E. Tractable Probabilistic Models (TPM)

A probabilistic model can be seen as a black box to answer queries about the quantities of interest of the joint probability distribution, such as computing marginal probability and performing maximum-a-posterior inference given some evidence. Tractable probabilistic models provide more guarantees when answering probabilistic queries: (i) it can perform exact inference to the model's distribution and no approximations are required. (ii) the query computation can be carried out efficiently, that is, in time polynomial (linear in many cases) in the size of the model. Notably, tractability is defined for a family of models only w.r.t. a class of queries and not an absolute property. Indeed, a tractable representation for one query class might not admit polynomial time inference for another query class.

F. Probabilistic Circuits (PC)

F.1. Definition of PC

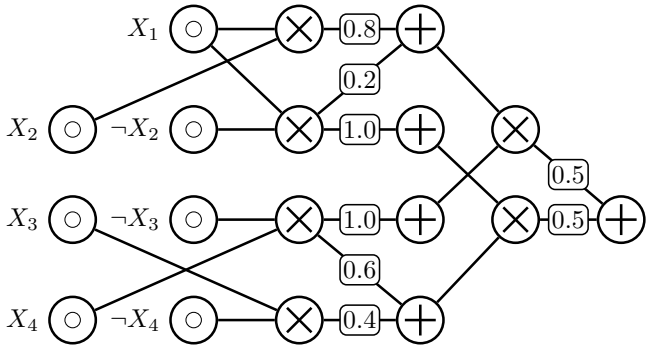

Figure 4: An example PC.

Probabilistic circuits (PCs) represent a wide class of TPMs that model probability distributions with a parameterized directed acyclic computation graph (DAG). Specifically, a PC $p(\mathbf{X})$ defines a joint distribution over a set of random variables $\mathbf{X}$ by a single root node n_r. A PC contains three kinds of computational nodes: *input*, *sum*, and *product*. As illustrated in Figure 4, the example PC in Figure 4 defines a joint distribution over 4 random variables X_1, X_2, X_3, X_4. Each leaf node in the DAG serves as an *input* node that encodes a univariate distribution (e.g., Guassian, Categorical), while *sum* nodes or *product* nodes are inner nodes, distinguished by whether they are doing mixture or factorization over their child distributions (denoted $\mathsf{in}(n)$). A PC defines a probability distribution in the following recursive way:

$$p_n(\boldsymbol{x}) := \begin{cases} f_n(\boldsymbol{x}) & \text{if } n \text{ is an input unit,} \\ \sum_{c \in \mathrm{in}(n)} \theta_{n,c} \cdot p_c(\boldsymbol{x}) & \text{if } n \text{ is a sum unit,} \\ \prod_{c \in \mathrm{in}(n)} p_c(\boldsymbol{x}) & \text{if } n \text{ is a product unit,} \end{cases}$$

where $\theta_{n,c}$ represents the parameter corresponding to edge (n, c) in the DAG. For sum units, we have $\sum_{c \in \mathsf{in}(n)} \theta_{n,c} = 1, \theta_{n,c} \geq 0$, and we assume w.l.o.g. that a PC alternates between the sum and product layers before reaching its inputs.

F.2. Tractability and Expressivity

Definition 1 (Decomposability)**.** A PC is decomposable if for every product unit n, its children have disjoint scopes:

$$\forall c_1, c_2 \in \mathsf{in}(n)\ (c_1 \neq c_2),\ \phi(c_1) \cap \phi(c_2) = \varnothing.$$

Definition 2 (Smoothness)**.** A PC is smooth if for every sum unit n, its children have the same scope:

$$\forall c_1, c_2 \in \mathsf{in}(n),\ \phi(c_1) = \phi(c_2).$$

Similar to neural networks, a large PC contains millions of computation units which enables expressive modeling. However, to guarantee the desired tractability, i.e., the ability to answer numerous probabilistic queries, some properties have to be imposed on their DAG structures. For instance, *smoothness* together with *decomposability* ensure that a PC can compute arbitrary marginal/conditional probabilities in linear time w.r.t. its size, i.e., the number of edges in its DAG. These are properties of the variable scope $\phi(n)$ of PC unit n, that is, the variable set comprising all its descendent nodes. Here we introduce how we execute marginalization inference in practice on a decomposable and smooth PC. Take the example PC in Figure 4 as an example, if we want to acquire $p_n(X_1, X_2, X_3)$ by marginalizing out X_4, we only need to set the output probability of X_4's input nodes to 1 and follow the same forward process as computing $p_n(X_1, X_2, X_3, X_4)$.

Notably, such structural constraints raise higher requirements for PC optimizers to acquire a comparably expressive PC. Thankfully, recent advancement achieved in PC modeling largely bridges this gap and makes it possible to model distributions of more complex data including the offline RL data.