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[6]

[1] Indiana University, Bloomington, USA; [2] The University of Texas at Dallas, USA; [3] Allen Institute of Artificial Intelligence, USA; [4] Technische Universität Darmstadt, Germany; [5] Carnegie Mellon University, USA; [6] The University of Texas at Dallas, USA
[1]navdkaur@indiana.edu, [2]gautam.kunapuli@utdallas.edu, [3]tushar.v.khot@gmail.com, [4]kersting@cs.tu-darmstadt.de, [5]wcohen@gmail.com, [6]sriraam.natarajan@utdallas.edu

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 six relational domains.

Introduction

Restricted Boltzmann machines (RBMs, [30]) are popular models for learning probability distributions due to their expressive power. Consequently, they have been applied to various tasks such as collaborative filtering [39], motion capture [41] and others. Similarly, there has been significant research on the theory of RBMs: approximating log-likelihood gradient by contrastive divergence (CD, [17]), persistent CD [42], parallel tempering [11], 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, [15, 9]) methods use richer symbolic features during learning; however, they have not been fully exploited in deep-learning methods. Learning SRL models is computationally intensive [33] 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, since weak rules cannot always model rich relationships that exist in the domain. For instance, a weak rule could say something like: “a professor is popular if he teaches a course”. When learning discriminatively, this rule could have been true if some professors teach at least one course, while at least one not so popular popular professor did not teach a course in the current data set. We propose to use a set of interpretable rules based on the successful Path Ranking Algorithm (PRA, [28]). Recently, Hu et al. [20] employed logical rules to enhance the representation of neural networks. There has also been work on lifting neural networks to relational settings [4]. While 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. Specifically, we consider random walks constructed by the PRA approach of Lao and Cohen [28] to develop features that can be trained using RBMs. We consider the formalism of discriminative RBMs as our base classifier and use these relational walks with the base classifier.

We propose two approaches to instantiating RBM features: (1) similar to the approach of Markov Logic Networks (MLNs, [12]) and Relational Logistic Regression (RLR, [21]), we instantiate features with *counts* of the number of times a random walk is satisfied for every training example; and (2) similar to Relational Dependency Networks (RDNs, [32]), we instantiate features with *existentials* (1 if $\exists$ at least one instantiation of the path in the data, otherwise 0). Given these features, 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 RLR, and (4) we demonstrate the effectiveness of this novel approach by empirically comparing against state-of-the-art methods that also learn from relational data.

The rest of the paper is organized as follows: Section 2 presents the background on relational random walks and RBMs, Sections 3 and 4 present our RRBM approach and algorithms in detail, and explore its connections to some well-known probabilistic relational models. Section 5 presents the experimental results on standard relational data sets. Finally, the last section concludes the paper by outlining future research directions.

Prior Work and Background

In this article, we represent relational data using standard first-order logic notation. We refer to objects of a particular type as *constants* of that type. Relations in the domain are called *predicates* and the true relations in the data are called *ground atoms*.

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 predicates with shared variables: $\mathtt{takes}(\mathtt{S}, \mathtt{C}) \wedge \mathtt{taughtBy}(\mathtt{C}, \mathtt{P})$.

Relational Probabilistic Models

Markov Logic Networks (MLNs, [12]) 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. Intuitively, a possible world where formula f_i is true one more time than a different possible world is e^{w_i} times as probable, all other things being equal. We focus on discriminative learning, where we learn a conditional distribution of one predicate given all other predicates.

Another such discriminative model is relational logistic regression (RLR, [21]), which extends logistic regression to relational settings, and where training examples can have differing feature sizes. An interesting observation is that RLR can be considered as an aggregator when there are multiple values for the same set of features.

Structure Learning Approaches

Many structure learning approaches for Probabilistic Logical Models (PLMs), including MLNs, use graph representations. For example, Learning via Hypergraph Lifting (LHL, [23]) builds a hypergraph over ground atoms; LHL then clusters the atoms to create a “lifted” hypergraph, and traverses this graph to obtain rules. Specifically, they use depth-first traversal to create the paths in this “lifted” hypergraph to create potential clauses by using the conjunction of predicates from the path as the body of the clause.

Learning with Structural Motifs (LSM, [24]) 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. Our hypothesis space includes the clauses generated by both these approaches without the additional complexity of clustering the nodes.

Propositionalization Approaches

To learn powerful deep models on relational data, propositionalization is used to convert ground atoms into a fixed-length feature vector. For instance, kFoil [27] uses a *dynamic* approach to learn clauses to propositionalize relational examples for SVMs. Each clause is converted into a Boolean feature that is 1, if an example satisfies the clause boyd and each clause is scored based on the improvement of the SVM learned using the clause features. Alternately, the Path Ranking Algorithm (PRA) [28], which has been used to perform knowledge base completion, 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

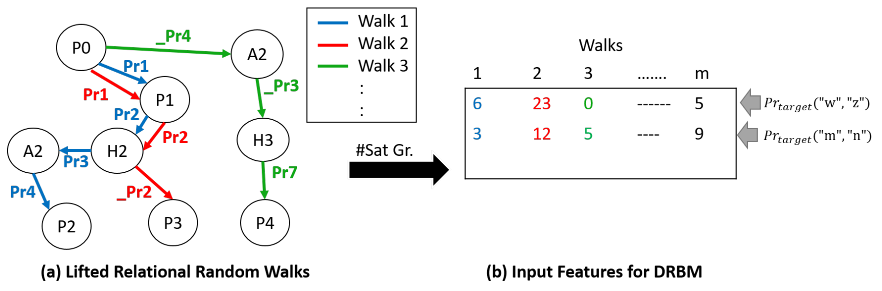
Boltzmann machines (BMs, [30]) model probability distributions and are interpretable as artificial neural networks [1]. A BM consists of visible units V (representing observations) and hidden units H (representing dependencies between features). A general BM is a fully-connected Markov random field [26], 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*. Stacking uses RBMs as building blocks to construct deep belief networks (DBNs) with multiple layers of non-linear transformations of input data; this results in powerful deep belief networks [18].

RBMs have been used as feature extractors for supervised learning [14] and to initialize deep neural networks [19]. Larochelle and Bengio [29] 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 nonlinear classifiers learned by discriminative RBMs.

Relational Restricted Boltzmann Machines

Reconsider MLNs, arguably one of the leading relational approaches unifying logic and probability. The use of relational formulas as features within a log-linear model allows the exploitation of "deep" knowledge. Nevertheless, this is still a shallow architecture as there are no "hierarchical" formulas defined from lower levels. The hierarchical stacking of layers, however, is the essence of deep learning and, as we demonstrate in this work, critical for relational data, even more than for propositional data. This is due to one of the key features of relational modeling: predictions of the model may depend on the number of individuals, that is, the population size. Sometimes this dependence is desirable, and in other cases, model weights may need to change. In either case, it is important to understand how predictions change with population size when modeling or even learning the relational model [21].

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 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



(c) Lifted Random Walk 3 represented as clause: _Pr4(P0,A2) ^ _Pr3(A2,H3) ^ Pr7(H3,P4) => Pr_{target}(P0,P4)

Fig. 1: Lifted random walks are converted into feature vectors by explicitly grounding every random walk for every training example. Nodes and edges of the graph in (a) represent types and predicates, and underscore (_Pr) represents the inverted predicates. The random walks counts (b) are then used as feature values for learning a discriminative RBM (DRBM). An example of random walk represented as clause is (c).

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 are given data, $D = \{(\mathbf{x}_i, \hat{y}_i)_{i=1}^{\ell}\}$, where each training example is a vector, $\mathbf{x}_i \in \mathbb{R}^m$ with a multi-class label, $\hat{y}_i \in \{1, \dots, C\}$. 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, for which 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 representation

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 by a directed edge from the node $\mathtt{t}_1$ to $\mathtt{t}_2$ in the graph. 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 n-ary predicate. 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 $\mathtt{isa}(\mathtt{t}, \mathtt{r})$.

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

Step 2: Relational transformation layer

Now, we generate the input feature vector $\mathbf{x}_i$ from a relational example, $\mathtt{T}(\mathtt{a}_{1j}, \mathtt{a}_{2j})$. Inspired by the Path Ranking Algorithm [28], 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 $\mathbf{x}$. Specifically, starting from the node for the first argument's type, we repeatedly perform random walks till we reach the node for the second argument. Since random walks also correspond to the set of candidate clauses considered by structure-learning approaches for MLNs [23, 24], 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 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 $\mathtt{professor} - \mathtt{student}$ combination, not all $\mathtt{professor} - \mathtt{student}$ combinations will have the same number of feature values. This is commonly referred as *multiple-parent* problem [34]. To alleviate this problem, SRL methods consider one of two approaches – aggregators or combining rules. Aggregators combine multiple values to a single value, while combining rules combine multiple probability distributions into one. While these solutions are reasonable for traditional probabilistic models that estimate distributions, they are not computationally feasible for the current task.

Our approach to the multiple-parent problem is to consider *existential semantics*: if there exists *at least one instance of the random walk that is satisfied for an example*, the feature value corresponding to that random walk is set to 1 (otherwise, to 0). This approach was also recently (and independently of our work) used by Wang and Cohen [43] for ranking via matrix factorization. This leads to our first model: RRBM-Existentials, or RRBM-E, where E denotes the existential semantics used to construct the RRBM. One limitation of RRBM-E is that it does not differentiate between a $\mathtt{professor} - \mathtt{student}$ combination that has only one paper and another that has 10 papers, that is, it does not take into account how often a relationship is true in the data. Inspired by MLNs, we also consider *counts of the random walks* as feature values, a model we denote RRBM-Counts or RRBM-C (Figure 1). For example, if a $\mathtt{professor} - \mathtt{student}$ combination has written 10 papers, the feature value corresponding to this random walk for that combination is 10. To summarize, we define two transformation functions, $\mathbf{x}_j = g(\mathtt{a}_{1j}, \mathtt{a}_{2j})$

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

For example, consider that the walk $\mathtt{takes}(\mathtt{S}, \mathtt{C}) \wedge \mathtt{taughtBy}(\mathtt{C}, \mathtt{P})$ is used to generate a feature for $\mathtt{advisedBy}(\mathtt{s}_1, \mathtt{p}_1)$. The feature from g_c would be set to the count: $|\{\mathtt{C} \mid \mathtt{takes}(\mathtt{s}_1, \mathtt{C}) \wedge \mathtt{taughtBy}(\mathtt{C}, \mathtt{p}_1)\}|$. With the function, g_e, this feature would be set to 1, if $\exists \mathtt{C}, \mathtt{takes}(\mathtt{s}_1, \mathtt{C}) \wedge \mathtt{taughtBy}(\mathtt{C}, \mathtt{p}_1)$.

These transformation functions also allow us to relate our approach to other well-known relational models. For instance, g_c uses counts similar to MLNs, while g_e uses existential semantics similar to RDNs [32]. 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 [7] that leverages the power of graph databases. Our empirical evaluation did not require count approximations; we defer integration of approximate counting to future research.

Step 3: Learning Relational RBMs

The output of the relational transformation layer is fed into multilayered discriminative RBM (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 are able to learn a much more expressive model than traditional MLNs and RLRs. Recall that the DRBM as defined by [29] 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 $\Theta \equiv (W, \mathbf{b}, \mathbf{c}, \mathbf{d}, U)$:

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

As with most generative models, computing the joint probability $p(\mathbf{y}, \mathbf{x})$ is intractable, but the conditional distribution $P(\hat{y}|\mathbf{x})$ can be computed exactly [39] 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: $2 \max(\texttt{count}(x_i^j))$ for every feature; bounds are the same for both train and test sets to avoid overfitting. In other words, the bound is simply twice the max feature count over all the examples of the training set. We can choose the scaling factor through cross-validation, but value 2 seems to be a reasonable scale in our experiments. For the test examples, we can use the random walks to generate the features and the RBM layers to generate predictions from these features.

Algorithm 1 LearnRRBM($\mathtt{T}, \mathtt{G}, \mathtt{P}$): Relational Restricted Boltzmann Machines
Input $\mathtt{T}(\mathtt{t_1}, \mathtt{t_2})$: 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 $0 \leq j < l$ do   ▷ Iterate over all training examples
 4:     ▷ Generate features for $\mathtt{T}(\mathtt{a_{1j}}, \mathtt{a_{2j}})$
 5:     for $0 \leq p < m$ 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($\Theta$, $\mathbf{rw}$)
```

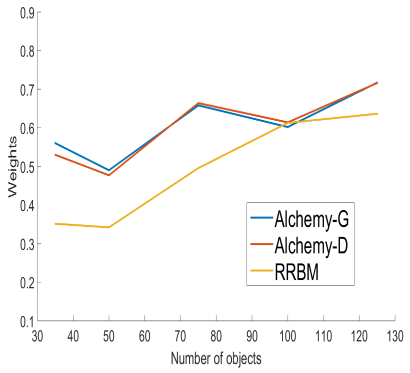
RRBM Algorithm: The complete approach to learn Relational RRBMs is shown in Algorithm 1. In Step 1, we generate type-restricted random walks using PRA. These random walks ($\mathbf{rw}$) are used to construct the feature matrix. For each example, we obtain exact counts for each random walk, which becomes the corresponding feature value for that example. A DRBM can be trained on the features as explained in Step 3.

Relation to Probabilistic Relational Models

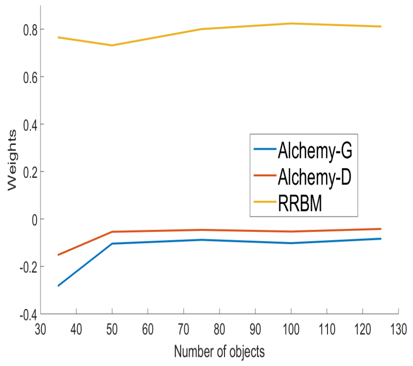
The random walks can be interpreted as logical clauses (that are used to generate features) and the DRBM input feature weights $\mathbf{b}$ in (1) can be interpreted as clause weights (w_p). This interpretation highlights connections between our approach and Markov logic networks. Intuitively, the relational 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 data set for a university domain with varying number of objects (professors and students). We picked a subset of $\mathtt{professor} - \mathtt{student}$ pairs to have an `advisedBy` relationship and add common papers or common courses based on the following two clauses:

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

The first clause states that if a professor A co-authors a paper P with the student B, then A advises B. The second states that if a student B registers for a course C taught by professor A then A advises 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



(a) Co-author clause



(b) Course clause

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

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 (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 discover new concepts. While MLNs can learn arbitrary N-ary target predicates, due to the definition of random walks in the original work, we are restricted to learning binary relations.

Experiments

To compare RRBM approaches to state-of-the-art algorithms, we consider `RRBM-E`, `RRBM-C` and `RRBM-CE`. The last approach, `RRBM-CE` 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 baseline MLNs and Decision Trees?
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-CE` generalize across all domains?
Q4: How do random-walk generated features compare to propositionalization?

To answer **Q1**, we compare RRBMs to Learning with Structural Motifs (LSM, [24]). Specifically, we perform structure learning with LSM followed by weight learning with Alchemy [25] and denote this as MLN. We would also like to answer the question: how crucial is it to use a RBM, and not some other ML algorithm? We use decision trees [36] 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 features. We denote the decision tree model `Tree-C`. For LSM, we used the parameters recommended by [24]. However, we set the maximum path length of random walks of LSM structure learning to 6 to be consistent with the maximum path length used in RRBM. We used both discriminative and generative weight-learning for Alchemy and present the best-performing result.

To answer **Q2**, we compare `RRBM-C` to MLN-Boost [22], and `RRBM-E` to RDN-Boost [32] 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 [28] are performed on binary predicates, we convert unary and ternary predicates into binary predicates. For example, predicates such as $\texttt{teach}(\texttt{a1}, \texttt{a2}, \texttt{a3})$ are converted to three binary predicates: $\texttt{teachArg1}(\texttt{id}, \texttt{a1})$, $\texttt{teachArg2}(\texttt{id}, \texttt{a2})$, $\texttt{teachArg3}(\texttt{id}, \texttt{a3})$ where `id` is the unique identifier for a predicate. As another example, unary predicates such as $\texttt{student}(\texttt{s1})$ are converted to binary predicates of the form $\texttt{isa}(\texttt{s1}, \texttt{student})$. To ensure fairness, we used binary predicates as inputs to all the methods considered here. We also allow inverse relations in random walks, that is, we consider a relation and its inverse to be distinct relations. For one-to-one and one-to-many relations, this sometimes leads to uninteresting random walks of the form $\texttt{relation} \rightarrow \texttt{relation}^{-1} \rightarrow \texttt{relation}$. In order to avoid this situation, we add additional sanity constraints on walks that prevent relations and their inverses from immediately following one another and avoid loops.

To answer **Q4**, we compare our method with Bottom Clause Propositionalization [13] (`BCP-RBM`), which generates one bottom clause for each example and considers each atom in the body of the bottom clause to be a unique feature. We utilize Progol [38] to generate bottom clauses by using its default configuartion but setting variable depth = 1 to handle large data sets. Contrary to the original work [13] that uses a neural network, we use RBM as the learning model, as our goal is to demonstrate the usefulness of random walks to generate features.

In our experiments, we subsample training examples at a $2:1$ ratio of negatives to positives during training. The number of RBM hidden nodes are set to 60% of visible nodes, the learning rate, $\eta = 0.05$ and the number of epochs to 5. These hyperparameters have been optimized by line search.

A Note On Hyperparameter Selection: An important hyperparameter for RRBMs is the maximum path length of random walks, which influences the number of RRBM features. Fig. 3 shows that the number of features generated grows exponentially with maximum path length. We restricted the maximum path length of random walks to $\lambda = 6$ in order to strike a balance between tractability and performance; $\lambda = 6$ demonstrated consistently good performance across a variety of data sets, while keeping the feature

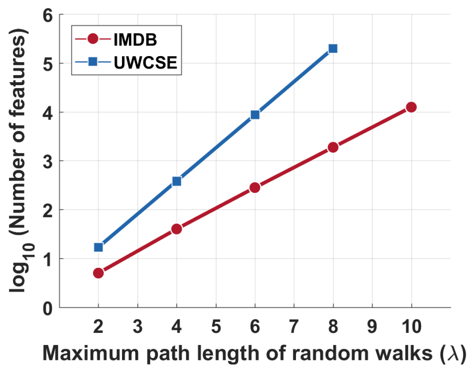


Fig. 3: The number of RRBM features grows exponentially with maximum path length of random walks. We set $\lambda = 6$ to balance tractability with performance.

size tractable. As mentioned above, other benchmark methods such as LSM were also restricted to a maximum random walk length of 6 for consistency and fairness.

Hyperparameter selection is an open issue in both relational learning as well as deep learning; in the latter, careful tuning of hyperparameters and architectures such as regularization constants and number of layers is critical. Recent work on automated hyperparameter selection can also be used with RRBMs, if a more systematic approach to hyperparameter selection for RRBMs is desired, especially in practical settings. Bergstra and Bengio [2] demonstrated that random search is more efficient for hyperparameter optimization than grid search or manual tuning. This approach can be used to select optimal η and λ jointly. Snoek et al [40] recently used Bayesian optimization for automated hyperparameter tuning. While this approach was shown to be highly effective across diverse machine learning formalisms including for support vector machines [6], latent Dirichlet allocation [3] and convolutional neural networks [16], it requires powerful computational capabilities and parallel processing to be feasible in practical settings.

Data Sets

We used several benchmark data sets to evaluate the performance of our algorithms. We compare several approaches using conditional log-likelihood (CLL), area under ROC curve (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 [8]. As a result, we use this metric to report statistical significant improvements at $p = 0.05$. We employ 5-fold cross validation across all data sets.

UW-CSE: The UW-CSE data set [37] is a standard benchmark that consists of predicates and relations such as `professor`, `student`, `publication`, `hasPosition` and `taughtBy` etc. The data set contains 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 MLNs, we present results from generative weight learning as it performed better than discriminative weight learning.

Mutagenesis: The MUTAGENESIS data set[2] has two entities: `atom` and `molecule`, and consists of predicates that describe attributes of atoms and molecules, as well as the types of relationships that exist between atom and molecule. The target predicate is $\texttt{moleatm}(\texttt{aid}, \texttt{mid})$, to predict whether a molecule contains a particular atom. For MLN, we present generative weight learning as it had better results.

Cora Entity Resolution is a citation matching data set [35]; in the citation-matching problem, a "group" is a set of citations that refer to the same publication. Here, a large fraction of publications belong to non-trivial groups, that is, groups that have more than one citation; the largest group contains as many as 54 citations, which makes this a challenging problem. It contains the predicates such as `Author`, `Title`, `Venue`, `HasWordTitle`, `SameAuthor` and the target predicate is `SameVenue`. Alchemy did not complete running after 36 hours and therefore we report results from [22].

IMDB: This data set was first created by Mihalkova and Mooney [31] and contains nine predicates: `gender`, `genre`, `movie`, `samegender`, `samegenre`, `samemovie`, `sameperson`, `workedunder`, `actor` and `director`; we predict the `workedUnder` relation. Since `actor` and `director` are unary predicates, we converted them to one binary predicate $\texttt{isa}(\texttt{person}, \texttt{designation})$ where designation can take two values - `actor` and `director`. For MLNs, we report the generative weight learning results here.

Yeast: contains millions of facts [28] from papers published between 1950 and 2012 on the yeast organism *Saccharomyces cerevisiae*. It includes predicates like `gene`, `journal`, `author`, `title`, `chem`, etc. The target predicate is `cites`, that is, we predict the citation link between papers. As in the original paper, we need to prevent models from using information obtained later than the publication date. While calculating features for a citation link, we only considered facts that were earlier than a publication date. Since we cannot enforce this constraint in LSM, we do not report Alchemy results for Yeast.

Sports: NELL [5] is an online[3] never-ending learning system that extracts information from online text data, and converts this into a probabilistic knowledge base. We consider NELL data from the `sports` domain consisting of information about players and teams. The task is to predict whether a team plays a particular sport or not. Alchemy did not complete its run after 36 hours, thus we do not report its result for this data set.

Results

Q1: Fig. 4 compares our approaches to baseline MLNs and decision trees to answer **Q1**. `RRBM-E` and `RRBM-C` have significant improvement over `Tree-C` on UW and Yeast data sets, with comparable performance on the other four. Across all data sets (except Cora) and all 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.

[2] `cs.sfu.ca/∼oschulte/BayesBase/input`
[3] `rtw.ml.cmu.edu/rtw/`

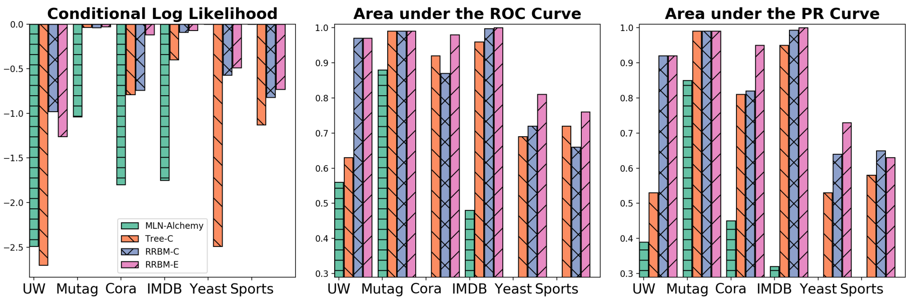


Fig. 4: **(Q1)**: Results show that RRBMs generally outperform baseline MLN and decision-tree (`Tree-C`) models.

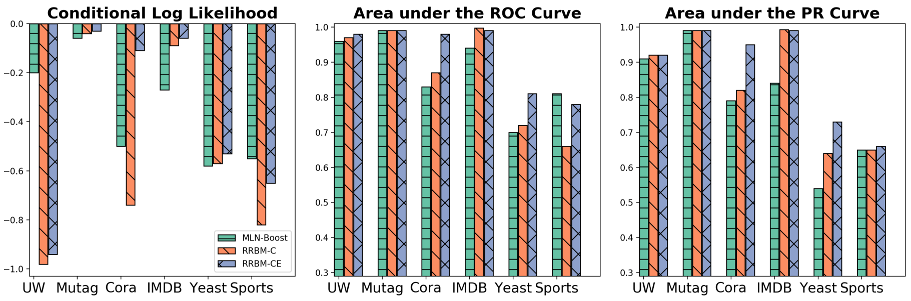


Fig. 5: **(Q2)** Results show better or comparable performance of `RRBM-C` and `RRBM-CE` to MLN-Boost, which all use counts.

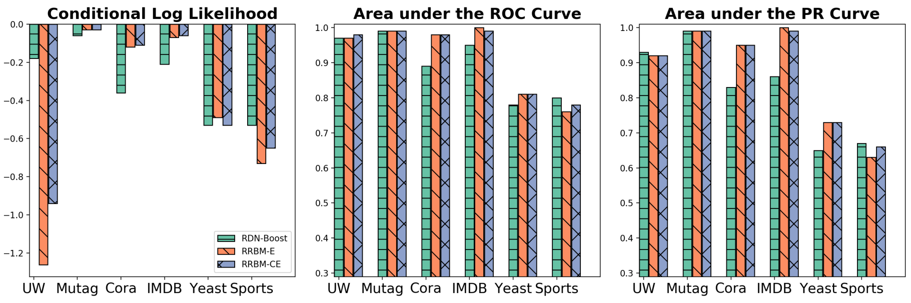


Fig. 6: **(Q2)** Results show better or comparable performance of `RRBM-E` and `RRBM-CE` to RDN-Boost, which all use existentials.

Q2: We compare `RRBM-C` to MLN-Boost (count-based models) and `RRBM-E` to RDN-Boost (existential-based models) in Figs. 5 and 6. Compared to MLN-Boost on CLL, `RRBM-C` has a statistically significant improvement or is comparable on all data sets. `RRBM-E` is comparable to RDN-Boost on all the data sets 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

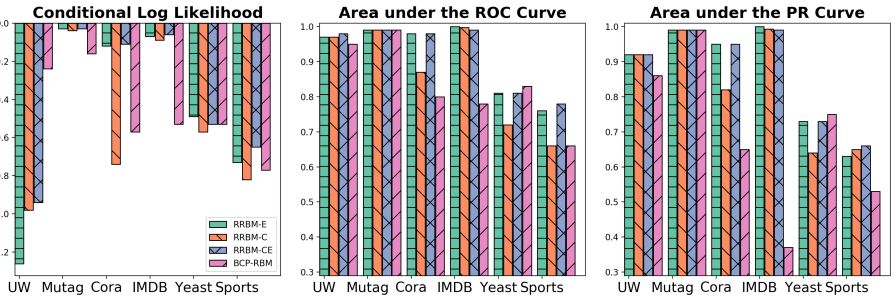


Fig. 7: **(Q4)** Results show better or comparable performance of our random-walk-based feature generation approach (RRBM) compared to propositionalization (BCP-RBM).

or comparable to the current best structure learning methods.

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

Q4: Since BCP-RBM only generates existential features, we compare BCP-RBM with RRBM-E to answer **Q4**. Fig. 7 shows that RRBM-E has statistically significantly better performance than BCP-RBM on three data sets on CLL. Further, RRBM-E demonstrates significantly better performance than BCP-RBM on four data sets: Cora, Mutagenesis, IMDB and Sports - both on AUC-ROC and AUC-PR. This allows us to state positively that random-walk features yield better or comparable performance than propositionalization. For IMDB, BCP-RBM generated identical bottom clauses for all positive examples, resulting in an extreme case of just a single positive example to be fed into RBM. This results in a huge skew (distinctly observable in AUC-PR of IMDB for BCP-RBM).

Conclusion

Relational data and knowledge bases are useful in many tasks, but feeding them to deep learners is a challenge. To address this problem, we have presented a combination of deep and statistical relational learning, which 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. [10]), we proposed and explored a paradigm connecting PLM features as inputs to deep learning. While statistical relational models depend much more on the discriminative quality of the clauses that are fed as input, Relational RBMs can learn useful hierarchical relational features through its hidden layer and model non-linear decision boundaries. The benefits were illustrated on several SRL benchmark data sets, 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 RRBMs. Since the relational feature extraction is separated from the deep learning method, different types of deep learning methods can easily be used and should be explored. 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.

Acknowledgements

Kristian Kersting gratefully acknowledges the support by the DFG Collaborative Research Center SFB 876 projects A6 and B4. Sriraam Natarajan gratefully acknowledges the support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, ARO, AFRL, or the US government.

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. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. JMLR 13 (2012)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. JMLR 3 (2003)
4. Blockeel, H., Uwents, W.: Using neural networks for relational learning. In: ICML-2004 Workshop on SRL. pp. 23–28 (2004)
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI. pp. 1306–1313 (2010)
6. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods. Cambridge University Press, New York, USA (2000)
7. Das, M., Wu, Y., Khot, T., Kersting, K., Natarajan, S.: Scaling lifted probabilistic inference and learning via graph databases. In: SDM (2016)
8. Davis, J., Goadrich, M.: The relationship between Precision-Recall and ROC curves. In: ICML (2006)
9. De Raedt, L., Kersting, K., Natarajan, S., Poole, D. (eds.): Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Morgan and Claypool Publishers (2016)
10. Deng, L.: Connecting deep learning features to log-linear models. In: Log-Linear Models, Extensions and Applications. MIT Press (2015)
11. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. AISTATS 9, 145–152 (2010)
12. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for AI. Morgan & Claypool Publishers (2009)
13. França, M.V.M., Zaverucha, G., d’Avila Garcez, A.S.: Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning 94 (2014)
14. 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)
15. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press (2007)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

17. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14, 1771–1800 (2002)
18. Hinton, G.E., Osindero, S.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)
19. Hinton, G.E.: To recognize shapes first learn to generate images. In: Computational Neuroscience: Theoretical Insights into Brain Function. Elsevier (2007)
20. Hu, Z., Ma, X., Liu, Z., Hovy, E.H., Xing, E.P.: Harnessing deep neural networks with logic rules. In: ACL (2016)
21. Kazemi, S., Buchman, D., Kersting, K., Natarajan, S., Poole, D.: Relational logistic regression. In: KR (2014)
22. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning Markov logic networks via functional gradient boosting. In: ICDM (2011)
23. Kok, S., Domingos, P.: Learning Markov logic network structure via hypergraph lifting. In: ICML (2009)
24. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML (2010)
25. Kok, S., Sumner, M., Richardson, M., et al.: The Alchemy system for statistical relational AI. Tech. rep., University of Washington (2010)
26. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press (2009)
27. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: Fast learning of relational kernels. Machine Learning 78, 305–342 (2010)
28. Lao, N., Cohen, W.: Relational retrieval using a combination of path-constrained random walks. Journal of Machine Learning 81(1) (2010)
29. Larochelle, H., Bengio, Y.: Classification using discriminative restricted Boltzmann machines. In: ICML. pp. 536–543 (2008)
30. Lecun, Y., Chopra, S., Hadsell, R., Marc'aurelio, R., Huang, F.: A tutorial on energy-based learning. In: Predicting Structured Data. MIT Press (2006)
31. Mihalkova, L., Mooney, R.: Bottom-up learning of Markov logic network structure. In: ICML (2007)
32. Natarajan, S., Khot, T., Kersting, K., Guttmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. MLJ 86(1) (2012)
33. Natarajan, S., Khot, T., Kersting, K., Shavlik, J. (eds.): Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine. SpringerBriefs in CS, Springer (2016)
34. Natarajan, S., Tadepalli, P., Dietterich, T., Fern, A.: Learning first-order probabilistic models with combining rules. AMAI 54(1-3), 223–256 (2008)
35. Poon, H., Domingos, P.: Joint inference in information extraction. In: AAAI (2007)
36. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc. (1993)
37. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62 (2006)
38. S., M.: Inverse entailment and progol. New Generation Computing 13 (1995)
39. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: ICML. pp. 791–798 (2007)
40. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS. pp. 2951–2959 (2012)
41. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: NIPS (2007)
42. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: ICML. pp. 1064–1071 (2008)
43. Wang, W., Cohen, W.: Learning first-order logic embeddings via matrix factorization. In: IJCAI (2016)