Learning Sum-Product Networks

Martin Trapp

Probabilistic Machine Learning

Introduction

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.

Probabilistic Inference

Problem: Probabilistic inference task 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] are a sub-class of so-called tractable probabilistic models or probabilistic circuits[2], that admit tractable probabilistic inference.
- 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.

[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.

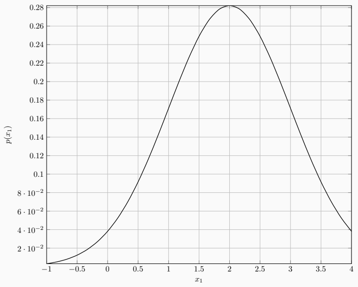
What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of random variables.
- A Probabilistic Circuit (PC) 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* and *decomposable* PC.

Leaves L in $\mathcal{G}$

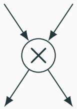
Leaf/input nodes are arbitrary (tractable) distributions, e.g. Gaussian, Multinomial, variational autoencoder.

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

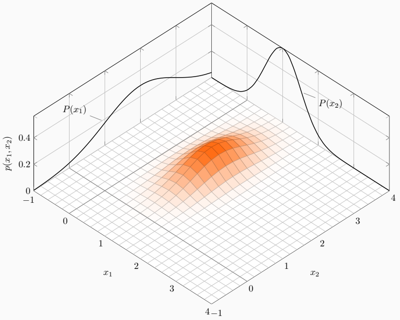


Product Nodes P in $\mathcal{G}$

Product nodes encode independence assumptions between sets of random variables.

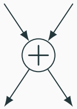


$$\mathsf{P}(x) = \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}(x)$$

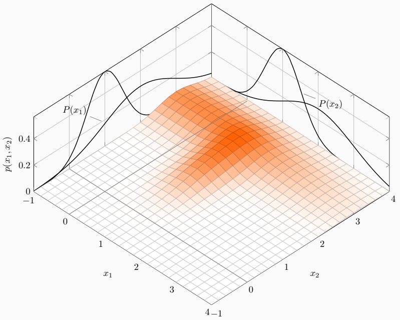


Sum Nodes S in $\mathcal{G}$

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] $\sum_{C \in \mathbf{ch}(S)} w_{S,C} = 1$ and $w_{S,C} \geq 0$.

Computational Graph $\mathcal{G}$

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

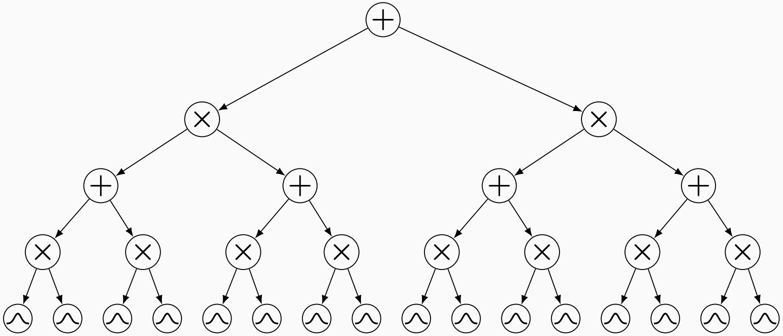


Figure 1: Example of a tree-shaped computational graph.

Scope Function ψ

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

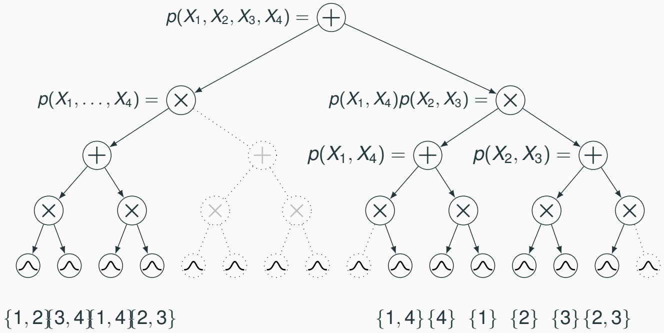
1. If N is the root node, then $\psi(\mathsf{N}) = \mathbf{X}$.
2. If N is a sum or product, then $\psi(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \psi(\mathsf{N}')$.

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*).

[4] 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 $\psi(\mathrm{L}) = \emptyset$.

Parameter Learning

Parameter Learning in SPNs

SPNs are differentiable multi-linear NNs with non-linear inputs, thus common parameter learning for NNs can be applied.

Approaches that go beyond NN techniques:

- 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 sum-product networks. In AISTATS, 2016.]
- **Safe Semi-Supervised Learning** [M. Trapp et al.: Safe semi-supervised learning of sum-product networks. In UAI, 2017.]

Safe Semi-Supervised Learning

Contrastive Pessimistic Likelihood Estimation (CPLE)

- Assume $\mathcal{X} = \{\boldsymbol{x}_n, \lambda_n\}_{n=1}^{N}$, $\lambda_n \in \mathcal{R}^K$ to be a small set of labelled data points and $\mathcal{U} = \{\boldsymbol{u}_m\}_{m=1}^{M}$ a set of unlabelled data points.
- θ^+ is an SPN trained solely on $\mathcal{X}$
- We propose soft labels $\boldsymbol{q}_m \in \Delta_{K-1}$ for all unlabelled data points, e.g. randomly.
- We initialise the semi-supervised SPN θ randomly, then:

Safe Semi-Supervised Learning

Contrastive Pessimistic Likelihood Estimation (CPLE)

- Assume $\mathcal{X} = \{\boldsymbol{x}_n, \lambda_n\}_{n=1}^{N}$, $\lambda_n \in \mathcal{R}^K$ to be a small set of labelled data points and $\mathcal{U} = \{\boldsymbol{u}_m\}_{m=1}^{M}$ a set of unlabelled data points.
- θ^+ is an SPN trained solely on $\mathcal{X}$
- We propose soft labels $\boldsymbol{q}_m \in \Delta_{K-1}$ for all unlabelled data points, e.g. randomly.
- We initialise the semi-supervised SPN θ randomly, then:
  1. Update all soft labels so that the performance of θ^+ on all data points is better than θ.
  2. Update θ so that the performance of θ on all data points is better than θ^+.
  3. If not converged, move to step 1.

Semi-Supervised Learning

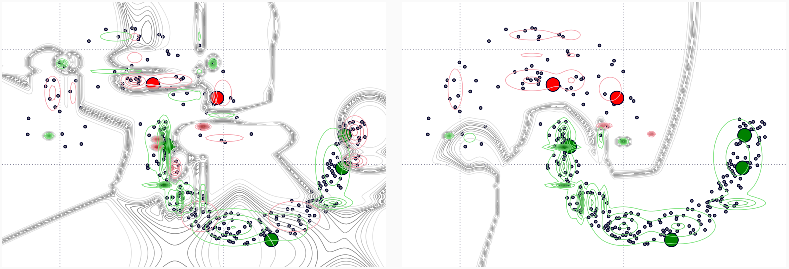
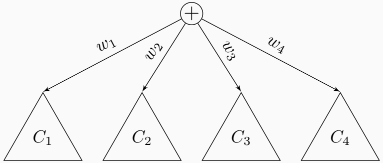
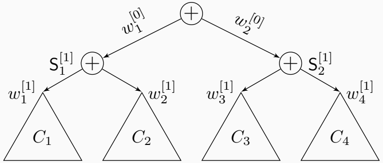


Figure 2: Decision boundary of a semi-supervised SPN at iteration 1 (left) and after convergence (right).

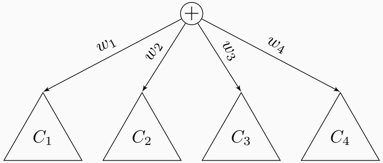
Over-parametrization in SPNs[5]

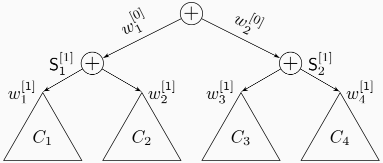




[5] M. Trapp et al.: Optimisation of Overparametrized Sum-Product Networks. ICML Workshop on Tractable Probabilistic Models, 2019.

Over-parametrization in SPNs[5]

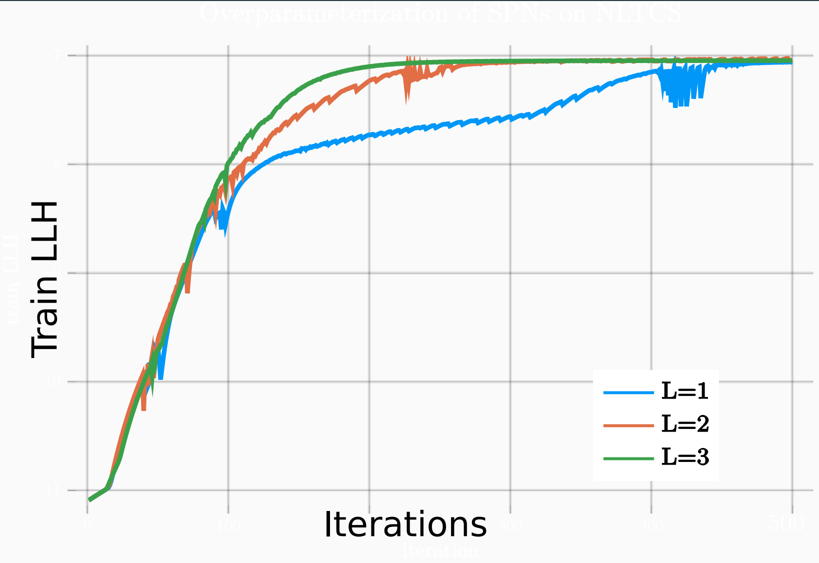




$$w_k^{(t)} \approx w_k^{(t)} + \rho^{(t)} \nabla_{w_k^{(t)}} + \sum_{\tau=1}^{t-1} \mu^{(t,\tau)} \nabla_{w_k^{(\tau)}} \tag{1}$$

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

Over-parametrization Experiment



Structure Learning

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?

General-Purpose Learners (Selection)

- LearnSPN [R. Gens & P. Domingos: Learning the structure of sum-product networks. In ICML, 2013.]
- ID-SPN [A. Rooshenas & D. Lowd: Learning Sum-Product Networks with Direct and Indirect Variable Interactions. In ICML, 2014.]
- **RAT-SPN** [R. Peharz et al.: Random sum-product networks: A simple but effective approach to probabilistic deep learning. In UAI, 2019.]
- **Bayesian SPN** [M. Trapp et al.: Bayesian learning of sum-product networks. In NeurIPS, 2019.]

Random Sum-Product Networks

Todo

Bayesian Structure

Why care about Bayesian structure learning?

- Competitive results.
- Occam's razor effect prevents overfitting.
- Works on heterogeneous data domains
- We can use nonparametric formulations, e.g. infinite SPNs.
- Can be used in cases of missing values, by using exact marginalisation of missing values.

Bayesian Structure

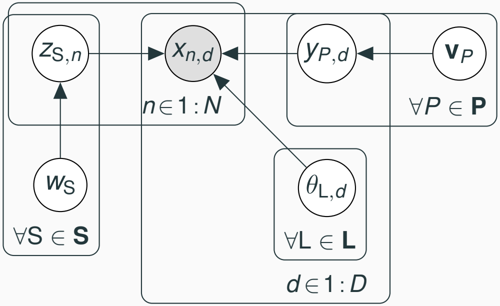


Figure 3: Generative model for Bayesian structure learning.

Bayesian Structure

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

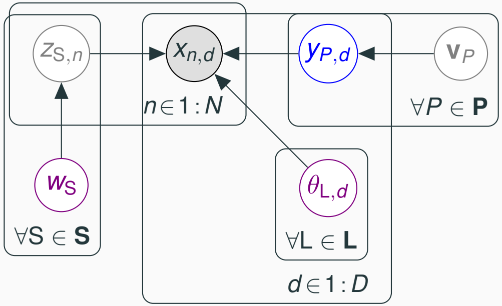
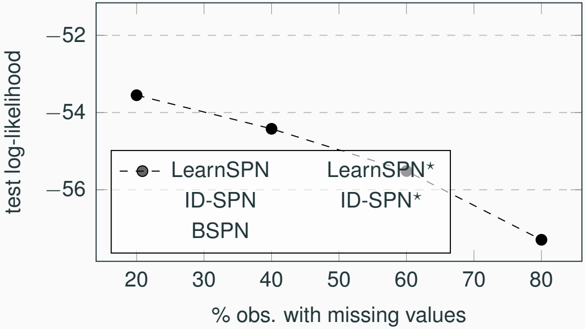


Figure 4: Generative model for Bayesian structure and learning.

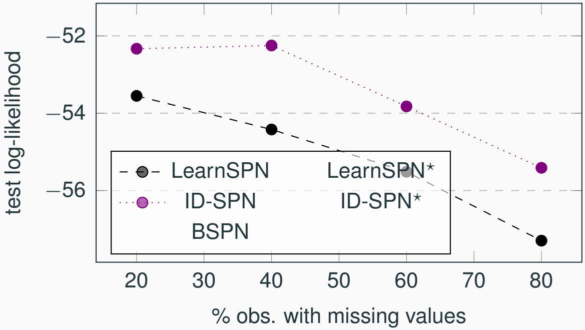
Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values (missing completely at random).



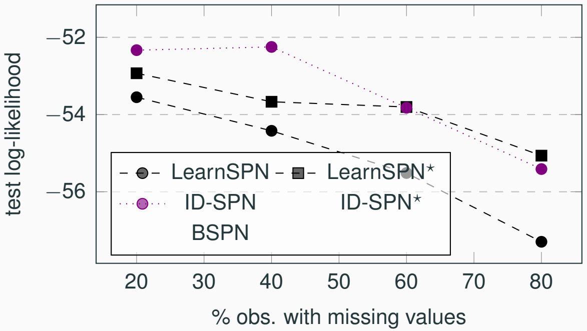
Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values (missing completely at random).



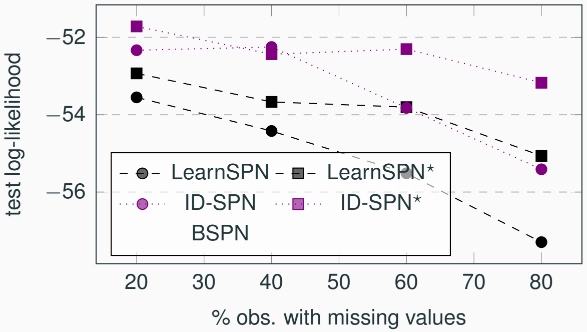
Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values (missing completely at random).



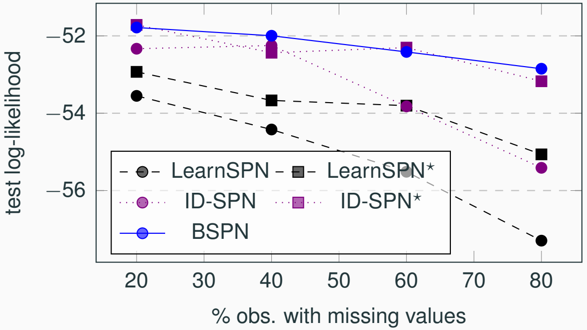
Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values (missing completely at random).



Bayesian Structure - Missing Values Experiment

Performance under increasing amount of missing values (missing completely at random).



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[6]

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

SPNs for Regression

We have recently[7] 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.

[7] 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.

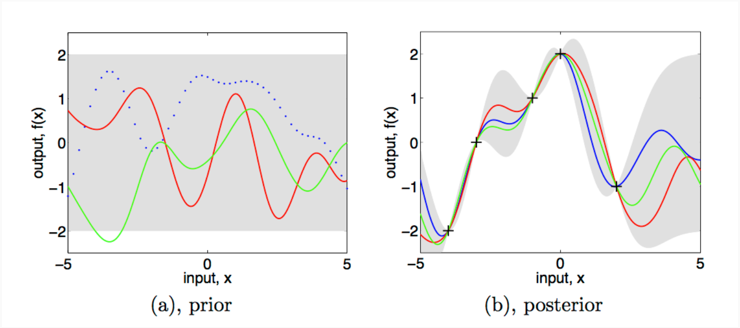


Figure 5: C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, 2006.