Pr_{\text{D}}(X_i | X \setminus \{X_i\})$.

Later, we remove the node X_i from $\mathcal{B}$ to be $\mathcal{B}'$, which represents the PDF $\Pr_{\text{D}}(X \setminus \{X_i\})$. We call BN-TO-FSPN on $\mathcal{B}'$ and node N_{L}, the left child of N to further model the PDF.

Finally, we obtain the FSPN $\mathcal{F}$ representing the same PDF of $\mathcal{B}$. Next, we analyze the model size of $\mathcal{B}$ and $\mathcal{F}$. The storage cost of each node X_i in $\mathcal{B}$ is the number of entries in CPT of $\Pr_{\text{D}}(X_i | X_{\text{pa}(i)})$. The FSPN $\mathcal{F}$ represents $\Pr_{\text{D}}(X_i)$ in step ② when $X_{\text{pa}(i)}$ is empty and $\Pr_{\text{D}}(X_i|y)$ for each value y of $X_{\text{pa}(i)}$ in step ③. In the simplest case, if $\mathcal{F}$ also represents the distribution in a tabular form, the storage cost is the same as $\mathcal{B}$. Therefore, the model size of $\mathcal{F}$ can not be larger than that of $\mathcal{B}$.

Consequently, proposition 1 holds.



Ranges and Probabilties of Events

Events	X_1 🟥	X_2 🟦	X_3 🟨	X_4 🟩	Probability
E	[1, 7]	[0, 20]	[3, 6]	[0, 20]	0.171
E_1	[1, 7]	[0, 20]	[3, 5]	[0, 20]	0.051
E_2	[1, 7]	[0, 20]	(5, 6]	[0, 20]	0.12

Figure 3: An example of probability inference on FSPN.

B Example of Probability Inference on FSPN

We show an example of probability inference on the FSPN in Figure 1. This FSPN models the joint PDF on four variables X_1, X_2, X_3, X_4. The two highly correlated attributes X_1, X_2 are first separated from X_3, X_4 on node N_1. $\Pr(X_3, X_4)$ are decomposed by sum node (N_2) and product nodes (N_4, N_5). The uni-leaf nodes L_1, L_3 and L_2, L_4 models $\Pr(X_3)$ and $\Pr(X_4)$, respectively. $\Pr(X_1, X_2 | X_3, X_4)$ are split into two regions by the value of X_3 on node N_3. On each region, X_1, X_2 are independent of X_3, X_4 and modeled as multi-leaf nodes L_5 and L_6.

We assume that the domain of each variable is $[0, 20]$. We consider the example event $E : X_1 \in [1, 7], X_3 \in [3, 6]$, whose canonical form is $E : X_1 \in [1, 7], X_2 \in [0, 20], X_3 \in [3, 6], X_4 \in [0, 20]$. We then obtain its probability on the FSPN by following the procedures step by step in Figure 3.

First, we consider the factorize root node N_1. The right child N_3 splits the domain on whether X_3 is greater than 5 to the two multi-leaf nodes L_5 and L_6. The domains on L_5 and L_6 could be represented as $X_1 \in [0, 20], X_2 \in [0, 20], X_3 \in [0, 5], X_4 \in [0, 20]$ and $X_1 \in [0, 20], X_2 \in [0, 20], X_3 \in (5, 20], X_4 \in [0, 20]$, respectively. We divide the range of E into two parts by intersecting with the domains of L_5 and L_6 as $E_1 : X_1 \in [1, 7], X_2 \in [0, 20], X_3 \in [3, 5], X_4 \in [0, 20]$ and $E_2 : X_1 \in [1, 7], X_2 \in [0, 20], X_3 \in (5, 6], X_4 \in [0, 20]$. Obviously, we have $\Pr(E) = \Pr(E_1) + \Pr(E_2)$.

Second, we consider how to compute $\Pr(E_1)$ and $\Pr(E_2)$. Within the range of E_1, the variables X_3, X_4 is locally independent of the highly correlated variables X_1, X_2, so we have $\Pr(E_1) = \Pr(X_1, X_2) \cdot \Pr(X_3, X_4)$. The joint PDF of highly correlated variables X_1, X_2 is modeled together by the multivariate leaf node L_5, so we get the probability $\Pr(X_1 \in [1, 7], X_2 \in [0, 20]) = 0.3$ from L_5. The joint PDF of variables X_3, X_4 is modeled by the FSPN rooted at node N_2, the left child of N_1. We can obtain the probability in a similar way of SPN. Specifically, N_2 is a sum node, so we have $\Pr_{\mathrm{N}_2}(X_3 \in [3, 5], X_4 \in [0, 20]) = 0.3 \cdot \Pr_{\mathrm{N}_4}(X_3 \in [3, 5], X_4 \in [0, 20]) + 0.7 \cdot \Pr_{\mathrm{N}_5}(X_3 \in [3, 5], X_4 \in [0, 20])$. N_4 is a product node, so we have $\Pr_{\mathrm{N}_4}(X_3 \in [3, 5], X_4 \in [0, 20]) = \Pr_{\mathrm{L}_1}(X_3 \in [3, 5]) \cdot \Pr_{\mathrm{L}_2}(X_4 \in [0, 20])$. The probability that $\Pr_{\mathrm{L}_1}(X_3 \in [3, 5]) = 0.1$ and $\Pr_{\mathrm{L}_2}(X_4 \in [0, 20]) = 1$ could be obtained from the univariate leaf nodes L_1 and L_2, respectively. Hence, we get $\Pr_{\mathrm{N}_4}(X_3 \in [3, 5], X_4 \in [0, 20]) = 0.1$. The probability $\Pr_{\mathrm{N}_5}(X_3 \in [3, 5], X_4 \in [0, 20]) = 0.2$ could be obtained from leaf nodes L_3 and L_4 in the same way. As a result, we have $\Pr_{\mathrm{N}_2}(X_3 \in [3, 5], X_4 \in [0, 20]) = 0.3 * 0.1 + 0.7 * 0.2 = 0.17$ and $\Pr(E_1) = 0.3 * 0.17 = 0.051$.

Third, the probability of E_2 could be computed in the same way as E_1. For E_2, the probability $\Pr(X_1 \in [1, 7], X_2 \in [0, 20]) = 0.4$ is obtained from the multivariate leaf node L_6, and the probability $\Pr_{\mathrm{N}_2}(X_3 \in (5, 6], X_4 \in [0, 20]) = 0.3$ is also computed by the FSPN rooted at node N_2. We have $\Pr(E_2) = 0.4 * 0.3 = 0.12$.

Finally, we obtain the probability of E as $\Pr(E) = \Pr(E_1) + \Pr(E_2) = 0.171$.

C Details of the Learn-FSPN Algorithm

In our implementation of the Learn-FSPN algorithm, we use the RDC score (Lopez-Paz et al., 2013) as the independence test oracle since it can capture dependencies between variables of hybrid domains. Two variables are identified to be independent and highly correlated if their RDC score is lower than a threshold τ_{L} or larger than a threshold τ_{H}, respectively. In our experiment, we set $\tau_{\mathrm{L}} = 0.3$ and $\tau_{\mathrm{H}} = 0.7$, respectively.

For the clustering method in line 17 for sum nodes, we use k-means, an EM method. For the partition method in line 26 for split nodes, we design two methods as follows:

```
Algorithm LEARN-FSPN($D, X, C$)
1: if $C = \emptyset$ then
2:     test the correlations $c_{ij}$ for each pair of attributes $X_i, X_j \in X$
3:     $H \leftarrow \{X_i, X_j | c_{ij} \geq \tau\}$
4:     recursively enlarge $H \leftarrow H \cup \{X_k | c_{ik} \geq \tau, X_i \in H, X_k \in X \setminus H\}$
5:     if $H \neq \emptyset$ then
6:         set $N$ to be a factorize node
7:         call LEARN-FSPN($D, X \setminus H, \emptyset$)
8:         call LEARN-FSPN($D, H, X \setminus H$)
9:     else if $|X| = 1$ then
10:        set $N$ to be a uni-leaf node
11:        model the univariate distribution $\Pr_{\mathrm{D}}(X)$
12:    else if subsets $S_1, S_2, \ldots, S_k$ are mutually indepedent by the independence oracle then
13:        set $N$ to be a product node
14:        call LEARN-FSPN($D, S_i, \emptyset$) for each $1 \leq i \leq k$
15:    else
16:        set $N$ to be a sum node
17:        let $D_1, D_2, \ldots, D_t$ be the generated cluster of data with weights $w_1, w_2, \ldots, w_t$
18:        $w_i \leftarrow |T_i|/|T_{\mathrm{N}}|$ for all $1 \leq i \leq n$
19:        call LEARN-FSPN($D_i, X, \emptyset$) for each $1 \leq i \leq t$
20: else
21:    if $X$ is independent of $C$ on $D$ by the independence oracle then
22:        set $N$ to be a multi-leaf node
23:        model the multivariate distribution $\Pr_D(X)$
24:    else
25:        set $N$ to be a split node
26:        let $D'_1, D'_2, \ldots, D'_t$ be the generated partition of data
27:        call LEARN-FSPN($D'_i, X, C$) for each $1 \leq i \leq t$
```

• Grid approximation of k-means: At first, we use the k-means method to cluster all data into two clusters. By the properties of k-means, each clustering forms a hyper-spheroid in the space. Let c_1 and c_2 be the central points and r_1 and r_2 by the radius of the two clusters, respectively. Certainly, on the straight line across c_1 and c_2 in the space, there must exist two boundary points b_1 and b_2 of the two clusters. Let b be the mid-point of b_1 and b_2, we can split all data into two parts by one dimension of b's value. Some data points x in one cluster would be divided into the other part. Hence, we choose the dimension of b's value with minimal $|x|$ as the splitting point.

• Greedy splitting: Let $c \in C$ be the variable that maximizes the pairwise correlations between variables in $X \setminus C$ and C. Intuitively, dividing the space by c would largely break the correlations between $X \setminus C$ and C. Then, we randomly choose d values $c_1, c_2, \ldots, c_d$ in the range of variable c. For each value c_i, we could divide all data into two parts. We compute the pairwise correlations between variables in $X \setminus C$ and C in each part. The value c_i minimizes this value is chosen as the splitting point.

Notice that, LEARN-FSPN is a framework support different independent test oracles, clustering and partition algorithms. The problem to design best plug-ins of these methods for a specific application is still open. In our experiments, we have also tried some other kinds of independent test and clustering methods. We find that using RDC scores, k-means clustering and greedy splitting methods attains the best performance on our datasets.

D ADDITIONAL EXPERIMENTAL RESULTS

We present additional experimental results on the model size and training time of PGMs. Figure 4 shows the model size and training time of each PGM on the synthetic datasets. Table 2 gives the detailed number of nodes in each model.

In terms of the model and number of nodes, we observe that:

• The model and number of nodes of FSPNs are consistently much smaller than SPN-BTBs and MSPNs by up to two orders of magnitude. In particular, the FSPNs' model size is up to 27×

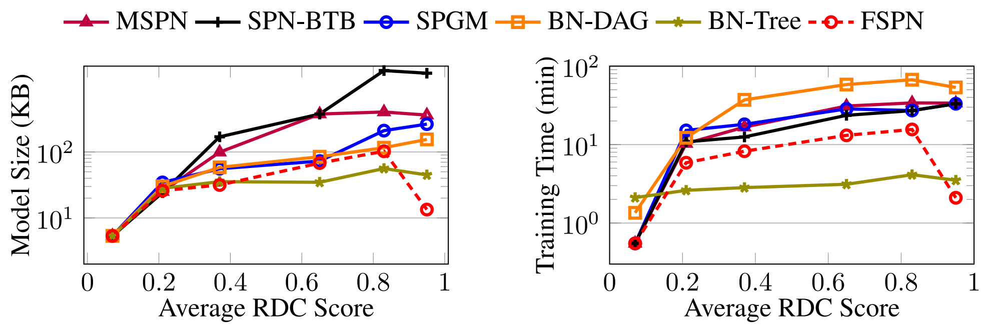


Figure 4: Model size and training time on synthetic data.

Table 2: Number of nodes on synthetic data.

RDC Score of Dataset	# of nodes in MSPN	# of nodes in SPN-BTB	# of nodes in SPGM	# of nodes in BN-DAG/Tree	# of nodes in FSPN
0.07	21	21	21	20	21
0.21	765	562	321	20	168
0.37	1, 482	1, 175	472	20	375
0.65	3, 520	1, 626	481	20	437
0.83	3, 847	2, 753	718	20	661
0.95	4, 173	2, 398	563	20	33

and 116× smaller than MSPNs and SPN-BTBs, respectively. The FSPNs' number of nodes is up to 126× and 73× smaller than MSPNs and SPN-BTBs, respectively. This is because tree-structured SPNs may generate a large number of nodes in presence of highly correlated variables. SPN-BTBs have lots of leaf nodes, and for each leaf node, they create a Chow-Liu tree. As a result, the space cost of SPN-BTBs is the highest among all models.

• The model size of FSPNs is also smaller than SPGMs and BN-DAGs. This is because they require to store the CPTs over multiple variables, while FSPNs store the lightweight univariate distributions and multivariate distributions on only highly correlated variables. In terms of the number of nodes, FSPNs are also much smaller than SPGMs by up to 17×. The number of nodes in BN-DAGs and BN-Trees always equals to the variable number so it is not informative to make a comparison.

• The model size of BN-Tree is the smallest since the tree-width is only one for the learned Chow-Liu tree.

In terms of the training time, we find that FSPNs are several times faster to train than other models, except BN-Trees. Whereas for other models (BN-DAGs, SPN-BTBs, SPGMs) related to BNs, the structure learning process is time costly. MSPNs also require a longer training time since the structure learning algorithm repeatedly splits nodes in presence of highly correlated variables.

In summary, these results verify our claims in Section 2. The results show that the design choices underlying FSPNs, i.e. separating highly correlated variables from others and modeling them adaptively may represent the joint PDF in a more compact form and learn it efficiently.