Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning

Antonio Vergari, Nicola Di Mauro and Floriana Esposito {firstname.lastname@uniba.it}

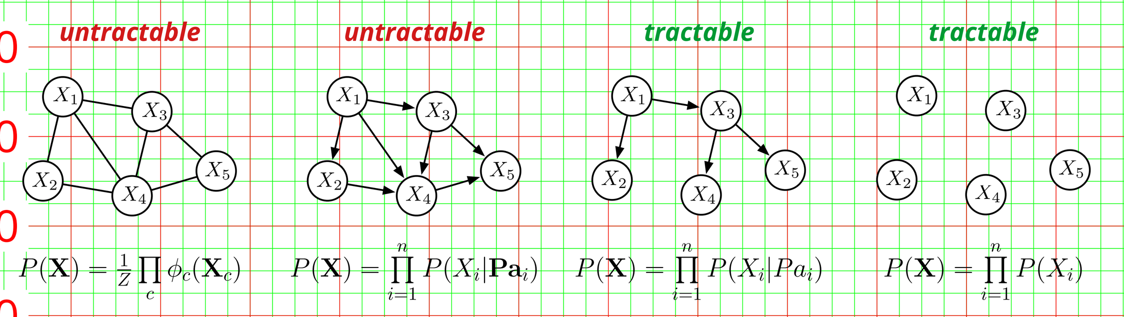
University of Bari "Aldo Moro", Italy
Department of Computer Science

LACAM
Machine Learning

Sum-Product Networks and Tractable Models

Probabilistic Graphical Models (PGMs) provide a tool to compactly represent joint probability distributions $P(\mathbf{X})$.

However, ***inference***, the main task one may want to perform on a PGM, is generally ***untractable***.

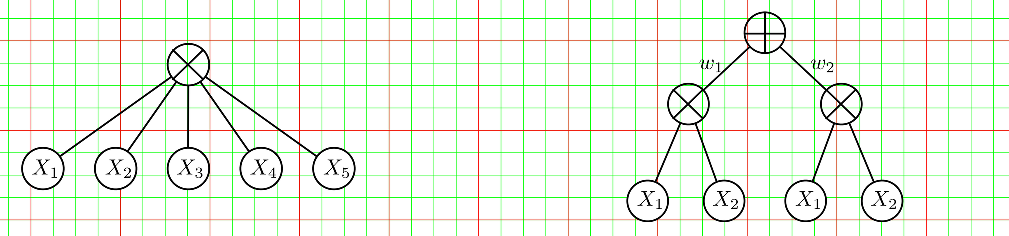


To guarantee polynomial inference, tractable models trade off model expressiveness.

Sum-Product Networks (SPNs) are DAGs *compiling* a pdf $P(\mathbf{X})$ into a ***deep*** architecture of **sum** and **product** nodes over univariate distributions $X_1, \dots, X_n$ as leaves. The parameters of the network are the weights w_{ij} associated to sum nodes children edges.

Product nodes define factorizations over independent vars, sum nodes mixtures.

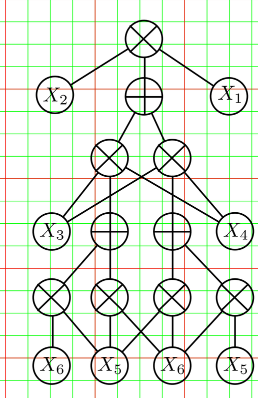
Products over nodes with different scopes (*decomposability*) and sums over nodes with same scopes (*completeness*) guarantee modeling a pdf (*validity*).



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*):

- $Z = S(*)$ (all leaves output 1)
- $P(\mathbf{e}) = S(\mathbf{e})/S(*)$
- $P(\mathbf{q}|\mathbf{e}) = \frac{P(\mathbf{q},\mathbf{e})}{P(\mathbf{e})} = \frac{S(\mathbf{q},\mathbf{e})}{S(\mathbf{e})}$
- $MPE(\mathbf{q},\mathbf{e}) = \max_{\mathbf{q}} P(\mathbf{q},\mathbf{e}) = S^{max}(\mathbf{e})$, turning sum nodes into max nodes

The ***depth of the network*** (# *layers*) determines expressive efficiency [5, 9]

How and why to perform structure learning

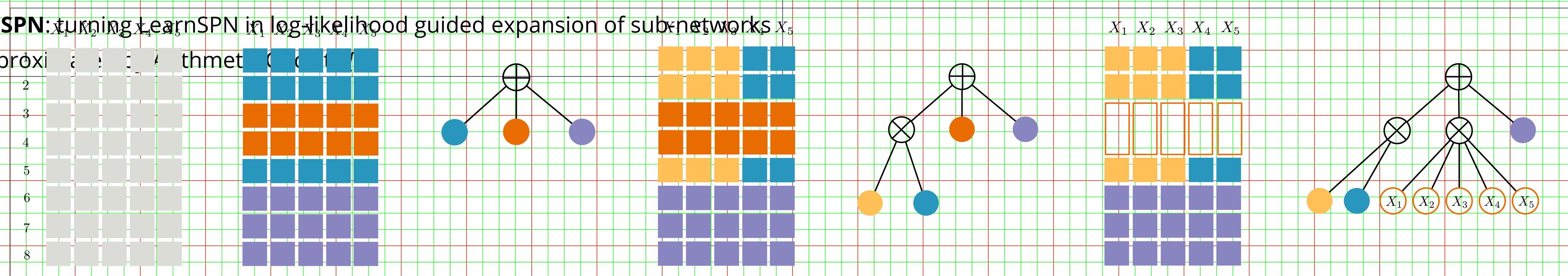
Constraint-based search formulation. Discover hidden variables for sum node mixtures and independences for product node components:

- greedy top-down: KMeans on features *[1]*;
- greedy bottom up: merging feature regions by a *Bayesian-Dirichlet independence test*, and reducing edges by maximizing MI *[7]*
- **ID-SPN**: turning LearnSPN in log-likelihood guided expansion of sub-networks approximated by arithmetic circuits

LearnSPN [2] builds a tree-like SPN by recursively split the data matrix:

- splitting columns in 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 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 the proportions of instances falling into each cluster

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

Tractability is guaranteed if the network size is polynomial in # vars.

Comparing network sizes is better than comparing inference times.

Overcomplex networks do not generalize well

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

Simplifying by limiting node splits

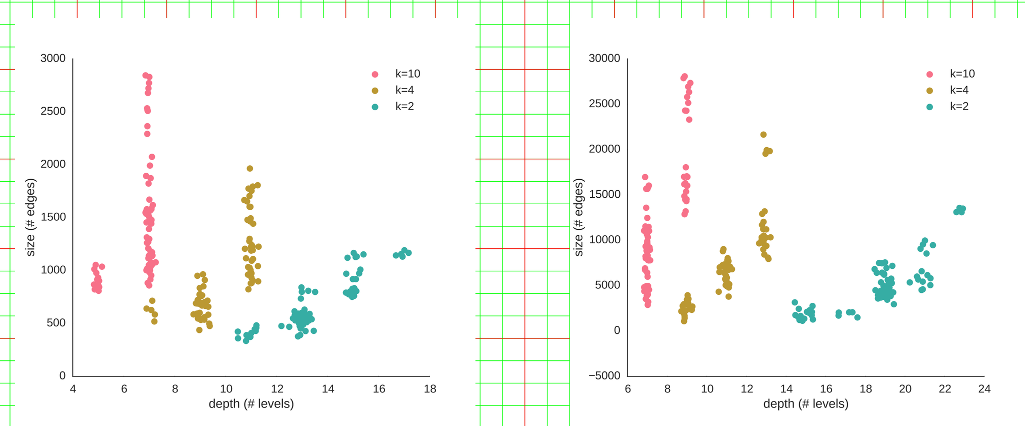
LearSPN performs two interleaved ***greedy hierarchical*** divisive ***clustering*** processes (co-clustering). Each process benefits from the other one improvements/highly suffers from other's mistakes.

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.

Pros:

- not committing to complex structures too early
- same expressive power: successive splits allow for more node children
- reducing node out fan increases the depth
- same accuracy, smaller networks

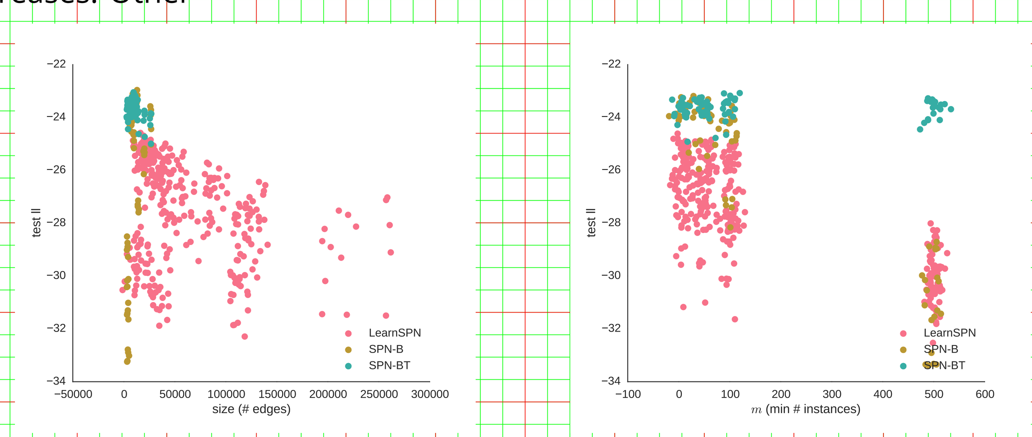
By increasingly limiting the max number of allowed splits the depth of the structures increases. It is also worth noting how the size of the network decreases. Other



Regularizing by introducing tree distributions as leaves

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

By increasingly limiting the max number of allowed splits the depth of the structures increases. It is also worth noting how the size of the network decreases. Other



Strengthening by model averaging

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Experiments

Classical setting for ***generative*** graphical models structure learning [2]:

- 19 binary datasets from classification, recommendation, frequent pattern mining...[4] [3]
- Training 75% Validation 10% Test 15% splits (no cv)

Comparing both accuracy and structure quality:

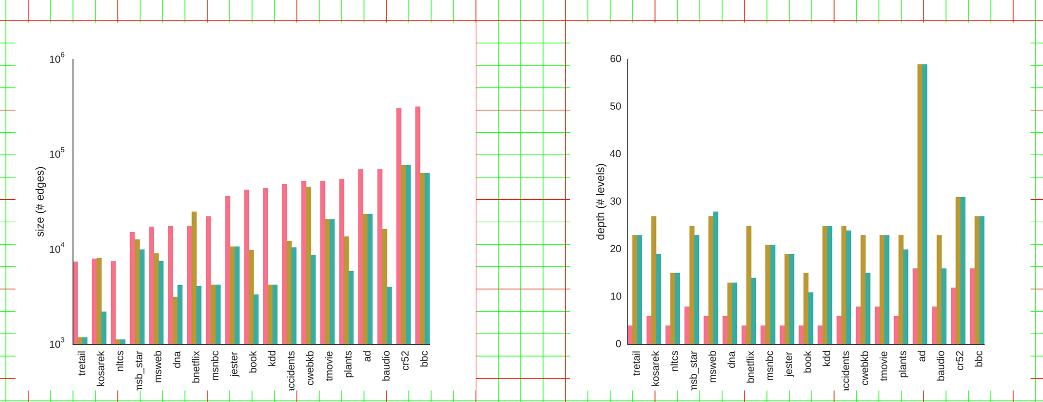
- ***average log-likelihood*** on predicting test instances
- networks sizes (# edges)
- network depth (# alternated type layers)

Comparing the state-of-the-art, LearnSPN, ID-SPN and MT [6], against our variations:

- SPN-B using only Binary splits
- SPN-BT with Binary splits and Trees as leaves
- SPN-BB combining Binary splits and Bagging
- SPN-BTB including all variants

Model selection via *grid search* in the same parameter space:

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



	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 all algorithms. In bold the best values after a Wilcoxon signed rank test with p-value of 0.05.

References

- Aaron Dennis and Dan Ventura. ``Learning the Architecture of Sum-Product Networks Using Clustering on Varibles''. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 2033–2041.
- Robert Gens and Pedro Domingos. ``Learning the Structure of Sum-Product Networks''. In: *Proceedings of the 30th International Conference on Machine Learning*. JMLR Workshop and Conference Proceedings, 2013, pp. 873–880.
- Jan Van Haaren and Jesse Davis. ``Markov Network Structure Learning: A Randomized Feature Generation Approach''. In: *Proceedings of the 26th Conference on Artificial Intelligence*. AAAI Press, 2012.
- Daniel Lowd and Jesse Davis. ``Learning Markov Network Structure with Decision Trees''. In: *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society Press, 2010, pp. 334–343.
- James Martens and Venkatesh Medabalimi. ``On the Expressive Efficiency of Sum Product Networks''. In: *CoRR* abs/1411.7717 (2014).
- Marina Meilă and Michael I. Jordan. ``Learning with mixtures of trees''. In: *Journal of Machine Learning Research* 1 (2000), pp. 1–48.
- Robert Peharz, Bernhard Geiger, and Franz Pernkopf. ``Greedy Part-Wise Learning of Sum-Product Networks''. In: *Machine Learning and Knowledge Discovery in Databases*. Vol. 8189. LNCS. Springer, 2013, pp. 612–627.
- Amirmohammad Rooshenas and Daniel Lowd. ``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, 2014, pp. 710–718.
- Han Zhao, Mazen Melibari and Pascal Poupart. ``On the Relationship between Sum-Product Networks and Bayesian Networks''. In: CoRR abs/1501.01239 (2015). URL: http://arxiv.org/abs/1501.01239.