Towards Representation Learning with Tractable Probabilistic Models

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito

Department of Computer Science, University of Bari, Italy
{antonio.vergari, nicola.dimauro, floriana.esposito}@uniba.it

Abstract. Probabilistic models learned as density estimators can be exploited in representation learning beside being toolboxes used to answer inference queries only. However, how to extract useful representations highly depends on the particular model involved. We argue that tractable inference, i.e. inference that can be computed in polynomial time, can enable *general* schemes to extract features from black box models. We plan to investigate how Tractable Probabilistic Models (TPMs) can be exploited to generate embeddings by random query evaluations. We devise two experimental designs to assess and compare different TPMs as feature extractors in an unsupervised representation learning framework. We show some experimental results on standard image datasets by applying such a method to Sum-Product Networks and Mixture of Trees as tractable models generating embeddings.

1 Background And Motivation

Density estimation is the unsupervised task of learning a model θ estimating a joint probability distribution p over a set of random variables (r.v.s) $\mathbf{X}$ that are assumed to have generated a given set of observed samples $\{\mathbf{x}^i\}_{i=1}^m$. Once such an estimator is learned, one can use it to do *inference*, that is computing the probability of queries about certain states of the r.v.s, e.g. the marginals $p_\theta(\mathbf{Q}), \mathbf{Q} \subseteq \mathbf{X}$. Many machine learning problems can be reframed as different kinds of inference tasks; for instance, the classification of a set of r.v.s $\mathbf{Y}$ can be done by Most Probable Explanation (MPE) inference [13].

Examples of commonly used density estimators comprise Probabilistic Graphical Models (PGMs) [13] like Bayesian Networks (BNs) and Markov Networks (MNs), that represent the conditional dependence assumptions in p in a graph formalism. On the other hand, (deep) neural models represent a factorization of p by the means of neural architectures like Restricted Boltzmann Machines (RBMs) and its variations [18], Fully Visible Sigmoid Belief Networks [8,2] or embed an inference algorithm into a network evaluation, like in the case of Variational Autoencoders [12]. Accurately estimating *complex* distributions from data and *efficiently* querying them are the required qualities for a good density estimator. Performing exact inference is, however, still a hard task in general. For PGMs, for example, even approximate inference routines may end up being exponential in the worst case [23].

We want to investigate how to extract useful representations from general purpose density estimators. That is, finding a way to exploit probabilistic models, already learned to encode p in an unsupervised fashion by using existing learning algorithms, in order to transform the initial data into another representation space. More specifically, we want to build an *embedding* for each sample, $\mathbf{e}^i = f_{\theta,p}(\mathbf{x}^i)$, such that the transformation f is provided by a probabilistic model θ, learned to estimate p. These embeddings could be employed in other tasks such as clustering and classification, allowing for new transfer and unsupervised learning schemas, as usually done in representation learning [3]. The usefulness of these representations can be stated by proxy performance metrics on the subsequent supervised tasks, e.g. the accuracy scored on predicting some previously unseen sample labels [3]. Moreover, they could enable new ways to assess and compare different models, going beyond the classic likelihood comparison, which can be highly misleading sometimes [26].

To build a mapping $f_{\theta,p}$, one can take advantage of the geometric space induced by p_θ, the estimate of p according to model θ. A straightforward way is to build each embedding component as the result of a single inference step on θ according to some query $\mathbf{Q} = \mathbf{q}_j$, i.e. $e^i_j = p_\theta(\mathbf{q}_j)$. The classic way in which this has been implemented in the past is to employ the probability space embeddings in the construction of hand-crafted and problem dependent kernels, like P-kernels [25]. Another possibility is to leverage additional first order information as the one encoded in the gradient of p, as already done by Fisher vectors and Fisher kernels [11]. However, to implement these classic approaches, one has to derive analytically the embedding computation according to each model parameters, a task not always feasible for all models [25].

We are interested in a generic procedure that employs different density estimators as black boxes, despite their parametrizations or inner representations. By following the first line of thought, we try to capture the differences among the samples projected in various spaces induced by the probability densities encoded in a model θ. The basic idea is to construct features with a random query generator, computing their values according to the evaluation of θ. Hence, *tractable* inference routines become crucial to construct complex enough representations by evaluating θ several times.

In the following sections we introduce Tractable Probabilistic Models and we define two possible schemas to generate embeddings from them by leveraging the tractability property. We then discuss a possible empirical evaluation of such a framework and show some results for an experimental application on benchmark image datasets, showing promising results.

2 Tractable Probabilistic Models

Tractable Probabilistic Models (TPMs) are density estimators for which exact inference is polynomial in $|\mathbf{X}|$. It is important to note that tractability of inference is not a global property, but it is associated to classes of queries. For instance, pointwise evidence, marginals, MPE inference and the computation of

a partition function, etc can each result tractable for some models while the others are unfeasible for the same models [6].

Here we provide a rough classification and review for some TPMs commonly found in the literature:

- *low treewidth* PGMs, alleviating the computation of the partition function of p by limiting the model expressiveness in terms of the representable conditional dependencies. They comprise models such as Hidden Markov Models [13], tree distributions and their mixture [19] or latent r.v. variants [4], for which pointwise and marginal queries can be answered in time linear to the number of r.v.s.
- computational models derived from a *knowledge compilation* process, involving the elicitation of another representation form for p (e.g. the network polynomial), usually encoding it into a computation graph. This is the case of Arithmetic Circuits (ACs) [6], and Sentential Decision Diagrams (SDDs) [7], both enabling intractable BNs and MNs to be compiled into data structures for which marginals and even MPE inference can be answered in time linear in the size of such structures [17].
- *neural autoregressive* models, factorizing p according to the chain rule and modeling each factor by a (possibly deep) neural network. Models like Neural Autoregressive Distribution Estimators (NADEs) [16] and Masked Autoencoder Distribution Estimators (MADE) [10] leverage constrained feedfoward and autoencoder networks to provide pointwise inference linear in the size of the networks. To answer marginal queries in polynomial time, variants like EoNADE, allowing for an order agnostic training of the factors, are necessary [27].

Complete evidences, marginals, and even the computation of the partition function for p are computable in tractable time for Sum-Product Networks (SPNs) [22]. SPNs are deep neural architectures equivalent to ACs for finite domains, compiling the network polynomial of p into a more sophisticated architecture that introduces a latent variable semantics [21,20]. Differently from other tractable neural models, SPNs guarantee that each hidden neuron still models a correct probability distribution over a restriction of the input r.v.s $\mathbf{X}$, named its *scope*. This constrained form allows for a *direct encoding* of the input space. The great interest around SPNs is also motivated by the increasing arsenal of structure learning algorithms arising in the literature [9,28]. This makes SPNs one of the few deep architectures for which the structure can be directly and effectively learned and not crafted or tuned by hand.

3 Representation Learning for TPMs

For tractable models like HMMs one can leverage the emission probabilities for each evaluated sample to generate a particular kernel [25]. This case is an example of a model dependent embedding space construction, in which not only the tractability of the model is mandatory, but also the input space definition

shall be suitable (e.g. sequences as samples). In the same way, building a Fisher kernel for Gaussian Mixture Models can be done easily because it is possible to compute the gradient of the likelihood in a closed form for each parameter.

For neural architectures, representations are usually extracted as the activations of hidden neurons after the evaluation of each input sample. It is common practice to extract the embeddings from their last layers, comprising higher level representations, hence the more informative ones [3]. While one could adopt the same approach for other neural autoregressive density estimators, the probabilistic meaning of the extracted features would be lost and a direct comparison against other models would not be possible. Extracting node activations from an SPN would retain their probabilistic interpretation, however they would highly depend on the (structure or weight) learning algorithm employed to train the model. Moreover, it is still not clear which hidden neurons to choose to extract meaningful embeddings from an SPN. Following a layer-wise procedure for the embedding extraction does not cope with the different scope semantics of an SPN. As a general rule of thumb, one could employ all the nodes or all the nodes of one kind (sum or products only).

We argue that different TPMs as black box density estimators can be used to generate vector embeddings on a common ground, by defining a set of template queries to be answered. By doing so, one is able to evaluate different regions of the probability density surfaces induced by different queries. For instance, defining marginalized conditional queries would represent probability distributions different from the joint p, highlighting local interactions for the input r.v.s.

The most basic way to do so is to generate these queries in a random fashion. However, defining more sophisticated schemes is possible. In the end, a domain knowledge guided embedding extraction process could be thought as a sort of feature engineering step in which query template families are devised as features.

We propose two different embedding generation schemas exploiting randomness. In the first one, each feature template is constructed as a random marginal query evaluation, i.e. a subset of r.v.s $\mathbf{Q}_j \subseteq \mathbf{X}, j = \ldots, k$ is randomly chosen for each component of the k-dimensional embedding to generate. Each sample embedding is then exactly computed as $e^i_j = p_\theta(\mathbf{Q}_j = \mathbf{x}^i_{\mathbf{Q}_j})$, where $\mathbf{x}^i_{\mathbf{Q}_j}$ indicates the restriction of the sample vector $\mathbf{x}^i$ to the r.v.s in $\mathbf{Q}_j$. Again, the random selection criterion can be lead by domain knowledge, e.g. if the samples $\mathbf{x}^i$ are images, it is meaningful to extract adjacent pixels as the r.v.s $\mathbf{Q}_j$. A sketch of this process is presented in Algorithm 1.

The second approach, presented as Algorithm 2, extracts a set of random "patches" from the training data, as s random portions of size d, $\{\mathbf{r}^i\}_{i=1}^s$, from randomly chosen input samples. Then a TPM θ is trained on this reduced dataset. Each embedding component is then generated by evaluating the original samples for different subsets $\{\mathbf{Q}_h \subset \mathbf{X}\}_{h=1}^w, |\mathbf{Q}_h| = d$ of the training samples of length d, according to model θ. For instance, in a sliding window approach, with unitary stride, the total number of adjacent subset of length d from a vector of size n would be $w = n - d$. This is more meaningful for image samples as it is a way to take into account pixel autocorrelation and translation invariance. If

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

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

Algorithm 2 randPatchEmbedding($\mathcal{D}$, s, d)

Input: a set of instances $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^m$ over r.v.s $\mathbf{X} = \{X_1, \ldots, X_n\}$, 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**
        $e_j^i = p_\theta(\mathbf{q}^i)$
        $j \leftarrow j + 1$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathbf{e}^i\}$
return $\mathcal{E}$

that were the case, this approach would be similar to the dictionary learning one proposed in [5]. In this case, however, the feature generation scheme is directly done by a pointwise evaluation of a whole TPM, which in turn is directly learned on the patches.

In the first approach it is mandatory that the models involved guarantee tractable marginal inference. Indeed, in a naive computation one would have to ask a model $k \cdot m$ marginal queries, and even if one cashes the answers for each possible observable state configuration for the r.v.s $\mathbf{Q}_j$, assuming all of them to have at most l values, $l < m$, it would end up with $k \cdot l$ evaluations. In a random generation approach, the value of k will probably be large enough to demand each single evaluation to be as fast as possible.

The first approach is also the more flexible one, since it can be adapted to other types of queries, possibly involving more complex computations but still

tractable for certain models. For example, symmetric queries like parity counting, end up being tractable for SDDs [1].

On the other hand, the second one requires only tractable pointwise inference. Therefore, it is more easily adaptable to a larger set of models, since all locally learned models are globally evaluated and no marginal computations are involved.

Both models allow for very different TPMs to be evaluated against the same exact set of feature templates, since the generation of the random queries and patches can be done just once in a comparative experiment. This aspect allows for fairer comparisons among different models, moreover, it enables an additional way to understand the structure of such models: feature selection routines applied on the generated embedding spaces could reveal particularly effective sets of features hinting at powerful r.v.s interactions.

4 Evaluation

We plan to conduct a thorough empirical evaluation comprising at least two TPMs for each of the kinds presented in the second section. The meaningfulness and usefulness of the generated embeddings shall be measured against different metrics and in different tasks. For instance, both multi-class and multi-label classification tasks are worth investigating, and for each one of them, several metrics like accuracy, hamming loss and exact match shall be taken into account. The main objective of this intensive empirical comparison is to investigate how different TPMs would behave under this general embedding extraction framework.

Here we present a smaller empirical evaluation of the first proposed approach in a classification task on three standard benchmark image datasets. We employ both SPNs and Mixture of Trees (MT) [19] as a PGM for which marginal inference is tractable. We want to determine how different models, with different expressive capabilities, generate embeddings against the *same* marginal query evaluations. To learn a supervised classifier, we employ a logistic regressor with an L2 regularizer in a one-versus-rest setting on top of all the embeddings we generate. We use the implementation available in the scikit-learn framework[1]. For each representation we determine the regularization coefficient C value in $\{0.0001, 0.001, 0.01, 0.1, 1.0\}$, choosing the model with the best validation accuracy. As the simplest baseline possible, we apply such a classifier directly on the initial data representation $\mathbf{x}^i$, denoting it with LR. To declare an extracted representation as useful, we expect it to surpass the accuracy score of LR.

We consider the following datasets: Rectangles (REC) as introduced in [15], Convex (CON), always introduced in [15], OCR Letters (OCR) as presented in [14], Caltech-101 Silhouettes (CAL) [18] and a binarized version of MNIST (BMN) as processed in [24]. The accuracy scores for the best LR models are 75.58% for OCR, 62.67% for CAL and 90.62% for BMN.

We employ models of the same family but with different model capacities to study the effect of regularization on the same set of features. We learn three

[1] `http://scikit-learn.org/`

differently regularized SPN models on each dataset, by employing LearnSPN-b [28], a structure learning algorithm adopting a simplified iterated clustering strategy for determine the insertion of the inner nodes in the networks. We let vary the early stopping parameter $m \in \{500, 100, 50\}$ while we fix the G-test threshold ρ to 15 for OCR and 20 for all the other datasets. The effect of parameter m is to regularize the learned SPNs, therefore it governs the model capacity. We denote these models as SPN-I, SPN-II and SPN-III respectively. For MT, we let the number of the mixture component vary among 3, 15 and 30, ending up with three models, MT-I, MT-II, MT-III.

We evaluate all the models on 1000 randomly generated marginal queries on each dataset. For each query we select r.v.s 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.

Figure 1 shows the accuracy scores by SPN-I, SPN-II, SPN-III and MT-I, MT-II, MT-III on each dataset, while adding 100 features at a time. The first consideration one can draw is that all models are able to beat the LR baselines without using all the random features. On OCR, CAL and BMN, 300 features appear to be enough, for better scoring models, even 200 features provide a nice boost of 1 to 2 points in accuracy compared to the baselines. On REC and CON, just 100 features score a very high accuracy, indicating that the geometric space induced by the respective models is indeed meaningful. These considerations hint to the effectiveness of such an approach despite its randomness.

It is also visible how such embedding accuracies improve as the number of feature increases. This is true in almost all scenarios with the exception of REC and CON. On these two datasets, a sort of model "herding" can be noted: model performances are not growing noticeably while features are added, with the exception of the more reguarized model, SPN-I on REC and of the least regularized one, SPN-III on CON. Adding even more features seemed to leave the performance as it is, with a decreasing pattern when the embedding length has reached 2000. A behavior that is likely due to the introduction of too many irrelevant features and to the curse of dimensionality.

Moreover, it is visible how the SPN extracted embeddings outperform the MT generated ones in almost scenarios, with the exception of CON. If we were to compare only the likelihood of these models, the MT ones would outperform the SPN ones. One explanation for this behavior can be found in the better capacity of the SPN structure learner in optimizing the marginal loglikelihood [9]. Nevertheless, this clearly suggests that a learned representation comparison can be new informative dimension along which to assess different probabilistic models.

Concerning the performance of differently regularized models, one can note how more specialized models like SPN-III and SPN-II tend to perform less well that SPN-I, a pattern that can be seen for MT-III on CAL as well. Such an evidence can be considered as a form of overfitting for the density estimators. The greater degree with which they are able to reconstruct and model the training set, even if does not damage the test data likelihood, makes the extracted embedding probably too specific, at least for this set of marginal queries.

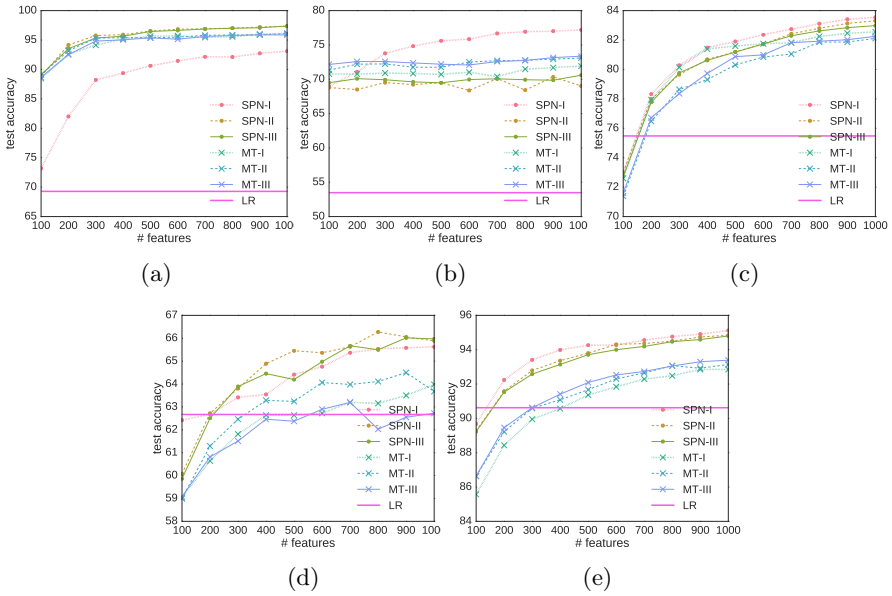

Fig. 1: Test accuracies for SPN and MT models on REC (Figs. 1a), CON (Figs. 1b), OCR (Figs. 1c), CAL (Figs. 1d) and BMN (Figs. 1e). against 1000 features generated as random marginal queries evaluations. Logistic regressor as a baseline is reported as LR.

5 Conclusions

We devised a general and model agnostic schema to extract embeddings from Tractable Probabilistic Models by exploiting random query generations and evaluations. We plan to conduct an extensive empirical experimentation to investigate in depth several models behavior according to different embedding evaluation performances. A proposed empirical comparison on five different standard image datasets and exploiting random marginal queries hints to the effectiveness of this approach.

All in all, the proposed approaches could be used to compare differently uncomparable models in a representation learning framework. They could serve a role similar to Parzen windows when models for which the likelihood cannot be easily computed have to be compared in a generative framework.

More sophisticated query types can improve this general but basic schema. It would be interesting to correlate the embedding performances to different query types according to different TPM models.

References

1. Bekker, J., Davis, J., Choi, A., Darwiche, A., Van den Broeck, G.: Tractable learning for complex probability queries. In: Advances in Neural Information Processing Systems 28 (NIPS) (2015)
2. Bengio, Y., Bengio, S.: Modeling high-dimensional discrete data with multi-layer neural networks. In: Advances in Neural Information Processing Systems 12. pp. 400–406. MIT Press (2000)
3. Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised Feature Learning and Deep Learning: A review and new perspectives. CoRR abs/1206.5538 (2012)
4. Choi, M.J., Tan, V.Y.F., Anandkumar, A., Willsky, A.S.: Learning latent tree graphical models. Journal of Machine Learning Research 12, 1771–1812 (2011)
5. Coates, A., Lee, H., Ng, A.Y.: An analysis of single layer networks in unsupervised feature learning. AISTATS 2011 (2011)
6. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge (2009)
7. Darwiche, A.: Sdd: A new canonical representation of propositional knowledge bases. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two. IJCAI'11 (2011)
8. Frey, B.J.: Graphical Models for Machine Learning and Digital Communication. MIT Press, Cambridge, MA, USA (1998)
9. Gens, R., Domingos, P.: Learning the Structure of Sum-Product Networks. In: Proceedings of the ICML 2013. pp. 873–880 (2013)
10. Germain, M., Gregor, K., Murray, I., Larochelle, H.: MADE: masked autoencoder for distribution estimation. CoRR abs/1502.03509 (2015)
11. Jaakkola, T.S., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II. pp. 487–493. MIT Press, Cambridge, MA, USA (1999), `http://dl.acm.org/citation.cfm?id=340534.340715`
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR). No. 2014 (2013)
13. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
14. Larochelle, H., Bengio, Y., Turian, J.P.: Tractable Multivariate Binary Density Estimation and the Restricted Boltzmann Forest. Neural Computation 22 (2010)
15. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation. In: Proceedings of the ICML 2007. pp. 473–480 (2007)
16. Larochelle, H., Murray, I.: The Neural Autoregressive Distribution Estimator. In: International Conference on Artificial Intelligence and Statistics. pp. 29–37 (2011)
17. Lowd, D., Rooshenas, A.: Learning Markov networks with arithmetic circuits. In: Proceedings of the 16th International Conference on Artificial Intelligence and Statistics. JMLR Workshop Proceedings, vol. 31, pp. 406–414 (2013)
18. Marlin, B.M., Swersky, K., Chen, B., Freitas, N.D.: Inductive Principles for Restricted Boltzmann Machine Learning. In: AISTATS 2010. pp. 509–516 (2010)
19. Meilă, M., Jordan, M.I.: Learning with mixtures of trees. Journal of Machine Learning Research 1, 1–48 (2000)
20. Peharz, R., Gens, R., Pernkopf, F., Domingos, P.M.: On the latent variable interpretation in sum-product networks. CoRR abs/1601.06180 (2016), `http://arxiv.org/abs/1601.06180`

21. Peharz, R., Tschiatschek, S., Pernkopf, F., Domingos, P.: On theoretical properties of sum-product networks. The Journal of Machine Learning Research (2015)
22. Poon, H.: Tutorial on spn. NIPS2011 (2011)
23. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(1–2), 273–302 (1996)
24. Salakhutdinov, R., Murray, I.: On the Quantitative Analysis of Deep Belief Networks. In: Proceedings of the ICML 2008. vol. 25 (2008)
25. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
26. Theis, L., van den Oord, A., Bethge, M.: A note on the evaluation of generative models. In: International Conference on Learning Representations (Nov 2016)
27. Uria, B., Murray, I., Larochelle, H.: A Deep and Tractable Density Estimator. ArXiv e-prints (Oct 2013)
28. Vergari, A., Di Mauro, N., Esposito, F.: Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning. In: ECML-PKDD 2015 (2015)