Non-Parametric Learning of Embeddings for Relational Data using Gaifman Locality Theorem

Devendra Singh Dhami[1,2], Siwen Yan[2], Gautam Kunapuli[3], and Sriraam Natarajan[2]

[1] Technical University of Darmstadt, Darmstadt, Germany
devendra.dhami@cs.tu-darmstadt.de
[2] University of Texas at Dallas, Richardson, USA
{siwen.yan,sriraam.natarajan}@utdallas.edu
[3] Verisk Analytics, New Jersey, USA
gautam.kunapuli@verisk.org

Abstract. We consider the problem of full model learning from relational data. To this effect, we construct embeddings using symbolic trees learned in a non-parametric manner. The trees are treated as a decision-list of first order rules that are then partially grounded and counted over local neighborhoods of a Gaifman graph to obtain the feature representations. We propose the first method for learning these relational features using a Gaifman graph by using relational tree distances. Our empirical evaluation on real data sets demonstrates the superiority of our approach over handcrafted rules, classical rule-learning approaches, the state-of-the-art relational learning methods and embedding methods.

Keywords: Statistical relational learning · Gaifman locality theorem · Embeddings · Relational density estimation

1 Introduction

Learning embeddings of large knowledge bases has become a necessity due to the importance of reasoning about objects, their attributes and relations in large graphs. Statistical Relational AI (StaRAI) [25,9] has the ability to learn and reason with multi-relational data in the presence of uncertainty. A scalable approach, Discriminative Gaifman Model, via *Gaifman networks* was proposed recently [23] that exploits *Gaifman's locality theorem* [8]: every first-order sentence is equivalent to a boolean combination of sentences over local entity neighborhoods of the Gaifman graph. Relational Gaifman models seek to identify locally-connected relational neighborhoods within knowledge bases for effective representation, learning and inference. While effective, discriminative Gaifman Models used relational features that were hand crafted rather than learned, i.e., there was *no structure learning*. Consequently, their applicability and adaptability can become severely limiting.

Motivated by this limitation, we present the *first* set of approaches for relational embeddings that are guided by Gaifman locality theorem. Given that we

are in a *symbolic* setting, these approaches have the distinct advantage of being both **explainable** and **interpretable**. Specifically, we propose and investigate three approaches: (1) As suggested by Niepert [23], we employ Inductive Logic Programming (ILP) to learn discriminative first-order rules. These (conjunctive) rules can then be treated as Boolean relational features. (2) Inspired by the success of random walks in deep relational models [11], we employ relational random walks (RRWs) as relational features. To this effect, we developed our own random walk implementation built on an underlying ILP engine as against a relational database as with the original work. (3) Finally, we use first-order trees learned via relational one-class classification (relOCC [14]); specifically, each path from root to leaf of a first-order trees is considered a relational feature. That is, the structure is captured by the "*relational density estimate*", which is learned from data as a set of first-order trees. The motivation behind using a density estimation technique to learn the relational features is that *learning first order rules for positives and sampled negatives independently results in a better utilization of the search space thereby learning better discriminative features.*

Given these relational features, that can be grounded based on Gaifman's locality theorem, one could apply traditional discriminative learning algorithms such as Gradient Boosting and Logistic Regression. This allows for the embedding creation method to be decoupled from the underlying classifier.

We consider the challenging problem of structure learning to exploit the importance of local neighborhoods in knowledge graphs. Our key contributions are: (1) We present the **first method for learning relational embeddings** for reasoning over large graphs using Gaifman's locality theorem. (2) We adapt a recently developed relational learning method for constructing relational features (relOCC). (3) We adapt well-known first-order rule learners for learning local neighborhood representations (ILP, RRWs). (4) We combine these relational features with discriminative classifiers to learn discriminative Gaifman models. (5) We demonstrate that **combining the more novel first-order trees with a discriminative classifier is more effective in learning on large graphs** compared to a standard ILP learner. Specifically **our novelty lies in the fact that we are learning rules for the positive and negative instances** separately using density estimation that allows for better discrimination. An important side-effect is that these rules are **explainable** in contrast to many traditional embedding methods. (6) Our experiments reveal an important characteristic of our approach: **high recall without sacrificing precision** in both medical and imbalanced data sets.

2 Background and Related Work

Discriminative Gaifman Models: The Gaifman graph $\mathcal{G}$ of a knowledge base $\mathcal{B}$ is an **undirected graph**, where the nodes are the entities $e \in \mathcal{D}$. $\mathcal{G}$ contains edges joining two nodes only if the entities a and b corresponding to those nodes are present in a relation together $\mathtt{R}(\ldots, a, \ldots, b, \ldots) \in \mathcal{B}$. $\mathcal{G}$ can be used to easily identify co-occurrences (or lack thereof) among every pair of entities in $\mathcal{B}$. Fig. 1 shows a knowledge-base fragment and the corresponding Gaifman graph

```
TransSubstr(Pravastatin, BileSaltExportPump)
TransInhib(Simvastatin, MultidrugResistProtein1)
EnzInhib(Pravastatin, CytochromeP4502C9)
EnzSubstr(Acetaminophen, CytochromeP4502C9)
EnzInhib(Simvastatin, CytochromeP4502C9)
```

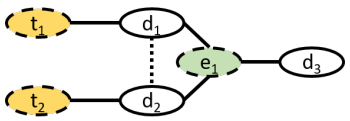


Fig. 1: An example Gaifman graph for a drug-drug interaction (DDI) knowledge base. Here, $\mathtt{d}_1 = \mathtt{Pravastatin}$, $\mathtt{d}_2 = \mathtt{Simvastatin}$, $\mathtt{d}_3 = \mathtt{Acetaminophen}$, $\mathtt{t}_1 = \mathtt{BileSaltExportPump}$, $\mathtt{t}_2 = \mathtt{MultidrugResistProtein1}$ and $\mathtt{e}_1 = \mathtt{CytochromeP4502C9}$. The Gaifman graph connects entities that appear in a relationship tuple; the dotted line between $\mathtt{d}_1$ and $\mathtt{d}_2$ is the link we want to predict.

for a drug-drug interaction (DDI) domain. Given entities (drugs, enzymes, transporters) and their relationships, the underlying learning task is to predict if two drugs interact. The dotted line is the target, and the task is **link prediction**.

The distance $d(a, b)$ between two nodes $(a, b) \in \mathcal{G}$ is the minimum number of hops required to reach b from a. The r-neighborhood of a node $a \in \mathcal{G}$ is the set of all nodes that are at most a distance r from a in the Gaifman graph: $N_r^{\mathcal{G}}(a) = \{\bar{a} \in \mathcal{G} \mid d(a, \bar{a}) \leq r\}$. When a first-order rule $\varphi(x)$ is relativized by the neighborhood of the free variable x, the resulting first-order rule $\psi^{N_r(x)}(x)$ is called r-local. A Gaifman neighborhood can be thought of as representing second-order proximity between nodes. The interpretation is that nodes with shared neighbors are more likely to be similar and more likely to have a link between them. Discriminative Gaifman Models (DGMs, [23]) are relational models that exploit structural features of a local neighborhood. These structural features are aggregated from locally-sampled neighborhoods, and the aggregation is based on the *Gaifman locality theorem* [8] stated as: *every first-order sentence is logically equivalent to a Boolean combination of basic r-local sentences.*

For example, if querying about the drug $\mathtt{d}_1$ in Fig. 1, a search within the 1-neighborhood of $\mathtt{e}_1$ (say), that is $\{\mathtt{t}_1, \mathtt{e}_1\}$ is more relevant than searching through the complete graph, which can be significantly computationally inefficient.

Representation Learning: Learning embeddings is well-studied and can be categorized based on the underlying approaches: matrix factorization, deep learning, edge reconstruction, graph kernels and generative models [3]. In general, Gaifman models tend to scale better than many such approaches to higher-arity relations and target-query complexity owing to their local view and incorporation of count-based features as opposed to the global view of (say) neural network or factorization methods which are forced to look at the entire graph to construct effective embeddings. Recent work has also included the study of holographic embeddings [21], which measure similarity through circular correlation and hyperbolic embeddings [22], which measure similarity and construct embeddings in a hyperbolic space. While highly effective, a key drawback of these approaches is their inability to incorporate new data, often requiring training of a new model.

Relational and Structure Learning: One of the most important tasks in relational learning is that of *link prediction* which determines whether a relation (link) exists between entities based on the given relational database[27]. Structure learning has been a well-studied problem in graphical models and can be

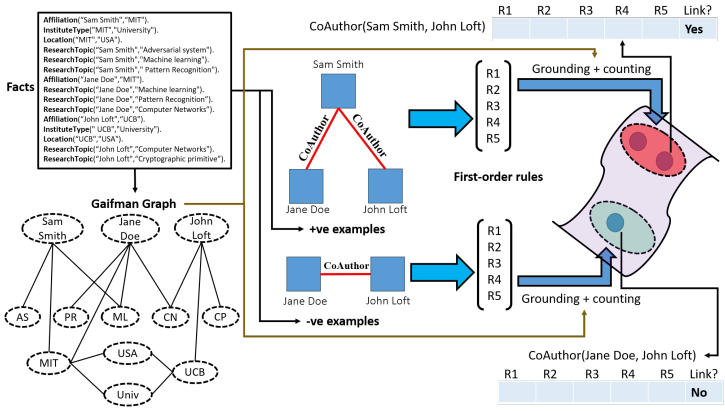


Fig. 2: An overview of learning embeddings using Gaifman locality theorem.

defined as the problem of identifying a graph structure to principally summarize dependencies in a data set and can be interpreted as learning probabilistic (relational) rules from data [4,13].

3 Gaifman-guided Learning of Relational Embeddings

Given: Knowledge base $\mathcal{B}$, facts F_s, and its corresponding Gaifman graph $\mathcal{G}$;
Output: A discriminative model $\mathcal{M}$ that is trained for a particular link prediction task $\mathcal{T}$;
To-Do: Construct a set of relational embeddings (features) Φ, and train a discriminative learner to predict $\mathcal{T}$.

We present the **first set of model learning algorithms that employs Gaifman local graphs.** Our approach, *Learning Gaifman-based relational Embeddings* (LGE), (1) constructs (explainable) rules Φ that form the base set of relational features. This is akin to structure learning in Statistical Relational AI [25] models; (2) instantiates rules (grounding) and performs counting based on task $\mathcal{T}$ to construct propositional features (embeddings) $\mathcal{F}$; and finally, (3) learns a discriminative classifier with $\mathcal{F}$. Fig. 2 shows an overview of our method.

We represent predicates and constants by capitialized letters and variables by small letters. For example, in our DDI domain, $\texttt{Interacts}(\texttt{d}_1, \text{“}\texttt{Metformin}\text{”})$ represents a partially grounded example where `Interacts` is the predicate, $\texttt{d}_1$ is a variable and “`Metformin`” is a constant.

Given a knowledge base $\mathcal{B}$, the Gaifman graph $\mathcal{G}$ is obtained by instantiating the entities that are connected by an edge type (relation) together in the form $\texttt{R}(\texttt{e}_1, \texttt{e}_2)$, that is, $relation(type_1, type_2)$. Gaifman neighborhood generation relies on three parameters: (1) r, the depth of neighborhood when counting, (2) k, the number of neighbors to sample, and (3) w, the number of neighborhoods to be generated. The relation (link) to be predicted, defined by the target predicate, forms the set of positive examples. We make the *closed-world assumption*, that is, unobserved edges in the graphs are considered to be negative examples.

Hence, the negative examples are constructed by taking the Cartesian product of the entire entity set with itself and removing the examples which are present in the positive example set. Following standard ILP terminology, each relational example also has *facts* associated with it, which are the ground predicates in $\mathcal{B}$ that describe relational example, its attributes and relationships. All such facts are denoted F_s.

ILP for rule extraction: Our first solution is inspired by the use of an Inductive Logic Programming (ILP) style learning method. This method learns a set of discriminative Horn clauses. Specifically, we use an ILP system called WILL [29] to learn the first-order rules as features[4]. This ILP system first selects an example from the set of all examples and then finds a clause (rule) that *best covers* the examples. Ideal coverage means *all* positive examples and *no* negative examples which can easily overfit. To avoid overfitting, we obtain the *best covering* which is the most general clause that maximizes the difference between the number of positive and negative examples covered. Each *best covering* clause becomes a first-order rule in our model. The examples covered by the clause are then removed and the process is repeated till a stopping criterion (e.g., maximum of n rules) is satisfied. Some possible stopping conditions are: (1) a certain number of examples are covered by the currently extracted set of rules, or (2) we have extracted a maximum number of rules/clauses. Note that when a stopping criterion requires n rules to be extracted, it is sometimes possible to extract $m < n$ rules that cover the examples adequately. Contrarily, if the first condition is specified and the expected coverage is very high, it may require a very large number of rules to be extracted before termination. Thus, in practice, a combination of both conditions is employed. The key advantage of this method is that these rules are both *interpretable* and *explainable*.

Features via Relational Random Walks: It is easy to view a relation $\texttt{R}(\texttt{e}_1, \texttt{e}_2)$ as an edge between two entity type nodes $\texttt{e}_1 \xrightarrow{\texttt{R}} \texttt{e}_2$ in a graph. A relational random walk (RW) through a graph is a chain of such edges corresponding to a conjunction of predicates. For a random walk to be semantically sound, we should ensure that the input type (domain) of the $i+1$-th predicate is same as output type (range) of the i-th predicate. Example RW for drug-discovery is:

$$\texttt{Interacts}(\texttt{d0}, \texttt{d3}) \Leftarrow \texttt{TargetInhib}(\texttt{d0}, \texttt{t0}) \wedge \texttt{\_TargetInhib}(\texttt{t0}, \texttt{d1}) \\ \wedge \texttt{TransporterSubstr}(\texttt{d1}, \texttt{t2}) \wedge \texttt{\_TransporterInhib}(\texttt{t2}, \texttt{d3}).$$

This is a semantically sound random walk as it is possible to chain the second argument of each predicate to the first argument of the succeeding predicate. This random walk also contains *inverse predicates* (prefixed by an underscore, such as `_Transporter`). Inverse predicates are distinct from their corresponding predicates as their arguments are *reversed*. Thus, this relational random walk chains the first variable `d0` in the target predicate $\texttt{Interacts}(\texttt{d0}, \texttt{d3})$ with the second variable `d3`. The RW chain represents a relational feature and constitutes

[4] Any ILP learner such as Aleph [26] or PROGOL [19] can be used.

a random local structure:

$$\texttt{d0} \xrightarrow{\texttt{TargetInhibitor}} \texttt{t0} \xrightarrow{\texttt{\_TargetInhibitor}} \cdots$$
$$\texttt{d1} \xrightarrow{\texttt{TransporterSubstrate}} \texttt{t2} \xrightarrow{\texttt{\_TransporterInhib}} \texttt{d3}.$$

Thus, to construct a relational random walk, only the schema describing the knowledge base is required. We adapt path-constrained random walks (PCRW, [16]) to construct relational random walks. The algorithm starts at the first entity in the target relation, and makes a walk over the (parameterized) graph to end at the second entity present in the target relation. One limitation of PCRW is that the random walks are only performed over binary relations. Consequently, we employ *our own implementation that uses a more general predicate representation that can learn with arbitrary n-ary relations*. This implementation is built on top of the WILL system [29] that we used in our first step. We constrain the length of the random walks to avoid/prevent overfitting.

Density estimation via relocc for rule learning: A common issue in many tasks, is that only the few "positive" instances of a relation are annotated due to severe class imbalance (exponentially many negatives). This is because the number of instances where the relation does not hold is very large, and annotation can be prohibitively expensive. Learning with highly imbalanced data sets requires reasoning over just the positive instances, commonly referred to as *one-class classification* (OCC). Intuitively, if we can construct a relational one-class classifier describing the positive examples, then rules characterizing this classifier are essentially features that describe positive examples. One-class classification typically requires a *distance measure* to characterize the density of the positive class. While, for standard vector and matrix data, many different distance measures exist, the issue is far more challenging for relational data, and depends on the underlying representation of the classifier.

Suppose we use an off-the-shelf learner to learn first-order trees [1] to describe each class in the data. Such first-order trees form a decision-list of logical rules (similar to ILP but with negations). These trees can then be used to compute the *relational distance* between a pair of examples x_1 and x_2, which are instances of the the target predicate $\texttt{R}(\texttt{e}_1, \texttt{e}_2)$ as in [14]. For a learned tree i:

$$\mathcal{RD}_i(x_1, x_2) = \begin{cases} 0, & \mathsf{LCA}(x_1,\, x_2) \text{ is leaf;} \\ e^{-\lambda \cdot \mathsf{depth}(\mathsf{LCA}(x_1, x_2))}, & \text{otherwise,} \end{cases} \tag{1}$$

where LCA is the *least common ancestor* of examples x_1, x_2. Typically more than one tree is learned, and the one-class classifier is a weighted combination of these trees. The trees in one-class classifier are learned iteratively by updating the distance measure. Then, the overall distance function is simply the weighted combination of the individual tree-level distances: $\mathcal{CD}(x_1,\, x_2) = \sum_i \beta_i \mathcal{RD}_i(x_1, x_2)$ where β_i is the weight of the i^{th} tree and $\sum_i \beta_i = 1, \beta_i \geq 0$. The non-parametric function $\mathcal{CD}(\cdot, \cdot)$ is a relational distance measure learned on the data.

The distance function can then be used to compute the density estimate for a new relational example z as a weighted combination of the distance of z from

all training examples x_j, $E(z \notin \texttt{class}) = \sum_j \alpha_j \mathcal{CD}(x_j, z)$, where α_j is the weight of the labeled example x_j and $\sum \alpha_j = 1, \alpha_j \geq 0$. Note that expectation above is for $z \notin \texttt{class}$, since the likelihood of class membership of z is inversely proportional to its distance from the training examples describing that `class`.

We learn a tree-based distance iteratively [14] to introduce new relational features that perform one-class classification. The left-most path in each relational tree is a conjunction of predicates with no negation, that is, a clause, which can be used as a relational feature. This makes the model more tractable but any path can be used if negation can be handled. Most importantly, *this allows for an interpretation similar to a Horn clause thus making the rules both interpretable and explainable*. We present the algorithm for learning rules using relOCC and generating embeddings externally[5]. Some example rules (first 2 rules for +ve examples and rest for -ve examples) learned for DDI are:

1. EnzymeInducer(A, C), EnzymeSubstrate(B, C), EnzymeInducer(B, D), EnzymeInducer(A, D) $\implies$ Interacts(A,B)
2. EnzymeSubstrate(A, C), EnzymeSubstrate(B, C), TransporterInducer(A, D) $\implies$ Interacts(A,B)
3. EnzymeInhibitor(A, C), Enzyme(C, B), TransporterInhibitor(A, D) $\implies$ Interacts(A,B)
4. TargetAgonist(B, C), TransporterSubstrate(A, D), EnzymeSubstrate(B, E), EnzymeSubstrate(A, E) $\implies$ Interacts(A,B)

Ground features from relational rules: Once extracted, relational rules are grounded and the number of satisfied groundings are aggregated . While several *feature aggregations* exist, we use *counts* as they have been previously successful in many statistical relational models [15,11]. For every predicate $\varphi \in \Phi$, the first and last entity are instantiated corresponding to the tuples satisfying the query (since it is a link prediction task, a query variable is of the type $q(e_1, e_2)$) to give a partially grounded predicate). For example, in Fig. 1, let the positive example be $Interacts(Pravastatin, Simvastatin)$. For the predicate $EnzymeInhib(d_0, t_0) \wedge \_EnzymeInhib(t_0, d_1)$, and the substitution $\{d_0/Pravastatin, d_1/Simvastatin\}$, we obtain the *partially-grounded* predicate $EnzymeInhib(Pravastatin, t_0) \wedge \_EnzymeInhib(t_0, Simvastatin)$.

Next, all the predicates that completely satisfy this partially grounded feature are obtained. The features for each query variable are then obtained as counts of the number of satisfied groundings that are also present in the neighborhood of the query entities in the Gaifman graph $\mathcal{G}$. For example, in Fig. 1, if $EnzymeInhib(Pravastatin, CP4502C9) \wedge \_EnzymeInhib(CP4502C9,$ $Simvastatin)$ satisfies the given predicate, since $CP4502C9$ (CP = Cytochrome) is present in the Gaifman neighborhood of $Pravastatin$ (as well as $Simvastatin$), the count of the predicate is increased by 1. Thus, for every query variable q we

[5] `https://bit.ly/3jp9NA2`

obtain a propositional feature $f = [f_1, ..., f_{|\Phi|}]$ of length $|\Phi|$:

$$f_i = \begin{cases} |\psi^{N_r(q)}(q)|, & \text{if } q(e_1, e_2) \text{ partially grounds } \Phi_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Recall that ψ refers to the relativized first-order formula, and consequently $\psi^{N_r(q)}(q)$ is the r-local formula for a neighborhood N of depth r. We count the number of entities in the satisfied grounded features that are also satisfied in the neighborhood structure of the Gaifman graph, thus constructing a propositionalized data set of $|pos| \times w$ +ve examples and $|neg| \times w$ -ve examples.

Learning a Discriminative Model: After learning the propositional features, any standard classifier can be used for link prediction. One could potentially use any classifier[6]. For our experiments, we employ gradient-boosting and logistic regression. The classification algorithm itself is not a key contribution of our work and as we demonstrate empirically next, any standard classifier will often suffice for learning an effective model.

Effectiveness of relOCC in learning structure: The motivation behind using a relational density estimation technique (relOCC) for rule learning is that *learning independently from different densities separately can potentially result in better discrimination in the learned feature space.*

The use of a relational one-class classifier [14] that classifies positive examples based on a distance metric ensures that positive examples are projected close to each other in the new feature space. Since we hypothesize that learning from the example densities separately and independently results in a better discriminative behavior, we go a step further and also learn a relational one class classifier for the negative relational examples. This results in negative examples being projected close to each other in the new feature space and further from positive examples.

An added advantage of using such a rule learning procedure to obtain the propositional examples is that the learned examples directly represent the query variable. *The link prediction problem is thus reduced to a prediction problem in the new feature space.* This makes the approach independent of the learning algorithm allowing for flexibility of the use of any learning algorithm.

Table 1: Evaluation domains and their properties.

Data set	**#Entities**	**#Relations**	**#Pos**	**#Neg**	**#RW rules**	**#ILP rules**	**#relOCC rules**
DDI	355	15	2832	3188	68	36	25
PPI	797	7	1915	1915	42	5	15
NELL Sports	4147	6	300	600	36	15	13
Financial NLP	650	7	186	1029	222	6	25
ICML CoAuthor	558	5	155	6498	7	15	7

4 Experiments

We have made our code available at `https://bit.ly/2YYHZZ4`. We aim to answer the following questions: **Q1:** How do different structure learning strategies compare across diverse domains from different applications? **Q2:** How do different structure learning strategies impact performance in the presence of high class

[6] `https://bit.ly/3jp9NA2`

imbalance? **Q3:** What are effects of Gaifman locality parameters r, w and k? **Q4:** How does our method compare with state-of the art probabilistic ILP systems? **Q5:** How does our method compare with state-of the art relational embedding methods? **Q6:** How does our method compare with different SOTA rule learning methods including Niepert's original approach [23] of using hand-crafted rules?

Data sets: We consider 5 ***real-world relational data sets*** (Table 1). **Drug-Drug Interactions (DDI)** [6] consists of 78 drugs obtained from DrugBank and the target is *interactions* between drug entities. **Protein-Protein Interactions (PPI)** [15] is obtained from Alchemy and the target is *interaction* relation between two protein entities. **NELL Sports** was generated by the Never Ending Language Learner [18] consisting of information about players and teams. The task is to predict whether a team plays a particular sport i.e *teamplayssport*. **Financial NLP** is obtained by extracting information from *SEC Form S-1* documents and the target is to predict whether a word occurs in a given sentence i.e. the relation *sentenceContainsTarget*. **ICML Co-Author** is obtained by mining publication data from ICML 2018 and the target is the *CoAuthor* relation between persons.

Baselines (Statistical Relational Learning methods): We compare the performance of our method with 3 state-of-the-art SRL methods. **RDN-Boost** [20] and **MLN-Boost** [13]: are SRL models that propose functional gradient boosting of relational dependency networks (RDNs) and Markov logic networks. **Tuffy** [24]: is an MLN learning and inference engine using RDBMS to obtain a solution to the scalability problems of the underlying networks.

Baselines (Relational Embedding methods): We compare the performance of our method with 9 relational embedding methods. The first 4 methods use AmpliGraph library[7] and the last 4 use PyKEEN python package[8]. **ConvE** [5]: uses convolutions over embeddings and fully connected layers. **ComplEx** [28]: uses a latent factorization based approach for the problem of link prediction. **DistMult** [31]: learns representations of entities and relations as low-dimensional vectors and bilinear and/or linear mapping functions. **HolE** [21]: uses circular correlation of the vector representations of entities to create holographic embeddings. **SimplE** [12]: adapts the concept of Canonical Polyadic decomposition to learn two dependent embeddings for each entity and relation. We use the tensorflow implementation[9]. **TransE/H/R/D** [2,30,17,10]: are different translation based relational embedding methods.

Baselines (Rule Learning methods): We compare our method to 3 rule learning methods. **Handcrafted rules (Gaifman)** [23]: consists of handwritten rules that are then simply enumerated following the Gaifman locality. **Neural LP** [32]: learns the first-order rules by ion an end-to-end differentiable model. We use author provided code[10] with #rules learned = 10. **metapath2vec** [7]:

[7] https://github.com/Accenture/AmpliGraph
[8] https://github.com/pykeen/pykeen
[9] https://github.com/Mehran-k/SimplE
[10] https://github.com/fanyangxyz/Neural-LP

generates random walks with user defined metapaths and uses a skip-gram model to generate embeddings. We use metapath2vec in Stellargraph[11] package.

Results: Table 1 also shows the number of relational rules learned by different techniques. Table 2 presents the results for all the relational domains (5-fold cross validation) with logistic regression (LR) and gradient boosting (GB)[12]. All experiments were run with parameter values r=1, k=10 and w=5.

Table 2: Comparison against SRL methods for the relational domains.

Data set	Methods	Accuracy		Recall		F1		AUC-ROC		AUC-PR	
		LR	GB	LR	GB	LR	GB	LR	GB	LR	GB
DDI	RW	0.657	0.669	0.469	0.530	0.564	0.602	0.647	0.662	0.581	0.593
	ILP	0.696	0.774	0.467	0.674	0.592	0.729	0.684	0.767	0.710	0.765
	relOCC	0.860	**0.897**	0.939	**0.991**	0.864	**0.901**	0.864	**0.902**	0.797	**0.853**
	Gaifman	0.534	0.771	0.469	0.658	0.564	0.691	0.672	0.697	0.581	0.710
	MLN-Boost	0.638		0.504		0.618		0.798		0.784	
	RDN-Boost	0.755		0.662		0.718		0.828		0.831	
PPI	RW	0.700	**0.785**	0.586	0.707	0.661	0.767	0.699	**0.785**	0.651	**0.740**
	ILP	0.613	0.661	0.397	0.553	0.506	0.620	0.613	0.661	0.579	0.614
	relOCC	0.727	0.733	0.996	**0.999**	0.785	**0.789**	0.727	0.733	0.647	0.652
	Gaifman	0.608	0.652	0.382	0.524	0.499	0.606	0.613	0.654	0.591	0.619
	MLN-Boost	0.548		0.453		0.571		0.743		0.733	
	RDN-Boost	0.671		0.615		0.652		0.728		**0.740**	
NELL Sports	RW	0.783	0.822	0.414	0.569	0.569	0.689	0.696	0.762	0.565	0.594
	ILP	0.782	0.824	0.431	0.590	0.578	0.699	0.699	0.769	0.530	0.564
	relOCC	0.793	**0.833**	0.431	0.6	0.59	**0.731**	0.708	0.778	0.574	0.643
	Gaifman	0.756	0.780	0.314	0.485	0.465	0.597	0.648	0.707	0.512	0.549
	MLN-Boost	0.605		0.533		0.667		**0.894**		**0.853**	
	RDN-Boost	0.812		**0.756**		0.714		0.884		0.834	
Financial NLP	RW	0.833	0.833	0.0	0.0	0.0	0.0	0.5	0.5	0.168	0.168
	ILP	0.838	0.921	0.068	0.633	0.112	0.727	0.530	0.806	0.200	0.6023
	relOCC	0.965	0.967	0.788	0.800	0.882	0.889	0.867	0.879	0.826	0.833
	Gaifman	0.827	0.914	0.0	0.59	0.0	0.705	0.5	0.787	0.173	0.587
	MLN-Boost	0.928		0.764		0.757		**0.989**		0.807	
	RDN-Boost	**0.975**		**0.963**		**0.929**		**0.989**		**0.901**	
ICML CoAuthor	RW	0.977	0.977	0.0	0.0	0.0	0.0	0.5	0.5	0.023	0.023
	ILP	0.983	0.985	0.272	0.339	0.427	0.506	0.636	0.669	0.289	0.356
	relOCC	0.986	**0.997**	0.346	0.386	0.517	**0.557**	0.653	**0.693**	0.370	**0.40**
	Gaifman	0.981	0.984	0.100	0.327	0.174	0.493	0.529	0.664	0.127	0.343
	MLN-Boost	0.938		0.326		0.214		0.294		0.210	
	RDN-Boost	0.940		**0.434**		0.231		0.153		0.157	

[Q1] Comparing different structure learning strategies: We compare the structure learning method by relOCC to two commonly used relational rule learning techniques, relational random walks and inductive logic programming and the results are shown in table 2. We note that relOCC outperforms the baselines ILP and relational RWs methods across a majority of the domains. This is expected since relOCC considers the density of the positive and negative examples separately, as opposed to the other rule learning methods, allowing the features it generates to discriminate better. This answers **Q1**.

[Q2] Effect of class imbalance: Imbalanced data sets are difficult to learn from for the classical machine learning algorithms since it is assumed that the number

[11] https://pypi.org/project/stellargraph/

[12] For performance of algorithms other than LR and GB see `https://bit.ly/3jp9NA2`

of examples are generally equally distributed among the classes to be predicted. We consider the ICML CoAuthor (neg-to-pos ratio of 42:1) data set which is highly imbalanced and Financial NLP (neg-to-pos ratio of 6:1) data set which is relatively imbalanced; consequently, we report AUC-PR. In both domains, AUC-PR for relOCC outperforms the other structure-learning methods by a large margin. Random walk rules, in particular, cause all the examples to be classified as negative, resulting in recall and F1-scores of 0 in both domains. Thus, we can answer **Q2**: highly-imbalanced domains benefit from density-estimation-based structure learning. This also verifies our hypothesis: *learning from the example densities separately and independently results in a better discriminative behavior.* **[Q3] Effect of locality parameters:** Fig. 3 shows the effects of varying r (depth of neighborhoods), k (number of neighbors) and w (number of neighborhoods) on the DDI data set. Generally, k does not affect performance significantly, but increasing r causes recall report to drop sharply. This is because, with $r = 1$, entities in the query neighborhood are more tightly coupled with entities in the query variables. This parametric sensitivity analysis addresses **Q3**. Also, another important takeaway is that relOCC rules exhibit *high clinically-relevant recall ($\approx$ 1) on medical data sets*: DDI and PPI. This has considerable implications for bioinformatics domains as recall is the most important metric; this is because a false negative (such as a misdiagnosis) results in much more serious consequences [6] than a false positive. Finally, from Fig. 3 (right), we note that varying r and k does not affect training time, as these parameters do not affect the search space. However, increasing w increases the run time since the size of the neighborhood graph to be searched increases.

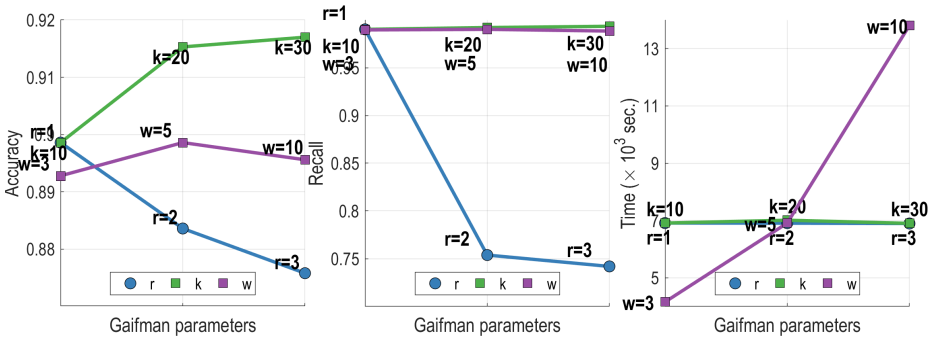
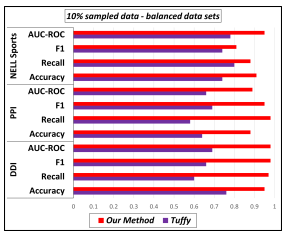


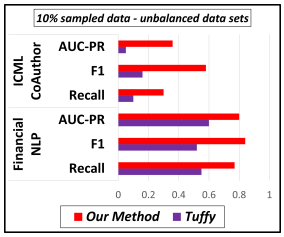
Fig. 3: (left) Accuracy, (middle) recall and (right) running time for various values of r, k and w for the DDI domain. For varying r: w=5 and k=10, for varying w: r=1 and k=10 and for varying k: w=5 and r=1.

[Q4] Comparison with PILP systems: *Our core contribution is end-to-end learning of Gaifman models that requires only data and no domain knowledge and thus we focus on comparison with a full model learning methods of MLN-Boost and RDN-Boost.* Table 2 shows that our method outperforms MLN-Boost by a

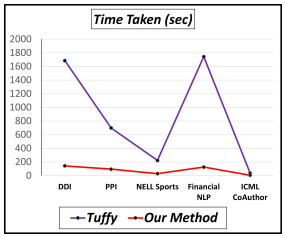
significant margin and outperforms/is comparable to the performance of RDN-Boost in 4 out of 5 domains. We also note that Tuffy[13] could not effectively scale to the amount of data that we have used in our learning framework, and could not learn the structure. Instead, we tried using the ILP rules that we learned, and learned the weights. In this case as well, Tuffy could not complete training after a few hours. To put this in perspective, we sampled 10% data from all the data sets and the results for the same are presented in Fig. 4 which shows that our method is significantly better than Tuffy on both balanced and unbalanced data sets. Figure 4c shows the time taken by our method as compared to Tuffy on the sampled data set. Since Tuffy does not support structure learning i.e. rule learning, in order to keep the comparisons fair, we use the rules learned by relOCC, convert them into the Tuffy format and run the inference. Thus, for Tuffy we report only the inference time and compare it with the inference time of our method (grounding + machine learning algorithm). We are able to perform the inference far quicker than compared to Tuffy. Thus, to answer **Q4**, we outperform several SOTA probabilistic ILP methods across domains.



(a) Comparison of our method with Tuffy on balanced data sets.



(b) Comparison of our method with Tuffy on unbalanced data sets.



(c) Comparison of running times of our method and Tuffy.

Fig. 4: Comparison of our method and Tuffy on 10% sampled data sets.

[Q5] Comparison with relational embedding models: To answer **Q5**, we compare against 9 state-of-the-art relational embedding methods. Table 3 shows that our method outperforms all relational embeddings by a huge margin especially in the case of imbalanced data sets i.e. Financial NLP and ICML CoAuthor. These results show the importance of constructing first order rules from the given data instead of directly using the triples since the inherent structure of the underlying graph can be captured by our method.

[Q6] Comparison with rule learning methods: Finally, to answer **Q6**, we compared against 2 state-of-the-art rule learning methods, NeuralLP and metapath2vec, and hand-written rules. For the handwritten rules, we created generic relational features as suggested by Niepert and of the form: $r(e1, e2)$; $r(e2, e1)$; $\exists x\ r(x, e)$, $\exists x\ r(e, x)$, $\exists x\ r(e1, x) \wedge r(x, e2)$, $\exists x\ r(e2, x) \wedge r(x, e1)$. These re-

[13] We also tried other systems: Alchemy, Problog, ProbCog.

Table 3: Comparison against relational embedding methods with results. We report the results for our method using relOCC rules with gradient boosting.

Data set	Metric	ConvE	ComplEx	SimplE	DistMult	HolE	TransE	TransH	TransR	TransD	relOCC
DDI	Accuracy	0.744	0.787	0.509	0.683	0.586	0.533	0.479	0.465	0.476	**0.897**
	Recall	0.931	0.832	0.051	0.988	0.922	0.522	0.662	0.802	0.793	**0.991**
	F1	0.544	0.618	0.030	0.567	0.483	0.320	0.348	0.387	0.389	**0.901**
	AUC-ROC	0.744	0.818	0.195	0.962	0.844	0.541	0.554	0.653	0.659	**0.902**
	AUC-PR	0.678	0.705	0.118	0.912	0.641	0.231	0.222	0.313	0.332	**0.853**
PPI	Accuracy	**0.747**	0.676	0.739	0.787	0.500	0.390	0.388	0.417	0.446	0.733
	Recall	0.685	0.603	0.793	0.707	0.0	0.401	0.408	0.449	0.512	**0.999**
	F1	0.729	0.650	0.752	0.768	0.0	0.397	0.400	0.435	0.480	**0.789**
	AUC-ROC	**0.829**	0.732	0.828	0.823	0.500	0.332	0.331	0.385	0.424	0.733
	AUC-PR	**0.855**	0.704	0.843	0.870	0.500	0.400	0.385	0.430	0.447	0.652
NELL Sports	Accuracy	0.667	0.629	0.548	0.607	0.756	0.544	0.530	0.470	0.448	**0.833**
	Recall	0.711	**0.733**	0.633	0.633	0.633	0.622	0.600	0.489	0.511	0.600
	F1	0.587	0.569	0.484	0.518	0.633	0.477	0.460	0.381	0.382	**0.731**
	AUC-ROC	0.743	0.762	0.620	0.694	0.745	0.589	0.571	0.456	0.489	**0.778**
	AUC-PR	0.517	0.628	0.437	0.645	0.730	0.452	0.423	0.332	0.3	**0.643**
Financial NLP	Accuracy	0.796	0.634	0.421	0.708	0.848	0.526	0.501	0.551	0.584	**0.967**
	Recall	0.963	0.472	**0.964**	0.982	0.0	0.673	0.527	0.691	0.691	0.800
	F1	0.589	0.281	0.335	0.505	0.0	0.301	0.243	0.318	0.335	**0.889**
	AUC-ROC	**0.953**	0.574	0.779	0.918	0.5	0.631	0.485	0.648	0.711	0.879
	AUC-PR	0.765	0.232	0.359	0.749	0.152	0.225	0.139	0.278	0.402	**0.833**
ICML CoAuthor	Accuracy	0.981	0.977	0.985	0.515	0.992	0.494	0.500	0.389	0.467	**0.997**
	Recall	0.636	0.85	0.200	0.964	0.0	0.909	0.836	0.727	**1.0**	0.386
	F1	0.020	0.030	0.007	0.032	0.0	0.029	0.027	0.020	0.031	**0.557**
	AUC-ROC	0.005	0.018	0.010	0.921	0.500	0.790	0.691	0.502	0.858	**0.693**
	AUC-PR	0.015	0.040	0.005	0.640	0.008	0.031	0.015	0.008	0.043	**0.400**

lational features are very simple, and do not cover the relational search space sufficiently, resulting in significantly poor performance. And hence, we created more domain-specific rules to enhance the score. For NeuralLP, the number of rules learned = 10 and for metapath2vec, the length of the learned random walk = 100, with the number of metapaths for each data set being: DDI = 3, PPI = 6, NELL Sports = 9, Financial NLP = 2 and ICML CoAuthor = 4.

It is clear from the results (Tab. 4) that even after enhancing the handcrafted rules and using different rule learning methods, the rules learned by density estimation leads to much better predictive models thus answering **Q6**.

5 Conclusion and Future Work

We propose the first work for full model learning for relational data using Gaifman locality theorem. In addition to exploring the viability of established structure learning methods we proposed a novel structure-learning approach based on relational density estimation. We constructs a set of rules, identify the appropriate instantiations and finally count the number of groundings per rule to obtain embeddings. We then train a discriminative classifier thus providing an effective method of doing link prediction. There are several avenues to explore such as joint learning of Gaifman models, generating explanations for a given prediction and extending Gaifman locality to hypergraphs. Another direction is employing more graph based embedding methods that can integrate with Gaifman's locality principle. Finally, evaluating on more real databases and knowledge graphs is an interesting direction.

Table 4: Comparison against several rule learning strategies. We use gradient boosting for our method using relOCC rules and handwritten rules (Gaifman).

Data set	Methods	Accuracy	Recall	F1	AUC-ROC	AUC-PR
DDI	Gaifman	0.771	0.658	0.691	0.697	0.710
	Neural LP	0.632	0.777	0.470	0.741	0.404
	metapath2vec	0.717	0.767	0.717	0.768	0.696
	relOCC	**0.897**	**0.991**	**0.901**	**0.902**	**0.853**
PPI	Gaifman	0.652	0.524	0.606	0.654	0.619
	Neural LP	0.395	0.336	0.357	0.345	0.440
	metapath2vec	0.642	0.767	0.715	0.660	**0.729**
	relOCC	**0.733**	**0.999**	**0.789**	**0.733**	0.652
NELL Sports	Gaifman	0.780	0.485	0.597	0.707	0.549
	Neural LP	0.663	0.400	0.442	0.583	0.412
	metapath2vec	0.778	**0.867**	**0.765**	**0.875**	**0.850**
	relOCC	**0.833**	0.600	0.731	0.778	0.643
Financial NLP	Gaifman	0.914	0.590	0.705	0.787	0.587
	Neural LP	0.705	0.745	0.434	0.768	0.314
	metapath2vec	0.699	**0.982**	0.568	**0.927**	0.675
	relOCC	**0.967**	0.800	**0.889**	0.879	**0.833**
ICML CoAuthor	Gaifman	0.984	0.327	0.493	0.664	0.343
	Neural LP	0.718	**0.800**	0.045	0.846	0.179
	metapath2vec	0.912	**0.800**	0.333	**0.922**	0.350
	relOCC	**0.997**	0.386	**0.557**	0.693	**0.400**

Acknowledgments

This work is supported by the Air Force Office of Scientific Research under award number FA9550-191-0391. SN also acknowledges AFOSR award FA9550-18-1-0462. Any opinions, findings, conclusion or recommendations expressed are those of the authors and do not necessarily reflect the view of AFOSR or the US government. DSD also acknowledges ICT-48 Network of AI Research Excellence Center "TAILOR" (EU Horizon 2020, GA No 952215) and the Collaboration Lab "AI in Construction" (AICO).

References

1. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. In: Artificial intelligence. vol. 101, pp. 285–297. Elsevier (1998)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)
3. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. In: IEEE TKDE (2018)
4. De Campos, C.P., Zeng, Z., Ji, Q.: Structure learning of bayesian networks using constraints. In: ICML (2009)
5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI (2018)
6. Dhami, D.S., Kunapuli, G., Das, M., Page, D., Natarajan, S.: Drug-drug interaction discovery: Kernel learning from heterogeneous similarities. In: Smart Health (2018)

7. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: KDD (2017)
8. Gaifman, H.: On local and non-local properties. In: Studies in Logic and the Foundations of Mathematics (1982)
9. Getoor, L., Taskar, B.: Intro to statistical relational learning. MIT press (2007)
10. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: ACL-IJCNLP (2015)
11. Kaur, N., Kunapuli, G., Khot, T., Kersting, K., Cohen, W., Natarajan, S.: Relational restricted boltzmann machines: A probabilistic logic learning approach. In: ILP (2017)
12. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: NeurIPS (2018)
13. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning markov logic networks via functional gradient boosting. In: ICDM (2011)
14. Khot, T., Natarajan, S., Shavlik, J.W.: Relational one-class classification: A non-parametric approach. In: AAAI (2014)
15. Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., Lowd, D., Wang, J., Domingos, P.: The alchemy system for statistical relational {AI} (2009)
16. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. In: Machine learning (2010)
17. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI (2015)
18. Mitchell, T., Cohen, W., et al.: Never-ending learning. In: ACM Communications (2018)
19. Muggleton, S.: Inverse entailment and progol. In: New generation computing (1995)
20. Natarajan, S., Khot, T., Kersting, K., Gutmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. In: Machine Learning (2012)
21. Nickel, M., Rosasco, L., Poggio, T.A., et al.: Holographic embeddings of knowledge graphs. In: AAAI (2016)
22. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. In: NIPS (2017)
23. Niepert, M.: Discriminative gaifman models. In: NIPS (2016)
24. Niu, F., Ré, C., Doan, A., Shavlik, J.: Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. In: VLDB (2011)
25. Raedt, L.D., Kersting, K., Natarajan, S., Poole, D.: Statistical relational artificial intelligence: Logic, probability, and computation. In: Synthesis Lectures on AI and ML (2016)
26. Srinivasan, A.: The aleph manual (2001)
27. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: NIPS (2004)
28. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. ICML (2016)
29. Walker T., e.a.: Ilp for bootstrapped learning: A layered approach to automating the ilp setup problem. In: ILP (2009)
30. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI (2014)
31. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
32. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: NeurIPS (2017)