Advances in Learning Sum-Product Networks

Martin Trapp

November 5, 2019

Graz University of Technology

Outline

- Introduction to Sum-Product Networks
- Parameter Learning
- Structure Learning
- Applications of Sum-Product Networks

Outline

- Introduction to Sum-Product Networks
- Parameter Learning
  - semi-supervised learning
  - effects of overparameterization
- Structure Learning
  - principled structure learning
- Applications of Sum-Product Networks
  - out-of-domain detection
  - non-linear probabilistic regression

Introduction

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

- Sum-product networks (SPNs) [Poon2011] are tractable general-purpose machine learning models that admit exact probabilistic inference.
- Why do we care about probabilistic inference?

- Sum-product networks (SPNs) [Poon2011] are tractable general-purpose machine learning models that admit exact probabilistic inference.
- Why do we care about probabilistic inference?
- q: How likely is it that there is a traffic jam on Monday on the A2?

- Sum-product networks (SPNs) [Poon2011] are tractable general-purpose machine learning models that admit exact probabilistic inference.
- Why do we care about probabilistic inference?
- q: How likely is it that there is a traffic jam on Monday on the A2?
- Train a prediction model, e.g. a neural network.

- Sum-product networks (SPNs) [Poon2011] are tractable general-purpose machine learning models that admit exact probabilistic inference.
- Why do we care about probabilistic inference?
- q: How likely is it that there is a traffic jam on Monday on the A2?
- ~~Train a prediction model, e.g. a neural network.~~

- Sum-product networks (SPNs) [Poon2011] are tractable general-purpose machine learning models that admit exact probabilistic inference.
- Why do we care about probabilistic inference?
- q: How likely is it that there is a traffic jam on Monday on the A2?
- ~~Train a prediction model, e.g. a neural network.~~
- Answer probabilistic queries q on a probabilistic model m.

- q: How likely is it that there is a traffic jam on Monday on the A2?
- $\mathsf{X} = \{\text{Day}, \text{Time}, \text{JamA1}, \text{JamA2}, \ldots, \text{JamA}_n\}$.

- q: How likely is it that there is a traffic jam on Monday on the A2?
- $\mathsf{X} = \{\text{Day, Time, JamA1, JamA2}, \ldots, \text{JamA}_n\}$.
- $\mathsf{q}(\mathsf{m}) = p_m(\text{Day = Monday, JamA2 = 1})$.

- q: How likely is it that there is a traffic jam on Monday on the A2?
- $\mathsf{X} = \{$Day, Time, JamA1, JamA2, $\ldots$, JamA$_n\}$.
- $q(m) = p_m($Day = Monday, JamA2 = 1$)$.
- Therefore, we need to be able marginalise out variables, such as Date and JamA2.

- What are *tractable probabilistic models*?

- 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 compuatational complexity is at most polynomial $O(|q| \cdot |m|)$.

- 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 compuatational complexity is at most polynomial $O(|q| \cdot |m|)$.
- SPNs admit many probabilistic inference tasks, such as marginalisation, in linear time.

- 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 compuatational complexity is at most polynomial $O(|q| \cdot |m|)$.
- SPNs admit many probabilistic inference tasks, such as marginalisation, in linear time.
- To model a probability distribution over $\mathbf{X}$, SPNs use an *explicit* representation.

Note: Most recent generative models, such as GANs, are implicitly defined, intractable and require approximate inference.

Sum-Product Networks

What is a Sum-Product Network?

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

What is a Sum-Product Network?

- Let $\mathsf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables.
- An SPN is a distribution over X defined as a 4-tuple $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$. [Trapp2019]

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$. [Trapp2019]
  - $\boxed{\mathcal{G}}$ is a computational graph.

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$. [Trapp2019]
    - $\boxed{\mathcal{G}}$ is a computational graph.
    - $\boxed{\psi}$ is a so-called scope function.

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$. [Trapp2019]
    - $\boxed{\mathcal{G}}$ is a computational graph.
    - $\boxed{\psi}$ is a so-called scope function.
    - $\boxed{w}$ denotes the set of sum-weights and $\boxed{\theta}$ the set of leaf node parameters.

What is a Sum-Product Network?

- Let $\mathbf{X} = \{X_1, \ldots, X_D\}$ be set of D random variables.
- An SPN is a distribution over $\mathbf{X}$ defined as a 4-tuple $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$. [Trapp2019]
    - $\boxed{\mathcal{G}}$ is a computational graph.
    - $\boxed{\psi}$ is a so-called scope function.
    - $\boxed{w}$ denotes the set of sum-weights and $\boxed{\theta}$ the set of leaf node parameters.

Note: This definition is conceptually different to the classic definition of SPNs.

Computational Graph $\boxed{\mathcal{G}}$

$\boxed{\mathcal{G}}$ is a connected directed acyclic graph (DAG), containing three types of nodes: sums (S), products (P) and leaves (L).

Computational Graph $\boxed{\mathcal{G}}$

$\boxed{\mathcal{G}}$ is a connected directed acyclic graph (DAG), containing three types of nodes: sums (S), products (P) and leaves (L).

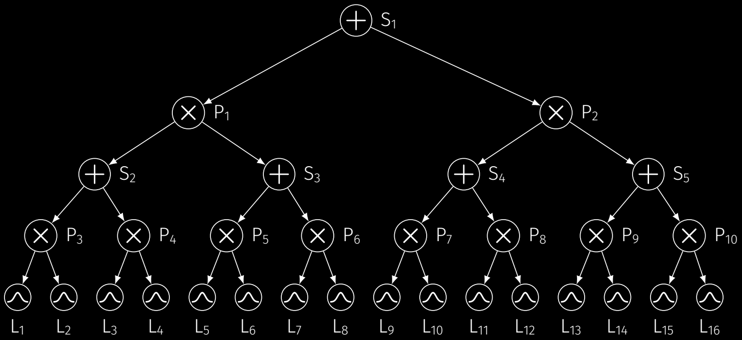


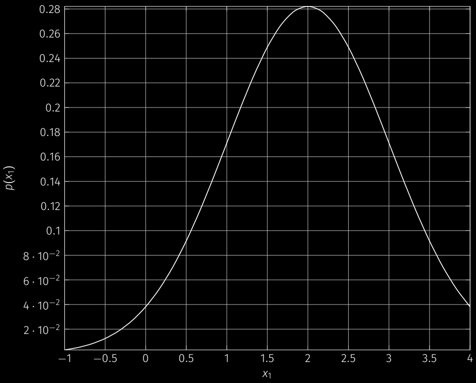
Figure 1: Example of a tree-shaped computational graph.

Leaves L in $\boxed{\mathcal{G}}$

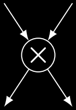
$$\mathsf{L}(x) = p(x \mid \boxed{\theta}_{\mathsf{L}})$$

Leaves L in $\boxed{\mathcal{G}}$

$$\mathrm{L}(x) = p(x \mid \boxed{\theta}_{\mathrm{L}})$$

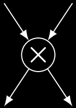


Product Nodes P in $\boxed{\mathcal{G}}$

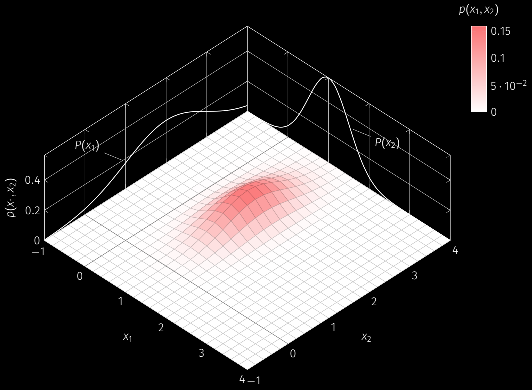


$$\mathsf{P}(x) = \prod_{\mathsf{C} \in \mathbf{ch}(\mathsf{P})} \mathsf{C}(x)$$

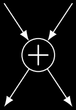
Product Nodes P in $\boxed{\mathcal{G}}$



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

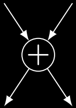


Sum Nodes S in $\boxed{\mathcal{G}}$

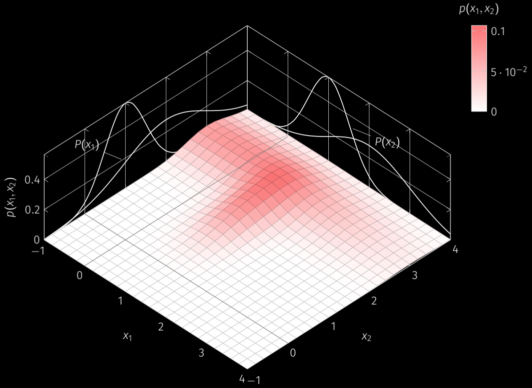


$$S(x) = \sum_{C \in \mathbf{ch}(P)} \boxed{w}_{S,C} C(x)$$

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



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



Note: We assume that $\boxed{w}_{S,C} \geq 0$ and $\sum_{C \in \mathbf{ch}(P)} \boxed{w}_{S,C} = 1$.

Scope Function $\boxed{\psi}$

- $\boxed{\psi}$ is a function assigning each node N in the graph a sub-set of X.[1]

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

Scope Function $\boxed{\psi}$

- $\boxed{\psi}$ is a function assigning each node N in the graph a sub-set of $\mathbf{X}$.[1]

A scope function has to fulfil the following properties:

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

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

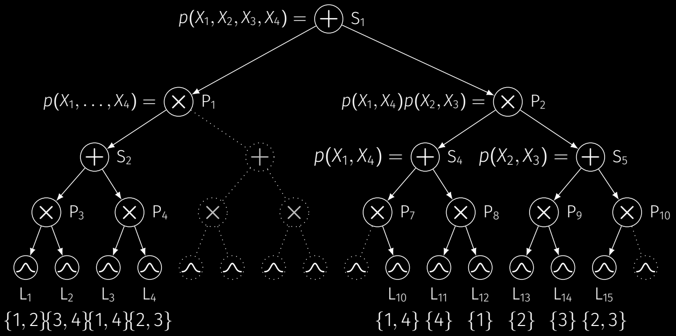
Completeness can be understood as requiring each sum node to be a well-defined mixture distribution.

Decomposability ψ

Decomposability ensures that each product node is a proper factorisation of its scope. Also, decomposability ensures we can “pull” down expensive operations, such as marginalisation or computation of expectations, down to the leaves enabelling tractable inference.

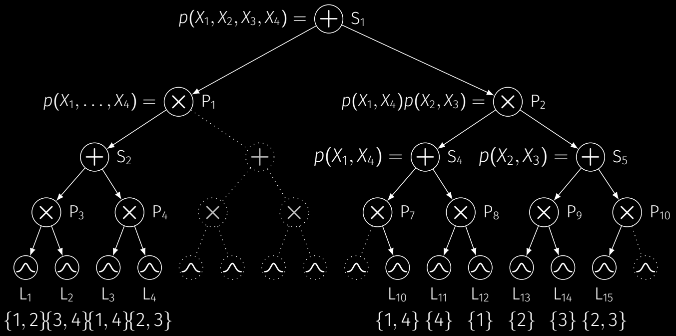
Example SPN $\boxed{\mathcal{S}} = (\boxed{\mathcal{G}}, \boxed{\psi}, \boxed{w}, \boxed{\theta})$

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



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

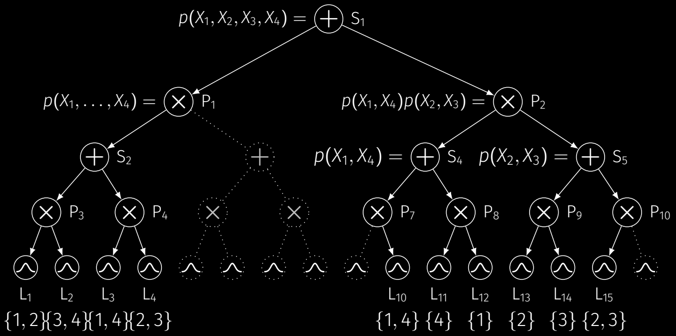
After applying a scope function $\boxed{\psi}$ on $\boxed{\mathcal{G}}$ we obtain the SPN.



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

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

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



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

Also note: Most structure learners learn $\boxed{\mathcal{G}}$ and $\boxed{\psi}$ in an entangled way.

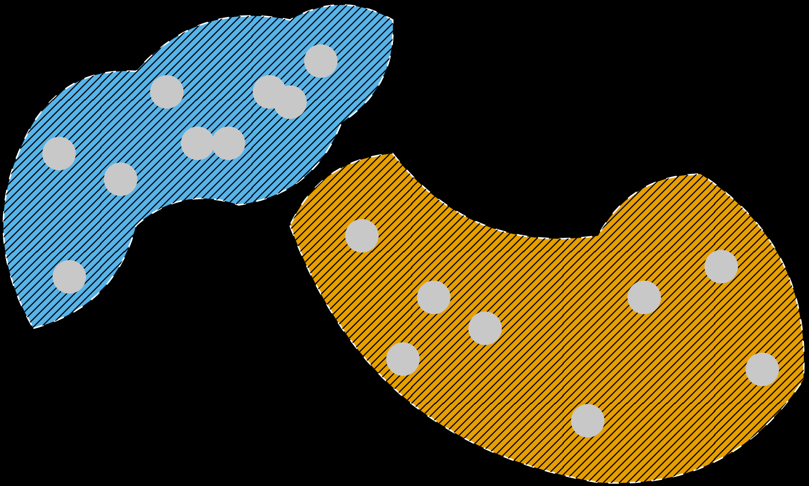
Parameter Learning of Sum-Product Networks

Parameter Learning

- State-of-the-art SPN parameter learning covers a wide range of well-developed techniques.
- The following selection is a small and *biased* sub-selection of the existing literature.

Generative Learning [Poon2011, Peharz2017]

Goal: Model the data distribution to be able to generate new data.



Generative Learning [Poon2011, Peharz2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\phi = (\boxed{w}, \boxed{\theta})$.

[2]Note: $\boxed{\mathcal{S}}(* \mid \phi)$ denotes the partition function which is a normalisation constant. If the SPN is normalised this evaluates to one.

Generative Learning [Poon2011, Peharz2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\phi = (\boxed{w}, \boxed{\theta})$.

To learn the parameters, we maximise the log-likelihood:[2]

$$\mathcal{L}(\phi \mid \mathcal{X}) = \sum_{n=1}^{N} \log \boxed{\mathcal{S}}(\mathrm{x_n} \mid \phi) - \log \boxed{\mathcal{S}}(* \mid \phi) \tag{1}$$

[2] Note: $\boxed{\mathcal{S}}(* \mid \phi)$ denotes the partition function which is a normalisation constant. If the SPN is normalised this evaluates to one.

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\phi = (\boxed{W}, \boxed{\theta})$.

To learn the parameters, we maximise the log-likelihood:[2]

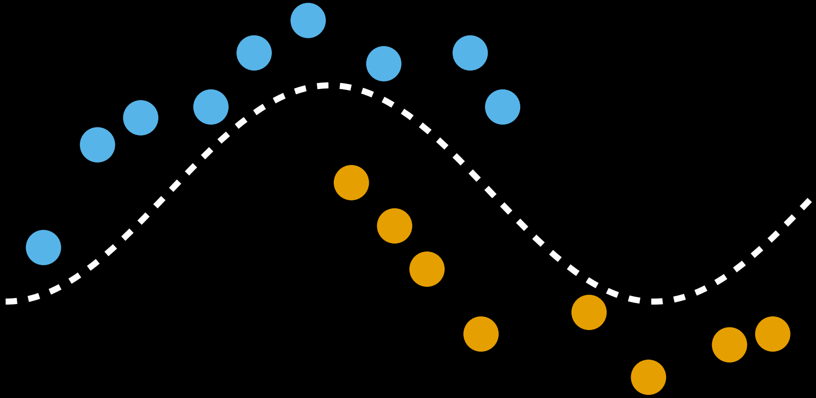
$$\mathcal{L}(\phi \mid \mathcal{X}) = \sum_{n=1}^{N} \log \boxed{\mathcal{S}}(\mathrm{x_n} \mid \phi) - \log \boxed{\mathcal{S}}(* \mid \phi) \tag{1}$$

This is usually done using expectation-maximisation or gradient-based optimisation.

[2]Note: $\boxed{\mathcal{S}}(* \mid \phi)$ denotes the partition function which is a normalisation constant. If the SPN is normalised this evaluates to one.

Supervised Learning [Gens2012]

Goal: Find a separation of pre-defined classes given labelled training examples.



Supervised Learning [Gens2012]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels and $\phi = (\boxed{W}, \boxed{\theta})$.

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels and $\phi = (\boxed{W}, \boxed{\theta})$.

To learn the parameters, we maximise the conditional log-likelihood (or the cross-entropy):

$$\mathcal{L}(\phi, \boldsymbol{\lambda} \mid \mathcal{X}) = \sum_{n=1}^{N} \log \boxed{\mathcal{S}}(\mathsf{x_n}, \lambda_\mathsf{n} \mid \phi) - \log \boxed{\mathcal{S}}(\mathsf{x_n} \mid \phi) \qquad (2)$$

Supervised Learning [Gens2012]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels and $\phi = (\boxed{W}, \boxed{\theta})$.

To learn the parameters, we maximise the conditional log-likelihood (or the cross-entropy):

$$\mathcal{L}(\phi, \boldsymbol{\lambda} \mid \mathcal{X}) = \sum_{n=1}^{N} \log \boxed{\mathcal{S}}(\mathsf{x_n}, \lambda_\mathsf{n} \mid \phi) - \log \boxed{\mathcal{S}}(\mathsf{x_n} \mid \phi) \tag{2}$$

Note: [Peharz2019] proposed a hybrid generative-discriminative loss.

Semi-Supervised Learning [Trapp2017]

Goal: Find a separation of pre-defined classes given few labelled and many unlabelled training examples.

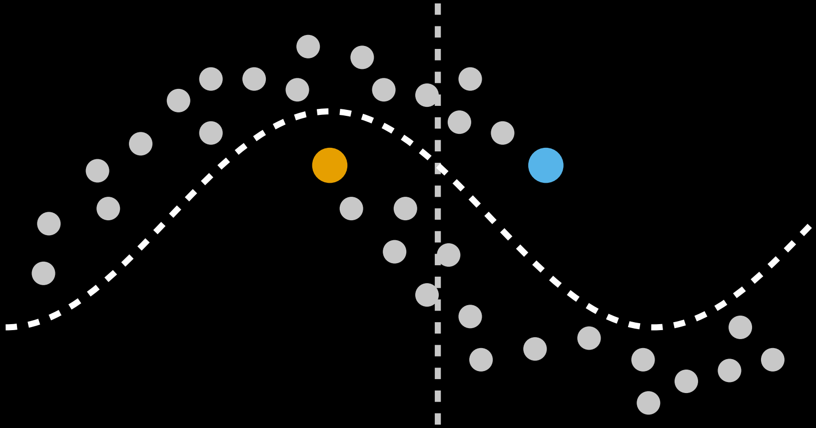


Figure 2: Example of a semi-supervised learning problem.

Semi-Supervised Learning [Trapp2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote labelled training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels.

Semi-Supervised Learning [Trapp2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote labelled training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels.

Further, let $\mathcal{U} = \{\boldsymbol{u}_n\}_m$ with $\boldsymbol{u}_n \in \mathbb{R}^D$ denote *unlabelled* training samples and let $\boldsymbol{q} \in \Delta_{K-1}$ denote soft-labels with $\boldsymbol{q}_m \in \Delta_{K-1} \in$ that we aim to infer. ϕ^+ denotes the parameters of an SPN trained solely on labelled data.

Semi-Supervised Learning [Trapp2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote labelled training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels.

Further, let $\mathcal{U} = \{\boldsymbol{u}_n\}_m$ with $\boldsymbol{u}_n \in \mathbb{R}^D$ denote *unlabelled* training samples and let $\boldsymbol{q} \in \Delta_{K-1}$ denote soft-labels with $\boldsymbol{q}_m \in \Delta_{K-1} \in$ that we aim to infer. ϕ^+ denotes the parameters of an SPN trained solely on labelled data.

$$\operatorname*{argmax}_{\phi \in \Phi} \arg \min_{\boldsymbol{q} \in \Delta_{K-1}} \mathcal{L}(\phi, \boldsymbol{\lambda}, \boldsymbol{q} \,|\, \mathcal{X}, \mathcal{U}) - \mathcal{L}(\phi^+, \boldsymbol{\lambda}, \boldsymbol{q} \,|\, \mathcal{X}, \mathcal{U}) \qquad (3)$$

Semi-Supervised Learning [Trapp2017]

Let $\mathcal{X} = \{x_n\}_n^N$ with $xn \in \mathbb{R}^D$ denote labelled training samples and $\boldsymbol{\lambda} = \{\lambda_n\}_n^N$ with $\lambda_n \in \mathbb{R}$ their respective class labels.

Further, let $\mathcal{U} = \{\boldsymbol{u}_n\}_m$ with $\boldsymbol{u}_n \in \mathbb{R}^D$ denote *unlabelled* training samples and let $\boldsymbol{q} \in \Delta_{K-1}$ denote soft-labels with $\boldsymbol{q}_m \in \Delta_{K-1} \in$ that we aim to infer. ϕ^+ denotes the parameters of an SPN trained solely on labelled data.

$$\operatorname*{argmax}_{\phi \in \Phi} \arg \min_{\boldsymbol{q} \in \Delta_{K-1}} \mathcal{L}(\phi, \boldsymbol{\lambda}, \boldsymbol{q} \,|\, \mathcal{X}, \mathcal{U}) - \mathcal{L}(\phi^+, \boldsymbol{\lambda}, \boldsymbol{q} \,|\, \mathcal{X}, \mathcal{U}) \qquad (3)$$

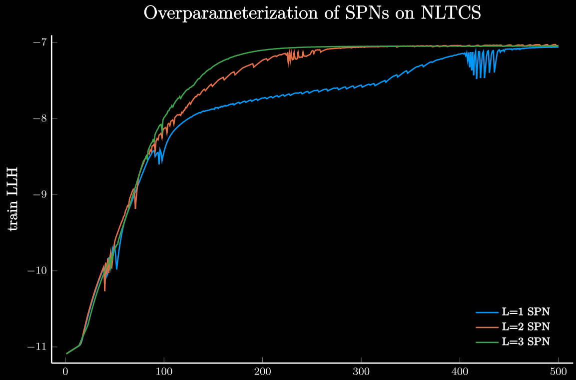
Intuitively, this objective (contrastive pessimistic likelihood estimation) tries to find the best semi-supervised learner (ϕ) under the worst soft-labels $\boldsymbol{q}$.

Overparameterization in SPNs [Trapp2019b]

Gradient-based optimisation of the weights of any deep tree-structured sum-product network with small (fixed) learning rate η and near zero initialisation of the weights $\boxed{w}$ is equivalent to gradient-based optimisation with adaptive and time-varying learning rate and momentum term.

Overparameterization in SPNs [Trapp2019b]

Gradient-based optimisation of the weights of any deep tree-structured sum-product network with small (fixed) learning rate η and near zero initialisation of the weights $\boxed{w}$ is equivalent to gradient-based optimisation with adaptive and time-varying learning rate and momentum term.



Structure Learning of Sum-Product Networks

Learners tailored to images

- [Poon2011] presented a structure learning approach which recursively sub-divides images into sub-regions.

[3] This approach can also be applied to non-image data.

Learners tailored to images

- [Poon2011] presented a structure learning approach which recursively sub-divides images into sub-regions.
- Later, [Gens2012] proposed a structure that decomposes images into parts.

[3] This approach can also be applied to non-image data.

Learners tailored to images

- [Poon2011] presented a structure learning approach which recursively sub-divides images into sub-regions.
- Later, [Gens2012] proposed a structure that decomposes images into parts.
- Recently, [Peharz2019] presented a randomly constructed structure that divides images into randomly defined sub-regions. [3]

[3] This approach can also be applied to non-image data.

General-purpose learners

- [Gens2013] proposed one of the most frequently used algorithms, called LearnSPN.

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

Structure Learning

General-purpose learners

- [Gens2013] proposed one of the most frequently used algorithms, called LearnSPN.
- Later, a wide range of variants of LearnSPN have been proposed to allow structure learning on continuous data or heterogeneous data and to prune the structures in order to improve the generalisation of the structures.[4]

---

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

General-purpose learners

- [Gens2013] proposed one of the most frequently used algorithms, called LearnSPN.
- Later, a wide range of variants of LearnSPN have been proposed to allow structure learning on continuous data or heterogeneous data and to prune the structures in order to improve the generalisation of the structures.[4]
- However, all of the existing approaches refrain from asking *What is a good structure?*

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

Structure Learning

General-purpose learners

- [Gens2013] proposed one of the most frequently used algorithms, called LearnSPN.
- Later, a wide range of variants of LearnSPN have been proposed to allow structure learning on continuous data or heterogeneous data and to prune the structures in order to improve the generalisation of the structures.[4]
- However, all of the existing approaches refrain from asking *What is a good structure?*
- Recently, we presented the first *principled* approach that aims to change this practice! [Trapp2019]

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

Bayesian Parameter Learning

- The key insight for Bayesian parameter learning [Zhao2016, Vergari2019] is that *sum nodes can be interpreted as latent variables* $\boxed{Z}_S$, clustering data instances.

Bayesian Parameter Learning

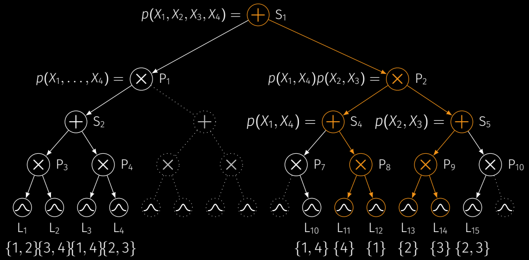
- The key insight for Bayesian parameter learning [Zhao2016, Vergari2019] is that *sum nodes can be interpreted as latent variables* $\boxed{Z}_S$, clustering data instances.
- Now, consider $\boxed{z}$ to be a vector containing a state for each S in the SPN.

Bayesian Parameter Learning

- The key insight for Bayesian parameter learning [Zhao2016, Vergari2019] is that *sum nodes can be interpreted as latent variables* $\boxed{Z}_S$, clustering data instances.
- Now, consider $\boxed{z}$ to be a vector containing a state for each S in the SPN.
- Then a $\boxed{\mathcal{T}}$ is a so-called induced tree [Zhao2016] which is induced by $\boxed{z}$.

Bayesian Parameter Learning

- The key insight for Bayesian parameter learning [Zhao2016, Vergari2019] is that *sum nodes can be interpreted as latent variables* $\boxed{Z}_S$, clustering data instances.
- Now, consider $\boxed{z}$ to be a vector containing a state for each S in the SPN.
- Then a $\boxed{\mathcal{T}}$ is a so-called induced tree [Zhao2016] which is induced by $\boxed{z}$.



Bayesian Parameter Learning

- Let $T(\boxed{z})$ denote a surjective (not injective) function that assigns to each value $\boxed{z}$ the induced tree $\boxed{\mathcal{T}}$, it is now conceptually straightforward to extend an SPN to a Bayesian setting.

Bayesian Parameter Learning

- Let $T(\boxed{z})$ denote a surjective (not injective) function that assigns to each value $\boxed{z}$ the induced tree $\boxed{\mathcal{T}}$, it is now conceptually straightforward to extend an SPN to a Bayesian setting.

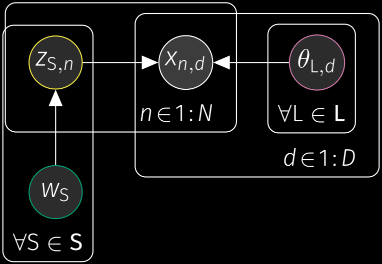


Figure 3: Generative model for Bayesian parameter learning.

Bayesian Structure & Parameter Learning [Trapp2019]

- For Bayesian structure & parameter learning of SPNs we assume $\boxed{\mathcal{G}}$ is a tree-shaped region graph.

Bayesian Structure & Parameter Learning [Trapp2019]

- For Bayesian structure & parameter learning of SPNs we assume $\boxed{\mathcal{G}}$ is a tree-shaped region graph.
- A region graph $\mathcal{R}$ is a vectorised representation of SPNs and is a connected DAG containing two types of nodes: regions ($R \in \mathbf{R}$) and partitions ($P \in \mathbf{P}$).

Bayesian Structure & Parameter Learning [Trapp2019]

- For Bayesian structure & parameter learning of SPNs we assume $\boxed{\mathcal{G}}$ is a tree-shaped region graph.
- A region graph $\mathcal{R}$ is a vectorised representation of SPNs and is a connected DAG containing two types of nodes: regions ($R \in \mathbf{R}$) and partitions ($P \in \mathbf{P}$).

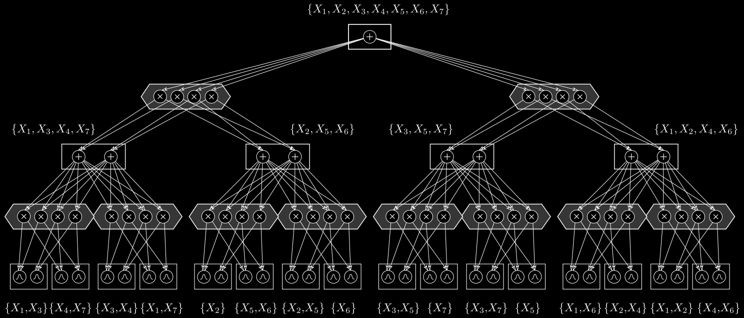


Figure 4: Example region-graph. Based on the illustration by [Peharz2019].

Bayesian Structure & Parameter Learning

- For each data d we introduce a latent variable $\boxed{Y}_{P,d}$.

Bayesian Structure & Parameter Learning

- For each data d we introduce a latent variable $\boxed{Y}_{P,d}$.
- The latent variable represents a decision to assign dimension d to a particular child, given that all partitions above decided to assign d onto the (unique) path leading to the partition.

Bayesian Structure & Parameter Learning

- For each data d we introduce a latent variable $\boxed{Y}_{P,d}$.
- The latent variable represents a decision to assign dimension d to a particular child, given that all partitions above decided to assign d onto the (unique) path leading to the partition.

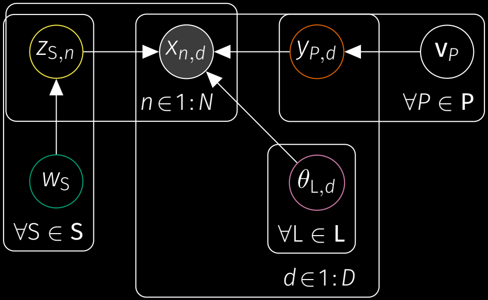


Figure 5: Generative model for Bayesian structure and parameter learning.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters $\boxed{w}$, $\boxed{\theta}$ (fixed $\boxed{\mathbf{y}}$), and ii) updating $\boxed{\mathbf{y}}$ (fixed $\boxed{w}$, $\boxed{\theta}$) to learn Bayesian SPNs.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters $\boxed{w}$, $\boxed{\theta}$ (fixed $\boxed{\mathbf{y}}$), and ii) updating $\boxed{\mathbf{y}}$ (fixed $\boxed{w}$, $\boxed{\theta}$) to learn Bayesian SPNs.
- This emperically approach has shown to be sufficient for real-world dataset with up to 1556 dimensions.

Posterior Inference

- We perform Gibbs sampling alternating between i) updating parameters $\boxed{w}$, $\boxed{\theta}$ (fixed $\boxed{\mathbf{y}}$), and ii) updating $\boxed{\mathbf{y}}$ (fixed $\boxed{w}$, $\boxed{\theta}$) to learn Bayesian SPNs.
- This emperically approach has shown to be sufficient for real-world dataset with up to 1556 dimensions.
- More sophisticated approaches, e.g. particle Gibbs sampling or variational inference, might be interesting future avenues.

Experiments

- We compared the performance of our approach against SOTA structure learners on discrete, heterogeneous and data with missing values.

Experiments

- We compared the performance of our approach against SOTA structure learners on discrete, heterogeneous and data with missing values.
- The discrete datasets are standard benchmark datasets.

Experiments

- We compared the performance of our approach against SOTA structure learners on discrete, heterogeneous and data with missing values.
- The discrete datasets are standard benchmark datasets.
- We evaluated against an increasing number of observations having 50% dimensions missing completely at random to assess the robustness.

Experiments

- We compared the performance of our approach against SOTA structure learners on discrete, heterogeneous and data with missing values.
- The discrete datasets are standard benchmark datasets.
- We evaluated against an increasing number of observations having 50% dimensions missing completely at random to assess the robustness.

Note: Existing approach cannot handle missing data during structure learning, so we either removed the samples with missing data or used k-NN imputation.

Experiments (discrete)

Dataset	LearnSPN	RAT-SPN	CCCP	ID-SPN	ours	ours$^\infty$	BTD
NLTCS	−6.11	−6.01	−6.03	−6.02	−6.00	−6.02	−5.97
MSNBC	−6.11	−6.04	−6.05	−6.04	−6.06	−6.03	−6.03
KDD	−2.18	−2.13	−2.13	−2.13	−2.12	−2.13	−2.11
Plants	−12.98	−13.44	−12.87	−12.54	−12.68	−12.94	−11.84
Audio	−40.50	−39.96	−40.02	−39.79	−39.77	−39.79	−39.39
Jester	−53.48	−52.97	−52.88	−52.86	−52.42	−52.86	−51.29
Netflix	−57.33	−56.85	−56.78	−56.36	−56.31	−56.80	−55.71
Accidents	−30.04	−35.49	−27.70	−26.98	−34.10	−33.89	−26.98
Retail	−11.04	−10.91	−10.92	−10.85	−10.83	−10.83	−10.72
Pumsb-star	−24.78	−32.53	−24.23	−22.41	−31.34	−31.96	−22.41
DNA	−82.52	−97.23	−84.92	−81.21	−92.95	−92.84	−81.07
Kosarak	−10.99	−10.89	−10.88	−10.60	−10.74	−10.77	−10.52
MSWeb	−10.25	−10.12	−9.97	−9.73	−9.88	−9.89	−9.62
Book	−35.89	−34.68	−35.01	−34.14	−34.13	−34.34	−34.14
EachMovie	−52.49	−53.63	−52.56	−51.51	−51.66	−50.94	−50.34
WebKB	−158.20	−157.53	−157.49	−151.84	−156.02	−157.33	−149.20
Reuters-52	−85.07	−87.37	−84.63	−83.35	−84.31	−84.44	−81.87
20 Newsgrp	−155.93	−152.06	−153.21	−151.47	−151.99	−151.95	−151.02
BBC	−250.69	−252.14	−248.60	−248.93	−249.70	−254.69	−229.21
AD	−19.73	−48.47	−27.20	−19.05	−63.80	−63.80	−14.00

Experiments (missing data)

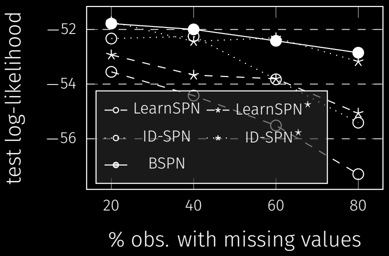


Figure 6: EachMovie (D: 500, N: 5526)

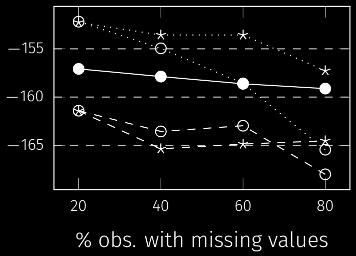


Figure 7: WebKB (D: 839, N: 3361)

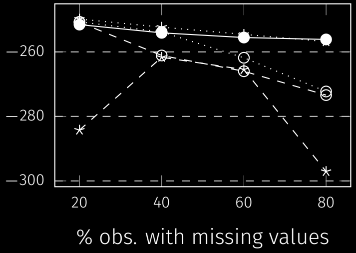
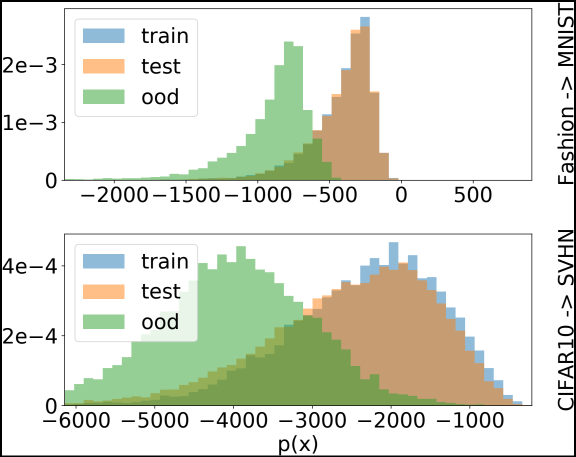


Figure 8: BBC (D: 1058, N: 1895)

Recent Applications

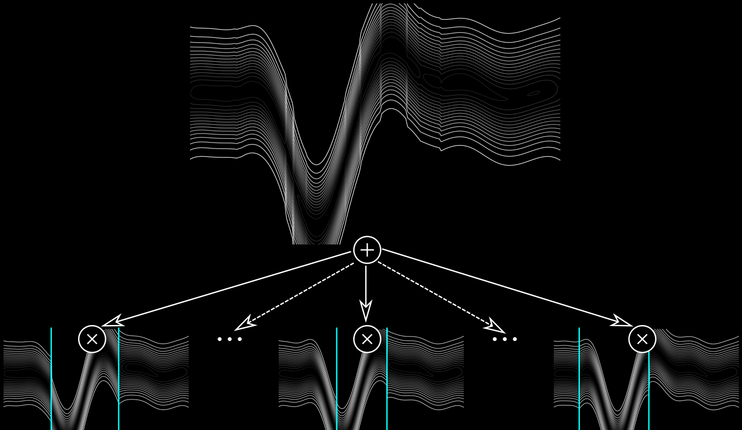
Out-Of-Domain Detection [Peharz2019]

Histograms of the log-likelihoods of RAT-SPNs on the native data set (blue: train, orange: test) and out-of-domain (ood) data set (green).



Non-linear Probabilistic Regression [Trapp2019c]

Deep structured mixtures of Gaussian processes combine exact inference in Gaussian processes with exact and efficient inference in SPNs to obtain an efficient probabilistic non-linear regression model for large-scale data.



Conclusion

- Sum-product networks (SPNs) are a effective class of probabilistic models that allow exact and efficient probabilistic inference.

Conclusion

- Sum-product networks (SPNs) are a effective class of probabilistic models that allow exact and efficient probabilistic inference.
- SPNs have already been applied to various interesting applications and gained traction in the last years.

Conclusion

- Sum-product networks (SPNs) are a effective class of probabilistic models that allow exact and efficient probabilistic inference.
- SPNs have already been applied to various interesting applications and gained traction in the last years.
- Parameter learning in SPNs is well-developed containing methods for generative learning, supervised learning and semi-supervised learning.

Conclusion

- Sum-product networks (SPNs) are a effective class of probabilistic models that allow exact and efficient probabilistic inference.
- SPNs have already been applied to various interesting applications and gained traction in the last years.
- Parameter learning in SPNs is well-developed containing methods for generative learning, supervised learning and semi-supervised learning.
- However, most structure learning approaches are based on intuition and refrain from declaring the goal of structure learning.

Conclusion

- Sum-product networks (SPNs) are a effective class of probabilistic models that allow exact and efficient probabilistic inference.
- SPNs have already been applied to various interesting applications and gained traction in the last years.
- Parameter learning in SPNs is well-developed containing methods for generative learning, supervised learning and semi-supervised learning.
- However, most structure learning approaches are based on intuition and refrain from declaring the goal of structure learning.
- We recently proposed an approach that aims to *change* this practice using a Bayesian formulation.

References

[Poon2011] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In Proceedings of UAI, pages 337–346, 2011.

[Gens2012] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In Proceedings of NeurIPS, pages 3248–3256, 2012.

[Gens2013] R. Gens and P. Domingos. Learning the structure of sum-product networks. Proceedings of ICML, pages 873–880, 2013.

[Zhao2016] H. Zhao, T. Adel, G. J. Gordon, and B. Amos. Collapsed variational inference for sum-product networks. In Proceedings of ICML, pages 1310–1318, 2016.

[Vergari2019] A. Vergari, A. Molina, R. Peharz, Z. Ghahramani, K. Kersting, and I. Valera. Automatic Bayesian density analysis. In Proceedings of AAAI, 2019.

[Peharz2017] R. Peharz, R. Gens, F. Pernkopf, and P. Domingos. On the latent variable interpretation in sum-product networks. TPAMI, 39(10):2030–2044, 2017.

[Peharz2019] R. Peharz, A. Vergari, K. Stelzner, A. Molina, X. Shao, M. Trapp, K. Kersting, and Z. Ghahramani. Random sum-product networks: A simple but effective approach to probabilistic deep learning. In Proceedings of UAI, 2019.

[Trapp2017] M. Trapp, T. Madl, R. Peharz, F. Pernkopf, and R. Trappl. Safe semi-supervised learning of sum-product networks. In Proceedings of UAI, 2017.

[Trapp2019] M. Trapp, R. Peharz, H. Ge, F. Pernkopf, and Z. Ghahramani. Bayesian Learning of Sum-Product Networks. To appear at NeurIPS, 2019.

[Trapp2019b] M. Trapp, R. Peharz, and F. Pernkopf. Optimisation of overparametrized sum-product networks. CoRR, abs/1905.08196, 2019.

[Trapp2019c] M. Trapp, R. Peharz, F. Pernkopf and Carl E. Rasmussen. Deep structured mixtures of Gaussian processes. CoRR, abs/1910.04536, 2019.