# Explaining Deep Tractable Probabilistic Models: The sum-product network case

**Athresh Karanam** athresh.karanam@utdallas.edu
**Saurabh Mathur** saurabhsanjay.mathur@utdallas.edu
**Predrag Radivojac** predrag@northeastern.edu
**David M. Haas** dahaas@iu.edu
**Kristian Kersting** kersting@cs.tu-darmstadt.de
**Sriraam Natarajan** sriraam.natarajan@utdallas.edu

## Abstract

We consider the problem of explaining a class of tractable deep probabilistic models, the Sum-Product Networks (SPNs) and present an algorithm $\mathcal{EXSPN}$ to generate explanations. To this effect, we define the notion of a context-specific independence tree(CSI-tree) and present an iterative algorithm that converts an SPN to a CSI-tree. The resulting CSI-tree is both interpretable and explainable to the domain expert. We achieve this by extracting the conditional independencies encoded by the SPN and approximating the local context specified by the structure of the SPN. Our extensive empirical evaluations on synthetic, standard, and real-world clinical data sets demonstrate that the CSI-tree exhibits superior explainability.

## 1. Introduction

**Tractable Deep Probabilistic Models** (TDPMs) exploit the efficiency of deep learning (Goodfellow et al. (2016)) while abstracting the representation of the underlying model. TDPMs implement **compositions of functions**, which increases their representation power considerably compared to deep learning. Specifically, they abstract the underlying representation by implementing a composition of probability distributions over domain features, which can be discrete, continuous, graphical, or even unstructured. Unsurprisingly, these benefits have led to considerable interest in TDPMs (Choi and Darwiche (2017); Poon and Domingos (2011); Cutajar et al. (2017)). Some TDPMs such as Arithmetic Circuits and Sum-product Networks (SPNs) explicitly model the joint distribution using a **network polynomial** over evidence indicators and network parameters. This makes them closely related to polynomial neural networks (PNNs, Nikolaev and Iba (2006)), which are a class of power-series function models with multiplicative activation functions and parsimonious structure.

We consider the specific formulation of SPNs and pose the following question – *can SPNs with their multiple layers be explained using existing tools inside probabilistic modeling?* To achieve this, we move beyond the traditional notions of conditional independencies that can be read off an SPN and instead focus on context-specific independencies (CSI, Boutilier et al. (1996)). CSIs provide a more in-depth look at the relationships between two variables when affecting the third variable. For instance, in a gestational diabetes prediction task (Karanam

et al. (2021)), one could state that gestational diabetes and education are conditionally independent given the age. This allows the care provider/physician to develop a good interventional treatment plan. Recent work on developing interventions given a learned SPN (Zecevic et al. (2021)) demonstrates the potential of such TDPMs and our work goes in the same direction by identifying CSIs that could potentially aid the expert in identifying appropriate interventions.

Specifically, we define the notion of a CSI-tree that is used as a *visual tool to explain SPNs*. We present an algorithm ($\mathcal{EXSPN}$ ) that grows a CSI-tree iteratively. We show clearly that the constructed CSI-tree can recover the full SPN structure. Once a tree is constructed, we then approximate the CSIs by learning a supervised model to fit the CSIs. The resulting feature importances can then be used to further approximate the tree. Our evaluations against association rule mining clearly demonstrate that the recovered CSI-trees indeed are shorter and more interpretable.

We make the following key contributions: (1) We develop CSI-trees that focus on the explanation. These trees are built on the earlier successes inside graphical models and we use them in the context of TDPMs. (2) We develop an iterative procedure that constructs a CSI-tree given an SPN. The resulting tree is a complete representation of the original SPN. We present an approximation heuristic that compresses these trees further to enhance the explainability of the model. One key aspect of $\mathcal{EXSPN}$ is that it is **independent** of the underlying SPN learning algorithm. Any SPN that is complete and consistent (as defined in the next section) can be used as input for $\mathcal{EXSPN}$ . [1] (3) We perform extensive experiments on synthetic, multiple standard data sets and a real clinical data set. Our evaluation demonstrates that the final model induces smaller rules/models compared to the original SPN.

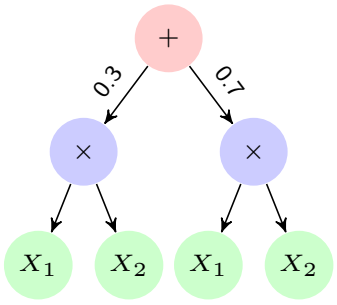


Figure 1: SPN that represents the joint distribution $P(X_1, X_2)$. Sum nodes are inscribed with "+", product nodes with "×" and leaf nodes with corresponding variable names(*best viewed in color).*

## 2. Background and Preliminaries

**Sum-product networks:** SPNs(Poon and Domingos (2011)) are weighted directed acyclic graphs(DAGs) with sum and product nodes as the internal nodes of the graph and probability distributions as the leaf nodes. They represent joint distributions over a set of variables $\mathbf{X}$. Let $\mathbf{X} = \{X_1, X_2, ..., X_m\}$ be the set of $m$ random variables for which $|\mathcal{D}|$ data samples are available. We use the following definitions from Zhao et al. (2015). A **computational graph** $G$ is a DAG with three node types: **sums**, **products** and **leaves**. $N$ denotes a generic node and $\mathbf{N}$ denotes a set of nodes in $G$. Similarly, $e$ denotes an edge and $\mathbf{E}$ denotes a set of edges in $G$. A **scope function** $\psi$: $\mathbf{N} \rightarrow 2^{\mathbf{X}}$ is a mapping between a node $N$ and a subset of $\mathbf{X}$. It confines the set of RVs that a node is defined over. An **SPN** $S$ is a 4-tuple $(G, \psi, \mathbf{w}, \theta)$, where $G$ is a computational graph, $\psi$ is a *scope function*, $\mathbf{w}$ is a set of sum-weights and $\theta$ is a set of parameters of the leaf distributions. $ch(N)$ denote the children of a node $N$ and $pa(N)$, its parents.

1. Code and appendix is available at `anonymous.4open.science/r/ExSPN-16FD`

**SPN properties**: An SPN is *complete* iff each sum node has children with the same scope. An SPN is *consistent* iff no variable appears negated in one child of a product node and non-negated in another. An SPN is *decomposable* iff for every product node $N$, the scope of its children are disjoint. An SPN is said to be **normal** if (1) It is complete and decomposable, (2) For each sum node, the weights of the edges emanating from the sum node are nonnegative and sum to 1, and (3) Every terminal node in the SPN is a univariate distribution over a Boolean variable and the size of the scope of a sum node is at least 2. An **instance function** $\phi_{\mathcal{D}} : N \rightarrow 2^{|\mathcal{D}|}$ is a mapping between a node $N$ and a subset of $\mathcal{D}$. For notational simplicity, we use $\phi$ instead of $\phi_{\mathcal{D}}$ when $\mathcal{D}$ is implied.
**Example:** Figure 1 shows an SPN defined over two random variables $X_1, X_2$. The nodes of the graph with "+" within them are sum nodes, those with "×" within them are product nodes and the rest of the nodes are leaf nodes. The variable name within each leaf node indicates that a univariate distribution over that variable is learnt at that node. The set of sum, product and leaf nodes, and the edges connecting them constitute the SPN's computational graph. The scope of the sum node at the root and the two product nodes is $X_1, X_2$. The scope of the leaf nodes from left to right is $X_1$, $X_2$, $X_1$ and $X_2$, respectively. The labels on the edges from sum node to product nodes are the sum-weights.
**A note on interpretability**: There is no unique definition of interpretability (Lipton (2018),Doshi-Velez and Kim (2017)). In addition, it is often used interchangeably with explanability, with some work making a distinction (Montavon et al. (2017), Rudin (2019)). In this work, by interpretable models, we mean representations whose random variables, dependencies(structure) and parameters are interpretable by humans (Towell and Shavlik (1991), Montavon et al. (2017)). CSI-trees do not introduce any latent variables, unlike SPNs(Peharz et al. (2017)), and their parameters are logical statements comprising observed variables, thus satisfying the criterion for interpretability.

**Learning SPNs:** Several learning algorithms(Gens and Domingos (2013), Molina et al. (2018)) have been proposed for SPNs. For brevity, we will limit our discussion to simultaneous parameter and structure learning for tree-structured SPNs using the popular learnSPN (Gens and Domingos (2013)) framework. Each recursive step of their algorithm either learns parameters of a leaf distribution, a sum node by splitting the data instances into subsets, or a product node by decomposition of the variables into subsets of mutually independent variables. In the base case, when conditions for learning the leaf distributions are satisfied, a univariate leaf distribution is learnt and the recursion ends. If the variables can be partitioned into mutually independent subsets $\mathbf{X}_i \subseteq \mathbf{X}$, the algorithm learns a product node and recurses over each subset. Otherwise, the data is partitioned into subsets $\mathcal{D}_j \subset \mathcal{D}$, the algorithm learns a sum node and recurses over each subset. The sum and product nodes of SPNs learnt using learnSPN are latent variables whereas the leaf nodes learn distributions over observed variables $\mathbf{X}$.

## 3. $\mathcal{EXSPN}$ - Explaning SPNs

Before we outline our procedure for explaining SPNs, we briefly explain the notion of Context-specific independence(CSI) (Boutilier et al. (1996)). CSI is a generalization of the concept of statistical independence of random variables. CSI-relations have been studied extensively over the last three decades (Boutilier et al. (1996); Nyman et al. (2014); Tikka et al. (2019); Nyman et al. (2016)). They can be used to speed-up probabilistic

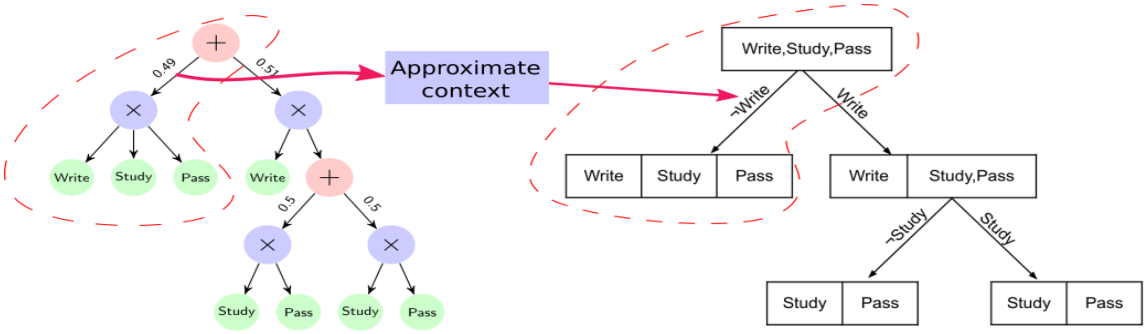


Figure 2: A normal SPN (left) and its corresponding CSI-Tree (right). The independencies induced by the product node are represented in the partitioned leaf node of the CSI-tree. The context in which these independencies hold is approximated to $\neg Write$.

inference, improve structure learning and explain graphical models that are learnt from data. We propose **a novel algorithm to extract these CSI-relations from SPNs** and empirically demonstrate that the extracted CSI-relations are interpretable. CSIs can be directly extracted from the data by computing conditional probabilities (Shen et al. (2020)) with context using parameterized functions such as neural networks(Kingma and Welling (2013)). While the literature on rule learning is vast (Fürnkranz and Kliegr (2015)), we note that our objective is *explaining SPNs* and not *rule learning*. Additionally, we note that the relationship between SPNs and BNs (Zhao et al. (2015)), and that between SPNs and multi-layer perceptrons (Vergari et al. (2019)) is well established. However, unlike our work, these methods do not explicitly attempt to explain SPNs through a compact representation of CSIs. The algorithm to convert SPNs to BNs presented by Zhao et. al.(Zhao et al. (2015)) introduces unobserved hidden variables into the BN. It generates a BN with a directed bipartite structure with a layer of hidden variables pointing to a layer of observable variables. In this case, the BN associates each sum node in the SPN to a hidden variable in BN. We argue that the introduction of these hidden variables renders the resulting BN uninterpretable.

To construct meaningful explanations, we define a compact and interpretable representation of CSIs called **CSI-tree**.

**Definition 3.1** *(CSI-tree): A CSI-tree $\tau$ is a 4-tuple $(G, \psi_{csi}, \chi, \zeta)$ where $G$ is a tree with a set of variables $V \subseteq \mathbf{X}$,* scope function $\psi_{csi}$, partition function $\chi$ *and edge labels $\zeta$.*

**Definition 3.2** *(Partition function): A partition function $\chi_\psi : N \rightarrow 2^{|\psi(N)|}$ is a mapping from a node to a set $C$ of disjoint sets $P_i \subseteq \psi(N)$ under a given scope function $\psi$ such that $\cup_{i \in |C|} P_i = \psi(N)$.*

Each node, $N$, of a CSI-tree has a scope $\psi_{csi}(N) \subseteq \mathbf{X}$. The partition function divides the variables within the scope of each node into disjoint subsets such that the union of these subsets is $\psi_{csi}(N)$. The edges are labeled with a conjunction $\zeta$ over a subset of $\mathbf{X}$. An example of an edge label is $(X_i \geq 0.5 \wedge X_j \leq 1)$. Essentially, the edge label narrows the scope of the context from parent to child node.

**Example CSI-tree:** The right side of the Figure 2 shows a CSI-tree defined over the binary variables $\langle Write, Study, Pass \rangle$. The set of variables inscribed within each node represent the scope of that node. For example, the scope of the root node is $\langle Write, Study, Pass \rangle$.

The edge label *confines the context* by conditioning on a set of variables (on a singleton set in this example). The variables in the scope of a node that are conditionally independent when conditioned on the context are separated into subsets (denoted by a vertical bar). It is the output of the partition function for that node. For example, the right child of the root node specifies a partition $\langle\langle Write\rangle, \langle Study, Pass\rangle\rangle$ of the scope of that node.

The left child of the root node induces a CSI-relation $Write \perp\!\!\!\perp Study \perp\!\!\!\perp Pass|\neg Write$. Similarly, the right child of the root node induces two CSI-relations $Write \perp\!\!\!\perp Study|Write$ and $Write \perp\!\!\!\perp Pass|Write$. To summarize, the subsets of variables within the scope of a node, as defined by the partition function, are independent of each other when conditioned on the proposition specified in the label of the edge connecting that node to its parent. Note that *reading this CSI-tree is significantly easier than a SPN* on the left, due to the internal nodes entirely comprising of observed variables, and thus allows for a more explainable and interpretable representation. Now, we formally define our goal:

**Given:** SPN $S = (G, \psi, w, \theta)$, data $\mathcal{D}$
**To Do:** Extract CSI-tree $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$

The left side of Figure 2 shows an SPN learnt over the variables $Write, Study, Pass$. The root node $N_0$ is a sum node, implying that $Write, Study, Pass$ are not independent of each other in the context of the entire dataset. Now, consider the left child of the root node $N_1$, which is a product node with three leaf nodes as its children. It implies that $Write \perp\!\!\!\perp Study \perp\!\!\!\perp Pass|\phi(N_1)$. In other words, the $Write, Study, Pass$ are **independent of each other in the context of** $\phi(N_1)$. While the context of $\phi(N_1)$ accurately explains the conditional independence induced by the product node, it requires $2^{|\psi(N_1)|}$ parameters to fully parameterize the context which renders the CSI specified under $\phi(N_1)$ uninterpretable. Therefore, we need to approximate this context in order to ensure interpretability of these CSI-relations. Additionally, while the instantiation of a subset of $\psi(N_1)$ that is consistent with $\phi(N_1)$ can be used to define the context for a particular CSI, this method would be highly sensitive to noise.

To address these issues, we propose to **first learn a discriminative model** $f$ in the supervised learning paradigm to predict if a data point belongs to $\phi(N_1)$ and use the notion of **feature importance to approximate the context**. We identify a set $I \subseteq \psi(N_1)$ of the most important features as determined by the feature importance scores for $\psi(N_1)$ w.r.t $f$ and approximate the context to a proposition of the form $\wedge_{i \in I}$. For this example, we choose $f$ to be a *decision tree* and the *mean decrease in impurity* as the measure of feature importance and obtain $(\neg Write)$ as the approximated context. So, the original CSI $Write \perp\!\!\!\perp Study \perp\!\!\!\perp Pass|\phi(N_1)$ is approximated to $Write \perp\!\!\!\perp Study \perp\!\!\!\perp Pass|\neg Write$.

### 3.1 $\mathcal{EXSPN}$ Algorithm

For simplicity, we present our approach with binary variables. However, in our experiments, we demonstrate that our method **can be applied to continuous and multi-class discrete variables** as well. We now outline our approach to infer CSI-trees from SPNs. The steps involved in converting an SPN into a CSI-tree are: (1) Convert $S$ into a normal-SPN $S_{normal}$. (2) Infer the instance function $\phi$. (3) Create a CSI-tree $\tau_{unlabeled}$ with no edge labels using $\phi$. (4) Compute edge labels for $\tau_{unlabeled}$, and create CSI-tree $\tau$. (5) Optionally, compress $\tau$ into $\tau_{compressed}$ by pruning $\tau$.

Algorithm 1 presents $\mathcal{EXSPN}$ : **Ex**plaining **S**um-**P**roduct **N**etworks. It infers a CSI-tree $\tau$, given an SPN $S$, data $\mathcal{D}$, set of random variables $\mathbf{X}$ and feature importance score threshold $\lambda$.

**Algorithm 1:** $\mathcal{EXSPN}$

**input :** $\mathcal{D}$, $\mathbf{X}$, $S = (G = (\mathbf{N}, \mathbf{E}), \psi, w, \theta)$, $\lambda$
**output:** CSI-tree $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$

1 Convert $S$ to a normal-spn $S_{normal}$
2 Infer $\phi$ using alg. 2
3 Initialize $G_{csi} = G_{normal}, \psi_{csi} = \psi_{normal}$,
4 $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$
5 st = [$G_{csi}$.root]
6 Add $\psi(G_{csi}.root)$ to $\chi(G_{csi}.root)$
7 **while** *st is not empty* **do**
8  $N_{current} = st.pop()$
9  **if** $N_{current}$ *is not the root node* **then**
10  **if** $N_{current}$ *is a leaf node* **then**
11  Add $\psi(N_{current})$ to $\chi(pa(N_{current}))$
12  **end**
13  **if** $N_{current}$ *is a sum node* **then**
14  Add $\psi(N_{current})$ to $\chi(pa(N_{current}))$
15  Connect $pa(N_{current})$ and each $ch(N_{current})$
16  Replicate sub-SPN rooted at $ch(N_{current})$ for each $|pa(ch(N_{current}))|$ to ensure $G_{csi}$ has a tree structure
17  Delete $N_{current}$ from $G_{csi}$
18  **end**
19  **end**
20  Add $ch(N_{current})$ to $st$
21 **end**
22 $\tau = ComputeLabels(\tau, \mathcal{D}, \phi, \lambda)$
23 **return** $\tau$

(**Step 1:**) Convert $S$ to a normal-spn $S_{normal} = (G_{normal}, \psi_{normal}, w_{normal}, \theta_{normal})$ **[line 1]** using the conversion scheme proposed by Zhao et. al. (Zhao et al. (2015)). Any arbitrary SPN $S$ can be converted into a normal SPN $S_{normal}$ that represents the same joint probability over variables and $|S_{normal}| = \mathcal{O}(|S|^2)$.
(**Step 2:**) It then infers the instance function for $S_{normal}$**[line 2]** using algorithm 2. This algorithm is similar to the algorithm proposed in Poon and Domingos (2011) for approximating most probable explanation(MPE) inference in arbitrary SPNs. For each instance $\mathcal{D}_j \in \mathcal{D}$ it first performs an upward pass from leaf nodes to the root node and computes $S_i^{max}(\mathcal{D}_j)$ for each node $N_i$ as follows:

- if $N_i$ is a sum node, then, $S_i^{max}(\mathcal{D}_j) = \max_{k \in ch(N_i)} w_{ik}.S_j^{max}(\mathcal{D}_j)$
- otherwise $S_i^{max}(\mathcal{D}_j) = S_i(\mathcal{D}_j)$

Then the algorithm backtracks from the root to the leaves, appending $\mathcal{D}_j$ to the instance function associated with each child of a sum node that led to $S_i^{max}(\mathcal{D}_j)$ and to the instance function associated with all product nodes along the path from the root to a leaf node.
(**Step 3:**) Next, it performs a depth-first search(DFS) on the computational graph $G_{normal}$ associated with $S_{normal}$ and constructs the tree-structured graph $G_{csi}$ associated with $\tau$ **[lines 4-19]**. $G_{csi}$ is initialized with $G$ and the scope of root of $G_{csi}$, $\psi(G_{csi}.root)$ is added to the value of the partition function associated with $\tau$**[line 5]**. $\mathcal{EXSPN}$ maintains a stack $st$ that is initialized with the root of $G_{csi}$**[line 4]**. The current node selected in DFS, $N_{current}$, is popped from the stack $st$**[line 7]**. It then considers three cases: 1. $N_{current}$ is a leaf node 2. $N_{current}$ is a sum node 3. $N_{current}$ is a product node**[lines 9-16]**. If $N_{current}$ is a leaf node, its scope $\psi(N_{current})$ is added to the partition function of its parent node $\chi(pa(N_{current}))$**[lines 9-10]**. If $N_{current}$ is a sum node, its scope $\psi(N_{current})$ is added to the partition function of its parent node $\chi(pa(N_{current}))$, edges are added to $G_{csi}$ to connect the parent of $N_{current}$, $pa(N_{current})$, to

each child of $N_{current}$, and deleting $N_{current}$ and all edges $e$ in $G_{csi}$ of the form $e(\alpha, N_{current})$ or $e(N_{current}, \alpha)$[**line 12-16**]. If $N_{current}$ is a product node, it is ignored. It then continues DFS over $G_{csi}$ by adding all the children of $N_{current}$ to $st$[**line 19**]. Note that the CSI-tree $\tau$ is unlabeled.

**Algorithm 2:** Infer Instance Function

```
input : D, X, normal SPN S
output: Instance function φ
for Each instance D_j ∈ D do
    Perform upward pass for D
    Compute S_N^max(D_j) for each node N
    Perform a downward pass
    Append D_j to the φ(N_ch) of a child
      N_ch of a sum node that led to
      S^max(D_j)
    Append D_j to φ(N_product) for all
      product nodes N_product along the
      path from the root to a leaf node
end
return φ
```

(**Step 4:**) Algorithm 3 presents the procedure to train a model $f$, compute a set of important features $I$, and finally compute the edge labels $\zeta$. For each edge $e(N_{from}, N_{to})$, it first computes class labels $y$ to indicate if an instance $\mathcal{D}_j \in \phi(N_{from})$ belongs to $\phi(N_{to})$. Then it computes a set of important features $I$ for $f$ using a suitable feature importance computation technique based on the choice of $f$ and a threshold for feature importance score $\lambda$. The edge label for $e$ is the conjunction of the features in $I$.

(**Step 5:**) The labeled CSI-tree $\tau$ created in the previous step might be too large in some cases and the CSIs induced by $\tau$ at a product node $N$ may be supported by a small number of examples given by $|\phi(N)|$. To avoid these two issues, we propose deleting the sub-tree induced by a product node $N$ for which $|\phi(N)| < min_{instances}$. This heuristic significantly reduces the size of the CSI-tree while retaining CSIs induced by product nodes closer to the root node of the SPN, as demonstrated in our experiments.

**Algorithm 3:** ComputeLabels

```
input : τ = (G, ψ, χ, NULL), D, φ, λ
output: Labeled CSI-tree
        τ = (G, ψ, χ, ζ)
for Each edge e(N_from, N_to) in G do
    Compute class labels y
    f = TrainModel(D, φ(N_from), y)
    Compute a set of important
      features I for f
    ζ(e) = ∧_{i∈I} i thresholded by λ
end
return τ = (G, ψ, χ, ζ)
```

**Computational Complexity**: The computational complexity of $\mathcal{EXSPN}$ algorithm depends on the complexity of each of its constituent 5 steps. The size $(\#nodes + \#edges)$ of the normal-SPN $S'$ obtained from the original SPN $S$ has a space-complexity of $O(|S|^2)$. This conversion can be done is time linear in the size of the normal-SPN (Zhao et al. (2015)). The generation of instance function for the CSI-tree involves MAX inference for each of $\mathcal{D}$ data points which can be performed in $O(|\mathcal{D}||S'|)$ (Poon and Domingos (2011)). The generation of unlabeled CSI-tree has a time-complexity of $O(|S'|)$. Generating edge labels involves training a discriminative model which can be performed in $O(|\mathcal{D}||\theta_D|)$ for a universal approximator such as neural network with parameters $\theta_D$. Here $|N_{sum}|$ is the number of sum nodes in the network and $|\mathbf{X}|$ is the number of variables. Finally the compression can be performed in time linear in the size of the network. This gives us an *overall time complexity of* $O(max(|\theta|, |X|)|S|^2|\mathcal{D}|)$.

**Properties of CSI-tree:** $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$, corresponding to an SPN $S, \psi, w, \theta)$ obtained through $\mathcal{EXSPN}$ has the following properties: (1) Num nodes in $G_{csi}$ = num product nodes in $G_{normal} + 1$. (2) The context induced by a product node $N_p$ in $S$ requires $2^{|\psi(N_p)|}$ parameters to be sufficiently expressed, while the approximate context from $\mathcal{EXSPN}$ has

$< |\psi(N_p)|$ parameters. Additionally, the structure of a tree-structured normal-SPN can be retrieved from its corresponding CSI-tree in time linear in the size of the SPN.

**Theorem 1** *The CSI-tree $\tau$, inferred from an SPN, $S_{normal}$, using $\mathcal{EXSPN}$ can infer $G'_{normal}$, and $\psi'_{normal}$ which encodes the same CSIs as $S_{normal}$.*

We present the proof in Appendix A.2.

## 4. Experimental Evaluation

We explicitly answer the following questions: (**Q1: Correctness**) Does $\mathcal{EXSPN}$ recover all the CSIs encoded in an SPN? (**Q2: Compression**) Can the CSIs be compressed further? (**Q3. Baseline**) How do the CSIs extracted using $\mathcal{EXSPN}$ compare with a strong rule learner? (**Q4. Real data**) Does $\mathcal{EXSPN}$ extract reasonable CSIs in a real clinical study? **System:** We implemented $\mathcal{EXSPN}$ using SPFlow library (Molina et al. (2019)). We assume that the instance function is computed during the training process. For experiments where the instance function is computed separately after training, see Table 2. Since Decision Trees can be represented as a set of decision rules, we used the Classification and Regression Trees (CART, Breiman et al. (1984)) algorithm as the explainable function approximator. We used scikit-learn's DecisionTreeClassifier (Pedregosa (2011)) to implement CART. The hyperparameter configuration of these algorithms is shown in Table 5 in the appendix.

**Baseline:** To evaluate the CSIs extracted by $\mathcal{EXSPN}$ , we compared them with the association rules mined using the Apriori algorithm (Agrawal et al. (1994)). We used the Mlxtend library (Raschka (2018)) to implement this baseline. Since the Apriori algorithm requires binary features, we discretized the continuous variables in the datasets into 5 categories and one-hot encoded the categorical variables.

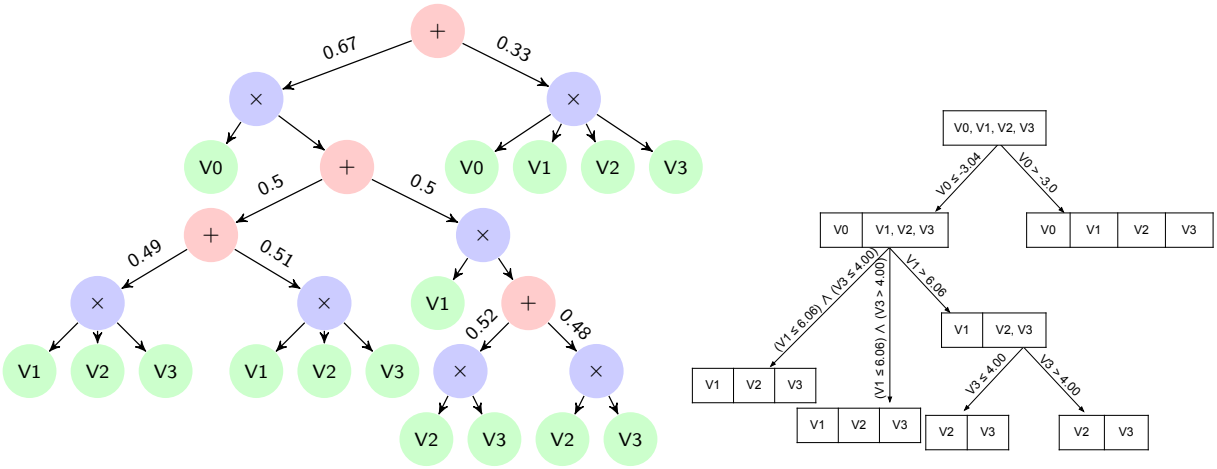


Figure 3: The sum-product network (left) trained on the synthetic data set and its corresponding CSI-tree extracted by $\mathcal{EXSPN}$ (right). Each edge in the CSI-tree corresponds to an edge from a sum node to a product node in the SPN. All other edges are collapsed. Clearly, the CSI-tree encodes all the CSIs represented in the sum-product network.

Table 1: Summary statistics for the CSI rules extracted from SPNs by $\mathcal{EXSPN}$ and the association rules by Apriori algorithm. NP - # Product Nodes, NR - # Rules, MA - Mean Antecedent length, MC - the Mean Consequent length, and CR - Compression Ratio.

| | SPN | All CSIs | | | Reduced CSIs | | | | Association Rules | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | NP | NR | MA | MC | NR | MA | MC | CR | NR | MA | MC |
| Synthetic | 7 | 7 | 2.29 | 2.57 | **3** | 1.33 | 2.67 | 2.33 | 12 | 1.25 | 1.25 |
| Mushroom | 39 | 39 | 5.90 | 8.54 | **14** | 4.79 | 7.93 | 2.79 | 10704 | 2.87 | 2.43 |
| Plants | 342 | 342 | 9.60 | 9.61 | **23** | 6.22 | 7.09 | 14.87 | 1043 | 1.72 | 1.40 |
| NLTCS | 74 | 74 | 9.84 | 3.32 | **19** | 6.32 | 4.05 | 3.89 | 165 | 1.96 | 1.28 |
| MSNBC | 8 | **8** | 4.12 | 5.75 | **8** | 4.12 | 5.75 | 1.00 | 16 | 2.38 | 1.00 |
| Abalone | 194 | 194 | 11.31 | 7.00 | **4** | 4.25 | 2.00 | 48.50 | 730 | 2.15 | 1.71 |
| Adult | 263 | 263 | 14.49 | 4.02 | **19** | 7.37 | 2.74 | 13.84 | 917 | 2.24 | 1.72 |
| Wine quality | 236 | 236 | 12.45 | 6.76 | **5** | 3.60 | 2.60 | 47.20 | 337 | 1.99 | 1.56 |
| Car | 18 | 18 | 5.22 | 2.50 | **14** | 5.21 | 2.64 | 1.29 | 19 | 1.58 | 1.00 |
| Yeast | 181 | 181 | 16.20 | 3.26 | **10** | 7.90 | 2.30 | 18.10 | 50 | 1.52 | 1.52 |
| nuMoM2b | 104 | 104 | 10.60 | 2.33 | **31** | 6.55 | 2.19 | 3.35 | 21 | 1.29 | 1.14 |

Table 2: Summary statistics for the case where the instance function is inferred after training. The columns are the same as Table 1.

| | SPN | All CSIs | | | Reduced CSIs | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | NP | NR | MA | MC | NR | MA | MC | CR |
| Synthetic | 7 | 7 | 2.29 | 2.57 | 3 | 1.33 | 2.67 | 2.33 |
| Mushroom | 39 | 39 | 5.90 | 8.54 | 13 | 5.08 | 8.38 | 3.00 |
| Plants | 342 | 342 | 8.33 | 9.61 | 32 | 4.62 | 6.72 | 10.69 |
| NLTCS | 74 | 74 | 9.74 | 3.32 | 19 | 6.32 | 4.05 | 3.89 |
| MSNBC | 8 | 8 | 4.12 | 5.75 | 8 | 4.12 | 5.75 | 1.00 |
| Abalone | 194 | 157 | 9.61 | 6.85 | 8 | 5.75 | 2.00 | 19.63 |
| Adult | 263 | 244 | 11.27 | 3.89 | 12 | 6.17 | 3.08 | 20.33 |
| Wine | 236 | 235 | 9.18 | 6.78 | 6 | 3.50 | 2.50 | 39.17 |
| Car | 18 | 18 | 5.22 | 2.50 | 14 | 5.21 | 2.64 | 1.29 |
| Yeast | 181 | 181 | 13.06 | 3.26 | 10 | 6.60 | 2.30 | 18.10 |
| nuMoM2b | 104 | 98 | 9.64 | 2.29 | 35 | 6.97 | 2.17 | 2.80 |

**Datasets:** We evaluated $\mathcal{EXSPN}$ on 11 datasets – one synthetic, 9 benchmark, and one **real clinical study**. We generated the synthetic dataset by sampling 10,000 instances each from 3 multivariate Gaussians. We used 8 datasets from the UCI repository and the National Long Term Care Survey (NLTCS, Lowd and Davis (2010)) data from CMU StatLib (`http://lib.stat.cmu.edu/datasets/`). The 8 datasets from the UCI machine learning repository were Mushroom, Plants, MSNBC, Abalone, Adult, Wine quality, Car, and Yeast. In MSNBC and Plants, we used the rows which had at least two items present. For Wine quality, we concatenated the red wine and white wine tables. In addition, we used **Nulliparous Pregnancy Outcomes Study: Monitoring Mothers-to-Be** (nuMoM2b, Haas et al. (2015)) study data. Our subset has 8 variables - *oDM, Age, Race, Education, BMI, Gravidity, Smoked3Months,and SmokedEver*. Of these, oDM is the boolean represent-

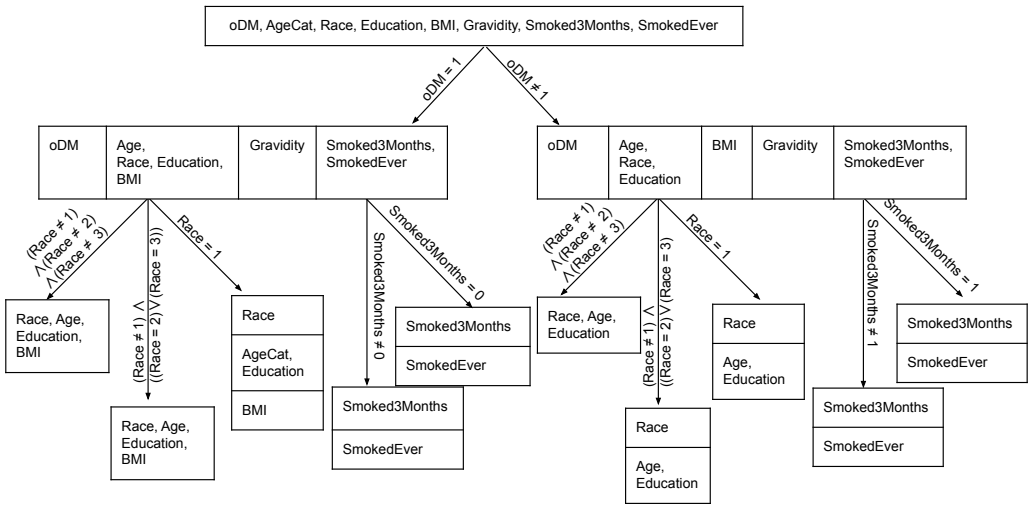


Figure 4: First two levels of the CSI-tree for the nuMoM2b dataset. Here, *oDM* is a boolean variable that represents whether or not the person has Gestational Diabetes. *Race* is a categorical variable having 8 categories namely, Non-Hispanic White (1), Non-Hispanic Black (2), Hispanic (3), American Indian (4), Asian (5), Native Hawaiian (6), Other (7), and Multiracial (8). *Smoked3Months* and *SmokedEver* are boolean variables representing tobacco consumption.

ing Gestational Diabetes (0/1). We split each dataset into a train set having 75% of the examples and a test set having the remaining 25%. To ensure a balanced split, we stratified the splits on the target variable for the classification datasets. Table 4, in the appendix. summarizes the datasets.

**Metrics:** We defined the following metrics on the CSIs - min_precision ($mp$), min_recall ($mr$), and n_instances ($ni$). $mp$ and $mr$ of a CSI are the minimum values of precision and recall respectively for each of the decision rules that approximate the context. $ni$ of a CSI is the number of training instances in that context. We used thresholds on these metrics to obtain a reduced set of CSIs (0.7 for $mp$ and $mr$, and $5 \times$ `min_instances_slice` for $ni$). We quantified this reduction as the *Compression Ratio (CR)*, which is the fraction of the total number of CSI rules in reduced set to the total number of rules. To compare association rules, we use mean antecedent length (mean $|A|$) and mean consequent length (mean $|C|$).

### 4.1 Results

(Q1: Correctness) Table 1 summarizes the SPNs, the full set of CSI rules extracted by $\mathcal{EXSPN}$ , and the reduced set of CSI rules. For each dataset in the table, the number of CSIs extracted by $\mathcal{EXSPN}$ (NR) **is exactly equal to** the number of product nodes of the SPN(NP). Figure 3 shows the SPN learnt from the synthetic data and the CSI-tree extracted from that SPN using $\mathcal{EXSPN}$ . Clearly, the CSI-tree recovers all the CSIs encoded in the synthetic data. We further quantify how well the combination of SPN and $\mathcal{EXSPN}$ approximates the ground truth CSIs using data samples from BNs. Since the data is sampled from a BN, the ground truth CSIs can be obtained by converting the conditional distributions $P(X_i|X \setminus X_i) \forall X_i \in X$ to *t*ree-*s*tructured *c*onditional *p*robability *d*istributions (Tree-CPDs). We use the ground truth CSIs to evaluate the CSIs extracted by $\mathcal{EXSPN}$

Table 3 summarizes the results of this evaluation. The ratios of the CSIs extracted by $\mathcal{EXSPN}$ to the ground truth is significantly high. Thus, Q1 is answered affirmatively.

(Q2: Compression) We can also infer from Table 1 that filtering the CSI rules using min_precision, min_recall and n_instances results in high compression ratios for **all but the MSNBC dataset**. This is because the MSNBC dataset already had 8 rules and all of the rules satisfied the threshold conditions. Hence, Q2 is answered strongly affirmatively.

Table 3: The Number of CSIs extracted by ExSPN from an SPN fit on each dataset (Total), The number of CSIs where at least 80% of the datapoints in the context matched with the CSIs from the Bayesian Network(BN) (Correct) and the Ratio of correct CSIs (Ratio).

| **Dataset** | Total | Correct | Ratio |
|---|---|---|---|
| Earthquake | 12 | 10 | 0.83 |
| Cancer | 12 | 10 | 0.83 |
| Asia | 25 | 22 | 0.88 |

(Q3. Baseline) Table 1 summarizes the association rules extracted from the data using the Apriori algorithm, and the mean confidence of the rules on the test set. Comparing the number of rules and the mean antecedent and consequent length from Table 1 allows us to answer Q3. We can infer that while the CSI rules extracted by $\mathcal{EXSPN}$ are longer than the rules extracted by the Apriori algorithm, the set of CSI rules is much smaller.

(Q4. Are the explanations correct?) Figure 4 shows the first two levels of the CSI-tree extracted by $\mathcal{EXSPN}$ from the nuMoM2b dataset. The first split of the CSI-tree is on the target variable $oDM$. While the $BMI$ variable is independent of other variables when $oDM \neq 1$, it is dependent on $Age, Race, Education$ for the case when $oDM = 1$. These **independencies are validated by our domain expert, Dr. David Haas**. This clearly demonstrates the potential of explaining a joint model such as SPN in a real clinically relevant domain.

## 5. Discussion and Conclusion

We considered the challenging problem of explaining SPNs by defining a CSI-tree that captures the CSIs that exist in the data. We presented an iterative procedure for inducing the CSI-tree from a learned SPN by approximating the context induced by a product node using supervised learning. We then presented an algorithm for recovering an SPN that encodes the same CSIs as the original SPN from the CSI-tree thus establishing the correctness of the conversion. Our experiments in synthetic, benchmarks and a real clinical study demonstrate the effectiveness of the approach by identifying the correct CSIs from the data. As far as we are aware, this is the first work on explaining joint distributions using the lens of CSI. Validating our method on more relevant clinical studies, allowing for domain experts to interact with our learned model, extending the algorithm to work on the broader class of distributions in general and TDPMs in particular, including more type of explanations, and finally, scaling the algorithm to large number of features remain interesting directions.

## References

R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *VLDB*, 1994.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *UAI*, 1996.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

A. Choi and A. Darwiche. On relaxing determinism in arithmetic circuits. In *ICML*, 2017.

K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep Gaussian processes. In *PMLR*, 2017.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017.

J. Fürnkranz and T. Kliegr. A brief overview of rule learning. Springer International Publishing, 2015.

R. Gens and P. M. Domingos. Learning the structure of sum-product networks. In *ICML*, 2013.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

D. M. Haas, C. B. Parker, et al. A description of the methods of the nulliparous pregnancy outcomes study: monitoring mothers-to-be (numom2b). *American journal of obstetrics and gynecology*, 2015.

A. Karanam, A. L. Hayes, H. Kokel, D. M. Haas, P. Radivojac, and S. Natarajan. A probabilistic approach to extract qualitative knowledge for early prediction of gestational diabetes. In *AIME*, 2021.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. 16(3):31–57, 2018. ISSN 1542-7730. doi: 10.1145/3236386.3241340.

D. Lowd and J. Davis. Learning markov network structure with decision trees. In *ICDM*, 2010.

A. Molina, A. Vergari, N. D. Mauro, S. Natarajan, F. Esposito, and K. Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *AAAI*, 2018.

A. Molina, A. Vergari, K. Stelzner, R. Peharz, P. Subramani, N. D. Mauro, P. Poupart, and K. Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

G. Montavon, W. Samek, and K. Müller. Methods for interpreting and understanding deep neural networks. *CoRR*, abs/1706.07979, 2017. URL `http://arxiv.org/abs/1706.07979`.

N. Nikolaev and H. Iba. *Adaptive Learning of Polynomial Networks: Genetic Programming, Backpropagation and Bayesian Methods*. Springer-Verlag, 2006. ISBN 0387312390.

H. Nyman, J. Pensar, T. Koski, and J. Corander. Stratified graphical models-context-specific independence in graphical models. *Bayesian Analysis*, 9(4):883–908, 2014.

H. Nyman, J. Pensar, T. Koski, and J. Corander. Context-specific independence in graphical log-linear models. *Computational Statistics*, 2016.

F. e. a. Pedregosa. Scikit-learn: Machine learning in Python. *JMLR*, 12(85):2825–2830, 2011.

R. Peharz, R. Gens, F. Pernkopf, and P. M. Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2030–2044, 2017.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *UAI*, 2011.

S. Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 2018.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, pages 206–215, 2019. doi: 10.1038/s42256-019-0048-x.

Y. Shen, A. Choi, and A. Darwiche. A new perspective on learning context-specific independence. In *PGM2020*. PMLR, 2020.

S. Tikka, A. Hyttinen, and J. Karvanen. Identifying causal effects via context-specific independence relations. In *NeurIPS*, 2019.

G. Towell and J. Shavlik. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1991.

A. Vergari, N. D. Mauro, and F. Esposito. Visualizing and understanding sum-product networks. *Mach. Learn.*, 108(4):551–573, 2019.

M. Zecevic, D. S. Dhami, A. Karanam, S. Natarajan, and K. Kersting. Interventional sum-product networks: Causal inference with tractable probabilistic models. *CoRR*, abs/2102.10440, 2021.

H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and bayesian networks. In *ICML '15*, volume 37, pages 116–124, 2015.