Learning Sum-Product Networks

Martin Trapp

Probabilistic Machine Learning

Introduction

- Machine learning: How can we make machines that learn from data?
- Probabilistic machine learning: How can we make machines that learn from data using tools from probability theory?
- Uncertainty is key to probabilistic machine learning and is expressed in all forms, e.g. noise in the data, uncertainty over the predictions, uncertainty over the model.

Example

Example: How likely is a traffic jam on highway X_1 today?

- today = Thursday = $Y = 4$.
- Traffic jam model: θ for all $\{X_i\}_{i=1}^{n}$ highways in the country.
- $p(X_1 = 1, Y = 4 \mid \theta) = \int_{X_2} \int_{X_3} \cdots \int_{X_n} p(X_1 = 1, Y = 4, X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n \mid \theta) dx_2, dx_3, \ldots dx_n$
- **We need to be able to marginalise out $X_2, \ldots, X_n$ to answer the query.**

Probabilistic Inference

Problem: Probabilistic inference task, such as marginalisation, are often intractable for interesting models.

	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

- 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.

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?

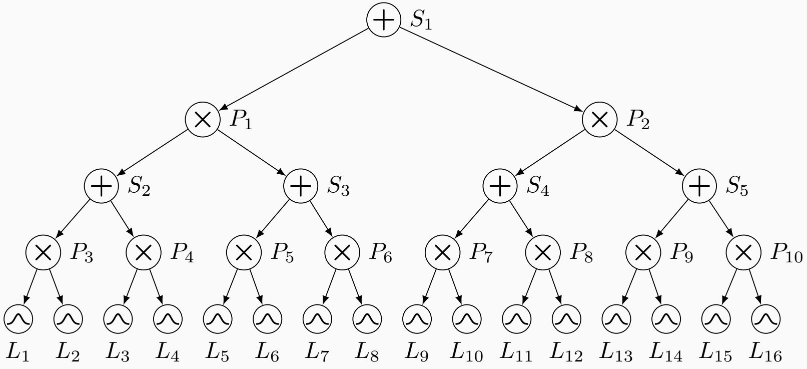
- Let $\mathbf{X} = \{X_1, \dots, X_D\}$ be set of D random variables.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\mathcal{S} = (\mathcal{G}, \psi, \boldsymbol{w}, \theta)$.
    - $\mathcal{G}$ is a computational graph.
    - ψ is a so-called scope function.
    - $\boldsymbol{w}$ denotes the set of sum-weights and θ 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).



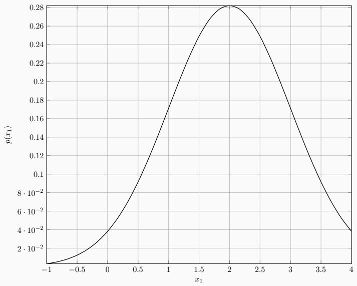
S_1
P_1
P_2
S_2
S_3
S_4
S_5
P_3
P_4
P_5
P_6
P_7
P_8
P_9
P_{10}
L_1 L_2 L_3 L_4 L_5 L_6 L_7 L_8 L_9 L_{10} L_{11} L_{12} L_{13} L_{14} L_{15} L_{16}

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

Leaves L in $\mathcal{G}$

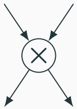
Leaf nodes are input nodes with arbitrary distribution, e.g. Gaussian, Multinomial, variational autoencoder.

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

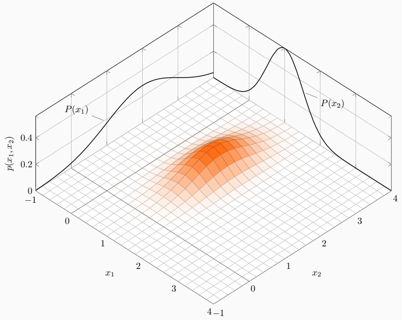


Product Nodes P in $\mathcal{G}$

Product nodes encode independence assumptions between sets of random variables.

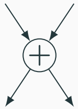


$$\mathsf{P}(x) = \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}(x)$$

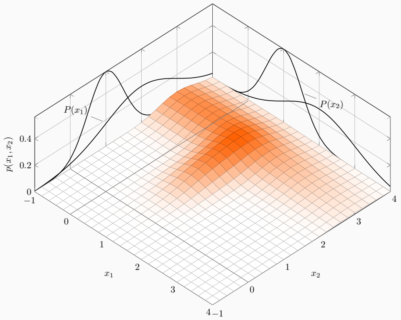


Sum Nodes S in $\mathcal{G}$

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



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



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

Scope Function ψ

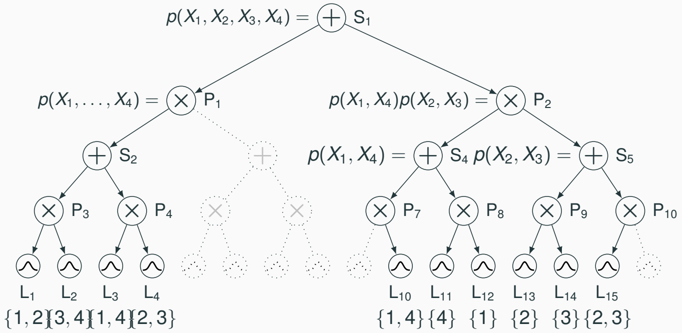
ψ is a function assigning each node N in a sub-set of **X**,[4] and has to fulfil the following properties:

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 ψ 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]

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

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

Discriminative Learning[b]

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

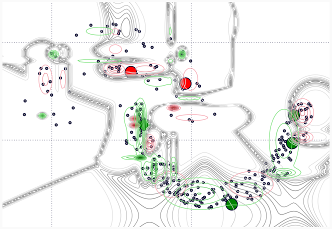
[a] H. Poon & P. Domingos: Sum-product networks: A new deep architecture. In UAI, 2011.

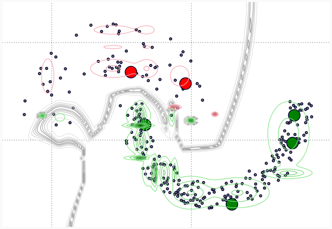
[b] R. Gens & P. Domingos: Discriminative learning of sum-product networks. In NeurIPS, 2012.

Parameter Learning in SPNs

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

$$\mathrm{CPLE} = \operatorname*{argmax}_{\theta \in \Theta} \operatorname*{arg\,min}_{\boldsymbol{q} \in \Delta^{M}_{K-1}} \mathcal{L}(\theta, \lambda, \boldsymbol{q} \mid \mathcal{X}, \mathcal{U}) - \mathcal{L}(\theta^{+}, \lambda, \boldsymbol{q} \mid \mathcal{X}, \mathcal{U}) \tag{3}$$





[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.

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[7], there has been no clear defined goal or principle of what makes a good structure.

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

General-Purpose Learners (Selection)

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

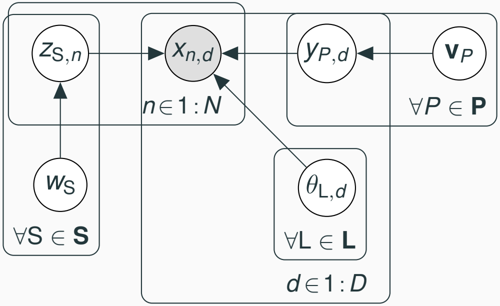
[8] R. Gens & P. Domingos: Learning the structure of sum-product networks. In ICML, 2013.

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

[10] 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.

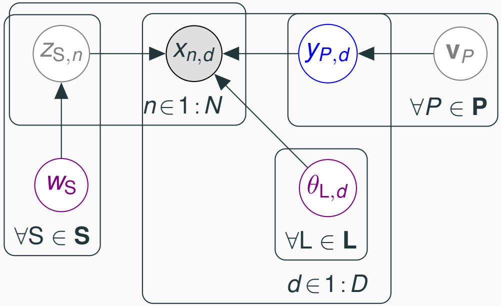
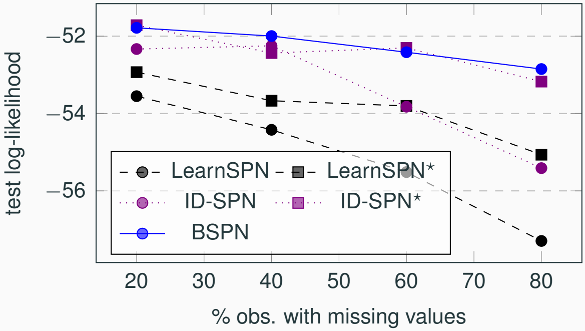


Figure 3: Generative model for Bayesian structure and parameter learning.

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

Existing 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 more[11]

[11] https://github.com/arranger1044/awesome-spn#applications

SPNs for Regression

We have recently[12] 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.

[12] M. 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.

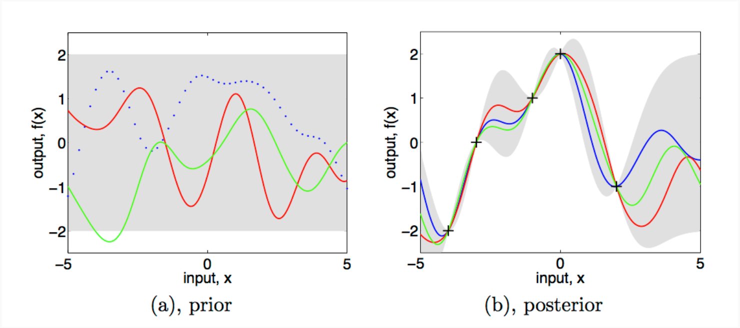


Figure 4: C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, 2006.