Poisson Sum-Product Networks: A Deep Architecture for Tractable Multivariate Poisson Distributions

Alejandro Molina
TU Dortmund University, Germany
alejandro.molina@tu-dortmund.de

Sriraam Natarajan
Indiana University, USA
natarasr@indiana.edu

Kristian Kersting
TU Dortmund University, Germany
kristian.kersting@cs.tu-dortmund.de

Abstract

Multivariate count data are pervasive in science in the form of histograms, contingency tables and others. Previous work on modeling this type of distributions do not allow for fast and tractable inference. In this paper we present a novel Poisson graphical model, the first based on sum product networks, called PSPN, allowing for positive as well as negative dependencies. We present algorithms for learning tree PSPNs from data as well as for tractable inference via symbolic evaluation. With these, information-theoretic measures such as entropy, mutual information, and distances among count variables can be computed without resorting to approximations. Additionally, we show a connection between PSPNs and LDA, linking the structure of tree PSPNs to a hierarchy of topics. The experimental results on several synthetic and real world datasets demonstrate that PSPN often outperform state-of-the-art while remaining tractable.

Introduction

Count data is at the center of many scientific endeavors - from citation counts to web page hit counts, from counts of procedures in medicine to the count of births and deaths in census, from counts of words in a document to the count of gamma rays in physics. Count data is omnipresent and in many cases interconnected. Modeling one event such as the number of times a certain lab test yields a particular result can provide an idea of the number of potentially invasive procedures that need to be performed on a patient. Thus, modeling such data faithfully can yield great insights into the properties of these domains. However, most prior learning and modeling methods view these problems using the lens of univariate distributions. Thus, they essentially assume that these events are not connected to one another. More recent approaches on Poisson Graphical Models treat these as the exponential family of distributions allowing for modeling the dependencies between the count variables.

More precisely, making the assumption of normality of count data poses difficulties. In cases of low counts, the left tail of a Gaussian distribution can become negative allowing for predicting counts with negative values. Also, if the mean λ of a Poisson is small, say less than 5, then its probability histogram is markedly asymmetrical, making a Normal-approximation ill-suited. If we model the data using a multinomial assumption, then there are two key issues. First is that they assume an upper bound on the count which may not be reasonable in many domains. Second, there is not necessarily an order among the different counts, implying that two very close values can have widely different probabilities. Poisson Graphical Models (PGMs) aim at overcoming these limitations. The Poisson MRF distribution (PMRF) (Yang et al. 2012) assumes that every conditional distribution is 1D Poisson. Unfortunately, the original formulation allowed only for negative dependencies between count variables. Consequently several modifications were proposed to allow for positive dependencies. Consider for instance, the Truncated PMRF (Yang et al. 2013) and the Fixed-Length PMRF (Inouye, Ravikumar, and Dhillon 2015). However, they assume an upper bound on the counts. Alternatively, one can drop the requirement of a consistent joint distribution. The resulting Poisson Dependency Network (PDN) model (Allen and Liu 2013; Hadiji et al. 2015) permits an efficient structure learning approach using gradient tree boosting, which was proven competitive to other Poisson GMs. Nevertheless, none of the Poisson GMs naturally provide tractable inference. This also holds for the recent Square Root GMs (Inouye, Ravikumar, and Dhillon 2016), the recent deep models for count data such as the Deep Exponential Family (Ranganath et al. 2015), Deep Poisson Factor Modeling (Henao et al. 2015), and Poisson Gamma Belief Networks (Zhou, Cong, and Chen 2015). The difficulties in achieving tractable inference arise in modelling, as the number of parameters of log-linear models (Lauritzen 1996) in the context of contingency tables grows exponentially with the number of variables. And from the general problem of inference in graphical models, which is known to be #P even for binary random variables. We, on the other hand, propose the first tractable Poisson GM.

Inspired by the successes of deep models, we present *Poisson Sum-Product Networks* (PSPNs)[1]. They allow for tractable inference based on polynomials of univariate Poissons and symbolic evaluation. They require, however, novel decomposition and conditioning steps tailored towards Poissons. Overall, we make a few important contributions: (1)

[1] *http://www-ai.cs.uni-dortmund.de/weblab/code.html*

We present the first tractable, deep multivariate Poisson model. (2) We derive an algorithm for learning tree PSPNs efficiently that involves the first mixtures of PDNs and a novel independency test for Poisson variables. (3) We show the connection between the more popular LDA models with PSPNs by employing that the conditionals of the LDA models are equivalent to the product of independent Poissons. (4) Finally, we demonstrate empirically the superiority of PSPNs against standard machine learning models on real and synthetic data sets.

We begin by presenting background and related work on SPNs as well as a generic algorithm for learning tree SPNs. Afterwards, we introduce PSPNs and device a learning approach for tree PSPNs. Before concluding we present our experimental results.

Sum-Product Networks

Sum Product Networks (SPNs) (Poon and Domingos 2011) are deep Graphical Models (GMs) capable of representing tractable distributions. The key benefit of SPNs over conventional graphical models such as Markov Random Fields (MRFs) is that common probabilistic inference tasks such as maximum-a-posteriori (MPE) and posterior marginal (MAR) estimation scales linearly with the size of the model. In GMs, these tasks are known to be NP-hard in general. The caveat is that SPNs can be exponentially larger than other GMs.

Definition of SPNs: Formally, an SPN as shown in Fig. 1 is a rooted directed acyclic graph model defined recursively as follows: (1) A tractable univariate distribution is an SPN. (2) A product of SPNs defined over different variables is an SPN. And (3), a weighted sum of SPNs with the same scope variables is an SPN. Thus, an SPN can be constructed with univariate distributions as leaves, sums and products as internal nodes, and the edges from a sum node to its children labeled with the corresponding weights. A valid SPN is an SPN that encodes a consistent distribution, that is, the scope or set of variables reachable in the descendants of every node is identical for sum nodes and disjoint for product nodes.

Tractable Inference in SPNs: To compute probabilities in an SPN, we compute the values of the nodes starting at the leaves. Since each leaf is an univariate distribution, we set the evidence on those distributions, obtain the probabilities and evaluate bottom up. On product nodes, we multiply the values of the children nodes. On sum nodes, we sum the weighted values of the children nodes. The value at the root indicates the probability of the given configuration. To compute marginals, i.e., the probability of partial configurations, we set the probability at the leaves for those variables to 1 and then proceed as before. Conditional probabilities can then be computed as the ratio of partial configurations. To compute MPE states, we replace sum by max nodes and then evaluate the graph first with a bottom up pass, but instead of weighted sums we pass along the weighted maximum value. Finally, in a top down pass, we select the paths that lead to the maximum value, finding approximate MPE states (Poon and Domingos 2011). All these operations traverse the tree at most twice and therefore can be achieved in linear time with respect to the size of the SPN.

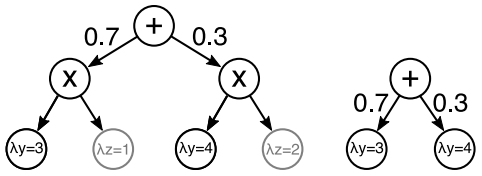

Figure 1: A PSPN over y, z **(left)** and the corresponding marginalized PSPN for y **(right)**. Tractable inference can be performed using symbolic evaluation. **(left)** $P(y,z) = 0.7\left((3^y e^{-3})/y!\ (1^z e^{-1})/z!\right) + 0.3\left((4^y e^{-4})/y!(2^z e^{-2})/z!\right)$. **(right)** $P(y) = 0.7\left((3^y e^{-3})/\ y!\right) + 0.3\left((4^y e^{-4})/y!\right)$. Univariate Poissons for y are shown in black, and for z in gray.

Learning SPNs: One way of learning SPNs is to consider a valid dense SPN and learn the weights using e.g. backpropagation (Poon and Domingos 2011). On convergence, zero weight edges are pruned. To overcome potential gradient diffusion problems (Bengio 2009) one may train in a discriminative fashion or use "max product" gradients (Gens and Domingos 2012). One may also learn a Bayesian resp. Markov network and compile it into an SPN, see e.g. (Rooshenas and Lowd 2013). Here, we focus on a top-down approach that directly learns the structure of (tree) SPNs (Gens and Domingos 2013) using three steps: (1) base case, (2) decomposition and (3) conditioning. In the base case, if only one variable remains, the algorithm learns a univariate distribution and terminates. In the decomposition step, the algorithm tries to partition the variables into independent components $V_j \subset \mathbf{V}$ such that $P(\mathbf{V}) = \prod_j P(V_j)$ and recurses on each component, inducing a product node. If the conditions for the base case or for the decomposition step are not satisfied, then we partition the training instances into clusters, inducing a sum node, and recurse on each part.

For multinomial and Gaussian distributions, standard approaches can be used for conditioning and decomposition. For conditioning, Gens and Domingos (2013) proposed to use of hard EM. That is, we assume a naive Bayes mixture model of independent variables conditioned on the cluster: $P(\mathbf{V}) = \sum_c P(C_c) \prod_j P(V_j|C_c)$, for a given cluster C_c and variable V_j. After obtaining the assignments to the clusters, the weights of the sum nodes are given by the mixing proportions $P(C_c)$. For decomposition, they propose to use a test based on ML scores, the G-test for independency.

From this we can create an undirected graph where there is an edge between V_i and V_j if the value $G(V_i, V_j)$ passes a threshold of significance. We then search for connected components and reject the feature partition if we find a single connected component in the graph.

Poisson Sum-Product Networks

Unfortunately, multivariate binomial, multinomial and Gaussian distributions—the targets of regular SPNs so far (Poon and Domingos 2011; Vergari, Di Mauro, and Esposito 2015)—are not well suited for modeling count data. Also, learning a regular SPN on transformed data using e.g. the element-wise $\log(x+1)$ is likely to result in poor

approximations (O'Hara and Kotze 2010). Consequently, we now propose conditioning and decomposition techniques tailored towards Poissons SPNs.

Multivariate Poisson Clustering for Conditioning: Clusters of multivariate count data may be found in a number of ways. Here, we propose to cluster multivariate Poisson data by mixtures $P(\mathbf{V}) = \sum_c P(C_c)\,\text{PDN}_c(\mathbf{V}|C_c)$ of PDNs with $\mathbf{V}$ being the count variables and C_c being the assignment to PDN_c representing the c-th cluster as PDN. More precisely, for learning the mixtures, we follow a stochastic EM approach[2]. We fix the cluster PDNs and update the cluster assignments. This follows standard formulas for mixtures. Then, we fix the cluster assignments and apply gradient tree boosting to improve the cluster PDNs. The boosting formulae are adaptions of the ones for boosting PDNs (Hadiji et al. 2015).

These PDNs can be built using regular regression trees, however, they just fit empirical averages in each node, hence, tend to produce large trees and have a variable selection bias (e.g., towards partitioning variables with many potential splits). To overcome this, Zeileis *et al.* (2008) introduced model trees for generalized linear models (GLMs, McCullagh and Nelder (1989)). They are unbiased and employ a parameter instability tests for detecting differences in GLMs across terminal nodes. To learn them for $P(V_i|\mathbf{V}_{\setminus i})$, we initially consider $\mathbf{S} = \mathbf{V}_{\setminus i}$ as set of partitioning variables and cycle iteratively through the following steps: (1) Fit an GLM to the data using maximum likelihood. (2) Test for parameter instability over $\mathbf{S}$: if the (possibly Bonferroni corrected) significance value for any $V \in \mathbf{S}$ falls below some threshold p, choose the variable $S \in \mathbf{S}$ associated with the smallest significance value for partitioning, otherwise stop. (3) Find the split in S with highest improvement of the model fit. (4) Split the dataset correspondingly and repeat the steps for each resulting subgroup.

Here, step (2) employs parameter instability tests (Zeileis and Hornik 2007). For the Gaussian case they include many well-established tests from the literature as special cases (including tests based on ML scores, F statistics and OLS residuals) but yielded novel tests for Poisson data.

They consider the gradients of the log-likelihood of $P(V_i|\mathbf{V}_{\setminus i})$ for each of the l training examples, the so called scores. By definition, the empirical scores of all observations in a dataset should sum to zero, and when the model is correctly specified, the expected value of the score for each observation is also zero. Under the null hypothesis of parameter stability, the scores do not systematically deviate from the expected value of zero, when the observations are ordered by the values of $V_j \in \mathbf{V}_{\setminus i}$. To test whether the scores systematically deviate from zero, the generalized M-fluctuation process $\text{epf}_j(k/l)$ is used (Zeileis and Hornik 2007), $\text{epf}_j(t) =$

$$\Big(\underbrace{\sum_{k=1}^{l} \lambda_i \mathbf{v}_{\setminus i}^{k} \mathbf{v}_{\setminus i}^{k\top}}_{\text{covariance matrix of scores}}\Big)^{-0.5} \Big(\underbrace{\sum_{k=1}^{\lfloor nt \rfloor} (v_i^{\sigma(kj)} - \lambda_i)\mathbf{v}_{\setminus i}^{\sigma(kj)}}_{\text{accumulated permuted scores}}\Big)$$

where $t \in [0,1]$, v_i^u resp. $\mathbf{v}_{\setminus i}^u$ are the u-th observed value of V_i resp. $V_{\setminus i}$, $\sigma(kj)$ is the ordering permutation which gives the antirank of the observations of V_j, and λ_i the estimated mean of $\mathbf{V}_i$. Thus, $\text{epf}_j(t)$ is the partial sum process of the scores ordered by V_j, scaled by the number of observations and the covariance. To aggregate it into a scalar test statistics, the "supLM" statistic (Andrews 1993)

$$\lambda_{\text{supLM}}(j) = \max_{k=\underline{k},\ldots,\overline{k}} \left(\frac{k}{l}\frac{l-k}{l}\right)^{-1} \left\| \text{epf}_j\left(\frac{k}{l}\right)\right\|_2^2$$

in the interval $[\underline{k}, \overline{k}]$ is used. It is the maximum of the squared L_2-norm of the fluctuation process scaled by its variance function. Its limiting distribution is given by the supremum of a squared, k-dimensional tied-down Bessel process from which significance values can be computed (Hansen 1997).

Decomposition via Instability Tests: Independence may be found in a number of ways. Given that we have Poisson GLM model trees at hand, one sensible idea is to estimate them for $P(V_i|\mathbf{V}_{\setminus i})$ and $P(V_j|\mathbf{V}_{\setminus j})$ and check whether V_i resp. V_j is below a significance threshold p in corresponding trees. If in both cases not, then V_i and V_j are independent.

Specifically, the model trees provide significance values $p_{i|j}$ of how much V_i depends on V_j. We aggregate them into $p_{ij} = \min(p_{i|j}, p_{j|i})$ and create an undirected graph with an edge between V_i and V_j if p_{ij} is below a given threshold p. On the graph, we proceed as when learning regular SPNs.

Structure Learning Tree PSPNs: Using the Poisson conditioning and decomposition steps result in the structure learning approach summarized in Alg. 1; it learns valid PSPNs with normalized weights on the sum nodes. Since

Algorithm 1 Tree PSPN Structure Learning. The inputs are data D, instance indexes I, feature indexes F, minimum number of instances for splitting m, and strength of dependency p value. The output is a tree PSPN S.

procedure LEARNPSPN(D, I, F, p, m)
  if $|F| = 1$ **then**
    return $\text{Pois}(\lambda \leftarrow (1/|I|)\sum_i D[I_i, F])$
  if $|I| \leq m$ **then**
    $\lambda_j \leftarrow 1/|I| \sum_i D[I_i, F_j]$
    return $\text{ProductNode}(\text{Pois}(\lambda_1), \text{Pois}(\lambda_2), \ldots)$
  $P \leftarrow \text{partitionFeat}(D, I, F, p)$ ▷ Instability Test
  if $|P| = 1$ **then**
    $C_i \leftarrow \text{clusterInst}(D, I, F)$ ▷ PDN Clustering
    $S_c \leftarrow \text{LearnPSPN}(D, C_i, F, p, m)$
    return $\text{SumNode}\Big(\frac{|C_1|}{|I|}S_1, \frac{|C_2|}{|I|}S_2, \ldots\Big)$
  else
    $S_j \leftarrow \text{LearnPSPN}(D, I, P_j, p, m)$
    return $\text{ProductNode}(S_1, S_2, \ldots)$

[2] The approach itself is a contribution and useful for clustering multivariate count data, independently of PSPNs. Since it relies on GLM model trees only, it can be used for other data distributions beyond Poisson, even hybrid ones, in contrast to Poisson-only alternatives such as admixtures of PMRFs (Inouye *et al.* (2014)).

the clustering algorithm always splits the current set of instances, at every iteration we reduce the amount of data left to work with and the algorithm always converges. We stop growing the PSPN, if there are less than m instances left, so that the univariate distributions have enough data to fit.

PSPNs are Hierarchical Topic Models:

Product nodes provide part-based representations of the features, sum nodes provide mixtures of these parts, and the base case typically consist of products of a large number of independent Poissons. This is not easy to understand visually. Luckily, the product of independent Poissons is equivalent to a Poisson-Multinomial (Bishop, Fienberg, and Holland 2007),

$$\prod_{j=1}^{n} P_{\text{Pois}}(v_i|\lambda_i) = \frac{e^{-\tilde{\lambda}}}{\tilde{v}!}\tilde{\lambda}^{\tilde{v}}\frac{\tilde{v}!}{\prod_{j=1}^{n} v_j!}\prod_{j=1}^{n}\left(\frac{\lambda_j}{\tilde{\lambda}}\right)^{v_j}$$
$$= P_{\text{Pois}}\left(\tilde{v}|\tilde{\lambda}\right) P_{\text{Mult}}\left(v|\theta = (\lambda_1, \cdots, \lambda_n)/\tilde{\lambda}, N = \tilde{v}\right)$$

where $\tilde{\lambda} = \sum_{j=1}^{n} \lambda_i$ and $\tilde{v} = \sum_{j=1}^{n} v_i$. Thus we can compress any terminal product node by a single Poisson-Multinomial distribution and, in turn, reduce their visual complexity. More importantly, the Poisson-Multinomial is nothing else than the conditional distribution of Latent Dirichlet Allocation (Inouye, Ravikumar, and Dhillon 2014). In other words, PSPNs are essentially hierarchical topic models with tractable inference: terminal product nodes are basic topics; inner product nodes are together "with" and sum nodes are mutual exclusive "or" combinations of more refined topics. The expected number of words per topic is given by the corresponding $\tilde{\lambda}$ associated with edges going out from mutual exclusive "or" nodes. We refer to Fig. 2 for an illustration.

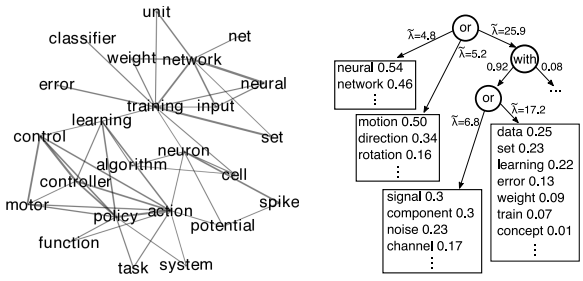

Figure 2: Understanding the NIPS PSPN. **(left)** Mutual information (MI) between some words (the darker,the higher). **(right)** Encoded hierarchical topics (only top words shown). Papers spend many words on "learning" and less many on what is learned such as "neural networks" or "motions".

Tractable Inference via Symbolic Evaluation: By definition, PSPNs encode polynomials over univariate Poissons and we can encode it in a symbolic equation to be evaluated by a symbolic solver for inference as illustrated in Fig. 1. This is realized in Alg. 2. These resulting symbolic representation can be optimized an then compiled to C or CUDA, for even faster inference. They can also be composed under infinite sums to be evaluated by numerical solvers (Graves-Morris, Roberts, and Salam 2000; Weniger 2003). This way one can e.g. compute expectations $E[X] = \sum_i^{\infty} x_i P(x_i)$ and information-theoretic measures.

Algorithm 2 Inference via symbolic evaluation. For variables $\mathbf{V}$ it computes recursively a symbolic equation for $P(\mathbf{V})$ as encoded by the PSPN rooted by node N.

procedure MARGINALIZETOEQ($\mathbf{V}, N$)
  if N is SumNode **then**
    $eq \leftarrow 0$
    for all $c \in$ children$(N), w \in$ weights(N) **do**
      $eq \leftarrow eq + w *$ MARGINALIZETOEQ$(\mathbf{V}, c)$
    return eq
  else if N is ProductNode **then**
    $eq \leftarrow 1$
    for all $c \in$ children(N) **do**
      $eq \leftarrow eq *$ MARGINALIZETOEQ$(\mathbf{V}, c)$
    return eq
  else if N.variable $\in \mathbf{V}$ **then return** 1
  else return $\left(e^{-N.\lambda} N.\lambda^{N.\text{variable}}\right)/N.\text{variable!}$

Interpreting and understanding the gist of PSPNs: While the structure of an induced PSPN may provide interesting insights into the data, the alternation of conditioning and decomposition is not easy to interpret for non-experts. So, how can we best extract the gist of PSPNs? Using symbolic evaluation of tree PSPNs this is easy. We can directly compute information-theoretic measures. This is difficult, if not impossible, to compute them for previous Poisson GMs.

Experimental Evaluation

We aim to evaluate the benefits of PSPNs compared to other PGMs by answering the following questions:

(Q1) Is the PSPN distribution a flexible multivariate Poisson distribution? **(Q2)** Can PSPNs find independencies of count variables well compared to a state-of-the-art Poisson graphical model? **(Q3)** Are there benefits of having tractable inference for multivariate Poisson distributions? **(Q4)** Is there a benefit of having a deep architecture for learning these distributions? **(Q5)** Can PSPNs be used for other machine learning tasks?

We implemented PSPNs in Python for growing Poisson GLM model trees. All experiments ran on a Linux machine with 56 cores, 4 GPUs and 512GB RAM.

To speed up decomposition, we grew Poisson GLM model trees of depth 1 and used the significance values of all the considered potential tests.

(Q1) Flexible Multivariate Poisson Distribution: PSPNs are motivated by the need for a flexible multivariate count distribution. While other Poisson graphical models allow flexible dependencies between count variables, they do not provide tractable inference. So, does the tractability of PSPNs make them less flexible? Fig. 3 shows that this is not the case. The PSPN distribution allows flexible dependencies between variables. They can even capture univariate mixtures of Poissons and, hence, are more flexible than PMRFs. We can answer **(Q1)** affirmatively.

(Q2) Network Discovery from Simulated Data: Do

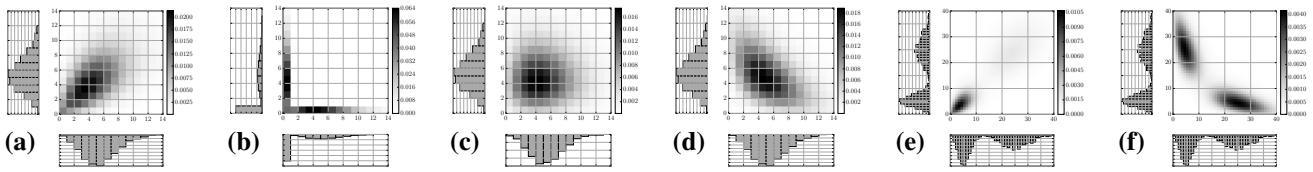

Figure 3: PSPNs are flexible multivariate Poisson distributions. Shown are the densities and marginals of six 2D Poisson SPNs with different dependencies between the two count variables. (a) Positive dependency suggests that two events often co-occur, (b) negative dependency that two events rarely co-occur, (c) zero dependency that events are not correlated, and (d) anti dependency that if one event occurs the other one does not occur. Mixture with positive (e) and anti (f) dependencies.

PSPNs recover the network structure well? To investigate this, we used the "XMRF" R-package[3] for both, the data generation and comparison. It contains an implementation of the models described in (Yang et al. 2013). The datasets were generated as follows: Using XMRF.SIM, multivariate count data with $100, 500, 1000, 2000, 4000$ instances were sampled using LPGMs with $n = 5, 10, 20, 50, 100$ nodes; we created 50 graphs of mixed "hub" and "scale-free" structured models. We then estimated LPGMS and shallow 1-level PSPNs. The adjacency matrix of the original LPGM was used to measure the network discovery performance. As Fig. 4**(a)** shows, the PSPN decomposition step recovers the networks well and is not very sensitive to the significance parameter p. This answers **(Q2)** affirmatively.

As there was no difference in performance among the significance thresholds p, we fixed $p = 0.001$ in all remaining experiments, which are 5**(b)** and 10**(c)**-fold cross-validated.

(Q3) Predictive Power on Benchmark Datasets: We investigated the predictive performance on the previous synthetic datasets as well as on the UCI datasets used in (Hadiji et al. 2015). The **Communities & Crime** (C&C) dataset was obtained from 2,215 communities in the United States, reporting different crime statistics. We focus on instances with complete data and target count values specifying crimes such as the number of robberies, burglaries, and others. Our cleaned dataset contains 1,902 communities with 8 features. The **NIPS** bag-of-words dataset contains $1,500$ documents with a vocabulary above 12k words. We considered the 100 most frequent words. The **MSNBC** is a count representation of *MSNBC.com* dataset from the UCI repository. The data gives sequences corresponding to a user's page views for an entire day, which are grouped into 17 categories. We used the post-processed version from the *SMPF* library[4], which removed very short click sequences. In total, this dataset contains information of about 31,790 users. We analyzed solely the frequencies of the visited categories. In contrast to the C&C dataset, the means of the 17 categories have all low mean values. We also note that the variance in this data is much lower than in the C&C-dataset.

The predictive performances are summarized in Fig. 4**(b)**. For Poisson GMs, we compared only to PDNs since (Hadiji et al. 2015) showed empirically that they are competitive to LPGMs, and in terms of likelihood, LPGMs are competitive to SPGMs and TPGMs (Yang et al. 2013). PSPNs are competitive to GSPNs and comparable to PDNs. This provides an affirmative answer to **(Q3)**. In contrast to PDNs, inference for PSPNs is tractable, and hence more general queries can be answered more efficiently as considered next.

(Q3) Poisson Mutual Information: In PDNs, (Hadiji et al. 2015) suggested the use of relative influence to interpret the structure. This is a measure based on the number of times a variable is selected for splitting in the regression trees, weighted by the squared improvement to the model as a result of each split, and averaged over all trees. Unfortunately, this has no clear information-theoretic meaning. Since PSPNs provide tractable inference, we propose to use mutual information (MI), as a way to interpret the PSPNs and to focus attention by considering relevant variable associations only. Fig. 2**(left)** shows the MI network induced over the NIPS corpus. There are natural groupings of words. This provides an affirmative answer to **(Q3)**.

(Q3,4,5) Shallow versus Deep Architectures: Poisson trees are actually (2-levels) shallow learners. They can be written as a sum-product network where each product (over the edges from the root to a leaf) selects a leaf. Boosting them, as done in PDNs, adds one level of depth, which may explain the good predication performance of PDNs; Moreover, a product of independent Poissons (a single product node) encodes the conditional distribution of a Latent Dirichlet Allocation (LDA) model. Thus, LDA is also a rather shallow learner, and hierarchical variants add some depth. Since both are popular models for count data, we compared PSPNs to them in order to gain further insights into having a benefit of a deep architecture. We computed the predictive perplexity, which is a decreasing function of the log-likelihood of unseen examples (the lower, the better). Unfortunately, the LDA likelihood of one document is intractable, which makes the evaluation of the perplexity intractable, too, and one has to resort to approximative inference. For PSPNs, this is tractable using symbolic evaluation. Fig. 4**(d)** summarizes the results. As one can see, PSPNs outperform the shallow LDA models, are competitive to PDNs, and in contrast to PDNs, provide naturally hierarchical topics as illustrated in Fig. 2**(right)**. Only the run time is considerably higher for PSPNs. Therefore we also evaluated a randomized PSPN learner (RPSPN): the decomposition is performed on a subsample of 2000 examples and the conditioning is using a random hyperplane after taking the square root

[3] *https://cran.r-project.org/web/packages/XMRF/index.html*

[4] *http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php*

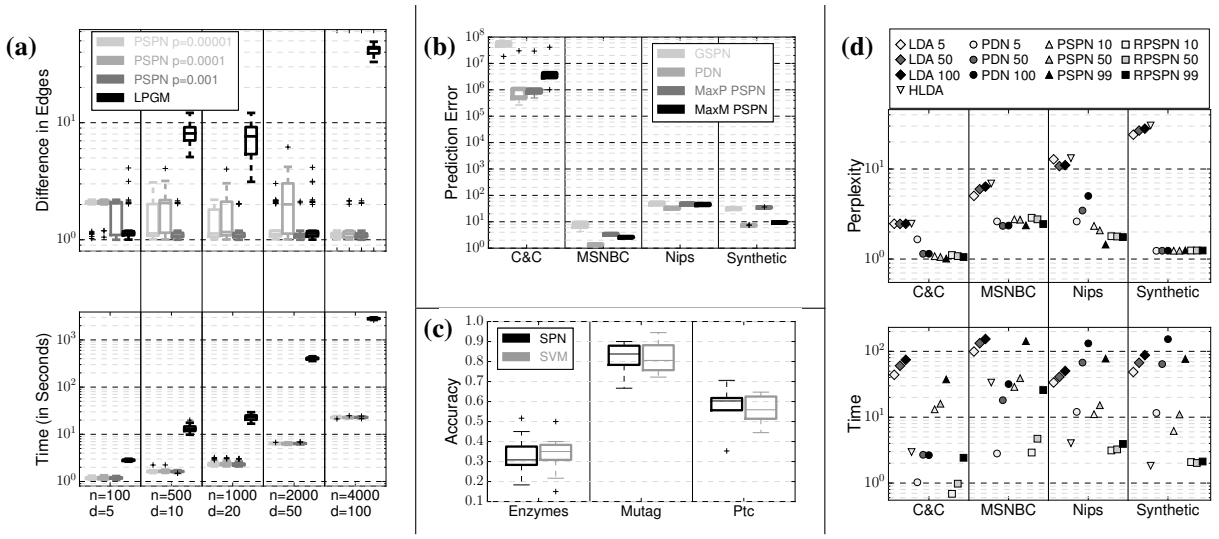

Figure 4: Experimental evaluation of PSPNs. **(a, top)** Comparison of network discovery performance (the lower, the better) between 1-level PSPN and LPGM. Differences in the recovered graphs from the original ones are measured by the number of different edges. **(a, bottom)** The corresponding running times in seconds. n is the number of training instances and d is the number of nodes. **(b)** Predictive performance (the lower, the better) of PSPNs ($m = 80$) on several multivariate count data benchmarks. MaxP denotes max-product, MaxM maximum-marginal inference, and GSPN, a Gaussian SPN. A Gaussian treatmeant can result in an order of magnitude large predictive error. The predictive performance of PSPNs is comparable with PDNs. **(c)** Deep graph classification using PSPNs ($m = 80$). Training a PSPN using the explicit feature maps of the Weisfeiler-Lehman graph kernel is comparable to feeding them into a SVM, the standard approach. Shown are the predictive accuracies (the higher, the better). **(d, top)** Predictive perplexities (the lower, the better) of Hierarchical LDA, LDA, PDN, PSPN and randomized PSPN (RPSPN). Models are getting deeper from left to right as indicated by the numbers next to model names: (H)LDA numbers of topics; PDNs number of boosting iterations; (R)PSPNs $m = (100 - \text{number}) \cdot \text{dataset size}$. As one can see, PSPNs outperform (Hierarchical) LDA and are competitive to PDNs. **(d, bottom)** Learning PDNs is faster than learning PSPN. RPSPNs are competitive in predictive performance and run time. (Best viewed in color)

of the sum-to-1 normalized examples. Fig. 4**(d)** shows that the performance drops slightly but training is significantly faster. Therefore **(Q3,4)** can be answered affirmatively.

(Q5) Graph classification: To further investigate **(Q4)**, we considered the task of graph classification. The Weisfeiler-Lehman graph kernel (Shervashidze et al. 2011) proceeds in iterations. Starting e.g. with the degree of a node as its label, it augments the node labels by the sorted set of node labels of neighbouring nodes, and compresses these augmented labels into new, short labels. Then, it computes the histogram of the new labels and repeats the steps for a pre-specified number of iterations. Typically, these histograms are fed into a SVM for classification. Here we trained a PSPN per class and merged them with a Sum Node with weights as priors on the classes. We compared both approaches on three graph data benchmarks. **Mutag** (Debnath et al. 1991) is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds with 2 class labels. PTC is a dataset of 344 chemical compounds that reports the carcinogenicity for male and female rats and it has 2 classes (Toivonen et al. 2003). **Enzymes** is a data set obtained from (Borgwardt et al. 2005) consisting of 600 enzymes. The goal is to assign each enzyme to one of the 6 EC top-level classes. As the results summarized in Fig. 4**(c)** show, PSPNs are comparable to SVMs, which answers **(Q5)** affirmatively.

Conclusion

Poisson sum-product networks (PSPNs) can be viewed as a deep combination of multivariate Poisson mixture models and feature hierarchies. In contrast to other Poisson graphical models (GMs), inference in PSPNs is tractable. As our experimental results demonstrate, the performance of the estimated PSPNs is competitive to other Poisson GMs, also compared to LDA and graph classification via SVMs.

We aim to explore other learning methods such as randomized and boosted ones, and mixtures of PSPNs along further applications. Using different types of GLM trees, PSPNs directly provide a deep and tractable version of Manichean graphical models (Yang et al. 2014), making them useful in many other real applications.

Acknowledgements: The authors would like to thank the anonymous reviewers for their feedback. AM and KK gratefully acknowledge the supported by the DFG Collaborative Research Center SFB 876 projects A6 and B4, SN the support of CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Any opinions, findings and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, ARO or the US government.

References

Allen, G. I., and Liu, Z. 2013. A local poisson graphical model for inferring networks from sequencing data. *IEEE Transactions on Nanobioscience* 12(3):189–198.

Andrews, D. W. K. 1993. Tests for Parameter Instability and Structural-Change with Unknown Change-Point. *Econometrica* 61:821–856.

Bengio, Y. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.* 2(1):1–127.

Bishop, Y. M.; Fienberg, S. E.; and Holland, P. W. 2007. *Discrete Multivariate Analysis: Theory and Practice*. Springer.

Borgwardt, K.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. In *Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA, 25-29 June 2005*, 47–56.

Debnath, A.; Lopez de Compadre, R.; Debnath, G.; Shusterman, A.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medical Chemistry* 34:786–797.

Gens, R., and Domingos, P. M. 2012. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012.*, 3248–3256.

Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In Dasgupta, S., and Mcallester, D., eds., *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, 873–880. JMLR.

Graves-Morris, P. R.; Roberts, D. E.; and Salam, A. 2000. The epsilon algorithm and related topics. *Journal of Computational and Applied Mathematics* 122:51–80.

Hadiji, F.; Molina, A.; Natarajan, S.; and Kersting, K. 2015. Poisson Dependency Networks: Gradient Boosted Models for Multivariate Count Data. *MLJ* 100(2):477–507.

Hansen, B. E. 1997. Approximate Asymptotic P Values for Structural-Change Tests. *Journal of Business & Economic Statistics* 15(1):60–67.

Henao, R.; Gan, Z.; Lu, J.; and Carin, L. 2015. Deep poisson factor modeling. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2800–2808.

Inouye, D.; Ravikumar, P.; and Dhillon, I. 2014. Admixture of Poisson MRFs: A Topic Model with Word Dependencies. *Journal of Machine Learning Research* 683–691.

Inouye, D.; Ravikumar, P.; and Dhillon, I. 2015. Fixed-length poisson MRF: adding dependencies to the multinomial. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 3213–3221.

Inouye, D.; Ravikumar, P.; and Dhillon, I. 2016. Square root graphical models: Multivariate generalizations of univariate exponential families that permit positive dependencies. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, 2445–2453.

Lauritzen, S. 1996. *Graphical Models*. Oxford Statistical Science Series. Clarendon Press.

McCullagh, P., and Nelder, J. 1989. *Generalized Linear Models*. Chapman & Hall.

O'Hara, R. B., and Kotze, D. J. 2010. Do not log-transform count data. *Methods in Ecology and Evolution* 1(2):118–122.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 689–690.

Ranganath, R.; Tang, L.; Charlin, L.; and Blei, D. M. 2015. Deep exponential families. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Rooshenas, A., and Lowd, D. 2013. Learning markov networks with arithmetic circuits. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS*. JMLR Workshop and Conference Proceedings.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E.; Mehlhorn, K.; and Borgwardt, K. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12:2539–2561.

Toivonen, H.; Srinivasan, A.; King, R.; Kramer, S.; and Helma, C. 2003. Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics* 19(10):1183–1193.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2015. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal*. Springer. chapter Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning, 343–358.

Weniger, E. J. 2003. Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series. *ArXiv math/0306302*.

Yang, E.; Ravikumar, P.; Allen, G.; and Liu, Z. 2012. Graphical models via generalized linear models. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 1367–1375.

Yang, E.; Ravikumar, P.; Allen, G. I.; and Liu, Z. 2013. On poisson graphical models. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 1718–1726.

Yang, E.; Baker, Y.; Ravikumar, P.; Allen, G.; and Liu, Z. 2014. Mixed graphical models via exponential families. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1042–1050.

Zeileis, A., and Hornik, K. 2007. Generalized M-fluctuation tests for parameter instability. *Statistica Neerlandica* 61(4):488–508.

Zeileis, A.; Hothorn, T.; and Hornik, K. 2008. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics* 17(2):492–514.

Zhou, M.; Cong, Y.; and Chen, B. 2015. The poisson gamma belief network. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 3043–3051.