Bayesian Learning of Sum-Product Networks

Martin Trapp[1,2], Robert Peharz[3] Hong Ge[3], Franz Pernkopf[1], and Zoubin Ghahramani[4,3]

[1] Graz University of Technology, [2] Austrian Research Institute for AI, [3] University of Cambridge, Cambridge, and [4]Uber AI

martin.trapp@tugraz.at, rp587@cam.ac.uk, hg344@cam.ac.uk, pernkopf@tugraz.at, zoubin@eng.cam.ac.uk

Sum-product networks (SPNs) [Poon2011] are flexible density estimators and have received significant attention due to their attractive inference properties.
While parameter learning in SPNs is well developed, most structure learners are somewhat ad-hoc and based on intuition rather than a clear learning principle.

In this work, we introduce a well-principled Bayesian framework for SPN structure learning.
We decompose the problem into:

1) laying out a computational graph, and
2) learning the so-called scope function over the graph.

The first encodes the topological layout of the nodes, while the scope function characterises the effective structure. In our framework, only the scope function has to respect the usual structural constraints in SPNs, i.e. completeness and decomposability.

We propose a natural parametrisation for an important and widely used special case of SPNs. These structural parameters are incorporated into a Bayesian model, allowing us to perform posterior inference over structure and parameters jointly.

Let $\mathbf{X} = \{X_1, \dots, X_D\}$ be a set of D random variables (RVs), for which N i.i.d. samples are available. An SPN is a 4-tuple $\mathcal{S} = (\mathcal{G}, \psi, \mathbf{w}, \theta)$

computational graph · scope functions · sum-weights · leaf parameters

The computational graph is a connected directed acyclic graph, containing three types of nodes: sums (S), products (P) and leaves (L). Generic nodes are denotes as N.

The scope function is a function $\psi \colon \mathsf{N} \mapsto 2^{\mathbf{X}}$ assigning each node in the graph a sub-set of X.

Completeness:	For each $\mathsf{S} \in \mathcal{G}$ we have $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{S}) \colon \psi(\mathsf{N}) = \psi(\mathsf{N}')$
Decomposability:	For each $\mathsf{P} \in \mathcal{G}$ we have $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{P}) \colon \psi(\mathsf{N}) \cap \psi(\mathsf{N}') = \emptyset$

The key insight for Bayesian parameter learning [Zhao2016, Rashwan2016, Vergari2019] is that sum nodes can be interpreted as latent variables Z_S, clustering data instances.

$$\mathcal{S}(\mathbf{x}) = \sum_{\mathbf{z}} \prod_{\mathsf{S} \in \mathbf{S}} w_{\mathsf{S}, z_{\mathsf{S}}} \prod_{\mathsf{L} \in T(\mathbf{z})} L(\mathbf{x}_{\mathsf{L}}) = \sum_{\mathcal{T}} \prod_{(\mathsf{S},\mathsf{N}) \in \mathcal{T}} w_{\mathsf{S},\mathsf{N}} \prod_{\mathsf{L} \in \mathcal{T}} \mathsf{L}(\mathbf{x}_{\mathsf{L}}) \underbrace{\left(\sum_{\bar{\mathbf{z}}} \prod_{\mathsf{S} \in \bar{\mathbf{S}}_{\mathcal{T}}} w_{\mathsf{S}, \bar{z}_{\mathsf{S}}} \right)}_{=1}$$

This function assigns to each value z the induced tree (T) determined by z.

It is now conceptually straightforward to extend an SPN to a Bayesian setting, by equipping the sum-weights and leaf-parameters with suitable priors, e.g. Dirichlet priors for the weights.

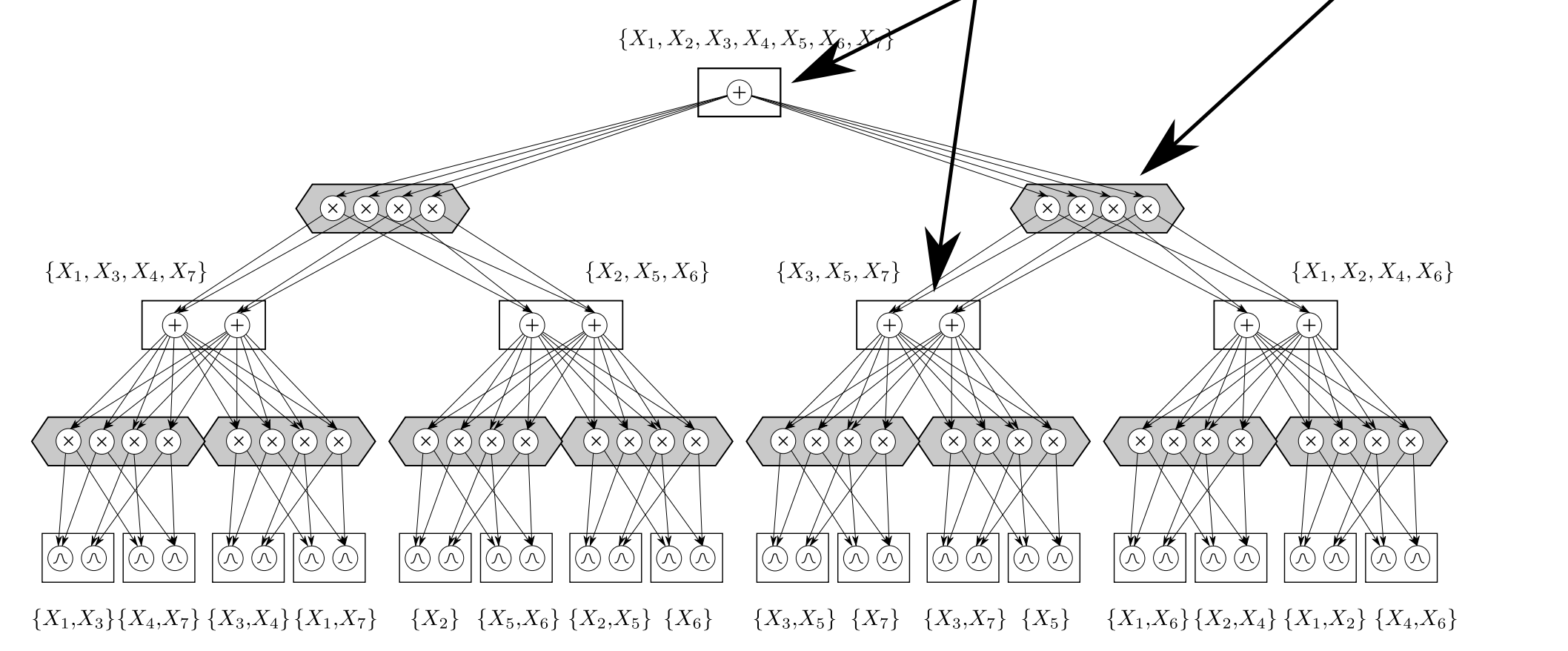
$$\mathbf{w}_{\mathsf{S}} \,|\, \alpha \sim \mathcal{D}ir(\mathbf{w}_{\mathsf{S}} \,|\, \alpha) \;\; \forall \mathsf{S}, \qquad z_{\mathsf{S},n} \,|\, \mathbf{w}_{\mathsf{S}} \sim \mathcal{C}at(z_{\mathsf{S},n} \,|\, \mathbf{w}_{\mathsf{S}}) \;\; \forall \mathsf{S}\, \forall n,$$
$$\theta_{\mathsf{L}} \,|\, \gamma \sim p(\theta_{\mathsf{L}} \,|\, \gamma) \;\; \forall \mathsf{L}, \qquad \mathbf{x}_n \,|\, \mathbf{z}_n, \theta \sim \prod_{\mathsf{L} \in T(\mathbf{z}_n)} \mathsf{L}(\mathbf{x}_{\mathsf{L},n} \,|\, \theta_{\mathsf{L}}) \;\; \forall n.$$

[Poon2011] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In Proceedings of UAI, pages 337–346, 2011.
[Zhao2016] H. Zhao, T. Adel, G. J. Gordon, and B. Amos. Collapsed variational inference for sum-product networks. In Proceedings of ICML, pages 1310–1318, 2016.
[Rashwan2016] A. Rashwan, H. Zhao, and P. Poupart. Online and distributed Bayesian moment matching for parameter learning in SPNs. In Proceedings of AISTATS, pages 1469–1477, 2016.
[Vergari2019] A. Vergari, A. Molina, R. Peharz, Z. Ghahramani, K. Kersting, and I. Valera. Automatic Bayesian density analysis. In Proceedings of AAAI, 2019.

Bayesian Sum-Product Networks

We now introduce join structure and parameter learning by considering the case in which the computational graph is a tree-shaped region graph.

A region graph can be understood as a vectorised representation of SPNs and is a connected directed acyclic graph containing two types of nodes: regions (R) and partitions (P).

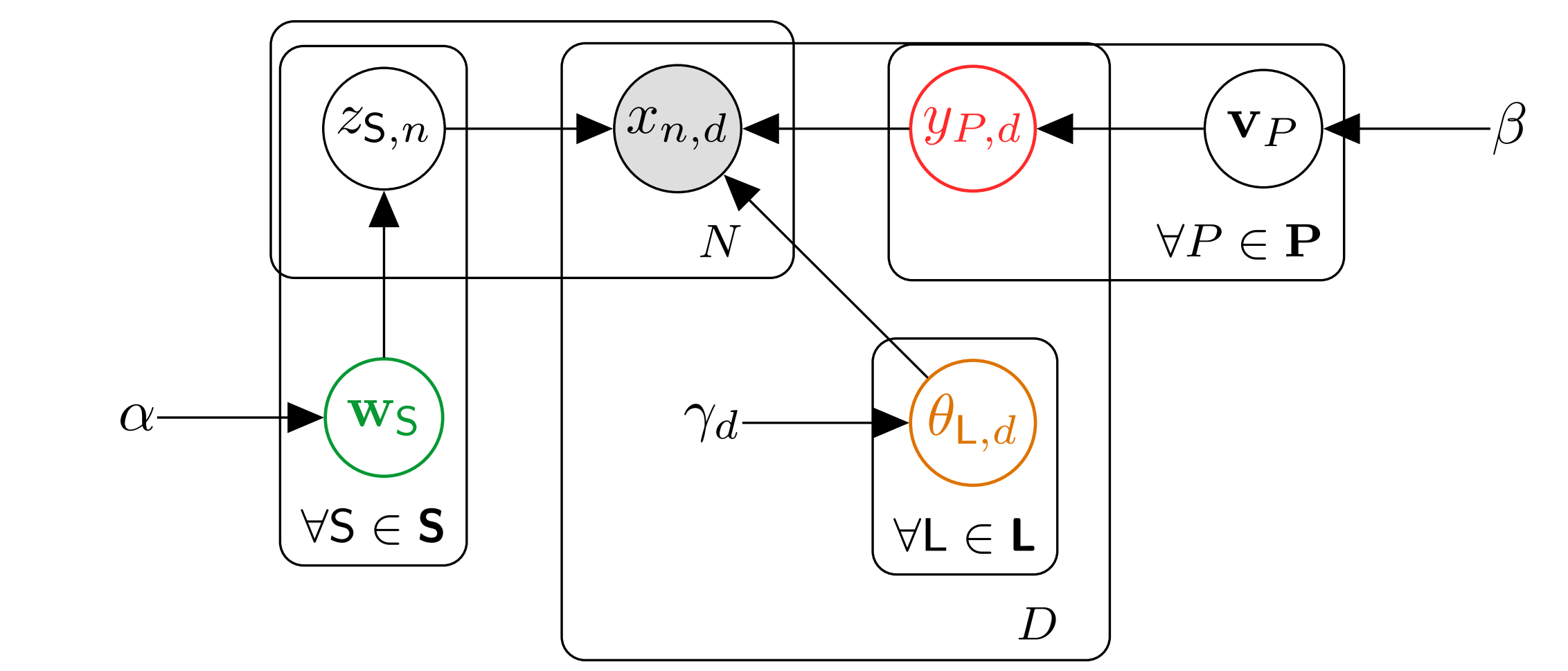

$\{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$
$\{X_1, X_3, X_4, X_7\}$
$\{X_2, X_5, X_6\}$
$\{X_3, X_5, X_7\}$
$\{X_1, X_2, X_4, X_6\}$
$\{X_1,X_3\}$ $\{X_4,X_7\}$ $\{X_3,X_4\}$ $\{X_1,X_7\}$ $\{X_2\}$ $\{X_5,X_6\}$ $\{X_2,X_5\}$ $\{X_6\}$ $\{X_3,X_5\}$ $\{X_7\}$ $\{X_3,X_7\}$ $\{X_5\}$ $\{X_1,X_6\}$ $\{X_2,X_4\}$ $\{X_1,X_2\}$ $\{X_4,X_6\}$
Illustration of a region-graph based on [Peharz2019].

For each data dimension d we introduce a discrete latent variable $Y_{P,d}$. The latent variable represents a decision to assign dimension d to a particular child, given that all partitions above have decided to assign d onto the (unique) path leading to the partition.

$$\mathbf{w}_{\mathsf{S}} \,|\, \alpha \sim \mathcal{D}ir(\mathbf{w}_{\mathsf{S}} \,|\, \alpha) \;\; \forall \mathsf{S}, \qquad z_{\mathsf{S},n} \,|\, \mathbf{w}_{\mathsf{S}} \sim \mathcal{C}at(z_{\mathsf{S},n} \,|\, \mathbf{w}_{\mathsf{S}}) \;\; \forall \mathsf{S}\, \forall n,$$
$$\mathbf{v}_P \,|\, \beta \sim \mathcal{D}ir(\mathbf{v}_P \,|\, \beta) \;\; \forall P, \qquad y_{P,d} \,|\, \mathbf{v}_P \sim \mathcal{C}at(v_{P,d} \,|\, \mathbf{v}_P) \;\; \forall P\, \forall d,$$
$$\theta_{\mathsf{L}} \,|\, \gamma \sim p(\theta_{\mathsf{L}} \,|\, \gamma) \;\; \forall \mathsf{L}, \qquad \mathbf{x}_n \,|\, \mathbf{z}_n, \mathbf{y}, \theta \sim \prod_{\mathsf{L} \in T(\mathbf{z}_n)} \mathsf{L}(\mathbf{x}_{\mathbf{y},n} \,|\, \theta_{\mathsf{L}}) \;\; \forall n.$$

[Peharz2019] R.Peharz, A.Vergari, K.Stelzner, A.Molina, X.Shao, M.Trapp, K.Kersting, and Z.Ghahramani. Random Sum-Product Networks: A simple but effective approach to probabilistic deep learning. In Proceedings of UAI, 2019.

Approximate Bayesian inference


$z_{\mathsf{S},n}$
$x_{n,d}$
$y_{P,d}$
$\mathbf{v}_P$
β
N
$\forall P \in \mathbf{P}$
α
$\mathbf{w}_{\mathsf{S}}$
$\forall \mathsf{S} \in \mathbf{S}$
γ_d
$\theta_{\mathsf{L},d}$
$\forall \mathsf{L} \in \mathbf{L}$
D

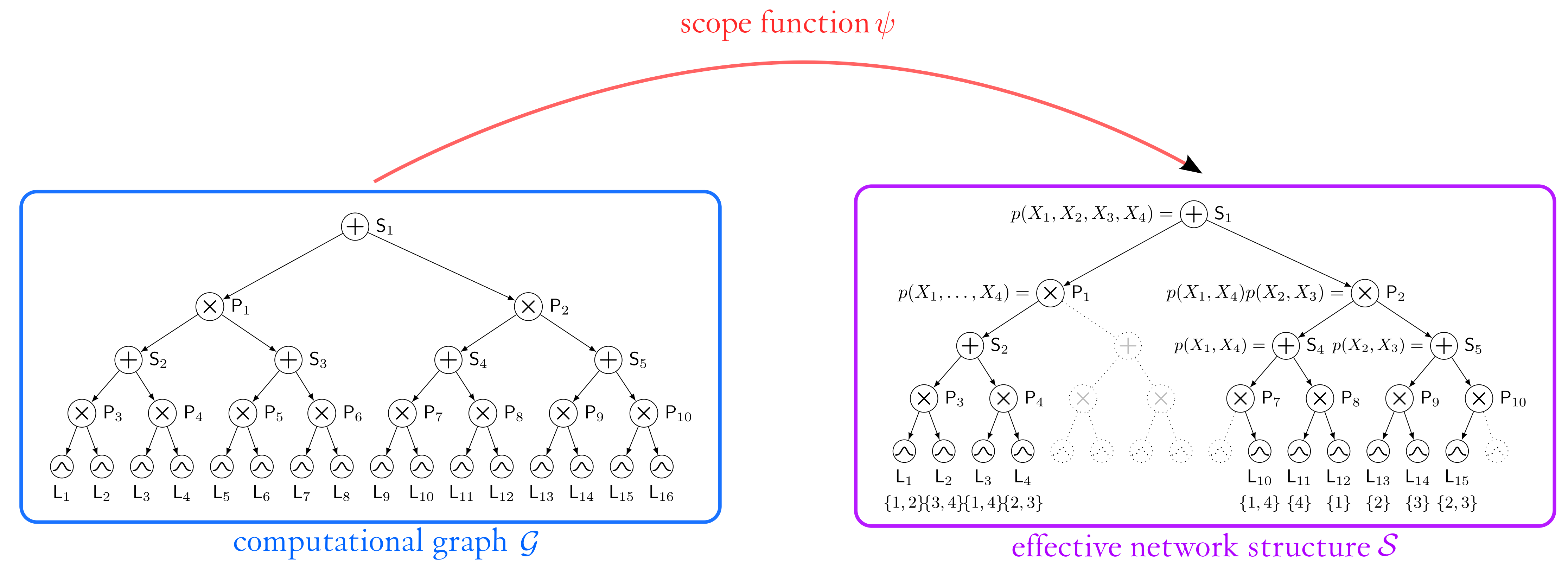
Updating Parameters: $\mathbf{w}_{\mathsf{S}}$ $\theta_{\mathsf{L},d}$ with fixed: $y_{P,d}$

We first sample assignments z_n for all sum latent variables in the SPN, and subsequently sample new weights and leaf parameters. The assignments can be drawn independently using ancestral sampling. All latent variables which are not visited during ancestral sampling, are drawn from the prior. Then we sample the weights and leaf parameters from the respective posterior.

Updating the Structure: $y_{P,d}$ with fixed: $\mathbf{w}_{\mathsf{S}}$ $\theta_{\mathsf{L},d}$

We use a collapsed Gibbs sampler to sample all structure LVs at the partitions. For this purpose, we marginalise out all v_P variables and sample the discrete structure LVs $y_{P,d}$ from their conditional $p(y_{P,d} = k \,|\, \mathbf{y}_{P,\cancel{d}}, \mathbf{y}_{\mathbf{P} \setminus P,d}, \mathcal{X}, \mathbf{z}, \theta, \beta)$

All assignments at partition P excluding the assignment for d · The trainingset


scope function ψ
S_1 P_1 P_2 S_2 S_3 S_4 S_5 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10}
L_1 L_2 L_3 L_4 L_5 L_6 L_7 L_8 L_9 L_{10} L_{11} L_{12} L_{13} L_{14} L_{15} L_{16}
computational graph $\mathcal{G}$
$p(X_1, X_2, X_3, X_4) =$
$p(X_1, \dots, X_4) =$
$p(X_1, X_4)p(X_2, X_3) =$
$p(X_1, X_4) =$
$p(X_2, X_3) =$
$\{1,2\}$ $\{3,4\}$ $\{1,4\}$ $\{2,3\}$
$\{1,4\}$ $\{4\}$ $\{1\}$ $\{2\}$ $\{3\}$ $\{2,3\}$
effective network structure $\mathcal{S}$

A computational graph (left) and a SPN structure (right), defined by the scope function, discovered using posterior inference on an encoding of Ψ.
The SPN contains only a subset of the nodes in the graph as some sub-trees are allocated with an empty scope (dotted) -- evaluating to constant 1.
Note that the computational graph only encodes the topological layout of the nodes, while the effective SPN structure is encoded via Ψ.

Experiments: On discrete, heterogeneous and data with missing values

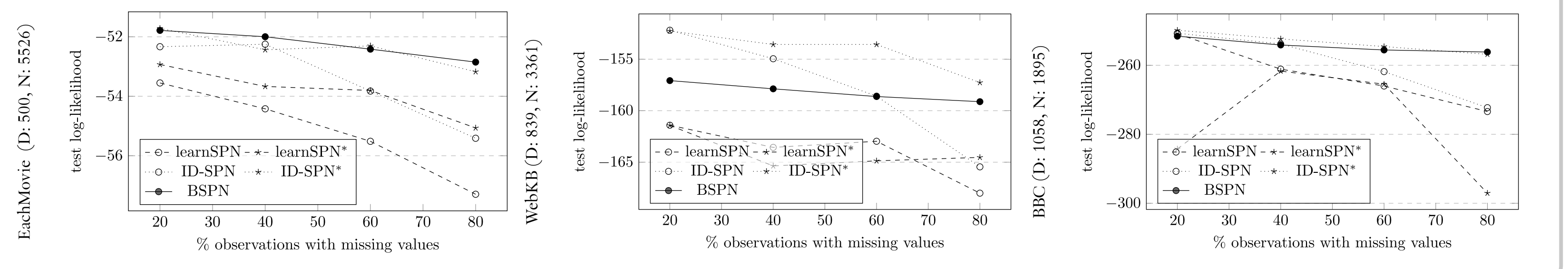
Average test log-likelihoods on discrete datasets using SOTA, Bayesian SPNs (ours) and infinite mixtures of SPNs (ours$^\infty$). Significant differences are underlined. Overall best result is in bold.
In addition we list the best-to-date (BTD) results obtained using SPNs, PSDDs or CNets.
Note that BTD results are often obtain by large ensembles of structures.

Dataset	LearnSPN	RAT-SPN	CCCP	ID-SPN	ours	ours$^\infty$	BTD
NLTCS	−6.11	−6.01	−6.03	−6.02	**−6.00**	−6.02	−5.97
MSNBC	−6.11	−6.04	−6.05	−6.04	−6.06	**−6.03**	−6.03
KDD	−2.18	−2.13	−2.13	−2.13	**−2.12**	−2.13	−2.11
Plants	−12.98	−13.44	−12.87	**−12.54**	−12.68	−12.94	−11.84
Audio	−40.50	−39.96	−40.02	−39.79	**−39.77**	−39.79	−39.39
Jester	−53.48	−52.97	−52.88	−52.86	**−52.42**	−52.86	−51.29
Netflix	−57.33	−56.85	−56.78	−56.36	**−56.31**	−56.80	−55.71
Accidents	−30.04	−35.49	−27.70	**−26.98**	−34.10	−33.89	−26.98
Retail	−11.04	−10.91	−10.92	−10.85	−10.83	**−10.83**	−10.72
Pumsb-star	−24.78	−32.53	−24.23	**−22.41**	−31.34	−31.96	−22.41
DNA	−82.52	−97.23	−84.92	**−81.21**	−92.95	−92.84	−81.07
Kosarak	−10.99	−10.89	−10.88	**−10.60**	−10.74	−10.77	−10.52
MSWeb	−10.25	−10.12	−9.97	**−9.73**	−9.88	−9.89	−9.62
Book	−35.89	−34.68	−35.01	−34.14	**−34.13**	−34.34	−34.14
EachMovie	−52.49	−53.63	−52.56	−51.51	−51.66	**−50.94**	−50.34
WebKB	−158.20	−157.53	−157.49	**−151.84**	−156.02	−157.33	−149.20
Reuters-52	−85.07	−87.37	−84.63	**−83.35**	−84.31	−84.44	−81.87
20 Newsgrp	−155.93	−152.06	−153.21	**−151.47**	−151.99	−151.95	−151.02
BBC	−250.69	−252.14	**−248.60**	−248.93	−249.70	−254.69	−229.21
AD	−19.73	−48.47	−27.20	**−19.05**	−63.80	−63.80	−14.00

Average test log-likelihoods on heterogeneous datasets using SOTA, Bayesian SPN (ours) and infinite mixtures of SPNs (ours∞). Overall best result is indicated in bold.

Dataset	MSPN	ABDA	ours	ours$^\infty$
Abalone	**9.73**	2.22	3.92	3.99
Adult	−44.07	−5.91	**−4.62**	−4.68
Australian	−36.14	**−16.44**	−21.51	−21.99
Autism	−39.20	−27.93	**−0.47**	−1.16
Breast	−28.01	−25.48	**−25.02**	−25.76
Chess	−13.01	−12.30	**−11.54**	−11.76
Crx	−36.26	**−12.82**	−19.38	−19.62
Dermatology	−27.71	−24.98	**−23.95**	−24.33
Diabetes	−31.22	**−17.48**	−21.21	−21.06
German	−26.05	**−25.83**	−26.76	−26.63
Student	−30.18	**−28.73**	−29.51	−29.9
Wine	**−0.13**	−10.12	−8.62	−8.65

Performance under missing values for discrete datasets with increasing dimensionality (D).
Results for LearnSPN are shown in dashed lines, results for ID-SPN in dotted lines and our approach is indicated using solid lines. Star indicates k-NN imputation and a circle no imputation.



Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)