Bayesian Structure Scores for Probabilistic Circuits

Yang Yang[*]
Eindhoven University of Technology
KU Leuven

Gennaro Gala[*]
Eindhoven University of Technology

Robert Peharz
Graz University of Technology
Eindhoven University of Technology

Abstract

Probabilistic circuits (PCs) are a prominent representation of probability distributions with tractable inference. While parameter learning in PCs is rigorously studied, structure learning is often more based on heuristics than on principled objectives. In this paper, we develop Bayesian structure scores for deterministic PCs, i.e., the structure likelihood with parameters marginalized out, which are well known as rigorous objectives for structure learning in probabilistic graphical models. When used within a greedy cutset algorithm, our scores effectively protect against overfitting and yield a fast and almost hyper-parameter-free structure learner, distinguishing it from previous approaches. In experiments, we achieve good trade-offs between training time and model fit in terms of log-likelihood. Moreover, the principled nature of Bayesian scores unlocks PCs for accommodating frameworks such as structural expectation-maximization.

1 INTRODUCTION

Probabilistic circuits (PCs) (Vergari et al., 2020) have emerged as a powerful framework for describing tractable probabilistic models, such as *Chow-Liu trees* (Chow & Liu, 1968), *arithmetic circuits* (Darwiche, 2003), *sum-product networks* (Poon & Domingos, 2011), *probabilistic sentential decision diagrams* (Kisa et al., 2014) and *cutset networks* (CNets) (Rahman et al., 2014). PCs can be categorized into several sub-families specified by *structural properties* or *constraints*, which go hand in hand with certain tractable inference routines (Vergari et al., 2020), such as *marginalization*, *conditioning*, *most-probable explanation*, *expectations*, etc.

[*]These authors contributed equally.

Proceedings of the 26[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

PCs can either be compiled from other models (Darwiche, 2003) or learned directly from data (Lowd & Domingos, 2008; Poon & Domingos, 2011). When learning PCs from data, we might—similar as in classical probabilistic graphical models (PGMs) (Koller & Friedman, 2009)—distinguish between *parameter learning* and *structure learning*. Parameter learning has been rigorously studied in PCs and mirrors state-of-the-art in PGMs. Specifically, techniques for parameter learning in PCs are *(closed-form) maximum-likelihood estimation* (for the sub-class of deterministic PCs) (Kisa et al., 2014; Peharz et al., 2014), *expectation-maximization* (Peharz et al., 2016), *concave-convex procedures* (Zhao, Poupart, & Gordon, 2016), *Bayesian approaches* (Zhao, Adel, et al., 2016; Rashwan et al., 2016; Vergari et al., 2019; Trapp et al., 2019), *discriminative methods* (Rashwan et al., 2018; Peharz, Vergari, et al., 2020), *continuous embeddings* (Shao et al., 2022; Correia et al., 2022), and *latent variable distillation* (Liu et al., 2022).

Structure learning in PCs, however, is generally less principled than in PGMs. In the latter, the two main styles of structure learning are *constraint-based*, i.e., detecting conditional independencies in the data and translating them into a corresponding graph structure (Koller & Friedman, 2009); and *score-based*, i.e., phrasing structure learning as a discrete optimization problem, aiming to find a graph that maximizes some structure score. Score-based approaches have turned out to be more practical, as constraint-based approaches tend to be sensitive to statistical noise in conditional independence tests (Koller & Friedman, 2009, page 790). In particular, *Bayesian structure scores* (Buntine, 1991; Geiger & Heckerman, 1994; Heckerman et al., 1995) are some of the most prominent scores, due to their principled nature and asymptotic consistency guarantees (Koller & Friedman, 2009). Bayesian scores are the (log-)likelihood of the candidate structures, with parameters marginalized out, and have been the target of both greedy (Cooper & Herskovits, 1992) and exact (De Campos et al., 2009) structure optimizers.

In PCs, however, structure learners are often based on heuristics. The most prominent structure learning scheme is probably *LearnSPN* (Gens & Domingos, 2013), which,

in a nutshell, uses top-down multi-clustering to lay out the PC structure. The same approach is taken in *ID-SPN* (Rooshenas & Lowd, 2014), which augments LearnSPN by using expressive distribution models as sub-modules. Other structure learners are based on singular-value decomposition (Adel et al., 2015), bottom-up information bottleneck learning (Peharz et al., 2013) or are motivated by decision tree learning (Rahman et al., 2014). None of these approaches declare an *explicit objective* for structure learning, although at test time they are usually evaluated in terms of log-likelihood. Moreover, they often rely on a large number of hyper-parameters, which are tedious to set or need to be tuned on a separate validation set.

There are also score-based approaches to learn PC structures, e.g. the ones by Lowd and Domingos (2008); Peharz et al. (2014), who use a weighted sum of training log-likelihood and circuit size as a structure score. The circuit size directly corresponds to the worst-case inference cost in PCs, thus these scores have the interesting interpretation of trading off "model fit" vs. "inference complexity." However, a deeper theoretical justification of these scores has yet to be shown. Similarly, the *Strudel* method (Dang et al., 2020) also performs a structure search based on "*heuristic scores calculated from data*." Thus, these score-based methods are—similarly as the previously mentioned structure learners—of heuristic nature, usually equipped with various (often unintuitive) hyper-parameters for regularizing the model, and require a separate validation set.

In this paper, we propose a principled avenue to PC structure learning and put a particular emphasis on elegance and simplicity. Specifically, we derive *Bayesian structure scores* for *deterministic* PCs,[1] by equipping parameters with suitable priors and marginalizing them from the Bayesian model. Thus, similarly as in PGMs, a Bayesian score is the marginal likelihood of the PC structure under a particular choice of parameter prior. Intuitively, since the Bayesian score averages over *all* possible choices of parameters, it effectively protects against overfitting, as overly complex structures will have a significant portion of "bad" parameters that are not well-supported by the data.

In the special case of discrete data and Dirichlet priors, one yields the well-known *Bayes-Dirichlet* (BD) score (Buntine, 1991; Heckerman et al., 1995). We employ the BD score within *greedy cutset learning* (Rahman et al., 2014), a simple and fast structure learning scheme for PCs. Our method has a single hyper-parameter governing the Dirichlet prior, the *equivalent sample size* (ESS), which we, however, keep fixed to 0.1 throughout our experiments, rendering our method effectively *hyper-parameter-free*.

Additionally, we consider the *Bayesian information criterion* (BIC), which is frequently used as an alternative to the BD score, as they are asymptotically equivalent (Schwarz, 1978; Koller & Friedman, 2009). The BIC score has indeed been considered for PCs (Di Mauro et al., 2015), but in a form which undercounts the actual number of parameters. Our corrected version of the BIC score improves the results by Di Mauro et al. (2015) and is demonstrated to be a viable and efficient alternative to our full BD score.

Both BD and BIC are de facto *hyper-parameter-free* and designed to reflect generalization. Therefore, they can use the full training data, without the need to dedicate part of it to a validation set. Both scores deliver models competitive with state-of-the-art when learned on 20 common benchmark density estimation data sets, even though we are using a simplistic structure learner. We compare our approach with various other PC structure learners, both simple ones—which we run, like our methods, with their default hyper-parameters on the entire training data—and sophisticated ones—which use extensive tuning but consume orders of magnitude more computational resources. Our structure learners are often located on the Pareto front determined by training time and test log-likelihood, i.e. they are among the best fast structure learners. This, together with the fact that we use well-principled structure objectives, make our algorithms highly practical.

Moreover, since a Bayesian score is just the marginal structure likelihood, we can naturally embed our structure learner within a *structural expectation maximization (EM)* algorithm (Friedman, 1998) for learning mixtures of PCs. Structural EM, which to the best of our knowledge has not been applied to PCs before, consistently improves over single PCs learned with BD, and delivers models close to or surpassing state-of-the-art.

The source code to replicate the experiments is available at `https://github.com/yangyang-pro/bayesian-scores-pc`.

2 PROBABILISTIC CIRCUITS

Given a set of random variables $\mathbf{X}$, a probabilistic circuit (PC) over $\mathbf{X}$ is a computational (directed acyclic) graph $\mathcal{G} = (V, E)$ containing three types of computational nodes, namely *distribution nodes*, *weighted sums* and *products*. As we elaborate below, each node $\mathsf{N} \in V$ computes a (possibly unnormalized) probability distribution over some subset of $\mathbf{X}$, denoted as the *scope* of N. Hence, we associate with each PC a *scope function* $\sigma \colon V \to 2^{\mathbf{X}}$, assigning to each node $\mathsf{N} \in V$ its scope $\sigma(\mathsf{N}) \subseteq \mathbf{X}$. Naturally, the scope of any internal node N satisfies $\sigma(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{in}(\mathsf{N})} \sigma(\mathsf{N}')$, where $\mathbf{in}(\mathsf{N})$ is the set of all input nodes to N, thus the scope function σ is completely specified by the scopes of the leaves (also called input nodes) of $\mathcal{G}$.

Any leaf $\mathsf{L} \in V$ of the PC is a probability distribution over its scope $\sigma(\mathsf{L})$, where one typically uses some "sim-

[1] Extensions to general, non-deterministic PCs are left to future work, but possible directions are given in the Discussion.

ple" parametric distribution like Gaussian, Poisson, categorical, etc., whose parameters are denoted as θ_{L}. The internal nodes of the PC are either sums or products. A sum node S, equipped with weights $\boldsymbol{w}_{\mathsf{S}} = \{w_{\mathsf{SN}}\}_{\mathsf{N}\in\mathbf{in}(\mathsf{S})}$, computes a convex combination (mixture) of its inputs, i.e., $\mathsf{S} = \sum_{\mathsf{N}\in\mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}\,\mathsf{N}$, where $\sum_{\mathsf{N}\in\mathbf{in}(\mathsf{S})} w_{\mathsf{SN}} = 1$ and $w_{\mathsf{SN}} \geq 0$; a product node P computes the product of its inputs, i.e., $\mathsf{P} = \prod_{\mathsf{N}\in\mathbf{in}(\mathsf{P})} \mathsf{N}$. The parameters $\{\Theta, \boldsymbol{w}\}$ of the PC are all parameters of the leaf distributions $\Theta = \{\theta_{\mathsf{L}}\}_{\mathsf{L}\in V}$ and all sum weights $\boldsymbol{w} = \{\boldsymbol{w}_{\mathsf{S}}\}_{\mathsf{S}\in V}$, where the expression $\mathsf{L} \in V$ (respectively $\mathsf{S} \in V$) means that we range over all leaves (respectively sum nodes) in V.

Since all leaves are probability distributions and all sum-weights are non-negative, it follows that each node in V computes some (possibly unnormalized) distribution over its scope. We assume that the PC has a single output node, having full scope $\mathbf{X}$, which represents the model distribution. It can be computed for any input sample $\mathbf{x}$ using a feed-forward pass in the PC.

Besides evaluating the model density, PCs also allow a wide range of tractable probabilistic inference routines, if they satisfy certain structural properties (or constraints), denoted as *decomposability*, *smoothness*, *determinism*, and *structured decomposability*. Specifically:

- A PC is *decomposable*, if for each product node P it holds that $\sigma(\mathsf{N}) \cap \sigma(\mathsf{N}') = \emptyset$, for all $\mathsf{N} \neq \mathsf{N}' \in \mathbf{in}(\mathsf{P})$.
- A PC is *smooth*, if for each sum node S it holds that $\sigma(\mathsf{N}) = \sigma(\mathsf{N}')$, for all $\mathsf{N}, \mathsf{N}' \in \mathbf{in}(\mathsf{S})$.
- A PC is *deterministic*, if for each sum node S and each sample $\mathbf{x}$, at most one of the inputs of S is non-zero.
- Furthermore, *structured decomposability* denotes a stricter form of decomposability, where the decompositions of all products adhere to a common pattern described by a so-called *vtree* (Pipatsrisawat & Darwiche, 2008; Kisa et al., 2014; Ahmed et al., 2022).

Not all properties are needed for each and every inference scenario, and studying the correspondence between structural properties and tractable inference routines is one of the intriguing facets of PCs (Darwiche & Marquis, 2002; Vergari et al., 2020). Typically one assumes at least the properties decomposability and smoothness, which ensure that each probability distribution computed by any node $\mathsf{N} \in V$ is already normalized. More importantly, they enable tractable *marginals* and *conditionals*—the two core routines of probabilistic reasoning (Ghahramani, 2015)—in time linear in the circuit size (Peharz et al., 2015). In the remainder of the paper, we assume that any PC is decomposable and smooth.

Adding determinism yields tractable inference of *most-probable explanations* (MPE), i.e., finding maximizers of the PC distribution, and exact computation of *maximum-likelihood parameters* (Kisa et al., 2014; Peharz et al., 2014). In this paper, we will see that determinism also allows to compute Bayesian structure scores exactly and efficiently.

Structured decomposability allows taking expectations of one circuit with respect to another (Khosravi et al., 2019), computing divergences between PCs (Vergari et al., 2021), etc.

PCs have evolved into a "lingua franca" of tractable models, since many of them can be cast into the PC framework. PCs might be compiled from other models (Darwiche, 2003) or learned directly from data, where we can distinguish between parameter learning and structure learning. As mentioned in the introduction, structure learning in PCs is usually rather ad-hoc and heuristic. In this paper, we develop *Bayesian scores* as well-principled objectives for PC structure learning.

3 BAYESIAN STRUCTURE SCORES FOR DETERMINISTIC PCS

Assume a training set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ of N samples drawn i.i.d. from an unknown probability distribution, and a structured, parametric density model $p(\mathbf{x} \mid \Theta, \mathcal{G})$, that is, a distribution over $\mathbf{X}$ conditional on continuous parameters Θ and a structure parameter $\mathcal{G}$. The basic idea of the *Bayesian structure score* is to (i) equip Θ with some prior distribution $p(\Theta \mid \mathcal{G})$, and to (ii) derive the marginal likelihood of $\mathcal{G}$ by marginalizing the parameters from the Bayesian model:

$$\mathcal{B}(\mathcal{G}) \coloneqq p(\mathcal{D} \mid \mathcal{G}) = \int p(\Theta \mid \mathcal{G}) \prod_{\mathbf{x}\in\mathcal{D}} p(\mathbf{x} \mid \Theta, \mathcal{G})\,\mathrm{d}\Theta. \quad (1)$$

The Bayesian score might be used as a direct target of optimization, i.e. a maximum likelihood approach with respect to $\mathcal{G}$. Alternatively, it might also be wrapped into a larger probabilistic framework, e.g. a Bayesian approach with respect to structure (Friedman & Koller, 2003) or structural expectation maximization (Friedman, 1998).

While eq. (1) is a hard computational problem in general, there are interesting special cases where it can be computed exactly and efficiently. In particular, in the context of Bayesian networks over discrete data, the well-known *Bayes-Dirichlet* (BD) score has been studied by Buntine (1991); Cooper and Herskovits (1992); Heckerman et al. (1995). By using Dirichlet priors on the parameters, together with particular assumptions such as independence among variable families, parameter modularity among different structures, etc., the BD score can be computed analytically in Bayesian networks. Similarly, for Bayesian networks with Gaussian parameters and normal-Wishart priors, the *Bayes-Gauss* (BG) score has been developed (Geiger & Heckerman, 1994).

Let now $\mathcal{G} = (V, E)$ be a candidate PC structure with parameters $\{\Theta, \boldsymbol{w}\}$, where $\Theta = \{\theta_{\mathsf{L}}\}_{\mathsf{L}\in V}$ and $\boldsymbol{w} = \{\boldsymbol{w}_{\mathsf{S}}\}_{\mathsf{S}\in V}$. The parameters of each leaf L are equipped with a prior $p(\theta_{\mathsf{L}})$, where we assume that eq. (1), when restricted to L, can be computed exactly and efficiently. That is, we can compute

$$\mathcal{B}_{\mathsf{L}} = \int p(\theta_{\mathsf{L}}) \prod_{\mathbf{x}\in\mathcal{D}} p(\mathbf{x} \mid \theta_{\mathsf{L}}) \,\mathrm{d}\theta_{\mathsf{L}}. \tag{2}$$

In order to keep the notation uncluttered, we write $p(\mathbf{x} \mid \theta_{\mathsf{L}})$, which should be understood as an evaluation of the leaf on the sub-vector of $\mathbf{x}$ whose entries correspond to $\sigma(\mathsf{L})$. Eq. (2) can be computed in closed form for many exponential families with conjugate priors, e.g. the Gaussian-Wishart and Binomial-Dirichlet families.

Further, let S be an arbitrary sum node with weights $\boldsymbol{w}_{\mathsf{S}} = \{w_{\mathsf{SN}}\}_{\mathsf{N}\in\mathbf{in}(\mathsf{S})}$, which we equip with a Dirichlet prior

$$p(\boldsymbol{w}_{\mathsf{S}}) = \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N}\in\mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}}-1}, \tag{3}$$

parameterized by $\boldsymbol{\alpha}_{\mathsf{S}} = \{\alpha_{\mathsf{SN}}\}_{\mathsf{N}\in\mathbf{in}(\mathsf{S})}$, where $\alpha_{\mathsf{SN}} > 0$, and $B(\boldsymbol{\alpha}_{\mathsf{S}})$ is the Beta function, i.e. the normalization constant of the Dirichlet distribution. Assuming a-priori parameter independence, we get the prior

$$\begin{aligned} p(\Theta, \boldsymbol{w} \mid \mathcal{G}) &= \prod_{\mathsf{S}\in V} p(\boldsymbol{w}_{\mathsf{S}}) \prod_{\mathsf{L}\in V} p(\theta_{\mathsf{L}}) \\ &= \prod_{\mathsf{S}\in V} \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N}\in\mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}}-1} \prod_{\mathsf{L}\in V} p(\theta_{\mathsf{L}}). \end{aligned} \tag{4}$$

The structure $\mathcal{G}$, leaf parameters Θ, and sum-weights $\boldsymbol{w}$ completely parametrize the PC distribution $p(\mathbf{x} \mid \Theta, \boldsymbol{w}, \mathcal{G})$,[2] such that the Bayesian structure score for PCs is given as

$$\mathcal{B}_{PC}(\mathcal{G}) = \int\int p(\Theta, \boldsymbol{w} \mid \mathcal{G}) \prod_{\mathbf{x}\in\mathcal{D}} p(\mathbf{x} \mid \Theta, \boldsymbol{w}, \mathcal{G}) \,\mathrm{d}\Theta \,\mathrm{d}\boldsymbol{w}. \tag{5}$$

While computing $\mathcal{B}_{PC}(\mathcal{G})$ is intractable for *non-deterministic* PCs, due to the high-dimensional integrals which do not simplify further, we can derive an analytic solution for *deterministic* PCs.

Recall that in deterministic PCs, for each possible sample $\mathbf{x}$ and each sum node S at most one input to S is non-zero. In general, whether a PC is deterministic or not might depend on both the structure $\mathcal{G}$ and the parameters $\{\Theta, \boldsymbol{w}\}$ in a convoluted manner. In this paper, however, we require that for given $\mathcal{G}$ determinism holds for each choice of parameters, a property which we call *structural determinism*. Moreover, we require that the indices of the non-zero sum inputs remain unchanged when varying parameters, i.e., for each sample $\mathbf{x}$ and sum node S it holds that if S has a non-zero input under two distinct parameters $\{\Theta', \boldsymbol{w}'\}$ and $\{\Theta'', \boldsymbol{w}''\}$, then the same input must be non-zero for both parameter choices. We call this property *parameter-consistent determinism*, which is often assumed by default in literature (Choi et al., 2020).

A simple way to enforce structural and parameter-consistent determinism is to associate with each sum S a corresponding random variable $X_{\mathsf{S}} \in \mathbf{X}$, and let S be of the form

$$\mathsf{S} = \sum_k p_k(X_{\mathsf{S}}) \times p_k(\sigma(\mathsf{S}) \setminus X_{\mathsf{S}}), \tag{6}$$

where $p_k(X_{\mathsf{S}})$ are distributions over X_{S} (leaves of the PC) whose supports[3] are disjoint and independent of any parameters, and $p_k(\sigma(\mathsf{S}) \setminus X_{\mathsf{S}})$ are arbitrary PC nodes with scope $\sigma(\mathsf{S}) \setminus X_{\mathsf{S}}$. This form of determinism is very common and known under various names such as *decision* (Darwiche & Marquis, 2002) or *regular selectivity* (Peharz et al., 2014).

The key to computing the Bayes score in deterministic PCs is the *tree decomposition* introduced by Zhao, Adel, et al. (2016). Specifically, they introduced the notion of *induced tree*, a sub-graph of $\mathcal{G}$ obtained by (i) removing all inputs except one from each sum node in $\mathcal{G}$, and (ii) removing all nodes which are rendered unreachable from the root. Since we can decide for each sum node independently which input to remove, the number of induced trees is exponential in the number of sum nodes. They show that the PC distribution can be written as a "large flat mixture," the so-called tree decomposition (cf. also Trapp et al. (2019))

$$p(\mathbf{x} \mid \Theta, \boldsymbol{w}, \mathcal{G}) = \sum_{\tau\in\mathcal{G}} \prod_{\mathsf{SN}\in E} w_{\mathsf{SN}}^{\mathbb{1}[\mathsf{SN}\in\tau]} \prod_{\mathsf{L}\in V} p(\mathbf{x} \mid \theta_{\mathsf{L}})^{\mathbb{1}[\mathsf{L}\in\tau]}, \tag{7}$$

where the sum runs over all induced trees contained in $\mathcal{G}$, the first product runs over all sum-input edges, and the second product over all leaves. Here, $\mathbb{1}[\mathsf{SN} \in \tau]$ is the indicator that the edge between S and N is contained in τ and $\mathbb{1}[\mathsf{L} \in \tau]$ indicates whether L is in τ. Thus, each sum term in (7) is the product of all sum-weights and leaves appearing in τ.

It turns out that, for some parameter set of non-zero measure under prior (4), there is for each $\mathbf{x} \in \mathcal{D}$ *exactly one* non-zero term in the exponential sum in eq. (7), corresponding to a particular induced tree $\tau_{\mathbf{x}}$. First, we can assume without loss of generality that there exists *at least* one non-zero term—otherwise (5) would evaluate to zero, which would be the correct score for the considered candidate structure. Further, in decomposable and deterministic PCs, there can be *at most* one non-zero term, corresponding to a particular induced tree $\tau_{\mathbf{x}}$. This induced tree can

[2] For simplicity, we leave the scope function implicit, i.e. we assume that each leaf "knows" its scope.

[3] The *support* is the set of inputs where the density is non-zero.

be found by tracing the PC backwards from root to leaves, where at sum nodes the single non-zero input is followed and at product nodes all inputs are followed. If at any sum node a zero-input was selected, it can be shown that the corresponding term in eq. (7) evaluates to zero (Peharz et al., 2014). Consequently, eq. (7) reduces to

$$p(\mathbf{x} \,|\, \Theta, \boldsymbol{w}, \mathcal{G}) = \prod_{\mathsf{SN} \in E} w_{\mathsf{SN}}^{\mathbb{1}[\mathsf{SN} \in \tau_{\mathbf{x}}]} \prod_{\mathsf{L} \in V} p(\mathbf{x} \,|\, \theta_{\mathsf{L}})^{\mathbb{1}[\mathsf{L} \in \tau_{\mathbf{x}}]}. \tag{8}$$

Substituting (4) and (8) into (5) and re-arranging products yields

$$\mathcal{B}_{PC}(\mathcal{G}) = \iint \left(\prod_{\mathsf{S} \in V} \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}}-1} \prod_{\mathbf{x} \in \mathcal{D}} w_{\mathsf{SN}}^{\mathbb{1}[\mathsf{SN} \in \tau_{\mathbf{x}}]} \right) \left(\prod_{\mathsf{L} \in V} p(\theta_{\mathsf{L}}) \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x} \,|\, \theta_{\mathsf{L}})^{\mathbb{1}[\mathsf{L} \in \tau_{\mathbf{x}}]} \right) \mathrm{d}\Theta \, \mathrm{d}\boldsymbol{w}. \tag{9}$$

Since we assume parameter-consistent determinism, the induced tree $\tau_{\mathbf{x}}$ remains fixed when ranging over $\Theta, \boldsymbol{w}$. Together with parameter independence, this allows us to pull the integrals into the products, yielding

$$\mathcal{B}_{PC}(\mathcal{G}) = \prod_{\mathsf{S} \in V} \left(\int \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}}+n[\mathsf{SN}]-1} \mathrm{d}\boldsymbol{w}_{\mathsf{S}} \right) \prod_{\mathsf{L} \in V} \left(\int p(\theta_{\mathsf{L}}) \prod_{\mathbf{x} \in \mathcal{D}_{\mathsf{L}}} p(\mathbf{x} \,|\, \theta_{\mathsf{L}}) \, \mathrm{d}\theta_{\mathsf{L}} \right), \tag{10}$$

where $n[\mathsf{SN}] \coloneqq \sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}[\mathsf{SN} \in \tau_{\mathbf{x}}]$ counts how often the edge between S and N appears in an induced tree throughout the dataset, and $\mathcal{D}_{\mathsf{L}} \coloneqq \{\mathbf{x} \in \mathcal{D} \,|\, \mathsf{L} \in \tau_{\mathbf{x}}\}$ is the sub-dataset of samples where L appears in the corresponding induced tree.

The first line of (10) takes essentially the same form as eq. (23) in Heckerman et al. (1995) and yields the widely know *Bayes-Dirichlet* score

$$\prod_{\mathsf{S} \in V} \left(\frac{\Gamma(\alpha_{\mathsf{S}})}{\Gamma(n[\mathsf{S}] + \alpha_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} \frac{\Gamma(n[\mathsf{SN}] + \alpha_{\mathsf{SN}})}{\Gamma(\alpha_{\mathsf{SN}})} \right), \tag{11}$$

where $\alpha_{\mathsf{S}} \coloneqq \sum_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} \alpha_{\mathsf{SN}}$ and $n[\mathsf{S}] \coloneqq \sum_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} n[\mathsf{SN}]$. The factors in the second line of (10) are exactly eq. (2), which we assumed to be tractable. For instance, if the leaves are categorical, the solution is again the Bayes-Dirichlet score; if they are Gaussian, we can use the Bayesian-Gaussian score (Geiger & Heckerman, 1994).

4 BAYESIAN INFORMATION CRITERION

As an alternative to the Bayesian score, we also adapt the *Bayesian information criterion* (BIC) as a structure score for PCs. In general, given a graph structure $\mathcal{G}$ and a training set $\mathcal{D}$, the BIC score is defined as

$$\mathrm{BIC}(\mathcal{G}) = \mathrm{LL}(\mathcal{G}; \mathcal{D}) - \frac{\log|\mathcal{D}|}{2} \|\mathcal{G}\|, \tag{12}$$

where $\mathrm{LL}(\mathcal{G}; \mathcal{D})$ is the log-likelihood under the graph structure $\mathcal{G}$ when using maximum likelihood parameters (potentially with Laplace smoothing), $|\mathcal{D}|$ is the size of $\mathcal{D}$ and $\|\mathcal{G}\|$ is the number of independent parameters encoded in $\mathcal{G}$. The second term $\frac{\log|\mathcal{D}|}{2}\|\mathcal{G}\|$ is essentially a regularization term penalizing complex structures. BIC is often used as an alternative to BD and for Bayesian networks its asymptotic consistency has been shown (Koller & Friedman, 2009).

The BIC score has been used to learn CNets in Di Mauro et al. (2015). However, the authors defined $\|\mathcal{G}\|$ as the number of *conditioning* nodes in the CNet, but did *not* count the parameters of the contained Chow-Liu trees (see next section for details), which actually undercounts the independent parameters in the corresponding PC and weakens the penalty. In our experiments, we show that the corrected BIC score reflects better generalization performance and outperforms Di Mauro et al. (2015).

5 SCORE-BASED CUTSET LEARNING

Ideally, we would like to use our structure scores to find a *global optimum* among all deterministic PCs, which is presumably NP-hard. Thus, in order to use our scores for structure learning, we need a method to "navigate" through the space of deterministic PCs. *Cutset learning* (Rahman et al., 2014) is an attractive method to this end, since it performs greedy search using large structural changes, making it fast and effective. In this section, we provide the required background and adapt cutset learning to our purposes, i.e., guide the search by our structure scores. We will for simplicity consider *binary* data, although cutset learning can be generalized to continuous data as well.

Chow-Liu Trees. A Chow-Liu tree (CLT) is a Bayesian network (Koller & Friedman, 2009), denoted as $(\mathcal{T}, \Theta)$ where $\mathcal{T}$ is a directed tree (meaning each node has at most one parent) over random variables $\mathbf{X}$ and $\Theta = \{\theta_{X|\pi(X)}\}_{X \in \mathbf{X}}$ is a collection of conditional probability tables (CPTs), where $\pi(X)$ denotes the parent of X in $\mathcal{T}$. The Chow-Liu Algorithm (Chow & Liu, 1968) learns the tree structure $\mathcal{T}$ by running a maximum spanning tree algorithm (Kruskal, 1956) on the fully-connected graph over $\mathbf{X}$, weighted with the mutual information between pairs of variables, which is estimated from data. As shown by Chow & Liu, this spanning tree yields a maximizer of the *log-likelihood structure score* (Koller & Friedman, 2009), making the Chow-Liu algorithm an early example of a principled structure learning algorithm. The edges of maximal spanning tree are then directed and Θ is estimated using maximum likelihood.

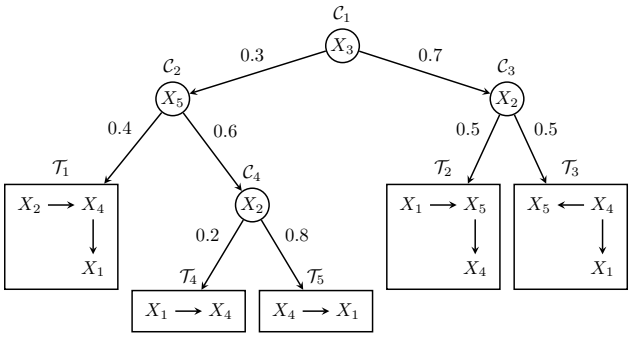


Figure 1: A CNet over binary random variables $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5\}$. Inner nodes $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ are decision nodes whose left (resp. right) branches represent conditioning on state 0 (resp. 1). The leaves of the CNet are CLTs $\{(\mathcal{T}_i, \boldsymbol{\Theta}_i)\}_{i=1}^5$.

Cutset Networks. Cutset networks (CNets) (Rahman et al., 2014) improve CLTs by embedding them in a hierarchical conditioning process. A CNet is a binary decision tree, whose decision nodes correspond to some variable in $\mathbf{X}$ and whose leaves are CLTs over the undecided variables, i.e., the variables not appearing in any decision node on the unique path from root to leaf. Further, the outgoing edges of decision nodes are equipped with normalized weights. For any sample $\mathbf{x}$, the probability assigned by the CNet is the product of weights on the path from root to the selected CLT leaf (following decisions according to the values in $\mathbf{x}$), times the probability the CLT assigns to the undecided variables. An example CNet is shown in Figure 1.

CNets have been extensively studied in literature since they deliver simple, effective and fast structure learning algorithms (Rahman et al., 2014; Di Mauro et al., 2015, 2017; Dang et al., 2020; Di Mauro et al., 2021). CNets can be converted into *smooth, deterministic and decomposable PCs* (Dang et al., 2020; Di Mauro et al., 2021), making them amenable to our structure scores.

Pseudo code for cutset learning with structure scores is shown in Algorithm 1, which recursively selects variables to condition upon. Our strategy is to select a variable, which, when used in a new decision node with two conditional CLT leaves, would increase our score most. This selection is done in SELECTBESTCUT, which uses either our Bayesian or BIC score of the corresponding PC.[4] As testing every unconditioned variable increases runtime linearly in the overall number of variables, we restrict the selection to the set of the λ most promising candidates (SELECTBESTCANDIDATES) according to the information gain heuristic in Rahman et al. (2014). This heuristic, akin to *beam search*, did not impact performance but yielded substantial runtime improvements. Our structure learner stops as soon as no improvement of the score is achieved.

[4] In our implementation we avoided explicit compilation to PCs, but performed the corresponding computations directly within the CNet.

Algorithm 1 CUTSET STRUCTURE LEARNING

Require: A training set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^N$ over binary RVs $\mathbf{X}$; number of candidate conditioning nodes λ.
Ensure: a CNet $\mathcal{C}^*$ representing a deterministic PC.

function CUT($\mathbf{X}, \mathcal{D}, \lambda$)
  $\mathcal{T}^* \leftarrow$ LEARNCLT($\mathbf{X}, \mathcal{D}$)
  $\tilde{\mathbf{X}} \leftarrow$ SELECTBESTCANDIDATES($\mathbf{X}, \mathcal{D}, \lambda$)
  $X^*, \mathcal{C}^*, \mathcal{D}_l, \mathcal{D}_r \leftarrow$ SELECTBESTCUT($\tilde{\mathbf{X}}, \mathcal{D}$)
  if $S(\mathcal{T}^*) > S(\mathcal{C}^*)$ **then**
    return $\mathcal{T}^*$
  else
    $\mathcal{C}^* \leftarrow$ CNET(X^*, CUT($\mathbf{X} \backslash X^*, \mathcal{D}_l, \lambda$), CUT($\mathbf{X} \backslash X^*, \mathcal{D}_r, \lambda$))
    return $\mathcal{C}^*$
  end if
end function

function SELECTBESTCUT($\tilde{\mathbf{X}}, \mathcal{D}$)
  for all $X \in \tilde{\mathbf{X}}$ **do**
    $\mathcal{D}_l, \mathcal{D}_r \leftarrow$ SPLIT($X, \mathcal{D}$)
    $\mathcal{T}_l \leftarrow$ LEARNCLT($\mathbf{X} \backslash X, \mathcal{D}_l$)
    $\mathcal{T}_r \leftarrow$ LEARNCLT($\mathbf{X} \backslash X, \mathcal{D}_r$)
    $\mathcal{C} \leftarrow$ CNET($X, \mathcal{T}_l, \mathcal{T}_r$)
    $S(\mathcal{C}) \leftarrow$ SCORE($\mathcal{C}$)
  end for
  $X^* \leftarrow$ best X with the highest $S(\mathcal{C})$
  return $X^*, \mathcal{C}^*, \mathcal{D}_l, \mathcal{D}_r$
end function

6 STRUCTURAL EXPECTATION-MAXIMIZATION

Since the Bayesian structure score is the (marginal) likelihood of the structure, it can naturally be incorporated in larger probabilistic frameworks such as structural expectation-maximization (Friedman, 1998). Specifically, we consider mixtures of CNets of the form

$$p_{mix}(\mathbf{x}) = \sum_{k=1}^{K} a_k \, p(\mathbf{x} \mid \Theta, \boldsymbol{w}, \mathcal{G})$$

where $a_k \geq 0$ and $\sum_k a_k = 1$, and K is the number of components. We initialize the model by clustering the data set into K clusters using k-means, and training individual CNets on each portion. The structural E-step consists of computing the responsibilities of each component proportional to $\gamma_k \propto a_k \, p(\mathbf{x} \mid \mathcal{G})$, where we use the posterior predictive distribution to average over parameters. The responsibilities are then used in the structural M-step as (i) weighted average to update a_k and (ii) as fractional samples within our cutset learner. It should be noted that also other (heuristic) structure learners can be employed in such a scheme, as they presumably also optimize the structure likelihood in some indirect way.

7 EXPERIMENTS

We evaluate our approach on both density estimation and image completion tasks, where for simplicity we focus on binary data. We use the full Bayesian score—yielding the BDeu score (Heckerman et al., 1995) in the context of binary data, denoted as CNetBD—and the BIC score, denoted as CNetBIC. We limit the number λ of candidate conditioning nodes in these two approaches to ten.

The following structure learners of PCs from literature are used as competitors: the original cutset network (CNet) (Rahman et al., 2014), XCNet (Di Mauro et al., 2017), dCSN (Di Mauro et al., 2015), Strudel (Dang et al., 2020), LearnSPN (Gens & Domingos, 2013) and ID-SPN (Rooshenas & Lowd, 2014). All cutset learners except for dCSN are developed in Python using the DeeProb-kit library (Loconte & Gala, 2022). For LearnSPN, we report results from (Gens & Domingos, 2013). For all other structure learners, we use the open-source implementations from respective authors.

7.1 Standard Density Estimation Benchmarks

As a first experiment, we evaluate all structure learning approaches as density estimators on twenty commonly-used benchmark data sets. For CNetBD and CNetBIC, the equivalent sample size (ESS, uniform Dirichlet parameter α) and the Laplace smoothing factor (β) are the only hyper-parameters, respectively. For all data sets, we fix $\alpha = 0.1$ for CNetBD and $\beta = 0.01$ for CNetBIC, rendering our methods are de facto hyper-parameter-free.

Moreover, the Bayesian nature of our scores should effectively protect against overfitting and directly correspond to generalization. Hence, to demonstrate this benefit, we train on all available data, without the need of a separate validation set. We apply the same procedure (fixed hyperparameter, no validation set) to the more simple structure learners CNet, XCNet and dCSN, and provide the more "sophisticated" structure learners Strudel, LearnSPN, and ID-SPN with a validation set.

Specifically, we re-train CNet on all data sets using the hyper-parameters reported in Rahman et al. (2014). In the case of XCNet and dCSN, we set $\beta = 0.5$, $\delta = 500$ (the minimum number of samples to decompose) and $\xi = 3$ (the minimum number of features to decompose). These hyper-parameter values are expected to perform well on all data sets and are used as default values in their open-source implementations. Due to the randomness of XCNet, we use the average test log-likelihood and report the average time on ten different runs.

The other structure learners—Strudel, LearnSPN and ID-SPN—have more complex learning strategies. Their implementations or reported results all rely on cross-tuning on a separate validation set. Therefore, we use the original training and validation splits for these methods, rather than the combined data.

Ideally, we wish to learn density estimators that are both accurate in terms of the test log-likelihood and efficient in terms of the training time. Therefore, we consider the trade-off between the learning time and the accuracy as a multi-objective optimization problem. Figure 2 shows visualizations of the learning time and test log-likelihoods of all structure learners for each data set. We plot a Pareto frontier as a dashed line for each data set. The Pareto-efficient methods on the line are emphasized with amplified "aura" markers. From Figure 2, we observe that both CNetBD and CNetBIC appear to be Pareto-efficient on most data sets. Compared to other cutset learners with the same greedy learning strategy, CNetBD and CNetBIC achieved leading performance among fast methods. In comparison with more complex learners, CNet and CNetBIC still outperform or are on par with Strudel and LearnSPN on many data sets.

The detailed results for each method are summarized in Table 1. Here we also include the mixture model based learned with structural EM (EM-CNetBD) to compare with the "complex structure learners." Since the mixture model has the number of mixture components as a hyper-parameter, we used the same split in training and validation sets as used for the complex learners, and considered $K \in \{2, 3, 5, 8, 10, 20\}$. We see that mixtures learned with structural EM consistently improved over single models. Moreover, our mixtures are close to or improve state-of-the-art PCs. Additional results can be found in Appendix C.

7.2 Binary MNIST

We further evaluate our models on the Binary-MNIST data set (Larochelle & Murray, 2011). We compare against Einsum networks (EiNets) (Peharz, Lang, et al., 2020), which allows large-scale training due to its tensor-based implementation. CNetBD (50K parameters) achieves a test log-likelihood of -118.71, and a mixture model based on CNetBD with 300 components (470K parameters) has a test log-likelihood of -103.95. In comparison, an EinNet generally has an rather large number of parameters, e.g. EinNet (1.2M parameters) has a test log-likelihood of -113.55 and a larger EinNet (84M parameters) achieves -99.82.

Figure 3 shows image generation from CNetBD and EM models with different numbers of components. We can clearly observe that the generated images become more plausible with increasing the number of components. Moreover, Figure 3 shows in inpainting experiments using approximate MPE inference (Poon & Domingos, 2011), where we can see the reconstructions of covered images from mixture models are plausible.

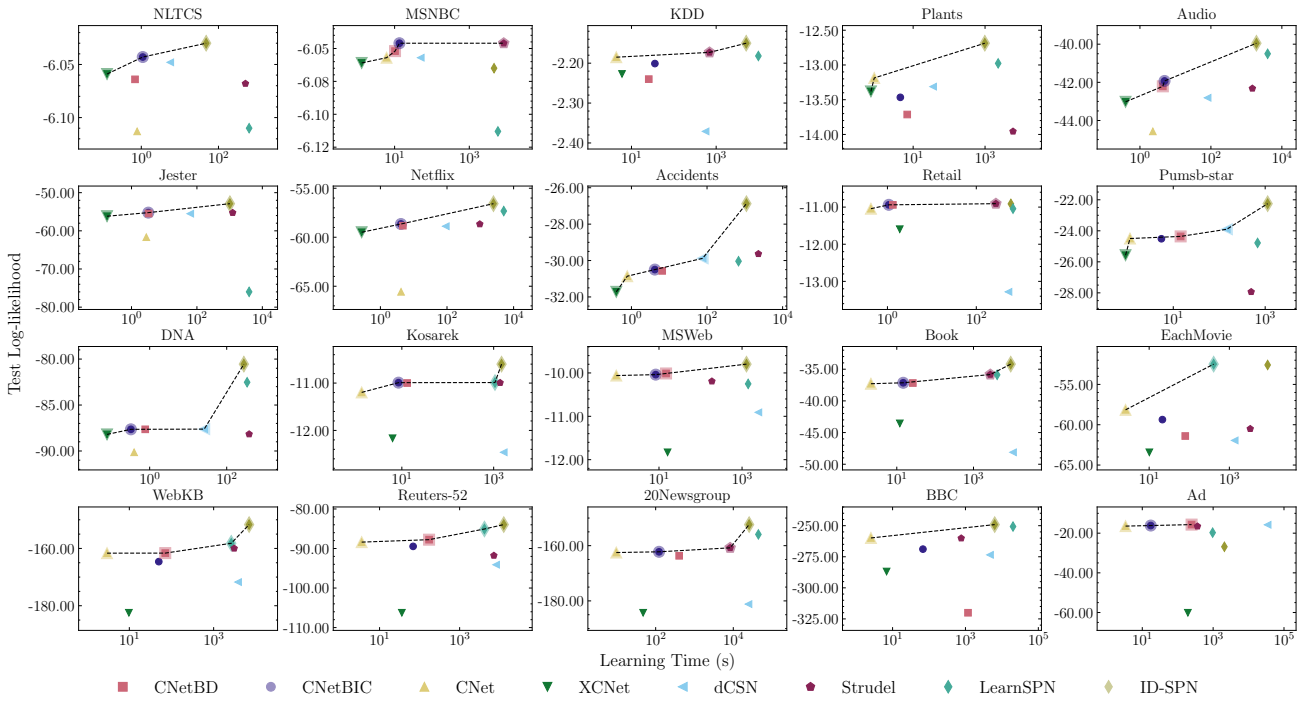
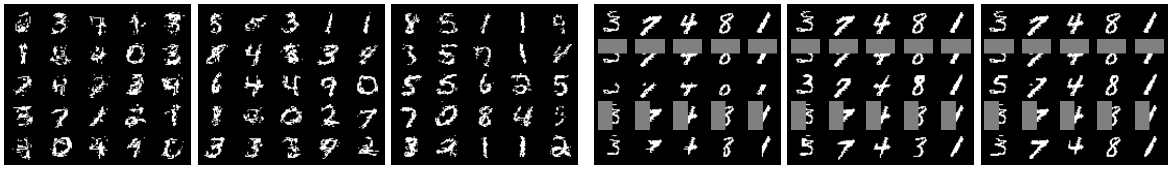


Figure 2: Test log-likelihoods (in nats) vs. learning times of all structure learners on 20 benchmark data sets. The optimal learners on the Pareto curve are highlighted with amplified "aura" markers.

Data Set	CNet	XCNet	dCSN	CNetBD	CNetBIC	Strudel	LearnSPN	ID-SPN	EM-CNetBD
NLTCS	-6.113	-6.059	-6.048	-6.064	**-6.043**	-6.068	-6.110	**-6.030**	-6.032
MSNBC	-6.057	-6.061	-6.057	-6.052	**-6.046**	-6.046	-6.113	-6.065	**-6.041**
KDD	**-2.185**	-2.227	-2.371	-2.240	-2.201	-2.173	-2.182	**-2.150**	-2.159
Plants	**-13.187**	-13.381	-13.312	-13.713	-13.466	-13.956	-12.977	**-12.687**	-12.902
Audio	-44.571	-43.022	-42.810	-42.210	**-41.927**	-42.319	-40.503	-39.950	**-39.943**
Jester	-61.670	-56.173	-55.506	-55.454	**-55.231**	-55.255	-75.989	**-52.886**	**-52.923**
Netflix	-65.572	-59.492	-58.877	-58.820	**-58.649**	-58.659	-57.328	**-56.575**	-56.604
Accidents	-30.860	-31.720	**-29.880**	-30.568	-30.497	-29.633	-30.038	**-26.876**	-29.919
Retail	-11.043	-11.595	-13.282	-10.940	**-10.939**	-10.908	-11.043	**-10.913**	-10.927
Pumsb-star	-24.497	-25.575	**-23.900**	-24.361	-24.513	-27.935	-24.781	**-22.251**	-24.009
DNA	-90.143	-88.199	**-87.621**	-87.643	-87.642	-88.167	-82.523	**-80.550**	-85.437
Kosarek	-11.197	-12.165	-12.462	-10.999	**-10.991**	-10.993	-10.989	**-10.600**	-10.695
MSWeb	-10.060	-11.830	-10.909	**-10.012**	-10.039	-10.192	-10.252	**-9.797**	-9.963
Book	-37.293	-43.550	-48.116	-37.181	**-37.143**	-35.841	-35.886	**-34.200**	-34.854
EachMovie	**-58.153**	-63.420	-61.946	-61.400	-59.365	-60.503	**-52.485**	-52.588	-53.378
WebKB	-161.650	-182.453	-171.743	**-161.634**	-164.611	-159.989	-158.204	**-151.680**	-157.053
Reuters-52	-88.386	-106.251	-94.102	**-87.766**	-89.461	-91.778	-85.067	**-83.954**	-84.993
20Newsgroup	-162.488	-184.297	-181.165	-163.671	**-162.227**	-160.786	-155.925	**-152.37**	-155.002
BBC	**-259.870**	-286.869	-273.465	-320.052	-268.861	-260.084	-250.687	**-249.12**	-258.691
Ad	-16.436	-60.120	-15.774	**-15.676**	-16.161	-16.521	-19.733	-26.85	**-14.923**

Table 1: Average test log-likelihoods (in nats) for all structure learners. The cutset learners are in the left group and other structure learners are in the right group. The model with the best performance in each group is highlighted in bold.



(a) Sampling results. (b) Inpainting results.

Figure 3: Samples (a) and inpainting results (b) for Binary-MNIST for CNetBD (left), EM-CNetBD with 100 components (middle) and 300 components (right). In (b), the original test images are shown in the top row; then in alternating rows the images with missing part and reconstructions are shown.

8 CONCLUSION

Structure learning is a challenging task in PCs. While many techniques have been proposed in recent years, most of them are based on heuristics rather than on principled learning objectives. This is in stark contrast to classical PGMs, where constraint-based and score-based approaches have been used. Our main motivation in this paper is to establish such principled approaches also for PC structure learning.

In particular, we derived the Bayesian score known from the PGM literature to structure learning of *deterministic* PCs, and furthermore proposed a correction to the BIC score for PCs. Both scores were used in a simple structure learner and were allowed to "speak for themselves:" no extensive cross-validation and no other defenses against overfitting, such as early stopping, were required. The results demonstrate that such a simple and principled approach can be effective and efficient. We hope that our results stimulate further research in this direction.

Our approach is appealing both from a theoretical and practical perspective, since our score is derived from probabilistic considerations, while being analytically tractable and fast in practice. The main caveat is that our score is restricted to *deterministic* circuits. Future directions, however, might include approximations of these scores to general (non-deterministic) PC, using e.g. sampling (Vergari et al., 2019; Trapp et al., 2019), variational techniques (Zhao, Adel, et al., 2016), continuous relaxations (Lang et al., 2022), or numeric integration (Correia et al., 2022).

Further interesting directions are studying the theoretical properties of Bayesian scores for PCs. For example, for PGMs, consistency of both the Bayesian and BIC score has been shown, meaning that they will identify the correct structure in the infinite data limit, provided that there exists a PGM which is a perfect map for the data-generating distribution (Koller & Friedman, 2009). It is likely that such properties also hold for PCs, which might explain the similar performance of the BD and BIC scores in our experiments. However, rigorously establishing these results is left to future work.

Acknowledgements

We thank the Eindhoven Artificial Intelligence Systems Institute (EAISI) for its support. This research was supported by the Graz Center for Machine Learning (GraML). We thank YooJung Choi, Antonio Vergari, and Martin Trapp for helpful discussions.

References

Adel, T., Balduzzi, D., & Ghodsi, A. (2015). Learning the structure of sum-product networks via an SVD-based algorithm. In *UAI* (pp. 32–41).

Ahmed, K., Teso, S., Chang, K.-W., den Broeck, G. V., & Vergari, A. (2022). Semantic probabilistic layers for neuro-symbolic learning. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in Neural Information Processing Systems.*

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Uncertainty Proceedings 1991* (pp. 52–60).

Choi, Y., Vergari, A., & Van den Broeck, G. (2020). Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA. URL: http://starai. cs. ucla. edu/papers/ProbCirc20. pdf*.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *Transactions on Information Theory*, *14*(3), 462–467.

Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*(4), 309–347.

Correia, A., Gala, G., Quaeghebeur, E., de Campos, C., & Peharz, R. (2022). Continuous mixtures of tractable probabilistic models. *arXiv preprint arXiv:2209.10584*.

Dang, M., Vergari, A., & Broeck, G. (2020). Strudel: Learning structured-decomposable probabilistic circuits. In *International Conference on Probabilistic Graphical Models* (pp. 137–148).

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

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

De Campos, C. P., Zeng, Z., & Ji, Q. (2009). Structure learning of Bayesian networks using constraints. In *International Conference on Machine Learning* (pp. 113–120).

Di Mauro, N., Gala, G., Iannotta, M., & Basile, T. M. (2021). Random probabilistic circuits. In *Uncertainty in Artificial Intelligence* (pp. 1682–1691).

Di Mauro, N., Vergari, A., Basile, T., & Esposito, F. (2017). Fast and accurate density estimation with extremely randomized cutset networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 203–219).

Di Mauro, N., Vergari, A., & Esposito, F. (2015). Learning accurate cutset networks by exploiting decomposability. In *Congress of the Italian Association for Artificial Intelligence* (pp. 221–232).

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Conference on Uncertainty in Artificial Intelligence* (pp. 129–138).

Friedman, N., & Koller, D. (2003). Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, *50*(1), 95–125.

Geiger, D., & Heckerman, D. (1994). Learning Gaussian networks. In *Uncertainty Proceedings 1994* (pp. 235–243).

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

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452–459.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*(3), 197–243.

Khosravi, P., Liang, Y., Choi, Y., & Van den Broeck, G. (2019). What to expect of classifiers? Reasoning about logistic regression with missing features. *arXiv preprint arXiv:1903.01620*.

Kisa, D., Van den Broeck, G., Choi, A., & Darwiche, A. (2014). Probabilistic sentential decision diagrams. In *International Conference on the Principles of Knowledge Representation and Reasoning*.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, *7*(1), 48–50.

Lang, S., Mundt, M., Ventola, F., Peharz, R., & Kersting, K. (2022). Elevating perceptual sample quality in PCs through differentiable sampling. In *NeurIPS 2021 workshop on pre-registration in machine learning* (Vol. 181, pp. 1–25).

Larochelle, H., & Murray, I. (2011). The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics* (pp. 29–37).

Liu, A., Zhang, H., & Broeck, G. V. d. (2022). Scaling up probabilistic circuits by latent variable distillation. *arXiv preprint arXiv:2210.04398*.

Loconte, L., & Gala, G. (2022). Deeprob-kit: a python library for deep probabilistic modelling. *arXiv preprint arXiv:2212.04403*.

Lowd, D., & Domingos, P. (2008). Learning arithmetic circuits. In *Conference on Uncertainty in Artificial Intelligence* (p. 383-392).

Peharz, R., Geiger, B. C., & Pernkopf, F. (2013). Greedy part-wise learning of sum-product networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 612–627).

Peharz, R., Gens, R., & Domingos, P. (2014). Learning selective sum-product networks. In *ICML Workshop on Learning Tractable Probabilistic Models*.

Peharz, R., Gens, R., Pernkopf, F., & Domingos, P. (2016). On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(10), 2030–2044.

Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., ... Ghahramani, Z. (2020). Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning* (pp. 7563–7574).

Peharz, R., Tschiatschek, S., Pernkopf, F., & Domingos, P. (2015). On theoretical properties of sum-product networks. In *International Conference on Artificial Intelligence and Statistics* (pp. 744–752).

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., ... Ghahramani, Z. (2020). Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Conference on Uncertainty in Artificial Intelligence* (pp. 334–344).

Pipatsrisawat, K., & Darwiche, A. (2008). New compilation languages based on structured decomposability. In *AAAI Conference on Artificial Intelligence* (Vol. 8, pp. 517–522).

Poon, H., & Domingos, P. (2011). Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (pp. 689–690).

Rahman, T., Kothalkar, P., & Gogate, V. (2014). 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* (pp. 630–645).

Rashwan, A., Poupart, P., & Zhitang, C. (2018). Discriminative training of sum-product networks by extended Baum-Welch. In *International Conference on Probabilistic Graphical Models* (pp. 356–367).

Rashwan, A., Zhao, H., & Poupart, P. (2016). Online and distributed Bayesian moment matching for parameter learning in sum-product networks. In *International Conference on Artificial Intelligence and Statistics* (pp. 1469–1477).

Rooshenas, A., & Lowd, D. (2014). Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning* (pp. 710–718).

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 461–464.

Shao, X., Molina, A., Vergari, A., Stelzner, K., Peharz, R., Liebig, T., & Kersting, K. (2022). Conditional sum-product networks: Modular probabilistic circuits via gate functions. *International Journal of Approximate Reasoning*, *140*, 298–313.

Trapp, M., Peharz, R., Ge, H., Pernkopf, F., & Ghahramani, Z. (2019). Bayesian learning of sum-product networks. *Advances in Neural Information Processing Systems*, *32*.

Vergari, A., Choi, Y., Liu, A., Teso, S., & Van den Broeck,

G. (2021). A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, *34*, 13189–13201.

Vergari, A., Choi, Y., Peharz, R., & Van den Broeck, G. (2020). Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the The 34th AAAI Conference on Artificial Intelligence.*

Vergari, A., Molina, A., Peharz, R., Ghahramani, Z., Kersting, K., & Valera, I. (2019). Automatic Bayesian density analysis. In *AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 5207–5215).

Zhao, H., Adel, T., Gordon, G., & Amos, B. (2016). Collapsed variational inference for sum-product networks. In *International Conference on Machine Learning* (pp. 1310–1318).

Zhao, H., Poupart, P., & Gordon, G. J. (2016). A unified approach for learning the parameters of sum-product networks. *Advances in Neural Information Processing Systems*, *29*.

A Learning a Chow-Liu Tree

The algorithm LEARNCLT for learning a Chow-Liu tree is shown in Algorithm 2. Overall, for a dataset $\mathcal{D}$, the algorithm has time complexity $O(|\mathbf{X}|^2|\mathcal{D}|)$.

Algorithm 2 LEARNCLT

Require: A training set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ over RVs $\mathbf{X}$, equivalent sample size (ESS) α.
Ensure: a parameterized Chow-Liu tree $\mathcal{T}$

1: **function** LEARNCLT($\mathcal{D}, \alpha$)
2: $\quad\mathbf{MI} \leftarrow$ COMPUTEMUTUALINFORMATION($\mathbf{X}$)
3: $\quad\mathcal{T}' \leftarrow$ COMPUTEMAXIMUMSPANNINGTREE($\mathbf{MI}$)
4: $\quad\mathcal{T} \leftarrow$ ASSIGNEDGEDIRECTIONS($\mathcal{T}'$)
5: $\quad\mathcal{T} \leftarrow$ COMPUTEBAYESIANPOSTERIORPARAMETERS($\mathcal{D}, \mathcal{T}, \alpha$)
6: $\quad$**return** $\mathcal{T}$
7: **end function**

B Candidate Selection Heuristic

We adapt the information gain heuristic introduced by Rahman et al. (2014) to restrict the conditioning node selection to a fixed number of candidates. Intuitively, the heuristic wishes to select variables with maximum information gain or expected entropy reduction by conditioning on the variables. Concretely, the entropy $\hat{H}(\mathcal{D})$ of a data set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ over random variables $\mathbf{X}$ is defined as

$$\hat{H}(\mathcal{D}) = \frac{1}{|\mathbf{X}|} \sum_{X \in \mathbf{X}} H_{\mathcal{D}}(X), \tag{13}$$

where $H_{\mathcal{D}}(X)$ is the entropy of RV X given $\mathcal{D}$, which is given by

$$H_{\mathcal{D}}(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x), \tag{14}$$

where $\mathcal{X}$ is the set of all possible states of X when assuming X is discrete. Based on the entropy of the data, the information gain $\mathrm{Gain}(X_i)$ after conditioning on an RV X_i is defined as

$$\mathrm{Gain}(X_i) = \hat{H}(\mathcal{D}) - \sum_{x_i \in \mathcal{X}_i} \frac{|\mathcal{D}_{x_i}|}{|\mathcal{D}|} \hat{H}(\mathcal{D}_{x_i}), \tag{15}$$

where $\mathcal{D}_{x_i} = \{\mathbf{x}^{(n)} \in \mathcal{D} \mid x_i^{(n)} = x_i\}$. In SELECTBESTCANDIDATES, we select top λ RVs with the maximum information gain to form a candidate set $\tilde{\mathbf{X}}$.

C Additional Experimental Results

C.1 Data Sets

The characteristics of the 20 benchmark density estimation data sets are reported in Table 2. The number of variables ranges from 16 to 1556.

C.2 Learning Time

To evaluate the efficiency of different structure learners, we report their learning times in Table 3, which are all measured in total seconds during training. Combining with the test log-likelihoods of various structure learners, we can observe both CNetBD and CNetBIC achieve a better trade-off between the performance and the training time.

C.3 Circuit Size

Table 4 summarizes the circuit size of models learned by different cutset structure learners. Clearly, CNetBD and CNetBIC deliver much smaller circuits than other cutset learners on almost all data sets. In addition, compare to other cutset learners, CNetBD and CNetBIC often achieve leading performance. These further validate our theoretical analysis that both the BDeu and the BIC score can effectively protect against overfitting.

Dataset	#Features	#Train	#Valid.	#Test
NLTCS	16	16181	2157	3236
MSNBC	17	291326	38843	58265
KDD	64	180092	19907	34955
Plants	69	17412	2321	3482
Audio	100	15000	2000	3000
Jester	100	9000	1000	4116
Netflix	100	15000	2000	3000
Accidents	111	12758	1700	2551
Retail	135	22041	2938	4408
Pumsb-star	163	12262	1635	2452
DNA	180	1600	400	1186
Kosarek	190	33375	4450	6675
MSWeb	294	29441	3270	5000
Book	500	8700	1159	1739
EachMovie	500	4525	1002	591
WebKB	839	2803	558	838
Reuters-52	889	6532	1028	1540
20Newsgroup	910	11293	3764	3764
BBC	1058	1670	225	330
Ad	1556	2461	327	491

Table 2: Characteristics of the benchmark density estimation data sets

Data Set	CNet	XCNet	dCSN	CNetBD	CNetBIC	Strudel	LearnSPN	ID-SPN
NLTCS	0.8	0.1	5.6	0.7	1.1	500	623	48
MSNBC	6.1	1.3	50.2	10.3	13.5	8374	5843	4604
KDD	4.4	6.0	563.4	25.6	35.7	697	9943	5126
Plants	0.8	0.7	35.9	6.8	4.4	5999	2318	986
Audio	2.3	0.4	80.8	4.4	4.9	1476	3977	1920
Jester	2.8	0.2	62.6	3.3	3.3	1239	3946	1020
Netflix	4.2	0.3	92.0	4.7	4.1	966	5062	2485
Accidents	0.8	0.4	76.5	6.5	4.2	2284	681	1102
Retail	0.4	1.9	530.6	1.3	1.1	276	671	601
Pumsb-star	1.1	0.9	146.4	14.5	5.5	496	684	1132
DNA	0.4	0.1	26.1	0.8	0.3	382	339	279
Kosarek	1.3	6.3	1658.7	13.1	8.5	1381	1068	1485
MSWeb	0.9	16.1	2474.1	14.8	8.3	188	1404	1279
Book	2.1	12.0	10729.9	26.2	14.7	2742	4137	9355
EachMovie	2.6	10.1	1360.6	79.4	21.4	3318	406	9014
WebKB	3.0	9.8	3800.4	71.2	50.0	3069	2616	6943
Reuters-52	3.6	36.0	8156.3	172.6	68.9	7166	4166	12653
20Newsgroup	9.7	47.5	24660.7	403.3	122.1	8303	44341	25637
BBC	2.5	6.8	4718.8	1175.7	68.1	765	20199	6348
Ad	3.4	193.7	34576.8	246.5	17.3	361	969	2080

Table 3: Learning times (in seconds) of all structure learners. Due to the randomness of XCNet, we use the average learning time on 10 different runs.

Data Set	CNet	XCNet	dCSN	CNetBD	CNetBIC
NLTCS	4543	1089.4	1017	205	315
MSNBC	20143	5414.0	7069	1695	1865
KDD	8601	47638.4	47888	2823	1651
Plants	18605	21263.8	8532	3399	3163
Audio	54651	15166.0	12530	1549	3437
Jester	72873	7037.6	6000	1359	2689
Netflix	100497	10412.6	8258	1741	2881
Accidents	13763	15273.6	9345	2355	2775
Retail	3667	52036.2	85730	269	269
Pumsb-star	16023	32990.8	12203	4129	3187
DNA	5625	2329.0	1767	359	359
Kosarek	10725	175426.0	166168	1875	1875
MSWeb	5801	455462.6	158034	2333	1753
Book	13887	219122.2	166226	2989	3981
EachMovie	16857	185578.2	18787	8931	6959
WebKB	10035	86246.6	16692	5023	10035
Reuters-52	10635	334000.0	35350	10633	12403
20Newsgroup	30783	402505.8	63209	23531	19939
BBC	6337	43752.4	12646	58849	10551
Ad	6219	1076052.0	77207	9325	3111

Table 4: Circuit sizes (measured in the number of independent parameters) of all structure learners. Due to the randomness of XCNet, we use the average circuit size on 10 different runs.