Exchangeability-Aware Sum-Product Networks

Stefan Lüdtke[1*] , **Christian Bartelt**[1] and **Heiner Stuckenschmidt**[2]

[1]Institute for Enterprise Systems, University of Mannheim, Germany
[2]Data and Web Science Group, University of Mannheim, Germany

{luedtke,bartelt}@es.uni-mannheim.de, heiner@informatik.uni-mannheim.de

Abstract

Sum-Product Networks (SPNs) are expressive probabilistic models that provide exact, tractable inference. They achieve this efficiency by making use of local independence.

On the other hand, mixtures of exchangeable variable models (MEVMs) are a class of tractable probabilistic models that make use of *exchangeability* of discrete random variables to render inference tractable. Exchangeability, which arises naturally in relational domains, has not been considered for efficient representation and inference in SPNs yet.

The contribution of this paper is a novel probabilistic model which we call *Exchangeability-Aware Sum-Product Networks* (XSPNs). It contains both SPNs and MEVMs as special cases, and combines the ability of SPNs to efficiently learn deep probabilistic models with the ability of MEVMs to efficiently handle exchangeable random variables. We introduce a structure learning algorithm for XSPNs and empirically show that they can be more accurate than conventional SPNs when the data contains repeated, interchangeable parts.

1 Introduction

The accurate representation of probability distributions and efficient inference in such distributions is fundamental in machine learning. Recently, probabilistic models based on deep neural networks, like Variational Autoencoders [Kingma and Welling, 2014], neural autoregressive models [Larochelle and Murray, 2011] and normalizing flows [Rezende and Mohamed, 2015] have received a lot of attention. They have been very successful for tasks like data generation, anomaly detection, and prediction. However, these models lack efficient and exact inference capabilities.

Probabilistic circuits (PCs) like Probabilistic Sentential Decision Diagrams [Liang *et al.*, 2017] or Cutset Networks [Rahman *et al.*, 2014] are deep probabilistic models that permit efficient, exact marginal inference. Sum-Product Networks (SPNs) [Poon and Domingos, 2011], one of the most

prominent PCs, represent probability distributions as a computation graph that consists of sum nodes (representing mixtures) and product nodes (representing independent factors). Time complexity of marginal inference in SPNs is linear in the network size.

In this paper, we are specifically interested in efficient inference in (and learning of) discrete distributions involving repeated, interchangeable components. Such distributions arise naturally in *relational* domains, consisting of multiple, interrelated entities, as illustrated by the following example:

Example 1. *Suppose n people are invited to a workshop, and we want to estimate how many of them will attend. We assume that attendance of each person depends on whether the topic of the workshop is considered "hot". Additionally, attendance of each person depends on how many of the other people will attend.*

In this domain, attendance of each person is not independent, but depends on the attendance of all other people. Instead, however, the probability that exactly k people attend does not depend on the specific *identities* of those k people. More formally, the random variables representing attendance are *exchangeable*—their joint distribution is invariant under permutation of their assignment. Unfortunately, tree-structured SPNs that are generated by SPN structure learning algorithms (as well as graphical models based on the notion of conditional independence) do not efficiently encode distributions over exchangeable RVs.

Still, similar to conditional independence, finite exchangeability can substantially reduce the complexity of the model and allow for tractable inference [Niepert and Van den Broeck, 2014]. This property is exploited by Mixtures of Exchangeable Variable Models (MEVMs) [Niepert and Domingos, 2014]. MEVMs are tractable probabilistic models that use efficient representations of exchangeable distributions as building blocks. MEVMs can be seen as shallow SPNs, consisting of a single sum and product layer, and efficient representations of finite exchangeable sequences at the leaves. However, deep SPN architectures can represent some functions more efficiently than shallow architectures [Delalleau and Bengio, 2011]. Despite their obvious connection, no attempts have been made yet to combine SPNs and MEVMs into a unified formalism.

The main contribution of this paper is to provide such a unified model, that contains both MEVMs and (conven-
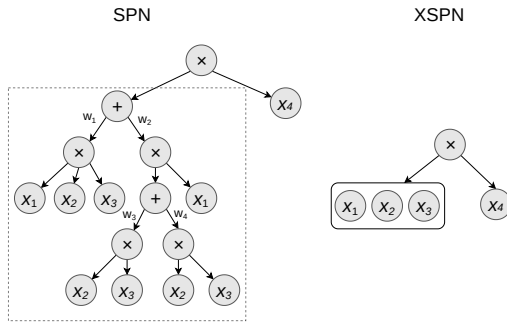
Figure 1: Left: SPN generated by LearnSPN from 1000 samples from a distribution of the form $p(X_1, X_2, X_3, X_4) = p(X_1, X_2, X_3)\, p(X_4)$, where $p(X_1, X_2, X_3)$ is exchangeable. This factor contains no (local) independence, thus LearnSPN learns a complicated structure that in essence represents the factor by full enumeration (framed subtree). Right: XSPN representing the same distribution. The XSPN is more compact, because it can directly represent exchangeable distributions efficiently.

tional) SPNs as special cases. The proposed model combines the ability of SPNs to efficiently represent and learn deep probabilistic models with the ability of MEVMs to efficiently handle exchangeable random variables. We show that marginal inference in the proposed model—which we call *Exchangeability-Aware* SPNs (XSPNs)—is tractable. The general concept is shown in Figure 1. Furthermore, we introduce a structure learning algorithm for XSPNs, which recursively performs statistical tests of finite exchangeability, and fits an exchangeable leaf distribution when appropriate. Finally, we empirically demonstrate that our structure learning algorithm can achieve significantly higher log-likelihoods than conventional SPN structure learning. This work is a first step towards learning probabilistic circuits for relational domains.

2 Sum-Product Networks

Representation. An SPN [Poon and Domingos, 2011] is a rooted directed acyclic graph that represents a probability distribution over a sequence of RVs $\mathbf{X} = X_1, \ldots, X_n$. Each node N in the graph represents a probability distribution P_N over a subset $\mathbf{X}_{\phi(N)} \subseteq \mathbf{X}$ of the RVs, where $\phi(N) \subseteq \{1, \ldots, n\}$ is called the *scope* of the node N. An SPN consists of three types of nodes: distribution nodes, sum nodes and product nodes. All leaf nodes of the graph are distribution nodes, and all internal nodes are either sum or product nodes. In the following, we denote the set of children of the node N as $\mathrm{ch}(N)$.

A distribution node N encodes a tractable probability distribution $P_N(\mathbf{X}_{\phi(N)})$ over the RVs in its scope. Early works on SPNs used univariate distributions, e.g. Bernoulli distributions or univariate Normals, but multivariate distributions are possible as well [Vergari *et al.*, 2015]. A product node N represents the distribution $P_N(\mathbf{X}_{\phi(N)}) = \prod_{C \in \mathrm{ch}(N)} P_C(\mathbf{X}_{\phi(C)})$. We require product nodes to be *decomposable*, meaning that the scopes of all children of a product node are pairwise disjoint. A sum node N represents the distribution $P_N(\mathbf{X}_{\phi(N)}) = \sum_{C \in \mathrm{ch}(C)} w_C\, P_C(\mathbf{X}_{\phi(C)})$.

We require sum nodes to be *complete*, which means that the scopes of all children of the sum node are identical.

Intuitively, product nodes represent distributions that decompose into independent factors (where decomposability ensures this independence), and sum nodes represent mixture distributions (where completeness ensures that sum nodes represent proper mixtures). More specifically, product nodes represent *local independence*, i.e., independence that holds conditional on latent variables. By definition, the distribution represented by an SPN is the distribution defined by its root node. Figure 1 (left) shows an example of an SPN. Many extensions and generalizations of this general model have been proposed, e.g., models for continuous and hybrid domains [Molina *et al.*, 2018].

Inference. The appealing property of SPNs is that they permit efficient marginal inference, i.e., computing $P(\mathbf{X}' = \mathbf{x}')$ for a subset $\mathbf{X}' \subset \mathbf{X}$ of the RVs. This is possible because summation over the marginalized RVs can be "pushed down" into the leaf nodes of the SPN [Peharz *et al.*, 2015]. Thus, marginal inference reduces to marginalization of the leaves and evaluating the internal nodes of the SPN once. Thus, when marginal inference of the leaf distributions is possible in constant time, marginal inference is linear in the number of nodes of the SPN. This task becomes specifically simple when the leaf distributions are univariate, because the value of marginalized leaves can simply be set to 1.

Learning. Efficient inference in SPNs is a result of decomposability and completeness of the SPN. Thus, a central challenge is to learn SPN structures that satisfy these constraints.

We focus on LearnSPN [Gens and Domingos, 2013], a greedy, top-down structure learning algorithms that creates a decomposable and complete tree-structured SPN. The algorithm constructs the tree recursively in a top-down way. At each step, the algorithm performs an independence test on the dataset, and creates a product node if the data can be split into sets of independent RVs. If no independence is identified, a clustering algorithm is used to partition the data into subsets and a corresponding sum node is created, with weights corresponding to the proportion of data in each cluster. In both cases, the algorithm is recursively called for the corresponding data subsets. When only a single RV is left in a call, a leaf node is created by estimating a univariate, smoothed distribution of that RV. To prevent overfitting, a leaf is also created when the number of instances falls below a pre-defined threshold m. In this case, either a full factorization is assumed, or a leaf representing a tractable multivariate distribution is created, e.g. a Chow-Liu tree [Vergari *et al.*, 2015].

Several other SPN learning algorithms have been proposed, e.g. a Bayesian structure learning algrithm [Trapp *et al.*, 2019], and an algorithm that generates a random (decomposable and complete) SPN structure and then optimize the parameters of that SPN by EM [Peharz *et al.*, 2020].

3 Exchangeability-Aware Sum-Product Networks

We propose Exchangeability-Aware Sum-Product Networks (XSPNs) as novel tractable probabilistic models. They are

based on explicitly modeling distributions over partially exchangeable RVs as leaf distributions of an SPN.

3.1 Finite Exchangeability

Exchangeability of *infinite* sequences of RVs is fundamental in Bayesian statistics. Here, we instead focus on exchangeability of *finite* sequences of RVs, as stated by the following definition.

Definition 1 (Finite Exchangeability). *Let* $X_1, \ldots, X_n$ *be a sequence of RVs with joint distribution* P *and let* $S(n)$ *be the group of permutations acting on* $\{1, \ldots, n\}$. *We call* $X_1, \ldots, X_n$ *exchangeable iff* $P(X_1=x_1, \ldots, X_n=x_n) = P(X_1=x_{\pi(1)}, \ldots, X_n=x_{\pi(n)})$ *for all* $\pi \in S(n)$.

Exchangeable RVs are not necessarily independent, and thus a graphical model representation of a distribution over exchangeable RVs can have high tree-width. Similarly, conventional SPNs do not encode distributions over exchangeable RVs efficiently (see Figure 1). However, as shown below, exchangeability still allows for an efficient representation and inference.

Full exchangeability is a strong assumption that is not necessary for our purposes. Instead, in this paper, we focus on the weaker property of *partial* exchangeability (first introduced by [De Finetti, 1938]), which is based on the notion of sufficient statistics.

Definition 2 (Partial Exchangeability). *Let* $X_1, \ldots, X_n$ *be a sequence of RVs with joint distribution* P, *let* $dom(X)$ *be the domain of* X *and let* $\mathcal{T}$ *be a finite set. The sequence* $X_1, \ldots, X_n$ *is* partially exchangeable *w.r.t. the statistic* $T : dom(X_1) \times \cdots \times dom(X_n) \to \mathcal{T}$ *if*

$$T(\mathbf{x}) = T(\mathbf{x}') \text{ implies } P(\mathbf{x}) = P(\mathbf{x}')$$

for all assignments $\mathbf{x}$ *and* $\mathbf{x}'$.

The statistic T partitions the assignments $\mathbf{x}$ into equivalence classes $S_t = \{\mathbf{x} \mid T(\mathbf{x}) = t\}$ with identical value t of the statistic T and identical probability. Partial exchangeability allows to represent a distribution by $|\mathcal{T}|$ parameters, where parameter w_t is the probability of each assignment $\mathbf{x}$ with $T(\mathbf{x}) = t$. Using this parametrization, the probability of an assignment $\mathbf{x}$ is given by $P(\mathbf{x}) = \sum_{t \in \mathcal{T}} [T(\mathbf{x}) = t] w_t$, where $[\cdot]$ is the indicator function [Diaconis and Freedman, 1980].

In the same vein, partial exchangeability allows for efficient marginal and MAP inference. Let $\mathbf{e}$ be a partial assignment, and let $\mathbf{x} \sim \mathbf{e}$ denote that $\mathbf{x}$ and $\mathbf{e}$ agree on the values of RVs in their intersection. Furthermore, let $S_{\mathbf{e},t} = \{\mathbf{x} \mid T(\mathbf{x}) = t \text{ and } \mathbf{x} \sim \mathbf{e}\}$ be the set of all assignments that are consistent with evidence $\mathbf{e}$ and that correspond to value t of the statistics.

Theorem 1 ([Niepert and Van den Broeck, 2014]). *Let* $X_1, \ldots, X_n$ *be partially exchangeable w.r.t.* T. *If we can efficiently,*

- *for all* $\mathbf{x}$*, evaluate* $P(\mathbf{x})$*, and*
- *for all* $\mathbf{e}$ *and* t*, decide whether there exists an* $\mathbf{x} \in S_{\mathbf{e},t}$ *and if so, construct it,*

then the complexity of MAP inference is polynomial in $|\mathcal{T}|$. *If we can additionally compute* $|S_{\mathbf{e},t}|$ *efficiently, then the complexity of marginal inference is also polynomial in* $|\mathcal{T}|$.

When the conditions of the theorem are satisfied for a statistic T, we call T a tractable statistic. As a simple example, consider binary RVs $X_1, \ldots, X_n$ and the statistic $T^{\#}(\mathbf{x}) = T^{\#}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$, which counts the number of ones in an assignment. For this statistic, the conditions of Theorem 1 are satisfied: For a given $\mathbf{e}$ and t, constructing an arbitrary $\mathbf{x} \in S_{\mathbf{e},t}$ is straightforward and $|S_{\mathbf{e},t}| = \binom{n-n_\mathbf{e}}{t-T(\mathbf{e})}$, where $n_\mathbf{e}$ is the number of values in $\mathbf{e}$.

Exchangeable Variable Models. (EVMs) [Niepert and Domingos, 2014] are tractable probabilistic models based on the notion of partial exchangeability. As basic building blocks, they use distributions over partially exchangeable RVs w.r.t. a statistic T, which are represented by parameters $w_1, \ldots, w_{|\mathcal{T}|}$.

An Exchangeable Variable Model (EVM) is a product of such exchangeable blocks. Specifically, let $\mathcal{X}$ be a partitioning of the RVs $X_1, \ldots, X_n$ into disjoint subsets. An EVM defines a joint distribution via $P(\mathbf{X}) = \prod_{\mathbf{X}_i \in \mathcal{X}} P(\mathbf{X}_i)$, where the factors $P(\mathbf{X}_i)$ are exchangeable blocks, as defined above. Finally, an MEVM is a mixture of such EVMs. Parameter estimation in MEVMs (i.e. estimation of the mixture weights, the independence structure inside each EVM and the parameters of the exchangeable blocks) can be done via Expectation Maximization (EM).

Statistical relational models, like Markov Logic Networks [Richardson and Domingos, 2006] or relational Sum-Product Networks [Nath and Domingos, 2015] are another class of probabilistic models that make use of exchangeability for efficient representation and inference. In these models, a high-level, relational structure which implicitly defines which RVs are exchangeable is defined a priori based on domain knowledge. In contrast, MEVMs (as well as our own contribution introduced below) identify exchangeability of RVs purely from the training data. Tractable inference in statistical relational models can only be guaranteed for certain subclasses of models, and can also become intractable when exchangeability breaks due to asymmetrical evidence [Jaeger and Van den Broeck, 2012].

3.2 Exchangeable Leaf Distributions

In this section, we introduce a variant of SPNs which we call *Exchangeability-Aware Sum-Product Network* (XSPN). An XSPN is an SPN with multivariate leaf distributions $p(X_1, \ldots, X_n)$, where $X_1, \ldots, X_n$ are partially exchangeable w.r.t. a given statistic T. Note that the leaves can have different cardinality and can be exchangeable w.r.t. different statistics T. Specifically, when all leaves are univariate, XSPNs are equivalent to conventional SPNs. Due to the fact that marginal inference in distributions of partially exchangeable RVs is tractable, inference in XSPNs is also tractable.

Corollary 1. *Let* S *be the set of all XSPNs with at most* N *nodes, where the leaf nodes represent distributions over RVs which are partially exchangeable w.r.t. a statistic* T, *which has* $|\mathcal{T}|$ *possible values. When the conditions from Theorem 1 are satisfied for* T, *then time complexity of marginal inference in the XSPNs contained in* S *is polynomial in* $N \cdot |\mathcal{T}|$.

Proof. Marginal inference in XSPNs requires a constant

Algorithm 1 LearnXSPN

1: **Input:** Set of instances D over variables V, minimum number of instances m
2: **Output:** SPN representing a distribution over V
3: **if** $|D| < m$ **then**
4:     **return** Exchangeable distribution estimated from D
5: **else if** $|V| = 1$ **then**
6:     **return** Univariate distribution estimated from D
7: **else if** V are exchangeable w.r.t. a given statistic T **then**
8:     **return** Exchangeable distribution estimated from D
9: **else if** V can be partitioned into independent sets V_j **then**
10:     **return** $\prod_j$ LearnXSPN(D,V_j,m)
11: **else**
12:     Partition D into subsets D_i
13:     **return** $\sum_i \frac{|D_i|}{|D|}$ LearnXSPN(D_i,V,m)
14: **end if**

---

number of evaluations of each inner node, and answering a marginal query at each leaf node [Peharz *et al.*, 2015]. There are at most N leaf nodes, and for each leaf node, inference is polynomial in $|\mathcal{T}|$ (Theorem 1). $\qquad\square$

For example, time complexity of marginal inference in XSPNs where the leaf distributions use the statistic $T^{\#}$ with at most n RVs is in $\mathcal{O}(n\,N)$. XSPNs unify conventional SPNs (with univariate leaf distributions) and MEVMs in a common framework: Conventional SPNs are a special case of XSPNs where all leaf distributions are univariate, and MEVMs are shallow XSPN with a single sum layer and a single product layer.

3.3 Learning XSPNs

Next, we consider structure learning of XSPNs. The learning algorithm is based on LearnSPN (see Section 2) and is shown in Algorithm 1. In addition to the usual LearnSPN scheme, the algorithm tests for the presence of partial exchangeability at each recursive call. When partial exchangeability is not rejected by the test, a leaf representing a distribution over partially exchangeable random variables is created directly, i.e. the algorithm does not recurse in that case.

Note that this approach for creating leaves is different from other approaches that allow for multivariate leaves, e.g. Chow-Liu trees [Vergari *et al.*, 2015]: These other approaches introduce a multivariate leaf as a fallback case, when neither a product node nor a sum node can be created, whereas our algorithm explicitly tests whether the assumptions made in the multivariate leaf distribution are satisfied. Still, multivariate leaves can also be created as a fallback case in our algorithm. Here, it is natural to also use distributions over exchangeable RVs.

We want to emphasize that—in contrast to statistical relational models like Markov Logic Networks or Relational Sum-Product Networks [Nath and Domingos, 2015]—XSPNs do not use any prior knowledge about exchangeability structure in the distribution, but detect exchangeability solely based on the data itself.

In the following, we discuss the additional operations that are required by the algorithm in more detail: First, we show

how the parameters $w_1, \ldots, w_{|\mathcal{T}|}$ of a partially exchangeable distributions can be estimated, and afterwards discuss statistical tests for partial exchangeability.

Parameter Estimation. Let $X_1, \ldots, X_n$ be a sequence of RVs that is partially exchangeable w.r.t. statistic T, and let $\{\mathbf{x}^{(i)}\}_{i=1}^N$ be a set of samples. Recall that a distribution over such RVs can be represented by parameters $w_1, \ldots, w_{|\mathcal{T}|}$, one parameter for each $t \in \mathcal{T}$. The Maximum Likelihood estimate of parameter w_t is given by [Niepert and Domingos, 2014][1]

$$w_t = 1/N \sum_{i=1}^N [T(\mathbf{x}^{(i)}) = t]\,|S_t|^{-1}$$

Intuitively, the estimate makes use of the fact that each partially exchangeable distribution can be seen as a mixture of uniform distributions. Each equivalence class S_t corresponds to a mixture component with weight $1/N \sum_{i=1}^n [T(\mathbf{x}^{(i)}) = t]$. The factor $|S_t|^{-1}$ ensures that w_t represents the probability of each of the assignments in S_t. Note that the complexity of computing $|S_t|$ is polynomial for tractable statistics T. For the counting statistic $T^{\#}$, this factor can even be computed in constant time, as $|S_t| = \binom{n}{t}$ in that case.

Exchangeability Tests. In the following, we discuss the problem of identifying whether the discrete RVs $X_1, \ldots, X_n$ are partial exchangeable, given a set of samples $\{\mathbf{x}^{(i)}\}_{i=1}^N$. A simple option is given by [Niepert and Domingos, 2014], who provide a number of necessary conditions for finite exchangeability. For example, when the RVs $X_1, \ldots, X_n$ are exchangeable, then $\forall i, j : E[X_i] = E[X_j]$.

We propose to use a more general approach: Given a set of samples $\{\mathbf{x}^{(i)}\}_{i=1}^N$, we use Pearson's χ^2 test for goodness of fit to test the null hypothesis that the joint distribution $P(\mathbf{X})$ is partially exchangeable w.r.t. a given statistic T. Specifically, we perform the following steps: (i) estimate the parameters $w_1, \ldots, w_{|\mathcal{T}|}$ w.r.t. the statistic T from the samples $\{\mathbf{x}^{(i)}\}_{i=1}^N$, using the ML estimate above; (ii) for each assignment $\mathbf{x}$, compute its expected frequency (assuming that it is distributed according to $P(\mathbf{x}) = \sum_t [T(\mathbf{x}) = t]\,w_t$) and its empirical frequency, (iii) compute the test statistic of Pearson's χ^2 test, and either reject or not reject the null hypothesis.

Unfortunately, the time complexity of this test grows exponentially with the number of RVs n, because the expected and empirical frequencies need to be computed for all assignments $\mathbf{x}$. Therefore, we propose to approximate the test that works by performing pairwise comparisons. Specifically, for a sequence $X_1, \ldots, X_n$, we test all pairs (X_i, X_j) with $i, j \in \{1, \ldots, n\}$, $i < j$ for partial exchangeability, using the test outlined above. When none of these tests rejects the null hypothesis, we assume that $X_1, \ldots, X_n$ are partially exchangeable.

Pairwise exchangeability is a necessary condition of (full) exchangeability [Schervish, 1995, Proposition 1.12], thus all cases of actual full exchangeability are also identified by the

---

[1]Note that we directly multiply by $|S_t|^{-1}$ when estimating the parameters, whereas [Niepert and Domingos, 2014] multiply by this factor when evaluating $P(\mathbf{x})$.

pairwise approach. We rarely encounter false positives in practice, as illustrated empirically in the supplementary material.

The pairwise test needs to perform $\mathcal{O}(n^2)$ χ^2 tests, and for each test, compute a Maximum Likelihood parameter estimate. For the statistic $T^{\#}$, parameter estimation can be done in $\mathcal{O}(N)$ time (where N is the number of samples), and thus the time complexity of all pairwise tests is in $\mathcal{O}(N\,n^2)$. In practice, this test only has a small effect on overall runtime of LearnXSPN, which is dominated by clustering.

4 Experimental Evaluation

We evaluated XSPNs on three types of datasets: Seven synthetic datasets that were constructed to contain (contextually) exchangeable RVs, four real-world datasets consisting of multiple interchangeable parts (where partial exchangeability arises naturally), and 20 commonly used benchmark datasets. We investigated probability estimation as well as classification performance of XSPNs, compared to conventional SPNs and additional baseline models.

4.1 Probability Estimation

Data. Four of the synthetic datasets were created by following [Niepert and Domingos, 2014]: We sampled uniformly from $\{0, 1\}^{100}$, and then only kept samples that satisfy certain constraints on the number of ones in the sample. Two datasets (MEVM-s and MEVM-l) were sampled from MEVM models. The conference dataset was sampled from a Markov Logic Network that describes decision-making of people attending or not attending a conference, as introduced in Example 1.

Additionally, we evaluated our approach on four real-world density estimation tasks. The *exams* dataset consists of exam participation information of 487 business students that started their studies in fall term 2019 at the University of Mannheim. 58 courses were attended by at least 10 of these students between fall 2019 and fall 2021. Each binary RV represents participation in one of these courses, and each example represents a student. The *senate* dataset contains all 720 roll call votes in the Senate of the 116th United States Congress, taken from [Lewis *et al.*, 2021]. We only kept votes of the 98 senators that participated in the majority of the votes. Each binary RV represents the votes of a senator, and each example represents a ballot. A similar procedure was applied to the votes from the House of Representatives of the 116th US Congress (*house* dataset) and the 17th German federal parliament (*bundestag* dataset, data taken from [Bergmann *et al.*, 2018]).

Finally, for completeness, we also evaluated SPNs and XSPNs on 20 benchmark datasets that are commonly used for comparing SPN learning algorithms [Gens and Domingos, 2013]. For a more detailed description of the datasets, we refer to the supplementary material.

Experiments. We compared the following models: SPNs with Chow-Liu trees as leaf distributions trained via LearnSPN [Vergari *et al.*, 2015] (which are closest to XSPNs as they also use multivariate leaf distributions); our XSPNs trained via LearnXSPN; MEVMs (as shallow variants of XSPNs) trained via EM [Niepert and Domingos, 2014]; and
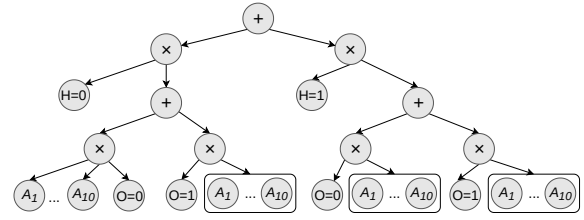

Figure 2: XSPN learned for the conference dataset. The XSPN faithfully represents the true distribution underlying the MLN specification: For fixed HotTopic and Overflow, all of the Attends random variables are exchangeable—except for the case where HotTopic = Overflow = 0, in which case all of the Attends variables are independent.

Masked Autoencoders for Distribution Estimation (MADEs) [Germain *et al.*, 2015]. MADEs do not provide tractable marginal inference and are thus not directly comparable to (X)SPNs, but are used as strong baseline density estimators here. We do not consider methods for modeling distributions over sets [Yang *et al.*, 2020], because they assume full exchangeability and are thus not suited for the cases investigated here. Our implementation[2] of XSPNs is based on the SPFlow library [Molina *et al.*, 2019] for Python.

In all SPN models, we used the g-test as independence test and clustered instances via EM for Gaussian Mixture Models. As proposed by [Vergari *et al.*, 2015], we limited the number of child nodes of each sum and product node to 2. For XSPN leaves, we used the statistic $T^{\#}$ introduced in Section 3.1. We performed an exhaustive grid search of the SPN hyperparameters. Specifically, we varied the g-test threshold values $\rho \in \{5, 15\}$, the minimum number of instances $m \in \{20, 200\}$, and the significance level of the χ^2-test for exchangeability $p \in \{0.05, 0.1, 0.2, 0.4\}$. For all models, we used Laplace smoothing with $\alpha = 0.1$. Chow-Liu tree leaf learning failed in some cases using SPFlow, in which case we used fully factorized leaf distributions. Parameter settings for the MEVM and MADE models can be found in the supplementary material.

Results. Table 1 shows the experimental results. For the 7 synthetic datasets, the XSPNs outperform all other baseline models (except for being on-par with MADE for some datasets). For these datasets, XSPNs can often learn the *true* distribution. As an example, consider Figure 2, which shows the XSPN learned for the conference dataset. This XSPN represents the true dependency structure in the dataset. Note that LearnXSPN was able to discover this structure completely bottom-up from the data without any prior knowledge about the domain, as usually required for top-down relational learning approaches. For these datasets, XSPNs also need substantially fewer parameters: For example, the XSPN shown in Figure 2 has 52 parameters, while the corresponding SPN (with identical hyperparameters) has 363 parameters.

For the real-world datasets (exam, senate, house, bundestag), XSPNs also generally outperform the baseline models. Notably, the difference between performance of XSPN and the other models is specifically pronounced for bundestag

---

[2]Available at https://github.com/stefanluedtke/XSPNFlow

	dataset	SPN	XSPN	MEVM	MADE
synthetic	threshold	-68.035	**-67.318**	-67.408	-67.628
	exact	-69.327	**-67.706**	-67.906	-69.474
	parity	-69.327	**-68.623**	-69.368	-69.505
	counting	-69.323	**-69.319**	-67.936	-69.468
	MEVM-s	-12.091	**-11.889**	-12.524	-11.885
	MEVM-l	87.795	**-84.858**	-94.309	-81.835
	conference	-5.936	**-5.907**	-5.91	-5.541
real	senate	-20.778	**-19.663**	-21.107	-20.972
	house	-86.517	**-80.032**	-83.472	-83.277
	bundestag	-228.459	**-103.763**	-106.42	-161.727
	exam	-6.952	-6.72	-7.066	-12.308
benchmark	nltcs	-6.059	-6.06	-6.04*	-6.04
	plants	-13.035	**-13.009**	-14.86*	-12.32
	webkb	-156.444	-156.65	-157.21*	-149.59
	msnbc	-6.043	-6.043	-6.23*	-6.06
	kdd	-2.151	-2.15	-2.13*	-2.07
	audio	-40.823	-40.82	-40.63*	-38.95
	jester	-53.811	-53.806	-53.22*	-52.23
	netflix	-58.213	-58.208	-57.84*	-55.16
	msweb	-9.882	-9.884	-9.96*	-9.59
	book	-35.079	**-34.937**	-34.63*	-33.95
	accidents	-28.957	**-28.937**	-38.258	-26.42
	dna	-81.53	-81.567	-98.34	-82.77
	kosarek	-10.77	**-10.741**	-10.997	-
	pumsb	-23.457	**-23.375**	-36.396	-22.3
	retail	-11.019	**-10.967**	-10.897	-10.81
	movie	-52.373	**-51.702**	-52.015	-48.7
	bbc	-252.994	-251.949	-252.023	-242.4
	ad	-18.996	**-15.747**	-32.359	-13.65
	20ng	-153.503	-153.545	-152.69*	-153.18
	reut.-52	-84.777	**-84.651**	-86.98*	-82.8

Table 1: Test log likelihoods for all models and datasets. For the SPN and XSPN, results that are significantly better than the other (X)SPN model are printed in bold (paired t-test, $p < 0.05$). * Results taken from [Niepert and Domingos, 2014].

dataset	SPN	XSPN	MEVM	SVM	XGBoost
counting	0.758	1.000	0.793	0.796	0.793
exact	0.975	1.000	0.977	0.979	0.978
parity	0.493	1.000	0.507	0.605	0.502
threshold	0.983	1.000	0.987	0.982	0.983
conference	0.854	0.854	0.851	0.850	0.852
exam	0.986	0.986	0.971	0.997	0.971
senate	0.983	0.992	0.958	0.974	0.967
house	0.928	0.967	0.908	0.986	0.954
bundestag	0.951	0.951	0.610	0.941	0.951

Table 2: Test accuracies for the classification experiments.

is to predict the votes of one of the representatives, given the votes of all other representatives. More details on the classification tasks can be found in the supplementary material.

The test accuracies are shown in Table 2. For the first four synthetic datasets, the XSPN can learn the true distribution of the data (conditional on the class, all RVs are fully exchangeable), and thus classifies all test samples correctly. The conventional SPN and the baseline classifiers, on the other hand, cannot learn the underlying structure of the data appropriately; their accuracies lie in the range of the prior probability of the majority class. For the conference and exam data, SPN and XSPN achieve the same performance, being competitive with the baseline classifiers.

In the vote prediction tasks, the XSPN achieves higher accuracies than the SPN for two datasets (senate, house) and the same accuracy for the bundestag dataset. Furthermore, even though (X)SPNs are not primarily designed for classification tasks, the accuracy of the XSPN model is competitive with the baseline classifiers.

5 Discussion and Conclusion

We proposed an extension of SPNs which allows to efficiently handle distributions over partially exchangeable RVs, as well as a structure learning algorithm for these models. The learning algorithm constructs a deep model with efficient representations of exchangeable sequences, without needing any prior knowledge about exchangeability relations among the RVs. XSPNs achieve state-of-the-art density estimation performance for datasets containing repeated, interchangeable parts, and are competitive with SPNs in general domains.

Our learning algorithm is based on the structure learning algorithm of [Vergari et al., 2015]. Future research will focus on integrating means to detect and represent exchangeability in other SPN learning algorithms, e.g. Bayesian learning [Trapp et al., 2019]. A further interesting direction for future work is to generalize our approach to other statistics (e.g. related to exchangeable decompositions [Niepert and Van den Broeck, 2014]) and continuous and hybrid domains.

Acknowledgements

data. We suspect that the inductive bias of XSPNs towards exchangeability is able to prevent overfitting for this dataset, which contains only few samples in relation to the number of variables. Overall, XSPNs achieve state-of-the-art results for datasets consisting of multiple, interchangeable parts.

The 20 benchmark datasets do not explicitly encode exchangeability, thus we did not expect a large benefit of XSPNs compared to conventional SPNs with Chow-Liu tree leaves. Interestingly, XSPNs achieve significantly (although only slightly) higher test log likelihoods than SPNs for 8 of the 20 datasets. Thus, XSPNs can also be used in such general domains without loss of performance, compared to SPNs.

4.2 Classification

Additionally, we evaluated the classification performance of XSPNs by training a separate SPN or XSPN for each class y to represent $P(\mathbf{x} \,|\, y)$, and computing the class posterior via $P(y \,|\, \mathbf{x}) \propto P(\mathbf{x} \,|\, y) \, P(y)$. As baselines, we used Support Vector Machines (SVMs) and gradient boosted trees (XGBoost). The classification tasks are based on the datasets described in Section 4.1, where one of the RVs is selected as classification target. For example, for the voting data, the task

References

[Bergmann *et al.*, 2018] Henning Bergmann, Stefanie Bailer, Tamaki Ohmura, Thomas Saalfeld, and Ulrich Sieberer. BTVote Voting Behavior. https://dataverse.harvard.edu/dataverse/btvote, 2018. Accessed: 2022-05-09.

[De Finetti, 1938] Bruno De Finetti. *Sur la condition d'equivalence partielle*. 1938. English translation in: Studies in Inductive Logic and Probability 2, 1980.

[Delalleau and Bengio, 2011] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. *Advances in neural information processing systems*, 24:666–674, 2011.

[Diaconis and Freedman, 1980] Persi Diaconis and David Freedman. De finetti's generalizations of exchangeability. *Studies in Inductive Logic and Probability II*, 1980.

[Gens and Domingos, 2013] Robert Gens and Predro Domingos. Learning the structure of sum-product networks. In *International conference on machine learning*, 2013.

[Germain *et al.*, 2015] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 2015.

[Jaeger and Van den Broeck, 2012] Manfred Jaeger and Guy Van den Broeck. Liftability of probabilistic inference: Upper and lower bounds. In *Proceedings of the 2nd international workshop on statistical relational AI*, 2012.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

[Larochelle and Murray, 2011] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Artificial Intelligence and Statistics*, pages 29–37, 2011.

[Lewis *et al.*, 2021] Jeffrey B. Lewis, Keith Poole, Howard Rosenthal, Adam Boche, Aaron Rudkin, and Luke Sonnet. Voteview: Congressional roll-call votes database. https://voteview.com, 2021. Accessed: 2022-05-09.

[Liang *et al.*, 2017] Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Uncertainty in Artificial Intelligence*, 2017.

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

[Molina *et al.*, 2019] 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 network. https://github.com/SPFlow/SPFlow, 2019. Accessed: 2022-05-09.

[Nath and Domingos, 2015] Aniruddh Nath and Pedro M Domingos. Learning relational sum-product networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Niepert and Domingos, 2014] Mathias Niepert and Pedro Domingos. Exchangeable variable models. In *International Conference on Machine Learning*, 2014.

[Niepert and Van den Broeck, 2014] Mathias Niepert and Guy Van den Broeck. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[Peharz *et al.*, 2015] Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. On theoretical properties of sum-product networks. In *Artificial Intelligence and Statistics*, 2015.

[Peharz *et al.*, 2020] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, 2020.

[Poon and Domingos, 2011] Hoifung Poon and Pedro Domingos. Sum-product networks: a new deep architecture. In *Uncertainty in Artificial Intelligence*, 2011.

[Rahman *et al.*, 2014] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 630–645. Springer, 2014.

[Rezende and Mohamed, 2015] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[Schervish, 1995] Mark J Schervish. *Theory of statistics*. Springer, 1995.

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

[Vergari *et al.*, 2015] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358. Springer, 2015.

[Yang *et al.*, 2020] Mengjiao Yang, Bo Dai, Hanjun Dai, and Dale Schuurmans. Energy-based processes for exchangeable data. In *International Conference on Machine Learning*, pages 10681–10692. PMLR, 2020.