Generative Clausal Networks: Relational Decision Trees as Probabilistic Circuits

Fabrizio Ventola[1]*, Devendra Singh Dhami[1] *, and Kristian Kersting[1,2]

[1] Department of Computer Science, TU Darmstadt, Germany
[2] Hessian Center for AI and Centre for Cognitive Science, Darmstadt, Germany
{ventola, devendra.dhami, kersting}@cs.tu-darmstadt.de

Abstract. In many real-world applications, the i.i.d. assumption does not hold and thus capturing the interactions between instances is essential for the task at hand. Recently, a clear connection between predictive modelling such as decision trees and probabilistic circuits, a form of deep probabilistic model, has been established although it is limited to propositional data. We introduce the first connection between relational rule models and probabilistic circuits, obtaining tractable inference from discriminative rule models while operating on the relational domain. Specifically, given a relational rule model, we make use of Mixed Sum-Product Networks (MSPNs)—a deep probabilistic architecture for hybrid domains—to equip them with a full joint distribution over the class and how (often) the rules fire. Our empirical evaluation shows that we can answer a wide range of probabilistic queries on relational data while being robust to missing, out-of-domain data and partial counts. We show that our method generalizes to different distributions outperforming strong baselines. Moreover, due to the clear probabilistic semantics of MSPNs we have informative model interpretations.

Keywords: Statistical relational learning · Tractable probabilistic models · Rule learning

1 Introduction

Relational decision trees (RDTs) [3] have been the workhorse in modern inductive logic programming for many years and form the backbone of several relational machine learning models. Recent work on the interpretation of Decision Trees as generative models [9] shed light on the surprising yet intuitive tight connection between robust discriminative models and powerful tractable generative models. This paves the way for the application of several methods and techniques designed for Decision Trees to the world of Probabilistic Circuits and vice-versa. However, in many real-world applications, the i.i.d. assumption does not hold and capturing the interactions between instances is fundamental for an arbitrary task such as classification or generation of new samples. Inspired by this, we aim to revisit RDTs using the techniques of principled propositionalization and a probabilistic interpretation.

Statistical Relational Learning [17, 11] models have been proposed to overcome the rigidity of well-known first-order rule learners such as TILDE [3] since they cannot

[*] Equal contribution

naturally deal with uncertainty. However, most of these models such as Markov Logic Networks [34] and Relational Dependency Networks [29] are difficult to scale and their inference process is generally intractable. Recent approaches have tried to tackle intractability by taking advantage of Arithmetic Circuits [10] representations that, under certain conditions, especially by imposing particular constraints on their structure, can guarantee to provide tractable inference for a set of probabilistic queries. Nevertheless, these models have been mostly designed for propositional data and contributions for relational domain [14, 21, 28] are rather limited since they need specific input representation and make strong assumptions on the type of data distributions.

Specifically, inspired by [9] and considering the aforementioned limitations of SRL models, we introduce the first connection from relational rule models, specifically relational decision trees, to probabilistic circuits, precisely Sum-Product Networks (SPNs). With this aim, we present *Generative Clausal Networks (GCLNs)* as tractable generative models that can model the joint distribution of counts of the firing of rules (clauses) and can be powerful discriminators at the same time. As a discriminator, GCLNs are both accurate and robust to missing data. As probabilistic generators, they can be used for accurate data imputation, out-of-domain (OOD) detection, sample new data and predicate invention. Moreover, thanks to the probabilistic semantics, GCLNs are easy to interpret. GCLNs can also be seen as alternatives to relational Naive Bayes [22, 14] since instead of learning a naive Bayes over clauses as features, we first learn all rules using an RDT and then turn it into joint distribution.

We make the following contributions: 1. We propose the first set of models that take advantage of RDT structure to learn powerful probabilistic circuits. 2. We take advantage of SPNs to learn both conditionals (discriminative) and joint (generative) models. 3. We show that our model is robust to noise, OOD data and missing values and takes advantage of both worlds of relational models and probabilistic circuits. 4. We show that our model is interpretable due to the use of first-order logic and SPNs.

2 Background and Related Work

Probabilistic Circuits: Sum-Product Networks (SPNs) are tractable deep density estimators [33] and they are part of the family of Probabilistic Circuits. SPNs can be seen as a deep extension of a particular class of Arithmetic Circuits [10] that encode probability distributions. They have been successfully applied on domains such as computer vision [41], natural language processing [7] and speech recognition [31].

Definition of SPNs. An SPN S, see figure 1, is a computational graph defined by a rooted DAG, encoding a probability distribution[3] $P_{\boldsymbol{X}}$ over a set of RVs $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, where inner nodes can be either weighted sum or product nodes over their children (graphically denoted respectively as $\bigoplus$ and $\bigotimes$), and leaves are valid distributions defined on a subset of the RVs $\boldsymbol{Z} \subseteq \boldsymbol{X}$. Each node $n \in S$ has a *scope* $\mathsf{scope}(n) \subseteq \mathbf{X}$, defined as the set of RVs appearing in its descendant leaves. The subnetwork S_i, rooted at node i, encodes a distribution over its scope i.e. $S_i(\boldsymbol{x}) = P_{\boldsymbol{X}_{|scope(i)}}(\boldsymbol{x})$ for each

[3] We are not strict on “density” vs. “distribution”.

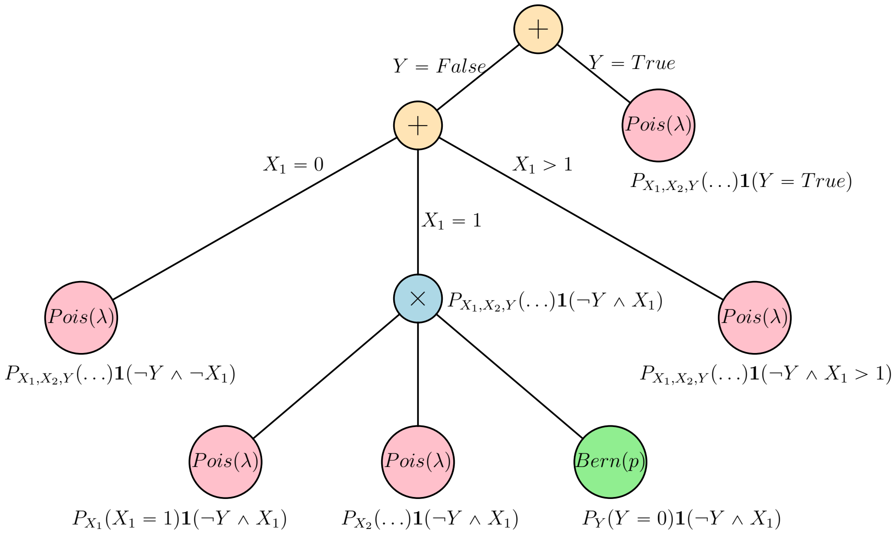
$+$
$Y = False$
$Y = True$
$Pois(\lambda)$
$P_{X_1,X_2,Y}(\ldots)\mathbf{1}(Y = True)$
$+$
$X_1 = 0$
$X_1 = 1$
$X_1 > 1$
$\times$
$P_{X_1,X_2,Y}(\ldots)\mathbf{1}(\neg Y \wedge X_1)$
$Pois(\lambda)$
$P_{X_1,X_2,Y}(\ldots)\mathbf{1}(\neg Y \wedge \neg X_1)$
$Pois(\lambda)$
$P_{X_1,X_2,Y}(\ldots)\mathbf{1}(\neg Y \wedge X_1 > 1)$
$Pois(\lambda)$
$Pois(\lambda)$
$Bern(p)$
$P_{X_1}(X_1 = 1)\mathbf{1}(\neg Y \wedge X_1)$
$P_{X_2}(\ldots)\mathbf{1}(\neg Y \wedge X_1)$
$P_Y(Y = 0)\mathbf{1}(\neg Y \wedge X_1)$

Fig. 1. A graphical interpretation of Generative Clausal Networks (GCLN). They are Sum-Product Networks (SPNs), where sum nodes (in yellow, weights are omitted here for simplicity) act on the number of true ground instances of clauses, while product nodes (in blue) can be view as realizing predicate conjunctions. Leaves consist of two types. Poisson distributions (in pink) encode exchangeable distribution templates and Bernoulli (in green) models the target class (some are omitted for simplicity). The symbol $\mathbf{1}()$ represents an indicator function. The root node and its children model the joint distribution $P(\boldsymbol{X}, Y)$ where $\boldsymbol{X}$ is the set of features and Y is the target variable. In this example, the root node introduces a latent variable that can be interpreted as a binary literal (best viewed in color).

$\boldsymbol{x} \sim \boldsymbol{X}_{|scope(i)}$. Each edge (i, j) emanating from a sum node i to one of its children j has a non-negative weight w_{ij}, with $\sum_j w_{ij} = 1$. Weighted sum nodes represent a mixture over the probability distributions encoded by their children, while product nodes represent factorizations over contextually independent distributions. Thus, an SPN can be viewed as a deep hierarchical mixture model, where the hierarchy is based on the scope of the nodes w.r.t. the whole set of RVs $\boldsymbol{X}$. In a valid SPN, the probability assigned to a given state $\boldsymbol{x}$ of the RVs $\boldsymbol{X}$ can be read out at the root node, and will be denoted $S(\boldsymbol{x}) = P_{\boldsymbol{X}}(\boldsymbol{X} = \boldsymbol{x})$.

Inference using SPNs. Given an SPN S, $S(\boldsymbol{x})$ can be computed by evaluating the network bottom-up. When evaluating a leaf node i concerning variable X_j, $S_i(x_j)$ corresponds to the probability of that state $P_i(X_j = x_j)$. The value of a product node corresponds to the product of its children's values: $S_i(\boldsymbol{x}_{|\mathsf{scope}(i)}) = \prod_{i \to j \in S} S_j(\boldsymbol{x}_{|\mathsf{scope}(j)})$; while, for a sum node, its value corresponds to the weighted sum of its children's values: $S_i(\boldsymbol{x}_{|\mathsf{scope}(i)}) = \sum_{i \to j \in S} w_{ij} S_j(\boldsymbol{x}_{|\mathsf{scope}(j)})$. All the exact marginal and conditional probabilities (with different amount of evidence), the exact partition function, and even approximate MPE queries and states can be computed in time linear in the *size* of the network i.e. its number of edges [30].

Structure Learning of SPNs. The prototypical structure learning algorithm for SPNs is LEARNSPN [16] which is a greedy learning schema to infer both the structure and the parameters of an SPN from data by executing a top-down structure search in the space of *tree-structured* SPNs. The algorithm first tries to find context-specific independencies among random variables (RVs) by means of a statistical test. When successful, it learns a product node where children represent the discovered context-specific factorization. When the variable splitting fails, the algorithm tries to slice the data matrix by rows i.e. clustering instances. In this case, the sum node weights represent the proportion of instances that fall in the relative cluster. Termination happens when a data slice contains only one random variable or when the number of instances is lower than a threshold μ.

Relational Learning: Most of the real-world data is relational in nature and has to be converted to a propositional form in order to use classic machine learning algorithms. Although standard, this can result in a significant loss of information. Thus, there has been a lot of research in developing methods that can handle relational data. TILDE [3] are logical decision trees based on the divide-and-conquer strategy and can be used to obtain relational clauses from the learned trees. Various methods [19, 27] that propose relational learning and inference by using ensembles of these TILDE trees were also proposed. SPNs are inherently propositional, and model examples with an independent and identically distributed (i.i.d) assumption. Learning SPNs for relational domains [28] relaxed the i.i.d assumption by defining a set of object classes to model the relationship among various instances, along with learning a probability distribution over the features themselves. The key idea behind learning a relational SPN is the use of an aggregation statistic over variables to take advantage of symmetries existing in relational data.

Probabilistic Circuits for Count Data: Sum-Product Networks were originally proposed for univariate parametric distributions either in the form of Bernoulli or Gaussian distributions at leaves [33]. The state-of-the-art structure learning algorithm LEARNSPN and part of its variants have been developed with the assumption that the data originated from a specific form of a multivariate distribution. Since a considerable amount of real-world data follows the Poisson distribution, recent developments have been focused on learning the structure of an SPN by assuming count data such as Poisson SPNs [25]. Several other models [39, 13] have also been proposed that pertain specifically to count data.

3 Probabilistic Circuits over Logical Clauses

Using the expressive power of first-order logic, ILP systems can learn complex programs and discover relations between data instances. This is useful in many domains and real-world applications where i.i.d. assumption does not hold. Despite the expressive power and clear formalism, ILP systems cannot deal with uncertainty and, in general, do not provide tractable inference and learning. SRL models, such as MLNs, have been devised to enhance first-order logic learners with probability in order to deal with uncertainty and be more flexible. Still, the major part of SRL models do not allow for

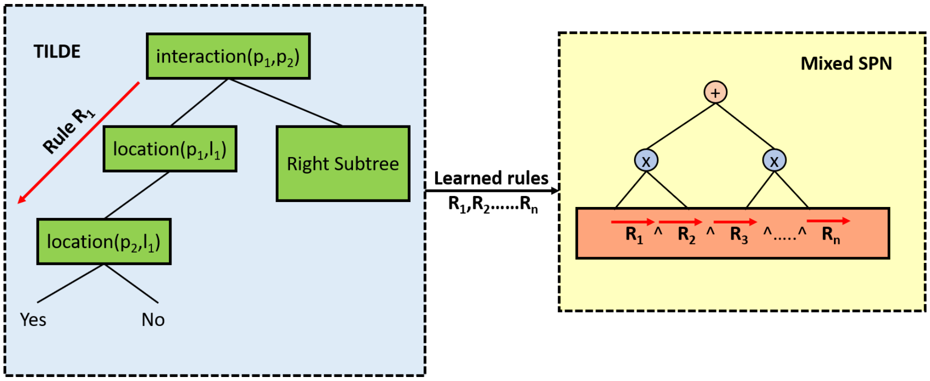

Fig. 2. An overview of learning the GCLN model (TILDE tree → rules → Mixed SPN on a conjunction of rules). The learned GCLN model is discriminative, generative and interpretable.

out-of-the-box tractable inference and tractable approximations have limited expressive power and generalize poorly. Thus, we draw a connection from logical clauses and their counts to Tractable Probabilistic Models [8] that allows, given an arbitrary first-order learner, to:

- Answer in *linear time* a wide range of probabilistic queries e.g. how likely one or more rules can fire together and how many times do the rules fire.
- Learn a powerful Tractable Probabilistic Discriminator that expresses *confidence* about its predictions and that is also *robust to missing and out-of-domain data*.
- *Generate* samples and new predicates.

Generative Clausal Networks. The generative clausal networks (GCLNs) capture the correlations among non mutually exclusive rules which can also share predicates and fire simultaneously. To learn the GCLNs, we start by learning a set of first-order rules by employing TILDE, a well-known top-down inducer of logical decision trees and we keep the counts on how many times a clause fires given a certain data sample. To obtain the counts, for every rule the first and last entity are instantiated and then all the predicates that completely satisfy the partially grounded rules are obtained. Then, we learn a joint distribution over these counts together with the target by encoding such distribution by means of a prominent deep tractable density estimator tailored for hybrid domains i.e. MSPNs [26]. Figure 2 shows the overview to learn the GCLN model.

To model the class, we make use of a Bernoulli distribution; it is either true or false with some probability p. For the clausal features, however, we have to be a bit more careful. They are Exchangeable Distribution Templates [28]. That is, each clausal feature is a function that takes all the ground instances of the clause as a set of random variables $\{X_1, \ldots, X_n\}$ as input (n is unknown a priori), and returns a joint probability distribution P with respect to which $\{X_1, \ldots, X_n\}$ is exchangeable[4]. All ground instance of a clause c share the same binomial distribution with value p associated with the clause. Since we do not know n apriori, we assume that the expectation $p \cdot n =: \lambda$ is constant, $0 < \lambda < \infty$. The Exchangeable Distribution Template then returns the following distribution:

$$\begin{aligned}
\lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} &= \lim_{n\to\infty} \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1-\frac{\lambda}{n}\right)^{n-k} \\
&= \lim_{n\to\infty} \frac{n(n-1)\ldots(n-k+1)}{k!} \left(\frac{\lambda}{n}\right)^k \left(1-\frac{\lambda}{n}\right)^{n-k} \\
&= \lim_{n\to\infty} \frac{\lambda^k}{k!} \left(1-\frac{\lambda}{n}\right)^{n-k} = \frac{\lambda^k e^{-\lambda}}{k!} \ .
\end{aligned}$$

In other words, the expected rate of a rule firing equals λ, and we get overall a Poisson distribution. Although here we assume that domain elements are exchangeable, our method is rather flexible and it works also when the logical rules contain constants.

Actually, MSPNs allow one to abstract from parametric forms and combine SPNs and piecewise polynomial distributions to learn probabilistic circuits that provide exact and tractable inference without making specific distributional assumptions. In fact, to learn the SPN structure and parameters, MSPNs make use of nonparametric decomposition and conditioning steps using the Renyi Maximum Correlation Coefficient. Taking into account that in our setting we deal mostly with count data i.e. how many times a rule fires– and binary targets, we learn an MSPN by adopting the splitting test tailored for Poisson distributions from PSPNs [25]. In other words, we employ a clustering strategy based on the Renyi Maximum Correlation Coefficient and a variable independence test designed for Poisson distributions. We assume that the independence test is performed between Poisson RVs while clustering can include the target modeled as Bernoulli and therefore, it is better for the latter to be able to deal with arbitrary distributions. Thus, our variant can be seen as a generalization of PSPNs and it could be easily adapted and applied on other types of data distributions.

Interpreting GCLNs. Sum-Product Networks have been shown able to learn a useful representation of the data [38]. Thanks to the clear probabilistic semantics of probabilistic circuits such as SPNs we can easily interpret the learned model in different ways. First, considering that learning the MSPN splits the data matrix by rows and by columns, one could easily check these slicing operations performed during structure learning. By analyzing the learned SPN structure the propositional probabilistic interpretations can be extended for relational data. For example, sum nodes create clusters of instances that fire rules similarly while product nodes work similarly to a logical and. Second, an interpretation can be obtained by checking the samples the learned SPN generates at the sub-trees rooted at different inner nodes.

[4] A set of random variables is finitely exchangeable with respect to a joint distribution P, if all permutations of the variables result in the same joint probabilities. Note that finite exchangeable does not require independence; the random variables can have strong dependencies.

Consider a simple sub-tree rooted to a product node S_i having two Poisson distributions (with different scope) as children. This product node models how likely and how many times these two rules fire together "locally" in a specific context shaped by the sum nodes present along the path from the root node to S_i. Additionally, sum nodes introduce latent variables [30] that can be seen as introducing literals in the relational domain. Meanwhile, product nodes are defining new predicates by conjunction of other predicates i.e. given the conditioning done by sum nodes and splitting performed by product nodes one can interpret these operations as manipulating and creating new predicates. A graphical representation of this interpretation is depicted in figure 1.[5] For example, the density encoded by the product node $\bigotimes$ (in blue in the computation graph) is: $P_{X_1,X_2,Y}(X_1,X_2,Y) =$

$$P_{X_1}(X_1 = 1) \cdot P_{X_2}(X_2 = x_2) \cdot P_Y(Y = 0) \cdot \mathbf{1}(\neg Y \wedge X_1)$$

And for its two siblings, from left to right:

$$\begin{aligned} P_{X_1,X_2,Y}(X_1,X_2,Y) =& P_{X_1}(X_1 = 0) \cdot P_{X_2}(X_2 = x_2) \cdot \\ & P_Y(Y = 0) \cdot \mathbf{1}(\neg Y \wedge \neg X_1) \\ P_{X_1,X_2,Y}(X_1,X_2,Y) =& P_{X_1}(X_1 > 1) \cdot P_{X_2}(X_2 = x_2) \cdot \\ & P_Y(Y = 0) \cdot \mathbf{1}(\neg Y \wedge X_1 > 1) \end{aligned}$$

While the right branch of the root node $\bigoplus$ encodes:

$$\begin{aligned} P_{X_1,X_2,Y}(X_1,X_2,Y) =& P_{X_1}(X_1 = x_1) \cdot P_{X_2}(X_2 = x_2) \cdot \\ & P_Y(Y = 1) \cdot \mathbf{1}(Y = \textit{True}) \end{aligned}$$

The weights of the sum nodes are proportional to the training instances that fall in the contexts they define e.g. regarding the root node, the weights are proportional to the amount of negative instances for the left branch and to the amount of positive instances for the right branch. They indicate how the two main populations of the distribution i.e. the negative and the positive samples, are distributed and encoded down the tree. On a perfectly balanced data set with a binary target the weight of each root sum node branch would be 0.5 (see figure 3).

As mentioned before, an interpretation can be obtained by generating samples from sub-trees rooted at the nodes that one would like to inspect. Therefore, considering the ability to generate samples in linear time, together with the clear semantics, GCLN is able to provide clear interpretations of the learned models w.r.t. the well-known counterparts like Relational Decision Trees or boosting methods where one could end up with very large models that are hard to interpret and thus are limited to compute predictions.

Missing Data Prediction. Dealing with missing data is a crucial and active area of research for both Machine Learning and Data Science with several open questions [2]. When dealing with missing values, task-specific methods such as MICE [4] or KNN [1] are generally used. Despite their popularity, these methods are not easily

[5] In the figure and in the following text $\mathbf{1}()$ represents an indicator function.

scalable, can be time-consuming and difficult to tune. Furthermore, often they do not provide confidence of their predictions, they could ignore the interactions with target variables and, mostly, they are able to perform only a single task. To overcome these shortcomings, probabilistic approaches that make use of tractable density estimators for handling missing data have been recently proposed [9, 18] and can handle missing data "for free" without being trained specifically for that task but are restricted to propositional data. With GCLN one can gain these benefits in the relational domain. Our model can classify a sample having partial observation regarding the rule that it fires and for a given sample predict the rule counts that are not available i.e. *missing data imputation*. In general, being able to accurately classify samples with missing predicates or also predicting the counts is not a trivial task. Recently, this is getting a lot of attention in practical applications e.g. in case of *privileged features* [37], fairness or privacy issues one can access a set of predicates only at training time and access them at test time could be too expensive or even not allowed. For example, one can classify with a certain confidence if two drugs interact given the rule:

$$\boxed{\text{TargetAntagonist(B, C)} \wedge \text{EnzymeSubstrate(A, C)} \implies \text{Interacts(A, B)}}$$

even if EnzymeSubstrate(A, C) is not available or one can predict a disease if

$$\boxed{\text{PositiveBloodCheck(A)} \wedge \text{HasDiabetes(A)} \wedge \text{HighBloodPressure(A)} \wedge \text{Tomography(A, T)} \wedge \text{PositiveManualCheck(T)} \implies \text{HasDisease(A)}}$$

even if Tomography(A, T) and PositiveManualCheck(T) are too expensive to obtain.

Out-of-domain detection. Another relevant task in machine learning is out-of-domain detection. This task is useful in many real-world applications such as computer vision [6] and NLP [35, 15]. The goal is to distinguish between in-domain and out-of-domain data and this could be useful e.g. for being robust to adversarial attacks, another relevant application [23]. Within a probabilistic framework, one can consider the likelihood values to discriminate between in-domain and OOD data [32, 40]. With GCLNs, this question can be answered by considering the likelihoods of data. Similar to missing data prediction, this comes "for free" i.e. without training the model explicitly for the task. Thus, GCLNs are powerful and flexible models that can perform several relevant tasks in the relational domain by answering a wide range of probabilistic queries.

4 Experiments

We aim to show our connection from Relational Decision Trees to Tractable Statistical Relational Inference and show that this leads to generative models that can act also as more accurate classifiers and are easier to interpret. We aim to answer the following questions: **(Q1)** Are GCLNs accurate discriminators? **(Q2)** Can GCLNs deal with missing data i.e. can they compute accurate predictions with missing data? Are GCLNs accurate probabilistic predictors for missing data imputation? **(Q3)** Are GCLNs able to perform out-of-domain detection? **(Q4)** Can GCLNs provide easy interpretations by means of their clear probabilistic semantics? **(Q5)** Can GCLNs take the best out of both propositional and relational worlds i.e. can GCLNs perform better than propositional and relational models being tractable?

All results are cross-validated with 5 folds and averaged over 5 different seeds to mitigate randomness. Since we are dealing with rule counts, there can be spurious rule firings that we can capture as well which leads to outliers, so we identify and move them to the training folds so the model is more robust. In fact, keeping them in the test set results in an "optimistic" bias in performance evaluation [5].

Datasets. We use 4 balanced relational data sets. **Drug-Drug Interaction (DDI)** [12]: consists of 78 drugs obtained from DrugBank[6]. The data set has 15 relations and the target is Interactions between drug entities. **Protein-Protein Interaction (PPI)** [20]: has 7 relations and is obtained from Alchemy. The target is the interaction relation between two protein entities. **NELL Sports** [24]: consists of information about players and teams and obtained from Never Ending Language Learner. It has 6 relations and the task is to predict whether a team plays a particular sport. **CiteSeer** [20]: consists of publication citations for Alchemy. It has 17 relations and the task is to predict the author of a citation.

(Q1) Probabilistic Classification. After splitting the data into training and test set, we use the default hyper-parameters of MSPNs[7] except $\mu = 100$. We compute the predictions i.e. ground target atoms as results of MPE queries, observing the counts of test samples. We compared GCLN (denoted as GCLN-P for clarity) and its binary variant GCLN-B that considers only if a rule fires or not, with several well-known high-performance propositional (Logistic Regression (LR), Gradient Boosting (GB), Neural Networks with 3 hidden layers (NN), Decision Trees (DT)), relational (TILDE) and statistical-relational (RDN-Boost and MLN-Boost) models. The results are shown in table 1. GCLNs outperform the other models in the majority of data sets and have comparable performance on CiteSeer. This is due to the particular shape of the data set which consists of 15K instances and 9 features and is more likely to have spurious instances harder to discriminate. It is important to remark that, compared to the statistical-relational baselines, GCLNs provide general tractable—and exact in most of the common cases—inference. While compared to non-statistical models, GCLNs can provide also meaningful probabilities and interpretations, and they can be employed for several tasks as shown in the rest of the section. Thus, we can answer **(Q1)** affirmatively, GCLNs are accurate discriminators.

(Q2) Inference with Partial Clausal Counts. We want to test whether GCLNs are robust to missing data by: 1) predict the class of test instances with a variable amount of missing features, 2) missing data imputation, and 3) predict the class of test instances with partially observed counts i.e. during testing the clause might not fire always and thus we do not have the true counts.

For 1), we compute the class prediction accuracy with GCLNs removing a varying percentage of features at random at test time. Figure 4 shows that GCLNs are accurate even with a considerable amount of missing features and it degrades gracefully on CiteSeer. The improvement on PPI is probably due to the presence of noisy/redundant

[6] www.drugbank.ca

[7] https://github.com/SPFlow/SPFlow

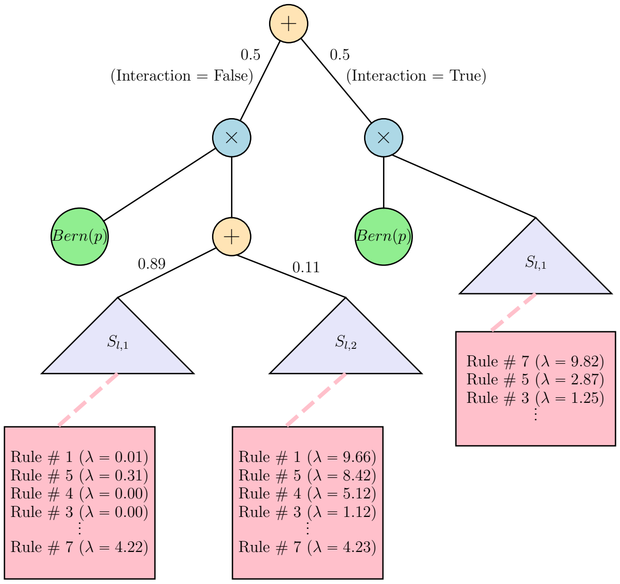

Fig. 3. Top levels of GCLN learned on DDI. GCLNs have clear semantics, the root sum node is clustering by label (balanced dataset) while the rest of the computation graph is encoding the different aspects of the distribution e.g. Rule #7 distributes differently basing on the label (i.e. the root node branch) while Rule #1 and #5 could fire many times for a small population of negative samples (consider the sum node weights along the path that brings to the sub-tree rooted at $S_{l,2}$ and the corresponding parameters λ).

features, considering no data preprocessing was done. Table 2 shows that, when the 30% of features are missing, GCLNs still perform strongly.

Regarding 2), we compared GCLNs with two standard data imputation methods: MICE and kNN[8]. Table 3 shows the accuracy as Mean Squared Error for the imputation of 30% of missing features chosen at random. The results show that GCLNs are competitive with task-specific state-of-the-art methods without being trained specifically for the task.

For 3), since GCLNs employ Poisson distributions, we can answer queries such as $P(X > k)$ i.e. how likely can a rule fire more times additionally to the observed count. To answer such queries, we propagate the computed probabilities

$$P(X > k) = 1 - \sum\nolimits_{i=0}^{k} P(X = i)$$

from the leaves to the root, given the Poisson leaf parameters. This way, GCLNs can also compute predictions and perform classification with partial knowledge. To see how accurate are GCLNs in these cases, we picked two of the most discriminative rules

[8] Both Scikit-learn implementation with default hyperparameters.

Table 1. Classification results of baselines compared to GCLN. The 1st 4 classifiers are propositional, the next 3 are purely relational and the last 2 are our models. In most of the cases GCLN outperforms the baselines. Furthermore, in the same cases, this happens also when employing Bernoulli distributions (GCLN-B). When comparing GCLN-P and GCLN-B one can clearly see that modelling counts with Poisson distributions indeed improves performances (denoted ↑).

Data	Methods	Accuracy	AUC-ROC	AUC-PR	Data	Methods	Accuracy	AUC-ROC	AUC-PR
DDI	LR	**93.79**	**87.62**	83.64	*NELL*	LR	83.96	58.90	28.27
	GB	86.77	86.17	67.95		GB	87.69	75.28	48.07
	NN	87.54	85.03	69.21		NN	88.26	74.42	48.35
	DT	85.52	83.22	65.12		DT	87.27	71.77	45.71
	TILDE	72.52	73.43	70.79		TILDE	81.48	86.27	78.23
	RDN-B	75.54	82.87	83.13		RDN-B	81.26	88.47	83.41
	MLN-B	63.80	79.83	78.40		MLN-B	60.54	**89.44**	**85.30**
	GCLN-B	85.05	70.57	70.84		GCLN-B	80.74	38.72	23.15
	GCLN-P	92.22 ↑	87.53 ↑	**83.81** ↑		GCLN-P	**89.83** ↑	**89.44** ↑	56.39 ↑
PPI	LR	78.13	81.54	52.44	*CiteSeer*	LR	76.33	83.78	53.30
	GB	77.21	78.25	49.54		GB	96.05	**97.21**	87.24
	NN	76.94	75.75	47.49		NN	**96.50**	96.88	88.10
	DT	76.47	77.52	48.71		DT	95.33	96.79	85.38
	TILDE	62.20	62.87	58.27		TILDE	91.47	83.33	73.45
	RDN-B	67.15	72.84	74.02		RDN-B	94.72	97.11	**89.23**
	MLN-B	54.87	74.39	73.34		MLN-B	81.98	94.67	80.54
	GCLN-B	79.72	82.75	59.43		GCLN-B	77.18	71.34	41.20
	GCLN-P	**81.56** ↑	**91.16** ↑	**80.08** ↑		GCLN-P	86.42 ↑	71.57 ↑	42.57 ↑

Table 2. Class prediction performance with 30% missing clauses.

Data	Accuracy	AUC-ROC	AUC-PR
DDI	89.90 ± 0.035	79.23 ± 0.036	75.00 ± 0.040
PPI	84.77 ± 0.025	89.18 ± 0.024	79.83 ± 0.040
NELL	86.06 ± 0.054	79.96 ± 0.063	47.30 ± 0.092
CiteSeer	68.09 ± 0.103	50.69 ± 0.016	32.51 ± 0.013

(e.g. rule #1 and #7 for DDI, see figure 5) and reduced the values of the counts by 20%. Table 4 shows that GCLNs can be accurate discriminators even when the most discriminative rules have underestimated counts and we can answer **(Q2)** affirmatively. GCLNs are not only robust in case of noisy or redundant features but are also beneficial with partial knowledge.

(Q3) Out-of-domain detection. We want to test whether GCLNs are robust to and can detect out-of-domain data. To this aim, we take the regular (in-domain) test data instances and we flip the binary target variable value of those to generate out-of-domain test data sets. Then, we compute the average log-likelihoods of these sets. For in-domain data the higher the log-likelihood the better, while for out-of-domain data the lower the better. We run a comparison with Discrete Flows, a state-of-the-art neural density estimator [36]. As one can clearly see in table 5, GCLNs assign, on average, remarkably lower log-likelihood to out-of-domain data compared to the one for in-domain test data, and lower than the one which Discrete Flows assign to out-of-domain data. This strong

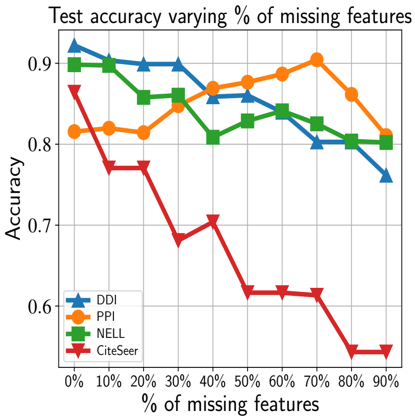

Fig. 4. Average test accuracy varying the amount of missing features. GCLNs are accurate also when a considerable amount of features is missing.

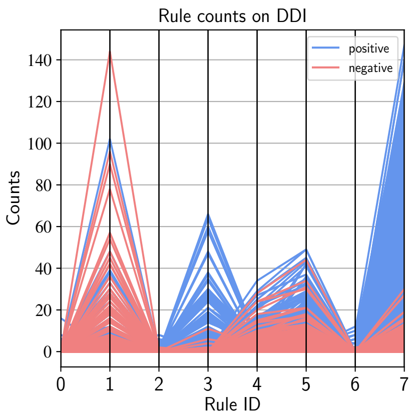

Fig. 5. Rule counts on DDI for each label i.e. no interaction and interaction between drugs.

Table 3. Missing data imputation MSE with 30% missing clauses.

Data	kNN	MICE	GCLN
DDI	37.41 ± 3.751	37.25 ± 3.451	40.72 ± 3.794
PPI	2.40 ± 3.451	1.44 ± 1.748	3.22 ± 4.320
NELL	18.53 ± 28.900	10.80 ± 16.644	42.58 ± 40.612
CiteSeer	0.76 ± 0.090	0.65 ± 0.011	1.02 ± 0.039

signal shows that GCLNs are effective in detecting out-of-domain data. Being able to perform out-of-domain detection also makes GCLNs more robust to potential adversarial attacks. Thus, we can answer **(Q3)** positively.

(Q4) Model Interpretation. We inspect the model interpretability by looking at the structural characteristics of the learned models and their parameters. We discovered that the root sum node of the SPNs does a high-level discrimination between instances, and respectively sub-populations of the distribution and thus discriminates between negative and positive examples. This is also consistent with the interpretation of deep models where the upper layers encode more abstract concepts. Similar observations can be done for the following layers where common patterns on how the counts distribute "locally" for both negative and positive samples are encoded down the SPN tree. For example, on DDI, a considerable group of positive instances has all the features equal to 0 (no rules fired) while several negative instances have all the features equal to 0 except the last one. This is a common pattern for negative samples i.e. when only the last rule fires one or more times. Figure 3 shows the top levels of the GCLN that performs the best on the training set. Some example rules learned for DDI are:

Table 4. Prediction with partially observed counts.

Data	Accuracy	AUC-ROC	AUC-PR
DDI	85.93 ± 0.042	94.11 ± 0.020	90.83 ± 0.015
PPI	42.65 ± 0.025	87.70 ± 0.063	83.09 ± 0.074
NELL	85.76 ± 0.058	63.02 ± 24.26	30.32 ± 0.092
CiteSeer	72.96 ± 0.014	50.00 ± 0.000	27.04 ± 0.089

Table 5. Average test log-likelihoods for in-domain data (the higher the better) and OOD (the lower the better). To create test OOD data we flip the target value of the test instances. One can see that GCLNs are also good OOD detectors, in fact, the log-likelihood of OOD data is much lower compared to the log-likelihood of in-domain instances, and also lower than the one which Discrete Flows assign to OOD data.

	GCLN		Discrete Flows	
Data	In-domain	OOD	In-domain	OOD
DDI	-9.02 ± 0.431	-21.55 ± 6.124	-6.92 ± 0.194	-6.95 ± 0.194
PPI	-7.91 ± 2.056	-8.61 ± 0.956	-6.98 ± 0.409	-6.93 ± 0.415
NELL	-21.37 ± 4.969	-23.02 ± 3.131	-11.81 ± 0.775	-12.24 ± 0.750
CiteSeer	-5.17 ± 0.122	-124.42 ± 23.824	-4.26 ± 0.200	-4.36 ± 0.200

Rule #1: Transporter(C, A) $\land$ Transporter(C, B) $\implies$ Interacts(A,B)
Rule #3: Transporter(C, A) $\land$ EnzymeInhibitor(A, D) $\land$ EnzymeSubstrate(B, D) $\land$ TransporterSubstrate(A, C) $\land$ TransporterInducer(A, E) $\implies$ Interacts(A,B)
Rule #7: TargetAntagonist(B, C) $\land$ EnzymeSubstrate(A, C) $\implies$ Interacts(A,B)

One can see that the top levels are capturing the high-level concepts in data where e.g. Rule #7 is likely to fire more many times for positive samples. The explanation is that the rule fires when the same protein acts as an antagonist for one drug and as a substrate for another i.e. they are more likely to interact. This is also confirmed by looking globally at how the rule counts distribute for different labels in figure 5. For example, one can see that the rules #7, #5 and #3 are very discriminative. Such interpretations can be provided also when computing predictions (see Q1) and used as explanations for XAI. Therefore, we can answer **(Q4)** positively.

(Q5) Ablation Study. We compare GCLNs with TILDE and GCLN-Bernoulli to check whether considering the rule counts is beneficial instead of the binary counterpart where we consider only if a rule fires or not and we model this by means of Bernoulli distributions. In this context, for clarity, GCLN-Poisson is an alias of GCLN. Looking at table 1 we can see that GCLN-Poisson outperforms TILDE in all cases (up to 31% relative increase on PPI) except for CiteSeer. GCLN-Bernoulli has competitive accuracy performance when compared with the other methods and it outperforms them on PPI. However, GCLN-Poisson outperforms GCLN-Bernoulli in all the cases. This means that *GCLNs can take the best out of the two worlds* by improving upon the relational model and when considering only when a rule fires or not. Moreover, in most of the cases, it outperforms all the propositional models and answers **(Q5)** affirmatively.

5 Conclusion

We introduce GCLNs and have drawn a connection from relational models to tractable probabilistic models that allow to compute general tractable inference, provide meaningful probabilities and have clear semantics that foster interpretability. Besides, GCLNs can act as both deep tractable generative model and accurate discriminator that is robust to missing and partially observed features, and can be used conveniently also for out-of-domain detection. Future work includes extending our model to the open-world domain and make use of multiple distributions. Encoding more relational models as probabilistic models thereby providing tractability is an important future direction.

Acknowledgments

This work was supported by the ICT-48 Network of AI Research Excellence Center “TAILOR” (EU Horizon 2020, GA No 952215), the Federal Ministry of Education and Research (BMBF; Competence Center for AI and Labour; “kompAKI”, FKZ 02L19C150), the German Science Foundation (DFG, German Research Foundation; GRK 1994/1 “AIPHES”), the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK; projects “The Third Wave of AI” and “The Adaptive Mind”), the Hessian research priority programme LOEWE within the project “WhiteBox”, and the Collaboration Lab “AI in Construction” (AICO).

References

1. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician (1992)
2. Austin, P.C., White, I.R., Lee, D.S., van Buuren, S.: Missing data in clinical research: A tutorial on multiple imputation. Canadian Journal of Cardiology (2020)
3. Blockeel, H., de Raedt, L.: Top-down induction of first-order logical decision trees. AI (1998)
4. van Buuren, S., Groothuis-Oudshoorn, K.: mice: Multivariate imputation by chained equations in r. J. Stat. Softw (2011)
5. Cawley, G.C., Talbot, N.L.: On over-fitting in model selection and subsequent selection bias in performance evaluation. JMLR (2010)
6. Chattopadhyay, P., Balaji, Y., Hoffman, J.: Learning to balance specificity and invariance for in and out of domain generalization. In: ECCV (2020)
7. Cheng, W., Kok, S., Pham, H.V., Chieu, H.L., Chai, K.M.A.: Language modeling with Sum-Product Networks. In: INTERSPEECH (2014)
8. Choi, Y., Vergari, A., Van den Broeck, G.: Probabilistic circuits: A unifying framework for tractable probabilistic models (2020)
9. Correia, A.H.C., Peharz, R., de Campos, C.P.: Joints in random forests. In: NeurIPS (2020)
10. Darwiche, A.: A differential approach to inference in bayesian networks. JACM (2003)
11. De Raedt, L., Kersting, K., Natarajan, S., Poole, D.: Statistical relational artificial intelligence: Logic, probability, and computation (2016)
12. Dhami, D.S., Kunapuli, G., Das, M., Page, D., Natarajan, S.: Drug-drug interaction discovery: Kernel learning from heterogeneous similarities. Smart Health (2018)
13. Dhami, D.S., Yen, S., Kunapuli, G., Natarajan, S.: Non-parametric learning of gaifman models. arXiv preprint arXiv:2001.00528 (2020)

14. Flach, P.A., Lachiche, N.: Naive bayesian classification of structured data. Machine Learning (2004)
15. Gangal, V., Arora, A., Einolghozati, A., Gupta, S.: Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. In: AAAI (2020)
16. Gens, R., Domingos, P.: Learning the Structure of Sum-Product Networks. In: ICML (2013)
17. Getoor, L., Taskar, B.: Statistical relational learning (2007)
18. Khosravi, P., Vergari, A., Choi, Y., Liang, Y., den Broeck, G.V.: Handling missing data in decision trees: A probabilistic approach. arXiv preprint arXiv:2006.16341 (2020)
19. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning markov logic networks via functional gradient boosting. In: ICDM (2011)
20. Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., Lowd, D., Wang, J., Domingos, P.: The alchemy system for statistical relational ai (2005)
21. Landwehr, N., Kersting, K., De Raedt, L.: nfoil: Integrating naıve bayes and foil. In: AAAI (2005)
22. Landwehr, N., Kersting, K., De Raedt, L.: Integrating naive bayes and foil. JMLR (2007)
23. Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: NeurIPS (2018)
24. Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al.: Never-ending learning. Comm. of the ACM (2018)
25. Molina, A., Natarajan, S., Kersting, K.: Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In: AAAI (2017)
26. Molina, A., Vergari, A., Mauro, N.D., Natarajan, S., Esposito, F., Kersting, K.: Mixed sum-product networks: A deep architecture for hybrid domains. In: AAAI (2018)
27. Natarajan, S., Khot, T., Kersting, K., Gutmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning (2012)
28. Nath, A., Domingos, P.: Learning relational sum-product networks. In: AAAI (2015)
29. Neville, J., Jensen, D.: Relational dependency networks. JMLR (2007)
30. Peharz, R., Gens, R., Pernkopf, F., Domingos, P.M.: On the latent variable interpretation in sum-product networks. TPAMI (2017)
31. Peharz, R., Kapeller, G., Mowlaee, P., Pernkopf, F.: Modeling speech with sum-product networks: Application to bandwidth extension. In: ICASSP (2014)
32. Peharz, R., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Shao, X., Kersting, K., Ghahramani, Z.: Random sum-product networks: A simple and effective approach to probabilistic deep learning. In: UAI (2019)
33. Poon, H., Domingos, P.M.: Sum-product networks: A new deep architecture. In: UAI (2011)
34. Richardson, M., Domingos, P.: Markov logic networks. Machine learning (2006)
35. Tan, M., Yu, Y., Wang, H., Wang, D., Potdar, S., Chang, S., Yu, M.: Out-of-domain detection for low-resource text classification tasks. In: EMNLP-IJCNLP (2019)
36. Tran, D., Vafa, K., Agrawal, K.K., Dinh, L., Poole, B.: Discrete flows: Invertible generative models of discrete data. In: NeurIPS 2019 (2019)
37. Vapnik, V., Vashist, A.: A new learning paradigm: Learning using privileged information. Neural Networks (Jul 2009)
38. Vergari, A., Di Mauro, N., Esposito, F.: Visualizing and understanding sum-product networks. Machine Learning (2019)
39. Yang, E., Ravikumar, P.K., Allen, G.I., Liu, Z.: On poisson graphical models. NIPS (2013)
40. Yu, Z., Ventola, F., Kersting, K.: Whittle Networks: A Deep Likelihood Model for Time Series. In: ICML (2021)
41. Yuan, Z., Wang, H., Wang, L., Lu, T., Palaiahnakote, S., Tan, C.L.: Modeling spatial layout for scene image understanding via a novel multiscale sum-product network. Expert Systems with Applications (2016)