Bayesian Structure Scores for Probabilistic Circuits

Yang Yang [*,a,c] Gennaro Gala [*,a] Robert Peharz [b,a]

[a]Eindhoven University of Technology, The Netherlands [b]Graz University of Technology, Austria [c]KU Leuven, Belgium

Problem Setup

Probabilistic Circuits (PCs) [6] are a powerful framework for describing Tractable Probabilistic Models (TPMs), which allow various tractable inference routines, including *marginalization*, *conditioning*, *most-probable explanation*, *expectations*, etc.

Similar to Probabilistic Graphical Models (PGMs), we can distinguish between *parameter learning* and *structure learning* when learning PCs from data.

In PCs, ***structure learning*** is generally less principled than in PGMs. Previous PC structure learners are often based on heuristics rather than principled objectives, for instance:

- *LearnSPN* [2] uses top-down multi-clustering to lay out the PC structure;
- *ID-SPN* [5] augments LearnSPN by using expressive distribution models as sub-modules;
- *Cutset networks (CNets)* [4] exploit decision tree algorithm and use *Chow-Liu trees (CLTs)* as leaf models.

None of the above approaches declare an *explicit objective* for structure learning. Moreover, they often rely on a large number of hyper-parameters, which need to be tuned on a separate validation set.

CONTRIBUTION

We propose a principled avenue to PC structure learning, putting particular emphasis on elegance and simplicity.

- We derive ***Bayesian structure scores*** for deterministic PCs by equipping parameters with Dirichlet priors, yielding the well-known ***Bayes-Dirichlet (BD)*** score.
- In addition, we present a correction for the ***Bayesian information criterion (BIC)***, a viable and efficient alternative to the BD score.
- We employ the BD and the BIC scores with ***cutset learning***, a simple and fast structure learner for PCs.
- Our approach has only one single hyper-parameter governing the Dirichlet prior, the *equivalent sample size (ESS)*, which we keep fixed to 0.1 throughout experiments, rendering our method effectively ***hyper-parameter-free***.
- Moreover, we embed our structure learner within a ***structural expectation maximization (EM)*** algorithm, which has not been applied to PCs before. This consistently improves test log-likelihood over single PCs and delivers mixture models close to or surpassing state-of-the-art.

Bayes-Dirichlet Score for Deterministic PCs

Let us consider a training set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ of i.i.d. samples and a distribution $p(\mathbf{x} \,|\, \Theta, \boldsymbol{w}, \mathcal{G})$ where $\mathcal{G} = (V, E)$ is the PC computational graph, $\Theta = \{\theta_{\mathsf{L}}\}_{\mathsf{L} \in V}$ parameters of the leaf nodes and $\boldsymbol{w} = \{\boldsymbol{w}_{\mathsf{S}}\}_{\mathsf{S} \in V}$ sum weights. The basic idea of the *Bayes-Dirichlet* score is to :

1. equip Θ and $\boldsymbol{w}$ with a Dirichlet prior distribution
$$p(\Theta, \boldsymbol{w} \,|\, \mathcal{G}) = \prod_{\mathsf{S} \in V} p(\boldsymbol{w}_{\mathsf{S}}) \prod_{\mathsf{L} \in V} p(\theta_{\mathsf{L}}) = \prod_{\mathsf{S} \in V} \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}}-1} \prod_{\mathsf{L} \in V} p(\theta_{\mathsf{L}}). \tag{1}$$
2. derive the marginal likelihood of $\mathcal{G}$ by marginalizing the parameters from the Bayesian model:
$$\mathcal{B}_{PC}(\mathcal{G}) = \int\!\!\int p(\Theta, \boldsymbol{w} \,|\, \mathcal{G}) \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x} \,|\, \Theta, \boldsymbol{w}, \mathcal{G}) \, \mathrm{d}\Theta \, \mathrm{d}\boldsymbol{w}. \tag{2}$$

The key insight to computing the BD score in deterministic PCs is the notion of *induced tree* [7], a subgraph of $\mathcal{G}$ obtained by

- removing all inputs except one from each sum node in $\mathcal{G}$, and
- removing all nodes which are rendered unreachable from the root.

In decomposable and deterministic PCs, there can be *at most* one non-zero term, corresponding to a particular induced tree $\tau_{\mathbf{x}}$. Based on the so-called *tree decomposition*, the PC distribution can be written as
$$p(\mathbf{x} \,|\, \Theta, \boldsymbol{w}, \mathcal{G}) = \prod_{\mathsf{SN} \in E} w_{\mathsf{SN}}^{\mathbb{1}[\mathsf{SN} \in \tau_{\mathbf{x}}]} \prod_{\mathsf{L} \in V} p(\mathbf{x} \,|\, \theta_{\mathsf{L}})^{\mathbb{1}[\mathsf{L} \in \tau_{\mathbf{x}}]} \tag{3}$$

With the assumption of *parameter-consistent determinism* and the *parameter independence*, we rewrite eq. (2) as
$$\mathcal{B}_{PC}(\mathcal{G}) = \prod_{\mathsf{S} \in V} \left(\int \frac{1}{B(\boldsymbol{\alpha}_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} w_{\mathsf{SN}}^{\alpha_{\mathsf{SN}} + n[\mathsf{SN}] - 1} \mathrm{d}\boldsymbol{w}_{\mathsf{S}} \right) \prod_{\mathsf{L} \in V} \left(\int p(\theta_{\mathsf{L}}) \prod_{\mathbf{x} \in \mathcal{D}_{\mathsf{L}}} p(\mathbf{x} \,|\, \theta_{\mathsf{L}}) \, \mathrm{d}\theta_{\mathsf{L}} \right), \tag{4}$$
which takes essentially the same form as eq. (23) in [3] and yields the widely know ***Bayes-Dirichlet*** score
$$\prod_{\mathsf{S} \in V} \left(\frac{\Gamma(\alpha_{\mathsf{S}})}{\Gamma(n[\mathsf{S}] + \alpha_{\mathsf{S}})} \prod_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} \frac{\Gamma(n[\mathsf{SN}] + \alpha_{\mathsf{SN}})}{\Gamma(\alpha_{\mathsf{SN}})} \right), \tag{5}$$
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, $\alpha_{\mathsf{S}} \coloneqq \sum_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} \alpha_{\mathsf{SN}}$ and $n[\mathsf{S}] \coloneqq \sum_{\mathsf{N} \in \mathbf{in}(\mathsf{S})} n[\mathsf{SN}]$.

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 [1]. Specifically, we consider mixtures of PCs of the form
$$p_{mix}(\mathbf{x}) = \sum_{k=1}^{K} a_k \, p(\mathbf{x} \,|\, \Theta, \boldsymbol{w}, \mathcal{G})$$
where $a_k \geq 0$ and $\sum_k a_k = 1$, and K is the number of components.

- The structural E-step consists of computing the responsibilities of each component proportional to $\gamma_k \propto a_k \, p(\mathbf{x} \,|\, \mathcal{G})$, 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.

Score-based Cutset Learning

Cutset networks (CNets) [4] 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. 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.

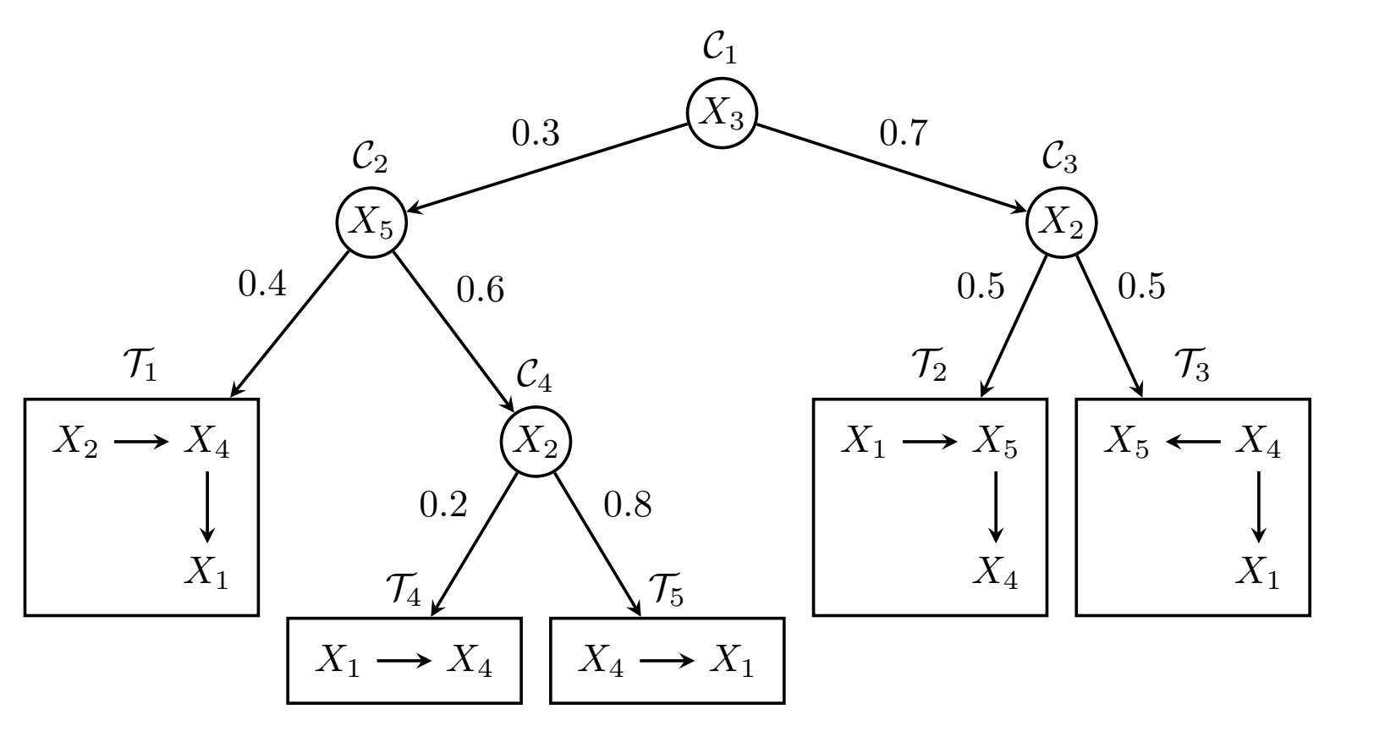

Figure 1. A CNet over binary random variables $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5\}$. Inner nodes $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ are decision nodes whose left (resp. right) branches represent conditioning on state 0 (resp. 1). The leaves of the CNet are CLTs $\{(\mathcal{T}_i,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

Algorithm 1 Score-based 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.

```
1:  function Cut($\mathbf{X}, \mathcal{D}, \lambda$)
2:      $\mathcal{T}^* \leftarrow$ LearnCLT($\mathbf{X}, \mathcal{D}$)
3:      $\tilde{\mathbf{X}} \leftarrow$ SelectBestCandidates($\mathbf{X}, \mathcal{D}, \lambda$)
4:      $X^*, \mathcal{C}^*, \mathcal{D}_l, \mathcal{D}_r \leftarrow$ SelectBestCut($\tilde{\mathbf{X}}, \mathcal{D}$)
5:      if $S(\mathcal{T}^*) > S(\mathcal{C}^*)$ then
6:          return $\mathcal{T}^*$
7:      else
8:          $\mathcal{C}^* \leftarrow$ CNet($X^*$, Cut($\mathbf{X} \backslash X^*, \mathcal{D}_l, \lambda$), Cut($\mathbf{X} \backslash X^*, \mathcal{D}_r, \lambda$))
9:          return $\mathcal{C}^*$
10:     end if
11: end function
```

```
1:  function SelectBestCut($\tilde{\mathbf{X}}, \mathcal{D}$)
2:      for all $X \in \tilde{\mathbf{X}}$ do
3:          $\mathcal{D}_l, \mathcal{D}_r \leftarrow$ Split($X, \mathcal{D}$)
4:          $\mathcal{T}_l \leftarrow$ LearnCLT($\mathbf{X} \backslash X, \mathcal{D}_l$)
5:          $\mathcal{T}_r \leftarrow$ LearnCLT($\mathbf{X} \backslash X, \mathcal{D}_r$)
6:          $\mathcal{C} \leftarrow$ CNet($X, \mathcal{T}_l, \mathcal{T}_r$)
7:          $S(\mathcal{C}) \leftarrow$ score($\mathcal{C}$)
8:      end for
9:      $X^* \leftarrow$ best $X$ with the highest $S(\mathcal{C})$
10:     return $X^*, \mathcal{C}^*, \mathcal{D}_l, \mathcal{D}_r$
11: end function
```

Experimental Results

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.

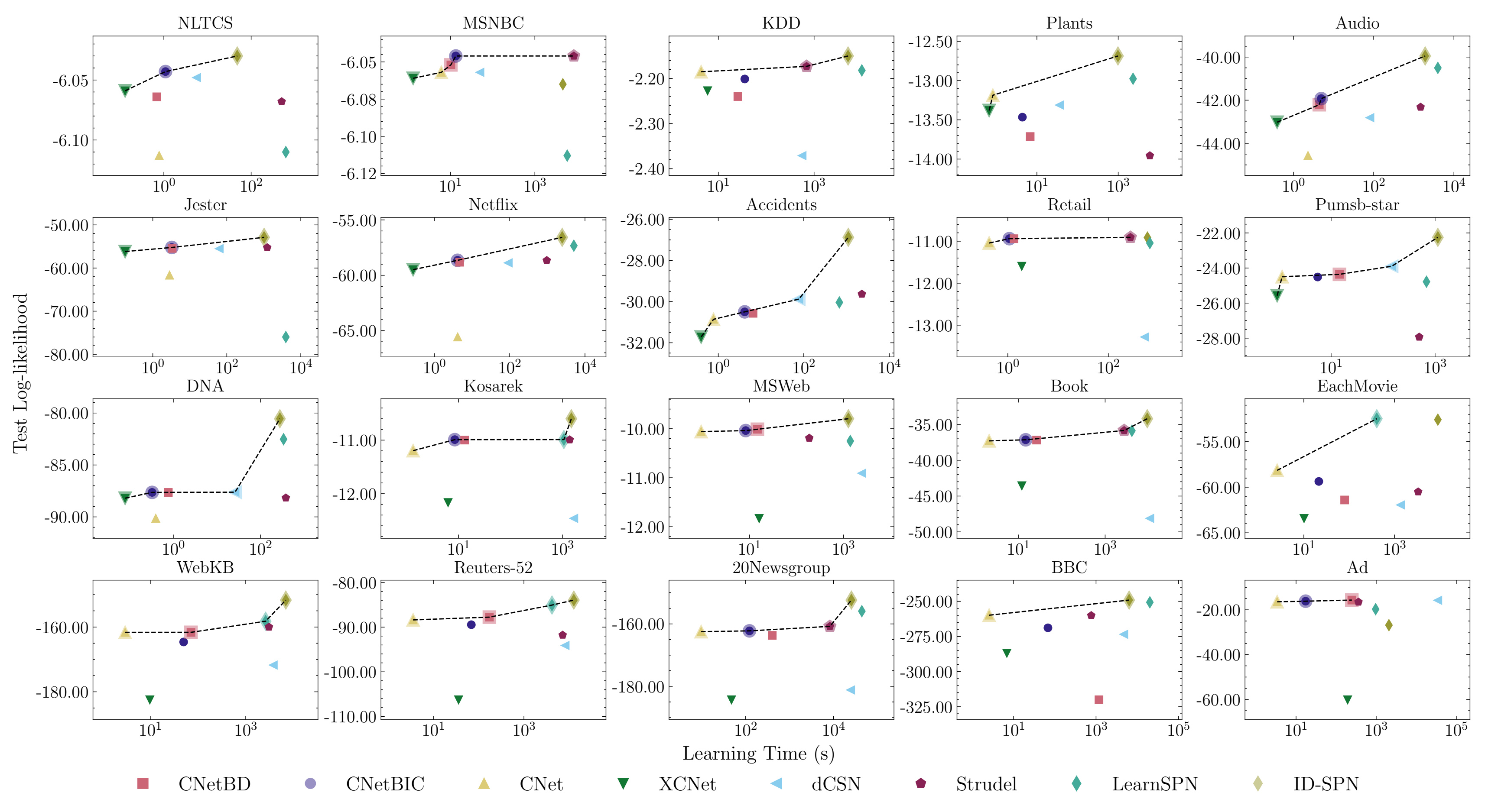

Figure 2. Accuracy and learning times of all structure learners on the 20 benchmark data sets. The optimal learners on the Pareto curve are highlighted with bigger markers.

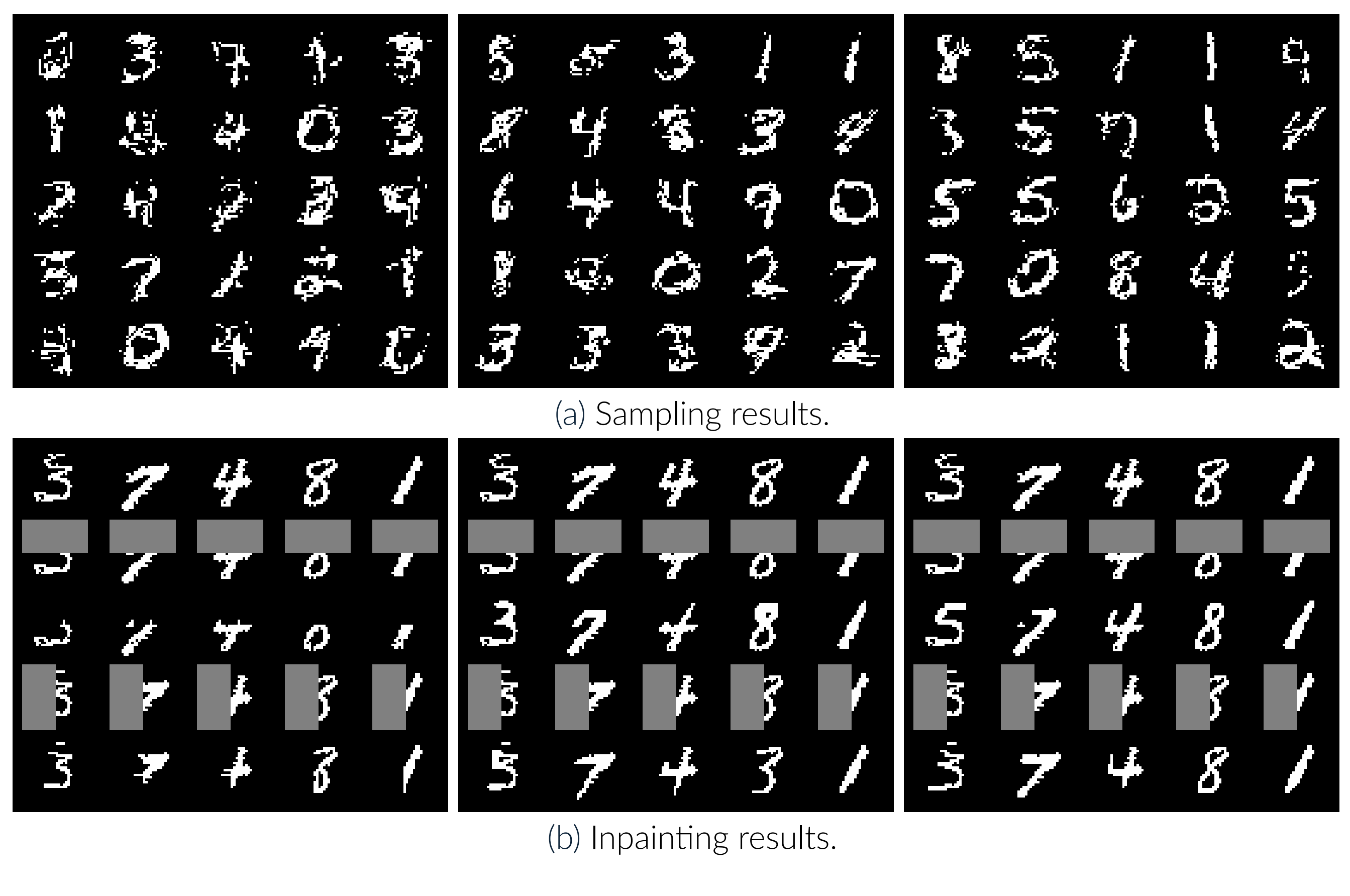

Figure 3. Samples (a) and inpainting results (b) for Binary-MNIST for CNetBD (left), EM-CNetBD with 100 components (middle) and 300 components (right). In (b), the original test images are shown in the top row; then in alternating rows the images with missing part and reconstructions are shown.

References

[1] N. Friedman. The Bayesian structural EM algorithm. In *Conference on Uncertainty in Artificial Intelligence*, pages 129–138, 1998.
[2] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pages 873–880. PMLR, 2013.
[3] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.
[4] T. Rahman, P. Kothalkar, and V. 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, 2014.
[5] A. Rooshenas and D. Lowd. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning*, pages 710–718. PMLR, 2014.
[6] A. Vergari, Y. Choi, R. Peharz, and G. Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the The 34th AAAI Conference on Artificial Intelligence*, 2020.
[7] H. Zhao, T. Adel, G. Gordon, and B. Amos. Collapsed variational inference for sum-product networks. In *International Conference on Machine Learning*, pages 1310–1318, 2016.

yang.yang@kuleuven.be g.gala@tue.nl robert.peharz@tugraz.at