Learning Sum-Product Networks

Martin Trapp

TU Graz

Sum-Product Networks

Probabilistic Inference

	GANs	VAEs	Flows
Sampling	Y	Y	Y
Density	N	N/Y	Y
Marginals	N	N	?
Conditionals	N	N	?
Moments	N	N	?
MAP	N	N	?

Table 1: Robert Peharz, Sum-Product Networks and Deep Learning: A Love Marriage. Talk at ICML, 2019.

Sum-Product Networks

- Sum-product networks (SPNs)[1] are a sub-class of so-called probabilistic circuits[2], that admit tractable probabilistic inference.
- Probabilistic circuits admit many probabilistic inference tasks, such as marginalisation, in linear time in their representation size.

[1] H. Poon & P. Domingos: Sum-product networks: A new deep architecture. In UAI, 2011.
[2] Van den Broeck et al.: Tractable probabilistic models: Representations, algorithms, learning and applications. Tutorial at UAI, 2019.

Probabilistic Inference

	GANs	VAEs	Flows	SPNs
Sampling	Y	Y	Y	Y
Density	N	N/Y	Y	Y
Marginals	N	N	?	Y
Conditionals	N	N	?	Y
Moments	N	N	?	Y
MAP	N	N	?	N/Y

Table 2: Robert Peharz, Sum-Product Networks and Deep Learning: A Love Marriage. Talk at ICML, 2019.

What is a Sum-Product Network?

Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of random variables.

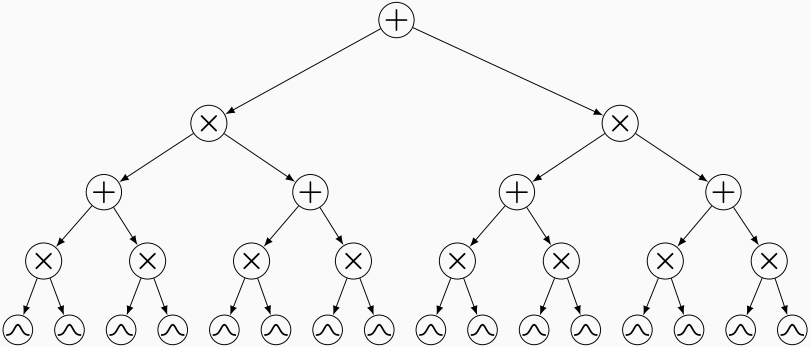
A probabilistic circuit over $\mathbf{X}$ is a tuple $\mathcal{S} = (\mathcal{G}, \psi, \theta)$, where

- $\mathcal{G}$ is a computational graph.
- ψ is a scope function.
- θ is a set of parameters, e.g. sum-weights and leaf node parameters.

A Sum-Product Network (SPN) is a *smooth* (complete) and *decomposable* probabilistic circuit.

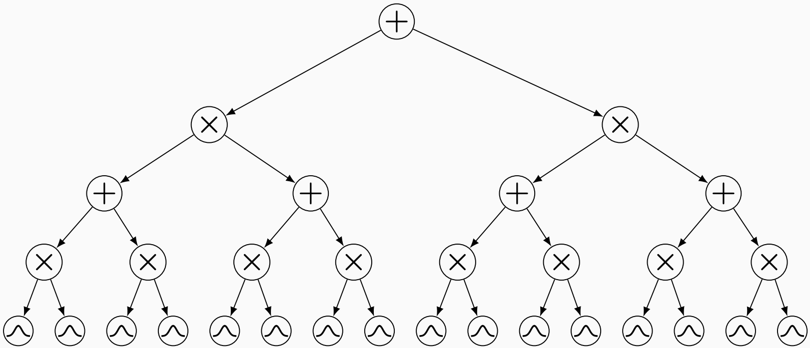
Computational Graph $\mathcal{G}$

$\mathcal{G}$ is a rooted connected directed acyclic graph (DAG), containing: sum (S), product (P) and leaf nodes (L).



Computational Graph $\mathcal{G}$

$\mathcal{G}$ is a rooted connected directed acyclic graph (DAG), containing: sum (S), product (P) and leaf nodes (L).



$\mathrm{S} = \sum w_{\mathrm{S,C}} \mathrm{C}(x)$ $\quad$ $\mathrm{P} = \prod \mathrm{C}(x)$ $\quad$ $\mathrm{L} = p(x \mid \theta_{\mathrm{L}})$

Scope Function ψ

The scope function assigning each node N in a sub-set of **X**,[3] and has to fulfil the following properties:

1. If N is the root node, then $\psi(\mathrm{N}) = \mathbf{X}$.
2. If N is a sum or product, then $\psi(\mathrm{N}) = \bigcup_{\mathrm{N}' \in \mathbf{ch}(\mathrm{N})} \psi(\mathrm{N}')$.

[3] This sub-set is often referred to as the scope of a node.

Scope Function ψ

The scope function assigning each node N in a sub-set of **X**,[3] and has to fulfil the following properties:

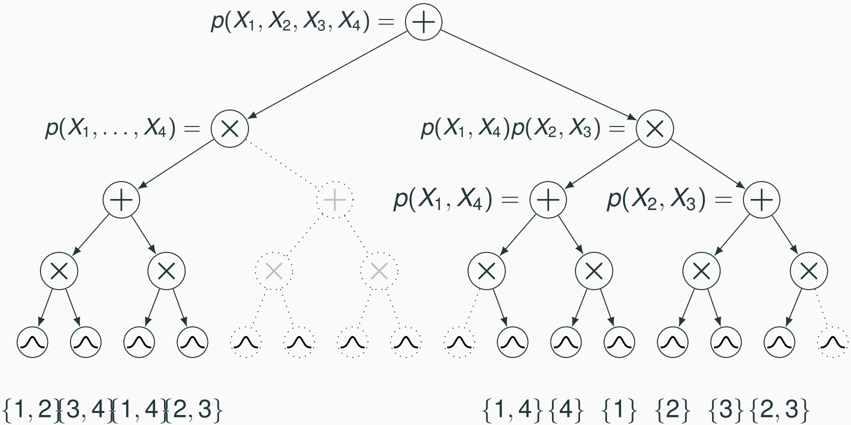
1. If N is the root node, then $\psi(\mathsf{N}) = \mathbf{X}$.
2. If N is a sum or product, then $\psi(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \psi(\mathsf{N}')$.

In case of SPNs we also assume that:

1. For each $\mathsf{S} \in \mathbf{S}$ we have $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{S})\colon \psi(\mathsf{N}) = \psi(\mathsf{N}')$ (*smoothness*)
2. 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*).

[3] This sub-set is often referred to as the scope of a node.

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



Note that we define $\mathrm{L}(x) := 1$ for every x if and only if $\psi(\mathrm{L}) = \emptyset$.

Learning Sum-Product Networks

Parameter Learning in SPNs

We can use backprop for parameter learning in SPNs.

More advanced approaches:

- Expectation Maximisation [R. Peharz et al.: On the latent variable interpretation of sum-product networks. TPAMI, 2017.]
- Variational Inference [H. Zhao et al.: Collapsed variational inference for sum-product networks. In ICML, 2016.]
- Bayesian moment matching [A. Rashwan et al.: Online and distributed Bayesian moment matching for parameter learning in SPNs. In AISTATS, 2016.]
- Safe Semi-Supervised Learning [M. Trapp et al.: Safe semi-supervised learning of sum-product networks. In UAI, 2017.]

Challenges in Structure Learning

- The structure has to be smooth and decomposable, i.e., a sparsely connected graph.
- Structure learning has to be efficient.
- How to learn structures that generalise well, many approaches learn deep trees that are prune to overfitting.
- What is a good SPN structure? or What is a good principle to derive an SPN structure?

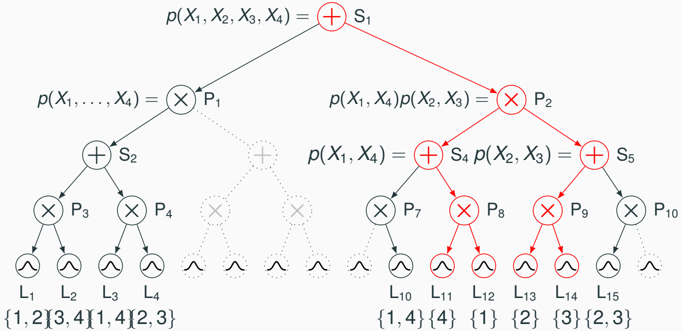
Why do we want Bayesian structure learning?

- Occam's razor effect prevents overfitting.
- Works on discrete, continuous and heterogeneous data domains.
- We can use nonparametric formulations, e.g. infinite SPNs, for continual learning.
- Structures can be inferred even in cases of missing values using exact marginalisation.

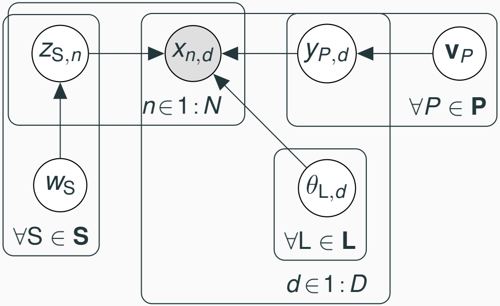
Bayesian Parameter Learning

The key insight for Bayesian parameter learning is that *sum nodes can be interpreted as latent variables*.

The key insight for Bayesian parameter learning is that *sum nodes can be interpreted as latent variables*.

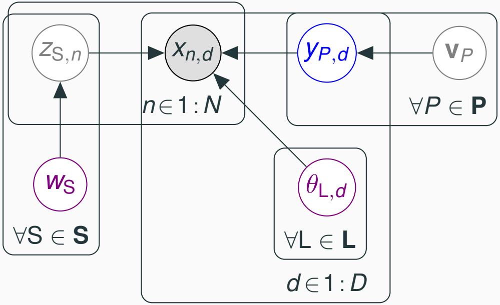


Generative model for Bayesian learning of SPNs.



Bayesian Structure

Posterior inference using ancestral sampling within Gibbs.



Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values.

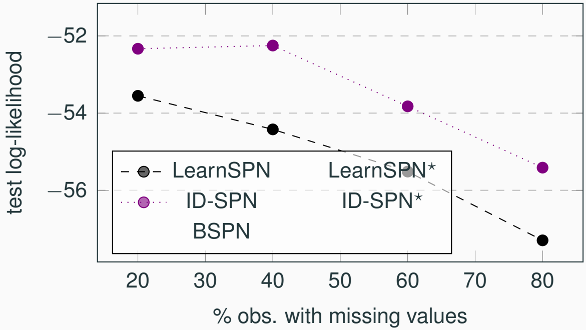


Figure 1: Results on EachMovie (D: 500, N: 5526) dataset.

Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values.

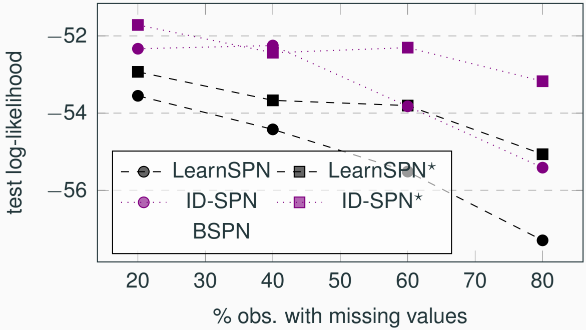


Figure 1: Results on EachMovie (D: 500, N: 5526) dataset.

Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values.

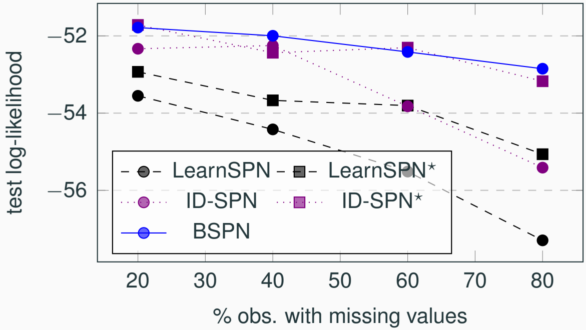


Figure 1: Results on EachMovie (D: 500, N: 5526) dataset.

Applications

Existing Applications

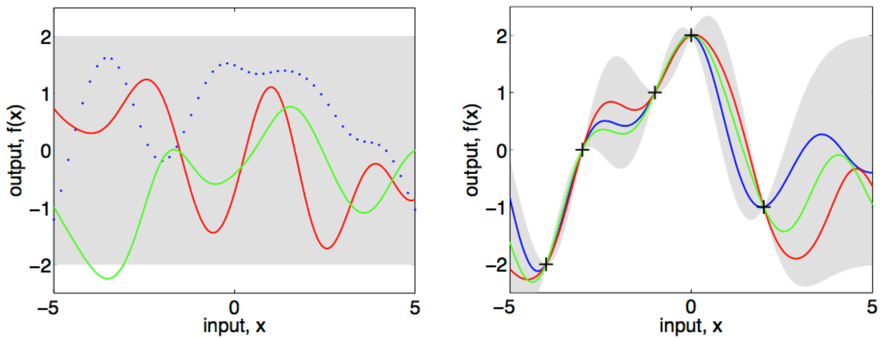
Some existing applications of SPNs:

- Computer vision, e.g. image classification, medical image processing, attend-infer-repeat.
- Language processing, e.g. language modelling, bandwidth extension.
- Robotics, e.g. semantic mapping.
- **Non-linear regression**, and many more[4]

[4] https://github.com/arranger1044/awesome-spn

A Gaussian Process (GP) is a collection of random variables indexed by an arbitrary covariate space $\mathcal{X}$, where any finite subset is Gaussian distributed.

A GP is a prior over functions, that admits exact posterior inference.



(a), prior (b), posterior

Gaussian Processes

A GP is uniquely specified by a *mean-function* $m : \mathcal{X} \to \mathbb{R}$ and a *covariance function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

The posterior predictive distribution (used for predictions) of a GP is Gaussian, i.e.,

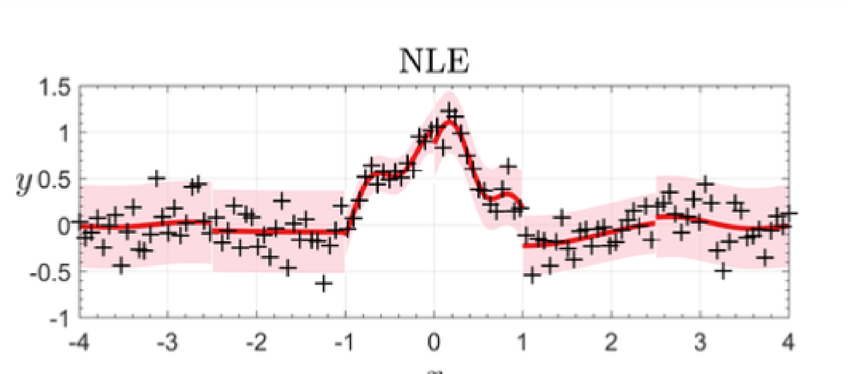
$$p(f^* \mid \boldsymbol{f}) = \mathcal{N}\left(k_{\mathbf{x}^*,\mathbf{X}}^T k_{\mathbf{X},\mathbf{X}}^{-1} \boldsymbol{f},\, k_{\mathbf{x}^*,\mathbf{x}^*} - k_{\mathbf{x}^*,\mathbf{X}} k_{\mathbf{X},\mathbf{X}}^{-1} k_{\mathbf{x}^*,\mathbf{X}}^T\right) \tag{2}$$

The inversion of $k_{\mathbf{X},\mathbf{X}}$ is computed using the Cholesky decomposition of $k_{\mathbf{X},\mathbf{X}}$, which scales $\mathcal{O}(N^3)$.

Local Experts

Local experts to approximate the GP or approximate the computation of predictions.

A natural way is to partition $\mathcal{X}$ into sub-sets $\mathcal{X}^{(k)}$, $k = 1, \ldots, K$. This is called the naive-local-experts model.



Local Experts

Existing solutions to discontinuities.

1. Product-of-Experts (PoE) / Bayesian Committee Machine (BCM)
    - Instead of partition $\mathcal{X}$, partition $\mathcal{X}$ into sub-sets $\mathcal{X}^{(k)}$.
    - Use an algorithm that works only on the sub-sets.
    - **Problem:** Not a stochastic process, results in over-conservative or over-confident estimates.
2. Mixture-of-Experts (MoE)
    - Use a gating network to assign observations to experts instead of hard boundaries.
    - Often intractable (due to the gating network).
3. Impose Continuity Constraints
    - Suffers from inconsistent variances and does not scale.

Deep Structured Mixtures of Gaussian Processes

Why not use a large (finite) mixture of NLEs?

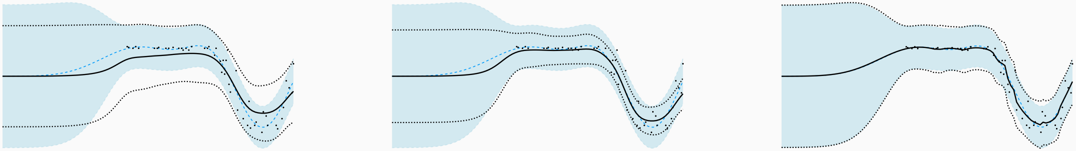
Deep Structured Mixture of GPs (DSMGP)[5]:

1. A DSMGP is a hierarchically defined convex combination of product measures with Gaussian measures as base measure.
2. DSMGPs perform exact Bayesian model averaging over a large set of NLEs.

[5] M. Trapp et al.: Deep structure mixtures of Gaussian Processes. To appear in AISTATS, 2020.

Deep Structured Mixtures of Gaussian Processes

- DSMGPs are a *sound stochastic process*.
- We can perform *exact posterior inference, efficiently*.
- DSMGPs capture predictive uncertainties consistently better than existing approximations.
- In DSMGPs we can model non-stationary data and perform exact inference over kernel functions.



DSMGPs - Example

Thank you for your attention!