The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains

Alejandro Molina*
alejandro.molina@tu-dortmund.de
TU Dortmund, Germany

Antonio Vergari*
antonio.vergari@uniba.it
University of Bari, Italy

Nicola Di Mauro
nicola.dimauro@uniba.it
University of Bari, Italy

Sriraam Natarajan
natarasr@indiana.edu
Indiana University, USA

Floriana Esposito
floriana.esposito@uniba.it
University of Bari, Italy

Kristian Kersting
kersting@cs.tu-darmstadt.de
TU Darmstadt, Germany

Abstract

While all kinds of mixed data—from personal data, over panel and scientific data, to public and commercial data—are collected and stored, building probabilistic graphical models for these hybrid domains becomes more difficult. Users spend significant amounts of time in identifying the parametric form of the random variables (Gaussian, Poisson, Logit, etc.) involved and learning the mixed models. To make this difficult task easier, we propose the first trainable probabilistic deep architecture for hybrid domains that features tractable queries. It is based on Sum-Product Networks (SPNs) with piecewise polynomial leaf distributions together with novel nonparametric decomposition and conditioning steps using the Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient. This relieves the user from deciding a-priori the parametric form of the random variables but is still expressive enough to effectively approximate any distribution and permits efficient learning and inference. Our experiments show that the architecture, called Mixed SPNs, can indeed capture complex distributions across a wide range of hybrid domains.

Introduction

Machine learning has achieved considerable successes in recent years, and an ever-growing number of disciplines rely on it. Data is now ubiquitous, and there is great value in understanding the data, building probabilistic models and making predictions with them. However, in most cases, this success crucially relies on the data scientists to posit the right parametric form of the probabilistic model underlying the data, to select a good algorithm to fit their data, and finally to perform inference on it. These can be quite challenging even for experts and often go beyond non-experts' capabilities, specifically in hybrid domains, consisting of mixed—continuous, discrete and/or categorical—statistical types. Building a probabilistic model that is both expressive enough to capture complex dependencies among random variables of different types as well as allows for effective learning and efficient inference is still an open problem.

More precisely, most existing graphical models for hybrid domains—also called *mixed models*—are limited to particular combinations of variables of parametric forms such as the Gaussian-Ising mixed model (Lauritzen and Wermuth 1989), where there are Gaussian and multinomial random variables, and the continuous variables are conditioned on all configurations of the discrete variables. Unfortunately, inference in this Gaussian-Ising mixed graphical model scales exponentially with the number of discrete variables, and only recently, 3-way dependencies have been realized (Cheng et al. 2014). Therefore it is not surprising that hybrid Bayesian networks (HBNs) have restricted their attention to simpler parametric forms for the conditional distributions such as conditional linear Gaussian models (Heckerman and Geiger 1995). While extensions based on copulas aim to provide more flexibility (Elidan 2010), selecting the best parametric copula distribution for each application requires significant engineering effort. Probably the most recent approach is Manichean graphical models (Yang et al. 2014), and we refer to this paper for an excellent and current overview of mixed graphical models. Manichean models specify that each of the conditional distributions is a member of a possibly different univariate exponential family. Although indeed more flexible than Gaussian-Ising mixed models, Manichean models are still demanding, in particular when it comes to inference. Alternatively, one may make a piecewise approximation to continuous distributions (Shenoy and West 2011). In their purest form, piecewise constant functions are often adopted in the form of histograms or staircase functions, and more expressive approximations comprise mixtures of truncated polynomials (Langseth et al. 2012) and exponentials (Moral, Rumi, and Salmerón 2001). This has resulted in a number of novel inference approaches for hybrid domains (Sanner and Abbasnejad 2012; Belle, Passerini, and Van den Broeck 2015; Belle, Van den Broeck, and Passerini 2015; Morettin, Passerini, and Sebastiani 2017). Although expressive, learning these non-parametric models does not scale.

To overcome the difficultness of mixed probabilistic graphical modeling and inspired by the successes of deep models, we introduce Mixed Sum-Product Networks (MSPNs). They are a general class of mixed probabilistic models that, by combining Sum-Product Networks (Poon and Domingos 2011) and piecewise polynomials, allow for a broad range of exact and tractable inference without making distributional assumptions. Learning MSPNs from data, however, requires different decomposition and conditioning

*Contributed equally

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

steps for Sum-Product Networks (SPNs) tailored towards nonparametric distributions. Providing them based on the Rényi Maximum Correlation Coefficient (Lopez-Paz, Hennig, and Schölkopf 2013)—the first application of it to learning sum-product networks—via a series of variable transformations is our main technical contribution. This then naturally results in the first automated tool for learning multivariate distributions over hybrid domains without requiring users to decide the parametric form of random variables or their dependencies, yet enabling them to answer complex probabilistic queries efficiently on tasks previously unfeasible by classical mixed models.

We proceed as follows. We start off by reviewing SPNs. Afterwards, we introduce MSPNs and show how to learn tree-structured MSPNs from data using the Rényi Maximum Correlation Coefficient. Before concluding, we present our experimental evaluation.

Sum-Product Networks (SPNs)

Recent years have seen a significant interest in tractable probabilistic representations such as Arithmetic Circuits (ACs), see (Choi and Darwiche 2017) for a discussion. In particular, SPNs, an instance of ACs, are deep probabilistic models that can represent high-treewidth models (Zhao, Melibari, and Poupart 2015) and facilitate *exact* inference for a range of queries in time *polynomial* in the network size (Poon and Domingos 2011; Bekker et al. 2015).

Definition of SPNs: Formally, an SPN is a rooted directed acyclic graph, comprising *sum*, *product* or *leaf* nodes. The scope of an SPN is the set of random variables appearing on the network. An SPN can be defined recursively as follows: (1) a tractable univariate distribution is an SPN; (2) a product of SPNs defined over different scopes is an SPN; and (3), a convex combination of SPNs over the same scope is an SPN. Thus, a product node in an SPN represents a factorization over independent distributions defined over different random variables, while a sum node stands for a mixture of distributions defined over the same variables. From this definition, it follows that the joint distribution modeled by such an SPN is a valid probability distribution, i.e., each complete and partial evidence inference query produces a consistent probability value (Poon and Domingos 2011; Peharz et al. 2015). This also implies that we can construct multivariate distributions from simpler univariate ones. Furthermore, any node in the network could be replaced by any tractable multivariate distribution over the same scope, obtaining still a valid SPN.

Tractable Inference in SPNs: To answer probabilistic queries in an SPN, we evaluate the nodes starting at the leaves. Given some evidence, the probability output of querying leaf distributions is propagated bottom up. For product nodes, the values of the children nodes are multiplied and propagated to their parents. For sum nodes, instead, we sum the weighted values of the children nodes. The value at the root indicates the probability of the asked query. To compute marginals, i.e., the probability of partial configurations, we set the probability at the leaves for those variables to 1 and then proceed as before. Conditional probabilities can then be computed as the ratio of partial configurations. To compute MPE states, we replace sum by max nodes and then evaluate the graph first with a bottom-up pass, but instead of weighted sums, we pass along the weighted maximum value. Finally, in a top-down pass, we select the paths that lead to the maximum value, finding approximate MPE states (Poon and Domingos 2011). All these operations traverse the tree at most twice and therefore can be achieved in linear time w.r.t. the size of the SPN.

Learning SPNs: While it is possible to craft a valid SPN structure by hand, doing so would require domain knowledge and weight learning afterwards (Poon and Domingos 2011). Here, we focus on a top-down approach (Gens and Domingos 2013) that directly learns both the structure and weights of (tree) SPNs at once.

It uses three steps: (1) base case, (2) decomposition and (3) conditioning. In the base case, if only one variable remains, the algorithm learns a univariate distribution and terminates. In the decomposition step, it tries to partition the variables into independent components $V_j \subset \mathbf{V}$ such that $P(\mathbf{V}) = \prod_j P(V_j)$ and recurses on each component, inducing a product node. If both the base case and the decomposition step are not applicable, then training samples are partitioned into clusters (conditioning), inducing a sum node, and the algorithm recurses on each cluster.

This scheme for learning tree SPNs has been instantiated for several well-known distributions with parametric forms. Conditioning for Gaussians can be realized using hard clustering with EM or K-means (Gens and Domingos 2013; Rooshenas and Lowd 2014). For Poissons, mixtures of Poisson Dependency Networks have been proven successful (Molina, Natarajan, and Kersting 2017). For the decomposition step, one typically employs pairwise independence tests with some associated independence score ρ. For categorical variables, Gens and Domingos (2013) proposed to use the G-test, and Rooshenas and Lowd (2014) a pairwise mutual information test. For variables of the generalized linear model family, Molina *et al.* (2017) proposed the use of parameter instability tests based on generalized M-fluctuation processes. Then, one creates an undirected graph where there is an edge between random variables V_i and V_j if the value $\rho(V_i, V_j)$ passes a threshold of significance α. That is, the decomposition step equals to partitioning the graph into its connected components. It is rejected if there is only a single connected component.

Mixed Sum-Product Networks (MSPNs)

Unfortunately, all previous decomposition and conditioning approaches for SPNs are only suitable for multivariate distributions of known parametric form: categorical, binomial, Gaussian and Poisson distributions (Poon and Domingos 2011; Vergari, Di Mauro, and Esposito 2015; Molina, Natarajan, and Kersting 2017). To model hybrid domains without making parametric assumptions, one has to introduce new conditioning and decomposition approaches tailored towards mixed models.

Rényi Decomposition: We approach the problem of seeking independent subsets of random variables of mixed but unknown types as a dependency discovery problem. Al-

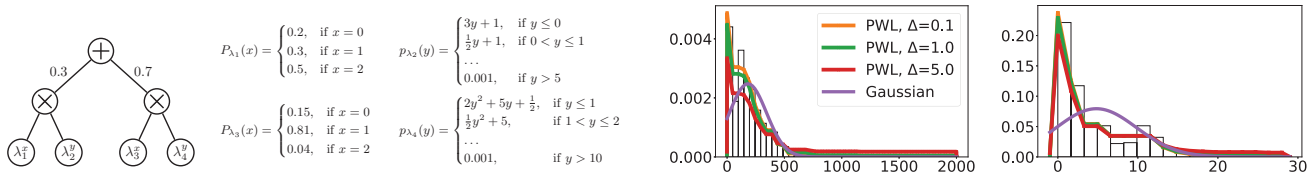

Figure 1: Mixed Sum-Product Networks (MSPNs). From left to right: **(left)** An MSPN representing a mixture over random variables x (discrete) and y (continuous). **(middle)** In each leaf λ_i an MSPN approximates the univariate distribution as a piecewise polynomial. In our experiments we employ piecewise linear models **(right)**. Shown are fitted distributions on the UCI Australian dataset, see experiment section. Shown are the empirical distributions (histograms), piecewise linear approximations using isotonic regression (PWL) with different smoothing values (Δ), and superimposed maximum likelihood Gaussians. This shows that piecewise polynomials are expressive enough to effectively approximate continuous distributions, often better than the classical Gaussian assumption. Within an MSPN, they permit efficient learning and inference. (Best viewed in color)

fred Rényi (1959) argued that a measure of maximum dependence $\rho^* : V_i \times V_j \to [0, 1]$ between random variables V_i and V_j should satisfy several fundamental properties, such as symmetry, transformation invariance, and it should also hold that $\rho^*(V_i, V_j) = 0$ iff V_i and V_j are statistically independent. He also showed the Hirschfeld-Gebelein-Renyi (HGR) Maximum Correlation Coefficient due to Gebelein (1941) to satisfy all these properties. Recently, Lopez-Paz, Hennig, and Schölkopf (2013) provided a practical estimator for the HGR ρ^*, the randomized dependency coefficient (RDC). The RDC is appealing for hybrid domains because it can be applied to both multivariate, continuous and discrete random variables. Also, its $\mathcal{O}(M \log M)$ running time, with M being the number of instances, makes it one of the fastest non-linear dependency measures. The general idea behind the RDC is to look for linear correlations between the representations of two random samples that have undergone a series of non-linear transformations. The two samples are deemed statistically independent iff the transformed samples are linearly uncorrelated. This is the same reasoning behind the adoption of higher space projections for the kernel-trick in classification and the stacking of representations in deep architectures.

Specifically, consider two random samples $\mathcal{D}_{V_i} = \{v_i^m | v_i^m \sim V_i\}_{m=1}^M$ and $\mathcal{D}_{V_j} = \{v_j^m | v_j^m \sim V_j\}_{m=1}^M$ drawn from variables V_i and V_j, we decide that V_i and V_j are independent iff $\rho(\mathcal{D}_{V_i}, \mathcal{D}_{V_j}) = 0$, where ρ is the RDC. Instead of operating directly on $\mathcal{D}_{V_i}$ and $\mathcal{D}_{V_j}$, and in order to achieve invariance against scaling and shifting data transformations, we first compute their *empirical copula transformations* (Póczos, Ghahramani, and Schneider 2012), $\mathcal{C}_{V_i}$ and respectively $\mathcal{C}_{V_j}$, in the following way:

$$\mathcal{C}_{V_i} = \left\{ \frac{1}{M} \sum_{r=1}^{M} \mathbb{1}\{v_i^r \leq v_i^m\} \middle| v_i^m \in \mathcal{D}_{V_i} \right\}_{m=1}^{M} \quad (1)$$

Then, we apply a random linear projection on the obtained samples to a k-dimensional space, finally passing them through a non-linear function σ. We compute:

$$\phi(\mathcal{C}_{V_i}) = \sigma(\mathbf{w} \cdot \mathcal{C}_{V_i}^T + b), (\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k \times k}) \quad (2)$$

for the first sample, an equivalent transformation yields $\phi(\mathcal{C}_{V_j})$. Note that $\mathbf{w} \in \mathbb{R}^{k \times 1}$, $b \in \mathbb{R}$ and that random sampling $\mathbf{w}$ from a zero-mean k-dimensional Gaussian is analogous to the use of a Gaussian kernel (Rahimi and Recht 2009). We choose $k = 20$, σ to be sine function and $s = \frac{1}{6}$ as both have proven to be reasonable empirical heuristics, see (Lopez-Paz, Hennig, and Schölkopf 2013). Lastly, we compute the canonical correlations (CCA) ρ^2 for $\phi(\mathcal{C}_{V_i})$ and $\phi(\mathcal{C}_{V_j})$ as the solutions for the following eigenproblem:

$$\begin{pmatrix} 0 & \Sigma_{ii}^{-1}\Sigma_{ij} \\ \Sigma_{jj}^{-1}\Sigma_{ji} & 0 \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \rho^2 \begin{pmatrix} \beta \\ \gamma \end{pmatrix}, \quad (3)$$

where the covariance block matrices involved are:

$$\Sigma_{ij} = \text{cov}(\phi(\mathcal{C}_{V_i}), \phi(\mathcal{C}_{V_j})), \Sigma_{ji} = \text{cov}(\phi(\mathcal{C}_{V_j}), \phi(\mathcal{C}_{V_i})),$$
$$\Sigma_{ii} = \text{cov}(\phi(\mathcal{C}_{V_i}), \phi(\mathcal{C}_{V_i})), \Sigma_{jj} = \text{cov}(\phi(\mathcal{C}_{V_j}), \phi(\mathcal{C}_{V_j})).$$

In the end, the actual value for the RDC coefficient is the largest canonical correlation coefficient:

$$\text{RDC}(V_i, V_j) = \sup_{\beta,\gamma} \rho(\beta^T \phi(\mathcal{C}_{V_i}), \gamma^T \phi(\mathcal{C}_{V_j})). \quad (4)$$

This RDC pipeline goes through a series of data transformations, which constitutes the basis of our decomposition procedure, cf. Alg. 1. We note that all the transformations presented so far are easily generalizable to the multivariate case (Lopez-Paz, Hennig, and Schölkopf 2013). We are applying these multivariate versions both when performing conditioning on multivariate samples (see below) and when we deal with decomposing categorical random variables. Since Eq. 1 is not well defined for categorical data, to treat them in the same way as continuous and discrete data, we proceed as follows. First we perform a one hot encoding transformation for each categorical random variables V_c, obtaining a multivariate binary random variable $\mathbf{B}_{V_c}$. Then, we apply Eq. 1 to each column $\mathbf{B}_{V_c}$ independently, obtaining the matrix $\mathcal{C}_{\mathbf{B}_{V_c}}$. This way we are preserving all the modalities of V_c. Finally, we apply the generalized version of Eq. 2 and Eq. 3 to the multivariate case.

We are looking for the RDC to be zero in case of independent random variables and apply thresholding approach on the adjacency graph induced by dependencies (see the previous section).

Rényi Conditioning: The task of clustering hybrid data samples depends on the choice of the metric space, which in turn, typically depends on the parametric assumptions made

Algorithm 1 splitFeaturesRDC ($\mathcal{D}$, α)

Input: samples $\mathcal{D} = \{\mathbf{v}^m = (v_1^m, \ldots, v_N^m) | \mathbf{v}^m \sim \mathbf{V}\}_{m=1}^M$ over a set of random variables $\mathbf{V} = \{V_1, \ldots, V_N\}$; α: threshold of significance
Output: a feature partition $\{\mathcal{P}_\mathcal{D}\}$
for each $V_i \in \mathbf{V}$ **do**
  $\mathcal{C}_{V_i} \leftarrow \left\{ \frac{1}{M} \sum_{r=1}^M \mathbb{1}\{v_i^r \leq v_i^m\} \middle| v_i^m \in \mathcal{D}_{V_i} \right\}_{m=1}^M$
  $(\mathbf{w_i}, b_i) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k\times k})$
  $\phi(\mathcal{C}_{V_i}) \leftarrow \sin(\mathbf{w_i} \cdot \mathcal{C}_{V_i}^T + b_i)$
$\mathcal{G} \leftarrow$ Graph($\{\}$)
for each $V_i, V_j \in \mathbf{V}$ **do**
  $c_{i,j} \leftarrow \mathsf{CCA}(\phi(\mathcal{C}_{V_i}), \phi(\mathcal{C}_{V_j}))$
  if $c_{i,j} > \alpha$ **then**
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{(i,j)\}$
return ConnectedComponents($\mathcal{G}$)

Algorithm 2 clusterSamplesRDC ($\mathcal{D}$)

Input: samples $\mathcal{D} = \{\mathbf{v}^m = (v_1^m, \ldots, v_N^m) | \mathbf{v}^m \sim \mathbf{V}\}_{m=1}^M$ over a set of random variables $\mathbf{V} = \{V_1, \ldots, V_N\}$
Output: a data partition $\{\mathcal{P}_\mathcal{D}\}$
$\mathcal{C}_{V_i} \leftarrow \left\{ \frac{1}{M} \sum_{r=1}^M \mathbb{1}\{v_i^r \leq v_i^m\} \middle| v_i^m \in \mathcal{D}_{V_i} \right\}_{m=1}^M$
$(\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_s, s\mathbf{I}_{k\times k})$
$\phi(\mathcal{C}_{V_i}) \leftarrow \sin(\mathbf{w} \cdot \mathcal{C}_{V_i}^T + b)$
$\mathcal{E} \leftarrow \{\phi(\mathcal{C}_{V_1}), \ldots, \phi(\mathcal{C}_{V_N})\}$
return KMeans($\mathcal{E}, 2$)

for each variable. Consider e.g. the popular choice of K-Means using the Euclidean metric. It makes a Gaussian assumption and therefore is not principled for categorical data.

To eliminate the reliance on knowing the type, we propose to cluster multivariate hybrid samples after the RDC pipeline has processed them. Not only does the series of non-linear transformations produce a feature space in which clusters may be more easily separable, but no distributional assumptions are required. More formally, given a set of samples $\mathcal{D}$ over RVs $\mathbf{V}$ we split it into a sample partitioning $\mathcal{P}_\mathcal{D} = \{\mathcal{D}_c\}_{c=1}^C, \bigcup_{c=1}^C \mathcal{D}_C = \mathcal{D}$, and $\mathcal{D}_q \cap \mathcal{D}_r = \emptyset, \forall \mathcal{D}_q, \mathcal{D}_r \in \mathcal{P}_\mathcal{D}$. The weights for the convex combination on the sum nodes are estimated as the proportions of the data belonging to each cluster, i.e., $w_c = \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$. The procedure is sketched in Alg. 2. First, we transform every feature V_i in $\mathcal{D}$ using Eq 2: $\mathcal{E} = \{\phi(\mathcal{D}_{V_n}) | \mathcal{D}_{V_n}\}_{n=1}^N$. Then, all our features are projected into a new k-dimensional non-linear space. In this new space, we can safely apply now K-Means to obtain c clusters. In Alg. 2, we set $c = 2$ as this generally leads to deeper networks (Vergari, Di Mauro, and Esposito 2015).

Nonparametric Univariate Leave Distributions: Finally, to be fully type agnostic, i.e., to realize MSPNs, we adopt piecewise polynomial approximations of the univariate leaf densities. The simplest and most straightforward approximation we consider are piecewise constant functions, i.e. histograms. More precisely, we adopt the scheme proposed in (Rozenholc, Mildenberger, and Gather 2010) offering an adaptive binning, i.e. with irregular intervals, that is learned from data by optimizing a penalized likelihood function. This allows MSPNs to model both multimodal and skewed univariate distributions without further assumptions. We apply Laplacian smoothing by a factor Δ to cope with unseen values and the natural overfitting of histograms.

Algorithm 3 LearnMSPN ($\mathcal{D}$, Δ, η, α)

Input: samples $\mathcal{D} = \{\mathbf{v}^m = (v_1^m, \ldots, v_N^m) | \mathbf{v}^m \sim \mathbf{V}\}_{m=1}^M$ over a set of random variables $\mathbf{V} = \{V_1, \ldots, V_N\}$; η: minimum number of instances to split; Δ: histogram smoothing factor; α: threshold of significance
Output: an MSPN S encoding a joint pdf over $\mathbf{V}$ learned from $\mathcal{D}$
if $|\mathbf{V}| = 1$ **then**
  $\{\mathcal{D}_c\}_{c=1}^C \leftarrow$ clusterSamplesRDC($\mathcal{D}$)
  if $C > 1$ **then**
    $S \leftarrow \sum_{i=1}^C \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$ LearnMSPN($\mathcal{D}_c, \Delta, \eta$)
  else
    $S \leftarrow$ LearnIsotonicLeaf($\mathcal{D}, \Delta$)
else if $|\mathcal{D}| < \eta$ **then**
  $S \leftarrow \prod_{n=1}^{|\mathbf{V}|}$ LearnMSPN($\{v_n^m | v_n^m \sim V_n\}_{m=1}^M, \Delta, \eta$)
else
  $\{\mathbf{V}_c\}_{c=1}^C \leftarrow$ splitFeaturesRDC($\mathcal{D}, \alpha$)
  if $C > 1$ **then**
    $\mathcal{D}_c \leftarrow \{\mathbf{v}_c^m | \mathbf{v}_c^m \sim \mathbf{V}_c\}_{m=1}^M$
    $S \leftarrow \prod_{c=1}^C$ LearnMSPN($\mathcal{D}_c, \Delta, \eta$)
  else
    $\{\mathcal{D}_c\}_{c=1}^C \leftarrow$ clusterSamplesRDC($\mathcal{D}$)
    $S \leftarrow \sum_{i=1}^C \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$ LearnMSPN($\mathcal{D}_c, \Delta, \eta$)
return S

Indeed, by increasing the degree of leaf polynomial approximations, one can favor more expressive models. To balance between the complexity of learning resp. inference and expressiveness, however, we restrain to piecewise linear approximations. We reframe the unsupervised task of estimating the density of univariate leaf distributions into a supervised one by fitting a nonparametric unimodal distribution function through isotonic regression (Frisen 1986), referred to as LearnIsotonicLeaf. Once we have collected a set of pairs of points, e.g. from the previously estimated histogram, we employ them as labeled instances to fit a monotonically increasing (resp. decreasing) piecewise linear function up to (resp. down from) the estimated distribution mode. To cope with this unimodality assumption we accommodate LearnMSPN, cf. Alg. 3, to grow a leaf only after no more clustering steps are possible, i.e. it is difficult if not impossible to separate two modalities in the observed data. Note how isotonic regression acts as an additional regularizer: to preserve monotonicity, it does not fit exactly the data points, resulting in a smoother piecewise function.

		MSPN			
		Gower		RDC	
dataset	HBN_{MMHC}	hist	iso	hist	iso
anneal-U	-42.647	-63.553	-38.836	-60.314	**-38.312**
australian	-38.423	-18.513	-30.379	**-17.891**	-31.021
auto	-71.530	-72.998	**-69.405**	-73.378	-70.066
balance-scale	-7.483	-8.038	**-7.045**	-7.932	-7.302
breast	-30.572	-34.027	**-23.521**	-34.272	-24.035
breast-cancer	**-9.193**	-15.373	-9.500	-16.277	-9.990
cars	**-28.596**	-30.467	-31.082	-29.132	-30.516
cleave	-26.296	-26.132	-25.869	-25.707	**-25.441**
crx	-34.563	**-22.422**	-31.624	-24.036	-31.727
diabetes	-29.797	**-15.286**	-26.968	-15.930	-27.242
german	-34.356	-40.828	-33.480	-38.829	**-32.361**
german-org	-29.051	-43.611	**-26.852**	-37.450	-27.294
heart	-28.519	-20.691	-26.994	**-20.376**	-25.906
iris	**-1.670**	-3.616	-2.892	-3.446	-2.843
wins over HBN_{MMHC}	-	**4/14**	**11/14**	**4/14**	**11/14**
wins	**3/14**		**11/14**		

Table 1: Average test set log likelihoods for UCI hybrid datasets (the higher, the better). The best results are bold. MSPNs win in 11 out of 14 cases, even without information about the statistical types (RDC, iso). A Wilcoxon sign test shows that this is significant ($p = 0.05$).

Now we have everything together to evaluate MSPNs empirically. Before doing so, we would like to stress that we here focused on a general setting. Instead of piecewise linear leaves, one can also employ existing hybrid densities as leave distributions such as HBNs, mixtures of truncated exponential families, or other nonparametric density estimators such as Kernel Density Estimators (KDEs) and even denoising and variational autoencoders.

Experimental Evaluation

We intend to investigate the benefits of MSPNs compared to other mixed probabilistic models concerning accuracy and flexibility of inference. Specifically, we investigate the following questions: **(Q1)** Is the MSPN distribution flexible for hybrid domains? **(Q2)** How do MSPNs compare to existing mixed models? **(Q3)** How do MSPNs compare to state-of-the-art parametric models in a single-type domain? **(Q4)** Can MSPNs effectively answer several inference query types over hybrid domains? **(Q5)** Can we leverage MSPNs for interpretability over hybrid domains, even via symbolic computation? We implemented MSPNs[1] in Python and R.

Hybrid UCI Benchmarks (Q1, Q2): We considered the 14 preprocessed UCI benchmarks from the MLC++ library[2] listed in Table 1. The domains span from survey data to medical and biological domains, and they contain both continuous, discrete and categorical variables in different proportions. As a baseline density estimator, we considered HBNs whose conditional dependencies are modeled as conditional linear gaussians (Heckerman and Geiger 1995).

[1]https://github.com/alejandromolinaml/MSPN

[2]https://www.sgi.com/tech/mlc/download.html

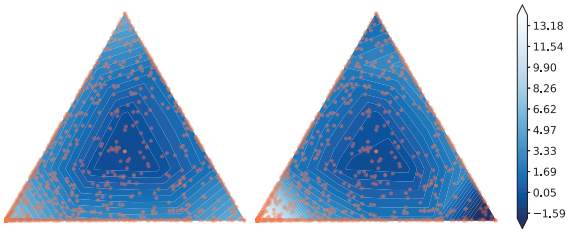

Figure 2: Simplex Distributions: Density of the topics spanning a 2-simplex from the NIPS dataset using **(left)** Dirichlet and **(right)** MSPN distributions. The more flexible MSPN distribution fits the topic distribution well and the lower-left topic better. (Best viewed in color).

To learn their structure we explored both score-based and constrained-based approaches, finding the Max-Min Hill-Climbing (MMHC) algorithm (Tsamardinos, Brown, and Aliferis 2006) to perform the best on the holdout data. For weight learning, we optimized the BDeu score. As an additional sanity check of our nonparametric RDC pipeline, we also trained MSPNs employing K-Medoids using the Gower distance (GowerMSPNs). The Gower distance (Gower and Gower 1971) defines a metric over hybrid domains, at the cost of making distributional assumptions for each variable involved: take the average $d(i,j)=(1/N)\sum_{n=1}^{N} d_{i,j}^n$ of distances $d_{i,j}^n$ per feature n. We assumed continuous variables to be Gaussian and discrete ones to be binomial.

The results are summarized in Table 1. MSPNs clearly outperform HBNs. Moreover, the performance of MSPNs is comparable to GowerMSPNs, proving that using RDC is a sensible idea and frees the user from making parametric assumptions. Using histogram representations allows one to capture mixtures, which turns out to be beneficial for some datasets, but also results in a higher variance in performance across datasets, showing the benefit of isotonic regression. This answers **(Q1, Q2)** affirmatively.

Learning Simplex Distributions (Q3): We considered data common in text and chemistry domains: proportional data, i.e., data lying on the probability simplex, the values are in $[0, 1]$ and sum up to 1. The Dirichlet distribution is arguably the most famous parametric distribution for this type of data. Hence, we used it as a baseline.

First, we considered the NIPS corpus, containing 1,500 documents over the 100 most frequent words. We ran Latent Dirichlet Allocation (LDA) (Blei 2012) with different numbers of topics (3,5,10,20,50) generating different data representations. Fig. 2 shows that the MSPN accurately fits the density, better than a Dirichlet. On NIPS, we also compared MSPNS to Poisson SPNs (PSPNs) of Molina *et al.* (2017), learning both models with $\eta = 200$. The average test log-likelihood of MSPNs was better than PSPNs: -144.41 vs -227.74. This proves how MSPNs are competitive to domain-specific models.

Then, we investigated the Air Quality dataset[3] contain-

[3]https://archive.ics.uci.edu/ml/datasets/Air+Quality. We used only complete instances and ignored the time feature and C6H6

(a) (b) (c) (d)

Figure 3: Towards symbol grounding using MSPN. **(a)** On the top left, the decoded sample predicted for the visual code $\{0, 0, 1, 0, 1, 1\}$ (corresponding to a "3"), on the bottom left its training closest sample. On the right, decoded conditional samples for the same visual code. **(b,c)** Decoded conditional samples for codes in between classes "3" and "5" ($\{0, 0, 1, 1, 1, 1\}$) as well as "1" and "5" ($\{1, 0, 1, 1, 1, 0\}$), respectively. **(d)** Some test images on the left and their MSPN reconstructions (left, right, up, down) on the right. The reconstructed parts are denoted by a red background. (Best viewed in color)

Dimension	Dirichlet	MSPN(RDC,iso)	MSPN(Grower,iso)
	NIPS + LDA		
3	2.045 (± 0.297)	4.071 (± 0.66)	**4.333 (± 0.627)**
5	7.311 (± 0.406)	10.376 (± 0.671)	**10.419 (± 0.711)**
10	25.047 (± 0.787)	**35.927 (± 1.755)**	34.205 (± 1.716)
20	69.668 (± 2.014)	**109.222 (± 4.179)**	92.981 (± 4.245)
50	245.008 (± 3.573)	338.477 (± 6.976)	**349.259 (± 9.916)**
	Air Quality + Archetypes		
3	2.939 (± 1.536)	5.852 (± 2.261)	**7.114 (± 2.272)**
5	14.625 (± 4.678)	**16.494 (± 7.574)**	15.099 (± 4.888)
10	61.317 (± 4.81)	84.124 (± 6.575)	**85.645 (± 5.887)**
20	174.171 (± 5.799)	232.075 (± 7.74)	**242.482 (± 10.224)**
	Hydrochemicals		
12	59.546 (± 1.781)	71.013 (± 3.591)	**82.377 (± 1.445)**
wins over Dir.	-	**10/10**	**10/10**
wins	**0/10**		**10/10**

Table 2: Average test set log likelihoods (the higher, the better) on proportional data; best results bold. Clearly, MSPNs outperform the less flexible Dirichlet distribution, even without information about the statistical type (RDC, iso). Positive values are due to continuous random variables.

ing 6,941 measurements for 12 features about air composition. We ran Archetypal Analysis (Cutler and Breiman 1994; Thurau et al. 2012) for 3, 5, and 10 archetypes and extracted the convex reconstructions of the original data. We also considered the hydro-chemical dataset of Tolosana-Delgado et al. (2005), containing 485 observations of 14 chemical measurements of a river. We fit MSPNs and the Dirichlet over relative concentrations. The 10-fold cross-validated mean log-likelihoods for all models on the three datasets are summarized in Table 2. As one can see, in all cases MSPNs can capture the distribution on the simplex better than the Dirichlet. This is to be expected as MSPNs can capture more complex (in)dependencies, whereas the Dirichlet makes stronger independence assumptions. All simplex experiments together answer **(Q3)** affirmatively.

Leveraging symbolic-semantic information (Q4): Symbol grounding is at the heart of AI, and we explored MSPNs as a step towards tackling this classical AI problem. We considered the 28×28 MNIST digit images, represented as 16 continuous features extracted from an autoencoder

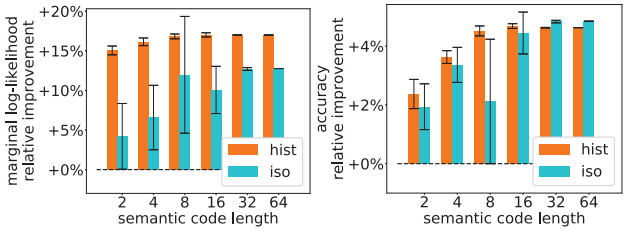

Figure 4: Average relative improvement over ten trials (y axis) for **(left)** the marginal test log-likelihood $P(\mathbf{X})$ and **(right)** for the class accuracy based on $P(C)$ of MSPNs learned an autoencodings augmented with semantic class codes of increasing length (x axis). (Best viewed in color)

(AE) trained on the training split: we trained two layers of 256 and 128 ReLU neurons for both the encoder and the decoder for 200 epochs using adam (learning rate 0.002 with no decay, β_1 and β_2 coefficients set to 0.9 resp. 0.999). We then augmented MNIST with symbolic semantic information encoded as binary codes. Each bit of the code is 1 if a digit contains one of the following visual features: (i) a vertical stroke (true for 1, 4 and 7), (ii) a circle (0, 6, 8 and 9), (iii) a left curvy stroke (2, 3, 5, 8 and 9), (iv) a right curvy stroke (5 and 6), (v) a horizontal stroke (7, 2, 3, 4, and 5), (vi) a double curve stroke (3 and 8). For instance, images representing a "3" are encoded as $(0, 0, 1, 0, 1, 1)$ while $(0, 0, 1, 1, 1, 0)$ corresponds to "5". We also added the class label C as a feature. Let $\mathbf{X}$ denote the continuous embedding variables, $\mathbf{Y}$ the additional 6 binary symbolic features, and C the categorical class variable.

In a first experiment, we trained an MSPN on a 10000 subsample of the augmented MNIST training data to model $P(\mathbf{X}, \mathbf{Y})$, setting $\eta = 200$ and $\Delta = 1$, $k = 20$. Then, we evaluated on the augmented MNIST test split whether the learned MSPN had captured the non-explicit dependencies between the three different feature domains. First, we predict $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} P(\mathbf{x}|\mathbf{Y} = \mathbf{y}_c)$, for each visual code $\mathbf{y}_c$ belonging to class $c \in C$. Fig. 3 (a) visualizes the prediction $\mathbf{x}^*$ as decoded by the autoencoder back in pixel space. As one can see, the MSPN is not only able to recover the cor-

rect class but also does not just memorize a training sample. Conditional sampling provides an additional visual proof: after propagating bottom-up the evidence for an observed code $\mathbf{y}_c$, we sample a configuration $\mathbf{x}$ (applying Vergari, Di Mauro, and Esposito's (2016) top-down approach). Decoded samples clearly belong to the class c, cf. Fig. 3 (a). Then, to evaluate how good the MSPN was able to *glue* the continuous and binary domains, we performed conditional sampling starting from unseen visual codes. For instance, for the code $(0, 0, 1, 1, 1, 1)$, merging the visual codes for 3 and 5, we expect a digit in between the two classes. Fig. 3 (b) confirms this: decoded samples belong to either class or are closely in between them. Similarly, Fig. 3 (c) shows samples conditioned on code $\{1, 0, 1, 1, 1, 0\}$, merging classes 5 and 1. Next, we investigated how much symbol groundings can help density estimation and classification. On the MNIST test split, we investigated the benefit of using visual codes $\mathbf{Y}$ of length 2,4,8,16,32,64. We measured the improvement of the marginal likelihood $P(\mathbf{X})$ resp. the classification accuracy based on $P(C)$ of an MSPN $\mathcal{B}$ trained on $(\mathbf{X}, \mathbf{Y}, C)$ over an MSPN $\mathcal{A}$ trained only over $(\mathbf{X}, C)$: $(\ell_{\mathcal{B}} - \ell_{\mathcal{A}})/\ell_{\mathcal{A}} \cdot 100$ for both measures ℓ. The results are summarized in Fig. 4. As one can see, increasing the number of symbolic features positively improves both the marginal likelihood over $\mathbf{X}$ and the classification performance. Note that for computing $P(\mathbf{X})$ and to predict $c^* = \operatorname{argmax}_c P(c|\mathbf{X})$, one has to marginalize over $\mathbf{Y}$, which cannot be done efficiently using classical mixed graphical models. Finally, we employed MSPNs for MNIST reconstruction. We processed the original images as two halves—left (l) and right (r), up (u) and down (d)—and encoded each half into 16 continuous features by learning one autoencoder independently for each one of them. Note that each variable set X_l, X_r, X_u and X_d forms a domain with a different distribution. We learned MSPNs for $P(\mathbf{X}_l, \mathbf{X}_r)$ and $P(\mathbf{X}_u, \mathbf{X}_d)$ and performed MPE inference to predict one half of a test image given the other. Predicted samples are shown in Fig. 3 (d). As one can see, the reconstructions are indeed very plausible. This suggests that MSPNs are a valuable tool to effectively learn distributions and make predictions across different domains. The experiments on leveraging symbolic-semantic information answer **(Q4)** affirmatively.

Mixed Mutual Information (Q5): Recall, an MSPN encodes a polynomial over leaf piecewise polynomials. Consequently, one can employ a symbolic solver to evaluate the overall network polynomial to efficiently compute information-theoretic measures that would be difficult to calculate otherwise, in particular for hybrid domains. To illustrate this for MSPNs, we consider computing mutual information (MI) in hybrid domains. MI also provides a way to extract the gist of MSPNs as it highlights relevant variable associations only. Fig. 5 shows the MI network induced over the Autism Dataset (Deserno et al. 2016), which reflects natural semantic connections. This not only answers **(Q5)** affirmatively but also indicates that MSPNs may pave the way to automated mixed statisticians: the MI together with the tree structure of MSPNs can automatically be compiled into textual descriptions of the model.

To summarize our experimental results as a whole, all questions **(Q1)**-**(Q5)** can be answered affirmatively.

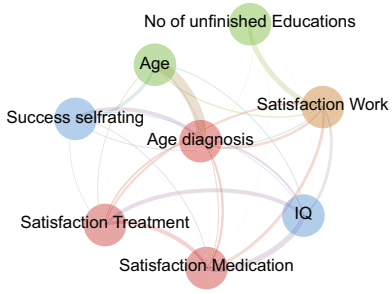

Figure 5: Visualizing the Autism MSPN via normalized mutual information (the thicker, the higher). The strong natural interactions between *Age* and *Age of diagnosis* and between *Satisfaction Work* and *No of unfinished Education* are correctly recovered as in the results of (Haslbeck and Waldorp 2015) using a pairwise mixed graphical model with known parametric forms from the exponential family (Yang et al. 2014). Node colors encode different feature groups: Demographics (green), Psychological (blue), Social Environment (orange) and Medical (red). (Best viewed in color)

Conclusions

The Mixed Sum-Product Networks (MSPNs), are a novel combination of nonparametric probability distributions and deep probabilistic models. In contrast to classical shallow mixed graphical models, they provide effective learning, tractable inference and enhanced interpretability. Our experiments demonstrate that MSPNs are competitive to parameterized distributions as well as mixed graphical models and make probabilistic queries easier to compute. They allow users to train multivariate mixed distributions more easily than previous approaches across a wide range of domains.

MSPNs suggest several avenues for future work: scaling structure learning to large number of instances or high-dimensional data; learning boosted and mixtures of MSPNs along with exploring other nonparametric leaves such KDE, other mixed graphical models, and variational autoencoders, or extending them to other instances of arithmetic circuits (Choi and Darwiche 2017); making use of weighted model integration solvers for capturing more complex types of queries (Belle, Passerini, and Van den Broeck 2015; Morettin, Passerini, and Sebastiani 2017). One interesting avenue is to turn MSPNs into automated statisticians, able to predict the statistical type of a variable—is it continuous or ordinal?—and ultimately its parametric form—is it Gaussian or Poisson (Valera and Ghahramani 2017)?

Acknowledgements: The authors would like to thank the anonymous reviewer for the valuable feedback. This work is motivated and partly supported by the BMEL/BLE project DePhenSe, FKZ 313-06.01-28-1-82.047-15. AM has been supported by the DFG CRC 876 "Providing Information by Resource-Constrained Analysis", project B4. SN has been supported by the CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO).

KK acknowledges the support by the Centre for Cognitive Science at the TU Darmstadt.

References

Bekker, J.; Davis, J.; Choi, A.; Darwiche, A.; and Van den Broeck, G. 2015. Tractable learning for complex probability queries. In *Proc. of NIPS*.

Belle, V.; Passerini, A.; and Van den Broeck, G. 2015. Probabilistic inference in hybrid domains by weighted model integration. In *In Proc. of IJCAI*, 2770–2776.

Belle, V.; Van den Broeck, G.; and Passerini, A. 2015. Hashing-based approximate probabilistic inference in hybrid domains. In *UAI*, 141–150.

Blei, D. M. 2012. Probabilistic topic models. *CACM* 55(4):77–84.

Cheng, W.; Kok, S.; Pham, H. V.; Chieu, H. L.; and Chai, K. M. A. 2014. Language modeling with Sum-Product Networks. In *Proc. of Interspeech*.

Choi, A., and Darwiche, A. 2017. On relaxing determinism in arithmetic circuits. In *Proceedings of ICML*, 825–833.

Cutler, A., and Breiman, L. 1994. Archetypal analysis. *Technometrics* 36(4):338–347.

Deserno, M. K.; Borsboom, D.; Begeer, S.; and Geurts, H. M. 2016. Multicausal systems ask for multicausal approaches: A network perspective on subjective well-being in individuals with autism spectrum disorder. *Autism* 1362361316660309.

Elidan, G. 2010. Copula bayesian networks. In *Proc. of NIPS*.

Frisen, M. 1986. Unimodal regression. *Journal of the Royal Statistical Society. Series D* 35(4):479–485.

Gebelein, H. 1941. Das statistische Problem der Korrelation als Variations- und Eigenwertproblem und sein Zusammenhang mit der Ausgleichsrechnung. *Zeitschrift fur Angewandte Mathematik und Mechanik* 21(6):364–379.

Gens, R., and Domingos, P. 2013. Learning the Structure of Sum-Product Networks. In *Proc. of ICML*.

Gower, J. C., and Gower, J. C. 1971. A general coefficient of similarity and some of its properties. *Biometrics*.

Haslbeck, J. M. B., and Waldorp, L. J. 2015. mgm: Estimating time-varying mixed graphical models in high-dimensional data. *ArXiv* 1510.06871.

Heckerman, D., and Geiger, D. 1995. Learning bayesian networks: a unification for discrete and gaussian domains. In *Proc. of UAI*.

Langseth, H.; Nielsen, T. D.; Rumı, R.; and Salmerón, A. 2012. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning* 53(2):212–227.

Lauritzen, S., and Wermuth, N. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics* 17(1):31–57.

Lopez-Paz, D.; Hennig, P.; and Schölkopf, B. 2013. The randomized dependence coefficient. In *Advances in neural information processing systems*, 1–9.

Molina, A.; Natarajan, S.; and Kersting, K. 2017. Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In *Proc. of AAAI*.

Moral, S.; Rumi, R.; and Salmerón, A. 2001. Mixtures of truncated exponentials in hybrid bayesian networks. In Benferhat, S., and Besnard, P., eds., *Proc. of ECSQARU*.

Morettin, P.; Passerini, A.; and Sebastiani, R. 2017. Efficient weighted model integration via smt-based predicate abstraction. In *Proc. of IJCAI*.

Peharz, R.; Tschiatschek, S.; Pernkopf, F.; and Domingos, P. 2015. On theoretical properties of sum-product networks. In *Proc. of AISTATS*.

Póczos, B.; Ghahramani, Z.; and Schneider, J. G. 2012. Copula-based kernel dependency measures. *arXiv 1206.4682*.

Poon, H., and Domingos, P. 2011. Sum-Product Networks: a New Deep Architecture. *Proc. of UAI*.

Rahimi, A., and Recht, B. 2009. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proc. of NIPS*.

Rényi, A. 1959. On measures of dependence. *Acta mathematica hungarica* 10(3-4):441–451.

Rooshenas, A., and Lowd, D. 2014. Learning sum-product networks with direct and indirect variable interactions. In *Proc. of ICML*, 710–718.

Rozenholc, Y.; Mildenberger, T.; and Gather, U. 2010. Combining regular and irregular histograms by penalized likelihood. *Comp. Statistics & Data Analysis* 54(12):3313–3323.

Sanner, S., and Abbasnejad, E. 2012. Symbolic variable elimination for discrete and continuous graphical models. In *Proc. of AAAI*.

Shenoy, P., and West, J. 2011. Inference in hybrid bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning* 52(5):641–657.

Thurau, C.; Kersting, K.; Wahabzada, M.; and Bauckhage, C. 2012. Descriptive matrix factorization for sustainability adopting the principle of opposites. *DAMI* 24(2):325–354.

Tolosana-Delgado, R.; Otero, N.; Pawlowsky-Glahn, V.; and Soler, A. 2005. Latent compositional factors in the llobregat river basin (spain) hydrogeochemistry. *Math. Geology* 37(7):681–702.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. 2006. The max-min hill-climbing bayesian network structure learning algorithm. *MLJ* 65(1):31–78.

Valera, I., and Ghahramani, Z. 2017. Automatic discovery of the statistical types of variables in a dataset. In *ICML*.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2015. Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning. In *Proc. of ECML-PKDD*.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2016. Visualizing and understanding sum-product networks. *arXiv 1608.08266*.

Yang, E.; Baker, Y.; Ravikumar, P.; Allen, G.; and Liu, Z. 2014. Mixed graphical models via exponential families. In *Proc. of AISTATS*.

Zhao, H.; Melibari, M.; and Poupart, P. 2015. On the Relationship between Sum-Product Networks and Bayesian Networks. In *Proc. of ICML*.