Learning First-Order Logic Embeddings via Matrix Factorization

William Yang Wang and **William W. Cohen**
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.
{yww,wcohen}@cs.cmu.edu

Abstract

Many complex reasoning tasks in Artificial Intelligence (including relation extraction, knowledge base completion, and information integration) can be formulated as inference problems using a probabilistic first-order logic. However, due to the discrete nature of logical facts and predicates, it is challenging to generalize symbolic representations and represent first-order logic formulas in probabilistic relational models. In this work, we take a rather radical approach: we aim at learning continuous low-dimensional embeddings for first-order logic from scratch. In particular, we first consider a structural gradient based structure learning approach to generate plausible inference formulas from facts; then, we build grounded proof graphs using background facts, training examples, and these inference formulas. To learn embeddings for formulas, we map the training examples into the rows of a binary matrix, and inference formulas into the columns. Using a scalable matrix factorization approach, we then learn the latent continuous representations of examples and logical formulas via a low-rank approximation method. In experiments, we demonstrate the effectiveness of reasoning with first-order logic embeddings by comparing with several state-of-the-art baselines on two datasets in the task of knowledge base completion.

1 Introduction

Learning to reason and understand the world's knowledge is a fundamental problem in Artificial Intelligence (AI). Traditional symbolic AI methods were popular in the 1980s, when first-order logic rules were mostly handwritten, and reasoning algorithms were built on top of them. In the 90s, more and more researchers were interested in statistical methods that deal with the uncertainty of the data, using probabilistic models.

In Statistical Relational Learning, many researchers ask a vital question: is it possible to integrate both the symbolic and statistical approaches for building intelligent systems? For example, in many probabilistic logic programming frameworks, from Stochastic Logic Programs [Muggleton, 2000] to Markov Logic Networks (MLNs) [Richardson and Domingos, 2006] to ProbLog [Gutmann *et al.*, 2008], one can determine the importance of each inference formula by learning a weight parameter from data. However, a logical formula is made of many important components: quantifiers, predicates, connectives, negations, the number of arguments, goals, variable order and positions etc. Therefore, a key research question that we ask in this paper is: what would be an expressive and learnable representation for logical inference formulas that can deal with uncertainty when working with large, noisy datasets in the real world?

In recent years, learning low-dimensional embeddings becomes very successful in representing language, entities, and relations. For example, Mikolov et al. [2013] have developed Word2Vec: a tool that allows one to learn the multidimensional embeddings of words and phrases from large corpora. When working with relational data, Bordes et al. [2013] find that learning continuous embedded vectors for relations together with head and tail entities improves knowledge base completion [Bordes *et al.*, 2011]. In addition to effectiveness, their approaches are also efficient enough to deal with very large knowledge bases.

In this work, we explore the possibility of learning low-dimensional first-order logic embeddings from scratch. More specifically, we are interested in learning latent, distributional representations of Horn clauses to enhance logic-based knowledge base completion for large datasets. To start with, we first use a structural gradient approach [Wang *et al.*, 2014a] to identify a set of plausible formulas from knowledge bases. Then, we utilize a scalable probabilistic logic called ProPPR [Wang *et al.*, 2013], and produce a grounded proof graph using these logical formulas, training examples, and the background knowledge base. In ProPPR's proof graph, the activated first-order logic formulas become the edges, and the nodes are intermediate states of the proof. To perform representation learning on the formulas, we map the proof graphs to a binary matrix, and use a scalable low-rank approximation based framework to learn the embeddings of the examples and formulas. Finally, we transform these learned embeddings to the parameters for the formulas, and enable first-order logic inference with learned formula embeddings. In experiments, we evaluate the proposed method on a Freebase 15K dataset and a WordNet dataset with hundreds of thousands of facts, and demonstrate the advantage of reason-

```
about(X,Z) :- handLabeled(X,Z)          # base.
about(X,Z) :- sim(X,Y),about(Y,Z)       # prop.
sim(X,Y) :- links(X,Y)                  # sim,link.
sim(X,Y) :-
   hasWord(X,W),hasWord(Y,W),
   linkedBy(X,Y,W)                      # sim,word.
linkedBy(X,Y,W) :- true                 # by(W).
```

Table 1: A simple ProPPR program. See text for explanation.

ing with first-order logic embeddings. Our main contributions are two-fold:

- We propose a practical algorithm for learning first-order logic embeddings for enhancing Statistical Relational Learning;
- We demonstrate that the proposed approach can be scaled to reasoning tasks on two large knowledge bases, outperforming various strong baselines.

2 Related Work

Our work is closely related to recent studies of learning knowledge graph embeddings. RESCAL [Nickel *et al.*, 2011] is among the first to consider tensor decomposition for the task of learning entity and relation embeddings. They use a rank-3 tensor to represent relations, head, and tail entities, and deploy a least squares training method. A number of energy based approaches for learning relational embeddings have also been proposed [Bordes *et al.*, 2011; 2014; Jenatton *et al.*, 2012], showing strong performances. Recently, Bordes et al. [2013] introduce a scalable method for translating embeddings in knowledge base completion, and together with its variants [Wang *et al.*, 2014b; Lin *et al.*, 2015b], they achieve the state-of-the-art results in this task. With the revival of neural network methods, Shi and Zhu [2015] have investigated a deep Convolutional Neural Networks approach for learning relational embeddings. Although these approaches have considered learning embeddings for entities and relations, none of the above have actually investigated the possibility of learning first-order logic embeddings.

There has been recent studies [Lin *et al.*, 2015a; Wang *et al.*, 2015] utilizing logic-like inference paths as constraints for learning entity and relation embeddings. Guu et al. [2015] leverage embedding learning techniques and path constraints for path query answering. Wei et al. [2015] learn knowledge base embeddings, and then use MLNs as a postpass for ranking the instances. Our work also aligns with a preliminary attempt of using entailment based first-order constraints [Rocktäschel *et al.*, 2014] for universal schema based knowledge base completion [Riedel *et al.*, 2013]. To the best of our knowledge, we are the first formal study to investigate the problem of learning low-dimensional first-order logic embeddings from scratch, while scaling formula embeddings based probabilistic logic reasoning to large knowledge bases.

3 Background: ProPPR

Before introducing our first-order logic embedding approach, we briefly review the scalable probabilistic inference framework that our method is built on. Below we will give an informal description of ProPPR, based on a small example. More formal descriptions can be found in [Wang *et al.*, 2013].

ProPPR (for Programming with Personalized PageRank) is a stochastic extension of the logic programming language Prolog. A simple program in ProPPR is shown in Table 1. Roughly speaking, the upper-case tokens are variables, and the ":-" symbol means that the left-hand side (the *head* of a rule) is implied by the conjunction of conditions on the right-hand size (the *body*). In addition to the rules shown, a ProPPR program would include a *database* of *facts*: in this example, facts would take the form *handLabeled(page,label)*, *hasWord(page,word)*, or *linkedBy(page1,page2)*, representing labeled training data, a document-term matrix, and hyperlinks, respectively. The condition "true" in the last rule is "syntactic sugar" for an empty body.

In ProPPR, a user issues a query, such as "about(a,X)?", and the answer is a set of possible bindings for the free variables in the query (here there is just one such varable, "X"). To answer the query, ProPPR builds a *proof graph*. Each node in the graph is a list of conditions $R_1, \ldots, R_k$ that remain to prove, interpreted as a conjunction. To find the children of a node $R1, \ldots, Rk$, you look for either

1. database facts that match R_1, in which case the appropriate variables are bound, and R_1 is removed from the list, or;
2. a rule $A \leftarrow B_1, \ldots, B_m$ with a head A that matches R_1, in which case again the appropriate variables are bound, and R_1 is replaced with the body of the rule, resulting in the new list $B_1, \ldots, B_m, R_2, \ldots, R_k$.

In Prolog, this proof graph is constructed on-the-fly in a depth-first, left-to-right way, returning the first solution found, and backtracking, if requested, to find additional solutions. In ProPPR, however, we will define a *stochastic process on the graph*, which will generate a score for each node, and hence a score for each answer to the query. The stochastic process used in ProPPR is *personalized PageRank* [Page *et al.*, 1998; Csalogny *et al.*, 2005], also known as random-walk-with-restart. Intuitively, this process upweights solution nodes that are reachable by *many short proofs* (i.e., short paths from the query node.) Formally, personalized PageRank is the fixed point of the iteration

$$\mathbf{p}^{t+1} = \alpha \chi_{v_0} + (1-\alpha) W \mathbf{p}^t \tag{1}$$

where $\mathbf{p}[u]$ is the weight assigned to u, v_0 is the seed (i.e., query) node, χ_{v_0} is a vector with $\chi_{v_0}[v_0] = 1$ and $\chi_{v_0}[u] = 0$ for $u \neq v$, and the parameter α is the reset probability. W is a matrix of transition probabilities, i.e., $W[v,u]$ is the probability of transitioning from node u to a child node v:

$$W[v,u] = \frac{1}{Z} f\big(\theta \cdot \phi_{[v,u]}\big) \tag{2}$$

Here Z is an appropriate normalizing constant, θ is the weight vector associated with the features $\phi_{[v,u]}$ on edge $[v,u]$. The edge strength functions f used in this study are rectified linear united (ReLU) [Nair and Hinton, 2010] and the hyperbolic tangent function (tanh) [Glorot and Bengio, 2010].

Like Prolog, ProPPR's proof graph is also constructed on-the-fly, but rather than using depth-first search, we use PageRank-Nibble, a fast approximate technique for incrementally exploring a large graph from a an initial "seed" node [Andersen *et al.*, 2008]. PageRank-Nibble takes a parameter ϵ and will return an approximation $\hat{\mathbf{p}}$ to the personalized PageRank vector $\mathbf{p}$, such that each node's estimated probability is within ϵ of correct. ProPPR can be viewed as a scalable extension of stochastic logic programs[Muggleton, 1996; Cussens, 2001; Van Daele *et al.*, 2014]. We close this background section with some final brief comments about ProPPR.

Scalability. ProPPR is currently limited in that it uses memory to store the fact databases, and the proof graphs constructed from them. ProPPR uses a special-purpose scheme based on sparse matrix representations to store facts which are triples, which allows it to accommodate databases with hundreds of millions of facts in tens of gigabytes.

With respect to run-time, ProPPR'S scalability is improved by the fast approximate inference scheme used, which is typically an order of magnitude faster than power iteration for moderate-sized problems [Wang *et al.*, 2013], and much faster on larger problems. Experimentation and learning are also sped up because with PageRank-Nibble, each query is answered using a "small"—size $O(\frac{1}{\alpha\epsilon})$—proof graph. Many operations required in learning and experimentation can thus be easily parallelized on a multi-core machine, by simply distributing different proof graphs to different threads.

Parameter learning. The personalized PageRank scores are defined by a transition probability matrix W. ProPPR allows "feature generators" to be attached to its rules, as indicated by the code after the hashtags in the example program: for instance, when matching the rule "sim(X,Y) :- links(X,Y)" to a condition such as "sim(a,X)" the two features "sim" and "link" are generated, and when matching the rule "linkedBy(X,Y,W) :- true" to the condition "linkedBy(a,c,sprinter)" the feature "by(sprinter)" is generated. Since edges in the proof graph correspond to rule matches, the edges can also be labeled by features, and a weighted combination of these features can be used to define a total weight for each edge, which finally can be normalized used to define the transition matrix W. Learning can be used to tune these weights to data; ProPPR's learning uses a parallelized SGD method, in which inference on different examples is performed in different threads, and weight updates are synchronized.

Structure learning. Prior work [Wang *et al.*, 2014a] has studied the problem of learning a ProPPR theory, rather than simply tuning parameters in an existing theory, a process called *structure learning*. In particular, inspired by recent advances in inductive logic programming [Muggleton *et al.*, 2014], Wang *et al.* [2014a] propose a scheme called the *structural gradient* which scores every rule in some (possibly large) user-defined space $\mathcal{R}$ of potential rules, and then adds high-scoring rules to a theory. In more detail, the space of potential rules $\mathcal{R}$ is defined by a "second-order abductive theory", which conceptually constructs proofs using all rules in $\mathcal{R}$. The second-order theory is defined in such a way such that each parameter in the second-order theory corresponds to a rule in $\mathcal{R}$, so the gradient of the parameter vector corresponds to a scoring scheme for the rules in $\mathcal{R}$. More specifically, we use three generic structural templates ("*if* entailment", "*inverse* entailment", and "*chain* entailment"), then each parameter of this second-order theory will correspond to a candidate first-order formula. Finally, a set of plausible formula candidates are selected based on the gradient ranks of each formula. The structure learning via parameter learning idea of ProPPR's structure learning method is broadly related to joint structure and parameter learning of Markov Logic Networks [Khot *et al.*, 2011]. The iterated structural gradient method that incrementally refines the hypothesized structure space is also closely related to a learn-and-join algorithm for learning Markov Logic Networks [Khosravi *et al.*, 2010].

4 Our Approach

We now describe the technical details of our approach. First, we explain how we formulate the representation learning task. Then, we introduce a scalable approach for learning low-dimensional embeddings of first-order logic formulas.

4.1 Problem Formulation

In this work, we propose a novel matrix factorization approach to learning first-order logic embeddings. Figure 1 shows an overview of the framework.

More specifically, we first use ProPPR's structural gradient method [Wang *et al.*, 2014a] to compute a set of candidate formulas from knowledge bases. We then use this set of formulas, background knowledge base, and training examples to produce standard ProPPR proof graphs (see the left part of Figure 1 for an example). In ProPPR's grounding mechanism, we start from a query node (training example), and incrementally apply each first-order logic formula to an edge as the proof process proceeds. The process will stop once it has reached a solution node, and now those activated formulas will be on the edges of this proof graph and the nodes will become the intermediate states of the proof.

We formulate this first-order logic formula embedding learning task as a matrix completion problem. Given a collection of proof graphs, we assume that there are m total examples, which are the rows in the our matrix. As for the columns, each column represents a first-order logic formula. The total number of columns in the input matrix is n. Our matrix F now encodes training examples, and the activated first-order logic formulas for each example. Here we use i to index the i-th example and j to index the j-th formula. The formulas take only binary values—either they are used in the proof graph for this example or not.

4.2 Low-Rank Approximation

Since the Netflix competition [Bell and Koren, 2007], collaborative filtering techniques with latent factor models have had huge success in recommender systems. These latent factors, often in the form of low-rank embeddings, capture not only explicit information but also implicit dependencies from the input data. Following prior work [Koren *et al.*, 2009], we are interested in learning two low-rank matrices $P \in \mathbf{R}^{k\times m}$ and $Q \in \mathbf{R}^{k\times n}$. The intuition is that P is the embedding of all

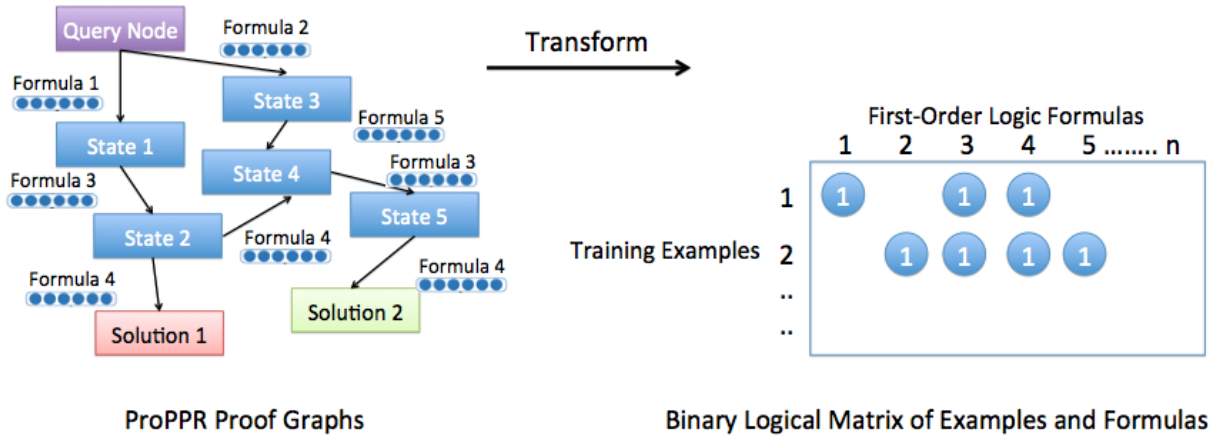

Figure 1: The Matrix Factorization Framework for Learning First-Order Logic Embeddings.

examples, and Q is the embedding of first-order logic formulas.

Here k is the number of latent dimensions, and we would like to approximate $F_{(i,j)} \simeq \vec{p_i}^T \vec{q_j}$, where $\vec{p_i}$ is the latent embedding vector for the i-th example and $\vec{q_j}$ is the latent embedding vector for the j-th column. We seek to approximate the matrix F by these two low-rank matrices P and Q. We can then formulate the optimization problem for this task:

$$\min_{P,Q} \sum_{(i,j)\in F} (F_{(i,j)} - \vec{p_i}^T \vec{q_j})^2 + \lambda_P ||\vec{p_i}||^2 + \lambda_Q ||\vec{q_j}||^2$$

here, λ_P and λ_Q are regularization coefficients to prevent the model from overfitting. To solve this optimization problem efficiently, a popular approach is stochastic gradient descent (SGD) [Koren *et al.*, 2009]. In contrast to traditional methods that require time-consuming gradient computation, SGD takes only a small number of random samples to compute the gradient. SGD is also natural to online algorithms in real-time streaming applications, where instead of retraining the model with all the data, parameters might be updated incrementally when new data comes in. Once we have selected a random sample $F_{(i,j)}$, we can simplify the objective function:

$$(F_{(i,j)} - \vec{p_i}^T \vec{q_j})^2 + \lambda_P (\vec{p_i}^T \vec{p_i}) + \lambda_Q (\vec{q_j}^T \vec{q_j})$$

Now, we can calculate the sub-gradients of the two latent vectors $\vec{p_i}$ and $\vec{q_j}$ to derive the following variable update rules:

$$\vec{p_i} \leftarrow \vec{p_i} + \delta(\ell_{(i,j)} q_j - \lambda_P \vec{p_i}) \quad (3)$$

$$\vec{q_j} \leftarrow \vec{q_j} + \delta(\ell_{(i,j)} p_i - \lambda_Q \vec{q_j}) \quad (4)$$

Here, δ is the learning rate, whereas $\ell_{(i,j)}$ is the loss function that estimates how well the model approximates the ground truth:

$$\ell(i,j) = F_{(i,j)} - \vec{p_i}^T \vec{q_j}$$

The low-rank approximation here is accomplished by reconstructing the F matrix with two low-rank matrices P and Q, and we use the row and column regularizers to prevent the model from overfitting to the training data.

In addition to standard loss functions such as logarithm loss, hinge loss, and squared hinge loss, we have also experimented with a pairwise loss function that resembles Bayesian Personalized Ranking [Rendle *et al.*, 2009]. It is recently shown that BPR is effective in many knowledge graph learning problems [Riedel *et al.*, 2013; Chen *et al.*, 2015]. The idea is that we assume that all positive, activated formulas should be ranked above all missing formulas (i.e., unused formulas) in each example row R. And the objective function could be written as:

$$\begin{aligned} min_{P,Q} \sum_{(i,j)\in R} \sum_{(i,w)\notin R_i} & \log(1+\exp(\vec{p_i}^T(\vec{q_w} - \vec{q_j}))) \\ & + \mu_p ||\vec{p_i}||_1 + \mu_q (||\vec{q_j}||_1 + ||\vec{q_w}||_1) \\ & + \frac{\lambda_p}{2} ||\vec{p_i}||_2^2 + \frac{\lambda_q}{2} (||\vec{q_j}||_2^2 + ||\vec{q_w}||_2^2) \end{aligned} \quad (5)$$

where R_i is the set of positive formulas in the i-th row of R. If we switch the notation of i as column index, j and w as row indexes, then this method will be optimizing column-oriented BPR instead of row-oriented BPR. In this work, we follow a recently proposed optimization approach called fast parallel stochastic gradient descent (FPSG) [Chin *et al.*, 2015] to learn the latent first-order logic embeddings.

4.3 Learning First-Order Logic Embeddings

We outline our matrix factorization based method in Algorithm 1. Since this is a supervised learning approach, we assume the dataset includes a collection of triples as examples T.

During training, we first apply the gradient-based structure learning method [Wang *et al.*, 2014a] to all training triples T^{tr} to derive a set of plausible first-order formulas S. Given this subset of formulas, we then traverse all training examples to produce grounded ProPPR proof graphs G. For each training example i, we construct a row F_i in a matrix F. The columns correspond to the formulas learned by structural gradient, and F_i is non-zero if this particular formula is used during grounding. Figure 1 illustrates the transformation process. Then, we perform stochastic gradient descent training to

Algorithm 1 A Matrix Factorization Based Algorithm for Learning First-Order Logic Embeddings

Input: training examples T in the form of relation triples.
procedure TRAINING(T^{tr})
  $S \leftarrow$ StructureLearning(T^{tr})
  for each training example T_i^{tr} in T^{tr} **do**
    $G_i \leftarrow$ Grounding(T_i^{tr},S)
    $F_i \leftarrow$ GraphToMatrix(G_i)
    for each epoch e **do**
      for each cell i, j in F_i **do**
        $\vec{p_i}^{(e)} \leftarrow \vec{p_i}^{(e)} + \delta(\ell_{(i,j)} q_j^{(e)} - \lambda_P \vec{p_i}^{(e)})$
        $\vec{q_j}^{(e)} \leftarrow \vec{q_j}^{(e)} + \delta(\ell_{(i,j)} p_i^{(e)} - \lambda_Q \vec{q_j}^{(e)})$
      end for
    end for
  end for
  for each formula embedding Q_j in Q **do**
    $\theta_j \leftarrow mean(Q_j)$
  end for
end procedure
procedure TESTING(T^{te})
  for each test example T_i^{te} in T^{te} **do**
    $\vec{A} \leftarrow$ QueryAnswering(T_i^{te},S,Θ)
  end for
end procedure

Datasets	#Rel.	#En.	#Train	#Valid	#Test
WordNet	18	40,943	141,442	5,000	5,000
FB15K	1,345	14,951	483,142	50,000	59,071

Table 2: Statistics of the two publicly available datasets used in the knowledge base completion experiments. *Rel.: relations. En.: entities.*

learn the hidden low-rank embeddings of examples and formulas P and Q using the update rules outlined earlier. After learning the embeddings for the formulas, we average all the dimensions of the learned embedding of each formula to derive a parameter θ_j. During testing, we still ground the testing examples using a set of candidate formulas S, but now the transition of the proof process will be guided by the θ_js.

5 Experiments

In this section, we first introduce datasets used in this knowledge base completion study and the evaluation protocol. Then, we highlight the baselines. Finally, we show empirical results on these two datasets, including analyses on robustness and the effects of different loss functions for learning first-order logic embeddings.

5.1 Datasets and Evaluation Setup

We choose two popular datasets for the task of knowledge base completion. The statistics of the datasets are shown in Table 2. The task is to predict the missing head or tail entities in a relation fact triple: given the relation and an entity, we rank the candidate entities. Following prior work [Bordes *et al.*, 2013], we use the Hits@10 measure, which indicates the portion of correct answers falling in the top-10 rank. For example, if all testing queries have correct answers returned in the top-10 positions, then the system will have a perfect Hits@10 score of 100. We do not filter any triples in the experiments. Except for subsection 5.4, the latent dimension of first-order logic embeddings is set to 8. ProPPR's reset parameter α is set to 0.1, and the approximation error parameter ϵ is set to 1×10^{-3}.

Methods	Hits@10
Unstructured [Bordes *et al.*, 2014]	4.5
RESCAL [Nickel *et al.*, 2011]	28.4
SE [Bordes *et al.*, 2011]	28.8
SME [Bordes *et al.*, 2014]	31.3
LFM [Jenatton *et al.*, 2012]	26.0
TransE [Bordes *et al.*, 2013]	34.9
ConvNets [Shi and Zhu, 2015]	37.7
TransH [Wang *et al.*, 2014b]	45.7
TransR [Lin *et al.*, 2015b]	48.4
PTransE [Lin *et al.*, 2015a]	51.8
ProPPR	59.0
ProPPR+MF (k=8)	**61.0**

Table 3: Comparing our approach with various baselines on the FB15K dataset.

5.2 Baselines

To demonstrate the effectiveness of our method, we compare with a number of competitive and state-of-the-art baselines— **Unstructured** [Bordes *et al.*, 2014]: a TransE [Bordes *et al.*, 2013] baseline that considers the data as mono-relational and sets all translations to 0; **RESCAL** [Nickel *et al.*, 2011]: a collective matrix factorization model that factorizes a rank-3 tensor; **SE** [Bordes *et al.*, 2011], **SME** [Bordes *et al.*, 2014], and **LFM** [Jenatton *et al.*, 2012]: energy-based structured embedding models of knowledge bases; **TransE** [Bordes *et al.*, 2013]: a popular multi-relational model that considers relationships as translations in the embedding space; **ConvNets** [Shi and Zhu, 2015]: a recent study on convolutional neural network based concept learning model. **TransH** [Wang *et al.*, 2014b]: an improved TransE-style model that considers reflexive, 1-to-N, N-to-1, and N-to-N relations; **TransR** [Lin *et al.*, 2015b]: a state-of-the-art model that builds entity and relation embeddings in separate entity space and relation spaces; **PTransE** [Lin *et al.*, 2015a]: a state-of-the-art TransE-style method that uses path-based constraints to learn KB embeddings[1].

5.3 Comparing with Various Baselines

We show the comparison of our approaches with various methods on the FB15K dataset in Table 3. We see that TransE and its variants outperform energy based models such as SE, SME, and LFM. The ConvNets model also achieves reasonable results. The best results from prior work on this dataset is from PTransE, a TransE-like model with relational path as constraints during training. We see that the structural gradient based approach from ProPPR obtains a strong performance, leveraging the effectiveness of symbolic reasoning and probabilistic modeling. Finally, leading

[1]Note that PTransE and ConvNets do include results for the WordNet dataset.

Methods	Hits@10
Unstructured [Bordes *et al.*, 2014]	35.5
RESCAL [Nickel *et al.*, 2011]	37.2
SE [Bordes *et al.*, 2011]	68.5
SME [Bordes *et al.*, 2014]	65.1
LFM [Jenatton *et al.*, 2012]	71.4
TransE [Bordes *et al.*, 2013]	75.4
TransH [Wang *et al.*, 2014b]	75.4
TransR [Lin *et al.*, 2015b]	79.8
ProPPR	83.0
ProPPR+MF (k=8)	**94.1**

Table 4: Comparing our approach with various baselines on the WordNet dataset.

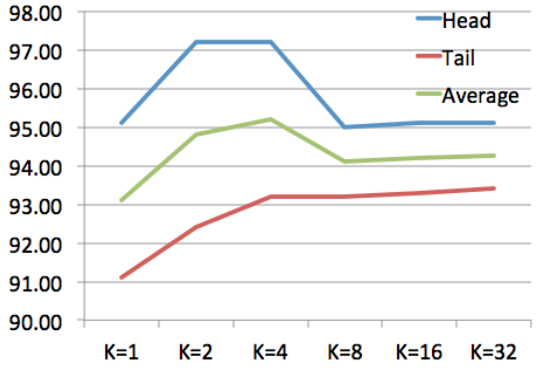

Figure 2: The Hits@10 scores of varying #dimensions for retrieving head and tail entities on the WordNet dataset.

the scoreboard is our first-order logic embedding enhanced ProPPR model with a Hits@10 score at 61.0.

On the WordNet dataset, we show the comparison of performances in Table 4. We observe similar trends of results from prior work: TransE and its variants have dominated the score sheet on this WordNet dataset. ProPPR's structure learning produces a strong result of 83.0, showing ProPPR's advantage on modeling hypernym and hyponym relations. When considering first-order formula embeddings, we observe a large improvement of Hits@10 score to 94.1. Comparing to the prior experimental results from the FB15K dataset, we believe that the large-margin improvement observed from the WordNet experiment is due to the nature of these two datasets. FB15K has more than 1K relations, but it has only about 15K entities. We observe a large number of first-order formula candidates after structure learning, while the training examples for learning first-order logic embeddings are relatively sparse. In contrast, the WordNet data set has only over a dozen relations, but the size of total entities are about three times larger, allowing more training examples.

5.4 Varying the Latent Dimensions

In this experiment, we vary the latent dimension k for learning first-order logic embeddings. The results are shown in Figure 2. We see that the latent dimension has an effect on the performance of the WordNet dataset. When choosing a single dimension or a large dimension, they do not yield the best result. $k = 4$ yields the best overall performance.

Figure 3: The Hits@10 scores of different loss functions for learning embeddings to retrieve head and tail entities on the WordNet dataset.

5.5 Varying the Loss Functions

In this subsection, we investigate the effects of choosing various loss functions for learning first-order logic embeddings. We see that the general loss functions, such as the logarithm loss, hinge loss, and the squared hinge loss do not make much difference on the results. Interesting, when considering row-oriented pairwise log loss, we obtain the best performance. This pairwise loss function resembles Bayesian Personalized Ranking [Rendle *et al.*, 2009], which assumes that all positive first-order logic formulas should be ranked above all missing formulas in an example. This is a useful assumption for the pairwise loss function, which allows us to model the non-existent "negative" formulas. Interestingly, column-oriented pairwise log loss obtains the lowest performances, because it is strange to compare formulas across totally different examples.

6 Conclusion

In this paper, we introduce a method of learning first-order logic embeddings for probabilistic inference. The approach is built on prior work of embedding entities and relations, as well as structure learning in a scalable probabilistic logic. We show that by using a matrix factorization method, it is possible to learn embeddings for first-order logic formulas. In empirical studies, we show that representation learning for first-order logics improves the task of knowledge base completion on two large datasets. In the future, we are interested in learning joint entity, relation, and formula embeddings for combining symbolic and statistical inference.

Acknowledgment

This work was sponsored in part by DARPA grant FA87501220342 to CMU.

References

[Andersen *et al.*, 2008] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local partitioning for directed graphs using pagerank. *Internet Mathematics*, 5(1):3–22, 2008.

[Bell and Koren, 2007] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[Bordes *et al.*, 2014] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[Chen *et al.*, 2015] Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alex I. Rudnicky. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL-IJCNLP*, Beijing, China, 2015. ACL.

[Chin *et al.*, 2015] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):2, 2015.

[Csalogny *et al.*, 2005] Kroly Csalogny, Dniel Fogaras, Balzs Rcz, and Tams Sarls. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[Cussens, 2001] James Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[Gutmann *et al.*, 2008] Bernd Gutmann, Angelika Kimmig, Kristian Kersting, and Luc De Raedt. Parameter learning in probabilistic databases: A least squares approach. In *Machine Learning and Knowledge Discovery in Databases*, pages 473–488. Springer, 2008.

[Guu *et al.*, 2015] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *EMNLP*, 2015.

[Jenatton *et al.*, 2012] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *NIPS*, 2012.

[Khosravi *et al.*, 2010] Hassan Khosravi, Oliver Schulte, Tong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, 2010.

[Khot *et al.*, 2011] Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, 2011.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[Lin *et al.*, 2015a] Yankai Lin, Zhiyuan Liu, and Maosong Sun. Modeling relation paths for representation learning of knowledge bases. *EMNLP*, 2015.

[Lin *et al.*, 2015b] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Muggleton *et al.*, 2014] Stephen H Muggleton, Dianhuan Lin, Jianzhong Chen, and Alireza Tamaddoni-Nezhad. Metabayes: Bayesian meta-interpretative learning using higher-order stochastic refinement. In *Inductive Logic Programming*, pages 1–17. Springer, 2014.

[Muggleton, 1996] Stephen Muggleton. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.

[Muggleton, 2000] Stephen Muggleton. Learning stochastic logic programs. *Electron. Trans. Artif. Intell.*, 4(B):141–153, 2000.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[Page *et al.*, 1998] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. In *Technical Report, Computer Science department, Stanford University*, 1998.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 2006.

[Riedel *et al.*, 2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*, 2013.

[Rocktäschel *et al.*, 2014] Tim Rocktäschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. Low-dimensional embeddings of logic. In *ACL 2014 Workshop on Semantic Parsing*, 2014.

[Shi and Zhu, 2015] Jiaxin Shi and Jun Zhu. Building memory with concept learning capabilities from large-scale knowledge base. *NIPS 2015 Cognitive Computation workshop*, 2015.

[Van Daele *et al.*, 2014] Dries Van Daele, Angelika Kimmig, and Luc De Raedt. Pagerank, proppr, and stochastic logic programs. 2014.

[Wang *et al.*, 2013] William Yang Wang, Kathryn Mazaitis, and William W Cohen. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *CIKM*, 2013.

[Wang *et al.*, 2014a] William Yang Wang, Kathryn Mazaitis, and William W Cohen. Structure learning via parameter learning. *CIKM*, 2014.

[Wang *et al.*, 2014b] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer, 2014.

[Wang *et al.*, 2015] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *IJCAI*, 2015.

[Wei *et al.*, 2015] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *CIKM*, 2015.