# Learning Sum-Product Networks

Martin Trapp

TU Graz

# 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.

[1] H. 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 $\mathcal{S} = (\mathcal{G}, \psi, \boldsymbol{w}, \theta)$.
  - $\mathcal{G}$ is a computational graph.
  - $\psi$ is a so-called scope function.
  - $\boldsymbol{w}$ denotes the set of sum-weights and $\theta$ the set of leaf node parameters.

This definition[2] is conceptually different to the original definitions as it disentangles the computational graph and the scope function.

[2] M. 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).

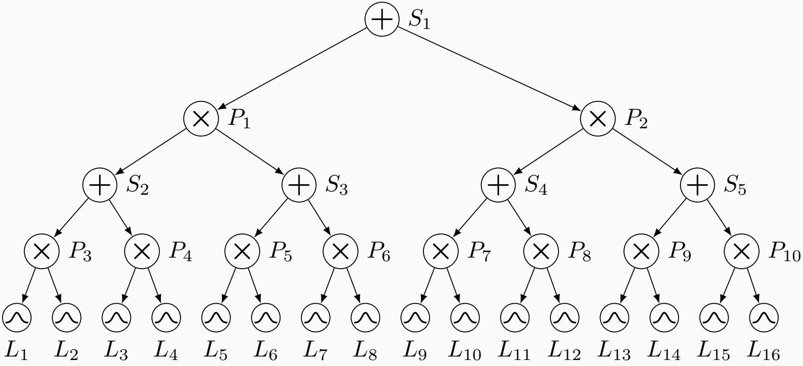
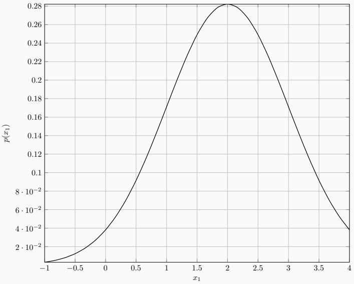



**Figure 1:** Example of a tree-shaped computational graph.

# Leaves L in $\mathcal{G}$

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

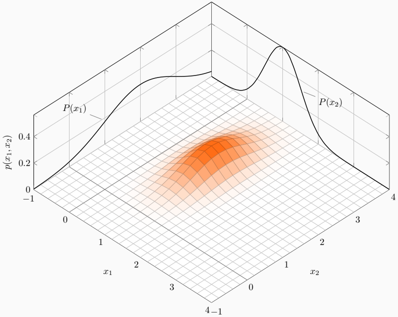
$$\mathsf{L}(x) = p(x \mid \theta_{\mathsf{L}})$$



# Product Nodes P in $\mathcal{G}$

Product nodes encode independence assumptions between sets of random variables.

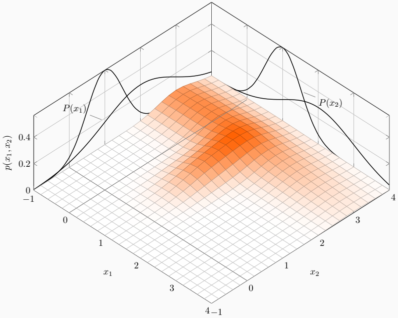
$$\mathsf{P}(x) = \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}(x)$$



# Sum Nodes S in $\mathcal{G}$

Sum nodes[3] replace independence with conditional independence within the network.

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




[3]We assume that $w_{\mathrm{S},\mathrm{C}} \geq 0$.

## Scope Function $\psi$

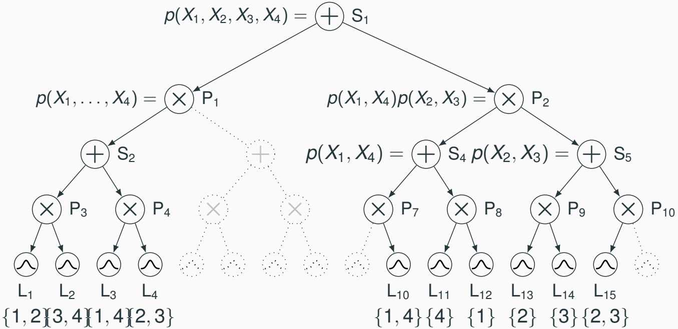
$\psi$ is a function assigning each node N in a sub-set of $\mathbf{X}$,[4] and has to fulfil the following properties:

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

[a] Complete and decomposable SPNs are referred to as valid SPNs.

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

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




After applying a scope function $\psi$ on $\mathcal{G}$ we obtain the SPN.
Most structure learners learn both in an entangled way.

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

# Parameter Learning

# Parameter Learning in SPNs

**Generative Learning**[a]

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

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

**Discriminative Learning**[b]

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

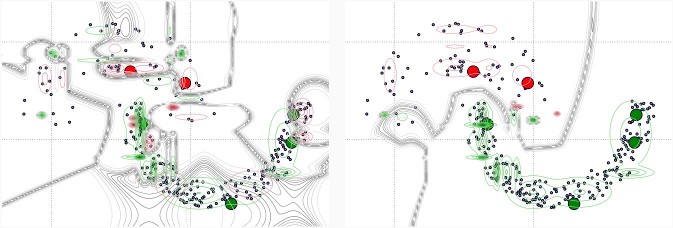
[a] H. Poon & P. Domingos: Sum-product networks: A new deep architecture. In UAI, 2011.

[b] R. 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]

$$\text{CPLE} = \underset{\theta \in \Theta}{\text{argmax}} \, \underset{\boldsymbol{q} \in \Delta_{K-1}^{M}}{\text{arg min}} \; \mathcal{L}(\theta, \lambda, \boldsymbol{q} \mid \mathcal{X}, \mathcal{U}) - \mathcal{L}(\theta^{+}, \lambda, \boldsymbol{q} \mid \mathcal{X}, \mathcal{U}) \tag{3}$$



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

# Overparameterization in SPNs[6]

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

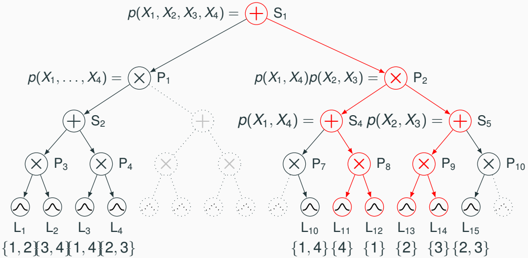
$$= w_k^{(t)} + \rho^{(t)} \nabla_{w_k^{(t)}} + \sum_{\tau=1}^{t-1} \mu^{(t,\tau)} \nabla_{w_k^{(\tau)}} \tag{5}$$

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.

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

# Bayesian Parameter Learning

- The key insight for Bayesian parameter learning[7] is that *sum nodes* can be interpreted as *latent variables* $Z_S$, clustering data instances.
- Given a vector of states for each sum, $\mathbf{z}$ induces a so-called induced tree ($\mathcal{T}$) on $\mathcal{S}$.




[7] Zhao et al.: Collapsed variational inference for sum-product networks. In ICML, 2016.

# 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 prune to overfit.
- Until recently[8], there has been no clear defined goal or principle of what makes a good structure.

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

# General-Purpose Learners (Selection)

- LearnSPN[9] recursively constructs sum nodes using clustering and product nodes using independence test. The resulting SPN is a tree.
- ID-SPN[10] is a generalisation of LearnSPN with tractable Markov networks as leaves.
- RAT-SPN[11] constructs region-graphs (meta-graph over SPNs) with random decompositions.
- BSPN[8] learns structures and parameters using Bayesian inference.

---

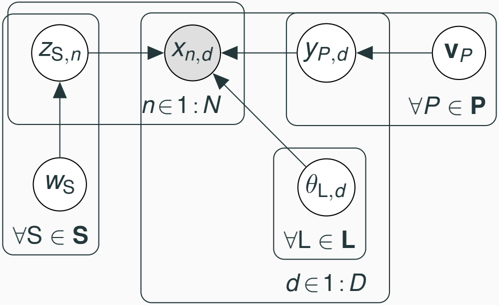
[9] R. Gens & P. Domingos: Learning the structure of sum-product networks. In ICML, 2013.

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

[11] R. 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.
- Bayesian structure learning obtains competitive results on benchmark datasets.
- We show that Bayesian SPNs can also be used in heterogeneous data domains and can be extended to nonparametric formulations, allowing principled online learning.
- Further, our approach is the only method that can consistently learn under missing data.

# Applications