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

Athresh Karanam Saurabh Mathur Predrag Radivojac David M. Haas
Kristian Kersting Sriraam Natarajan

1University of Texas at Dallas 2Northeastern University 3Technical University of Darmstadt 4Indiana 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 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]). CSI-trees 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 (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 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 EXSPN.

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 EXSPN - Explaining 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-
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 ¬Write.

Dependence 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 \( \tau \) is a 4-tuple \((G, \psi_{csi}, \chi, \zeta)\) where \( G \) is a tree with a set of variables \( V \subseteq X \), scope function \( \psi_{csi} \), partition function \( \chi \) and edge labels \( \zeta \).

**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 \( \psi \) such that \( \cup_{i \in |C|} P_i = \psi(N) \).

Each node, \( N \), of a CSI-tree has a scope \( \psi_{csi}(N) \subseteq X \). 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 \( X \). An example of an edge label is \( (X_i \geq 0.5 \land X_j \leq 1) \). 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 \( \langle \text{Write}, \text{Study}, \text{Pass} \rangle \). The set of variables inscribed within each node represent the scope of that node. For example, the scope of the root node is \( \langle \text{Write}, \text{Study}, \text{Pass} \rangle \). 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 \( (\langle \text{Write} \rangle, \langle \text{Study}, \text{Pass} \rangle) \) of the scope of that node.

The left child of the root node induces a CSI-relation \( \text{Write} \perp \text{Study} \perp \text{Pass} \mid \neg \text{Write} \). Similarly, the right child of the root node induces two CSI-relations \( \text{Write} \perp \text{Study} | \text{Write} \) and \( \text{Write} \perp \text{Pass} | \text{Write} \). 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 \( D \)

**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 \( \langle \text{Write}, \text{Study}, \text{Pass} \rangle \). The root node \( N_0 \) is a sum node, implying that \( \text{Write}, \text{Study}, \text{Pass} \) 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 \( Write \perp Study \perp Pass|\phi(N_1) \). 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 \( \land_{i \in I} \). 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 \( (-Write) \) as the approximated context. So, the original CSI \( Write \perp Study \perp Pass|\phi(N_1) \) is approximated to \( Write \perp Study \perp Pass|\neg Write \).

### 2.1 \( \varepsilon\lambda\Psi^N \) 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 \( S_{\text{normal}} \).
2. Infer the instance function \( \phi \).
3. Create a CSI-tree \( \tau_{\text{unlabeled}} \) with no edge labels using \( \phi \).
4. Compute edge labels for \( \tau_{\text{unlabeled}} \) and create CSI-tree \( \tau \).
5. Optionally, compress \( \tau \) into \( \tau_{\text{compressed}} \) by pruning \( \tau \).

Algorithm 1 presents \( \varepsilon\lambda\Psi^N : \) Explaining Som-Product Networks. It infers a CSI-tree \( \tau \), given an SPN \( S \), data \( D \), set of random variables \( X \) and feature importance score threshold \( \lambda \).

The five steps involved in \( \varepsilon\lambda\Psi^N \) are presented below briefly. We provide a detailed description of each step in Appendix A.1 (Step 1): It first converts \( S \) into a normal-SPN \( S_{\text{normal}} = (G_{\text{normal}}, \psi_{\text{normal}}, w_{\text{normal}}, \theta_{\text{normal}}) \)[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 \( S_{\text{normal}} \) that represents the same joint probability over variables and \( |S_{\text{normal}}| = O(|S|^2) \).

(Step 2): It then infers the instance function for \( S_{\text{normal}} \). 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 \( S_{\text{normal}} \) 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.

(Step 3): Next, it performs a depth-first search (DFS) on the computational graph \( G_{\text{normal}} \) associated with \( S_{\text{normal}} \) and constructs the tree-structured graph \( G_{\text{CSI}} \) associated with \( \tau \) [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 \( \tau \) 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 \( |\phi(N)| < \min \text{instances} \). 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**: \( \tau \), corresponding to an SPN \( S \) obtained through \( E\times S P N \) 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 \( 2^{(S(N_p))} \) parameters to be sufficiently expressed, while the approximate context from \( E\times S P N \) 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 \( E\times S P N \) can infer \( G_{normal} \), and \( \psi_{normal} \) 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 \( E\times S P N \) recover all the CSIs encoded in an SPN? (Q2: Compression) Can the CSIs be compressed further? (Q3. Baseline) How do the CSIs extracted using \( E\times S P N \) compare with a strong rule learner? (Q4. Real data) Does \( E\times S P N \) extract reasonable CSIs in a real clinical study? System: We implemented \( E\times S P N \) 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 1 in the Appendix.

**Baseline**: To evaluate the CSIs extracted by \( E\times S P N \), 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 \( E\times S P N \) 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 - \( \text{min}_\text{precision} \) (mp), \( \text{min}_\text{recall} \) (mr), and \( \text{n}_\text{instances} \) (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 \( 5 \times \text{min}_\text{instances}_\text{slice} \) 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 positive results and the reduced set of CSI rules. For each dataset in the table, the number of CSIs extracted by \( E\times S P N \) (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 \( E\times S P N \) 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 \( E\times S P N \) 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 \( Age, Race, Education \). 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.

(Q4. Are the explanations correct?) Figure 2 shows the first two levels of the CSI-tree extracted by \( E\times S P N \) 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 \( Age, Race, Education \). 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.

#### 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 and validated the correctness of the conversion. Our experiments in synthetic, benchmarks and a real clinical study demonstrate this potential for explaining a joint model such as SPN in a real clinically relevant domain.
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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SPN</th>
<th>All CSIs</th>
<th>Reduced CSIs</th>
<th>Association Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
<td>NR</td>
<td>MA</td>
<td>MC</td>
</tr>
<tr>
<td>Synthetic</td>
<td>7</td>
<td>7</td>
<td>2.29</td>
<td>2.57</td>
</tr>
<tr>
<td>Mushroom</td>
<td>39</td>
<td>39</td>
<td>5.90</td>
<td>8.54</td>
</tr>
<tr>
<td>Plants</td>
<td>342</td>
<td>342</td>
<td>9.60</td>
<td>9.61</td>
</tr>
<tr>
<td>NLTCS</td>
<td>74</td>
<td>74</td>
<td>9.84</td>
<td>3.32</td>
</tr>
<tr>
<td>MSNBC</td>
<td>8</td>
<td>8</td>
<td>4.12</td>
<td>5.75</td>
</tr>
<tr>
<td>Abalone</td>
<td>194</td>
<td>194</td>
<td>11.31</td>
<td>7.00</td>
</tr>
<tr>
<td>Adult</td>
<td>263</td>
<td>263</td>
<td>14.49</td>
<td>4.02</td>
</tr>
<tr>
<td>Wine quality</td>
<td>236</td>
<td>236</td>
<td>12.45</td>
<td>6.76</td>
</tr>
<tr>
<td>Car</td>
<td>18</td>
<td>18</td>
<td>5.22</td>
<td>2.50</td>
</tr>
<tr>
<td>Yeast</td>
<td>181</td>
<td>181</td>
<td>16.20</td>
<td>3.26</td>
</tr>
<tr>
<td>nuMoM2b</td>
<td>104</td>
<td>104</td>
<td>10.60</td>
<td>2.33</td>
</tr>
</tbody>
</table>

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


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.


The five steps involved in \(\text{EXSPN}\) are described below:

\[G_{\text{normal}} = \left( G_{\text{normal}}, w_{\text{normal}}, \theta_{\text{normal}} \right)\]

1. **Algorithm 3: ComputeLabels**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi, \lambda\)
   **Output**: Labeled CSI-tree \(\tau = (G, \psi, \chi, \zeta)\)

   - For each edge \(e(N_{\text{from}}, N_{\text{to}})\) in \(G\), do
     - Compute class labels \(y\)
     - \(f = \text{TrainModel}(D, \phi(N_{\text{from}}), y)\)
     - Compute a set of important features \(I\) for \(f\)
     - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)
   - Return \(\tau = (G, \psi, \chi, \zeta)\)

2. **Algorithm 2: InferInstanceFunction**

   **Input**: \(\tau = (G, \psi, \chi, \zeta)\), \(D\), \(\phi, \lambda\)
   **Output**: Instance function \(\phi\)

   - For each instance \(D_j \in D\) do
     - Perform upward pass for \(D\) and compute \(S_{\text{norm}}^\text{max}(D_j)\) for each node \(N\)
     - Perform a downward pass, appending \(\phi(N_{\text{from}})\) to the root of \(\tau\) and computing \(S_{\text{norm}}^\text{max}(D_j)\) for each product node \(N_{\text{product}}\) along the path from the root to a leaf node

3. **Algorithm 1: ComputeLabels**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi, \lambda\)
   **Output**: CSI-tree \(\tau\)

   - Label each node \(N\) in \(D\) with \(\phi\)
   - Compute a set of important features \(I\)
   - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

4. **Algorithm 0: LabeledCSI-tree**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi, \lambda\)
   **Output**: Labeled CSI-tree \(\tau = (G, \psi, \chi, \zeta)\)

   - For each edge \(e(N_{\text{from}}, N_{\text{to}})\) in \(G\), do
     - Compute class labels \(y\)
     - \(f = \text{TrainModel}(D, \phi(N_{\text{from}}), y)\)
     - Compute a set of important features \(I\) for \(f\)
     - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

5. **Algorithm 3: ComputeLabels**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi\)
   **Output**: CSI-tree \(\tau = (G, \psi, \chi, NULL)\)

   - Label each node \(N\) in \(D\) with \(\phi\)
   - Compute a set of important features \(I\)
   - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

6. **Algorithm 2: InferInstanceFunction**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D\), \(\phi, \lambda\)
   **Output**: Instance function \(\phi\)

   - For each instance \(D_j \in D\) do
     - Perform upward pass for \(D\) and compute \(S_{\text{norm}}^\text{max}(D_j)\) for each node \(N\)
     - Perform a downward pass, appending \(\phi(N_{\text{from}})\) to the root of \(\tau\) and computing \(S_{\text{norm}}^\text{max}(D_j)\) for each product node \(N_{\text{product}}\) along the path from the root to a leaf node

7. **Algorithm 1: ComputeLabels**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi\)
   **Output**: CSI-tree \(\tau = (G, \psi, \chi, NULL)\)

   - Label each node \(N\) in \(D\) with \(\phi\)
   - Compute a set of important features \(I\)
   - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

8. **Algorithm 0: LabeledCSI-tree**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi\)
   **Output**: CSI-tree \(\tau = (G, \psi, \chi, NULL)\)

   - Label each node \(N\) in \(D\) with \(\phi\)
   - Compute a set of important features \(I\)
   - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

9. **Algorithm 3: ComputeLabels**

   **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi\)
   **Output**: CSI-tree \(\tau = (G, \psi, \chi, NULL)\)

   - Label each node \(N\) in \(D\) with \(\phi\)
   - Compute a set of important features \(I\)
   - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)

10. **Algorithm 2: InferInstanceFunction**

    **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D\), \(\phi, \lambda\)
    **Output**: Instance function \(\phi\)

    - For each instance \(D_j \in D\) do
      - Perform upward pass for \(D\) and compute \(S_{\text{norm}}^\text{max}(D_j)\) for each node \(N\)
      - Perform a downward pass, appending \(\phi(N_{\text{from}})\) to the root of \(\tau\) and computing \(S_{\text{norm}}^\text{max}(D_j)\) for each product node \(N_{\text{product}}\) along the path from the root to a leaf node

11. **Algorithm 1: ComputeLabels**

    **Input**: \(\tau = (G, \psi, \chi, NULL)\), \(D, \phi\)
    **Output**: CSI-tree \(\tau = (G, \psi, \chi, NULL)\)

    - Label each node \(N\) in \(D\) with \(\phi\)
    - Compute a set of important features \(I\)
    - \(\zeta(e) = \land_{i \in I} \text{thresholded by } \lambda\)
We used Abalone, Adult, Wine quality, Car, and Yeast. In MSNBC and Plants, we these parameters in the leaf nodes of the CSI-tree alongside the scope also be retrieved by modifying the structure of the SPN, the parameters of the SPN though this algorithm retrieves only A.3 Details on datasets used

We generated the synthetic 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 (moM2b, 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 2 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 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.

Algorithm 4: RetrieveSPN

```
input : CSI-tree \( \tau = (G_{\text{cat}}, \psi_{\text{cat}}, \chi, \zeta) \)
output : \( G_{\text{normal}}, \psi_{\text{normal}} \)
1 initialize \( G_{\text{normal}} = G_{\text{cat}}, \psi_{\text{normal}} = \psi_{\text{cat}} \)
2 \( st = [G_{\text{normal}}, \text{root}] \)
3 while \( st \) is not empty do
4 \( N_{\text{current}} = st.\text{pop}() \)
5 if \( N_{\text{current}} \) has not been visited and is not the root node then
6 \( \text{if } \chi(N_{\text{current}}) = \psi_{\text{cat}}(N_{\text{current}}) \text{ then} \)
7 Replace \( N_{\text{current}} \) with \( N_0 \)
8 Append \( \chi(N_{\text{current}}) \) leaf nodes
9 \( \text{if } \chi(N_{\text{current}}) \neq \psi_{\text{cat}}(N_{\text{current}}) \text{ then} \)
10 for \( k \) in \( \chi(N_{\text{current}}) \) do
11 \( \text{if } |k| = 1 \text{ then} \)
12 Append \( N_1 \) s.t. \( \psi(N_1) = k \)
13 \( \text{if } |k| \neq 1 \text{ then} \)
14 Append \( N_a \) s.t. \( \psi(N_a) = k \) for \( c \)
15 \( \text{in } \psi_{\text{cat}}(k) \text{ do} \)
16 Append \( N_p \) to \( N_a \) if \( \psi_{\text{cat}}(c) = k \)
17 end
18 end
19 \( \text{Add } \psi_{\text{cat}}(N_{\text{current}}) \text{ to } st \)
20 return \( G_{\text{normal}}, \psi_{\text{normal}} \)
```

In the second case, it iterates over the subsets in \( \chi(N_{\text{current}}) \), adds a leaf node if that subset \( k \) is a singleton set and adds an intermediate sum node \( N_a \) for all other children associated with a subset \( k \) in lines 10-17. Then, it returns computational graph \( G \) and scope function \( \psi \). Although this algorithm retrieves only \( G_{\text{normal}} \) and \( \psi_{\text{normal}} \) which define the structure of the SPN, the parameters of the SPN \( w_{\text{normal}} \) and \( \theta \) can also be retrieved by modifying EXSPN to produce a CSI-tree that stores these parameters in the leaf nodes of the CSI-tree alongside the scope \( \psi_{\text{cat}} \) and the partition \( \chi \). 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.

\[
\mathcal{N} \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 0.01 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathcal{N} \begin{bmatrix} -8 \\ 4 \\ 4 \\ 4 \\ 0.01 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathcal{N} \begin{bmatrix} 8 \\ 8 \\ 8 \\ 8 \\ 0.01 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}
\]

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.
Table 2: Dataset details.

| Dataset     | Type       | $|X|$ | 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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN</td>
<td>rows</td>
<td>GMM</td>
</tr>
<tr>
<td></td>
<td>min_instances_slice (mis)</td>
<td>1% of $</td>
</tr>
<tr>
<td></td>
<td>max_depth</td>
<td>5% of $</td>
</tr>
<tr>
<td></td>
<td>oDM</td>
<td>$1_{oDM}$</td>
</tr>
<tr>
<td></td>
<td>class_weight</td>
<td>balanced</td>
</tr>
<tr>
<td></td>
<td>$EXSPN$</td>
<td>min_precision</td>
</tr>
<tr>
<td></td>
<td>min_recall</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>n_instances</td>
<td>$5 \times mis$</td>
</tr>
</tbody>
</table>

Figure 4: The sum-product network (left) trained on the synthetic data set and its corresponding CSI-tree extracted by $EXSPN$ (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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>LL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>2.83</td>
</tr>
<tr>
<td>Mushroom</td>
<td>-8.98</td>
</tr>
<tr>
<td>Plants</td>
<td>-14.03</td>
</tr>
<tr>
<td>NLTCS</td>
<td>-6.3</td>
</tr>
<tr>
<td>MSNBC</td>
<td>-6.68</td>
</tr>
<tr>
<td>Abalone</td>
<td>18.99</td>
</tr>
<tr>
<td>Adult</td>
<td>-5.52</td>
</tr>
<tr>
<td>Wine quality</td>
<td>-3.55</td>
</tr>
<tr>
<td>Car</td>
<td>-7.92</td>
</tr>
<tr>
<td>Yeast</td>
<td>-46.28</td>
</tr>
<tr>
<td>nuMoM2b</td>
<td>-6.92</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TC</th>
<th>MS</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>0.94</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.91</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Plants</td>
<td>0.84</td>
<td>0.15</td>
<td>0.7</td>
</tr>
<tr>
<td>NLTCS</td>
<td>0.83</td>
<td>0.25</td>
<td>0.7</td>
</tr>
<tr>
<td>MSNBC</td>
<td>0.75</td>
<td>0.01</td>
<td>0.7</td>
</tr>
<tr>
<td>Abalone</td>
<td>0.87</td>
<td>0.25</td>
<td>0.7</td>
</tr>
<tr>
<td>Adult</td>
<td>0.85</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Wine quality</td>
<td>0.86</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Car</td>
<td>0.93</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.9</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>nuMoM2b</td>
<td>0.87</td>
<td>0.5</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Table 6: Summary statistics for the case where the instance function is inferred after training. The columns are the same as Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SPN</th>
<th>All CSIs</th>
<th>Reduced CSIs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
<td>NR</td>
<td>MA</td>
</tr>
<tr>
<td>Synthetic</td>
<td>7</td>
<td>7</td>
<td>2.29</td>
</tr>
<tr>
<td>Mushroom</td>
<td>39</td>
<td>39</td>
<td>5.90</td>
</tr>
<tr>
<td>Plants</td>
<td>342</td>
<td>342</td>
<td>8.33</td>
</tr>
<tr>
<td>NLTCS</td>
<td>74</td>
<td>74</td>
<td>9.74</td>
</tr>
<tr>
<td>MSNBC</td>
<td>8</td>
<td>8</td>
<td>4.12</td>
</tr>
<tr>
<td>Abalone</td>
<td>194</td>
<td>157</td>
<td>9.61</td>
</tr>
<tr>
<td>Adult</td>
<td>263</td>
<td>244</td>
<td>11.27</td>
</tr>
<tr>
<td>Wine</td>
<td>236</td>
<td>235</td>
<td>9.18</td>
</tr>
<tr>
<td>Car</td>
<td>18</td>
<td>18</td>
<td>5.22</td>
</tr>
<tr>
<td>Yeast</td>
<td>181</td>
<td>181</td>
<td>13.06</td>
</tr>
<tr>
<td>nuMoM2b</td>
<td>104</td>
<td>98</td>
<td>9.64</td>
</tr>
</tbody>
</table>