Tractable Querying and Learning in Hybrid Domains via Sum-Product Networks

Tractable Querying and Learning in Hybrid Domains via Sum-Product Networks

Andreas Bueff [*]
University of Edinburgh

Stefanie Speichert[*]
University of Edinburgh

Vaishak Belle
University of Edinburgh &
Alan Turing Institute

Abstract

Probabilistic representations, such as Bayesian and Markov networks, are fundamental to much of statistical machine learning. Thus, learning probabilistic representations directly from data is a deep challenge, the main computational bottleneck being inference that is intractable. Tractable learning is a powerful new paradigm that attempts to learn distributions that support efficient probabilistic querying. By leveraging local structure, representations such as sum-product networks (SPNs) can capture high tree-width models with many hidden layers, essentially a deep architecture, while still admitting a range of probabilistic queries to be computable in time polynomial in the network size. The leaf nodes in SPNs, from which more intricate mixtures are formed, are tractable univariate distributions, and so the literature has focused on Bernoulli and Gaussian random variables. This is clearly a restriction for handling mixed discrete-continuous data, especially if the continuous features are generated from non-parametric and non-Gaussian distribution families. In this work, we present a framework that systematically integrates SPN structure learning with weighted model integration, a recently introduced computational abstraction for performing inference in hybrid domains, by means of piecewise polynomial approximations of density functions of arbitrary shape. Our framework is instantiated by exploiting the notion of propositional abstractions, thus minimally interfering with the SPN structure learning module, and supports a powerful query interface for conditioning on interval constraints. Our empirical results show that our approach is effective, and allows a study of the trade off between the granularity of the learned model and its predictive power.

1 INTRODUCTION

Probabilistic representations, such as Bayesian and Markov networks, are fundamental to much of statistical machine learning. Thus, learning probabilistic representations directly from data is a deep challenge, the main computational bottleneck being inference that is intractable (#-P hard [Valiant, 1979, Bacchus et al., 2009]). In practise, most systems approximate inference during the learning step as well as the querying step, often with weak guarantees, which is problematic for applications such as autonomous vehicles and health monitoring.

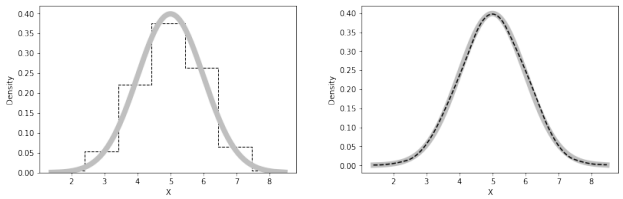
Tractable learning is a powerful new paradigm that attempts to learn distributions that support efficient probabilistic querying. Much of the initial work focused on low tree-width models [Bach and Jordan, 2002], but later, building on properties such as local structure [Chavira and Darwiche, 2008], data structures such as arithmetic circuits (ACs) emerged. These can also represent high tree-width models and enable exact inference for a range of queries in time polynomial in the circuit size. Sum-product networks (SPNs) [Poon and Domingos, 2011] are instances of ACs with an elegant recursive structure – essentially, an SPN is a weighted sum of products of SPNs, and the base case is a leaf node denoting a tractable probability distribution (e.g., a univariate Bernoulli distribution). In so much as deep learning models can be understood as graphical models with multiple hidden variables, SPNs can be seen as a tractable deep architecture. Of course, learning the architecture of standard deep models is very challenging [Bengio, 2009], and in contrast, SPNs, by their very design, offer a tractable structure learning paradigm. While it is possible to specify SPNs by hand, weight learning is additionally required to obtain a probability distribution, but also the specification of SPN has to obey conditions of completeness and decomposability, all of which makes structure learning an obvious choice. Since SPNs were introduced, a number of structure learning frameworks have been developed for those and related data structures, e.g., [Gens and Domingos, 2013, Hsu et al., 2017, Liang et al., 2017].

Since the base case for SPNs is a tractable distribution, SPNs are essentially a schema for composing tractable distributions. Although much of the literature focuses on univariate Bernoulli distributions [Gens and Domingos, 2013, Liang et al., 2017], continuous models can be considered too, which makes SPNs very appealing because real-world

[*] Contributed equally to the research.

data is often hybrid, with discrete, categorical and continuous entities. Nonetheless, this rests on the assumption of having a tractable query interface to the underlying continuous and hybrid distributions. In [Hsu et al., 2017], for example, the leaf nodes are assumed to be drawn from a Gaussian, and similarly, [Vergari et al., 2015, Molina et al., 2017] assume the data is drawn from other families with known parametric form.

In the literature on probabilistic inference, to handle non-parametric families, in (say) non-Gaussian graphical models, there has been considerable interest in so-called piecewise polynomial approximations [Sanner and Abbasnejad, 2012, Shenoy and West, 2011, Belle et al., 2015]. Essentially, these approximations can be made arbitrarily close to (say) exponential families and support efficient integration (e.g., [Baldoni et al., 2014, Albarghouthi et al., 2017]). Moreover, as shown in [Belle et al., 2015], just as inference in discrete graphical models reduces to weighted model counting [Chavira and Darwiche, 2008, Bacchus et al., 2009], obtained from the sum of products of atoms' weights in a propositional model, inference in hybrid graphical models reduces to the so-called weighted model integration (WMI), obtained by integrating the product of atoms' densities in a first-order model. For example, a Gaussian distribution $\mathcal{N}(5, 1)$ for a random variable x can be represented using (truncated) models:



(a) Constant Approximation (b) Third Order Polynomial

Figure 1: Comparison of piecewise constant vs polynomial using a 7 piece function

As a computational strategy, what makes WMI particularly effective is that: (a) the intervals can be understood as (and mapped to) propositions [Belle et al., 2015, Morettin et al., 2017], known in the literature as *propositional abstraction*, allowing us to piggyback on any propositional solver, and (b) WMI admits complex interval queries, such as $\Pr(x > 1.5 \mid y < 2)$ where y is some other random variable, yielding a powerful query interface. Notice here that the intervals in the queries do not have to correspond to the intervals used for specifying the distribution, and can overlap and subsume them arbitrarily.

In this paper, we combine the power and flexibility of WMI with SPNs to address the problem of tractable learning and querying in mixed discrete-continuous domains. Among other things, this allows us to learn both parametric and non-parametric models and their mixtures, including models such as Figure 1 and Figure 2, but with no prior knowledge of the true distribution, all of which is further composed over hidden layers in a deep architecture. In other words, unlike many conventional deep learning frameworks, leveraging SPNs enables a fully *unsupervised* learning regime for hybrid data.

From a representational viewpoint, our framework allows the modeller to study the granularity of the continuous distributions (in terms of the number of piecewise components and the degree of the polynomial fit), and determine the empirical trade off between accuracy and model size / interpretability. By lifting the propositional abstraction strategy of WMI, the framework is also instantiated in a manner that allows us to use any SPN learner in principle. To the best of our knowledge, this is a first attempt to combine SPNs and WMI in their full generality, with an interface for complex interval queries.

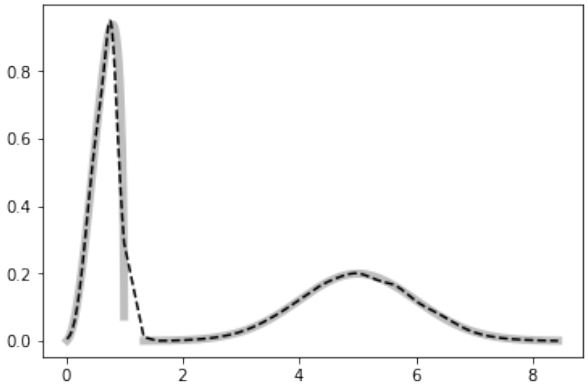


Figure 2: The piecewise polynomial approximate of a mixture of a Gaussian and Beta distribution

Outside of the above lines of research, research in learning in hybrid domains has been primarily limited to well-known parametric families. For example, [Heckerman et al., 1995] focus on conditional linear Gaussian models, and [Yang et al., 2014] focus on mixing exponential families. This is perhaps not surprising, because inference in hybrid domains has been primarily limited to approximate computations, e.g., [Murphy, 1999], and/or Gaussian models [Lauritzen and Jensen, 2001]. In similar spirit, approaches such as [Nitti et al., 2016, Ravkic et al., 2015] consider learning of complex symbolic constraints by assuming base distributions as being either Gaussian or a softmax equality.

We are organised as follows. We introduce the formal foundations for our work, and then turn to our approach that integrates SPNs and WMI. We then discuss our interface for complex interval queries. We turn to empirical evaluations after that, and finally conclude.

2 Preliminaries

2.1 Weighted Model Integration

Given a propositional formula Δ, the task of model counting is to compute the total number of satisfying assignments for Δ. For example, $p \vee q$ has 3 models, and a satisfying ratio of 3/4. Weighted model counting (WMC) is its extension that additionally accords weights to the models [Chavira and Darwiche, 2008]. Models are usually accorded weights by computing the product of weighted literals: that is, assuming w maps literals in Δ to positive reals, we define:

$$\mathrm{WMC}(\Delta, w) = \sum_{M \models \Delta} \prod_{l \in M} w(l)$$

where the sum ranges over propositional models, and $l \in M$ denotes the literals true at the model M. For example, suppose p and q are accorded a weight of .6 and .3 respectively, with the understanding that the weights of a negated atom $\neg a = 1 - w(a)$. Then, the weight accorded to $[p = 1, q = 1]$ would be .18, and $\mathrm{WMC}(p \vee q, w) = .18 + .42 + .12 = .72$.

WMC is a state-of-the-art exact inference scheme, and owing to its generality, inference in a number of formalisms, such as relational Bayesian networks and probabilistic programs [Fierens et al., 2011], are encoded as WMC tasks. Given a formula Δ, a weight function w, a query q and evidence e, probabilities are computed by means of the expression: $\Pr(q \mid e, \Delta) = \mathrm{WMC}(\Delta \wedge q \wedge e, w)/\mathrm{WMC}(\Delta \wedge e, w)$.

Due to the propositional setting, however, WMC is limited to finite domain discrete random variables, which has spurred considerable interest in generalising WMC to continuous and hybrid distributions [Belle et al., 2015, Chistikov et al., 2015, Albarghouthi et al., 2017]. Weighted model integration (WMI) is a computational abstraction for computing probabilities with continuous and mixed discrete-continuous distributions [Belle et al., 2015]. The key idea is to additionally consider inequality atoms, such as $0 \leq x \leq 10$, along with possibly non-numeric weights, such as x^2. The intuition is to let the inequality atom denote an interval of the domain of a density function, and let the weight define the function for that interval. Given such a knowledge base, the weighted model integration (WMI) is obtained from the model count of the theory together with a volume computation for the intervals. Formally:

$$\mathrm{WMI}(\Delta, w) = \sum_{M \models \Delta^{-}} \int_{\{l^{+} \,:\, l \in M\}} \prod_{l \in M} w(l)$$

where ϕ^{-} denotes a *propositional abstraction*, based on a mapping from linear arithmetic expressions to propositional atoms, and ϕ^{+} denotes a *refinement*, where the atoms are mapped back to to their linear arithmetic expressions. For example, $(0 \leq x \leq 10) \vee q$ on abstraction yields $p \vee q$, where p is a fresh atom chosen so as to not conflict with any of the existing atoms in the theory, which has a model count of 3 on abstraction. Suppose we have weights of x^2 and .3 respectively, and suppose the weights of all negated atoms are 0. Then, the WMI for the model $[p = 1, q = 1] = [(0 \leq x \leq 10) = 1, q = 1]$ is obtained as $\int_0^{10} x^2 \cdot .3 = 300/3 = 100$. Because negated atoms obtain a weight of 0, the WMI for $p \vee q$ is also 100. Conditional probabilities are calculated using precisely the same expression as for WMC.

The WMI apparatus is very flexible [Morettin et al., 2017]: for example, it is possible for query and evidence atoms to arbitrarily range over the intervals mentioned in the knowledge base. For example, suppose we have the query atom is $5 \leq x \leq 7$ and $e = \mathit{true}$. (We treat the weights of query atoms and their negations as being 1.) In that case, $\Pr(q \mid e, \Delta) = 21.8$.

2.2 Sum-Product Networks

SPNs are rooted acyclic graphs whose internal nodes are sums and products, and leaf nodes are tractable distributions, such as Bernoulli and Gaussian distributions [Poon and Domingos, 2011, Gens and Domingos, 2013]. More precisely, letting the *scope* of an SPN be the set of variables appearing in it, a recursive definition is given as follows:

(1) a tractable univariate distribution is an SPN (the base case); (2) a product of SPNs with disjoint scopes is an SPN (denoting a factorisation over independent distributions); and (3) a weighted sum of SPNs with the same scope, where the weights are positive, is an SPN (denoting a mixture of distributions).

Formally, SPNs encode a function for every node that maps random variables $X_1, \ldots, X_n$ to a (numeric) output. For node k, the corresponding function f_k maps a world $X_1 = x_1, \ldots, X_n = x_n$ to its probability if its a leaf node, the weighted sum $\sum_i w_i f_i^k(x_1, \ldots, x_n)$ for sum nodes where f_i^k are the children of node k, and $\prod_i f_i^k(x_1, \ldots, x_n)$ for product nodes. Conditional probabilities are expressed using:

$$\Pr(x_1, \ldots, x_m \mid x_{m+1}, \ldots, x_n) = f_0(x_1, \ldots, x_n)/f_0(x_{m+1}, \ldots, x_n)$$

where 0 is the root node, and the notation f_k with only a subset of the arguments used in the denominator denotes marginalisation over the remaining arguments. Similar expressions can be given for other queries, such as the MAP state and marginals. The benefit of SPNs here is that a bottom-up pass allows us to compute these classes of probabilistic queries in time polynomial in the circuit size. (See, for example, [Bekker et al., 2015] on other data structures that allow a more expressive class of queries.)

3 Structure Learning

We propose an algorithm to derive the structure and parameters for SPNs with piecewise polynomial leaves, equipped

with WMI (WMISPNs). The method LearnWMISPN is capable of deriving such a SPN structure from hybrid data and further, it allows for the retention of the distribution parameters in the continuous case for use in advanced querying.

The structure learning approach for LearnWMISPN is derived from LearnSPN [Gens and Domingos, 2013]. LearnSPN is a recursive top-down learning method which is capable of learning the structure and weights for SPNs by identifying mutually independent variables, and clustering similar instances. LearnSPN takes a simple approach to structure learning by recursively splitting data into a product of SPNs over independent variables, or a sum of SPNs comprising subsets of instances. More generally, LearnSPN represents an algorithm schema which allows for a modular approach in structure learning from differing domains, which is why our implementation changes only the base case.

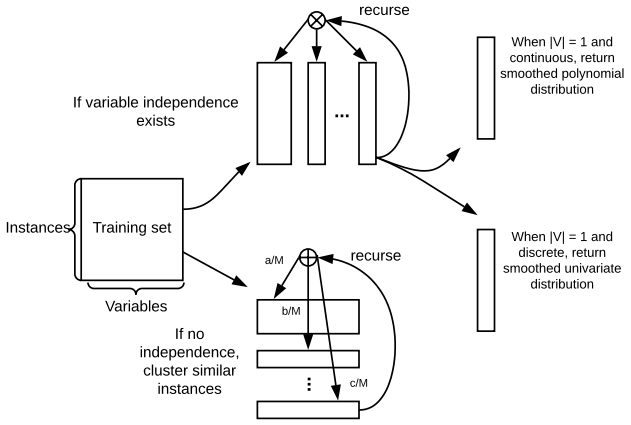


Figure 3: A recursive algorithm for learning hybrid domains

The primary extension with LearnWMISPN is the addition of generated polynomial leaf nodes. Given a dataset with discrete, categorical and continuous attributes, LearnWMISPN learns polynomial weight functions as density approximations for the continuous cases, mapped to leaf nodes. By doing so, LearnWMISPN provides a model which is capable of complex querying in the continuous case, while also conditioning over other variable types for hybrid domains.

Prior to structure learning, a preprocessing step is performed which first transforms discrete, categorical, and continuous features into a binary representation. Much like the algorithm schema of LearnSPN, the preprocessing step can implement any number of unsupervised binning methods including using a *mean split, equal frequency binning,* or *equal width binning*. This is followed by a polynomial learning function which identifies the polynomial function coefficients and probability densities for the continuous features in the hybrid domain without prior knowledge of the true density function.

Algorithm 1: LearnWMISPN (T, V')

Input : T: set of instances, V': set of variables, θ: parameters (cf. Algorithm 2)

Output: SPN representing a distribution over V' learned from T

if $|V'| = 1$ **then**
  if variable V' is an instance of a continuous feature **then**
    return univariate distribution estimated from polynomial probability densities
  else
    return univariate distribution estimated from binary counts in T
  end
else
  partition V' into approximately independent subsets V'_j
  if success **then**
    return $\prod_j$LearnWMISPN(T, V'_j);
  else
    partition T into subsets of similar instances T_i
    return $\sum_i \frac{|T_i|}{|T|}$LearnWMISPN(T_i, V')
  end
end

Algorithm 1 describes the recursive structure learning algorithm for LearnWMISPN and Figure 3 illustrates the recursive pipeline. LearnWMISPN takes as input the preprocessed binary dataset D' with T instances and V' variables and returns a WMISPN representing the distributions of the hybrid domains. LearnWMISPN recurses on subsets of V' until a vector of unit length is found, at which point a corresponding univariate distribution is returned.

The main novelty in LearnWMISPN is the handling of the base case. Without any prior knowledge, LearnWMISPN will return a smoothed univariate distribution for discrete and categorical variables, or piecewise polynomial distribution for continuous variables. These two outcomes sets LearnWMISPN apart from LearnSPN, and this is represented with polynomial leaf nodes corresponding to continuous distributions.

If the base case has not been reached, then LearnWMISPN continues the recursive structure learning of LearnSPN. For the decomposition step, a variable split is identified which results in mutually independent subsets, generating a product node for the resulting subset. Mutual independence for the variable splits is determined by a G-test for pairwise independence. For the conditioning step similar instances are clustered using hard incremental expected-maximization (EM) generating a sum node. The ratio split

on the clustered subset of instances determines the corresponding weight and returned is the sum of the resulting weighted clusters.

Preprocessing Step: A preprocessing step is required to first transform the discrete, categorical, and continuous variables into a binary representation. The methods for converting a dataset into a binary representation is contingent on the variable type in the hybrid dataset. The preprocessing pipeline is summarized in Algorithm 2.

Algorithm 2: Preprocessing step($D, \mathbf{f}, b$)

Input : D: dataset of real values, $\mathbf{f}$: vector of domain types, b: number of bins

Output: D': binary representation of D, M: matrix of coefficients for the polynomials, $\mathbf{p}$:vector of probability densities

1 $D' \leftarrow$EqualWidthBinning($D, \mathbf{f}, b$)

M, $\mathbf{p} \leftarrow$PolynomialLearner($D, \mathbf{f}, b$)

The first sub-task converts the real valued dataset D into a binary representation D'. In our model, equal width binning was performed on continuous features, with the number of equal width bins specified as a parameter. (A meta-learning step can be designed to choose the optimal number by studying log-likelihood or polynomial improvements.) Higher bin counts would increase the representative power and improve accuracy for advanced queries on continuous distributions (see section 5). Each new bin generated from a feature represents a discrete range on that continuous distribution. A one-hot encoding transformation was performed on the expanded feature set to get a binary representation. For discrete and categorical variables, the binary representation is achieved again with a one-hot encoded transformation with equal width binning not taken into consideration.

As an example for one-hot encoding in the discrete case, consider a random variable Y which has a domain of boolean values. Booleans result in two classification labels so Y is expanded to (Y_1, Y_2), which now correspond to each boolean instance. Formally:

$$Y(x) = \begin{cases} Y_1 = 1 \text{ if } x \\ Y_2 = 1 \text{ if } \neg x \end{cases}$$

The two cases for Y are then represented as $[x] \rightarrow [1, 0]$ and $[\neg x] \rightarrow [0, 1]$ respectively. Now consider the continuous case, where equal width binning is implemented, we say that random variable X is defined as having a range $[a, b] \implies x \in \mathbb{R} : a \leq x \leq b$. If we split X into say three bins (X_1, X_2, X_3), with the split point defined as $s = (b-a)/3$ then we again have another representation for one hot encoding. Formally:

$$X(x) = \begin{cases} X_1 = 1 \text{ if } a \leq x < (a + s) \\ X_2 = 1 \text{ if } (a + s) \leq x < (a + 2s) \\ X_3 = 1 \text{ if } (a + 2s) \leq x \end{cases}$$

So for the minimum and maximum values in X we get the following one-hot encoded representation $[a] \rightarrow [1, 0, 0]$ and $[b] \rightarrow [0, 0, 1]$. This means the binary transformation of an instance $D_{i,:} = [b, x]$ is $D'_{i,:} = [0, 0, 1, 1, 0]$.

Polynomial Learner: The second task learns a piecewise polynomial approximation for any univariate probability density function (PDF) without prior knowledge or simplifying assumption. The original dataset D, the indices for the continuous variables, and the number of bins are taken as input, and as output we receive a vector of probability densities for each continuous feature bin. This bin information is retained in its respective polynomial leaf node in the SPN for use in advance querying.

The piecewise polynomial functions are calculated through a linear combination of b-splines [Speichert, 2017]. Basis splines form a basis in the piecewise polynomial space. Linear combinations of splines can represent any piecewise polynomial function. This is calculated while ensuring that the piecewise polynomial follows the usual properties of density functions such as its integral being 1.

The order, and, if not previously specified, the number of bins are chosen during training time by means of the Bayesian Information Criterion (BIC) [Schwarz, 1978]. This scoring function has been shown to be robust and was chosen to avoid overfitting the model parameters.

The procedure is performed without any knowledge of the true underlying density function. It, furthermore, displays some desirable properties for tractable inference such as polynomials being computationally easy to find and integrate and being closed under multiplication and addition and, therefore, under mixtures and marginalisation. In addition, the chosen method favours smaller polynomial ranks [Speichert, 2017] which improves the efficiency of the integration without loosing accuracy.

4 Querying

SPNs provide tractable querying on univariate distributions, and have the capacity to calculate the marginal and conditional probabilities on learned features. Other schemas for SPNs limit the scope of querying to binary activations $\Pr(X_i = 1) = x_i$ to the leaf nodes [Poon and Domingos, 2011]. In WMISPNs, we are able to expand our queries to complex cases where f_k maps X_i to new probability ranges $(a \leq x \leq b)$, $a, b \in \mathbb{R}$.

4.1 SPN Inference

General SPN inference is initiated at the leaf nodes where queries are presented in the form of selected activations of the random variables $Q = [X_1 = 1, X_2 = 0, \ldots, X_n = 1]$. The mapped probabilities x_i propagate upward from the leaf nodes where values from children nodes are either multiplied at product nodes, or the weighted sum is taken at sum nodes. The final likelihood of the queries is returned at the root node 0.

In order to normalize the returned likelihood, a partition function is required to calculate the normalization constant. Formally:

$$Z = \sum_i f_0(x_i)$$

Inference is deemed tractable due to the partition function being computational in time linear to the number of edges in the learned SPN, which in turn results in queries that can be calculated in time linear $\Pr(Q) = f_0(x_1, 1 - x_2, \ldots, x_n)/Z$. The same is true for MAP and conditional likelihood estimates.

4.2 Complex Querying

We extend the SPN inference model to allow complex queries where inference can be performed over continuous as well as discrete features, and combinations thereof: for example, queries on discrete features such as $male(X)$ conditioned on continuous ranges such as $40 \leq weight(X) \leq 50$.

The probability of a query is calculated in the relevant leaf node(s). In the previous section, we introduced an algorithm that assigned ranges $r_i = [\alpha_i, \beta_i]$ and their piecewise polynomial density approximation $p_i(x)$ to a continuous SPN leaf node i. With that, we can perform inference at the leaf node itself through WMI. For example, assume a continuous leaf node for the attribute "weight" has the range $34 \leq weight(X) \leq 55$ and suppose its calculated polynomial is $-0.051 + 0.0016x$. The probability for the query $40 \leq weight(X) \leq 50$ can then be calculated using $\int_{40}^{50} -0.051 + 0.0016x \quad dx = 0.21$.

Once the leaf has processed the query its value is passed to the SPN where it is applied to the other variables of the query. We reiterate that inference using the polynomials is not performed in the SPN itself. Only the value is passed on to be interpreted. This demonstrates the built-in modularity of our approach.

Naturally, posing a query such as $(40 \leq weight(X) \leq 60)$, spanning over multiple intervals is less trivial. In this case, assume a second interval for $weight(X)$ is specified $55 \leq weight(X) \leq 77$ with a polynomial density of $0.1469 - 0.0019x$. Then, the probability of the query is calculated as $P(40 \leq weight(X) \leq 70) = P(40 \leq weight(X) \leq 55) + P(55 \leq weight(X) \leq 70) = \int_{40}^{55} -0.051 + 0.0016x \quad dx + \int_{55}^{60} 0.1469 - 0.0019x \quad dx = 0.375 + 0.291 = 0.666$. In general, as the ranges were defined to be mutually exclusive, the calculation of the probability boils down to:

1. dividing the range into $m - k$ subranges where each corresponds to a leaf node i, $\quad i \in \{1, ..., n\}$, $1 \leq k \leq m \leq n$;
2. performing inference through WMI in each leaf with the polynomial density approximation $p_i(x)$; and then
3. adding the calculated probability values to obtain the final value.

That is,

$$\int_a^b p(x)dx = \int_a^{\beta_k} p_k(x)dx + \sum_{j=k+1}^{m-1} \int_{\alpha_j}^{\beta_j} p_j(x) + \int_{\alpha_m}^{b} p_m(x)dx$$

By extension, if a query consists of multiple continuous features $X_1, ..., X_N$, with density approximations $p^1(X_1), ..., p^N(X_n)$, the probability can be calculated as:

$$\int_{a_1}^{b_1} \ldots \int_{a_N}^{b_N} p^1(x_1) \ldots p^N(x_N) dx_1 \ldots dx_N$$

$$= \int_{a_1}^{b_1} p^1(x_1)dx_1 \times \ldots \times \int_{a_N}^{b_N} p^N(x_N)dx_N.$$

When integrating the model into the SPN some SPN properties need to be considered. The root node will always calculate the likelihood of a query by taking the product of its children. Given the mutual exclusivity of a continuous feature's multiple ranges, two or more passes are required for an SPN to calculate the likelihood of a query atom over multiple intervals. In addition, queries containing multiple independent variables including combinations of continuous feature intervals and discrete query atoms, are passed separately and the product of the returned likelihoods is calculated to obtain the final value.

For simplicity, we assume that queries are limited to the following syntax:

1. Every query atom is either an interval $x \in [a, b]$ or a discrete feature $\mathrm{A} \in [a_1, ..., a_n]$.
2. A query consists of a number of conjunctions.
3. Each conjunction has to be distinct, e.g. no conjunction shares the same query atoms.

Here, (2) and (3) are not fundamental restrictions: it is easy to extend (2) by applying the rule $\Pr(A \vee B) =$

$\Pr(A) + \Pr(B) - \Pr(A \wedge B)$, and in the case of (3), linear arithmetic solvers can be used to reason about multiple constraints for the same variable: for example, $\Pr(30 \leq weight(X)) \wedge \Pr(50 \leq weight(X) < 60)$ can be resolved to $\Pr(50 \leq weight(X) < 60)$. So, (1) says computing the probability of expressions such as $x > y$ and $x + y > 2z$, where x, y, z are continuous variables, cardinality queries [Bekker et al., 2015], among others are not dealt with currently.

5 Experimental Evaluation

The goal of this section is to evaluate the merits of LearnWMISPN, and the complex query interface. Specifically, we attempt to address the following questions:

Q1 How effective (in terms of log-likelihood) is LearnWMISPN on complex mixed discrete-continuous data?

Q2 How reasonable is the learned distribution: that is, what is the order of the learned polynomials, and what is the spread of the probabilities for the underlying intervals?

Q3 How effective is the query interface in the presence of increasing query lengths?

We measured the performance of the learned SPNs on preprocessed hybrid domain datasets discussed in an independent and very recent effort [Molina et al., 2018] that introduces so-called mixed SPNs (MSPNs). Like our work, they are motivated by piecewise approximations for learning from hybrid data. However, we have strived for a full integration of SPNs and WMI. For example, we learn polynomials of the optimal order (by using BIC), whereas they only focus on piecewise constant or linear. We dynamically handle arbitrarily complex interval queries; no such interface is studied there. Perhaps most significantly, our approach neatly separates the propositional layer from the continuous aspects, especially the integration, which makes our framework generic, and applicable to any SPN learner. In particular, we are able to piggyback on LearnSPN's simple yet fast decomposition scheme.

Our performance evaluation of SPNs in conjunction with WMI was performed using three different regimes. We measured the performance of the learned SPNs from LearnWMISPN on preprocessed hybrid domain datasets. Doing so allowed us to directly compare the performance of LearnWMISPN to LearnMSPN [Molina et al., 2018], and test whether our approach with continuous distributions was effective. We then investigated whether further increasing the number of leafs per continuous feature resulted in improvements in the model. Next, measuring the complex querying capacity of WMI with SPNs was performed by recording the computation times dependent on query length. As we mentioned before, WMI is a very flexible framework for inference in hybrid domains, and as our empirical results show, our integration with SPNs has yielded a *scalable* unsupervised learning architecture that is able to handle MSPN benchmarks and many (preprocessed) UCI datasets involving hundreds of variables.

Q1 Hybrid Benchmarks: We evaluated LearnWMISPN on 11 hybrid domain datasets taken from the UCI machine learning repository [Dheeru and Karra Taniskidou, 2017]. Each dataset was composed of differing proportions of categorical, discrete, and continuous features. The diverse set of domains comprised data from financial, medical, automotive and other sectors. MSPNs were used as a comparative baseline. MSPNs proved to be a successful mixed probabilistic model [Molina et al., 2018], and given the similar nature of our objectives, a comparison was appropriate. Structure learning with LearnMSPN was performed with different variants. MSPNs were trained using the Gower distance, which is a metric over hybrid domains, with Gaussian distributional assumptions for continuous variables. MSPNs were also trained using the randomized dependency coefficient (RDC) which does not make parametric assumptions. For each model, histogram representations were compared against isometric regression models.

The dimensions of the datasets can be seen in Table 1. Given the original datasets used real values, our preprocessing results in an augmented binary matrix that has a larger variable count. In the original MSPN work, more datasets were used to measure performance, but in our case we omitted datasets which did not contain continuous features. We used the original LearnSPN training, validation, and test ratio splits, which were 75%, 10%, and 15% respectively.

Table 1: Dataset statistics

Dataset	$\lvert V\rvert$	$\lvert V'\rvert$	Train	Valid	Test
anneal-U	38	95	673	90	134
australian	15	50	517	69	103
auto	26	85	119	16	23
car	9	50	294	39	58
cleave	14	35	222	29	44
crx	15	54	488	65	97
diabetes	9	33	576	76	115
german	21	76	750	99	150
german-org	25	70	750	99	150
heart	14	35	202	27	40
iris	5	11	112	15	22

The log-likelihood results for LearnMSPN methods and LearnWMISPN are shown in Table 2. As can be seen, our model outperforms the MSPN model by a wide margin. The performance of our model results in significantly better likelihoods across the majority of datasets. In most cases, our implementation only required two bins per continuous feature, and one instance (the iris dataset) required a three bin split in order to produce a likelihood competitive

Table 2: Average test set log likelihoods for structured learning methods on hybrid datasets.

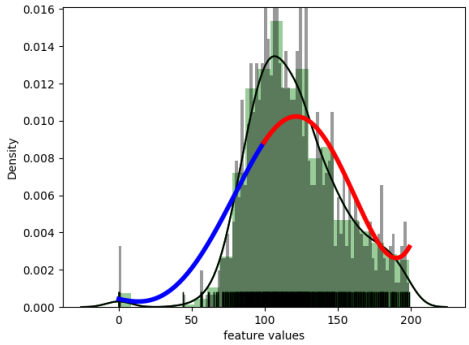
Dataset	WMI-SPN	Gower-MSPN		RDC-MSPN	
	LearnWMISPN	hist	iso	hist	iso
anneal-U	**-14.543**	-63.553	-38.836	-60.314	-38.312
australian	**-10.473**	-18.513	-30.379	17.891	-31.021
auto	**-27.126**	-72.998	-69.405	-73.378	-70.066
car	**-9.111**	-30.467	-31.082	-29.132	-30.516
cleave	**-13.829**	-26.132	-25.869	-29.132	-25.441
crx	**-10.525**	-22.422	-31.624	-24.036	-31.727
diabetes	**-6.299**	-15.286	-26.968	-15.930	-27.242
german	**-20.429**	-40.828	-26.852	-38.829	-32.361
german-org	**-21.144**	-43.611	-26.852	-37.450	-27.294
heart	**-14.875**	-20.691	-26.994	-20.376	-25.906
iris	-3.560	-3.616	-2.892	-3.446	**-2.843**

to the MSPN models.

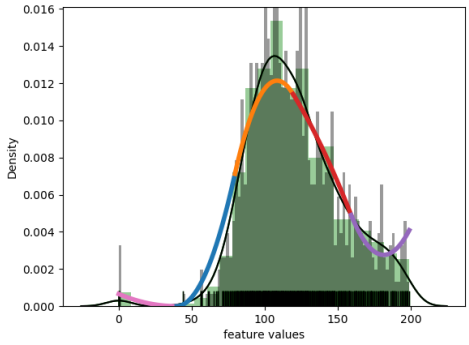
The effectiveness of the framework can be attributed to a number of factors. We observed that unsupervised binning interfaced with LearnSPN's existing structure learning performed well on the hybrid datasets. This can be thought of as piecewise constant WMISPNs. The approach is simple and thus, fast, adding negligible complexity to the original LearnSPN machinery: the effectiveness in learning binary representations and the simple counting of Boolean variable activations is inherited from LearnSPN. In addition to that, we support the learning of polynomial weights, and not just constant or linear ones. In particular, our use of the BIC criteria to choose the most optimal representation, which has bias towards smaller bin numbers and polynomial exponents.

Q2 Model complexity: When investigating the correlation of model accuracy to model complexity, we used datasets with the majority variables in the continuous domain. We also used datasets with significantly more instances to learn on, ranges being from 1024 to 17389 in order demonstrate the scalability of WMISPNs. The datasets investigated were the Cloud dataset, Statlog (Shuttle) dataset, and a subset of the MiniBooNE particle identification dataset, which were all from the UCI machine learning repository [Dheeru and Karra Taniskidou, 2017]. Our inquires below use equal width binning as a discretisation method for continuous features.

The most immediate question is what is the nature of learned polynomials? The BIC measure, as we mentioned, is used to determine the number of bins and polynomial order. So, by relaxing the criteria for the number of bins to be used, we can study the polynomials. In Table 3, we see statistics on the order of the learned polynomials for 2 vs 5 bins. The order never goes beyond 6, and in majority of the cases is ≤ 4, confirming once again that the learned orders stay manageable. Increasing the bin size favours low-order polynomials: only *australia* and *cloud* have order 6 polynomials with 5 bins.



(a) Diabetes 2 bin polynomial approximation



(b) Diabetes 5 bin polynomial approximation

Figure 4: Comparison of learned piecewise polynomial functions for feature 2(Plasma glucose concentration) of the diabetes dataset. Bin intervals are represented by alternating colors.

In Figure 4, It is evident that increased binning results in piecewise polynomial functions that can better approximate the spread of data. In the case of the diabetes dataset, at 5 bins the learned polynomial function was better able to capture the distribution of continuous feature points compared with the 2 bin approximation. The improved polynomial function approximation also lends itself to more accurate bin probabilities for SPN inference calculations. Also of note is the polynomial order for the diabetes continuous feature(Plasma glucose concentration) at 2 bins was a 4^{th} order polynomial, while at 5 bins the polynomial order was only 2.

The second question, then, is how are the probabilities spread from the learned representation? (That is, assume we learn $p_1(x)$ for $0 \leq x \leq 5$ and $p_2(x)$ for $5 \leq x \leq 10$; we consider the probabilities on computing the volumes and normalising.) Naturally, this spread depends very much on the data, but by considering multiple datasets, one can em-

pirically study the effectiveness of the learning regime. We see in Figure 6 that the representations match the characteristics of the data (e.g., spareness for some attributes, missing values), diverse as they are.

The third natural question is whether learning polynomials are beneficial at all? We mentioned earlier that unsupervised binning along with a simple binarisation scheme can be seen as a simplistic hybrid model – an instance of piecewise constant WMISPNs – for which LearnSPN suffices. Clearly such an endeavour would come at a significant loss of expressiveness, e.g., no interval querie. So, by letting BIC determine the polynomial order but explicitly setting the bin parameter, we can contrast LearnSPN and LearnWMISPN. It should be noted that as each bin range corresponds to a distribution represented by a given polynomial, further bin increases on a feature result in differing polynomial representations. (Analogously, classical SPNs will redistribute the spread of discrete probabilities.) In Figure 5, the plotted performance of average log-likelihoods over bin complexity is normalized by the number of new variables generated from equal width binning. We see that accuracy on the dataset does increase as the number of bins per feature increases, and thus LearnWMISPN yields a more accurate representation.

Dataset	Bins	2nd-Order	3rd-Order	4th-Order	5th-Order	6th-Order
australia	2	0	16.667	16.667	33.3	33.3
australia	5	16.667	33.333	16.667	16.667	16.667
auto	2	15.8	36.842	31.579	12.789	0
auto	5	73.684	15.789	10.526	0	0
german-org	2	0	33.333	0	0	66.666
german-org	5	33.333	33.333	0	33.333	0
heart	2	0	20	20	60	0
heart	5	0	60	40	0	0
iris	2	50	0	50	0	0
iris	5	75	25	0	0	0
statlog	2	42.857	0	0	0	57.143
statlog	5	28.571	57.143	0	14.289	0
cloud	2	10	10	40	10	30
cloud	5	30	50	10	0	10

Table 3: Comparison of the distributions of orders (in %) for 2 and 5 bins.

Q3 Query Interface: The capacity to query SPNs trained on continuous domains represents a significant improvement in terms of interpreting data. With regard to posing complex queries to WMISPNs, we studied the computation time for inferences. The three datasets used for measuring this are the Cloud, Statlog (Shuttle), and MiniBooNE datasets. All datasets are comprised of continuous features, with Statlog containing a single discrete feature. In order to measure complexity, we define query length of i as length$(Q_i) = q_i$ with query length of 1 corresponding to a query of form $Q_1 = (a \leq x < b)$ in the continuous case and $Q_1 = (\neg y)$ in the discrete case. It follows that a query length

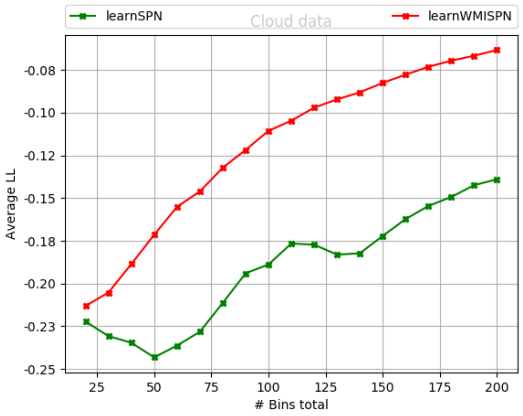


Figure 5: Instances of expanded Cloud datasets and resulting normalized log-likelihoods. Comparisons done with LearnWMISPN (red) and LearnSPN (green) [Gens and Domingos, 2013]. It should be noted that LearnWMISPN is yielding a highly granular and intricate representation. For example, at 5 bins per feature, the first bin of the first feature (the pixel mean) defined over a range of $(3.0 \leq x \leq 20.1)$ is given a density approximation that is a 3^{rd} degree polynomial, and then at 15 bins per feature, over a range of $(3.0 \leq x \leq 8.7)$ the density approximation is now a 4^{th} degree polynomial.

of two q_2 could be defined as $Q_2 = [(a \leq x < b) \wedge (\neg y)]$ and so forth. Continuous queries are defined as c^i and discrete as d^i with i representing the query count. We generated 10 random queries based on the continuous ranges for each dataset, including discrete outcomes for Statlog, and averaged the inference time. We also fixed the number of bins per continuous feature to two.

Table 4: Average query time for differing query lengths. Time per query is measured in (ns/query).

Dataset	q_1	q_2		q_3		q_4		q_5	
	c^1	c^2	c^1, d^1	c^3	c^2, d^1	c^4	c^3, d^1	c^5	c^4, d^1
cloud	1401	2154	n/a	2563	n/a	3659	n/a	4025	n/a
statlog	1299	n/a	1878	n/a	2365	n/a	2749	n/a	2982
miniboone	1168	2071	n/a	2219	n/a	2290	n/a	3048	n/a

In Table 4, we see the inference speeds on the complex queries. Overall, it is clear that complex queries do not slow down the model. For all query lengths, the time per query remains in nanosecond range. As query length increases, more time is required but overall the increase in query time is linear. The model was also capable of handling queries with mixed continuous and discrete atoms, further demonstrating the capability of WMISPNs.

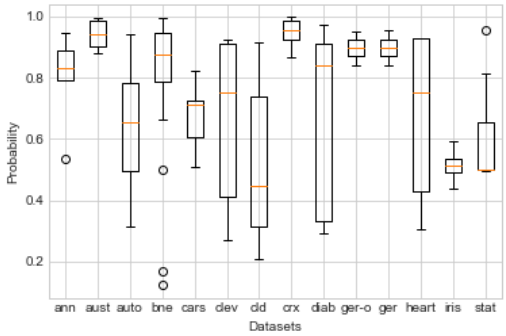


Figure 6: The spread of probabilities, with 2 bins per attribute.

6 Conclusion

Deep architectures are powerful learning paradigms that capture latent structure and have proven to be very successful in machine learning. Guessing the right architecture for complex data is challenging, and so paradigms such as SPNs are attractive alternatives in providing an *unsupervised* learning regime in addition to robust inference computations.

Here, we pushed the envelope further to consider a systematic integration of SPNs and WMI, allowing us to learn tractable, non-parametric distributions and convex combinations thereof for hybrid data, also in an *unsupervised* fashion. The integration was achieved by minimally adapting the base case for a SPN structure learning module, which makes our approach generic to a large extent. Different from our predecessors, we show for the first time how tractable distributions of arbitrary granularity can be learned, and more importantly, how to query these distributions over a rich interval syntax. Our empirical results show that our implemented system is effective, scalable and incurs very little cost for handling continuous features, all of which is very desirable for learning from big uncertain data. Challenges for the future include extending the query language with features like counting operators, which would allow us to reason about the cardinality of sets of objects in an image, thus enabling an interface for commonsensical reasoning within deep architectures.

7 Acknowledgements

This work is partly supported by the EPSRC grant 'Towards Explainable and Robust Statistical AI: A Symbolic Approach'.

References

A. Albarghouthi, L. D'Antoni, S. Drews, and A. Nori. Quantifying program bias. *arXiv preprint arXiv:1702.05437*, 2017.

F. Bacchus, S. Dalmao, and T. Pitassi. Solving #SAT and Bayesian inference with backtracking search. *J. Artif. Intell. Res. (JAIR)*, 34:391–442, 2009.

F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2002.

V. Baldoni, N. Berline, J. A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, M. Vergne, and J. Wu. A user's guide for latte integrale v1. 7.1. *Optimization*, 22:2, 2014.

J. Bekker, J. Davis, A. Choi, A. Darwiche, and G. Van den Broeck. Tractable learning for complex probability queries. In *Advances in Neural Information Processing Systems*, pages 2242–2250, 2015.

V. Belle, A. Passerini, and G. Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2770–2776, 2015.

Y. Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172 (6-7):772–799, 2008.

D. Chistikov, R. Dimitrova, and R. Majumdar. Approximate counting in smt and value estimation for probabilistic programs. In *TACAS*, volume 9035, pages 320–334. ACTA INFORMATICA, 2015.

D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

D. Fierens, G. Van den Broeck, I. Thon, B. Gutmann, and L. D. Raedt. Inference in probabilistic logic programs using weighted CNF's. In *UAI*, pages 211–220, 2011.

R. Gens and P. Domingos. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pages 873–880, 2013.

D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

W. Hsu, A. Kalra, and P. Poupart. Online structure learning for sum-product networks with gaussian leaves. *arXiv preprint arXiv:1701.05265*, 2017.

S. L. Lauritzen and F. Jensen. Stable local computation with conditional gaussian distributions. *Statistics and Computing*, 11(2):191–203, 2001.

Y. Liang, J. Bekker, and G. Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

A. Molina, S. Natarajan, and K. Kersting. Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In *AAAI*, pages 2357–2363, 2017.

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

P. Morettin, A. Passerini, R. Sebastiani, et al. Efficient weighted model integration via smt-based predicate abstraction. In *IJCAI*, pages 720–728, 2017.

K. P. Murphy. A variational approximation for bayesian networks with discrete and continuous latent variables. In *UAI*, pages 457–466, 1999.

D. Nitti, I. Ravkic, J. Davis, and L. De Raedt. Learning the structure of dynamic hybrid relational models. In *22nd European Conference on Artificial Intelligence (ECAI) 2016*, volume 285, pages 1283–1290, 2016.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 337–346, 2011.

I. Ravkic, J. Ramon, and J. Davis. Learning relational dependency networks in hybrid domains. *Machine Learning*, 100(2-3):217–254, 2015.

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

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

P. P. Shenoy and J. C. West. Extended shenoy–shafer architecture for inference in hybrid bayesian networks with deterministic conditionals. *International Journal of Approximate Reasoning*, 52(6):805–818, 2011.

S. Speichert. Learning Hybrid Relational Rules with Piecewise Polynomial Weight Functions for Probabilistic Logic Programming. Master's thesis, The University of Edinburgh, Scotland, 2017.

L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

A. Vergari, N. Di Mauro, and F. 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.

X. Yang, J. Cao, and W. Yu. Exponential synchronization of memristive cohen–grossberg neural networks with mixed delays. *Cognitive neurodynamics*, 8(3):239–249, 2014.