Relational Restricted Boltzmann Machines: A Probabilistic Logic Learning Approach

Navdeep Kaur[1], Gautam Kunapuli[2], Tushar Khot[3], Kristian Kersting[4], William Cohen[5], Sriraam Natarajan[1]

[1] Indiana University, Bloomington, USA [2] UtopiaCompression Corporation, USA
[3] Allen Institute of Artificial Intelligence, USA [4] TU Dortmund University, Germany
[5] Carnegie Mellon University, USA
[1]{navdkaur,natarasr}@indiana.edu, [2]gkunapuli@gmail.com,
[3]tushar.v.khot@gmail.com, [4]kristian.kersting@cs.tu-dortmund.de,
[5]wcohen@gmail.com

Abstract. We consider the problem of learning Boltzmann machine classifiers from relational data. Our goal is to extend the deep belief framework of RBMs to statistical relational models. This allows one to exploit the feature hierarchies and the non-linearity inherent in RBMs over the rich representations used in statistical relational learning (SRL). Specifically, we use lifted random walks to generate features for predicates that are then used to construct the observed features in the RBM in a manner similar to Markov Logic Networks. We show empirically that this method of constructing an RBM is comparable or better than the state-of-the-art probabilistic relational learning algorithms on four relational domains.

Introduction

Restricted Boltzmann machines (RBMs)([23]) are popular models for learning probability distributions due to their expressive power. Consequently, they have been applied to various tasks such as collaborative filtering [31], motion capture [32] and others. Similarly, there has been significant research on the theory of RBMs: approximating log-likelihood gradient by contrastive divergence (CD) [11], persistent CD [33], parallel tempering [7] , extending them to handle real-valued variables and discriminative settings. While these models are powerful, they make the standard assumption of using *flat feature vectors* to represent the problem.

In contrast to flat-feature representations, Statistical Relational Learning (SRL) [10, 5] methods use richer symbolic features during learning; however, they have not been fully exploited in deep-learning methods. Learning SRL models is computationally intensive [26] however, particularly model structure (qualitative relationships). This is due to the fact that structure learning requires searching over objects, their attributes, and attributes of related objects. Hence, the state-of-the-art learning method for SRL models learns a series of weak relational rules that are combined during prediction. While empirically successful, this method leads to rules that are dependent on each other making them uninterpretable. We propose to use a set of interpretable rules based on the successful Path Ranking Algorithm (PRA, [21]). Recently, [14] have employed logical rules to enhance the representation of neural networks. There has also been work on lifting neural networks to relational settings [2]. While the specific methodologies differ, all these methods employ relational and logic rules as features of neural networks and train them on relational data. In this spirit, we propose a methodology for lifting RBMs to relational data. While previous methods on lifting relational networks employed logical constraints or templates, we use relational random walks to construct relational rules, which are then used as features in a RBM.

We propose two approaches to instantiating RBM features: (1) similar to the approach of Markov Logic Networks (MLNs) [8] and relational logistic regression [15], we instantiate features with *counts* of the number of times that a random walk is satisfied for every training example; and (2) similar to Relational Dependency Networks [25], we instantiate features with *existentials* (that is, the feature will be 1 if there exists at least one instantiation of the path in the data, otherwise it is 0). Given these feature values, we train a discriminative RBM with the following assumptions: the input layer is multinomial (to capture counts and existentials), the hidden layer is sigmoidal, and the output layer is Bernoulli.

We make the following contributions: (1) we combine the powerful formalism of RBMs with the representation ability of relational logic; (2) we develop a *relational RBM* that does not fully propositionalize the data; (3) we show the connection between our proposed method and previous approaches such as RDNs, MLNs and relational logistic regression, and finally, (4) we demonstrate the effectiveness of this novel learning approach by empirically comparing against state-of-the-art methods that also learn from relational data.

Prior Work and Background

Random walks: Relational data is often represented using a ground (or lifted) graph. The constants (or types) form the nodes and the ground atoms (or predicates) form the edges. N-ary predicates can be represented with hyperedges or multiple binary relations, where a node is introduced for every ground atom (or predicate) and edges are introduced from this node to each argument. This graph representation allows the use of many path-based approaches for discovering the structure of the data. A path in a ground relational graph (where nodes are constants) corresponds to a conjunction of ground atoms. For example, the path $\mathtt{s_1}$-$\mathtt{takes}$-$\mathtt{c_1} - \mathtt{taughtBy} - \mathtt{p_1}$ describes an example where the student $\mathtt{s_1}$ takes class $\mathtt{c_1}$ taught by professor $\mathtt{p_1}$. In a ground relational graph, this path can be converted to: $\mathtt{takes}(\mathtt{s_1}, \mathtt{c_1}) \wedge \mathtt{taughtBy}(\mathtt{c_1}, \mathtt{p_1})$. In contrast, in a lifted relational graph (where nodes are types), paths are conjunctions of predicate with shared variables: $\mathtt{takes}(\mathtt{S}, \mathtt{C}) \wedge \mathtt{taughtBy}(\mathtt{C}, \mathtt{P})$.

Relational Probabilistic Models: MLNs [8] are relational undirected models, where first-order logic formulas correspond to cliques of a Markov network and formula weights correspond to the clique potentials. An MLN can be instantiated as a Markov network with a node for each ground predicate (atom) and a clique for each ground formula. All groundings of the same formula are assigned the same weight leading to the following joint probability distribution over all atoms: $P(X{=}x){=}\ \frac{1}{Z}\exp\left(\sum_i w_i n_i(x)\right)$, where $n_i(x)$ is the number of times the i-th formula is satisfied by possible world x, and Z is a normalization constant. We focus on discriminative learning, where we learn a conditional distribution of one predicate given all other predicates. One such discriminative model is relational logistic regression (RLR), which extends logistic regression to relational settings where training examples have different feature sizes.

Structure Learning Approaches: Many structure learning approaches for Probabilistic Logical Models (PLMs), including MLNs, use graph representations. For example, Learning via Hypergraph Lifting (LHL) [17] builds a hypergraph over ground atoms; LHL then clusters the atoms to create a "lifted" hypergraph, and traverses this graph to obtain rules. Learning with Structural Motifs (LSM) [18] performs random walks over the graph to cluster nodes and performs depth-first traversal to generate potential clauses. We use *random walks over a lifted graph* to generate all possible clauses, and then use a non-linear combination (through the hidden layer) of ground clauses, as opposed to linear combination in MLNs.

Propositionalization Approaches: To learn powerful deep models on relational data, propositionalization techniques are used to convert ground atoms into a fixed-length feature vector. For instance, kFoil [20] uses a *dynamic* approach to learn clauses to propositionalize relational examples for SVMs. The Path Ranking Algorithm (PRA) [21] creates features for a pair of entities by generating random walks from a graph. We use a similar approach to perform random walks on the lifted relational graph to learn the structure of our relational model.

Restricted Boltzmann Machines: Boltzmann machines (BMs) [23] model probability distributions and are interpretable as artificial neural networks [1]. A BM consists of visible units V (representing observations) and hidden units H (representing dependencies between features). A general BM is a fully-connected Markov random field [19], which makes learning computationally intensive. A more tractable model, the Restricted Boltzmann Machine (RBM), constrains the BM to a bipartite graph of visible and hidden units. A singular benefit of this representation is that hidden-layer outputs of one RBM can be used as input to another higher-level RBM, a procedure known as *stacking*; this results in the powerful deep belief networks [12].

RBMs have been used as feature extractors for other supervised learning problems [9] and to initialize deep neural network classifiers [13]. Recently, [22] proposed a standalone formulation for supervised classification called *discriminative RBMs*. We adopt this formalism to demonstrate our approach of combining rich first-order logic representations of relational data with non-linear classifiers learned by discriminative RBMs. We are given data, $\{(\mathbf{x}_i,\, \hat{y}_i)_{i=1}^{\ell}\}$, where each training example is a vector, $\mathbf{x}_i \in \mathbb{R}^m$, and multi-class labels $\hat{y}_i \in \{1, \ldots, C\}$. The training labels are represented by a one-hot vectorization: $\mathbf{y}_i \in \{0, 1\}^C$ with $y_i^k = 1$ if $\hat{y}_i = k$ and zero otherwise. For instance, in a three-class problem, if $\hat{y}_i = 2$, then $\mathbf{y}_i = [0,\, 1,\, 0]$. The goal is to train a classifier by maximizing the log-likelihood, $\mathcal{L} = \sum_{i=1}^{\ell} \log p(\mathbf{y}_i, \mathbf{x}_i)$. In this work, we employ discriminative RBMs.

Relational Restricted Boltzmann Machines

We now introduce Relational RBMs, a deep, relational classifier that can learn hierarchical relational features through its hidden layer and model non-linear decision boundaries. The idea is to use *lifted random walks* to generate relational features for predicates

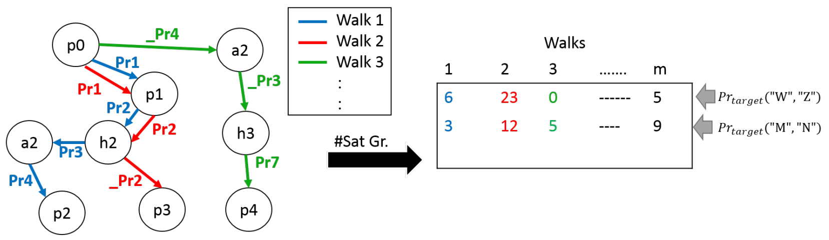


Fig. 1: Illustration of the transformation process. Random walks are converted into feature vectors by explicitly grounding every random walk (left) for every training example. Nodes and edges of the graph represent the objects and the predicates, respectively. The random walks counts (right) are then used as feature values for learning a discriminative RBM (DRBM).

that are then counted (or used as existentials) to become RBM features. Of course, more than one RBM could be trained, stacking them on top of each other. For the sake of simplicity, we focus on a single layer; however, our approach is easily extended to multiple layers. Our learning task can be defined as follows:

Given: Relational data, D; Target Predicate, T.
Learn: Relational Restricted Boltzmann Machine (RRBM) in a discriminative fashion.

We make some key modeling assumptions:

1. input layers (relational features) are modeled using a multinomial distribution, for counts or existentials;
2. the output layer (target predicate) is modeled using a Bernoulli distribution
3. hidden layers are continuous, with a range in $[0, 1]$.

Step 1: Relational data transformation: We use a lifted-graph representation to model relational data, D. Each type corresponds to a node in the graph and the predicate $\mathtt{r}(\mathtt{t}_1, \mathtt{t}_2)$ is represented using a directed edge from the node $\mathtt{t}_1$ to $\mathtt{t}_2$. For N-ary predicates, say $\mathtt{r}(\mathtt{t}_1, ..., \mathtt{t}_\mathtt{n})$, we introduce a special compound value type (CVT)[1], $\mathtt{r}_\mathtt{CVT}$. For each argument $\mathtt{t}_\mathtt{k}$, an edge $\mathtt{e}_{\mathtt{rk}}$ is added between the nodes $\mathtt{r}_\mathtt{CVT}$ and $\mathtt{t}_\mathtt{k}$. Similarly for unary predicates, $\mathtt{r}(\mathtt{t})$ we create a binary predicate $\mathtt{isa}(\mathtt{t}, \mathtt{r})$.

Step 2: Relational transformation layer: Now, we generate the input feature vector $\mathbf{x}_i$ from a relational example, $\mathtt{T}(\mathtt{a}_{1\mathtt{j}}, \mathtt{a}_{2\mathtt{j}})$. Inspired by the Path Ranking Algorithm [21], we use random walks on our lifted relational graph to encode the local relational structure for each example. We generate m unique random walks connecting the argument types for the target predicate to define the m dimensions of $\mathbf{x}$. Since random walks also correspond to the set of candidate clauses considered by structure-learning approaches for MLNs [17, 18], this transformation function can be viewed as the *structure of our relational model*. A key feature of an RBM trained on standard i.i.d. data is that the feature set $\mathbf{x}$ is defined in advance and is finite. With relational data, this set

[1] `wiki.freebase.com/wiki/Compound_Value_Type`

can potentially be infinite, and feature size can vary with *each training instance*. For instance, if the random walk is a paper written by a $\texttt{professor} - \texttt{student}$ combination, not all $\texttt{professor} - \texttt{student}$ combinations will have the same number of feature values. This is commonly referred as *multiple-parent* problem [27]. One approach to the multiple-parent problem is to consider *existential semantics*, i.e., if there exists at least one instance of the random walk that is satisfied for an example, the feature value corresponding to that walk is set to 1. This approach was also recently (and independently of our paper) used by [34] for ranking via matrix factorization. This leads to our first model: RRBM-E, where E denotes the existential semantics used to construct the RRBM.

Inspired by MLNs, we also consider the *counts* of the random walks as feature values generated from a path, and denote this model as RRBM-C (Figure 1). For example, if a $\texttt{professor} - \texttt{student}$ combination has written 10 papers, the feature value corresponding to this random walk for that combination is 10. To summarize, there are two possible definitions of our transformation function, $\mathbf{x}_j = g(\texttt{a}_{\texttt{1j}}, \texttt{a}_{\texttt{2j}})$

- $g_c(\texttt{a}_{\texttt{1j}}, \texttt{a}_{\texttt{2j}}, \texttt{p})$ = #groundings of p^{th} random walk connecting object $\texttt{a}_{\texttt{1j}}$ to $\texttt{a}_{\texttt{2j}}$ (RRBM-C),
- $g_e(\texttt{a}_{\texttt{1j}}, \texttt{a}_{\texttt{2j}}, \texttt{p}) = 1$, if $\exists$ a grounding of the p^{th} random walk connecting $\texttt{a}_{\texttt{1j}}$ to $\texttt{a}_{\texttt{2j}}$, otherwise 0 (RRBM-E).

For example, consider that the walk $\texttt{takes}(\texttt{S}, \texttt{C}) \wedge \texttt{taughtBy}(\texttt{C}, \texttt{P})$ is used to generate a feature for $\texttt{advisedBy}(\texttt{s}_1, \texttt{p}_1)$. The feature from g_c would be set to the count: $|\{\texttt{C} \,|\, \texttt{takes}(\texttt{s}_1, \texttt{C}) \wedge \texttt{taughtBy}(\texttt{C}, \texttt{p}_1)\}|$. With second function, g_e, this feature would be set to 1, if $\exists \texttt{C}, \texttt{takes}(\texttt{s}_1, \texttt{C}) \wedge \texttt{taughtBy}(\texttt{C}, \texttt{p}_1)$. Using features from g_e to learn weights for a logistic regression model would lead to an RLR model, while using features from g_c would correspond to learning an MLN (as we show later). One could also imagine using RLR as an aggregator from these random walks, but that is a direction for future work. While counts are more informative and connect to existing SRL formalisms such as MLNs, exact counting is computationally expensive in relational domains. This can be mitigated by using approximate counting approaches, such as the one due to [3] that leverages the power of graph databases. Our current empirical evaluation did not require the count approximation and we defer the integration of approximate counting techniques to future research.

Step 3: Learning Relational RBMs: The output of the relational transformation layer is fed into multi-layered DRBM to learn a regularized, non-linear, weighted combination of features. The relational transformation layer stacked on top of the DRBM forms the Relational RBM model. Due to non-linearity, we can to learn a much more expressive model than traditional MLNs and RLRs.

Recall that the DRBM as defined by [22] consists of n hidden units, $\mathbf{h}$, and the joint probability is modeled as $p(\mathbf{y}, \mathbf{x}, \mathbf{h}) \propto e^{-E(\mathbf{y},\mathbf{x},\mathbf{h})}$, where the energy function is parameterized $\Theta \equiv (W, \mathbf{b}, \mathbf{c}, \mathbf{d}, U)$:

$$E(\mathbf{y}, \mathbf{x}, \mathbf{h}) = -\mathbf{h}^T W \mathbf{x} - \mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{h} - \mathbf{d}^T \mathbf{y} - \mathbf{h}^T U \mathbf{y}. \tag{1}$$

Algorithm 1 `LearnRRBM(T, G, P)`: Relational Restricted Boltzmann Machines
Input `T(t1, t2)`: target predicate, G: lifted graph over types, m: number of features

```
 1: ▷ Generate $m$ random walks between $\mathtt{t}_1$ and $\mathtt{t}_2$
 2: $\mathbf{rw}$ := PerformRandomWalks($G$, $\mathtt{t}_1$, $\mathtt{t}_2$, $m$)
 3: for $0 \le j < l$ do   ▷ Iterate over all training examples
 4:     ▷ Generate features for $\mathtt{T}(\mathtt{a_{1j}}, \mathtt{a_{2j}})$
 5:     for $0 \le p < m$ do   ▷ Iterate over all the paths
 6:         ▷ $p^{th}$ feature computed from the arguments of $x_j$
 7:         $x_j[p]$ := $g_c(\mathtt{a_{1j}}, \mathtt{a_{2j}}, \mathbf{rw}[p])$
 8:     end for
 9: end for
10: $\mathbf{x} := \{\mathbf{x}_j\}$   ▷ Input matrix
11: ▷ Learn DRBM from the features and examples
12: $\Theta$ := LearnDRBM($\mathbf{x}$, $\mathbf{y}$)
13: return RRBM($\Theta$, $\mathbf{rw}$)
```

As with most generative models, computing joint probability $p(\mathbf{y}, \mathbf{x})$ is intractable, but the conditional distribution $P(\hat{y}|\mathbf{x})$ can be computed exactly [31] as

$$p(\hat{y}|\mathbf{x}) = \frac{e^{d_{\hat{y}} + \sum_{j=1}^{n} \sigma(c_j + U_{j\hat{y}} + \sum_{f=1}^{m} W_{jf} x_f)}}{\sum_{k=1}^{C} e^{d_k + \sum_{j=1}^{n} \sigma(c_j + U_{jk} + \sum_{f=1}^{m} W_{jf} x_f)}}. \tag{2}$$

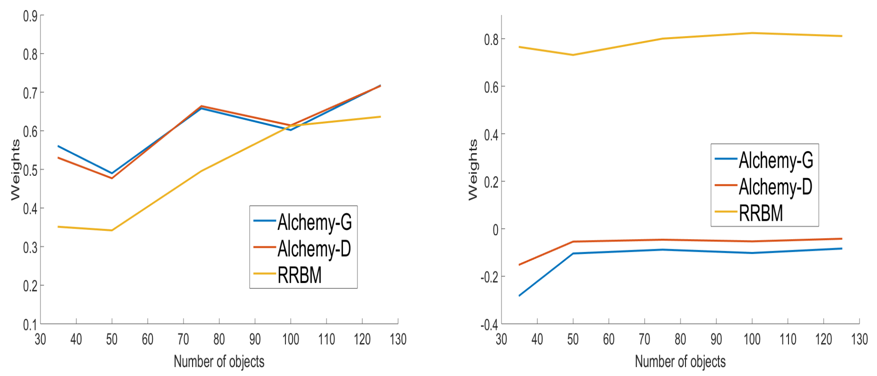
In (2), $\sigma(z) = e^z / \sum_i e_i^z$, the logistic softmax function and the index f sums over all the features x_f of a training example $\mathbf{x}$. During learning, the log-likelihood function is maximized to compute the DRBM parameters Θ. The gradient of the conditional probability (Eqn. 2) can be computed as:

$$\frac{\partial}{\partial \theta} \log p(\hat{y}_i|\mathbf{x}_i) = \sum_{j=1}^{n} \sigma\left(o_{\hat{y}j}(\mathbf{x}_i)\right) \frac{\partial o_{\hat{y}j}(\mathbf{x}_i)}{\partial \theta} + \sum_{k=1}^{C} \sum_{j=1}^{n} \sigma\left(o_{kj}(\mathbf{x}_i)\right) p(k|\mathbf{x}_i) \frac{\partial o_{kj}(\mathbf{x}_i)}{\partial \theta}. \tag{3}$$

In (3), $o_{\hat{y}j}(\mathbf{x}_i) = c_j + U_{j\hat{y}} + \sum_{f=1}^{m} W_{jf} x_{if}$, where $\mathbf{x}$ refers to random-walk features for every training example. As mentioned earlier, we assume that input features are modeled using a multinomial distribution. To consider counts as multinomials, we use an upper bound on counts: $2 \max(\mathtt{count}(x_i^j))$ for every feature; bounds are the same for both train and test sets to avoid overfitting. The complete approach to learn Relational RRBMs is shown in Algorithm 1.

Relation to Probabilistic Relational Models: The random walks can be interpreted as logical clauses (that are used to generate features) and the DRBM input feature weights $\mathbf{b}$ in (1) can be interpreted as clause weights (w_p). This interpretation highlights connections between our approach and Markov logic networks. Intuitively, the relational transformation layer captures the structure of MLNs and the RBM layer captures the weights of the MLNs. More concretely, $exp(\mathbf{b}^T\mathbf{x})$ in (1) can be viewed as $exp(\sum_p w_p n_p(\mathbf{x}))$ in the probability distribution for MLNs.

To verify this intuition, we compare the weights learned for clauses in MLNs to weights learned by `RRBM-C`. We generated a synthetic dataset for a university domain



(a) Co-author clause (b) Course clause

Fig. 2: Weights learned by Alchemy and RRBMs for a clause as the size of the domain increases.

with varying number of objects (professors and students). We picked a subset of professor - student pairs to have an `advisedBy` relationship and add common papers or common courses based on the following two clauses:

1. $\texttt{author}(\texttt{A},\texttt{P}) \wedge \texttt{author}(\texttt{B},\texttt{P}) \rightarrow \texttt{advisedBy}(\texttt{A},\texttt{B})$
2. $\texttt{teach}(\texttt{A},\texttt{C}) \wedge \texttt{register}(\texttt{B},\texttt{C}) \rightarrow \texttt{advisedBy}(\texttt{A},\texttt{B})$

Figure 2 shows the weights learned by discriminative and generative weight learning in Alchemy and RRBM for these two clauses as a function of the number of objects in the domain. Recall that in MLNs, the weight of a rule captures the confidence in that rule — the higher the number of instances satisfying a rule,the higher is the weight of the rule. As a result, the weight of the rule learned by Alchemy also increases in Figure 2. We observe a similar behavior with the weight learned for this feature in our RRBM formulation as well. While the exact values differ due to difference in the model formulation, this illustrates clearly that the intuitions of the model parameters from standard PLMs are still applicable.

In contrast to standard PLMs, RRBMs are not a shallow architecture. This can be better understood by looking at the rows of the weights W in the energy function (Eqn. (1)). They act as additional filter features, combining different clause counts. That is, $E(\mathbf{y}, \mathbf{x}, \mathbf{h})$ looks at how well the usage profile of a clause aligns with different filters associated with rows $W_{j\cdot}$. These filters are shared across different clauses, but different clauses will make comparisons with different filters by controlling clause-dependent biases U_{jy} in the σ terms. Notice also that two similar clauses could share some filters in W, that is, both could simultaneously have large positive values of U_{jy} for some rows $W_{j\cdot}$. This can be viewed as a form of statistical predicate invention as it discovers new concepts and is akin to (discriminative) second-order MLNs. In contrast to second-order MLNs, however, no second-order rules are required as input to invent new concepts.

Experiments

To compare RRBM approaches to state-of-the-art algorithms, we consider `RRBM-E`, `RRBM-C`and `RRBM-EC`. The last approach, `RRBM-EC` combines features from both existential and count RRBMs (i.e., union of count and existential features). Our experiments seek to answer the following questions:

Q1: How do `RRBM-E` and `RRBM-C` compare to a baseline MLN and Decision trees learning algorithms?
Q2: How do `RRBM-E` and `RRBM-C` compare to the state-of-the-art SRL approaches?
Q3: How do `RRBM-E`, `RRBM-C`, and `RRBM-EC` generalize across all domains?

To answer **Q1**, we compare RRBMs to Learning with Structural Motifs (LSM) [18] and Decision trees [29] whose features are obtained from count transformation layer, which we denote `Tree-C`. We use decision-trees as a proof-of-concept for demonstrating that a good probabilistic model when combined with our random walk features can potentially yield better results than naive combination of ML algorithm with the features. For LSM, we used the parameters recommended by [18]. However, we set the path length to 6 to be consistent with the path length used in RRBM. For parameter learning using Alchemy, we used both discriminative and generative weight-learning and present the best-performing result.

To answer **Q2**, we compare `RRBM-C` to MLN-Boost [16], and `RRBM-E` to RDN-Boost [25] both of which are SRL models that learn the structure and parameters simultaneously. For MLN-Boost and RDN-Boost, we used default settings and 20 gradient steps.

For RRBM, since path-constrained random walks [21] are performed on binary predicates, we convert unary and ternary predicates into binary predicates. For example, predicates such as $\mathtt{teach}(\mathtt{a1}, \mathtt{a2}, \mathtt{a3})$ are converted to three binary predicates: $\mathtt{teachArg1}(\mathtt{id}, \mathtt{a1})$, $\mathtt{teachArg2}(\mathtt{id}, \mathtt{a2})$, $\mathtt{teachArg3}(\mathtt{id}, \mathtt{a3})$ where `id` is the unique identifier for a predicate. As another example, unary predicates such as $\mathtt{student}(\mathtt{s1})$ are converted to binary predicates of the form $\mathtt{isa}(\mathtt{s1}, \mathtt{student})$. To ensure fairness across all the methods being compared, we used these resulting binary predicates as inputs to all the comparative methods considered here. We also allow inverse relations in random walks, that is, we consider a relation and its inverse to be distinct relations. In the case of one-to-one and one-to-many relations, this sometimes leads to uninteresting random walks of the form $\mathtt{relation} \rightarrow \mathtt{relation}^{-1} \rightarrow \mathtt{relation}$. In order to avoid this situation, we add additional sanity constraints on walks that prevent relations and their inverses from immediately following one another.

For each of our experiments, we subsample training examples at a $2 : 1$ ratio of negatives to positives during training. The number of RRBM hidden nodes are set to 60% of the visible nodes for every data set. Finally, we set the learning rate, $\lambda = 0.05$ and the number of epochs to 5. We compare the approaches using conditional log-likelihood (CLL), area under receiver-operator characteristics (AUC-ROC), and area under precision-recall curve (AUC-PR). Measuring PR performance on skewed relational data sets yields a more conservative view of learning performance [4]. As a result, we use this metric to report statistical significant improvements at $p = 0.05$. We employ 5-fold cross validation across all datasets and use paired t-test on folds.

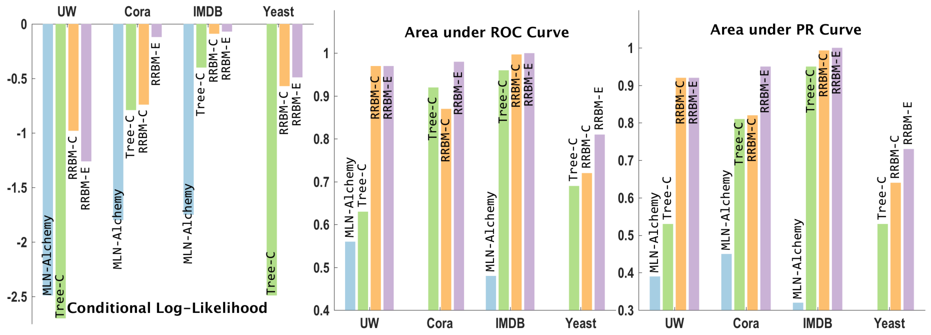


Fig. 3: **(Q1)**: RRBMs outperform baseline MLN and decision-tree (`Tree-C`) models

Datasets

UW-CSE: The UW-CSE data set [30] is a standard benchmark that consists of information from five different areas of computer science about professors, students and courses, and the task is to predict the `advisedBy` relationship between a professor and a student. For MLN, we present generative weight learning as it performed better than discriminative weight learning.

Cora Entity Resolution is a citation matching data set modified by [28]. It contains the predicates such as `Author`, `Title`, `Venue`, `HasWordAuthor`, `HasWordTitle`, `HasWordVenue`, `SameAuthor`, and `SameTitle`. The target predicate is `SameVenue`. Alchemy did not complete the runs after 36 hours and therefore we report their results presented in [16], since we are using the same settings and folds from Khot et al.

IMDB: This data set was first created by [24] and contains nine predicates: `gender`, `genre`, `movie`, `samegender`, `samegenre`, `samemovie`, `sameperson`, `workedunder`, `actor` and `director` where `workedunder` is the target predicate.

Yeast: This publicly-available data set was created by [21] and contains millions of facts from papers on the yeast organism *Saccharomyces cerevisiae*. The data set includes predicates like `gene`, `journal`, `author`, `title`, `chem`, etc. The target predicate is `cites`. As in the original paper, we need to prevent the model from using information (facts) obtained later than the publication date. While calculating features for a citation link, we only considered facts that were earlier than the publication date of the older paper. Since we cannot enforce this constraint in LSM, we do not report MLN/Alchemy results in Figure 3.

Results

Q1: Figure 3 compares our approaches to baseline MLN and decision tree algorithms to help answer **Q1**. `RRBM-E` and `RRBM-C` have significant improvement over `Tree-C` on the UW and Yeast dataset, with comparable performance on Cora and IMDB. Across all the datasets (except Cora) and all the metrics, `RRBM-E` and `RRBM-C` beat the baseline MLN approach. Thus, we can answer **Q1** affirmatively: RRBM models outperform baseline approaches in most cases.

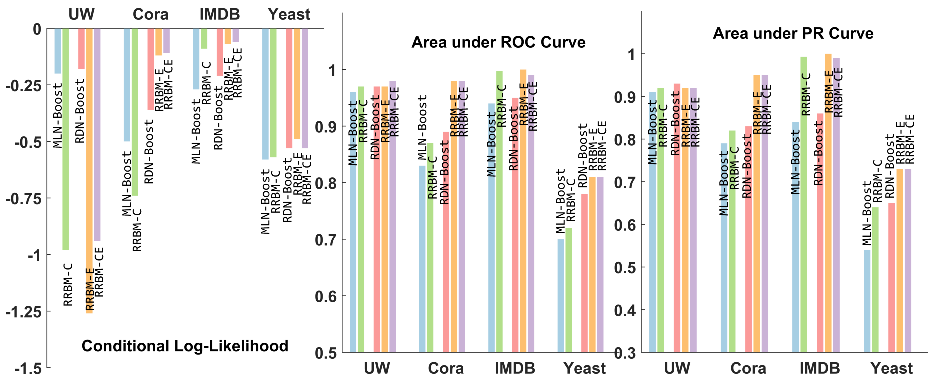


Fig. 4: **(Q2)** Better or comparable performance of `RRBM-C` to MLN-Boost, and `RRBM-E` to RDN-Boost.

Q2: We compare `RRBM-C` to MLN-Boost (count-based models) and `RRBM-E` to RDN-Boost (existential-based models) in Figure 4. Compared to MLN-Boost on CLL, `RRBM-C` has a statistically significant improvement on all datasets except Yeast. `RRBM-E` is comparable to RDN-Boost on all the datasets with statistical significant CLL improvement on Cora. We also see significant AUC-ROC improvement of `RRBM-C` on Cora and `RRBM-E` on IMDB. Thus, we confirm that `RRBM-E` and `RRBM-C` are better or comparable to best structure learning methods.

Q3: Broadly, the results show that RRBM approaches generalize well across different datasets. The results also indicate that `RRBM-EC` generally improves upon `RRBM-C` and has comparable performance to `RRBM-E`. This shows that existential features are sufficient or better at modeling these datasets. This is also seen in the boosting approaches where RDN-Boost, using existential semantics, generally outperforms MLN-Boost which uses count semantics.

Conclusion

We have presented a combination of deep and statistical relational learning that gives rise to a powerful deep architecture for relational classification tasks, called Relational RBMs. In contrast to propositional approaches that use deep learning features as inputs to log-linear models (e.g. [6]), we proposed and explored a paradigm connecting PLM features as inputs to deep learning. The benefits were illustrated on several SRL benchmark datasets, where RRBMs outperformed state-of-the-art structure learning approaches—showing the tight integration of deep learning and statistical relational learning. Our work suggests several interesting avenues for future work. First, one should explore stacking several Relational RBMs. Alternatively, one could explore to jointly learn the underlying relational model and the deep model using stochastic EM. This "deep predicate invention" holds promise to boost relational learning. Ultimately, one should close the loop and feed the deep learning features back into a (relational) log-linear model.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science 9(1), 147–169 (1985)
2. Blockeel, H., Uwents, W.: Using neural networks for relational learning. In: ICML-2004 Workshop on SRL. pp. 23–28 (2004)
3. Das, M., Wu, Y., Khot, T., Kersting, K., Natarajan, S.: Scaling lifted probabilistic inference and learning via graph databases. In: SDM (2016)
4. Davis, J., Goadrich, M.: The relationship between Precision-Recall and ROC curves. In: ICML (2006)
5. De Raedt, L., Kersting, K., Natarajan, S., Poole, D. (eds.): Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Morgan and Claypool Publishers (2016)
6. Deng, L.: Connecting deep learning features to log-linear models. In: Log-Linear Models, Extensions and Applications. MIT Press (2015)
7. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. AISTATS 9, 145–152 (2010)
8. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for AI. Morgan & Claypool (2009)
9. Gehler, P.V., Holub, A.D., Welling, M.: The rate adapting Poisson model for information retrieval and object recognition. In: ICML. pp. 337–344 (2006)
10. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press (2007)
11. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (2002)
12. Hinton, G.E., Osindero, S.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)
13. Hinton, G.E.: To recognize shapes first learn to generate images. In: Computational Neuroscience: Theoretical Insights into Brain Function. Elsevier (2007)
14. Hu, Z., Ma, X., Liu, Z., Hovy, E.H., Xing, E.P.: Harnessing deep neural networks with logic rules. In: ACL (2016)
15. Kazemi, S., Buchman, D., Kersting, K., Natarajan, S., Poole, D.: Relational logistic regression. In: KR (2014)
16. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning Markov logic networks via functional gradient boosting. In: ICDM (2011)
17. Kok, S., Domingos, P.: Learning Markov logic network structure via hypergraph lifting. In: ICML (2009)
18. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML (2010)
19. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press (2009)
20. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: Fast learning of relational kernels. Machine Learning 78, 305–342 (2010)
21. Lao, N., Cohen, W.: Relational retrieval using a combination of path-constrained random walks. Journal of Machine Learning 81(1), 53–67 (2010)
22. Larochelle, H., Bengio, Y.: Classification using discriminative restricted Boltzmann machines. In: ICML. pp. 536–543 (2008)
23. Lecun, Y., Chopra, S., Hadsell, R., Marc'aurelio, R., Huang, F.: A tutorial on energy-based learning. In: Predicting Structured Data. MIT Press (2006)
24. Mihalkova, L., Mooney, R.: Bottom-up learning of Markov logic network structure. In: ICML (2007)

25. Natarajan, S., Khot, T., Kersting, K., Guttmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning 86(1), 25–56 (2012)
26. Natarajan, S., Khot, T., Kersting, K., Shavlik, J. (eds.): Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine. SpringerBriefs in Computer Science, Springer (2016)
27. Natarajan, S., Tadepalli, P., Dietterich, T., Fern, A.: Learning first-order probabilistic models with combining rules. AMAI 54(1-3), 223–256 (2008)
28. Poon, H., Domingos, P.: Joint inference in information extraction. In: AAAI (2007)
29. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc. (1993)
30. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62, 107–136 (2006)
31. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: ICML. pp. 791–798 (2007)
32. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: NIPS (2007)
33. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: ICML. pp. 1064–1071 (2008)
34. Wang, W., Cohen, W.: Learning first-order logic embeddings via matrix factorization. In: IJCAI (2016)