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

Athresh Karanam[1] **Saurabh Mathur**[1] **Predrag Radivojac**[2] **David M. Haas**[4]
Kristian Kersting[3] **Sriraam Natarajan**[1]

[1]University of Texas at Dallas
[2]Northeastern University
[3]Technical University of Darmstadt
[4]Indiana University Bloomington

Abstract

We consider the problem of explaining a class of tractable deep probabilistic model, the Sum-Product Networks (SPNs) and present an algorithm $\mathcal{EXSPN}$ to generate explanations. 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.

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 abstract the underlying representation by implementing a composition of probability distributions over domain features, which can be discrete, continuous, graphical, or even unstructured. 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. 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.

A note on interpretability: There is no unique definition of interpretability (Lipton [2018],Doshi-Velez and Kim [2017]). 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.

2 $\mathcal{EXSPN}$ - 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 inde-

[1]Code: `anonymous.4open.science/r/ExSPN-D7C3/`

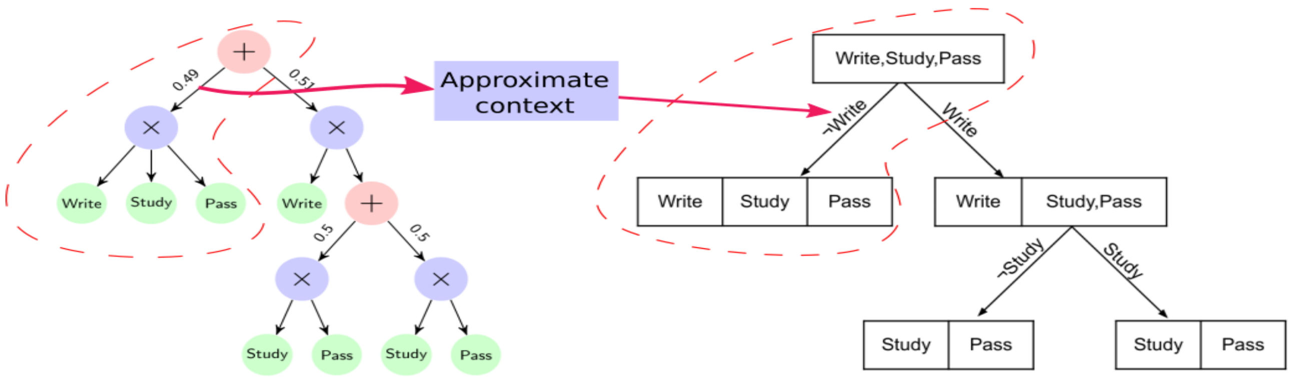


Figure 1: A normal SPN (left) and its corresponding CSI-Tree (right). The independencies induced by the product node are represented in the partitioned leaf node of the CSI-tree. The context in which these independencies hold is approximated to $\neg Write$.

pendence 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 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 2.1. *(CSI-tree): A CSI-tree τ 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 ψ_{csi}, partition function χ *and edge labels ζ.*

DEFINITION 2.2. *(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 ψ 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 ζ 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 1 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 1 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 $Write \perp\!\!\!\perp Study \perp\!\!\!\perp Pass|\neg Write$.

2.1 $\mathcal{EXSPN}$ 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 ϕ.
3. Create a CSI-tree $\tau_{unlabeled}$ with no edge labels using ϕ.
4. Compute edge labels for $\tau_{unlabeled}$, and create CSI-tree τ.
5. Optionally, compress τ into $\tau_{compressed}$ by pruning τ.

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

The five steps involved in $\mathcal{EXSPN}$ are presented below briefly. We provide a detailed description of each step in Appendix A.1. (**Step 1:**) First, it converts S into 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}. This algorithm is similar to the algorithm proposed in Poon and Domingos [2011] for approximating most probable explanation(MPE) inference in arbitrary SPNs. It involves an upward pass to compute values of each sub-SPN in S_{normal} followed by a downward pass to select children of sum nodes with the highest weighted contribution to their respective parent nodes. The algorithm is presented in detail in algorithm 2 in the appendix.

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 $G_{csi} = G_{normal}, \psi_{csi} = \psi_{normal}, \tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$
4  st = [$G_{csi}$.root]
5  while st is not empty do
6      $N_{current} = st.pop()$
7      if $N_{current}$ is not the root node then
8          if $N_{current}$ is a leaf node then
9              Add $\psi(N_{current})$ to $\chi(pa(N_{current}))$
10         end
11         if $N_{current}$ is a sum node then
12             Delete $N_{current}$ from $G_{csi}$ Add $\psi(N_{current})$ to $\chi(pa(N_{current}))$
13             Connect parents and children of $N_{current}$
14             Duplicate children with multiple parents and ensure tree-structure
15         end
16     end
17     Add $ch(N_{current})$ to $st$
18 end
19 $\tau = ComputeLabels(\tau, \mathcal{D}, \phi, \lambda)$
20 return $\tau$
```

(**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 τ **[lines 4-19]**. The CSI-tree is initialized with the SPN's computational graph and scope. Each SPN is then deleted and its parents and children are connected with a directed edge while maintaining a tree structure **[lines 12-17]**. The scope of the leaf nodes is added to the partition function of its parents.

(**Step 4:**) For each edge, it first computes class labels to indicate if a data point in the instance function of the parent node also belongs to the child node. Then it trains a discriminative model f. Finally it computes a set of important features I for f using any feature importance measure. The edge label is then the conjunction of the features in I.

(**Step 5:**) The labeled CSI-tree τ created in the previous step might be too large in some cases and the CSIs induced by τ 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.

Properties of CSI-tree: τ, corresponding to an SPN S 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 2.1. *The CSI-tree inferred from an SPN, S_{normal}, using $\mathcal{EXSPN}$ can infer G'_{normal}, and ψ'_{normal} which encodes the same CSIs as S_{normal}.*

We present the proof in Appendix A.2.

3 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 6 in the appendix. 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 3 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.

Datasets: We evaluated $\mathcal{EXSPN}$ on 11 datasets – one synthetic, 9 benchmark, and one **real clinical study**. We present the details on the datasets used in Appendix A.3.

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 $|C|$).

3.1 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). A visualization of the learned SPN and the corresponding CSI-tree is provided in the Appendix. Hence, 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.

(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 2 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 $oDM = 1$. Many of the CSIs in our tree are validated by an earlier learned model (Karanam et al. [2021]). These **independencies are valdiated by our domain experts** and can be easily checked with related publications. This clearly demonstrates the potential of explaining a joint model such as SPN in a real clinically relevant domain.

4 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.

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

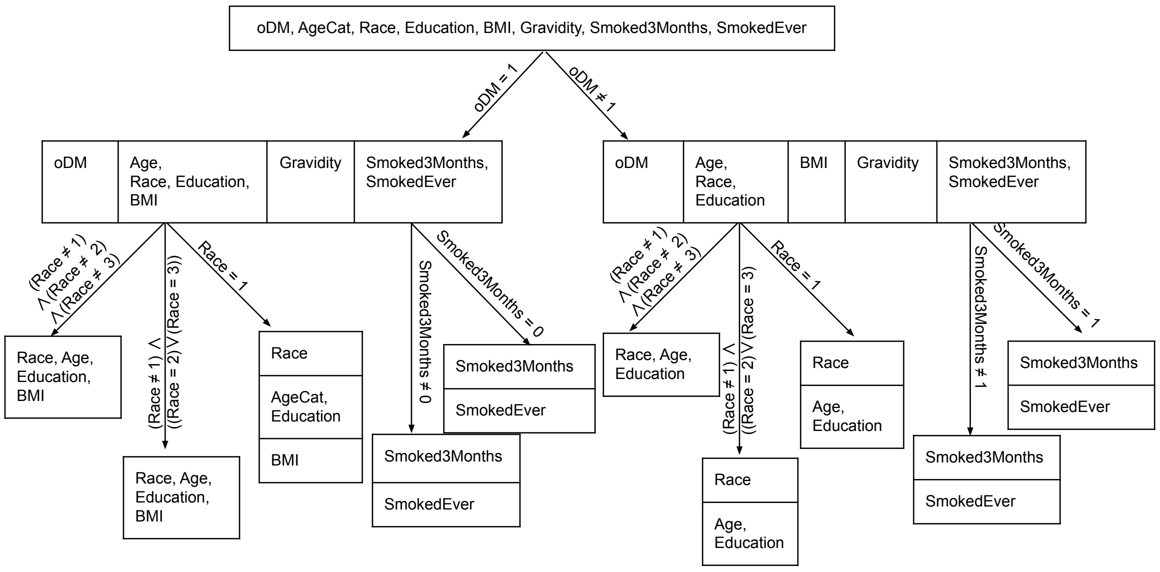

oDM, AgeCat, Race, Education, BMI, Gravidity, Smoked3Months, SmokedEver
oDM = 1
oDM ≠ 1
oDM
Age, Race, Education, BMI
Gravidity
Smoked3Months, SmokedEver
oDM
Age, Race, Education
BMI
Gravidity
Smoked3Months, SmokedEver
(Race ≠ 1) ∧(Race ≠ 2) ∧(Race ≠ 3)
(Race ≠ 1) ∧ ((Race = 2)∨(Race = 3))
Race = 1
Race, Age, Education, BMI
Race, Age, Education, BMI
Race
AgeCat, Education
BMI
Smoked3Months ≠ 0
Smoked3Months = 0
Smoked3Months
SmokedEver
Smoked3Months
SmokedEver
(Race ≠ 1) ∧(Race ≠ 2) ∧(Race ≠ 3)
(Race ≠ 1) ∧ ((Race = 2)∨(Race = 3))
Race = 1
Race, Age, Education
Race
Age, Education
Race
Age, Education
Smoked3Months ≠ 1
Smoked3Months = 1
Smoked3Months
SmokedEver
Smoked3Months
SmokedEver

Figure 2: 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.

References

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

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *UAI*, 1996.

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

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

Johannes Fürnkranz and Tomáš Kliegr. A brief overview of rule learning. Springer International Publishing, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

David M Haas, Corette 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.

Athresh Karanam, Alexander L. Hayes, Harsha Kokel, David M. Haas, Predrag Radivojac, and Sriraam Natarajan. A probabilistic approach to extract qualitative knowledge for early prediction of gestational diabetes. In *AIME*, 2021.

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

Zachary 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.

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

Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *CoRR*, abs/1706.07979, 2017. URL `http://arxiv.org/abs/1706.07979`.

Henrik Nyman, Johan Pensar, Timo Koski, and Jukka Corander. Stratified graphical models-context-specific independence in graphical models. *Bayesian Analysis*, 9(4):883–908, 2014.

Henrik Nyman, Johan Pensar, Timo Koski, and Jukka Corander. Context-specific independence in graphical log-linear models. *Computational Statistics*, 2016.

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

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

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

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

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

Santtu Tikka, Antti Hyttinen, and Juha Karvanen. Identifying causal effects via context-specific independence relations. In *NeurIPS*, 2019.

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

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Visualizing and understanding sum-product networks. *Mach. Learn.*, 108(4):551–573, 2019.

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

A Appendix

A.1 Details on $\mathcal{EXSPN}$

The steps involved in converting an arbitrary SPN into a CSI-tree is illustrated in figure 3.

The five steps involved in $\mathcal{EXSPN}$ are described below:

(**Step 1:**) First, it converts S into 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}. This algorithm, presented in 2, 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 τ **[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 τ**[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 τ is unlabeled.

(**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 ζ. 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 λ. The edge label for e is then the conjunction of the features in I.

(**Step 5:**) The labeled CSI-tree τ created in the previous step might be too large in some cases and the CSIs induced by τ 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.

Computational Complexity of $\mathcal{EXSPN}$: 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 $O(|S|^2)$, as mentioned in line 400 of the original manuscript. 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 θ_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* $O(max(|\theta|, |X|)|S|^2|\mathcal{D}|)$.

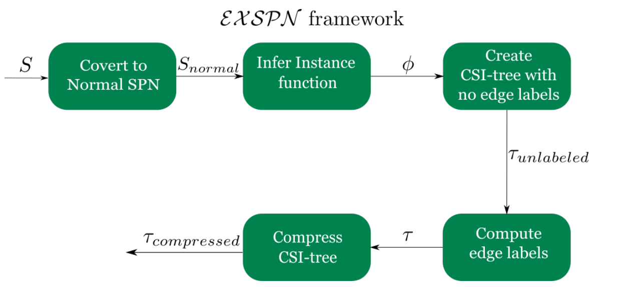


Figure 3: Flowchart illustrating the steps involved in converting an arbitrary SPN into a more interpretable CSI-tree.

Algorithm 2: InferInstanceFunction

```
input  : $\mathcal{D}$, $\mathbf{X}$, normal SPN $S$
output : Instance function $\phi$
1 for Each instance $\mathcal{D}_j \in \mathcal{D}$ do
2     Perform upward pass for $\mathcal{D}$ and compute $S_N^{max}(\mathcal{D}_j)$
        for each node $N$
3     Perform a downward pass, appending $\mathcal{D}_j$ to the
        $\phi(N_{ch})$ of a child $N_{ch}$ of a sum node that led to
        $S^{max}(\mathcal{D}_j)$ and to $\phi(N_{product})$ for all product
        nodes $N_{product}$ along the path from the root to a
        leaf node
4 end
5 return $\phi$
```

Algorithm 3: ComputeLabels

```
input  : $\tau = (G, \psi, \chi, NULL), \mathcal{D}, \phi, \lambda$
output : Labeled CSI-tree $\tau = (G, \psi, \chi, \zeta)$
1 for Each edge $e(N_{from}, N_{to})$ in $G$ do
2     Compute class labels $y$
3     $f = TrainModel(\mathcal{D}, \phi(N_{from}), \mathbf{y})$
4     Compute a set of important features $I$ for $f$
5     $\zeta(e) = \wedge_{i \in I} i$ thresholded by $\lambda$
6 end
7 return $\tau = (G, \psi, \chi, \zeta)$
```

A.2 Proofs

THEOREM A.1. *The CSI-tree $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$, inferred from an SPN, $S_{normal} = (G_{normal}, \psi_{normal}, w_{normal}, \theta_{normal})$, using $\mathcal{EXSPN}$ can infer G'_{normal}, and ψ'_{normal} which encodes the same CSIs as S_{normal}.*

Proof. Algorithm 4 that retrieves G_{normal} and ψ_{normal} associated with S_{normal}, given τ provides the proof. It performs DFS over G_{normal} and identifies two main cases of $N_{current}$: **1.** A node where the length of the partition function $\chi(N_{current})$ is equal to length of the scope function $N_{current}$**[line 6] 2.** A node where that's not the case**[line 10]**. In the first case, it replaces $N_{current}$ with a product node N_p such that $\psi(N_p) = \psi_{csi}(N_{current})$ and adds $|\chi(N_{current})|$ number of leaf

Algorithm 4: RetrieveSPN

```
   input  : CSI-tree $\tau = (G_{csi}, \psi_{csi}, \chi, \zeta)$
   output : $G_{normal}, \psi_{normal}$
 1 initialize: $G_{normal} = G_{csi}, \psi_{normal} = \psi_{csi}$
 2 $st = [G_{normal}.root]$
 3 while st is not empty do
 4     $N_{current} = st.pop()$
 5     if $N_{current}$ has not been visited and is not the root node then
 6         if $|\chi(N_{current})| = |\psi_{csi}(N_{current})|$ then
 7             Replace $N_{current}$ with $N_p$
 8             Append $|\chi(N_{current})|$ leaf nodes
 9         if $|\chi(N_{current})| \neq |\psi_{csi}(N_{current})|$ then
10             for k in $\chi(N_{current})$ do
11                 if $|k| = 1$ then
12                     Append $N_l$ s.t. $\psi(N_l) = k$
13                 if $|k| \neq 1$ then
14                     Append $N_s$ s.t. $\psi(N_s) = k$ for c in $ch(N_{current})$ do
15                         Append $N_p$ to $N_s$ if $\psi_{csi}(c) = k$
16                     end
17             end
18     Add $ch(N_{current})$ to $st$
19 end
20 return $G_{normal}, \psi_{normal}$
```

nodes. In the second case, it iterates over the subsets in $\chi(N_{current})$, adds a leaf node if that subset k is a singleton set and adds an intermediate sum node N_s for all other children associated with a subset k**[lines 10-17]**. Then, it returns computational graph G and scope function ψ. Although this algorithm retrieves only G_{normal} and ψ_{normal} which define the structure of the SPN, the parameters of the SPN w_{normal} and θ can also be retrieved by modifying $\mathcal{EXSPN}$ to produce a CSI-tree that stores these parameters in the leaf nodes of the CSI-tree alongside the scope ψ_{csi} and the partition χ. This modification is trivial and does not improve the interpretability. Hence, we do not consider it. □

A.3 Details on datasets used

We generated the synthetic dataset by sampling 10,000 instances each from 3 multivariate Gaussians.

$$\mathcal{N}\left[\begin{pmatrix}2\\2\\2\\2\end{pmatrix}, 0.01\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\right],$$

$$\mathcal{N}\left[\begin{pmatrix}-8\\4\\4\\4\end{pmatrix}, 0.01\begin{pmatrix}1&0&0&0\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{pmatrix}\right],$$

$$\mathcal{N}\left[\begin{pmatrix}8\\8\\8\\8\end{pmatrix}, 0.01\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&1\\0&0&1&1\end{pmatrix}\right]$$

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 representing 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 2 summarizes the datasets.

A.4 Additional Experimental Results

Table 4 presents the mean log-likelihood over the test set for the SPNs trained on each of the datasets.

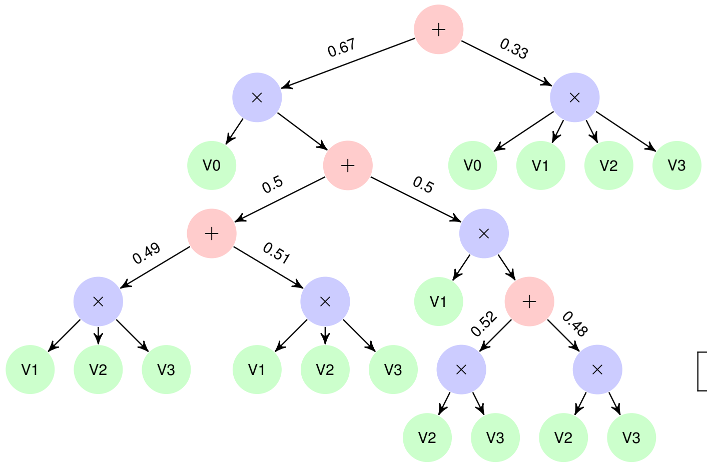
Figure 4 shows the SPN learnt from the synthetic data and the CSI-tree extracted from that SPN using $\mathcal{EXSPN}$. The structure of the CSI-tree closely resembles the structure of the SPN and the CSI-tree, in this case, is smaller than the original SPN. Also, the internal parameters of the CSI-tree are logical statements comprising of the observed variables $V0, V1, V2, V3$. These properties of the CSI-tree make it interpretable when presented to a domain expert.

Table 2: Dataset details.

Dataset	Type	$\lvert\mathbf{X}\rvert$	Train	Test
Synthetic	Continuous	4	22,500	7,500
Mushroom	Categorical	23	4,233	1,411
Plants	Binary	70	17,411	5,804
NLTCS	Binary	16	16,180	5,394
MSNBC	Binary	17	291,325	97,109
Abalone	Mixed	9	3,132	1,045
Adult	Mixed	15	33,916	11,306
Wine quality	Continuous	12	4,872	1,625
Car	Categorical	7	1,296	432
Yeast	Mixed	9	1,113	371
nuMoM2b	Categorical	8	8,832	388

Table 3: Hyperparameter configurations.

Component	Hyperparameter	Value
SPN	rows	GMM
	min_instances_slice (mis)	1% of $\lvert Train\rvert$
		5% of $\lvert Train\rvert$ (Synthetic)
		1% of $\lvert Train_{oDM=1}\rvert$ (nuMoM2b)
DT	max_depth	2
	min_impurity_decrease	0.1
	class_weight	balanced
$\mathcal{EXSPN}$	min_precision	0.7
	min_recall	0.7
	n_instances	$5 \times mis$
		mis (Synthetic)



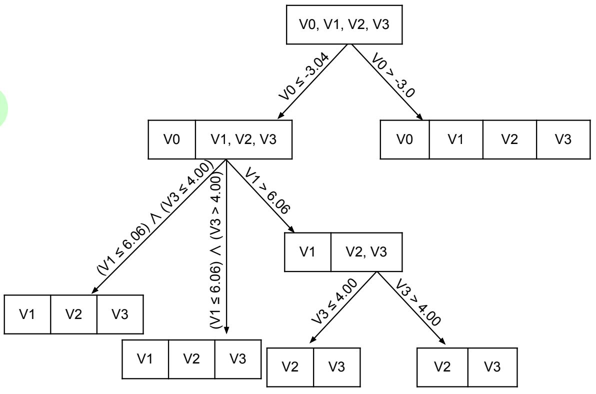


Figure 4: The sum-product network (left) trained on the synthetic data set and its corresponding CSI-tree extracted by $\mathcal{EXSPN}$ (right). Each edge in the CSI-tree corresponds to an edge from a sum node to a product node in the SPN. All other edges are collapsed. Clearly, the CSI-tree encodes all the CSIs represented in the sum-product network.

Table 4: The mean log-likelihood over the test set (LL) for the SPNs trained on each of the datasets.

Dataset	LL
Synthetic	2.83
Mushroom	-8.98
Plants	-14.03
NLTCS	-6.3
MSNBC	-6.68
Abalone	18.99
Adult	-5.52
Wine quality	-3.55
Car	-7.92
Yeast	46.28
nuMoM2b	-6.92

Table 5: The Minimum Support (MS) and Minimum Confidence (MC) parameters used for the apriori algorithm and the mean confidence of the association rules over the test set (TC).

Dataset	TC	MS	MC
Synthetic	0.94	0.5	0.7
Mushroom	0.91	0.5	0.7
Plants	0.84	0.15	0.7
NLTCS	0.83	0.25	0.7
MSNBC	0.75	0.01	0.7
Abalone	0.87	0.25	0.7
Adult	0.85	0.5	0.7
Wine quality	0.86	0.5	0.7
Car	0.93	0.1	0.7
Yeast	0.9	0.5	0.7
nuMoM2b	0.87	0.5	0.7

Table 6: 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