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 r.v.s $\mathbf{X}$

Given such an estimator, one uses it to answers probabilistic queries about configurations on $\mathbf{X}$, i.e. to do ***inference***

Different kinds of inference queries to answer:

- $p(\mathbf{X}=\mathbf{x})$ pointwise evidence (EVI)
- $p(\mathbf{E}), \mathbf{E} \subset \mathbf{X}$ marginals (MAR)
- $p(\mathbf{Q}|\mathbf{E}), \mathbf{Q}, \mathbf{E} \subset \mathbf{X}, \mathbf{Q} \cap \mathbf{E} = \emptyset$ conditionals (CON)
- $\arg\max_{\mathbf{q}\sim\mathbf{Q}} p(\mathbf{q}|\mathbf{E})$ MPE assignments (MPE)
- $Z = \sum_{\mathbf{x}\sim\mathbf{X}} \phi(\mathbf{x})$ partition function (Z)
- sampling: generate $\mathbf{x} \sim p(\mathbf{X})$ (SAM)

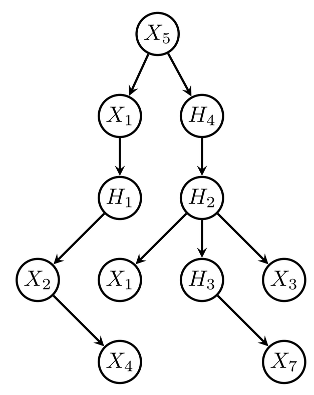
Tractable Probabilistic Models (**TPMs**) are density estimators for which some kind of ***exact*** inference is ***tractable***, i.e. polynomial in the number of r.v.s or their domains.

Different TPMs guarantee different inference kinds to be tractable.

TPMs can be roughly classified into:

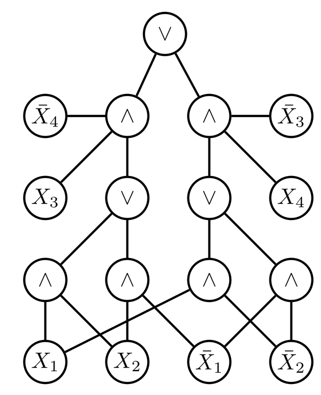
- ***low-treewidth*** Probabilistic Graphical Models (PGMs) (*)
- ***computational graphs*** compiling $p(\mathbf{X})$ ($^\times$)
- ***neural autoregressive*** models ($^+$)

Tree distributions [6] (*)



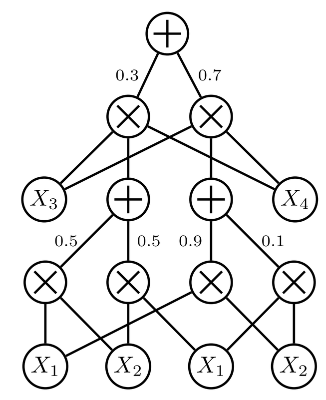
✓EVI, ✓SAM, ✓MAR, ✓CON, ✓MPE, ✓Z

d-DNNF [3] ($^\times$)



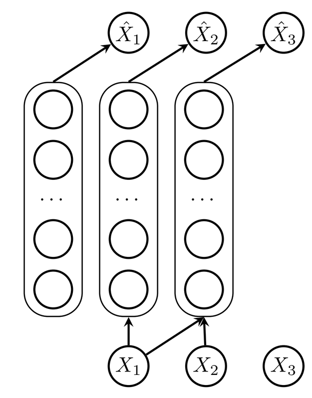
✓EVI, ✗SAM, ✓MAR, ✓CON, ✓MPE, ✓Z

Sum-Product Networks [7] ($^\times$)



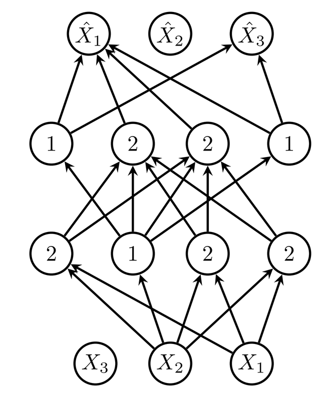
✓EVI, ✓SAM, ✓MAR, ✓CON, ✗MPE, ✓Z

NADEs [5] ($^+$)



✓EVI, ✓SAM, ✗MAR, ✗CON, ✗MPE, ✗Z

MADEs [4] ($^+$)



✓EVI, ✓SAM, ✗MAR, ✗CON, ✗MPE, ✗Z

Representation Learning with TPMs

Representation Learning deals with generating new representations for the initial data $\{\mathbf{x}^i\}_{i=1}^m$, e.g. ***embeddings*** $\{\mathbf{e}^i | \mathbf{e}^i \in \mathbb{R}^k\}_{i=1}^m$, k-dimensional continuous arrays.
Once learned, one can employ them in new tasks such as supervised classification or clustering [2].

Given a TPM θ we want to generate an embedding such as:

$$\mathbf{e}^i = f_{p,\theta}(\mathbf{x}^i)$$

for each sample, with f being a transformation provided by θ estimating the probability distribution p.

Simple idea: exploit the ***geometric space*** induced by p_θ.
E.g. P-kernels, Fisher vectors... [8], but:

- model dependent extraction
- closed form analytical derivation needed

We argue that TPMs can be employed as ***black box*** embedding extractors, by answering generated templated queries.

We define two approaches exploiting a random query generator schema:

I templates are constructed by **random marginal queries**, e.g. generating subsets of r.v.s $\mathbf{Q}_j \subseteq \mathbf{X}, j = 1\dots, d$:

$$e^i_j = p_\theta(\mathbf{Q}_j = \mathbf{x}^i_{\mathbf{Q}_j})$$

II generating a dataset of random *patches* from samples, then training a TPM θ on it; embeddings are generated by evaluating θ a sliding "*window*" of size d along the samples

$$e^i_j = p_\theta(\mathbf{q}^i), \forall \mathbf{q}^i \sim \mathbf{x}^i, |\mathbf{q}^i| = d$$

Tractable inference is mandatory: for a congruous embedding size k, one has to perform $k \cdot m$ queries, e.g. on training split of BMNIST $k = 10^3, m = 5 \cdot 10^4 \rightarrow 5 \cdot 10^7$ evals for approach I

Random query embedding extraction

Approach I demands **marginals** to be **tractable**

- more flexible: other kinds of inference kinds can be embedded, e.g. conditionals or more complex query types [1]
- exploiting already learned TPMs

Algorithm 1 randQueryEmbedding($\mathcal{D}$, k)

Input: a set of instances $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^m$ over r.v.s $\mathbf{X} = \{X_1, \dots, X_n\}$, 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 $j = 1, \dots, k$ **do**
  $\mathbf{Q}_j \leftarrow$ selectRandomRVs($\mathbf{X}$)
  for $i = 1, \dots, m$ **do**
    $e^i_j = p_\theta(\mathbf{x}^i_{\mathbf{Q}_j})$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathbf{e}^i\}$
return $\mathcal{E}$

Approach II requires only **tractable pointwise evidence**

- largely employable: many more models can answer pointwise queries in a tractable way
- needs to learn a TPM anew

Algorithm 1 randPatchEmbedding($\mathcal{D}$, s, d)

Input: a set of instances $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^m$ over r.v.s $\mathbf{X} = \{X_1, \dots, X_n\}$, 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 $i = 1, \dots, s$ **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 $i = 1, \dots, m$ **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 $\mathcal{E}$

Experimental Design

Future: Planning an extensive experimentation comprising at least two TPMs from each family class, on a wide range of query types, from datasets covering different application domains and employing both embedding generation approaches

Present: Empirical evaluation of approach I on **five binary image datasets** used for classification:

- rectangles (REC), 28×28 pixels, wide VS tall rectangles
- convex (CON), 28×28 pixels, convex VS concave shapes
- ocr_letters (OCR), 16×8 pixels, ten letters
- caltech101 (CAL), 28×28 pixels, 101 object shapes
- binary MNIST (BMN), 28×28 pixels, ten digits

Learning the structure of ***differently regularized*** **TPMs** on RVs $\mathbf{X}$ alone (unsupervised) to compare different *model capacities* ($\rightarrow$ does a model able of expressing more complex probability distribution provide more accurate embeddings?):

- 3 SPN architectures trained with LearnSPN-b [9] with hyperparameters: $\rho = 15$ for OCR and $\rho = 20$ for all the rest, $m \in \{500, 100, 50\}$ ($\rightarrow$ SPN-I, SPN-II, SPN-III), then grid search on $\alpha \in \{0.1, 0.2, 0.5, 1.0, 2.0\}$
- 3 Mixture of trees models (MT) [6] with $k \in \{3, 15, 30\}$ components ($\rightarrow$ MT-I, MT-II, MT-III) trained with EM

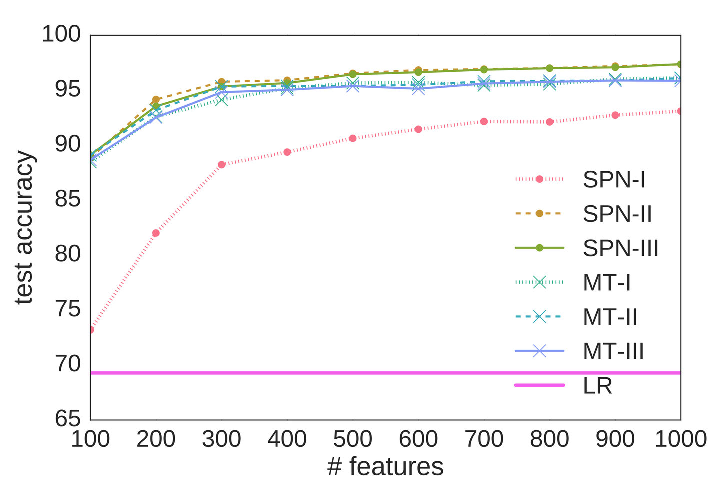
Extracting embeddings, then training a **linear classifier** on top of them **to predict** Y (supervised):

- OVR L2-reg logistic regressor for all representations, grid search for regularization coefficient C value in $\{0.0001, 0.001, 0.01, 0.1, 1.0\}$
- baseline with initial representations ($\rightarrow$ LR)

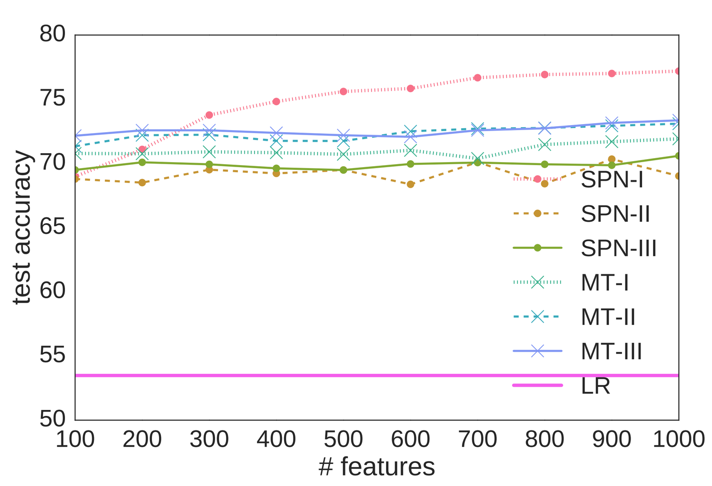
Random query generation scheme:

- up to 1000 randomly generated *marginal queries*
- generating subsets of RVs in $\mathbf{X}$ over adjacent pixels in rectangular patches of min sizes of 2, max of 7 pixels for OCR and 10 pixels for the others

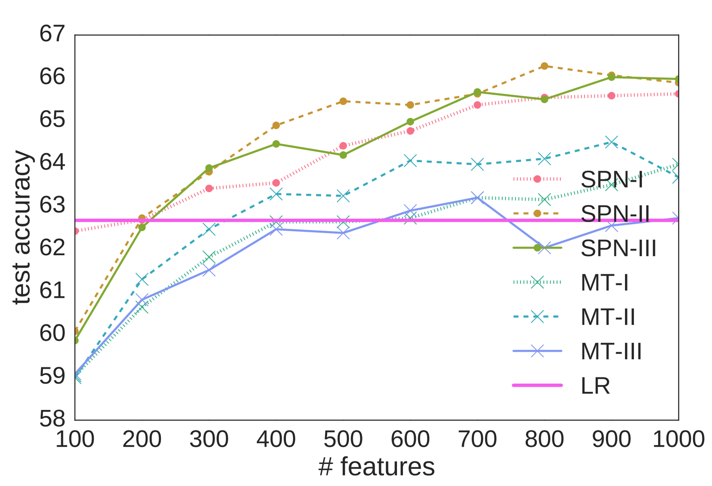
Results



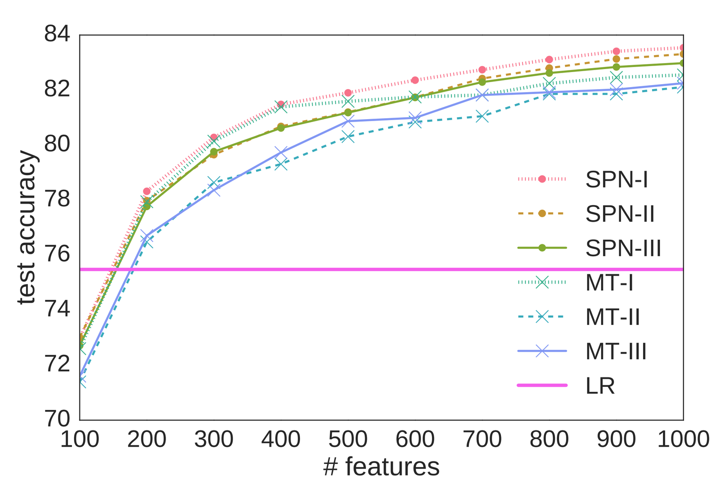
REC



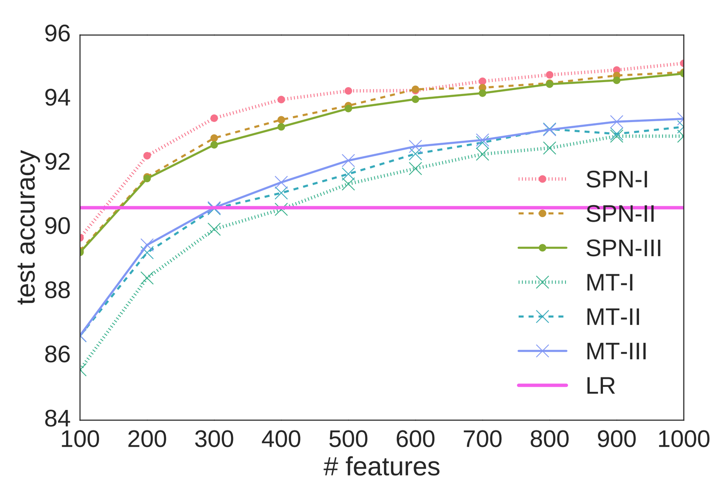
CON



OCR



CAL



BMN

For all models, less than 300 features to beat the LR baseline
$\rightarrow$ the **geometric space** induced by $p_\theta(\mathbf{X})$ is **meaningful**

SPN embeddings outperform MT ones all datasets, but CAL
$\rightarrow$ **better likelihoods** on $\mathbf{X}$ may lead to **worse accuracies** for Y

SPN-I models are often more predictive than SPN-III ones
$\rightarrow$ **more regularized models better** than specialized ones

References

[1] Jessa Bekker et al. "Tractable Learning for Complex Probability Queries". In: *Advances in Neural Information Processing Systems 28 (NIPS)*. 2015.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538 (2012).

[3] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge, 2009.

[4] Mathieu Germain et al. "MADE: Masked Autoencoder for Distribution Estimation". In: *CoRR* abs/1502.03509 (2015).

[5] Hugo Larochelle and Iain Murray. "The Neural Autoregressive Distribution Estimator". In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 29–37.

[6] Marina Meilă and Michael I. Jordan. "Learning with mixtures of trees". In: *Journal of Machine Learning Research* 1 (2000), pp. 1–48.

[7] Hoifung Poon and Pedro Domingos. "Sum-Product Networks: a New Deep Architecture". In: *UAI 2011* (2011).

[8] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004. ISBN: 0521813972.

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

Links

paper available at:
`https://arxiv.org/abs/1608.02341`
code available at:
`https://github.com/arranger1044/spyn-repr`
more references:
`https://github.com/arranger1044/awesome-spn`

ECML-PKDD 2016 - 19th September 2016, Riva del Garda, IT