Università degli Studi di Bari
Dipartimento di Informatica

LACAM
Machine Learning

Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning

Antonio Vergari, Nicola Di Mauro and Floriana Esposito

September 2, 2015

Introduction

The main task on ***Probabilistic Graphical Models*** (PGMs), *inference*, is untractable.

Tractable PGMs are recently gaining interest: tree based models [Meilă and Jordan 2000], Arithmetic Circuits (ACs) [Rooshenas and Lowd 2014], ***Sum-Product Networks*** (SPNs) [Poon and Domingos 2011].

Sum-Product Networks structure learning: ***LearnSPN***.

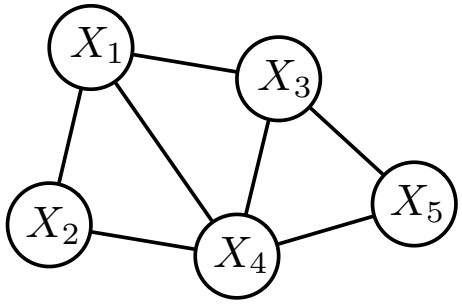
Improving LearnSPN in three ways:

I **simplifying** by limiting node splits

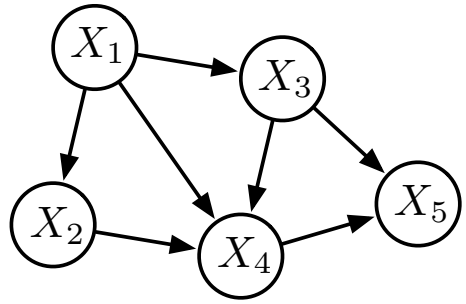
II **regularizing** by effective early stopping

III **strengthening** by model averaging

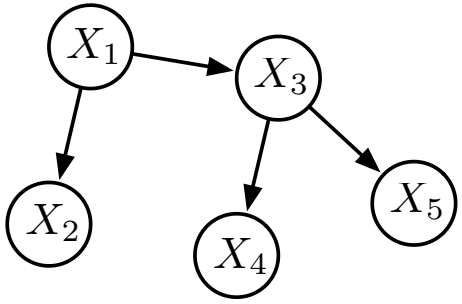
PGMs and Tractability



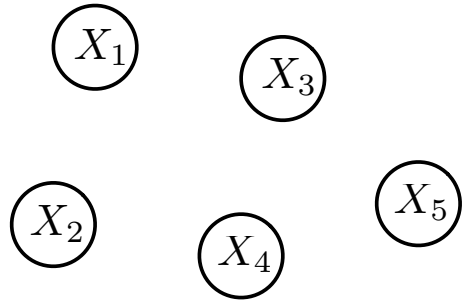
$P(\mathbf{X}) = \frac{1}{Z} \prod_c \phi_c(\mathbf{X}_c)$



$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i)$

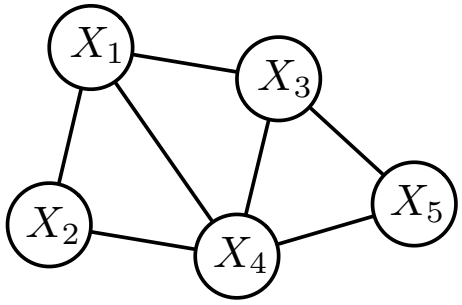


$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa_i)$

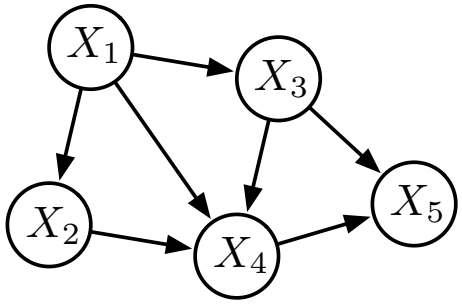


$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i)$

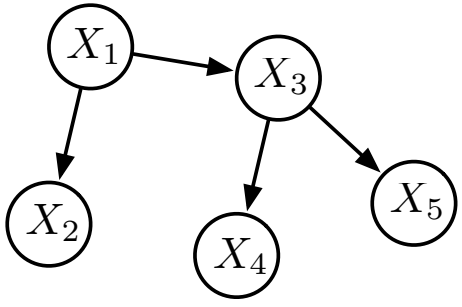
PGMs and Tractability



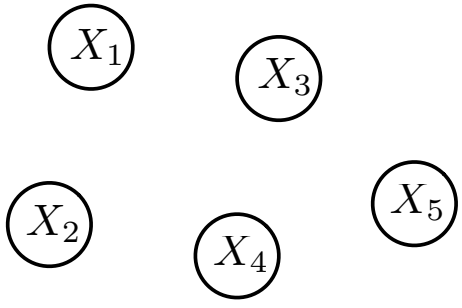
untractable



untractable



tractable

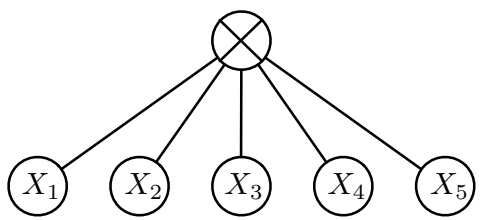


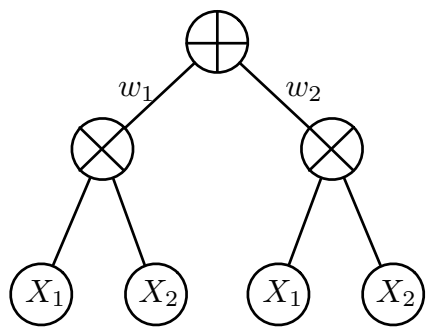
tractable

Sum-Product Networks (I)

SPNs are DAGs *compiling* the partition function of a joint pdf into a ***deep*** architecture of **sum** and **product** nodes.

Product nodes define factorizations over independent components, sum nodes weighted mixtures. Leaves are tractable univariate distributions.



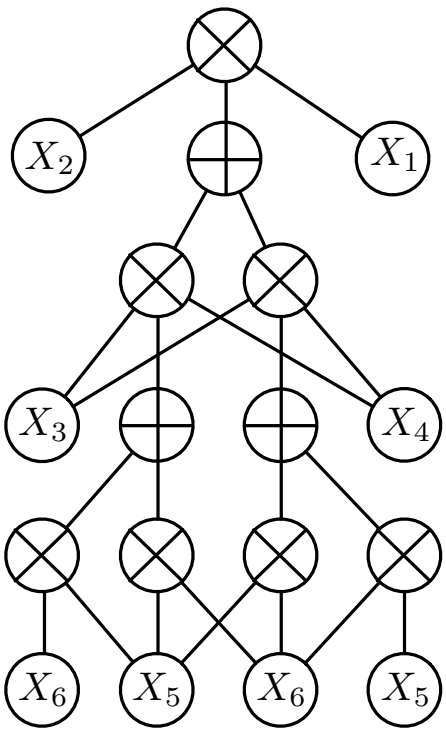


Products over nodes with different scopes (*decomposability*) and sums over nodes with same scopes (*completeness*) guarantee modeling a pdf (*validity*).

The ***size*** of the network is the number of *edges* in it.

The ***depth*** of the network is the number of alternated *layers*.

Sum-Product Networks (II)



Bottom-up evaluation of the network:

$$S_{X_i}(x_j) = P(X_i = x_j)$$

$$S_+(\mathbf{x}) = \sum_{i \in ch(+)} w_i S_i(\mathbf{x}), \quad S_\times(\mathbf{x}) = \prod_{i \in ch(\times)} S_i(\mathbf{x})$$

Inferences linear in the size of the network (# edges):

- $Z = S(*)$
- $P(\mathbf{e}) = S(\mathbf{e})/S(*)$
- $P(\mathbf{q}|\mathbf{e}) = \frac{P(\mathbf{q},\mathbf{e})}{P(\mathbf{e})} = \frac{S(\mathbf{q},\mathbf{e})}{S(\mathbf{e})}$
- $MPE(\mathbf{q},\mathbf{e}) = \max_{\mathbf{q}} P(\mathbf{q},\mathbf{e}) = S^{max}(\mathbf{e})$
  (by substituting sum nodes with *max* nodes)

How and Why Structure Learning

Fixed structures are hard to engineer and train (fully connected layers).

Automatic discovery of latent vars.

Constraint-based search formulation. Discover hidden variables for sum node mixtures and independences for product node components:

- ⊕ greedy top-down: KMeans on features *[Dennis and Ventura 2012]*; alternating clustering on instances and independence tests on features, **LearnSPN** *[Gens and Domingos 2013]*
- ⊕ greedy bottom up: merging feature regions by a *Bayesian-Dirichlet independence test*, and reducing edges by maximizing MI *[Peharz, Geiger, and Pernkopf 2013]*
- ⊕ **ID-SPN**: turning LearnSPN in log-likelihood guided expansion of sub-networks approximated by Arithmetic Circuits *[Rooshenas and Lowd 2014]*

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 [Martens and Medabalimi 2014; Zhao, Melibari, and Poupart 2015]*.

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:

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

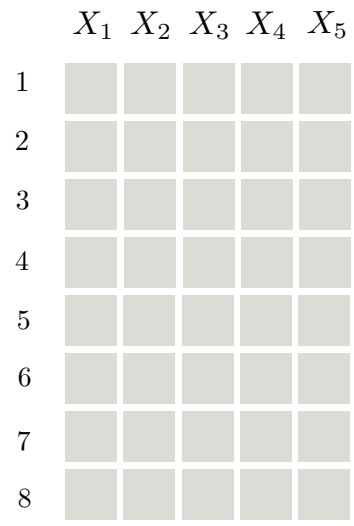
- clustering instances into $|C|$ sets with ***online Hard-EM*** with cluster penalty λ:

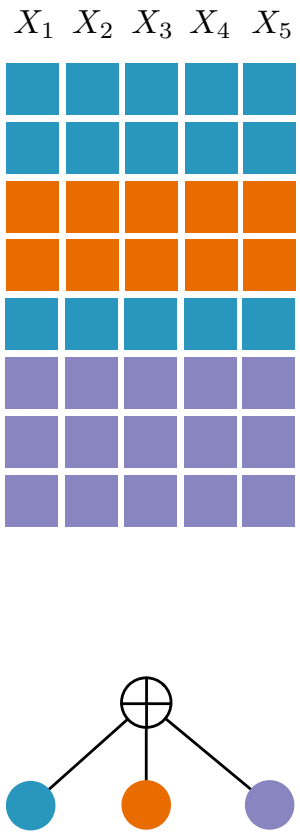
$$Pr(\mathbf{X}) = \sum_{C_i \in \mathbf{C}} \prod_{X_j \in \mathbf{X}} Pr(X_j | C_i) Pr(C_i)$$

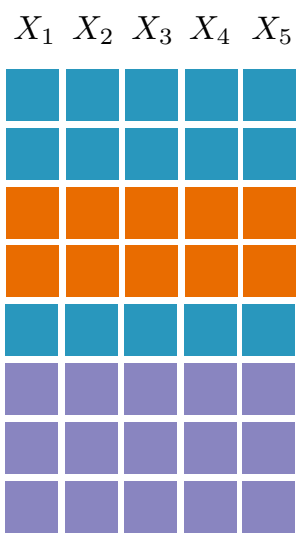
  weights are estimated as cluster proportions

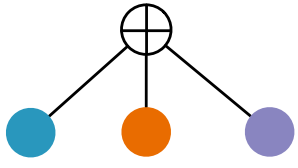
- if there are less than m instances, put a ***naive factorization*** over leaves
- each univariate distribution get ***ML estimation*** smoothed by α

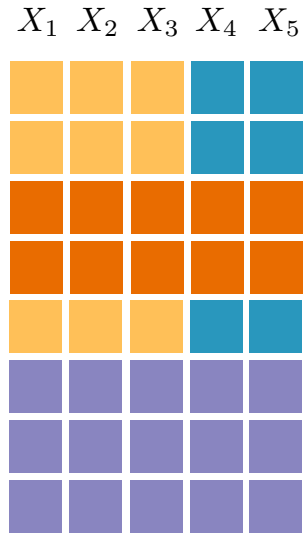
Hyperparameter space: $\{\rho, \lambda, m, \alpha\}$.

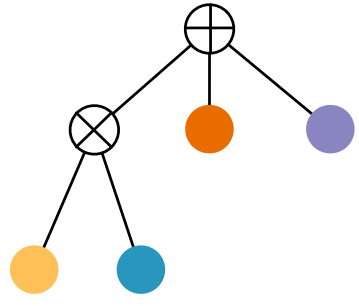

X_1 X_2 X_3 X_4 X_5
1
2
3
4
5
6
7
8

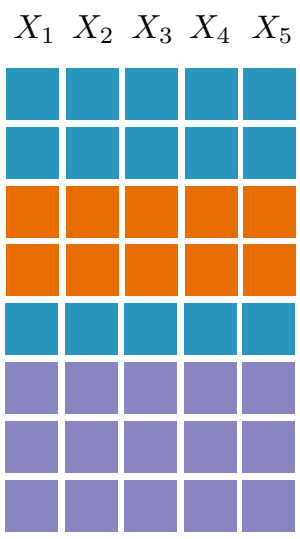

X_1 X_2 X_3 X_4 X_5

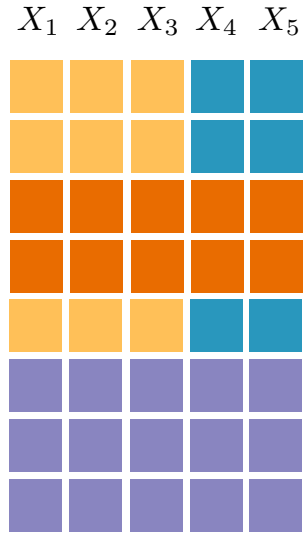

X_1 X_2 X_3 X_4 X_5

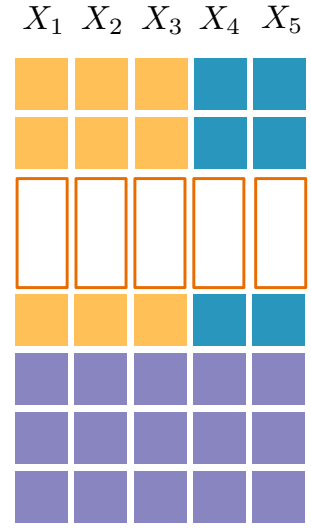


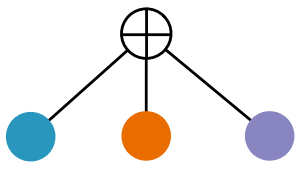

X_1 X_2 X_3 X_4 X_5

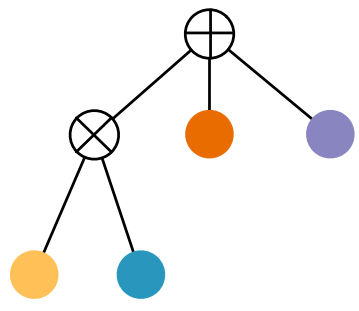


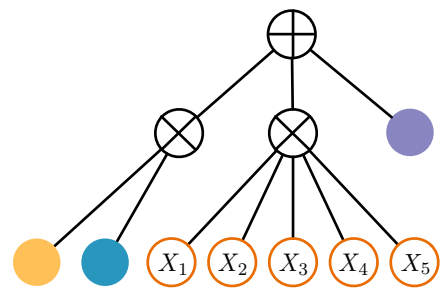

X_1 X_2 X_3 X_4 X_5


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 (II)

LearSPN 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.

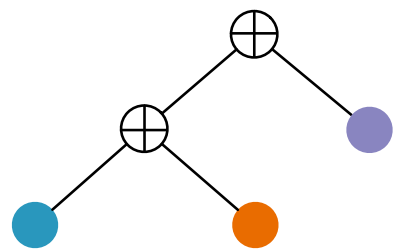
Simplifying by limiting node splits

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



SPN-B: depth VS size

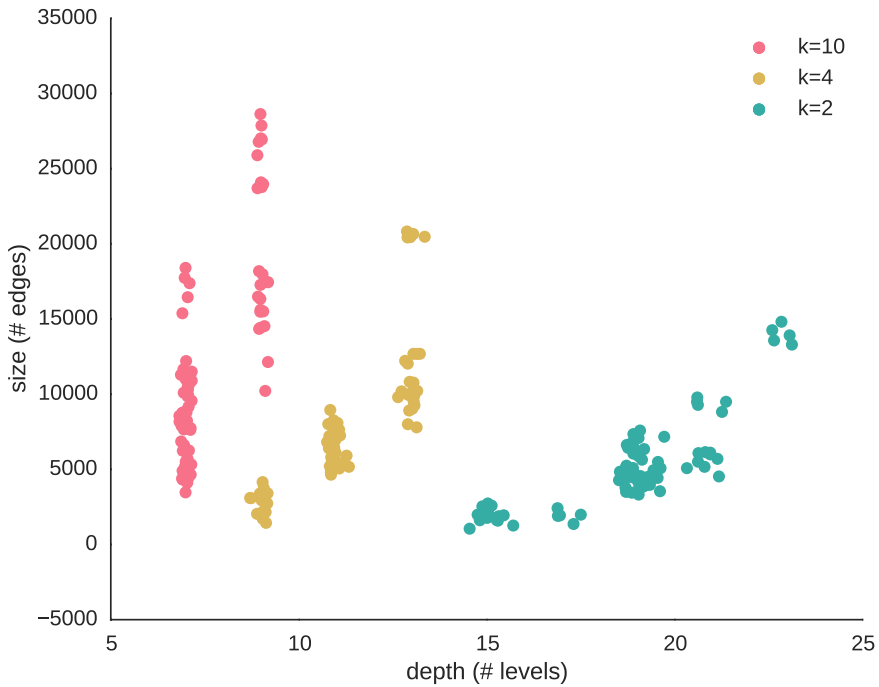


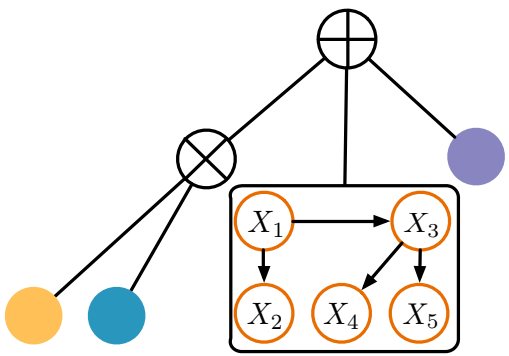
Figure : Comparing network sizes and depths while varying the max number of sum node children splits ($k \in \{10, 4, 2\}$). Each dot is an experiment in the grid search hyperparameter space performed by SPN-B on the dataset Plants.

Regularizing by effective early stopping

LearnSPN regularization is governed by α and m, however can be very ineffective:

- ⊕ naive factorizations are too strong assumptions
- ⊕ best likelihood structures prefer smaller values for m to get accurate naive factorizations

Idea: substituting naive factorizations with Bayesian trees as leaf distributions $P(\mathbf{X}) = \prod_j P(X_j|Pa_j)$:



- ⊕ learnable with Chow-Liu algorithm
- ⊕ still tractable (linear) multivariate distributions for marginals, conditionals and MPE
- ⊕ same or higher accuracy
- ⊕ less complex structures for larger values of m

SPN-BT: preserving likelihoods

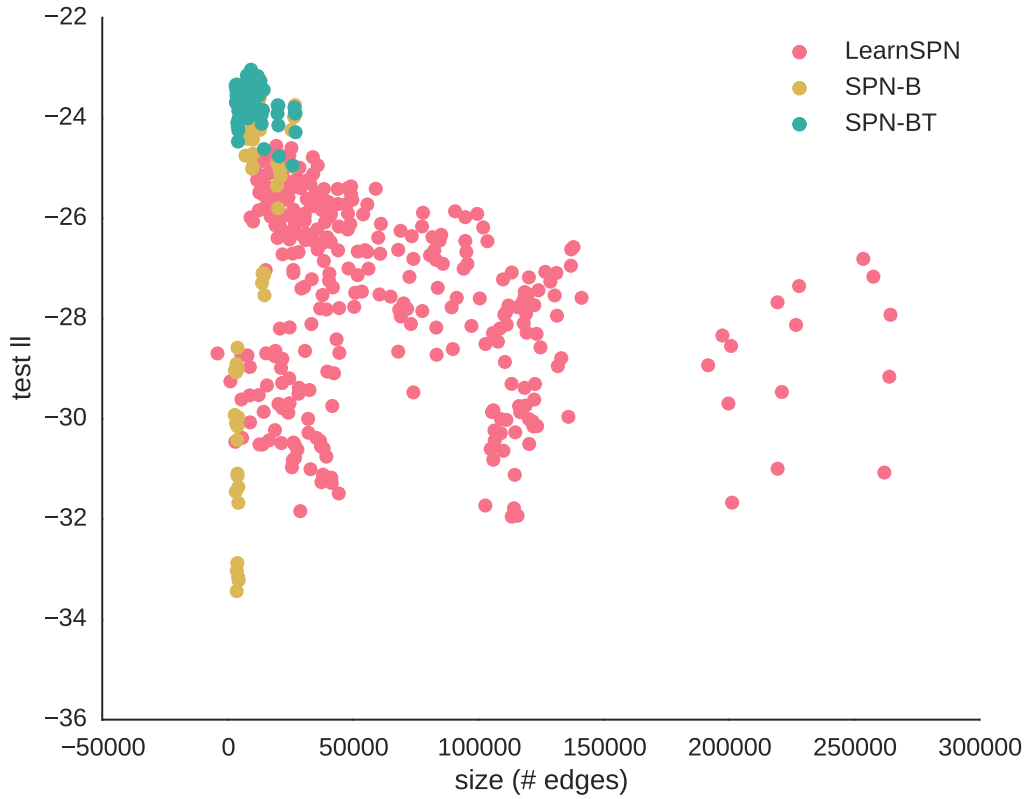


Figure : Comparing the network sizes against the average test log-likelihood obtained by LearnSPN, SPN-B and SPN-BT. Each dot is an experiment in the grid search performed for the dataset Pumsb-star.

SPN-BT: effective early stopping

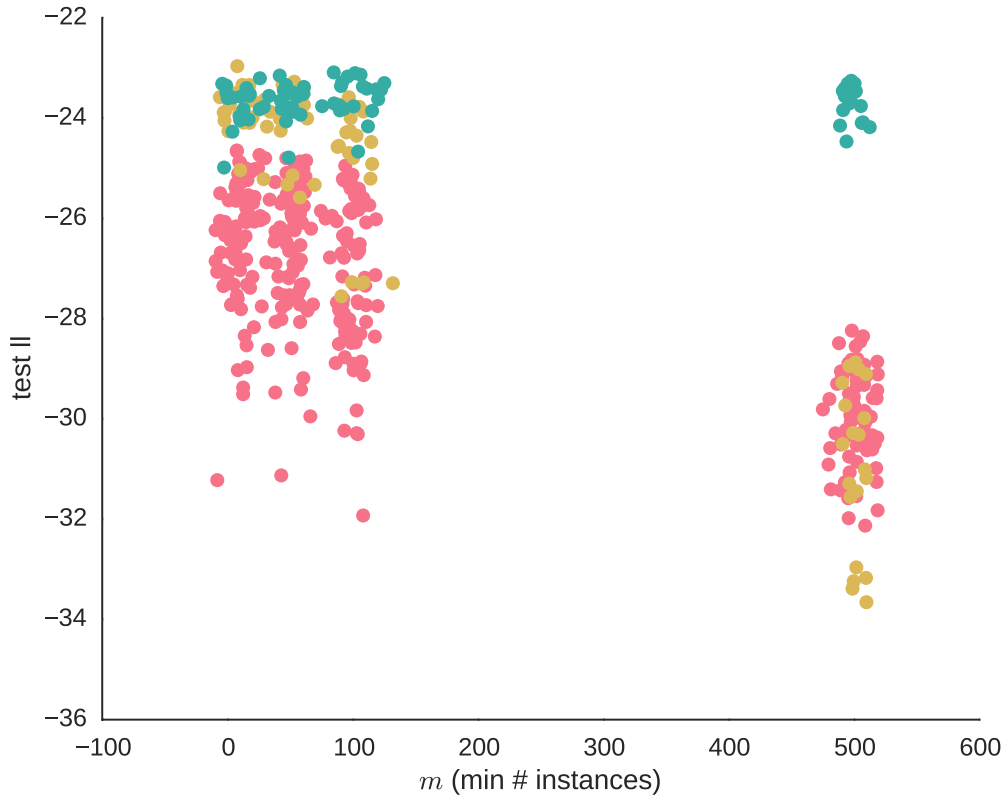


Figure : Comparing the values for m (10, 50, 100, 500) against the average test log-likelihood obtained by LearnSPN, SPN-B and SPN-BT. Each dot is an experiment in the grid search performed for the dataset Pumsb-star.

Strengthening by model averaging

Interpreting sum nodes as *general additive estimators*. Leveraging classic statistical tools to learn them: ***bagging***.

Draw k bootstrapped samples from the data, then grow an SPN S_{B_i} on each of them. Join them into a single SPN $\hat{S}$ with a sum node:

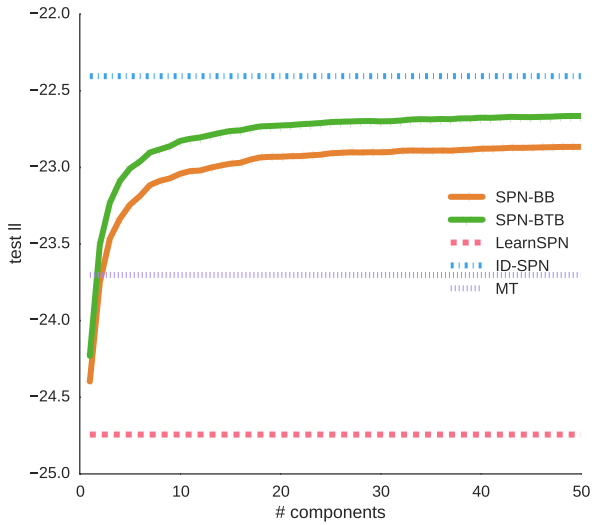
$$\hat{S} = \sum_{i=1}^{k} \frac{1}{k} S_{B_i}$$

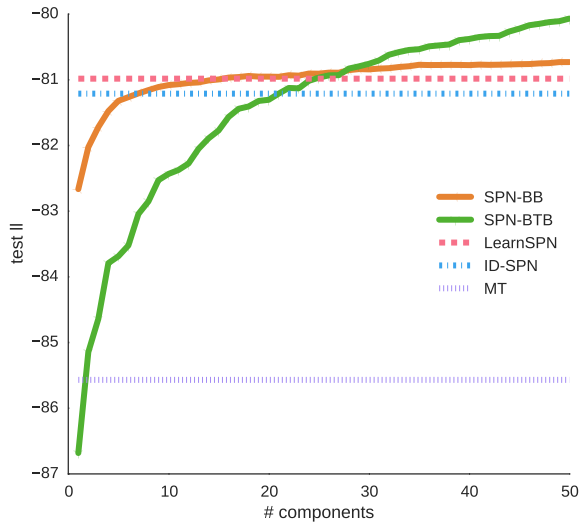
More robustness and less variance in the model.

Exponential number of nodes if done for each sum node (bootstrapping only at the root).

Two variants in the experiments: SPN-BB and SPN-BTB, whether Chow-Liu trees are employed or not.

Bagging exp


−22.0
−22.5
−23.0
−23.5
−24.0
−24.5
−25.0
test ll
0
10
20
30
40
50
components
SPN-BB
SPN-BTB
LearnSPN
ID-SPN
MT


−80
−81
−82
−83
−84
−85
−86
−87
test ll
0
10
20
30
40
50
components
SPN-BB
SPN-BTB
LearnSPN
ID-SPN
MT

Experimental setting

Classical setting for ***generative*** PGMs structure learning [Gens and Domingos 2013]:

- 19 binary datasets from classification, recommendation, frequent pattern mining...[Lowd and Davis 2010] [Haaren and Davis 2012]
- Training 75% Validation 10% Test 15% splits (no cv)

Observing both structure quality parameters and the network accuracy:

- test set ***average log-likelihood***
- network ***size*** (# edges) and network ***depth*** (# levels)

We perform a model selection via a *grid search* in the same parameter space for LearnSPN, SPN-B, SPN-BT:

- $\lambda \in \{0.2, 0.4, 0.6, 0.8\}$,
- $m \in \{1, 50, 100, 500\}$,
- $\rho \in \{5, 10, 15, 20\}$,
- $\alpha \in \{0.1, 0.2, 0.5, 1.0, 2.0\}$.

Up to 50 components and best grid parameters for SPN-BB and SPN-BTB.
Comparing ll against LearnSPN, ID-SPN and MT [Meilă and Jordan 2000]

Network sizes

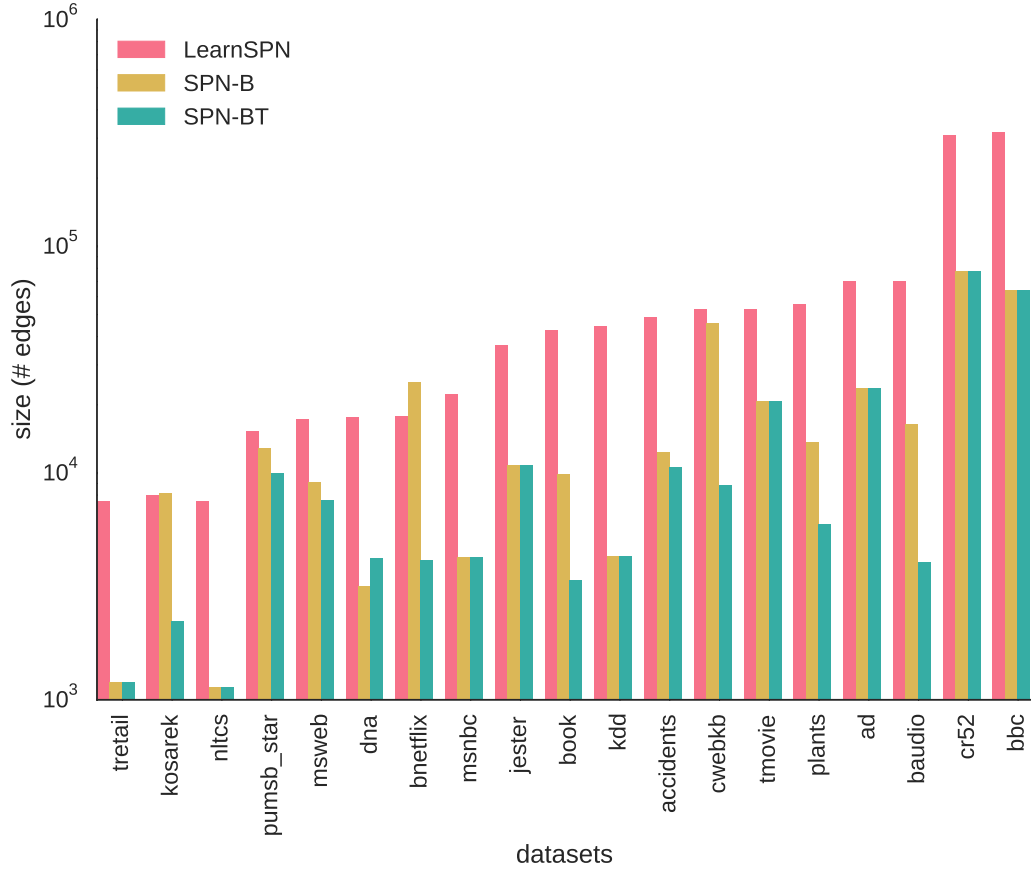


Figure : Comparing network sizes for the networks scoring the best log-likelihoods in the grid search as obtained by LearnSPN, SPN-B and SPN-BT for each dataset.

Network depths

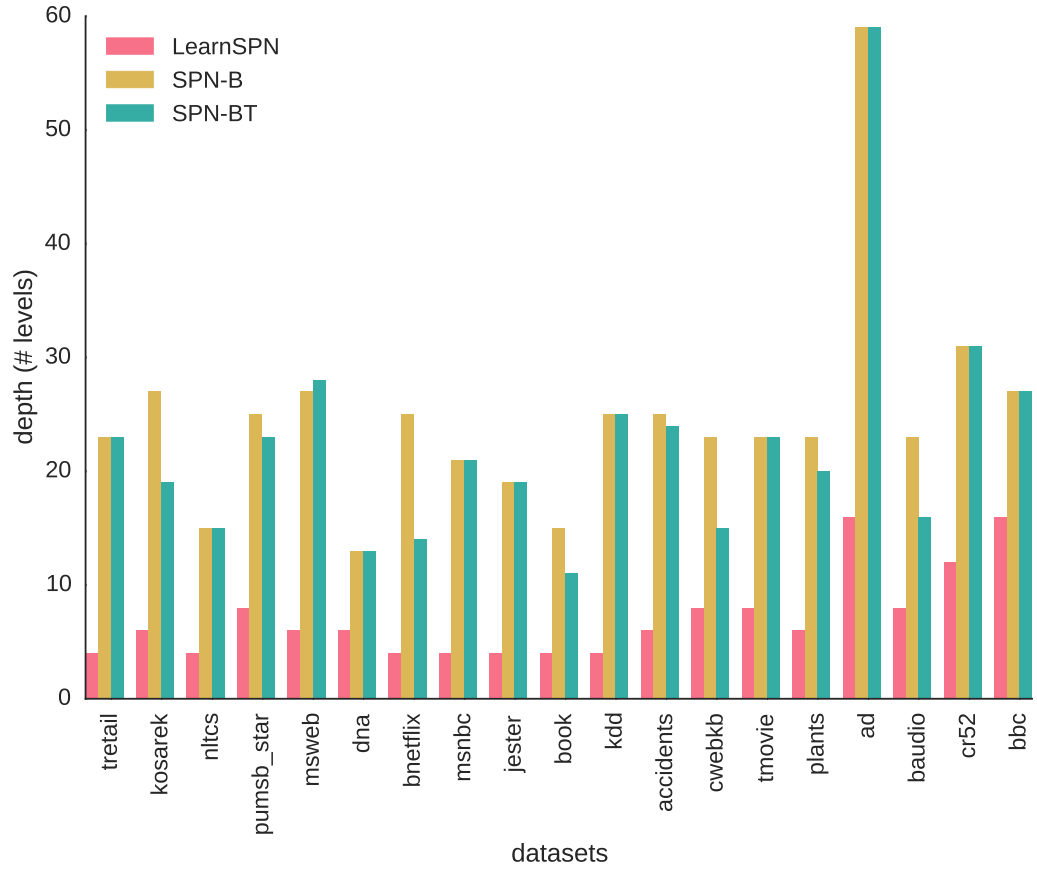


Figure : Comparing network depths for the networks scoring the best log-likelihoods in the grid search as obtained by LearnSPN, SPN-B and SPN-BT for each dataset.

Log-likelihoods

	LearnSPN	SPN-B	SPN-BT	ID-SPN	SPN-BB	SPN-BTB	MT
NLTCS	-6.110	-6.048	-6.048	**-5.998**	-6.014	-6.014	-6.008
MSNBC	-6.099	-6.040	-6.039	-6.040	**-6.032**	-6.033	-6.076
KDDCup2k	-2.185	-2.141	-2.141	-2.134	-2.122	**-2.121**	-2.135
Plants	-12.878	-12.813	-12.683	-12.537	-12.167	**-12.089**	-12.926
Audio	-40.360	-40.571	-40.484	-39.794	-39.685	**-39.616**	-40.142
Jester	-53.300	-53.537	-53.546	**-52.858**	**-52.873**	-53.600	-53.057
Netflix	-57.191	-57.730	-57.450	**-56.355**	-56.610	**-56.371**	-56.706
Accidents	-30.490	-29.342	-29.265	**-26.982**	-28.510	-28.351	-29.692
Retail	-11.029	-10.944	10.942	**-10.846**	-10.858	-10.858	**-10.836**
Pumsb-star	-24.743	-23.315	-23.077	**-22.405**	-22.866	-22.664	-23.702
DNA	-80.982	-81.913	-81.840	-81.211	-80.730	**-80.068**	-85.568
Kosarek	-10.894	-10.719	-10.685	-10.599	-10.690	**-10.578**	-10.615
MSWeb	-10.108	-9.833	-9.838	-9.726	-9.630	**-9.614**	-9.819
Book	-34.969	-34.306	-34.280	-34.136	-34.366	**-33.818**	-34.694
EachMovie	-52.615	-51.368	-51.388	-51.512	**-50.263**	**-50.414**	-54.513
WebKB	-158.164	-154.283	-153.911	-151.838	-151.341	**-149.851**	-157.001
Reuters-52	-85.414	-83.349	-83.361	-83.346	**-81.544**	-81.587	-86.531
BBC	-249.466	-247.301	-247.254	-248.929	**-226.359**	**-226.560**	-259.962
Ad	-19.760	-16.234	-15.885	-19.053	-13.785	**-13.595**	-16.012

Table : Average test log likelihoods for the best networks learned by all algorithms on all datasets after the grid search. In bold the values that are statistically better than all the others according to a Wilcoxon signed rank test with p-value of 0.05.

Conclusions

- Structure quality evaluation matters
- Deeper networks by applying a simplicity bias when splitting
- Regularized SPNs by introducing Chow-Liu trees as leaves
- More robust and accurate SPNs with bootstrapped sum nodes

Further works

- Are other sum node interpretations as gam effective?
- Is it possible to compress learned structures?
- How these variations influence representation learning?
- How can these variations be generalized to other tractable PGMs?

References

- Dennis, Aaron and Dan Ventura (2012). "Learning the Architecture of Sum-Product Networks Using Clustering on Varibles". In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., pp. 2033–2041 (cit. on p. 7).

- Gens, Robert and Pedro Domingos (2013). "Learning the Structure of Sum-Product Networks". In: *Proceedings of the 30th International Conference on Machine Learning*. JMLR Workshop and Conference Proceedings, pp. 873–880 (cit. on pp. 7, 22).

- Haaren, Jan Van and Jesse Davis (2012). "Markov Network Structure Learning: A Randomized Feature Generation Approach". In: *Proceedings of the 26th Conference on Artificial Intelligence*. AAAI Press (cit. on p. 22).

- Lowd, Daniel and Jesse Davis (2010). "Learning Markov Network Structure with Decision Trees". In: *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society Press, pp. 334–343 (cit. on p. 22).

- Martens, James and Venkatesh Medabalimi (2014). "On the Expressive Efficiency of Sum Product Networks". In: *CoRR* abs/1411.7717 (cit. on p. 8).

- Meilă, Marina and Michael I. Jordan (2000). "Learning with mixtures of trees". In: *Journal of Machine Learning Research* 1, pp. 1–48 (cit. on pp. 2, 22).

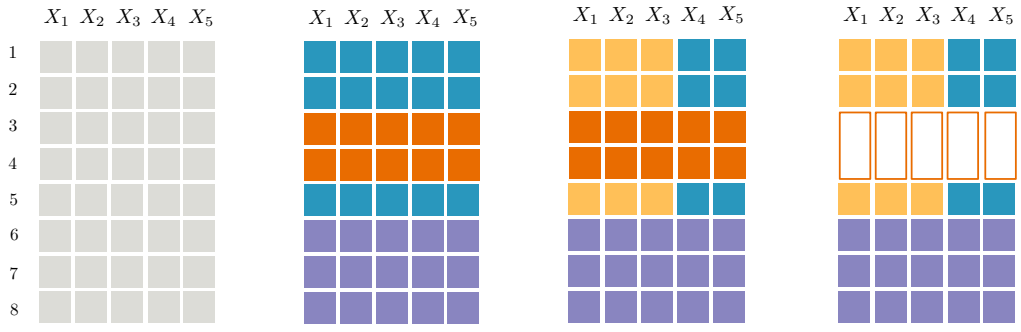
- Peharz, Robert, Bernhard Geiger, and Franz Pernkopf (2013). "Greedy Part-Wise Learning of Sum-Product Networks". In: *Machine Learning and Knowledge Discovery in Databases*. Vol. 8189. LNCS. Springer, pp. 612–627 (cit. on p. 7).

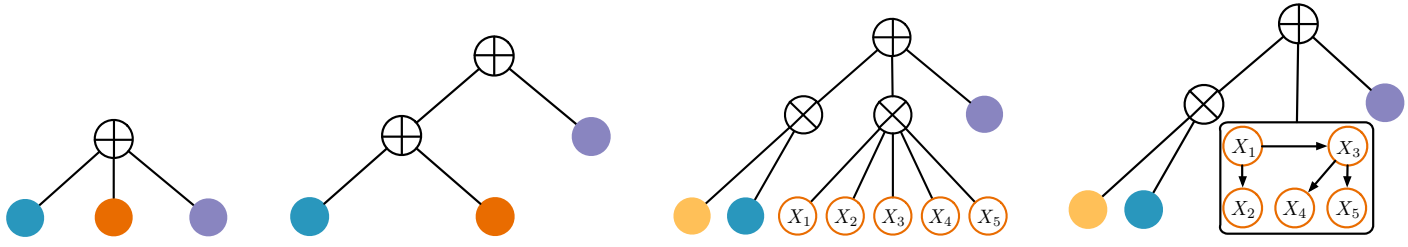
- Poon, Hoifung and Pedro Domingos (2011). "Sum-Product Network: a New Deep Architecture". In: *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning* (cit. on p. 2).

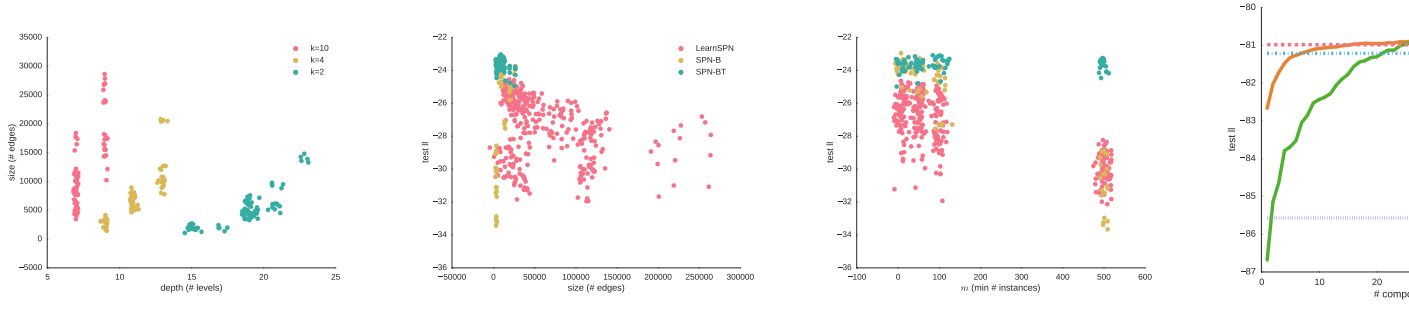
- Rooshenas, Amirmohammad and Daniel Lowd (2014). "Learning Sum-Product Networks with Direct and Indirect Variable Interactions". In: *Proceedings of the 31st International Conference on Machine Learning*. JMLR Workshop and Conference Proceedings, pp. 710–718 (cit. on pp. 2, 7).

- Zhao, Han, Mazen Melibari, and Pascal Poupart (2015). "On the Relationship between Sum-Product Networks and Bayesian Networks". In: *CoRR* abs/1501.01239. URL: `http://arxiv.org/abs/1501.01239` (cit. on p. 8).

Discuss


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


X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5


size (# edges)
depth (# levels)
k=10
k=4
k=2
test ll
size (# edges)
LearnSPN
SPN-B
SPN-BT
test ll
m (min # instances)
test ll
components
SPN-BB
SPN-BTB
LearnSPN
ID-SPN
MT