Università degli Studi di Bari
Dipartimento di Informatica

LACAM
Machine Learning

Learning Sum-Product Networks

Nicola Di Mauro **Antonio Vergari**

September 2016

The need for SPN

Why should you work on SPNs?

Probabilistic modeling of data aims at

- representing probability distributions *compactly*
- computing their marginals and modes *efficiently* (inference)
- learning them *accurately*

A solution is to use Probabilistic Graphical Models (PGMs).
However, PGMs are limited in

- representing compact distributions
- having intractable exact inference in the worst case
  - falling back on approximate inference
- requiring and exponential sample size (wrt the number of variables)
- learning the structure since it requires inference

The need for SPN (II)

Why should you work on SPNs?

Sum-Product Networks (SPNs) are a type of probabilistic model[1]

- a deep model with multiple layer of hidden variables
- tractable exact inference

SPNs represent probability distributions and a corresponding exact inference machine for the represented distribution at the same time

[1] H. Poon and P. Domingos, *Sum-Product Network: a New Deep Architecture*, UAI 2011

Representation

Density estimation

(Different kinds of) Inference

Different kinds of queries:

- $p(\mathbf{X})$ (evidence)
- $p(\mathbf{E}), \mathbf{E} \subset \mathbf{X}$ (marginals)
- $p(\mathbf{Q}|\mathbf{E}), \mathbf{Q}, \mathbf{E} \subset \mathbf{X}, \mathbf{Q} \cap \mathbf{E} = \emptyset$ (conditionals)
- $\arg\max_{\mathbf{q} \sim \mathbf{Q}} p(\mathbf{q}|\mathbf{E})$ (MPE assignment)
- complex queries

Tractable Probabilistic Models

Sum-Product Networks

Definition (Sum-Product Network)

A *Sum-Product Network* (SPN) S over RVs $\mathbf{X}$ is a rooted weighted DAG (network) consisting of distribution *leaves* (network inputs), *sum* and *product* nodes (inner nodes). Let $\mathsf{ch}(n)$ be the set of all the child (input) nodes of a node n and $\mathsf{pa}(n)$ the set of all its parent (output) nodes. A leaf n defines a tractable, possibly unnormalized, distribution ϕ_n over some RVs in $\mathbf{X}$. A nonnegative weight w_{nc} is associated to each edge linking a sum node n to $c \in \mathsf{ch}(n)$. We indicate with $\mathbf{w}$ the set of all weights in the network S (network parameters). S_n denotes the sub-network rooted at node n and parametrized by $\mathbf{w}_n$, comprising all descendant nodes of n, recursively defined as the children of n and their descendants.

Scopes

Definition (Scope)

Let S be an SPN over RVs $\mathbf{X}$. The *scope* of a node n in S is denoted as $\mathsf{sc}(n) \subseteq \mathbf{X}$. The scope of a leaf node n is defined as the set of RVs over which ϕ_n is defined. The scope of an inner node n is defined as $\mathsf{sc}(n) = \bigcup_{c \in \mathsf{ch}(n)} \mathsf{sc}(c)$. The scope of S is the scope of its root, i.e. $\mathbf{X}$.

Definition (SPN evaluation)

Let S be an SPN over RVs $\mathbf{X}$ with parameters $\mathbf{w}$ and S_n be the sub-network rooted at node n. For each $\mathbf{x} \sim \mathbf{X}$, $S_n(\mathbf{x}_{|\mathsf{sc}(n)})$ and $S_n(\mathbf{x})$ are the same notation to indicate the output value of node n after $\mathbf{X} = \mathbf{x}$ is observed as the network input. Then, $S_n(\mathbf{x})$ can be computed as follows:

$$S_n(\mathbf{x}) = \begin{cases} \phi_n(\mathsf{sc}(n) = \mathbf{x}_{|\mathsf{sc}(n)}), & \text{if } n \text{ is a leaf node} \\ \sum_{c \in \mathsf{ch}(n)} w_{nc} S_c(\mathbf{x}) & \text{if } n \text{ is a sum node} \\ \prod_{c \in \mathsf{ch}(n)} S_c(\mathbf{x}) & \text{if } n \text{ is a product node.} \end{cases} \tag{1}$$

The value of the whole network, i.e. $S(\mathbf{x})$, corresponds to the value of its root

Structural Properties

If a network S is *valid*, evaluating it corresponds to evaluate a joint unnormalized probability distribution $p_{\mathbf{X}}$. That is, $\forall \mathbf{x}, S(\mathbf{x})/Z = p(\mathbf{X} = \mathbf{x})$, where Z is the normalizing *partition* function defined as $Z = \sum_{\mathbf{x} \sim \mathbf{X}} S(\mathbf{x})$. This actually is possible since a complete and decomposable SPN correctly compiles the *extended network polynomial* encoding the distribution $p_{\mathbf{X}}$ **Peharz2015a** To satisfy validity, it is sufficient for a network to be *complete* and *decomposable* **Darwiche2009**; **Poon2011**

Definition (Completeness, decomposability and validity)

Let S be an SPN and let $\mathbf{S}_{\oplus}$ (resp. $\mathbf{S}_{\otimes}$) be the set of all sum (resp. product) nodes in S.

1. S is *complete* iff $\forall n \in \mathbf{S}_{\oplus}, \forall c_1, c_2 \in \mathsf{ch}(n) : \mathsf{sc}(c_1) = \mathsf{sc}(c_2)$.
2. S is *decomposable* iff
   $\forall n \in \mathbf{S}_{\otimes}, \forall c_1, c_2 \in \mathsf{ch}(n), c_1 \neq c_2 : \mathsf{sc}(c_1) \cap \mathsf{sc}(c_2) = \emptyset$.
3. If S is complete and decomposable, then it is *valid*.

Inference

Complete evidence

Complete evidence inference consists of a single bottom-up (feedforward) evaluation of the network and proceeds according to Eq. 1. Therefore, it is guaranteed to be tractable as long as the network size is polynomial in $|\mathbf{X}|$.

Marginal inference

Exact marginal inference can be computed with the same time complexity if the validity property holds **Poon2011** the network can be evaluated in a similar fashion as for the complete evidence case. To compute a marginal query like $p(\mathbf{Q} = \mathbf{q}), \mathbf{Q} \subset \mathbf{X}$, one has to evaluate each leaf n as:

$$S_n(\mathbf{q}) = \begin{cases} p(\text{sc}(n) = \mathbf{q}_{|\text{sc}(n)}) & \text{if sc}(n) \subseteq \mathbf{Q} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

and then propagate the outputs as before. This mechanism highlights the recursive probabilistic interpretation of SPNs, in that each sub-network modeling a probability distribution over its own scope, shall output 1 as the probability of marginalizing over all the RVs that scope. As a consequence, even conditional probability queries are also computable in tractable time, since $p(\mathbf{Q}|\mathbf{E}) = p(\mathbf{Q}, \mathbf{E})/p(\mathbf{E})$, for $\mathbf{Q}, \mathbf{E} \subset \mathbf{X}$. In addition to that, setting all leaf outputs to 1 equals to compute the *partition function* Z (whose value, for an unnormalized SPN, will differ from 1).

MPE inference

An approximation of MPE inference and the computation of an MPE assignment can be answered in linear time as well **Peharz2015b**; **Peharz2016** Given an SPN S over $\mathbf{X}$, to find an MPE assignment:

$$\mathbf{q}^* = \operatorname*{argmax}_{\mathbf{q} \sim \mathbf{Q}} p(\mathbf{E}, \mathbf{q}) \tag{3}$$

for some RVs $\mathbf{E}, \mathbf{Q} \subset \mathbf{X}, \mathbf{E} \cap \mathbf{Q} = \emptyset, \mathbf{E} \cup \mathbf{Q} = \mathbf{X}$, S is transformed into a Max-Product Network (MPN) M. M is built by substituting each $n \in \mathbf{S}_{\oplus}$ for a max node computing $\max_{c \in \mathsf{ch}(n)} w_{nc} M_n$ instead. Then M is evaluated bottom-up after setting all leaves n, $\mathsf{sc}(n) \subseteq \mathbf{Q}$, to output 1. A Viterbi-like top-down traversal traces back the MPE assignment for each RV in $\mathbf{Q}$. By starting from the root and following only the max output child branch of a max node and all the child branches of a product node **Poon2011a** each instance determines a *tree* path whose leaves MPE assignments union forms the query answer **Darwiche2009**

Interpretation

Interpretation

- probabilistic model
- deep feedforward neural network

Network Polynomials

Arithmetic Circuits

Differences with ACs:

- probabilistic semantics
  - learning
  - sampling
- no shared weights

SPNs as NNs (I)

SPNs are a particular kind of ***labelled constrained*** *and* ***fully probabilistic*** neural networks.

Labelled: each neuron is associated a *scope*
Constrained: completeness and decomposability determine network topology.
Fully probabilistic: each valid sub-SPN is still a valid-SPN.

SPNs provide a direct encoding of the input space into a deep architecture $\rightarrow$ ***visualizing representations*** (back) into the ***input space***.

Vergari2016a.

SPNs as NNs (II)

A classic MLP hidden layer computes the function:

$$h(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

SPNs can be reframed as *DAGs* of MLPs, each sum layer computing:

$$\mathbf{S}(\mathbf{x}) = \log(\mathbf{W}\mathbf{x})$$

and product layers computing:

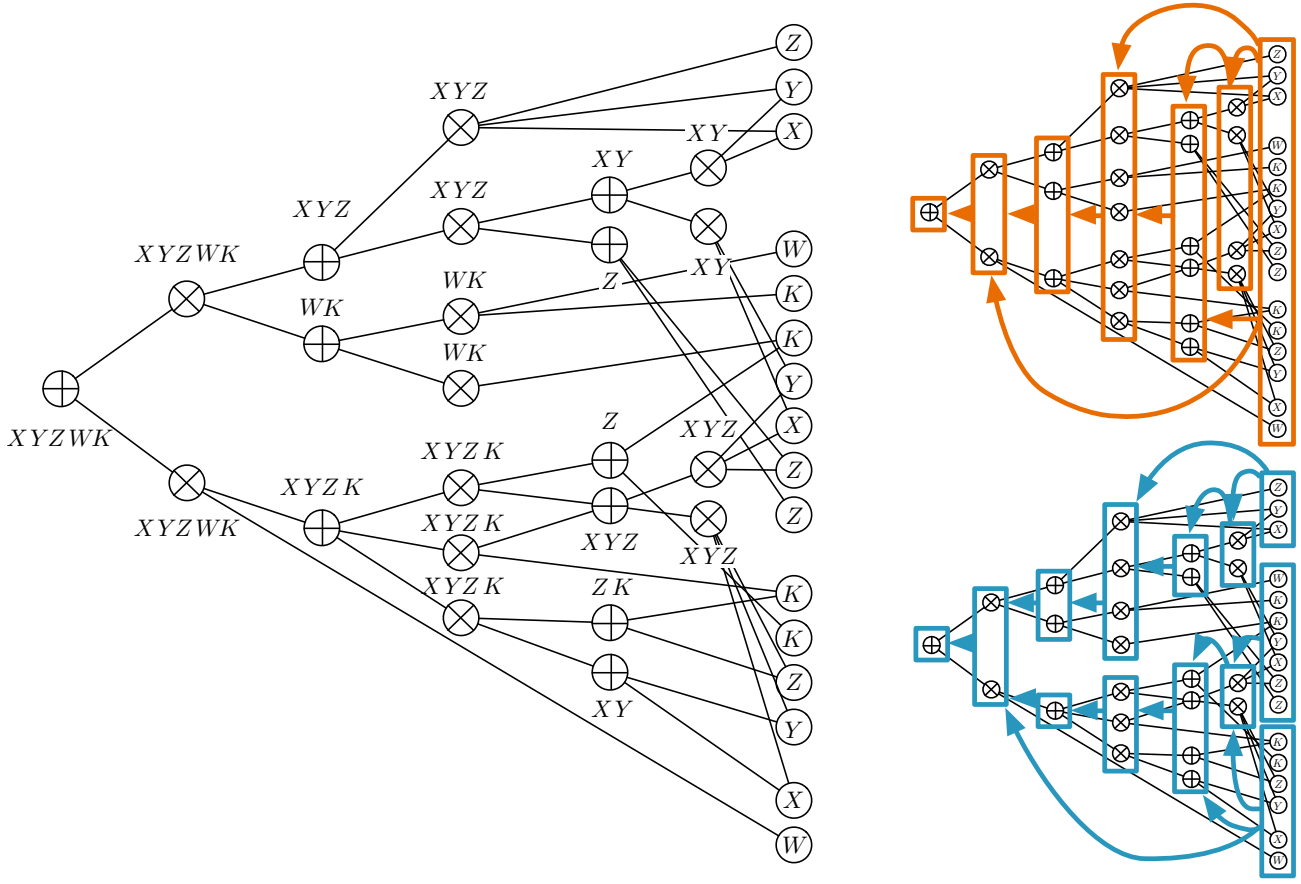
$$\mathbf{S}(\mathbf{x}) = \exp(\mathbf{P}\mathbf{x})$$

where $\mathbf{W} \in \mathbb{R}_{+}^{s \times r}$ and $\mathbf{P} \in \{0, 1\}^{s \times r}$ are the weight matrices:

$$\mathbf{W}_{(ij)} = \begin{cases} w_{ij} & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{P}_{(ij)} = \begin{cases} 1 & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

Vergari2016a.

SPNs as NNs (III)



SPNs as NNs (IV): filters

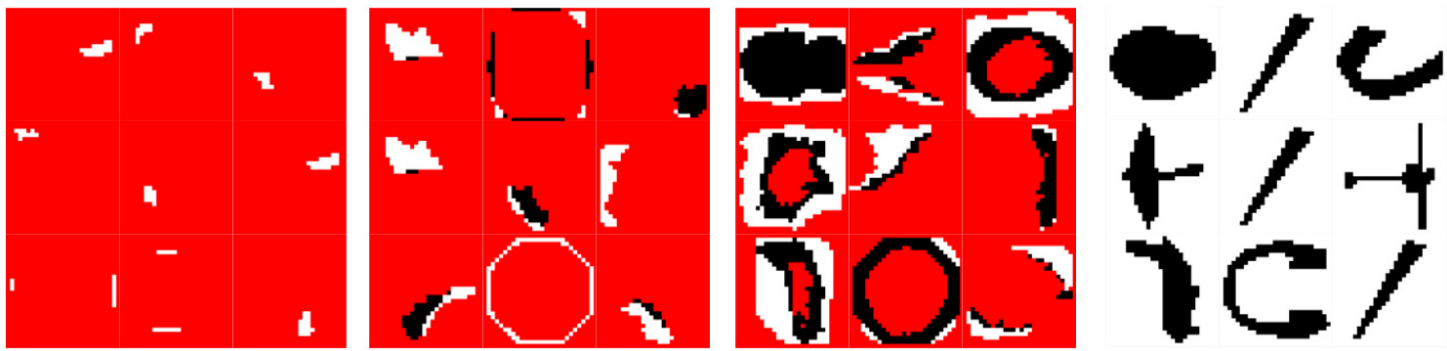
Learned features as images maximizing neuron activations [**Erhan2009**]:

$$\mathbf{x}^* = \underset{\mathbf{x}, ||\mathbf{x}||=\gamma}{\operatorname{argmax}} \, h_{ij}(\mathbf{x}; \boldsymbol{\theta}).$$

With SPNs, joint solution as an MPE assignment for all nodes (linear time):

$$\mathbf{x}^*_{|\mathsf{sc}(n)} = \underset{\mathbf{x}}{\operatorname{argmax}} \, S_n(\mathbf{x}_{|\mathsf{sc}(n)}; \mathbf{w})$$

.



$\rightarrow$ *scope length* ($|\mathsf{sc}(n)|$) correlates with feature abstraction level

---

Vergari2016a.

SPNs as BNs I

Zhao and Poupart

SPNs as BNs II

Peharz

Myths about SPNs

SPNs are PGMs. false

SPNs are convolutional NNs. false

SPNs

Learning

Learning SPNs

Parameter learning: estimate $\mathbf{w}$ from data considering an SPN as a latent RV model, or as a NN

Structure learning: build the network from data by assigning scores to tentative structures or by exploiting constraints

How to learn a “complete” SPN:

- handcrafted structure, then parameter learning [**Poon2011**; **Gens2012**]
- random structures, then parameter learning [**Rashwan2016**]
- structure learning, then parameter learning (fine tuning) [**Zhao2016a**]
- learn both weight and structure at the same time [**Gens2013**; **Rooshenas2014**; **Vergari2015**; **Adel2015**]...

Structure Learning

Score vs constraint based search. No closed form for likelihood scores, need heuristics [**Rooshenas2014**].
No need for it by exploiting the inner nodes probabilistic semantics

Learning graph vs tree structures: Easier to learn a tree SPN (sometimes SPT) with greedy approaches. Graph SPNs may be more compact and expressive efficient.

Top-down vs bottom-up approaches: iteratively cluster data matrix (top-down) or start by the marginal RVs (bottom-up)

LearnSPN [**Gens2013**] is a ***greedy***, ***top down***, ***constraint based*** learner for ***tree*** SPNs
$\rightarrow$ First principled top-down learner, inspired many algorithms and variations.
$\rightarrow$ Surprisingly simple and accurate.

LearnSPN (I)

Build a tree SPN by recursively split the data matrix:

- splitting columns into pairs by a greedy ***G Test***-based procedure with threshold ρ:

$$G(X_i, X_j) = 2 \sum_{x_i \sim X_i} \sum_{x_j \sim X_j} c(x_i, x_j) \cdot \log \frac{c(x_i, x_j) \cdot |T|}{c(x_i) c(x_j)}$$

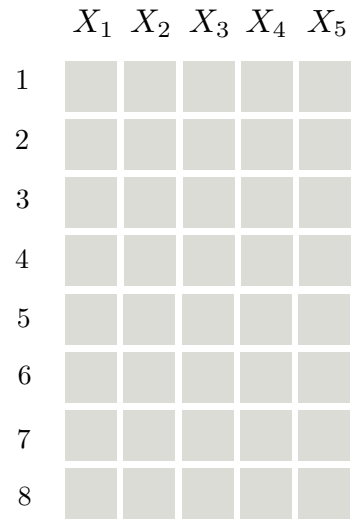
- clustering instances into $|C|$ sets with ***online Hard-EM*** with cluster penalty λ:

$$Pr(\mathbf{X}) = \sum_{C_i \in \mathbf{C}} \prod_{X_j \in \mathbf{X}} Pr(X_j | C_i) Pr(C_i)$$

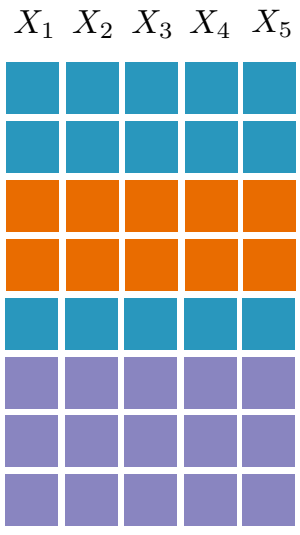
  weights are estimated as cluster proportions
- if there are less than m instances, put a ***naive factorization*** over leaves
- each univariate distribution get ***ML estimation*** smoothed by α

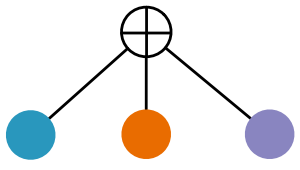
Hyperparameter space: $\{\rho, \lambda, m, \alpha\}$.

LearnSPN (II)

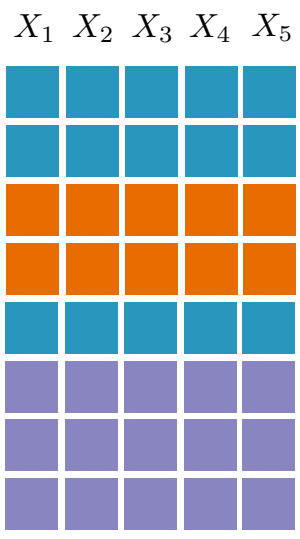


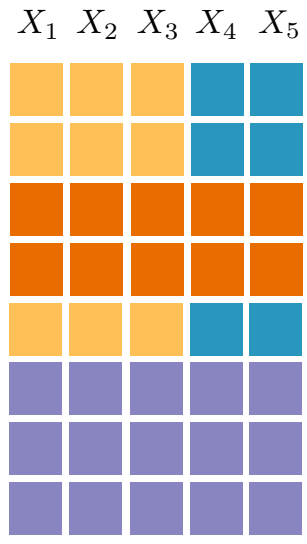
LearnSPN (II)

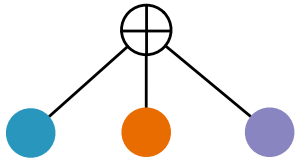


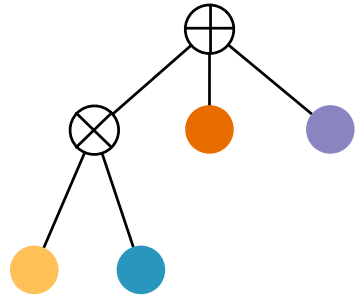


LearnSPN (II)

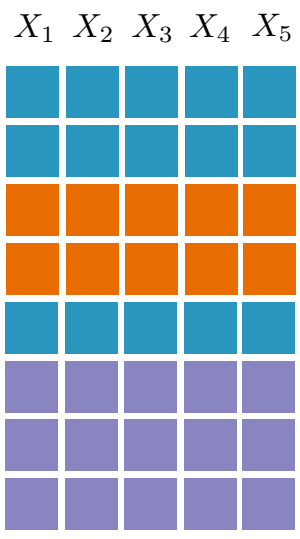


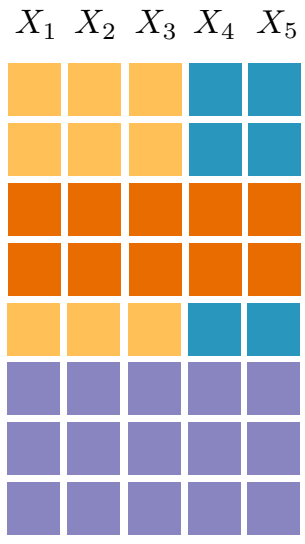


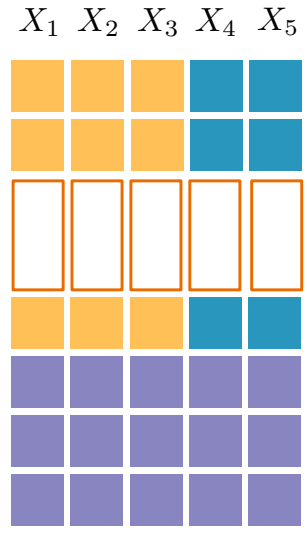


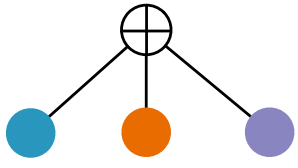


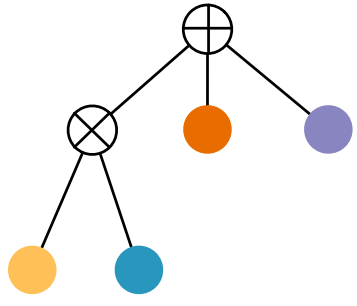
LearnSPN (II)

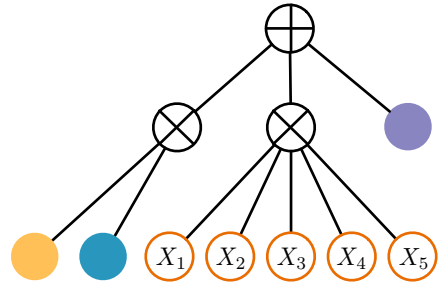












Tweaking LearnSPN

LearnSPN performs two interleaved greedy ***hierarchical divisive clustering*** processes (co-clutering on the data matrix).

Fast and simple. But both processes never look back and are committed to the choices they take $\rightarrow$ *slow down the two processes*

Online EM does not need to specify the number of clusters k in advance. But overcomplex structures are learned by exploding the number of sum node children $\rightarrow$ *look for deeper networks*

Tractable leaf estimation. But too strong naive factorization independence assumptions, hard to regularize $\rightarrow$ *learn tree distributions as leaves*

ML estimations are effective. But they are not robust to noise, they can overfit the training set easily $\rightarrow$ *learn bagged sum nodes*

Why Structure Quality Matters

Tractable inference is guaranteed *if the network size is polynomial* in $|\mathbf{X}|$.

Network ***size*** influences inference complexity: smaller networks, faster inference!
$\rightarrow$ Comparing network sizes is better than comparing inference times

Network ***depth*** influences *expressive efficiency [***Martens2014***; ***Zhao2015***]*.

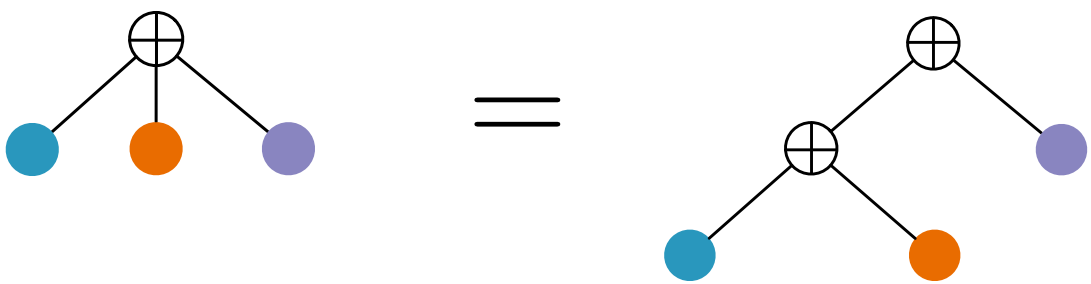
Structural simplicity as a bias: overcomplex networks may not generalize well.

Structure quality desiderata: **smaller** but **accurate**, ***deeper*** but not wider, SPNs.

LearnSPN-b

Observation: each clustering process benefits from the other one improvements/highly suffers from other's mistakes.

Idea: slow them down the processes by limiting the number of nodes to split to the minimum. LearnSPN-b, binary splitting $k = 2$.
$\rightarrow$ one hyperparameter less, λ.
$\rightarrow$ not committing to complex structures too early
$\rightarrow$ reducing node out fan increases the depth
$\rightarrow$ same expressive power as LearnSPN structures
$\rightarrow$ statistically same (or better) accuracy, smaller networks



LearnSPN-b: depth VS size

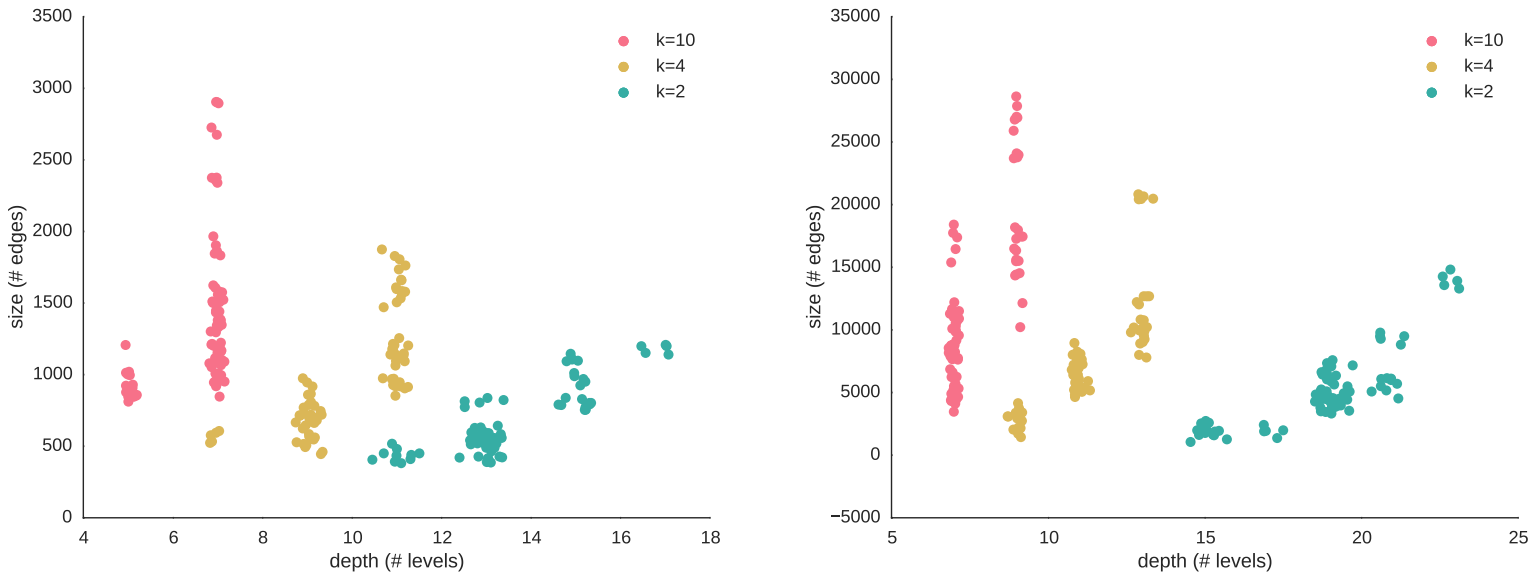


Figure : Network sizes VS depths while varying the max number of sum node children splits ($k \in \{10, 4, 2\}$). Each dot is an experiment in the grid search hyperparameter space performed by LearnSPN-b on NLTCS (left) and Plants (right).

LearnSPN-b: best ll VS size

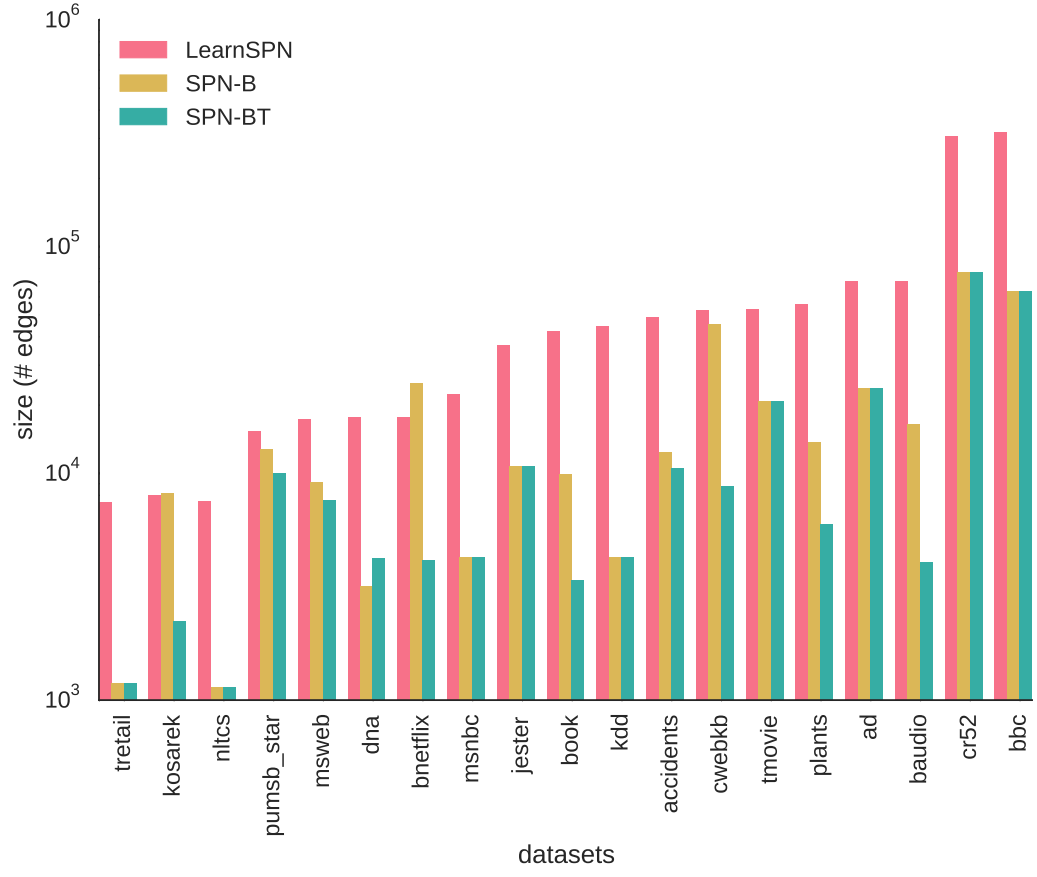


Figure : Comparing network sizes for the networks scoring the best log-likelihoods in the grid search as obtained by LearnSPN, LearnSPN-b and LearnSPN-bT for each dataset.

Other variations on LearnSPN

ACs modeling leaves by performing a greedy score search. ***ID-SPN*** best log-likelihood learner (but lots of hyperparameters).
Freely available in the Libra[2] toolkit. [**Rooshenas2014**]

Looking for **correlations instead of independencies** via matrix factorizations.
Splitting matrix rows and columns at the same time: SPN-SVD.
It can cope with continuous data [**Adel2015**]

Post-learning **mergining sub-SPNs** that model "similar" distributions.
Reducing network sizes [**Rahman2016**].

Learning Relational SPNs on **first order data** represented in Tractable Markov Logic (TML), LearnRSPN [**Nath2015**].

[2] http://libra.cs.uoregon.edu/

Other Tendencies in Structure Learning

Learning deterministic structures which enable closed form log-likelihood and weight estimation.
Selective SPNs, enabling efficient Stochastic Local Search [**Peharz2014b**].
Mixing latent and deterministic mixtures as sum nodes (a Cutset Network is an SPN!). [**Rahman2016**]

Learning DAGs structures instead of trees.
Substituting sub-structures with more complex ones by cloning mixtures. [**Dennis2015**]

Template learning for sequence models. Stochastic local search over well defined constrained structures. Dynamic SPNs [**Melibari2016c**]$\longrightarrow$ *PGM'16!*

Parameter Learning

Non convex optimization, solvable with iterative methods like: GD, EM,...and their online variants

Use backpropagation ($\rightarrow$ differential approach) to compute $\nabla_{\mathbf{w}} S(\mathbf{x})$

Hard/Soft gradients

Some issues:

- vanishing gradients
- hyperparameter choices

“Hard” gradients

Hard/Soft Parameter Learning

	soft	
Generative Gradient	$\frac{\partial S(\mathbf{x})}{\partial w_{pc}} = S_c(\mathbf{x}) \frac{\partial S(\mathbf{x})}{\partial S_p(\mathbf{x})}$	
Discriminative Gradient	$\frac{1}{S(\mathbf{y}\|\mathbf{x})} \frac{\partial S(\mathbf{y}\|\mathbf{x})}{\partial w_{pc}} - \frac{1}{S(*\|\mathbf{x})} \frac{\partial S(*\|\mathbf{x})}{\partial w_{pc}}$	$\underline{\sharp\{w_{pc} \in \text{MPE}-}$
Generative Posterior	$p(H_p = c\|\mathbf{x}) \propto \frac{1}{S(\mathbf{x})} \frac{\partial S(\mathbf{x})}{\partial S_p(\mathbf{x})} S_c(\mathbf{x}) w_{pc}$	$p(H_p$

Bayesian Parameter Learning

Parameter Learning VS LearnSPN

	LearnSPN[3]	LearnSPN-b[4]	CVB-SPN[5]	OBMM[6]	SGD[7]	EM[8]	SEG[9]
NLTCS	-6.11	-6.05	-6.08	-6.07	-8.76	-6.31	-6.85
MSNBC	-6.11	-6.04	-6.29	-6.03	-6.81	-6.64	-6.74
KDDCup2k	-2.18	-2.14	-2.14	-2.14	44.53	-2.20	-2.34
Plants	-12.98	-12.81	-12.86	-15.14	-21.50	-17.68	-33.47
Audio	-40.50	-40.57	-40.36	-40.70	-49.35	-42.55	-46.31
Jester	-53.48	-53.53	-54.26	-53.86	63.89	-54.26	-59.48
Netflix	-57.33	-57.73	-60.69	-57.99	64.27	-59.35	-64.48
Accidents	-30.04	-29.34	-29.55	-42.66	53.69	-43.54	-45.59
Retail	-11.04	-10.94	-10.91	-11.42	-97.11	-11.42	-14.94
Pumsb-star	-24.78	-23.31	-25.93	-45.27	-128.48	-46.54	-51.84
DNA	-82.52	-81.91	-86.73	-99.61	-100.70	-100.10	-105.25
Kosarek	-10.99	-10.72	-10.70	-11.22	34.64	-11.87	-17.71
MSWeb	-10.25	-9.83	-9.89	-11.33	-59.63	-11.36	-20.69
Book	-35.89	-34.30	-34.44	-35.55	-249.28	-36.13	-42.95
EachMovie	-52.49	-51.36	-52.63	-59.50	-227.05	-64.76	-84.82
WebKB	-158.20	-154.28	-161.46	-165.57	-338.01	-169.64	-179.34
Reuters-52	-85.07	-83.34	-85.45	-108.01	-407.96	-108.10	-108.42
20-Newsgrp	-155.93	-152.85	-155.61	-158.01	-312.12	-160.41	-167.89
BBC	-250.69	-247.30	-251.23	-275.43	-462.96	-274.82	-276.97
Ad	-19.73	-16.23	-19.00	-63.81	-638.43	-63.83	-64.11

[3] **Gens2013**.

[4]

Parameter Learning

Why learning parameters

Representation Learning

Extracting Embeddings

Exploiting SPNs as feature extractors

Given an SPN S, generate a dense vector

$$\mathbf{e}^i = f_S(\mathbf{x}^i)$$

Issues with SPNs as NNs:

- layer-wise extraction may be arbitrary
- power law distribution of nodes by scopes
- scope lengths as proxy for feature abstraction levels (see filter visualizations)

How to extract embeddings?

- from node outputs (one query)
  - all (non-leaf) nodes
  - sum/product nodes only
  - with a certain scope length
  - aggregating node outputs by scope
- from several queries evaluations (root outputs)

Supervised classification

Experimental settings

Embeddings (I)

Table : Test set accuracy scores for the embeddings extracted with the best SPN, RBM models and with the baseline LR model on all datasets. Bold values denote significantly better scores than all the others for a dataset.

	LR	SPN-I	SPN-II	SPN-III	RBM-5h	RBM-1k	RBM-5k
REC	69.28	77.31	**97.77**	97.66	94.22	96.10	96.36
CON	53.48	67.48	78.31	**84.69**	67.55	75.37	79.15
OCR	75.58	82.60	**89.95**	**89.94**	86.07	87.96	88.76
CAL	62.67	59.17	65.19	66.62	67.36	**68.88**	67.71
BMN	90.62	95.15	**97.66**	97.59	96.09	96.80	97.47

Embeddings (II)

Table : Test set accuracy scores for the embeddings extracted with SPN models and filtered by node type. Results for SPN-III embeddings filtered by Small , Medium and Large scope lengths are reported in columns 8-10. Bold values denote significantly better scores than all the others. ▲ indicates a better score than an RBM embedding with greater or equal size. ▽ indicates worse scores than an RBM embedding with smaller or equal size.

	SPN-I		SPN-II		SPN-III		SPN-III		
	sum	prod	sum	prod	sum	prod	S	M	L
REC	72.46	62.25	**98.03**$^{\blacktriangle}$	97.06$^{\blacktriangle}$	**98.00**$^{\blacktriangle}$	97.04$^{\blacktriangle}$	88.73	**98.45**$^{\blacktriangle}$	93.91
CON	62.36	64.03	77.13$^{\blacktriangle}$	76.07$^{\blacktriangle}$	**83.59**$^{\blacktriangle}$	82.06$^{\blacktriangle}$	70.51$^{\triangledown}$	77.18	**83.32**$^{\blacktriangle}$
OCR	74.19	81.58	89.73$^{\blacktriangle}$	88.78$^{\blacktriangle}$	**90.02**$^{\blacktriangle}$	89.32	87.22$^{\triangledown}$	**89.29**$^{\blacktriangle}$	88.19$^{\blacktriangle}$
CAL	38.19	56.95	62.64	64.80	**66.58**$^{\triangledown}$	66.40$^{\triangledown}$	63.37$^{\triangledown}$	**66.23**$^{\triangledown}$	66.10
BMN	93.50	94.75	97.67	96.90$^{\triangledown}$	**97.80**	97.20$^{\triangledown}$	96.02$^{\triangledown}$	**97.42**$^{\triangledown}$	97.38

Embeddings (III)

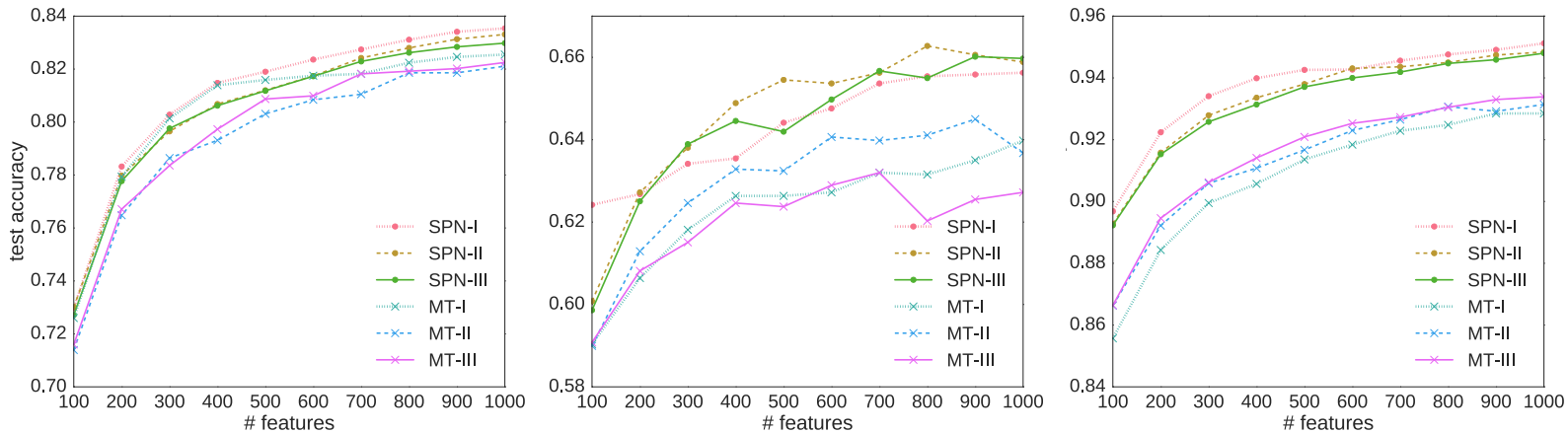
Table : Test set accuracy scores for the embeddings extracted with SPN models by aggregating node outputs with the same scope. Results for when leaves are not counted in the aggregarion (no-leaves columns) are reported alongside the case in which they are considered (leaves columns). Bold values denote significantly better scores than all the others for each dataset. ▲ indicates a better score than an RBM embedding with greater or equal size. ▽ indicates worse scores than an RBM embedding with smaller or equal size.

	SPN-I		SPN-II		SPN-III	
	no-leaves	leaves	no-leaves	leaves	no-leaves	leaves
REC	72.47	$75.92^{\triangledown}$	$\mathbf{97.94}^{\blacktriangle}$	$\mathbf{97.99}^{\blacktriangle}$	$\mathbf{97.94}^{\blacktriangle}$	$\mathbf{98.02}^{\blacktriangle}$
CON	62.35	$66.49^{\triangledown}$	$77.21^{\blacktriangle}$	78.05	$\mathbf{83.52}^{\blacktriangle}$	$\mathbf{83.84}^{\blacktriangle}$
OCR	74.32	81.85	$89.71^{\blacktriangle}$	$89.68^{\blacktriangle}$	$\mathbf{89.90}^{\blacktriangle}$	$\mathbf{89.91}^{\blacktriangle}$
CAL	38.10	$63.19^{\triangledown}$	62.59	$62.76^{\triangledown}$	$\mathbf{66.49}^{\triangledown}$	$\mathbf{66.58}^{\triangledown}$
BMN	93.51	$94.83^{\triangledown}$	$97.64^{\blacktriangle}$	$97.62^{\blacktriangle}$	**97.80**	**97.80**

Random Marginal Queries

Generate embeddings by asking several random queries to a black box density estimator.

Eg. marginals: $e^i_j = P_\theta(\mathbf{Q}_j = \mathbf{x}^i_{\mathbf{Q}_j})$, according to estimator θ where $\mathbf{Q}_j \subseteq \mathbf{X}, j = \ldots, k$.



Vergari2016a.

Encoding/Decoding Embeddings

MPN as autoencoders[10].

[10] Vergari et al. Encoding and Decoding Representations with Sum-Product Networks, 2016, to appear

Applications

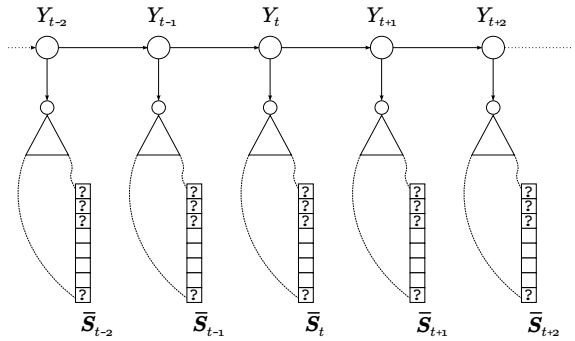
Applications I: computer vision

Applications II: language modeling

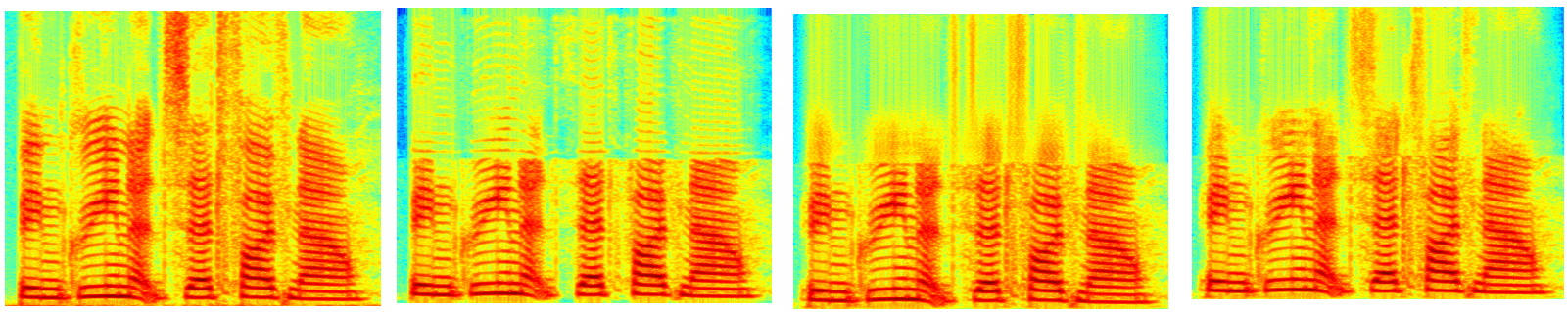
Applications III: activity recognition

Applications IV: speech

SPNs to model the joint pdf of observed RVs in HMMs (HMM-SPNs).



State-of-the-art high frequency reconstruction (MPE inference)



(a) Original full bandwidth (b) Reconstruction HMM-LP (c) Reconstruction HMM-GMM (d) Reconstruction HMM-SPN

Peharz2014a.

Trends & What to do next

Scalable structure learning Continuous RVs structure learning
End-to-end training with hybrid NN architectures

References

awesome-spn

A curated and structured list of resources about SPNs[11].
`https://github.com/arranger1044/awesome-spn`

[11] Inspired by the SPN page http://spn.cs.washington.edu/ at the Washington University