Lecture Notes

Probabilistic Circuits: Representation and Inference

YooJung Choi

Antonio Vergari

Guy Van den Broeck
Computer Science Department
University of California
Los Angeles, CA, USA

Abstract

These lecture notes accompany the AAAI 2020 tutorial on probabilistic circuits, by Antonio Vergari, Robert Peharz, YooJung Choi, and Guy Van den Broeck. They cover the first half of the tutorial, that is, the motivation of tractable models, the probabilistic circuit representation, and its inference algorithms.

This is a working document; updates will be made available at `starai.cs.ucla.edu/papers/LecNoAAAI20.pdf`. The tutorial slides covering additional topics such as learning of probabilistic circuits and a survey of their applications are available at `starai.cs.ucla.edu/slides/AAAI20.pdf`. Kindly send corrections and feedback to `yjchoi@cs.ucla.edu`.

1. Probabilistic Inference: Models, Queries and Tractability

In the following, we will formalize the idea of performing probabilistic inference as computing quantities of interest of a probability distribution by *querying probabilistic models*, here playing the role of compact representations of probability distributions. We will then categorize these queries into families of functions to characterize the complexity of different probabilistic inference tasks. We will later discuss what makes a family of probabilistic model tractable w.r.t. a class of queries. We assume the readers to be familiar with the basic concepts and rules of probability calculus while referring them to (Koller and Friedman, 2009; Feller, 2008; Rosenthal, 2006) for background.

Notation. We use uppercase letters for random variables (RVs), e.g., X, Y, and lowercase ones for their corresponding assignments e.g., x, y. Analogously, indexed sets of RVs are denoted by bold uppercase letters, e.g., $\mathbf{X}$, $\mathbf{Y}$, and their joint values by the corresponding lowercase ones, e.g., $\boldsymbol{x}$, $\boldsymbol{y}$. A specific RV in a set, and a value in a joint assignment, is indexed by a subscript, e.g., X_i and x_i, respectively. Concatenation $\mathbf{XY}$ denotes the union of disjoint sets. Let $\mathsf{val}(X)$ define the set of possible values that RV X can assume, i.e., its domain. Analogously, $\mathsf{val}(\mathbf{X})$ indicates the cartesian product $\mathsf{val}(X_1) \times \mathsf{val}(X_2) \times \ldots \times \mathsf{val}(X_k)$ for $\mathbf{X} = \{X_i\}_{i=1}^k$. Let $p(\mathbf{X})$ be a probability distribution of RV $\mathbf{X}$ and let $\mathsf{supp}(\mathbf{X})$ be the support of $p(\mathbf{X})$, i.e., $\{\boldsymbol{x} \in \mathsf{val}(\mathbf{X}) \mid p(\boldsymbol{x}) > 0\}$. For example, a Boolean variable X has $\mathsf{supp}(\mathbf{X}) = \{0, 1\}$; we adopt the shorthand X and $\neg X$ for the indicator functions $[\![X = 1]\!]$ and $[\![X = 0]\!]$, respectively.

1.1 Probabilistic Queries

Intuitively, a probabilistic model can be seen as a *black-box* that is able to answer questions we pose about the uncertainty around some configurations of the world (Koller and Friedman, 2009). That is, asking about some quantities of interest of the joint probability distribution it encodes. For instance, one might ask one probability distribution for the probability mass or density associated to certain observed configurations (also called *evidence*) alternatively for its mode or one of its moments, e.g., its mean value of probability. Following the terminology adopted in the (probabilistic) databases literature, we name these questions ***queries*** (Koller and Friedman, 2009; Suciu et al., 2011; Van den Broeck et al., 2017).

The traffic jam example. We demonstrate how probabilistic inference can be formalized into query answering by building a simple real-life scenario in which such queries are pivotal to support decision making under uncertainty. Imagine being a commuter in New York City. To decide which route to take each day to work and avoid traffic jam you could query the probabilistic model embedded in your GPS navigator. Among the several queries you might want to ask your navigator, there might be:

q_1 : ***"What is the probability that on Monday there will be a traffic jam on 5th Avenue?"***

q_2 : ***"Which day is the most likely for there to be a traffic jam on my route to work?"***

Both q_1 and q_2 are queries trying to quantify the uncertainty about specific configurations of the world, captured by the probabilistic model m inside your navigator. That is, m encodes a joint probability distribution $p_{\mathsf{m}}(\mathbf{X})$ over a collection of random variables $\mathbf{X}$. Let $\mathbf{X} = \{\mathsf{D}, \mathsf{T}, \mathsf{J}_{\mathsf{str}_1}, \ldots, \mathsf{J}_{\mathsf{str}_K}\}$ where D is a categorical RV describing the Day with $\mathsf{val}(\mathsf{D}) = \{\mathsf{Mon}, \mathsf{Tue}, \mathsf{Wed}, \mathsf{Thu}, \mathsf{Fri}, \mathsf{Sat}, \mathsf{Sun}\}$; T is a continuous variable with $\mathsf{val}(\mathsf{T}) = [0, 24)$ indicating the Time of the day with arbitrary precision. Lastly, let $\{\mathsf{J}_{\mathsf{str}_i}\}_{i=1}^{K}$ be a finite number of binary RVs, each indicating the presence of a traffic jam on the i-th street.

Note that in this example, as it frequently is in real-world scenarios, the joint probability distribution is defined over both continuous ***and*** discrete RVs. Moreover, these variables can be heterogeneous in nature; that is, each might represent a different likelihood model, e.g., Gaussian, Bernoulli, Poisson distibutions, etc.. In these lecture notes, we will always assume that these joint distributions, even when conditioned on some evidence, define proper probability masses (resp. probability densities) over the corresponding subset of discrete RVs (resp. continuous RVs). To lighten the notation, we will denote with p either a probability mass, a probability density or a mixed joint distribution making the distinction explicit only if ambiguous from context.

In essence, a query is a function of the probability distribution encoded in a probabilistic model, whose output is a quantity characterizing or characterized by that probability distribution. For instance, the output of query q_1 in the our example is defined as the probability mass of a partially-observed configuration $\boldsymbol{e} = (\mathsf{D} = \mathsf{Mon}, \mathsf{J}_{\mathsf{5th}} = 1)$ (also called *partial evidence*) for a subset of the RVs in $\mathbf{X}$, $\mathbf{E} \subset \mathbf{X}$:

$$p_{\mathsf{m}}(\mathsf{D} = \mathsf{Mon}, \mathsf{J}_{\mathsf{5th}} = 1). \tag{1}$$

This is an example of a classical *marginal* query, where to compute q_1 we are required to marginalize out the the RVs not in $\mathbf{E}$. On the other hand, the output of query q_2 returns the mode of the distribution obtained from p_{m} after observing a more complex kind of evidence, involving the disjunction of simpler traffic jam events:

$$\arg\max_{\mathsf{d}} p_{\mathsf{m}}(\mathsf{D} = \mathsf{d} \wedge \bigvee_{i \in \mathsf{route}} \mathsf{J}_{\mathsf{str}_i}). \tag{2}$$

As one might intuit at this point, computing q_2 must be at least as hard as computing q_1. Computing q_2 requires performing marginalization as for q_1 (as we do not care about the variable T) in addition to performing maximization and dealing with more complex logical constraints.

Query classes. Informally, a *class of queries* comprises of all the queries that share the same "structure", and hence present the same computational challenges. More formally, a query class, denoted as $\mathcal{Q}$, is a collection of functions that operate on probability distributions and that can be characterized by the mean of their inputs, outputs and/or by the operations their computation involves. As we will see later, query classes play a crucial role in inducing classes of tractable probabilistic models. Indeed, *tractability is not a universal property*, but it is specific to classes of queries: a model might be amenable to efficiently compute queries from one class but not from another.

In the following, we review the query classes that are most commonly used in probabilistic inference and learning: complete evidence (EVI) queries, marginals (MAR), conditionals (CON) and maximum a priori (MAP), and briefly tease more advanced classes including marginal MAP (MMAP) queries.

Definition 1 (Complete evidence queries (EVI)) *Let $p(\mathbf{X})$ be a joint distribution over RVs $\mathbf{X}$. The class of complete evidence queries $\mathcal{Q}_{\mathsf{EVI}}$ is the set of queries that compute the probability $p(\mathbf{X} = \boldsymbol{x})$ for any complete configuration (also called complete evidence) $\boldsymbol{x} \in \mathsf{val}(\mathbf{X})$ for RVs $\mathbf{X}$.*

For conciseness, we often write $p(\mathbf{X} = \boldsymbol{x})$ as $p(\mathbf{x})$.

Example 1 *Consider the probability distribution p_{m} defined over the RVs $\mathbf{X}$ as in the traffic jam example. Then the question "What is the probability that at 12 o'clock on Monday there will be a traffic jam only on 5th Avenue?" can be answered by the* EVI *query:*

$$p_{\mathsf{m}}(\mathsf{D} = \mathsf{Mon}, \mathsf{T} = 12.0, \mathsf{J}_{\mathsf{5th}} = 1, \bigwedge_{j \neq \mathsf{5th}} \mathsf{J}_j = 0)$$

As answering an EVI query corresponds to compute the likelihood of model m for a certain complete configuration, the class of queries $\mathcal{Q}_{\mathsf{EVI}}$ is at the core of maximum likelihood learning of probabilistic models (Koller and Friedman, 2009).

Definition 2 (Marginal queries (MAR)) *Let $p(\mathbf{X})$ be a joint distribution over RVs $\mathbf{X}$. The class of marginal queries $\mathcal{Q}_{\mathsf{MAR}}$ is the set of queries that compute probabilities of the*

following form:

$$p(Z_1 \in \mathcal{I}_1, Z_2 \in \mathcal{I}_2, \ldots, Z_k \in \mathcal{I}_k, \mathbf{E} = \boldsymbol{e}) = \int_{\mathcal{I}_1 \times \mathcal{I}_2 \times \ldots \times \mathcal{I}_k} p(\mathbf{Z}, \boldsymbol{e}) \; d\mathbf{Z} \tag{3}$$

$$= \int_{\mathcal{I}_1} \int_{\mathcal{I}_2} \ldots \int_{\mathcal{I}_k} p(Z_1, Z_2, \ldots, Z_k, \boldsymbol{e}) \, dZ_1 dZ_2 \ldots dZ_d \tag{4}$$

where $\boldsymbol{e} \in \mathsf{val}(\mathbf{E})$ *is a partial configuration (also called partial evidence) for any subset of RVs* $\mathbf{E} \subseteq \mathbf{X}$*, and* $\mathbf{Z} = \mathbf{X} \setminus \mathbf{E}$ *is the set of* k *RVs integrated over intervals* $\{\mathcal{I}_i\}_{i=1}^{k}$ *each of which is defined over the domain of its corresponding RV in* $\mathbf{Z}$*:* $\mathcal{I}_i \subseteq \mathsf{val}(Z_i)$ *for* $i = 1, \ldots, k$*.*

Note that for the sake of simplicity, in the above definition we adopted the more general integral symbol to subsume both multi-dimensional finite integrals for continuous RVs and (potentially infinite) nested summations for discrete ones. We will adopt this notational convention consistently from here on.

Moreover, note that the above definition for MAR queries generalizes the more usual one where the RVs $\mathbf{Z}$ are integrated (summed) over their whole domains, i.e., $\mathcal{I}_i = \mathsf{val}(Z_i)$. Furthermore, $\mathcal{Q}_{\mathsf{MAR}}$ also include queries on the joint cumulative distribution function (CDF) of a distribution p_{m}, which share with the classical MAR queries the same computational effort–solving multi-dimensional definite integrals (or summations), a task at the core of probabilistic inference. Lastly, it naturally follows that $\mathcal{Q}_{\mathsf{EVI}} \subset \mathcal{Q}_{\mathsf{MAR}}$.

Example 2 *Consider the probability distribution* p_{m} *defined over the RVs* $\mathbf{X}$ *as in the traffic jam example. Then question* q_1 *can be answered by the* MAR *query as defined in Equation 1.*

Queries in $\mathcal{Q}_{\mathsf{MAR}}$ represent the evaluation of a distribution projected over a subspace of the domain. This kind of probabilistic queries arises naturally when one has to probabilistically reason over sets of RVs that are not accessible at hand or can be flagged as "don't care", e.g., when dealing with features over data with missing values.

Definition 3 (Conditional queries (CON)) *Let* $p(\mathbf{X})$ *be a joint distribution over RVs* $\mathbf{X}$*. The class of conditional queries* $\mathcal{Q}_{\mathsf{CON}}$ *is the set of queries that compute probabilities* $p(\mathbf{Q} = \boldsymbol{q} \mid \mathbf{E} = \boldsymbol{e})$*, where* $\boldsymbol{e} \in \mathsf{val}(\mathbf{E}), \boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ *are partial configurations for an arbitrary partitioning of the RVs* $\mathbf{X}$*, i.e.,* $\mathbf{Q} \cup \mathbf{E} = \mathbf{X}$ *and* $\mathbf{Q} \cap \mathbf{E} = \emptyset$*.*

Example 3 *Consider the probability distribution* p_{m} *defined over the RVs* $\mathbf{X}$ *as in the traffic jam example. Then the question "What is the probability that there will be a traffic jam only on 5th Avenue on Monday at 12 o'clock?" can be answered by the* CON *query:*

$$p_{\mathsf{m}}(\mathsf{J}_{\mathsf{5th}} = 1, \bigwedge_{j \neq \mathsf{5th}} \mathsf{J}_j = 0 \mid \mathsf{D} = \mathsf{Mon}, \mathsf{T} = 12.0)$$

One can easily define the class $\mathcal{Q}_{\mathsf{CON}}$ in terms of $\mathcal{Q}_{\mathsf{EVI}}$ and $\mathcal{Q}_{\mathsf{MAR}}$ by noting that any conditional query can be rewritten as the ratio[1]:

$$p(\mathbf{Q} = \boldsymbol{q} \mid \mathbf{E} = \boldsymbol{e}) = \frac{p(\mathbf{Q} = \boldsymbol{q}, \mathbf{E} = \boldsymbol{e})}{p(\mathbf{E} = \boldsymbol{e})}.$$

1. Here we are making the commonly adopted assumption that the marginal normalization constant $p(\boldsymbol{e})$ is always positive.

Instead of evaluating masses or densities of events, we might be interested in properties of the distribution such as its *mode*. More generally, we might be interested in the mode of a distribution *after* we condition on some specific event.

Definition 4 (Maximum a posteriori queries (MAP)) *Let $p(\mathbf{X})$ be a joint distribution over RVs $\mathbf{X}$. The class of maximum a posteriori queries $\mathcal{Q}_{\mathsf{MAP}}$ is the set of queries that compute:*

$$\arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} p(\mathbf{Q} = \boldsymbol{q} \mid \mathbf{E} = \boldsymbol{e}) = \arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} p(\mathbf{Q} = \boldsymbol{q}, \mathbf{E} = \boldsymbol{e}) \tag{5}$$

where $\boldsymbol{e} \in \mathsf{val}(\mathbf{E}), \boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ are partial configurations for an arbitrary partitioning of the RVs $\mathbf{X}$, i.e., $\mathbf{Q} \cup \mathbf{E} = \mathbf{X}$ and $\mathbf{Q} \cap \mathbf{E} = \emptyset$.

Note that the right-hand side of Equation 5 follows from the fact that maximization is not affected by the normalization constant $p(\boldsymbol{e})$.

Example 4 *Consider the probability distribution p_{m} defined over the RVs $\mathbf{X}$ as in the traffic jam example. Then the question "Which combination of roads is most likely to be jammed on Monday at 9:30am?" can be answered by the following* MAP *query:*

$$\arg\max_{\mathsf{j}} p_{\mathsf{m}}(\mathsf{J} = \mathsf{j}, \mathsf{T} = 9.5.)$$

Sometimes one might be interested not only in the mode value of the distribution but also in its associated probability or density. To categorize these queries we can introduce the class in why the arg max operation is replaced by a simple maximization.

In the Bayesian networks literature, MAP queries are often referred to as *most probable explanation* (*MPE*) queries (Darwiche, 2009) and MAP refers to marginal MAP queries (Koller and Friedman, 2009), where one performs maximization on some subset of $\mathbf{Q}$, marginalizing over the remaining variables (Koller and Friedman, 2009).

Advanced query classes. More classes of queries can be defined by "combining" the previous ones. This is the case for marginal MAP (MMAP) queries, involving marginalization over one subset of RVs and maximization over another one. An example of a MMAP query is shown at the beginning in the form of question q_2 for the traffic jam example. More generally, in order to support complex decision making in the real-world, more sophisticated probabilistic inference routines than EVI, MAR, CON may be required. Examples of more advanced queries include computing the probability of an event described as a complex logical sentence (e.g., involving disjunctions) (Choi et al., 2015); computing expectations of probabilistic predictive models (Khosravi et al., 2019) and information theoretic quantities of a distribution such as its entropy or the Kullback-Leibler divergence between distributions (Liang and Van den Broeck, 2017).

Nevertheless, these "simple" classes already provide significant challenges to be answered exactly by the current landscape of probabilistic models, as we will discuss next.

1.2 Tractable Probabilistic Inference

When we say a probabilistic model is a tractable probabilistic model (TPM) we are implicitly expecting this model to provide two types of guarantees. The first is that the model is able to perform *exact* inference[2]. That is, its output given certain queries is faithful to the distribution the model encodes, and no approximations are involved in obtaining it. The second one is that the computation of such a query can be carried out *efficiently*. The next definition better formalizes these two desiderata for classes of models and queries.

Definition 5 (Tractable probabilistic inference) *A class of queries* $\mathbf{Q}$ *is* **tractable** *on a family of probabilistic models* $\mathcal{M}$ *iff any query* $q \in \mathbf{Q}$ *on a model* $m \in \mathcal{M}$ *can be computed in time* $\mathcal{O}(poly(|m|))$*. We also say that* $\mathcal{M}$ *is a tractable representation for* $\mathbf{Q}$*.*

In Definition 5, the concept of efficiency translates to polytime complexity w.r.t. the size of models in a class, $|m|$. One can express the model size of traditional probabilistic graphical models like Bayesian networks (BNs) and Markov random fields (MRFs) in terms of the size of their factors, and hence their *treewidth* (Darwiche, 2009; Koller and Friedman, 2009).

For models represented as computational graphs, such as in neural density estimators (Papamakarios et al., 2019) and our probabilistic circuits, model size directly translates to the number of edges in the graph. As it will be clear in later sections, for circuits it will be possible to encode also classical high treewidth BN and MRF distributions in compact graphs.

Moreover, from Definition 5 it follows that tractability is not an absolute property, but can be defined for a family of models only *w.r.t. a class of queries*: ***Tractability is a spectrum***. Indeed, a tractable representation for a query class $\mathcal{Q}'$ might not admit polynomial time inference for another class $\mathcal{Q}''$. For a model class, we define its *tractable band* as the set of query classes for which the model class is a tractable representation.

While looking for simple distributions guaranteeing large tractable bands, most univariate and unimodal parametric distributions are good candidates for tractable EVI, MAR and MAP by design. Indeed, it is generally the case that their parametric form lets you compute pointwise masses or densities in constant-time; deliver a normalized distribution, hence its partition function equals one; and specify a mode in closed form. These is true for well-known distributions such as Bernoulli, Categorical, Gaussian, Poisson, etc., which can be represented as exponential families.[3] When extended to the multivariate case, some of these distributions retain tractability. This is the case of the omnipresent multivariate Gaussian, allowing now marginalization in time cubic in $|\mathbf{X}|$ and constant-time maximization by design (their mean remains their mode).

Moving beyond these parametric forms and to general multivariate distributions, we are going to look at two powerful ideas to compose simpler models: *factorizations* and

2. As such, we are ruling out the discourse on deep generative models like GANs (Goodfellow et al., 2014) and RAEs (Ghosh et al., 2019), which do not have an explicit density model, and VAEs (Kingma and Welling, 2013) which even by having a well-defined density, cannot compute it exactly.
3. When dealing with generalized MAR queries as in Definition 2 we will need to access the corresponding CDF for these distributions. Sometimes this might not be computable in closed form, as for Gaussians. .

mixtures. Both act as "meta model" classes, i.e., induces a model class give a collection of "base" model classes. Note that it is not required that the models used as the mixture components belong to the same model class.

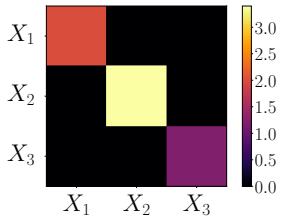
Definition 6 (Factorized models) *Consider the probabilistic model* m *encoding a probability distribution over a collection of RVs* $\mathbf{X} = \bigcup_{i=1}^{k} \mathbf{X}_i$ *partitioned into disjoint sets* $\mathbf{X}_i \cap \mathbf{X}_j = \emptyset$ *for any* $i \neq j$ *in* $1, \ldots, k$. m *is said to be a factorized model if*

$$p_{\mathsf{m}}(\mathbf{X}) = \prod_{i=1}^{k} p_{\mathsf{m}_i}(\mathbf{X}_i) \tag{6}$$

where each m_i *is a base probabilistic model over RVs* $\mathbf{X}_i$.

A classical example of factorized models are fully-factorized distributions.

Example 5 (Fully factorized distributions) *Consider a multivariate Gaussian* $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ *over RVs* X_1, X_2, X_3 *with mean* $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)$ *and diagonal covariance matrix* $\Sigma = \mathsf{diag}(\sigma_1, \sigma_2, \sigma_3)$ *as depicted on the left.*



$$\begin{aligned} p(X_1, X_2, X_3) &= p(X_1)p(X_2)p(X_3) = \\ &= \mathcal{N}(\mu_1, \sigma_1)\mathcal{N}(\mu_2, \sigma_2)\mathcal{N}(\mu_3, \sigma_3) \end{aligned}$$

Then its joint density $p(X_1, X_2, X_3)$ *can be fully factorized as in the expression on the right.*

A nice property of factorized models is that they preserve the tractable band of their base models. Let $\mathcal{M}^{\mathsf{FM}}$ be the meta-class of factorized models over RVs $\mathbf{X}$ and comprising some base model classes.

If all base model classes are tractable for $\mathcal{Q}_{\mathsf{EVI}}$, then $\mathcal{M}^{\mathsf{FM}}$ is also tractable for EVI queries. This follows from evaluating Equation 6 in time linear to the number of base models. Furthermore, if all base model classes are tractable for $\mathcal{Q}_{\mathsf{MAR}}$, then $\mathcal{M}^{\mathsf{FM}}$ is also tractable for MAR queries as the larger multidimensional integral of Equation 4 would factorize as

$$\int_{\mathcal{I}_1} \ldots \int_{\mathcal{I}_k} p(\mathbf{Z}_1, \ldots, \mathbf{Z}_k, \boldsymbol{e})\, dZ_1 \ldots dZ_d = \int_{\mathcal{I}_1} p(\mathbf{Z}_1, \boldsymbol{e}_1) d\mathbf{Z}_1 \ldots \int_{\mathcal{I}_k} p(\mathbf{Z}_k, , \boldsymbol{e}_k) d\mathbf{Z}_k$$

where each partition of RVs $\mathbf{X}_i$ individuates a corresponding partition $\mathbf{Z}_i, \boldsymbol{e}_i$ over the RVs to be marginalized and those in the partial evidence, and each interval $\mathcal{I}_i$ is defined accordingly. Analogously, if all base model classes are tractable for $\mathcal{Q}_{\mathsf{MAP}}$, then $\mathcal{M}^{\mathsf{FM}}$ is also tractable for MAP queries as the joint maximization problem of Equation 5 can be decomposed in smaller ones which can be solved independently:

$$\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} p(\mathbf{Q} = \boldsymbol{q}, \mathbf{E} = \boldsymbol{e}) = \max_{\boldsymbol{q}_1 \in \mathsf{val}(\mathbf{Q}_1)} p(\mathbf{Q}_1 = \boldsymbol{q}_1, \boldsymbol{e}_1) \times \ldots \times \max_{\boldsymbol{q}_k \in \mathsf{val}(\mathbf{Q}_k)} p(\mathbf{Q}_k = \boldsymbol{q}_k, \boldsymbol{e}_k)$$

where again the factors over RVs $\{\mathbf{X}_i\}_{i=1}^k$ induce a corresponding partitioning $\{\mathbf{Q}_i, \boldsymbol{e}_i\}_{i=1}^k$ over query RVs $\mathbf{Q}$ and evidence $\boldsymbol{e}$.

Models in $\mathcal{M}^{\mathsf{FM}}$ are able to preserve large tractable bands because they assume statistical independence among all RVs. In turn, independence allows to "break" larger problems into smaller ones and which can be solved in parallel. Clearly, modeling sets of RVs to be independent might be a too restrictive assumption for real-world scenarios, since we might discard important dependencies among those RVs. That is, $\mathcal{M}^{\mathsf{FM}}$ is not *expressive* enough to represent all possible probability distributions. Nevertheless, factorization suggests us a powerful *divide et impera* modeling principle which will be one of the powerful and pivotal ideas in the framework of probabilistic circuits in the next section.

What fully factorized models lack, expressiveness, can be increased by gradually introducing back dependencies among the RVs. One way to do that in the framework of classical probabilistic graphical models is to add edges in the graph. While we do so, however, we might reduce the tractable band for these new, more expressive classes of PGMs, as we increase their treewidth. It is very well known, for instance, that the complexity of computing MAR queries in discrete PGMs is exponential in their treewidth Cooper (1990); Roth (1996); Koller and Friedman (2009).

An alternative way to enhance the expressiveness of a model class is to add dependencies via *mixture models*. Analogous to factorization, mixture models act as meta-models and, again, it is not required that the models used as the mixture components belong to the same model class. However, to deal with a well-defined joint distribution we would require all the mixture components to be distributions over the same set of RVs.

Definition 7 (Mixture models) *Let $\{p_{\mathsf{m}_i}\}_{i=1}^k$ a finite collection of probabilistic models, each defined over the same collection of RVs $\mathbf{X}$. A mixture model is the probabilistic model defined as the convex combination*

$$p_{\mathsf{m}}(\mathbf{X}) = \sum_{i=1}^{k} w_i p_{\mathsf{m}_i}(\mathbf{X}) \tag{7}$$

for a set of positive weights (called the mixture parameters) $w_i > 0$, $i = 1, \ldots, k$ and $\sum_{i=1}^k w_i = 1$.

Mixtures increase the expressiveness of the base model classes while still preserving tractable inference for some query classes. If the component model classes are tractable representations for $\mathcal{Q}_{\mathsf{EVI}}$, then the induced mixture model class is also tractable for $\mathcal{Q}_{\mathsf{EVI}}$ since the complexity of computing EVI queries scales linearly in k according to Equation 7.

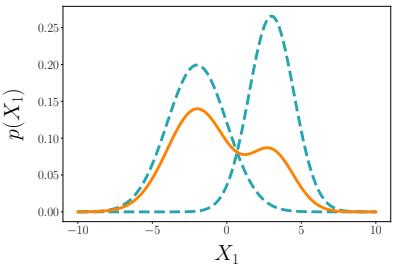
Furthermore, if the components model classes are tractable representations for $\mathcal{Q}_{\mathsf{MAR}}$, the mixture model class is also tractable for MAR queries, as one can "push" the outer integration in Equation 4 to the mixture component models as the following:

$$\int_{\mathcal{I}_1} \cdots \int_{\mathcal{I}_k} \sum_{i=1}^{k} w_i p_{\mathsf{m}_i}(\mathbf{Z}_1, \ldots, \mathbf{Z}_k, \boldsymbol{e})\, dZ_1 \cdots dZ_d = \sum_{i=1}^{k} w_i \int_{\mathcal{I}_1} \cdots \int_{\mathcal{I}_k} p_{\mathsf{m}_i}(\mathbf{Z}_1, \ldots, \mathbf{Z}_k, \boldsymbol{e})\, dZ_1 \cdots dZ_d.$$

However, even if the components model classes are tractable representations for $\mathcal{Q}_{\mathsf{MAP}}$, the induced mixture class is *not* tractable for $\mathcal{Q}_{\mathsf{MAP}}$. One can interpret a mixture model

by associating a categorical latent variable (LV) Z with k possible states that acts as a switch in selecting the mixture components. This allows us to see why $\mathcal{Q}_{\mathsf{MAP}}$ is hard for mixture models over RVs $\mathbf{X}$: maximization over $\mathbf{X}$ requires to first marginalize Z and hence corresponds to performing MMAP inference (de Campos, 2011).

Example 6 (Gaussian mixture models) *Consider the mixture model (orange) of two univariate Gaussians $\mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{N}(\mu_2, \sigma_2)$ (blue, dotted) as depicted on the left.*



$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1) = \\ = w_1 \mathcal{N}(\mu_1, \sigma_1) + w_2 \mathcal{N}(\mu_2, \sigma_2)$$

Then its joint density $p(X_1)$ can be expressed as the weighted sum on the right for the two positive real weights $w_1 = 0.8$ and $w_2 = 0.2$.

Mixtures, along with factorization, will be a powerful modeling principle for probabilistic circuits. As factorization helps reduce the complexity of inference by decomposing it into simpler problems, mixtures will serve the purpose of making the resulting model class expressive. For continuous RVs it is very well known that a sufficiently large mixture of (multivariate) Gaussians can approximate any (multivariate) continuous density arbitrarily well. Analogously, such is the case for discrete distributions when using factorized univariate distributions instead of Gaussians. However, these properties are valid asymptotically and therefore require a potentially exponential number of mixture components. We overcome this issue by stacking mixtures in multiple layers, as in *deep* (also known as hierarchical) mixture models.

2. Probabilistic Circuits: Representation

We now introduce probabilistic circuits as a general and unified computational framework for tractable probabilistic modeling. This serves two major purposes.

The first one is to unify the disparate formalisms proposed so far in the literature for tractable models. Probabilistic circuits reconcile and abstract from the different graphical and syntactic representations of recently introduced formalisms such as arithmetic circuits (Darwiche, 2003), sum-product networks (Poon and Domingos, 2011), cutset networks (Rahman et al., 2014) and probabilistic sentential decision diagrams (Kisa et al., 2014). Additionally, more classical tractable models such as treewidth-bounded probabilistic graphical models can be naturally cast in the probabilistic circuits framework by leveraging knowledge compilation (Darwiche, 2009).

The second purpose of the probabilistic circuit framework is to enable reasoning over the tractable band of a model class, represented as a circuit, in terms of some well-defined structural properties only. In turn, this allows for a deeper theoretical understanding of tractable probabilistic representations at large.

2.1 Computational graphs as tractable models

Probabilistic circuits define joint probability distributions in a recursive way, by means of a graph formalism. Even if they are *probabilistic* models expressed via a *graphical* formalism, probabilistic circuits are *not* probabilistic graphical models (PGMs) (Koller and Friedman, 2009). In fact, differently from PGMs, circuits have a clear *operational semantics*. That is, nodes in their graph structure directly represent how to evaluate the probability distributions they encode, i.e., how to answer probabilistic queries. In essence, probabilistic circuits are *computational graphs*. As such, they can also be interpreted as peculiar neural networks whose computational units abide some special constraints.

To begin, consider the smallest computational graph of this kind, consisting of a single node. This single node can represent a whole probability distribution over some RVs. The computation it performs, i.e., the output it emits given some input, would be determined by the query class considered and parametric form of the distribution it encodes.

Example 7 (Tractable densities as computational units) *Consider a unit encoding a Gaussian density $p(X) = \mathcal{N}(X; \mu, \sigma)$ as represented on the left as a circle and labeled by its RV.*

Then to answer some EVI *query, when it is fed some observed configuration $X = .74$ as evidence (orange), it will output the corresponding pdf $\mathcal{N}(x = .74; \mu, \sigma) = .33$ (blue).*

Since a computational node defined in this way effectively acts as a black-box encapsulating a distribution function, this formalism is quite versatile. First, we do not need to switch node type to answer queries from different classes. It would suffice to evaluate the encoded distribution accordingly: e.g., evaluating a pointwise density for EVI as in Example 7, marginalizing it over some interval for MAR, or returning its mode to answer MAP queries. Second, we can plug any out-of-the-box probability distribution as long as it is a tractable representation for the query class at hand. Moreover, note that we are not limited to normalized distributions, we just need the function encoded into a input unit to be non-negative and assume it to be tractable for MAR to readily obtain its partition function.

Clearly, if we were able to model all tractable distributions by a single node, we would not need a whole framework to represent them! We can however assume to always be able to encode simple distribution functions into these computational units. This idea will constitute the base case of our recursion. Building on this, it is natural to look for computational graphs that represent ways to compose other distributions. In the following, we look at the meta-model classes we introduced in the previous section– factorization and mixtures–to build our recursion rule.

We can represent factorization by introducing computational units that perform products. As the name suggests, they output the product of their inputs.

Example 8 (Factorizations as product units) *Consider the factorized multivariate Gaussian shown in Example 5. The computational graph below (left), comprising three input units feeding a product node, represents the factorized density of Example 5.*

To evaluate a probabilistic query, the output of the product node will the the product of the output of the input units. E.g., for an EVI *query, if the input units will output the density values in blue on the right, the circuit output will be the one in orange.*

Distributions from mixture models can also be easily represented as computational graphs. In order to do so we introduce sum nodes as computational units that compute weighted averages of the input they are fed. As a graphical convention, we represent the weights appearing in this computation as attached to the edges connecting the sum to its inputs.

Example 9 (Mixtures as sum units) *Consider the mixture of two Gaussians from Example 6. The computational graph below (left), comprising an input unit for each mixture component connected to the output sum node via an edge weighted by the mixture weight, represents the mixture density of Example 6, right.*

To evaluate the circuit for a query, the output of the sum node will the the sum of the output of the input units weighted by their respective edge parameters. E.g., for an EVI *query, if two input units output the density values in blue on the right, the circuit output is their convex combination in orange.*

We have now introduced the main ingredients of the probabilistic circuit framework: computational units encoding tractable distributions, product units for factorizations and sums for mixtures. Devising rules to compose them will yield the framework of probabilistic circuits.

2.2 Probabilistic circuits: structure, parameters

We now provide more rigorous definition of the syntax of probabilistic circuits as a whole. We start in a top-down fashion, by defining what is a structure and what are the parameters of a probabilistic circuit.

Definition 8 (Probabilistic circuits (PCs)) *A probabilistic circuit (PC)* $\mathcal{C}$ *over RVs* $\mathbf{X}$*, characterizing a (possibly unnormalized) joint probability distribution* $p(\mathbf{X})$*, is a pair* $(\mathcal{G}, \boldsymbol{\theta})$*,*

where $\mathcal{G}$ is a computational graph, also called the **circuit structure** *that is parameterized by $\boldsymbol{\theta}$, also called the* **circuit parameters**, *as defined next.*

Definition 9 (Probabilistic circuits: structure) *Let $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ be a PC over RVs $\mathbf{X}$. $\mathcal{G}$ is a computational graph in the form of rooted DAG equipped with a scope function ϕ which associates to each unit, also called node, $n \in \mathcal{G}$ a subset of $\mathbf{X}$, i.e., $\phi(n) \subseteq \mathbf{X}$. If there is an edge $n \to c$ from node $n \in \mathcal{G}$ to node $c \in \mathcal{G}$, we call n the parent of c and c its child or, equivalently, c the input of n*[4]*. Let $\mathsf{ch}(n)$ denote the set of a child nodes of node n in $\mathcal{C}$. $\mathcal{G}$ comprises three kinds of units: input nodes, product nodes and sum nodes as defined next.*

Definition 10 (Probabilistic circuits: computational semantics) *Let $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ be a PC over RVs $\mathbf{X}$. Each node $n \in \mathcal{G}$ encodes a non-negative function $\mathcal{C}_n$ over its scope: $\mathcal{C}_n : \mathsf{val}(\phi(n)) \to \mathbb{R}_+$. An input node n in $\mathcal{C}$, also called leaf node, encodes a non-negative function that has a support $\mathsf{supp}(\mathcal{C}_n)$ and is parameterized by $\boldsymbol{\theta}_n$.*[5] *A product node n defines the factorization $\mathcal{C}_n(\mathbf{X}) = \prod_{c \in \mathsf{ch}(n)} \mathcal{C}_c(\mathbf{X})$. A sum node n defines the weighted sum $\mathcal{C}_n(\mathbf{X}) = \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathcal{C}_c(\mathbf{X})$ parameterized by weights $\theta_{n,c} \geq 0$*[6].

Definition 11 (Probabilistic circuits: parameters) *The set of parameters of a PC $\mathcal{C}$ is $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathsf{S}} \cup \boldsymbol{\theta}_{\mathsf{L}}$ where $\boldsymbol{\theta}_{\mathsf{S}}$ is the set comprising all sum node weights $\theta_{n,c}$ and $\boldsymbol{\theta}_{\mathsf{L}}$ is the set of parameters of all input nodes in $\mathcal{C}$.*

By above definitions, the recursive semantics of probabilistic circuits emerges. Let $\mathcal{C}_n$ be the sub-circuit rooted at node n. According to above definitions, $\mathcal{C}_n$ is also a PC, i.e., a computational graph over $\phi(n)$, parameterized by $\boldsymbol{\theta}_n$.

In the rest of the paper, we will make the following simplifying assumptions. Moreover we will assume input units to model unique functions. The uniqueness of these function will come handy later. Note that we can safely assume that since if that were not the case, we could always replace all inputs encoding the same exact function by a single node and redirect its output to the union of the outputs of the removed input nodes. Lastly, when not specified otherwise, we will consider PCs to have input units modeling univariate input distributions for the sake of simplicity.

Recall from Section 2.1 that simple probability distributions can be easily represented as computational units serving as input nodes in PCs. Among these, exponential families, such as Poisson, Gaussians, and Bernoulli, are common choices guaranteeing large tractable

4. We adopt the classical PGM convention for edge directionality when traversing the DAG: from parent to child. Note, however that tf we interpret the DAG as a computational graph, edges directions shall go the other way around: from inputs to outputs. Note that when plotting computational graphs, e.g., Example 8, we do not graphically show the direction of the arrows connecting inputs to the product to avoid clutter and overcome this ambiguity. This is a graphical simplification we will adopt in all graphics in this paper.
5. Here we assume that $\mathsf{supp}(\mathcal{C}_n)$ is the *inherent* support for a leaf node n, i.e., it does not change for arbitrary choices of parameters $\boldsymbol{\theta}_n$.
6. The assumption of having normalized weights, i.e., $\sum_c \theta_{n,c} = 1$, delivers the classical intuition on mixture models defining normalized distributions. However, it is not needed because in mixture models supporting MAR and in circuits we can always normalize the encoded distribution by locally re-normalizing weights (Peharz et al., 2015).

bands. When we consider Boolean RVs, it might come handy to represent input nodes as indicator functions. For instance, to model an input unit whose scope is RV $X_i \in \mathbf{X}$ we can have either the indicator $[\![X_i = 0]\!]$ or $[\![X_i = 1]\!]$. Note that a classical Bernoulli RV with parameters θ can be represented by a small PC consisting of the two indicator nodes defined as above and feeding a sum node whose parameters are θ and $1 - \theta$.

Using Examples 8 and 9 in Section 2.1, we provided a flavor of how the evaluation of a PC corresponds to the feedforward (bottom-up) evaluation of its underlying computational graph, when targeting PCs encoding simple distributions. We now provide a more rigorous and general definition applicable to all EVI queries, that is when all input units in a PC are provided an input value.

Definition 12 (EVI query computations) *Given a complete configuration* $\mathbf{x}$ *for RVs* $\mathbf{X}$*, the output of any sub-circuit rooted at node* n *over the RVs* $\mathbf{Z} = \phi(n)$*, denoted* $\mathcal{C}_n(\mathbf{x}) = \mathcal{C}_n(\boldsymbol{z})$*, where* $\boldsymbol{z}$ *is the restriction of* $\boldsymbol{x}$ *to the RVs in* $\mathbf{Z}$*, is obtained by evaluating the sub-circuit in a feed-forward manner (also called bottom-up), i.e., children before parents.*

Before moving to inference with PCs for other query classes, we introduce the interpretation of a PC as a polynomial function whose unknowns are the input functions. We will leverage the notion of an induced sub-circuit to traverse the DAG $\mathcal{G}$ of a PC.

Definition 13 (Induced sub-circuit) *Let* $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ *be a PC over RVs* $\mathbf{X}$*. An induced sub-circuit* $\mathcal{T}$ *built from* $\mathcal{G}$ *is a tree built as follows in a recursive top-down manner. The root* n *of* $\mathcal{C}$ *is in* $\mathcal{T}$*. If* n *is a product node, then for every child* c*, the sub-circuit rooted at* c *and the edge* $n \to c$ *are in* $\mathcal{T}$*. If* n *is a sum node, then for one child* c*, the sub-circuit rooted at* c *and the weighted edge* $n \to c$ *are in* $\mathcal{T}$*.*

Note that each leaf node in a sub-circuit $\mathcal{T}$ of a graph $\mathcal{G}$ is also a leaf node in $\mathcal{G}$. Furthermore, the DAG $\mathcal{G}$ can be represented as the collection $\{\mathcal{T}_i\}_i$ of all the sub-circuit one can enumerate by taking different children at every sum node in $\mathcal{G}$. Given this "unrolled" representation of $\mathcal{G}$, we can now define the polynomial representation of a PC $\mathcal{C}$.

Definition 14 (Circuit polynomial) *Let* $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ *be a PC over RVs* $\mathbf{X}$*. For a complete configuration* $\boldsymbol{x} \in \mathsf{val}(\mathbf{X})$*,* $\mathcal{C}$ *computes the following polynomial*

$$\mathcal{C}(\boldsymbol{x}) = \sum_{\mathcal{T}_i \in \mathcal{G}} \left(\prod_{\theta_j \in \boldsymbol{\theta}_{\mathcal{T}_i}} \theta_j \right) \prod_{c \in \mathcal{T}_i} \mathcal{C}_c^{d_c}(\boldsymbol{x}) = \sum_{\mathcal{T}_i} \theta_{\mathcal{T}_i} \prod_{c \in \mathcal{T}_i} \mathcal{C}_c^{d_c}(\boldsymbol{x}) \tag{8}$$

where $\mathcal{T}_i \in \mathcal{G}$ *is a sub-circuit tree in the computational graph as previously defined,* $\boldsymbol{\theta}_{\mathcal{T}_i}$ *is the collection of weights attached to weighted edges in* $\mathcal{T}_i$*,* $c \in \mathcal{T}_i$ *denotes one of its input unit and* d_c *denotes how many times an input unit is reachable in* $\mathcal{T}_i$*.*

This polynomial representation becomes a multilinear polynomial when d_c becomes 1 for all possible input units. In that case the induced sub-circuit is a tree (and sometimes called only induced tree (Zhao et al., 2015)). Furthermore, if a $\mathcal{C}$ models only binary RVs, the circuit polynomial as defined above for $d_c = 1$ goes under the name of *network polynomial* in the Bayesian network and AC literature (Darwiche, 2003, 2009).

3. Tractable Circuits for Marginals

In this section, we will provide a precise characterization of the circuit structures that allow tractable computation of MAR queries. We first define what it means for a probabilistic circuit to compute marginals.

Definition 15 *A probabilistic circuit $\mathcal{C}$ over variables* $\mathbf{X}$ **computes marginals** *if the partial evidence computation $\mathcal{C}_{\mathcal{I}}(\mathbf{y})$ is equal to the marginal $\int_{\mathcal{I}} \mathcal{C}(\mathbf{yz})\, d\mathbf{z}$ for all subset $\mathbf{Y} \subseteq \mathbf{X}$ and $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$, instantiations $\mathbf{y}$, and intervals $\mathcal{I}$.*

Note that a probabilistic circuit that does not compute marginals is not necessarily intractable for certain marginal queries – there may be other polytime algorithms than partial evidence computation. Instead, circuits that compute marginals can intuitively be understood as encoding the marginal distributions for every subset of variables.

We next introduce the structural properties that are known to be sufficient for marginals.

Definition 16 *A product node n is* **decomposable** *if the scopes of its children do not share variables: $\phi(c_i) \cap \phi(c_j) = \emptyset$, $\forall\, c_i, c_j \in \mathsf{ch}(n), i \neq j$. A circuit is decomposable if all of its product nodes are decomposable.*

Definition 17 *A sum node n is* **smooth** *if its children depend on the same variables: $\phi(c) = \phi(n)$, $\forall\, c \in \mathsf{ch}(n)$. A circuit is smooth if all of its sum nodes are smooth.*

Decomposable and smooth circuits were used for marginal inference for Boolean indicator leaf nodes in the context of arithmetic circuits (Darwiche, 2003) which are decomposable, smooth, and deterministic (to be introduced in the next section), and also by Poon and Domingos (2011) using sum product networks. It was later shown by Peharz et al. (2015) that decomposability and smoothness are in fact sufficient for a circuit with arbitrary leaf nodes to compute marginals. Next, we will show that both structural properties are in fact necessary for a probabilistic circuit to compute marginals.

Theorem 18 *Suppose $\mathcal{G}$ is a circuit structure such that for any parameterization $\boldsymbol{\theta}$, the probabilistic circuit $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ computes marginals. Then, $\mathcal{G}$ must be decomposable and smooth.*

Every probabilistic circuit using sum and product nodes computes a polynomial in terms of the leaf functions:

$$\mathcal{C}(\mathbf{x}) = \sum_i \theta_i \prod_j L_{ij}^{d_{ij}}(\mathbf{x}).$$

Let us call each i a term and θ_i its coefficient. Here, each $L_{ij}(\mathbf{x})$ is a leaf function, and the product $\prod_j L_{ij}^{d_{ij}}$ makes up the term i; θ_i is then a product of some parameters in the circuit. Note that given a partial evidence $\mathbf{y}$ and a set of intervals $\mathcal{I}$ on $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$, the circuit then evaluates to:

$$\mathcal{C}_{\mathcal{I}}(\mathbf{y}) = \sum_i \theta_i \prod_j \left(\int_{\mathcal{I}} L_{ij}(\mathbf{yz}) d\mathbf{z} \right)^{d_{ij}}.$$

First, suppose $\mathcal{G}$ is not decomposable, and let X be the variable that is shared between children of an non-decomposable node. Given an instantiation $\mathbf{y}$ to $\mathbf{Y} = \mathbf{X} \setminus \{X\}$ and interval $\mathcal{I}$ on X, the circuit evaluates to $\mathcal{C}_{\mathcal{I}}(\mathbf{y}) = \sum_i \theta_i \prod_j \left(\int_{\mathcal{I}} L_{ij}(x\mathbf{y})dx\right)^{d_{ij}}$; whereas, the marginal is $\int_{\mathcal{I}} \mathcal{C}(x\mathbf{y})dx = \sum_i \theta_i \int_{\mathcal{I}} \prod_j L_{ij}^{d_{ij}}(x\mathbf{y})dx$. Since $\mathcal{G}$ is not decomposable, there must be a term i where either $d_{ij} > 1$ for some leaf function L_{ij} or more than one leaf functions depend on X. In either of the cases, $\mathcal{C}_{\mathcal{I}}(\mathbf{y})$ is not in general equal to the marginal. In other words, decomposability is necessary for computing marginals.

Next, suppose $\mathcal{G}$ is decomposable but not smooth. Then the+re must exist a variable $X \in \mathbf{X}$ such that there is a term i for which no leaf function depends on X. Let I be the set of all terms where this happens, which must be nonempty because $\mathcal{G}$ is not smooth. Also, w.l.o.g. let $j = 1$ be the leaf function that depends on X for each $i \notin I$. As $\mathcal{G}$ is decomposable, $d_{ij} = 1$ for all i, j and no two leaf functions for a term share variables. Therefore, for any instantiation $\mathbf{y}$ to $\mathbf{Y} = \mathbf{X} \setminus \{X\}$, the marginal is:

$$\begin{aligned}
\int_{\mathcal{I}} \mathcal{C}(x\mathbf{y})dx &= \sum_i \theta_i \int_{\mathcal{I}} \left(\prod_j L_{ij}(x\mathbf{y})\right) dx \\
&= \sum_{i\in I} \theta_i \left(\int_{\mathcal{I}} dx\right) \prod_j L_{ij}(\mathbf{y}) + \sum_{i\notin I} \theta_i \left(\int_{\mathcal{I}} L_{i1}(x\mathbf{y})dx\right) \prod_{j\neq 1} L_{ij}(\mathbf{y}) \\
&= \sum_{i\in I} \theta_i \,|\mathcal{I}| \prod_j L_{ij}(\mathbf{y}) + \sum_{i\notin I} \theta_i \left(\int_{\mathcal{I}} L_{i1}(x\mathbf{y})dx\right) \prod_{j\neq 1} L_{ij}(\mathbf{y}). \qquad (9)
\end{aligned}$$

Now suppose we evaluate the circuit on above partial evidence $\mathbf{y}$. Then,

$$\mathcal{C}_{\mathcal{I}}(\mathbf{y}) = \sum_{i\in I} \theta_i \prod_j L_{ij}(\mathbf{y}) + \sum_{i\notin I} \theta_i \left(\int_{\mathcal{I}} L_{i1}(x\mathbf{y})dx\right) \prod_{j\neq 1} L_{ij}(\mathbf{y}). \qquad (10)$$

Hence, the partial evidence evaluation is not necessarily equal to the marginal. In particular, if X is a discrete variable, $\int dx$ in Equation 9 refers to $\sum_x 1$ which is the number of potential values for X. Thus, $\mathcal{C}_{\mathcal{I}}(\mathbf{y})$ lower bounds the marginal.

Therefore, decomposability and smoothness precisely describe all circuit structures that allow for marginals under any parameterization.

We have now seen our first family of tractable probabilistic circuits, namely those that are decomposable and smooth. As tractability is defined in terms of the size of the model, naturally we ask how these structural constraints affect the expressive efficiency of circuits. Darwiche and Marquis (2002) showed that there are functions with a compact circuit representation, but no polynomial-sized decomposable circuit.[7] Thus, the family decomposable and smooth circuits are not as expressive efficient as the family of all probabilistic circuits. Indeed, this is not surprising as marginal inference is #P-hard.

4. Tractable Circuits for MAP

We next study the circuit structures that enable tractable MAP inference. Again, we first define when a circuit computes the MAP, using the notion of maximizer circuits.

7. This was shown for logical circuits, but the proof can easily be extended to probabilistic circuits.

Definition 19 (Distribution maximizer) *Let $\mathcal{C}_{\mathsf{L}}$ be an input function of a PC, characterizing some distribution, then its associated function maximizer, denoted as $\mathcal{C}_{\mathsf{L}}^{\max}$ computes $\max_{\boldsymbol{y}\in\mathsf{val}(\mathbf{Y})} \mathcal{C}_{\mathsf{L}}(\boldsymbol{y})$ where $\mathbf{Y} = \phi(n)$.*

Definition 20 (Circuit maximizer) *For a given circuit $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ over RVs $\mathbf{X}$, let $\mathcal{C}_{\max} = (\mathcal{G}_{\max}, \boldsymbol{\theta})$ be its* **maximizer circuit** *where $\mathcal{G}_{\max}$ is obtained by replacing every sum node n in $\mathcal{G}$ with a max node, i.e., computing the function $\mathcal{C}_n(\mathbf{X}) = \max_{c\in\mathsf{ch}(n)} \theta_{n,c}\mathcal{C}_c(\mathbf{X})$, and every input distribution with its distribution maximizer.*

Definition 21 *Given a probabilistic circuit $\mathcal{C}$ and its maximizer circuit $\mathcal{C}_{\max}$, we say $\mathcal{C}_{\max}$* **computes the MAP of** *$\mathcal{C}$ if $\mathcal{C}_{\max}(\mathbf{y}) = \max_{\mathbf{z}} \mathcal{C}(\mathbf{yz})$ for all subset $\mathbf{Y} \subseteq \mathbf{X}$ and $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$ and its instantiation $\mathbf{y}$.*

Let us now introduce a structural property that enables tractable MAP computations.

Definition 22 (Deterministic circuits) *A sum node n is* **deterministic** *if, for any fully-instantiated input, the output of at most one of its children is nonzero. A circuit is deterministic if all of its sum nodes are deterministic.*

Determinism is our first structural property that constrains the output of a node, instead of its scope. Note that it is still a restriction on the circuit structure and not its parameters, as the inherent support of leaf nodes given by the structure cannot be altered by the parameters. Chan and Darwiche (2006) showed that maximizer circuits of smooth, decomposable, and deterministic circuits compute the MAP. That is, these properties are sufficient conditions for MAP computations using maximizer circuits.

We now turn our focus to identifying the necessary conditions. First, we observe that decomposability is not in fact necessary, and that a strictly weaker restriction, namely consistency, is enough. We adopt the notion of consistency introduced by Poon and Domingos (2011) for Boolean variables, and generalize it to arbitrary (continuous or discrete) random variables as the following:

Definition 23 *A product node is* **consistent** *if each variable that is shared between multiple children nodes only appears in a single leaf node, in the subcircuit rooted at the product node. A circuit is consistent if all of its product nodes are consistent.*

Clearly, any decomposable product node is also consistent by definition.

Theorem 24 *Suppose $\mathcal{G}$ is a circuit structure such that for any parameterization $\boldsymbol{\theta}$, the maximizer circuit $\mathcal{C}_{\max} = (\mathcal{G}_{\max}, \boldsymbol{\theta})$ computes the MAP of $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$. Then, $\mathcal{G}$ must be consistent and deterministic.*

We will prove above theorem by first showing necessity of consistency then determinism. Maximizer circuit $\mathcal{C}_{\max}$ computes the MAP iff any product node n with children $\{n_i\}_i$ satisfies $\max_{\mathbf{x}} n(\mathbf{x}) = \prod_i \max_{\mathbf{x}} n_i(\mathbf{x})$. Suppose there exists an inconsistent node n. Let n_1 and n_2 denote its children nodes where a variable X appears in leaf nodes L_1 and L_2, respectively. We can assume w.l.o.g that n_1 and n_2 are consistent, and thus $\arg\max_{\mathbf{x}} n_1(\mathbf{x})$ (resp. $\arg\max_{\mathbf{x}} n_2(\mathbf{x})$) must agree with $\arg\max_{\mathbf{x}} L_1(\mathbf{x})$ (resp. $\arg\max_{\mathbf{x}} L_2(\mathbf{x})$) on the value of X.

However, we can parameterize the leaf functions at L_1 and L_2 such that their respective MAP assignments do not agree on the value of X. In other words, the maximizer circuit can return a value that does not correspond to a consistent assignment of the variables. Therefore, any circuit that computes the MAP must be consistent.

Next, we show that such circuit must also deterministic, by adapting the proof from Choi and Darwiche (2017) that any smooth, decomposable circuit computing the MAP must also be deterministic. Suppose $\mathcal{G}$ is not deterministic, and let n be a non-deterministic sum node. Hence, there exists one or more complete inputs (denote $\mathcal{X}$) such that more than one children of node n evaluate to non-zero values.

Since $\mathcal{G}$ must be consistent, at least one of those inputs must make the circuit at the root output non-zero. To show this, suppose all inputs in $\mathcal{X}$ lead to a circuit output of zero. Then there must exist an ancestor of n (or n itself) that is always multiplied with a node that outputs 0 for all inputs in $\mathcal{X}$. Note that the variables not in the scope of n can be assigned freely, and thus the output of zero must be caused by assignments to variables in the scope of n. This is only possible if the circuit is inconsistent.

We can parameterize the circuit such that an input in $\mathcal{X}$ is a MAP state. Then without performing additions at sum nodes, one cannot retrieve the value of $\mathcal{C}(\mathbf{x})$ and cannot correctly compute the MAP.

We have now shown the necessary and sufficient conditions for two classes of queries: MAR and MAP. Next, we study the succinctness of circuits when asserting these conditions.

Theorem 25 *There exists a function with a circuit of linear size that can compute the MAP but no poly-size circuit that computes its marginals. (Assuming that the polynomial hierarchy does not collapse)*

Consider a circuit $\mathcal{C}$ of the following form over Boolean variables $\mathbf{X} = \{X_1, \ldots, X_n\}, \mathbf{Y} = \{Y_1, \ldots, Y_r\}$:

$$\prod_{i}^{r} (Y_i \cdot Z_{i1} + (\neg Y_i) \cdot Z_{i2}), \tag{11}$$

where each $Z_{ij} \in \mathbf{X}$. Note that above circuit is consistent and deterministic, and thus allows for computation of MAP using its maximizer circuit. Next, we will show that computing the marginal of above function is a #P-hard problem. The proof is by reduction from SAT′ which was shown to be #P-complete by Valiant (1979).

- SAT′: Given a monotone 2CNF $\bigwedge_i^r (Z_{i1} \vee Z_{i2})$ where $Z_{ij} \in \mathbf{X}$, output $|\{(\mathbf{x}, \mathbf{t}) : \mathbf{t} \in \{1, 2\}^r, \mathbf{x} \text{ makes } Z_{i,t_i} \text{ true } \forall\, i\}|$.

Note that for a given $\mathbf{x}$, the number of $(\mathbf{x}, \mathbf{t})$ that is counted by SAT′ is 0 if $\mathbf{x}$ does not satisfy the 2CNF formula, and otherwise 2^m where m is the number of clauses i such that both literals Z_{i1} and Z_{i2} are set to true. Given a monotone 2CNF, let us construct a consistent and deterministic circuit by changing the logical AND into a product node, OR into a sum node, and adding auxiliary variables Y_i for each clause as in Equation 11. Then for any given $\mathbf{x}$, the marginal $\mathcal{C}(\mathbf{x})$ (with $\mathbf{Y}$ unobserved) computes $\prod(Z_{i1} + Z_{i2})$ which is 0 if $\mathbf{x}$ does not satisfy the formula and 2^m otherwise; the marginal $\mathcal{C}(.)$ is then the solution to the original SAT′ problem. Hence, there cannot exist a poly-sized circuit for this function that allows marginals, unless the polynomial hierarchy collapses.

Theorem 26 (Choi and Darwiche (2017)) *There exists a function with linear-size circuit that can compute marginals but no poly-size circuit that computes its MAP. (Assuming that the polynomial hierarchy does not collapse)*

Any naive Bayes network can be represented as a linear-size smooth and decomposable probabilistic circuit. The marginal feature distribution can be represented by forgetting the class variable from such circuit (i.e. set all leafs for the class variable to 1 then simplify the circuit). A poly-size circuit that computes the MAP of this marginal distribution then computes the marginal MAP of naive Bayes in polytime. However, marginal MAP is known to be NP-hard for naive Bayes (de Campos, 2011).

Therefore, even though marginal inference is computationally harder than MAP inference, there still exist functions with probabilistic circuits for tractable marginals but not tractable MAP.

References

Hei Chan and Adnan Darwiche. On the robustness of most probable explanations. In ***Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence***, pages 63–71, 2006.

Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In ***Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)***, 2017.

Arthur Choi, Guy Van den Broeck, and Adnan Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In ***Twenty-Fourth International Joint Conference on Artificial Intelligence***, 2015.

Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. ***Artificial intelligence***, 42(2-3):393–405, 1990.

Adnan Darwiche. A differential approach to inference in bayesian networks. ***Journal of the ACM (JACM)***, 50(3):280–305, 2003.

Adnan Darwiche. ***Modeling and reasoning with Bayesian networks***. Cambridge university press, 2009.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. ***Journal of Artificial Intelligence Research***, 17:229–264, 2002.

Cassio P de Campos. New complexity results for map in bayesian networks. In ***IJCAI***, volume 11, pages 2100–2106, 2011.

Willliam Feller. ***An introduction to probability theory and its applications***, volume 2. John Wiley & Sons, 2008.

Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. ***arXiv preprint arXiv:1903.12436***, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In ***Advances in neural information processing systems***, pages 2672–2680, 2014.

Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. In ***Advances in Neural Information Processing Systems***, pages 11167–11178, 2019.

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

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In ***Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning***, 2014.

Daphne Koller and Nir Friedman. ***Probabilistic graphical models: principles and techniques***. MIT press, 2009.

Yitao Liang and Guy Van den Broeck. Towards compact interpretable models: Shrinking of learned probabilistic sentential decision diagrams. In ***IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)***, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. ***arXiv preprint arXiv:1912.02762***, 2019.

Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. On theoretical properties of sum-product networks. In ***Artificial Intelligence and Statistics***, pages 744–752, 2015.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In ***2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)***, pages 689–690. IEEE, 2011.

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

Jeffrey S Rosenthal. ***A first look at rigorous probability theory***. World Scientific Publishing Company, 2006.

Dan Roth. On the hardness of approximate reasoning. ***Artificial Intelligence***, 82(12): 273–302, 1996.

Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. ***Synthesis lectures on data management***, 3(2):1–180, 2011.

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

Guy Van den Broeck, Dan Suciu, et al. Query processing on probabilistic data: A survey. ***Foundations and Trends® in Databases***, 7(3-4):197–341, 2017.

Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In ***International Conference on Machine Learning***, pages 116–124, 2015.