

Learning Sum-Product Networks

Martin Trapp

Probabilistic Machine Learning

- Sum-product networks (SPNs) [Poon2011] are general-purpose probabilistic machine learning models that admit tractable exact probabilistic inference.

- Sum-product networks (SPNs) [Poon2011] are general-purpose probabilistic machine learning models that admit tractable exact probabilistic inference.
- What are *tractable probabilistic models*?

- Sum-product networks (SPNs) [Poon2011] are general-purpose probabilistic machine learning models that admit tractable exact probabilistic inference.
- What are *tractable probabilistic models*?
- A class of queries Q on a class of models M is tractable, iff for any query $q \in Q$ and model $m \in M$ the computational complexity is at most polynomial.

- Sum-product networks (SPNs) [Poon2011] are general-purpose probabilistic machine learning models that admit tractable exact probabilistic inference.
- What are *tractable probabilistic models*?
- A class of queries Q on a class of models M is tractable, iff for any query $q \in Q$ and model $m \in M$ the computational complexity is at most polynomial.
- SPNs admit many probabilistic inference tasks, such as marginalisation, in linear time in their representation size.

Sum-Product Networks

- Sum-product networks (SPNs)¹ is a class of general-purpose probabilistic machine learning models that admit tractable probabilistic inference.
- SPNs are a sub-class of so-called tractable probabilistic models or probabilistic circuits.
- A class of queries Q on a class of models M is tractable, iff for any query $q \in Q$ and model $m \in M$ the computational complexity is at most polynomial.
- SPNs admit many probabilistic inference tasks, such as marginalisation, in linear time in their representation size.

¹H. Poon & P. Domingos: Sum-product networks: A new deep architecture. In UAI, 2011.

What is a Sum-Product Network?

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

This definition² is conceptually different to the original definitions as it disentangles the computational graph and the scope function.

²M. Trapp et al.: Bayesian Learning of Sum-Product Networks. In NeurIPS, 2019.

Computational Graph \mathcal{G}

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

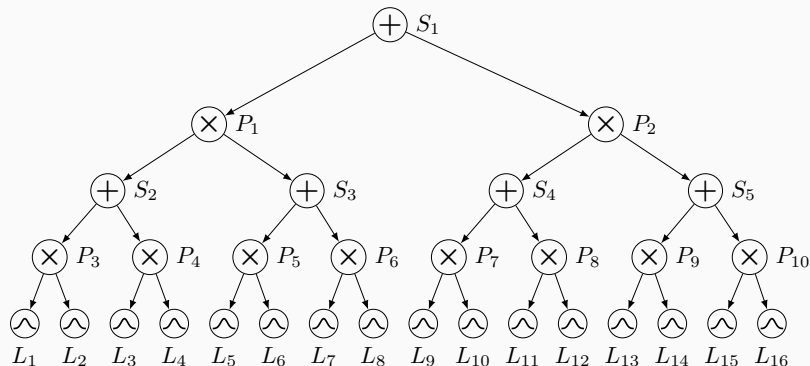


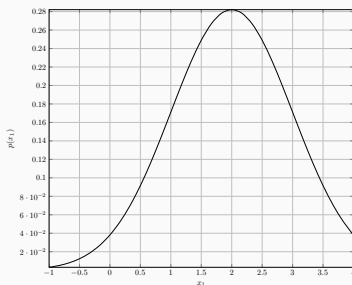
Figure 1: Example of a tree-shaped computational graph.

Leaves L in \mathcal{G}

Leaf nodes are input nodes with arbitrary distribution, e.g. Gaussian, Multinomial, variational autoencoder.



$$L(x) = p(x \mid \theta_L)$$

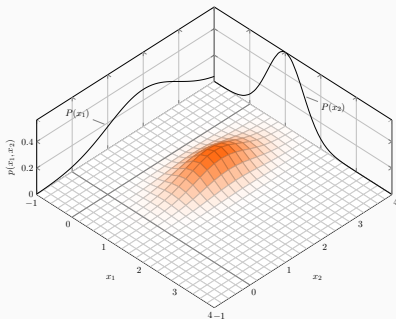


Product Nodes P in \mathcal{G}

Product nodes encode independence assumptions between sets of random variables.



$$P(x) = \prod_{C \in \text{ch}(P)} C(x)$$

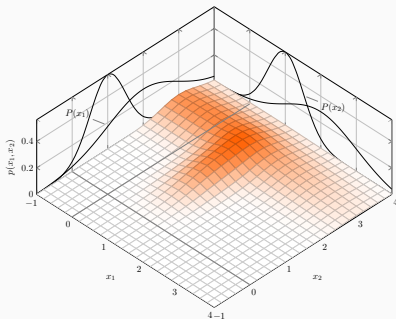


Sum Nodes S in \mathcal{G}

Sum nodes³ replace independence with conditional independence within the network.



$$S(x) = \sum_{C \in \text{ch}(P)} w_{S,C} C(x)$$



³We assume that $w_{S,C} \geq 0$.

Scope Function ψ

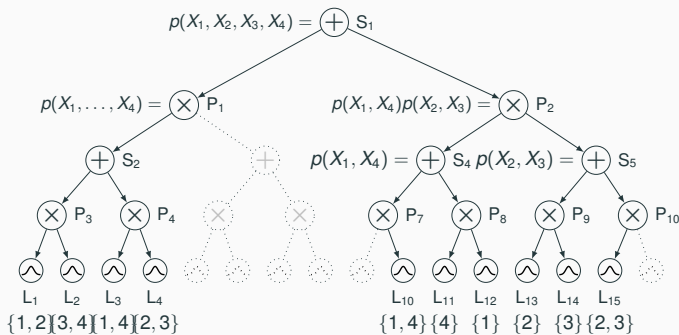
ψ is a function assigning each node N in a sub-set of \mathbf{X} ,⁴ and has to fulfil the following properties:

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

^aComplete and decomposable SPNs are referred to as valid SPNs.

⁴This sub-set is often referred to as the scope of a node.

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



After applying a scope function ψ on \mathcal{G} we obtain the SPN.

Most structure learners learn both in an entangled way.

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

Parameter Learning

Parameter Learning in SPNs

Generative Learning^a

$$\mathcal{L}(\theta \mid \mathcal{X}) = \sum_{n=1}^N \log \mathcal{S}(\mathbf{x}_n \mid \phi) - \log \mathcal{S}(* \mid \phi), \quad \mathbf{x}_n \in \mathbb{R}^D \quad (1)$$

Note that $\mathcal{S}(* \mid \phi)$ is the partition function which can be evaluated efficiently using a single upward pass.

Discriminative Learning^b

$$\mathcal{L}(\theta, \lambda \mid \mathcal{X}) = \sum_{n=1}^N \log \mathcal{S}(\mathbf{x}_n, \lambda_n \mid \phi) - \log \mathcal{S}(\mathbf{x}_n \mid \phi), \quad \mathbf{x}_n \in \mathbb{R}^D, \lambda_n \in \mathbb{R} \quad (2)$$

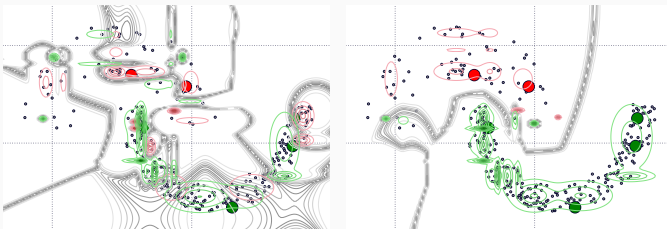
^aH. Poon & P. Domingos: Sum-product networks: A new deep architecture. In UAI, 2011.

^bR. Gens & P. Domingos: Discriminative learning of sum-product networks. In NeurIPS, 2012.

Parameter Learning in SPNs

Semi-Supervised Learning using Contrastive Pessimistic Likelihood Estimation (CPLE)⁵

$$\text{CPLE} = \arg\max_{\theta \in \Theta} \arg \min_{\mathbf{q} \in \Delta_{K-1}^M} \mathcal{L}(\theta, \lambda, \mathbf{q} \mid \mathcal{X}, \mathcal{U}) - \mathcal{L}(\theta^+, \lambda, \mathbf{q} \mid \mathcal{X}, \mathcal{U}) \quad (3)$$



⁵M. Trapp et al.: Safe semi-supervised learning of sum-product networks. In UAI, 2017.

Overparameterization in SPNs⁶

$$w_k^{(t)} \approx w_k^{(t)} + \rho^{(t)} \nabla_{w_k^{(t)}} + \left[\sum_{l=0}^{L-1} \eta \nabla_{w_{\phi(k,l)}^{[l]}} (w_{\phi(k,l)}^{[l]})^{-1} \right] w_k^{(t)} \quad (4)$$

$$= w_k^{(t)} + \rho^{(t)} \nabla_{w_k^{(t)}} + \sum_{\tau=1}^{t-1} \mu^{(t,\tau)} \nabla_{w_k^{(\tau)}} \quad (5)$$

Gradient-based optimisation in deep tree-structured sum-product network with small (fixed) learning rate and near zero initialisation of the weights is equivalent to gradient-based optimisation with adaptive and time-varying **learning rate** and **momentum term**.

⁶M. Trapp et al.: Optimisation of Overparametrized Sum-Product Networks. ICML Workshop on Tractable Probabilistic Models, 2019.

Structure Learning

Challenges in Structure Learning

- The generated structure has to be complete and decomposable, i.e., a sparsely connected graph.
- We are interested in structures that generalise well, many approaches learn deep trees that are prone to overfit.
- Until recently⁷, there has been no clear defined goal or principle of what makes a good structure.

⁷M. Trapp et al.: Bayesian learning of sum-product networks. In NeurIPS, 2019.

General-Purpose Learners (Selection)

- LearnSPN⁸ recursively constructs sum nodes using clustering and product nodes using independence test. The resulting SPN is a tree.
- ID-SPN⁹ is a generalisation of LearnSPN with tractable Markov networks as leaves.
- RAT-SPN¹⁰ constructs region-graphs (meta-graph over SPNs) with random decompositions.
- BSPN⁸ learns structures and parameters using Bayesian inference.

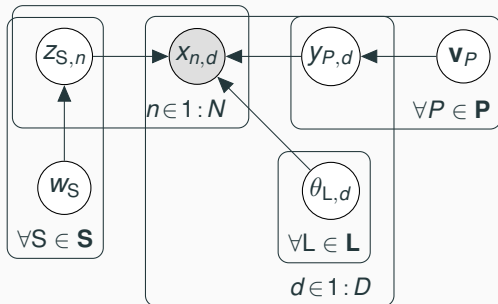
⁸R. Gens & P. Domingos: Learning the structure of sum-product networks. In ICML, 2013.

⁹A. Rooshenas & D. Lowd: Learning Sum-Product Networks with Direct and Indirect Variable Interactions. In ICML, 2014.

¹⁰R. Peharz et al.: Random sum-product networks: A simple but effective approach to probabilistic deep learning. In UAI, 2019.

Bayesian Structure & Parameter Learning

- We assume \mathcal{G} is a tree-shaped region graph, i.e., the SPN is a *not* a tree.
- For each dimension d we introduce a latent variable $Y_{P,d}$ at each partition node (bucket of product nodes).
- The latent variables represent an assign of d to a child, given a unique path leading to the node.



Bayesian Structure & Parameter Learning

- Posterior inference can be performed using ancestral within Gibbs sampling.

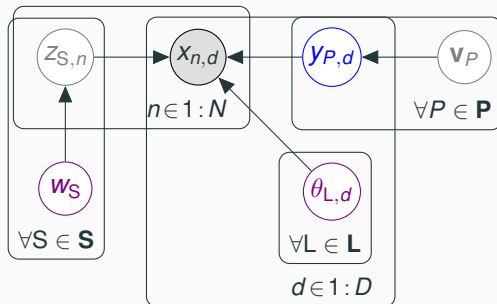
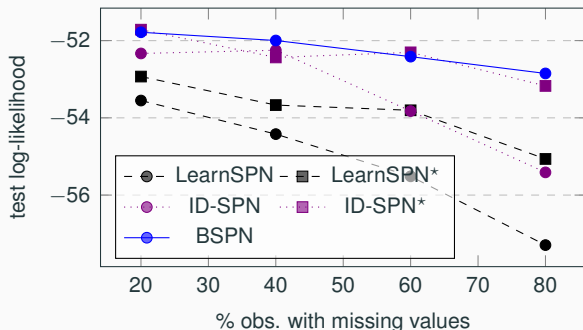


Figure 3: Generative model for Bayesian structure and parameter learning.

Bayesian Structure & Parameter Learning

- This approach obtains competitive results.
- It can be used in heterogeneous data domains and can be extended to nonparametric formulations.
- And this approach can be used in cases of missing values, using exact marginalisation.



Applications

There exist several applications of SPNs in various fields:

- 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.
- Nonlinear regression, and many more¹¹

¹¹<https://github.com/arranger1044/awesome-spn#applications>

We have recently¹² introduced a combination of SPNs with Gaussian processes, which yields an interesting nonparametric regression model that allows exact and efficient posterior inference.

Our model can be understood as an exponentially large mixture over naive-local-experts of Gaussian processes.

¹²M. Trapp et al.: Deep structured mixtures of Gaussian processes. To appear at AISTATS, 2020.

Deep Structured Mixtures of GPs

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

A GP can be understood as a prior over functions.

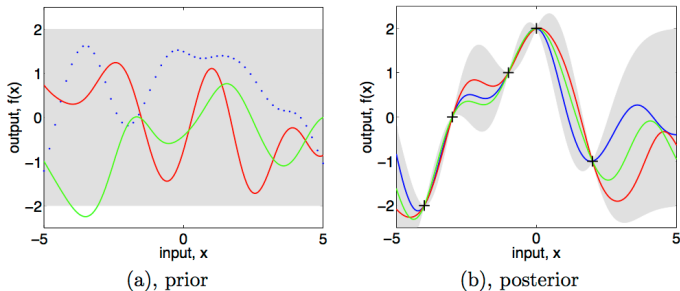


Figure 4: C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, 2006.