Towards Representation Learning with Tractable Probabilistic Models

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

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

LACAM Laboratory
Machine Learning

Tractable Probabilistic Models

Density estimation is the unsupervised task of learning an estimator for the joint probability distribution $p(\mathbf{X})$ from a set of i.i.d. samples $\{\mathbf{x}_i\}_{i=1}^m$ over RVs $\mathbf{X}$

Once such a density estimator is learned, one uses it to answers probabilistic queries about configurations on $p(\mathbf{X})$, that is to do***inference***

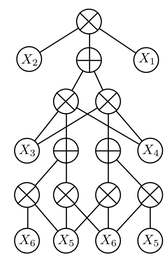
Operations that may be required to be efficient are
- $p(\mathbf{X} = \mathbf{x})$ (evidence)
- $p(\mathbf{E}), \mathbf{E} \subset \mathbf{X}$ (marginals)
- $p(\mathbf{Q}|\mathbf{E}), \mathbf{Q}, \mathbf{E} \subset \mathbf{X}, \mathbf{Q} \cap \mathbf{E} = \emptyset$ (conditionals)
- $\arg\max_{\mathbf{q} \sim \mathbf{Q}} p(\mathbf{q}|\mathbf{E})$ (MPE assignment)
- $Z = \sum_{\mathbf{x} \sim \mathbf{X}} \phi(\mathbf{x})$ (partition function)

Tractable Probabilistic Models (**TPMs**) are density estimators for which some kind of inference is *tractable*, i.e. polynomial in the number of r.v.s or their domains.

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 [3].

Representation Learning with TPMs

SPN structure learning is a constraint-based search. Main ideas: to discover hidden variables for sum nodes and independences for product nodes by applying some form of clustering along matrix axis. Different variations: using K-Means on features *[1]*; merging features bottom-up with IB heuristics *[4]*; ***LearnSPN*** *[2]* is the first principled top-down greedy algorithm.

LearnSPN builds a tree-like SPN by recursively splitting the data matrix: columns in pairs by a greedy ***G Test*** based procedure with threshold ρ: $G(X_i, X_j) = 2 \sum_{x_i \sim X_i} \sum_{x_j \sim X_j} c(x_i, x_j) \cdot \log \frac{c(x_i,x_j) \cdot |T|}{c(x_i)c(x_j)}$ (Figure 1.c); instances in $|C|$ clusters with ***online Hard-EM*** (Figure 1.b) with cluster number penalty λ: $Pr(\mathbf{X}) = \sum_{C_i \in \mathbf{C}} \prod_{X_j \in \mathbf{X}} Pr(X_j, C_i)$. Weights are the cluster proportions.

If there are less than m instances, it puts a ***naive factorization*** over leaves (Figure 1.d). For each univariate distribution it gets its ***ML estimation*** smoothed by α. LearnSPN hyperparameter space is thus: $\{\rho, \lambda, m, \alpha\}$.

The state-of-the-art, in terms of test likelihood, is **ID-SPN**: it turns LearnSPN in log-likelihood guided expansion of sub-networks approximated by Arithmetic Circuits *[5]*. However it is overparametrized, and slower.

Tractability is guaranteed if the network size is polynomial in # vars. ***Structure quality matters*** as much as likelihood. Comparing network sizes is more solid than comparing inference times.

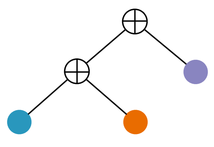
LearnSPN is too greedy and the resulting SPNs are overcomplex networks that may not generalize well. ***Structure quality desiderata***: *smaller* but *accurate*, *deeper* but not wider, SPNs.

Random query embedding extraction

LearSPN performs two interleaved ***greedy hierarchical*** divisive ***clustering*** processes. Each process benefits from the other one improvements and similarly suffers from the 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 but doing only Binary splits for sum nodes children ($k = 2$) when clustering rows.

Objectives: not committing to complex structures too early while retaining same expressive power (right Figure is equivalent to the SPN in Figure 1.b); moreover, reducing the node out fan increases the network depth. Plus, there is no need for λ anymore.



Input: a set of instances $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^m$ over r.v.s $\mathbf{X} = \{X_1, \ldots, X_n\}$, k as the number of features to generate
Output: a set of embeddings $\mathcal{E} = \{\mathbf{e}^i\}_{i=1}^m, \mathbf{e}^i \in \mathbb{R}^k$
$\theta \leftarrow$ learnDensityEstimator$(\mathcal{D})$
$\mathcal{E} \leftarrow \{\}$
for $j = 1, \ldots, k$ **do**
  $\mathbf{Q}_j \leftarrow$ selectRandomRVs$(\mathbf{X})$
  for $i = 1, \ldots, m$ **do**
    $e^i_j = p_\theta(\mathbf{x}^i_{\mathbf{Q}_j})$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathbf{e}^i\}$
return $\mathcal{E}$

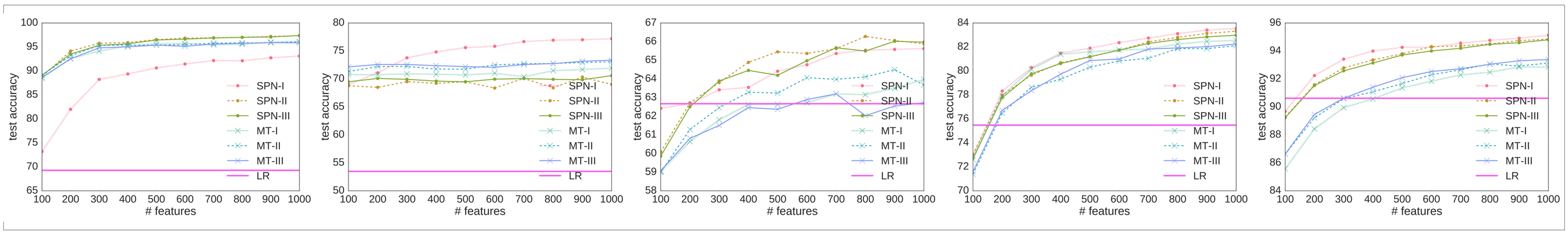
Input: a set of instances $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^m$ over r.v.s $\mathbf{X} = \{X_1, \ldots, X_n\}$, s as the number of patches to extract, d as the patch length,
Output: a set of embeddings $\mathcal{E} = \{\mathbf{e}^i\}_{i=1}^m, \mathbf{e}^i \in \mathbb{R}^k$
$\mathcal{R} \leftarrow \{\}$
for $i = 1, \ldots, s$ **do**
  $\mathbf{x}^{\text{rand}} \leftarrow$ selectRandomSample$(\mathcal{D})$
  $\mathbf{r}^i \leftarrow$ extractRandomPatch$(\mathbf{x}^{\text{rand}}, d)$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{r}^i\}$
$\theta \leftarrow$ learnDensityEstimator$(\mathcal{R})$
$\mathcal{E} \leftarrow \{\}$
for $i = 1, \ldots, m$ **do**
  $j \leftarrow 0$
  for each patch $\mathbf{q}^i, |\mathbf{q}^i| = d$ in $\mathbf{x}^i$ **do**
    $e^i_j = p_\theta(\mathbf{q}^i)$
    $j \leftarrow j + 1$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathbf{e}^i\}$
return $\mathcal{E}$

Evaluation

Empirical evaluation of the random marginal query approach:
- 5 binary image datasets: rectangles (REC), convex (CON), ocr_letters (OCR), caltech101 (CAL), binary MNIST (BMN)
- OVR L2-reg logistic regressor for all representations, LR baseline

- 3 SPN models trained with LearnSPN-b [6] with $m \in \{500, 100, 50\}$ (SPN-I, SPN-II, SPN-III)
- 3 Mixture of trees models with $k \in \{3, 15, 30\}$ (MT-I, MT-II, MT-III)

- 1000 randomly generated marginal queries corresponding to adjacent pixels in a rectangular image patch having minimum sizes of 2 pixels and maximum of 7 pixels for OCR and 10 pixels for the remaining datasets



References

[1] 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.

[2] Robert Gens and Pedro Domingos. "Learning the Structure of Sum-Product Networks". In: *Proceedings of the ICML 2013*. 2013, pp. 873–880.

[3] James Martens and Venkatesh Medabalimi. "On the Expressive Efficiency of Sum Product Networks". In: *CoRR* abs/1411.7717 (2014).

[4] Robert Peharz, Bernhard Geiger, and Franz Pernkopf. "Greedy Part-Wise Learning of Sum-Product Networks". In: *ECML-PKDD 2013*. 2013.

[5] 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*. 2014, pp. 710–718.

[6] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. "Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning". In: *ECML-PKDD 2015*. 2015.

ECML-PKDD 2016 - 19th-23rd September 2016, Riva del Garda, Italy