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

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

- for Pobabilistic Graphical Models (PGMs) there exist multi-purpose inference tools
  - the computational effort scales unproportional to the complexity of the graph
  - solution: using approximate inference

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

The need for SPNs

Why should you work on SPNs?

- exact tractable inference
- NN for which structure learning is easy

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

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

Scopes

Structural Properties

Inference

Complete evidence

Marginal inference

MPE inference

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:

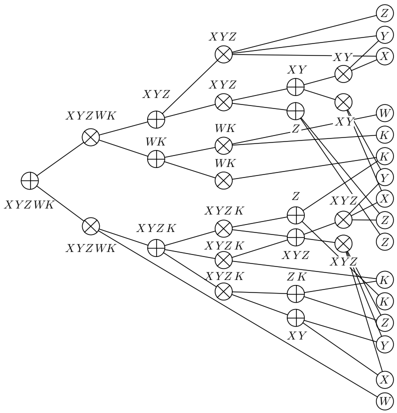
$$\mathbf{S}(\mathbf{x}) = \exp(\mathbf{P}\mathbf{x})$$

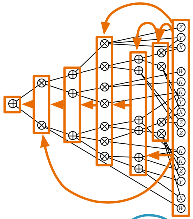
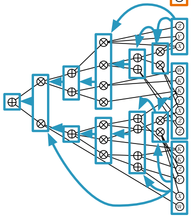
where $\mathbf{W} \in \mathbb{R}_{+}^{s \times r}$ and $\mathbf{P} \in \{0, 1\}^{s \times r}$ are the weight matrices:

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

Vergari2016a.

SPNs as NNs (III)



SPNs as NNs (IV): filters

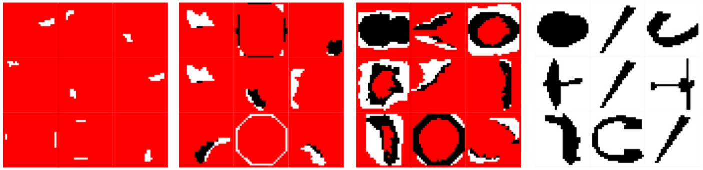
Learned features as images maximizing neuron activations [**Erhan2009**]:

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

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

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

.



$\rightarrow$ *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

Structure Learning

Structure matters

Alternatives:

- handcrafted structure, then weight learning [**Poon2011**]
- random structures, then weight learning [**Rashwan2016**]
- learned from data

Why Structure Quality Matters

Tractable inference is guaranteed *if the network size is polynomial* in # vars.

Smaller networks, faster inference (comparing network sizes is better than comparing inference times).

Deeper networks are possibly *more expressively efficient [***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 (I)

Build a tree-like 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:

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

  weights are estimated as cluster proportions

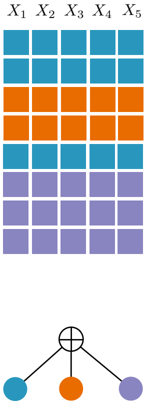
- if there are less than m instances, put a ***naive factorization*** over leaves
- each univariate distribution get ***ML estimation*** smoothed by α

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

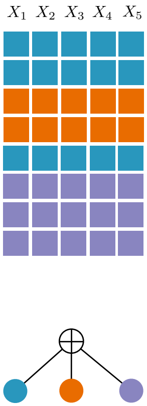
LearnSPN (II)

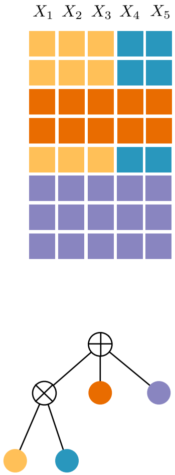
	X_1	X_2	X_3	X_4	X_5
1					
2					
3					
4					
5					
6					
7					
8					

LearnSPN (II)

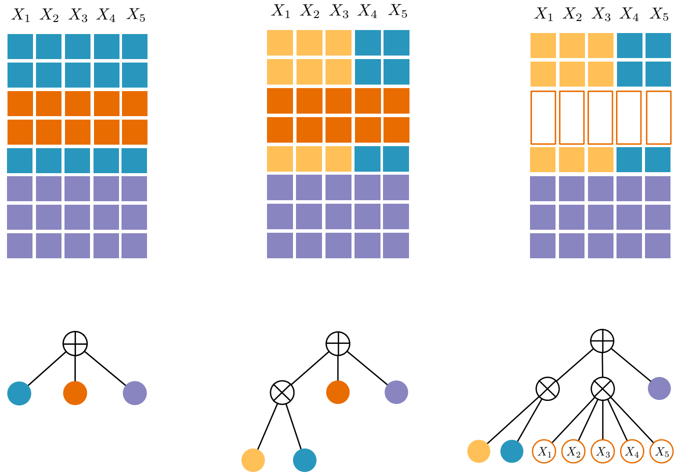


LearnSPN (II)





LearnSPN (II)



X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5

LearnSPN (III)

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.

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.

Tractable leaf estimation. But naive factorization independence assumptions may be too strong.

ML estimations are effective. But they are not robust to noise, they can overfit the training set easily.

LearnSPN

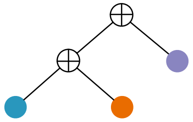
LearnSPN-b

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

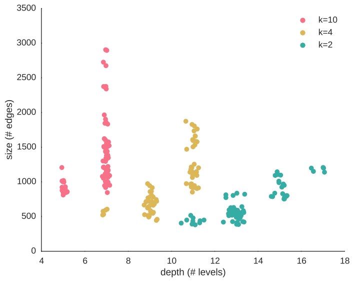
Idea: slowing down the processes by limiting the number of nodes to split into. SPN-B, variant of LearnSPN that uses EM for mixture modeling with $k = 2$ to cluster rows.
No need for λ anymore.

Objectives:

- not committing to complex structures too early
- same expressive power as LearnSPN
- reducing node out fan increases the depth
- same accuracy, smaller networks



LearnSPN-b: depth VS size



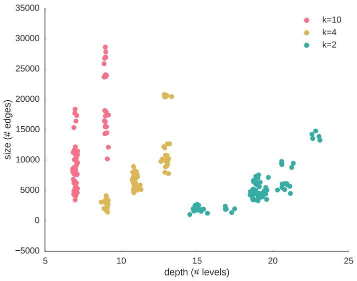


Figure : Comparing network sizes and depths while varying the max number of sum node children splits ($k \in \{10, 4, 2\}$). Each dot is an experiment in the grid search hyperparameter space performed by SPN-B on NLTCS (left) and Plants (right).

New Tendencies in Structure Learning

Pruning and compressing

Parameter Learning

Non convex optimization, solvable with iterative methods like: GD, EM,...

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

Hard/Soft gradients

Some issues:

- vanishing gradients
- hyperparameter choices

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

Bayesian Parameter Learning

Parameter Learning VS LearnSPN

	LearnSPN[2]	LearnSPN-b[3]	CVB-SPN[4]	OBMM[5]	SGD[6]	EM[7]	SEG[8]
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

[2] **Gens2013**.

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

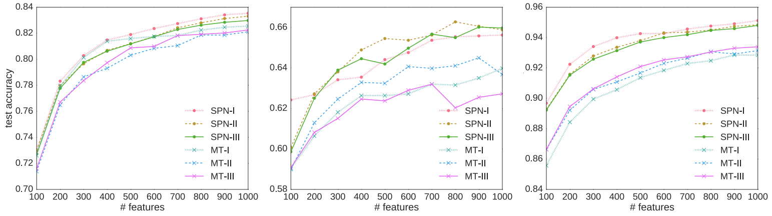
Embeddings (III)

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▽	**97.94**▲	**97.99**▲	**97.94**▲	**98.02**▲
CON	62.35	66.49▽	77.21▲	78.05	**83.52**▲	**83.84**▲
OCR	74.32	81.85	89.71▲	89.68▲	**89.90**▲	**89.91**▲
CAL	38.10	63.19▽	62.59	62.76▽	**66.49**▽	**66.58**▽
BMN	93.51	94.83▽	97.64▲	97.62▲	**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 $\mathbf{Q}_j \subseteq \mathbf{X}, j = \ \ldots, k$.



Encoding/Decoding Embeddings

MPN as autoencoders[9].

[9] 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

Peharz2014a.

Trends & What to do next

References

awesome-spn

A curated and structured list of resources about SPNs[10].
`https://github.com/arranger1044/awesome-spn`

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