Bayesian Learning of Sum-Product Networks

Martin Trapp[1,2] R. Peharz[3] H. Ge[3]
F. Pernkopf[1] Z. Ghahramani[4,3]

[1]Graz University of Technology, [2]OFAI

[3]University of Cambridge, [4]Uber AI

October 29, 2019

- Sum-product networks (SPNs) [Poon2011] are flexible density estimators for complex distributions.

- Sum-product networks (SPNs) [Poon2011] are flexible density estimators for complex distributions.
- SPNs are so-called tractable probabilistic models, i.e. they allow exact inference in linear time.

- Sum-product networks (SPNs) [Poon2011] are flexible density estimators for complex distributions.
- SPNs are so-called tractable probabilistic models, i.e. they allow exact inference in linear time.
- A critical topic learning SPNs is to learn an appropriate structure from data.
- However, most structure learners are somewhat adhoc and based on intuition rather than a clear learning principle.

- Sum-product networks (SPNs) [Poon2011] are flexible density estimators for complex distributions.
- SPNs are so-called tractable probabilistic models, i.e. they allow exact inference in linear time.
- A critical topic learning SPNs is to learn an appropriate structure from data.
- However, most structure learners are somewhat adhoc and based on intuition rather than a clear learning principle.
- In fact, all existing approaches *do not declare the global goal of structure learning*!

- We introduce a well-principled Bayesian framework for SPN structure learning by decomposing the problem into:
    1. laying out a computational graph, and
    2. learning the so-called scope function over the graph.

- We introduce a well-principled Bayesian framework for SPN structure learning by decomposing the problem into:
  1. laying out a computational graph, and
  2. learning the so-called scope function over the graph.
- The first part is rather unproblematic, and the second part is more involved in full generality.

- We introduce a well-principled Bayesian framework for SPN structure learning by decomposing the problem into:
  1. laying out a computational graph, and
  2. learning the so-called scope function over the graph.
- The first part is rather unproblematic, and the second part is more involved in full generality.
- However, we propose a natural parametrisation of the scope function for a widely used special case of SPNs.

- We introduce a well-principled Bayesian framework for SPN structure learning by decomposing the problem into:
  1. laying out a computational graph, and
  2. learning the so-called scope function over the graph.
- The first part is rather unproblematic, and the second part is more involved in full generality.
- However, we propose a natural parametrisation of the scope function for a widely used special case of SPNs.
- We show how to incorporate the parametrisation into a Bayesian model.

- We introduce a well-principled Bayesian framework for SPN structure learning by decomposing the problem into:
    1. laying out a computational graph, and
    2. learning the so-called scope function over the graph.
- The first part is rather unproblematic, and the second part is more involved in full generality.
- However, we propose a natural parametrisation of the scope function for a widely used special case of SPNs.
- We show how to incorporate the parametrisation into a Bayesian model.
- Finally, show how to perform posterior inference in such a model.

Sum-Product Networks

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables, for which N i.i.d. samples are available.

What is a Sum-Product Network?

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

What is a Sum-Product Network?

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

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables, for which N i.i.d. samples are available.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\mathcal{S} = (\mathcal{G}, \psi, w, \theta)$.
  - $\mathcal{G}$ is a computational graph.
  - ψ is a so-called scope function.

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables, for which N i.i.d. samples are available.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\mathcal{S} = (\mathcal{G}, \psi, w, \theta)$.
  - $\mathcal{G}$ is a computational graph.
  - ψ is a so-called scope function.
  - w denotes the set of sum-weights and θ the set of leaf node parameters.

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables, for which N i.i.d. samples are available.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\mathcal{S} = (\mathcal{G}, \psi, w, \theta)$.
  - $\mathcal{G}$ is a computational graph.
  - ψ is a so-called scope function.
  - w denotes the set of sum-weights and θ the set of leaf node parameters.
- Note: This definition is conceptually different to the classic definition of SPNs as it disentangles the definition of the SPNs structure into a computational graph, which has only few requirements, and a scope function, which ensures completeness and decomposability.

Computational Graph $\mathcal{G}$

- Is a connected directed acyclic graph (DAG), containing three types of nodes: sums (S), products (P) and leaves (L).

Computational Graph $\mathcal{G}$

- Is a connected directed acyclic graph (DAG), containing three types of nodes: sums (S), products (P) and leaves (L).
- Only encodes the topological layout of the nodes, while the effective SPN structure is encoded via the scope function ψ.

Computational Graph $\mathcal{G}$

- Is a connected directed acyclic graph (DAG), containing three types of nodes: sums (S), products (P) and leaves (L).
- Only encodes the topological layout of the nodes, while the effective SPN structure is encoded via the scope function ψ.

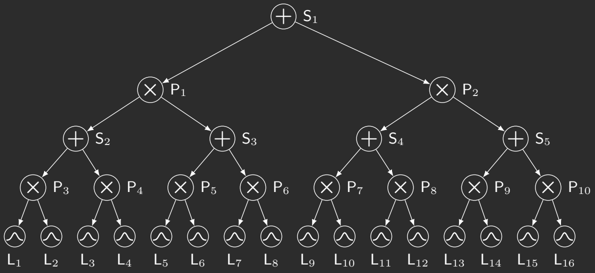


Figure: Example of a tree-shaped computational graph with two layers.

Scope Function ψ

- Let **N** denote the set of all nodes.

Scope Function ψ

- Let **N** denote the set of all nodes.
- ψ a function ψ: **N** $\mapsto 2^{\mathbf{X}}$ assigning each node in the graph a sub-set of X ($2^{\mathbf{X}}$ denotes the power set of $\mathbf{X}$).

Scope Function ψ

- Let **N** denote the set of all nodes.
- ψ a function $\psi\colon \mathbf{N} \mapsto 2^{\mathbf{X}}$ assigning each node in the graph a sub-set of X ($2^{\mathbf{X}}$ denotes the power set of $\mathbf{X}$).

A scope function has to fulfil the following properties:

1. If N is the root node, then $\psi(\mathsf{N}) = \mathbf{X}$.
2. If N is a sum or product, then $\psi(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \psi(\mathsf{N}')$.
3. For each $\mathsf{S} \in \mathbf{S}$ we have
   $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{S})\colon \psi(\mathsf{N}) = \psi(\mathsf{N}')$ (*completeness*).
4. For each $\mathsf{P} \in \mathbf{P}$ we have
   $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{P})\colon \psi(\mathsf{N}) \cap \psi(\mathsf{N}') = \emptyset$ (*decomposability*).

Scope Function ψ

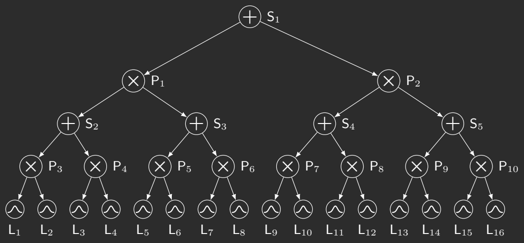
- Let **N** denote the set of all nodes.
- ψ a function $\psi\colon \mathbf{N} \mapsto 2^{\mathbf{X}}$ assigning each node in the graph a sub-set of X ($2^{\mathbf{X}}$ denotes the power set of $\mathbf{X}$).

A scope function has to fulfil the following properties:

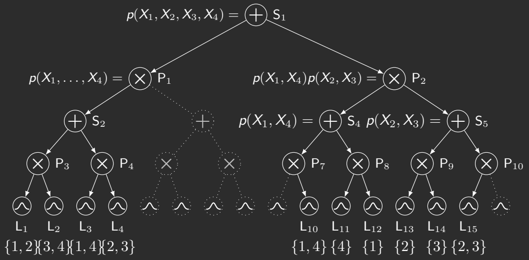
1. If N is the root node, then $\psi(\mathsf{N}) = \mathbf{X}$.
2. If N is a sum or product, then $\psi(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \psi(\mathsf{N}')$.
3. For each $\mathsf{S} \in \mathbf{S}$ we have
   $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{S})\colon \psi(\mathsf{N}) = \psi(\mathsf{N}')$ (*completeness*).
4. For each $\mathsf{P} \in \mathbf{P}$ we have
   $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{P})\colon \psi(\mathsf{N}) \cap \psi(\mathsf{N}') = \emptyset$ (*decomposability*).

Note: Completeness and decomposability are necessary for any SPN to be a well-defined probability distribution and to allow exact inference in linear time (in the model size).

Example SPN $\mathcal{S} = (\mathcal{G}, \psi, w, \theta)$



$\downarrow$ Apply scope function ψ on $\mathcal{G}$



Bayesian Learning of Sum-Product Networks

Bayesian Parameter Learning

- 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.

- 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.

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

Bayesian Parameter Learning

- 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.

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

- $\mathcal{T}$ is a so-called induced tree [Zhao2016] which is a sub-tree in $\mathcal{S}$ such that the root of $\mathcal{S}$ is the root of $\mathcal{T}$, each $\mathsf{S} \in \mathcal{T}$ has only one child and each $\mathsf{P} \in \mathcal{T}$ has the same children as in $\mathcal{S}$.
- $T(\mathbf{z})$ is a surjective (not injective) function that assigns to each value $\mathbf{z}$ the induced tree $\mathcal{T}$ determined by $\mathbf{z}$.

Bayesian Parameter Learning (cont.)

- It is now conceptually straightforward to extend an SPN to a Bayesian setting.

Bayesian Parameter Learning (cont.)

- It is now conceptually straightforward to extend an SPN to a Bayesian setting.

Generative model for Bayesian parameter learning:

$$
\begin{aligned}
w_{\mathsf{S}} \mid \alpha &\sim \mathcal{D}ir(w_{\mathsf{S}} \mid \alpha) \;\; \forall \mathsf{S}\,, & z_{\mathsf{S},n} \,\Big|\, w_{\mathsf{S}} &\sim \mathcal{C}at(z_{\mathsf{S},n} \mid w_{\mathsf{S}}) \;\; \forall \mathsf{S}\, \forall n, \\
\theta_{\mathsf{L}} \mid \gamma &\sim p(\theta_{\mathsf{L}} \mid \gamma) \;\; \forall \mathsf{L}\,, & \mathbf{x}_n \mid \mathbf{z}_n, \theta &\sim \prod_{\mathsf{L} \in T(\mathbf{z}_n)} \mathsf{L}(\mathbf{x}_{\mathsf{L},n} \mid \theta_{\mathsf{L}}) \;\; \forall n.
\end{aligned}
\tag{2}
$$

Bayesian Structure & Parameter Learning

- For Bayesian structure & parameter learning of SPNs we assume $\mathcal{G}$ is a tree-shaped region graph.

Bayesian Structure & Parameter Learning

- For Bayesian structure & parameter learning of SPNs we assume $\mathcal{G}$ is a tree-shaped region graph.
- A region graph $\mathcal{R}$ is a vectorised representation of SPNs and is a connected DAG containing two types of nodes: regions ($R \in \mathbf{R}$) and partitions ($P \in \mathbf{P}$).

Bayesian Structure & Parameter Learning

- For Bayesian structure & parameter learning of SPNs we assume $\mathcal{G}$ is a tree-shaped region graph.
- A region graph $\mathcal{R}$ is a vectorised representation of SPNs and is a connected DAG containing two types of nodes: regions ($R \in \mathbf{R}$) and partitions ($P \in \mathbf{P}$).

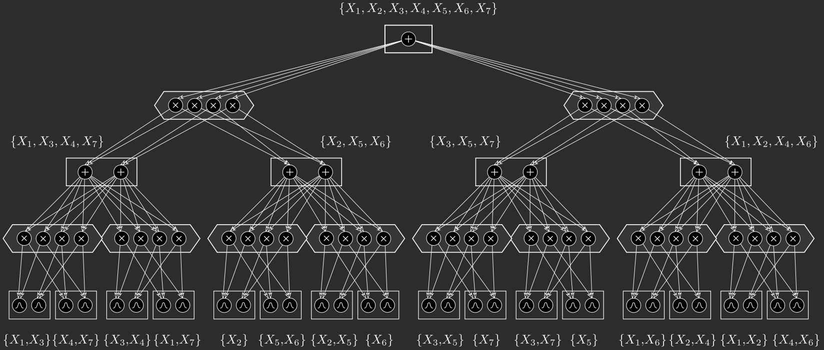


Figure: Example region-graph. Based on the illustration by [Peharz2019].

Bayesian Structure & Parameter Learning (cont.)

- For each data dimension d we introduce a discrete latent variable $Y_{P,d}$.

Bayesian Structure & Parameter Learning (cont.)

- 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 decided to assign d onto the (unique) path leading to the partition.

Bayesian Structure & Parameter Learning (cont.)

- 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 decided to assign d onto the (unique) path leading to the partition.

Generative model for joint Bayesian learning:

$$
\begin{aligned}
w_{\mathsf{S}} \mid \alpha &\sim \mathcal{D}ir(w_{\mathsf{S}} \mid \alpha) \;\; \forall \mathsf{S}, & z_{\mathsf{S},n} \mid w_{\mathsf{S}} &\sim \mathcal{C}at(z_{\mathsf{S},n} \mid w_{\mathsf{S}}) \;\; \forall \mathsf{S}\, \forall n, \\
\mathbf{v}_P \mid \beta &\sim \mathcal{D}ir(\mathbf{v}_P \mid \beta) \;\; \forall P, & y_{P,d} \mid \mathbf{v}_P &\sim \mathcal{C}at(v_{P,d} \mid \mathbf{v}_P) \;\; \forall P \forall d, \\
\theta_{\mathsf{L}} \mid \gamma &\sim p(\theta_{\mathsf{L}} \mid \gamma) \;\; \forall \mathsf{L}, & \mathbf{x}_n \mid \mathbf{z}_n, \mathbf{y}, \theta &\sim \prod_{\mathsf{L} \in T(\mathbf{z}_n)} \mathsf{L}(\mathbf{x}_{\mathbf{y},n} \mid \theta_{\mathsf{L}}) \;\; \forall n.
\end{aligned}
\tag{3}
$$

$\mathbf{x}_{\mathbf{y},n}$ denotes the evaluation of L on the scope induced by $\mathbf{y}$.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters w, θ (fixed $\mathbf{y}$), and ii) updating $\mathbf{y}$ (fixed w, θ) to learn Bayesian SPNs.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters w, θ (fixed $\mathbf{y}$), and ii) updating $\mathbf{y}$ (fixed w, θ) to learn Bayesian SPNs.
- This approach has shown to be sufficient for most real-world dataset.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters w, θ (fixed $\mathbf{y}$), and ii) updating $\mathbf{y}$ (fixed w, θ) to learn Bayesian SPNs.
- This approach has shown to be sufficient for most real-world dataset.
- More sophisticated approaches, e.g. particle Gibbs combined with Hamiltonian Monte Carlo sampling; variational inference or posterior bootstrap, might be a interesting future avenues for large-scale problems.

Experiments

- We performed some experiments agains SOTA structure learners for SPNs on discrete, heterogeneous and data with missing values.

Experiments

- We performed some experiments agains SOTA structure learners for SPNs on discrete, heterogeneous and data with missing values.
- The discrete datasets are standard benchmark datasets for SPNs and contain data with up to 1556 dimensions.
- The heterogeneous datasets are UCI datasets that have been used in previous literature.

Experiments

- We performed some experiments agains SOTA structure learners for SPNs on discrete, heterogeneous and data with missing values.
- The discrete datasets are standard benchmark datasets for SPNs and contain data with up to 1556 dimensions.
- The heterogeneous datasets are UCI datasets that have been used in previous literature.
- We evaluated against an increasing number of observations having 50% dimensions missing completely at random to assess the robustness.
- Note: Existing approach cannot handle missing data during structure learning, so we either removed the samples with missing data or used k-NN imputation.

Experiments (discrete data)

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	$\underline{-12.68}$	$\underline{-12.94}$	-11.84
Audio	-40.50	-39.96	-40.02	-39.79	$\underline{-39.77}$	$\underline{-39.79}$	-39.39
Jester	-53.48	-52.97	-52.88	-52.86	$\underline{-52.42}$	$\underline{-52.86}$	-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	$\underline{-34.10}$	$\underline{-33.89}$	-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	$\underline{-31.34}$	$\underline{-31.96}$	-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	$\underline{-10.74}$	$\underline{-10.77}$	-10.52
MSWeb	-10.25	-10.12	-9.97	-9.73	-9.88	$\underline{-9.89}$	-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	$\underline{-156.02}$	-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

Experiments (heterogeneous data)

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

Experiments (missing data)

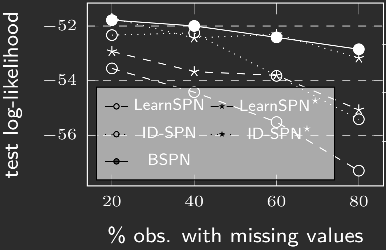


Figure: EachMovie (D: 500, N: 5526)

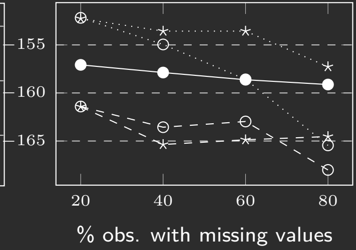


Figure: WebKB (D: 839, N: 3361)

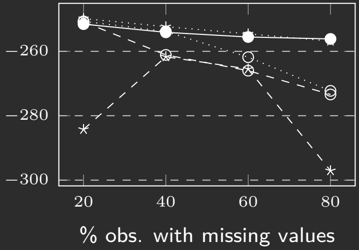


Figure: BBC (D: 1058, N: 1895)

Conclusion

- Structure learning is an important topic in SPNs, and many promising directions have been proposed in recent years.
- However, most of these approaches are based on intuition and refrain from declaring an explicit and global principle to structure learning.

Conclusion

- Structure learning is an important topic in SPNs, and many promising directions have been proposed in recent years.
- However, most of these approaches are based on intuition and refrain from declaring an explicit and global principle to structure learning.
- In this paper, our primary motivation is to *change* this practice

Conclusion

- Structure learning is an important topic in SPNs, and many promising directions have been proposed in recent years.
- However, most of these approaches are based on intuition and refrain from declaring an explicit and global principle to structure learning.
- In this paper, our primary motivation is to *change* this practice
- We phrase structure (and joint parameter) learning as Bayesian inference in a latent variable model.

Conclusion

- Structure learning is an important topic in SPNs, and many promising directions have been proposed in recent years.
- However, most of these approaches are based on intuition and refrain from declaring an explicit and global principle to structure learning.
- In this paper, our primary motivation is to *change* this practice
- We phrase structure (and joint parameter) learning as Bayesian inference in a latent variable model.
- In various experiments we show that this principled approach competes well with prior art and that we gain several benefits, such as *automatic protection against overfitting*, *robustness under missing data* and a natural extension to nonparametric formulations.

Conclusion

- Structure learning is an important topic in SPNs, and many promising directions have been proposed in recent years.
- However, most of these approaches are based on intuition and refrain from declaring an explicit and global principle to structure learning.
- In this paper, our primary motivation is to *change* this practice
- We phrase structure (and joint parameter) learning as Bayesian inference in a latent variable model.
- In various experiments we show that this principled approach competes well with prior art and that we gain several benefits, such as *automatic protection against overfitting*, *robustness under missing data* and a natural extension to nonparametric formulations.
- A critical insight for our approach is to decompose structure learning into: constructing a computational graph and separately learning the SPN's scope function.

References

[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 sum-product networks. 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.

[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.

[Trapp2019] M. Trapp, R. Peharz, H. Ge, F. Pernkopf, and Z. Ghahramani. Bayesian Learning of Sum-Product Networks. To appear at NeurIPS, 2019.