Low Resource Quadratic Forms for Knowledge Graph Embeddings

Zachary Zhou
University of Wisconsin – Madison
Industrial & Systems Engineering
1415 Engineering Drive
Madison, WI 53706
zzhou246@wisc.edu

Jeffery Kline
Devin Conathan
Glenn Fung
American Family Insurance
6000 American Parkway
Madison, WI 53783
{jklin1,dconatha,gfung}@amfam.com

Abstract

We address the problem of *link prediction* between entities and relations of knowledge graphs. State of the art techniques that address this problem, while increasingly accurate, are computationally intensive. In this paper we cast link prediction as a sparse convex program whose solution defines a quadratic form that is used as a ranking function. The structure of our convex program is such that standard support vector machine software packages, which are numerically robust and efficient, can solve it. We show that on benchmark data sets, our model's performance is competitive with state of the art models, but training times can be reduced by a factor of 40 using only CPU-based (and not GPU-accelerated) computing resources. This approach may be suitable for applications where balancing the demands of graph completion performance against computational efficiency is a desirable trade-off.

1 Introduction

Digital representations of knowledge are essential to many machine learning domains including social networks, citation networks, question-answering, information retrieval, recommender systems, and natural language processing (Zhang et al., 2016; Xiong et al., 2017; Hao et al., 2017; Yang and Mitchell, 2017). Multirelational data, when represented as a directed graph whose nodes (entities) are connected to each other by one or more edges (relations) are *knowledge graphs*.

Knowledge graphs are often both large and incomplete. As a result, the task of *link prediction* is a basic challenge. Much recent work on knowledge graph representation has focused on constructing embeddings of the relations and entities of a graph to facilitate good performance on the link prediction problem. But training modern models on large graphs often consumes many hours of GPU time.

In this paper, we formulate the link prediction problem as a sparse convex program that can be efficiently solved by off-the-shelf support vector machine (SVM) software packages. We demonstrate performance that is competitive with state of the art methods, but our programs are solved in substantially shorter times and without requiring specialized GPU hardware. Our approach takes advantage of existing SVM technology: large-scale convex problems are routinely solved using an arsenal of first and second order methods, as well as decades of empirically-derived heuristics. Solutions to convex problems are global, often unique, and in some cases (e.g. linear programs), solutions come with theoretical guarantees about program complexity. While we do not pursue this idea in the present work, it is possible to naturally incorporate external knowledge about relationships into convex programs (Fung et al., 2002). An unusual feature of our program is that the solution is not used as a binary classifier, but instead, the solution is used to define a symmetric quadratic form that ranks facts in the knowledge graph.

The main contributions of this work include 1) a novel convex formulation of the link prediction problem 2) a scalable approach to link prediction that that can be efficiently solved using standard SVM solvers and 3) a link prediction model that exhibits competitive performance.

We now describe the remaining sections of this paper. In the next section we present notation and background information on SVMs and the TransE model that we base our method on. In Section 3 we present details of our SVM formulation. In Section 4 we comment on implementation. Experiments are discussed in Section 5. Results are presented in Section 6. Related work and conclusions are in Section 7 and Section 8, respectively.

2 Background

Basic Concepts Each *fact* in a knowledge graph is represented as a triple, the entries of which are called a *head*, a *relation* and a *tail*. The head and

Proceedings of the 2nd Workshop on Simple and Efficient Natural Language Processing, pages 1–10
November 10, 2021.

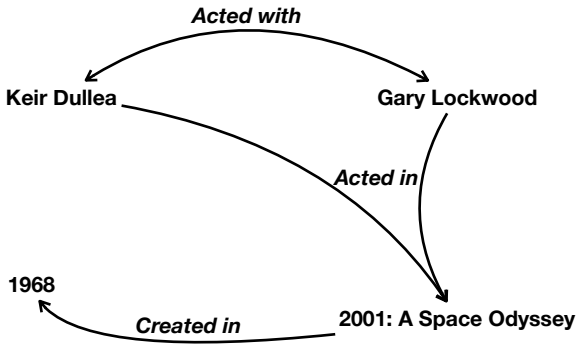

Figure 1: Example of a knowledge graph.

tail belong to a set of *entities*. More formally, given a set of relations R and entities E, the standard representation of a knowledge graph is a set $K \subset E \times R \times E$, and each $(h, r, t) \in K$ is a fact. Figure 1 illustrates a simple knowledge graph having four entities (Keir Dullea, Gary Lockwood, 2001: A Space Odyssey, and 1968) and three relations (Acted with, Acted in, and Created in).

Given a set of entities E and relations R, a knowledge graph embedding (KGE) model embeds the elements of $E \cup R$ as elements in $\mathbb{R}^k$ for a fixed positive integer k. For brevity, we abuse notation and will represent the elements and the embeddings of the elements in E and R with the same symbol. Thus, the expressions $e + r$, αe for any $\alpha \in \mathbb{R}$, and e' are well-defined, and represent component-wise addition, scalar multiplication, and transposition, respectively. Additionally, $\|x\|_p \coloneqq (\sum |x_i|^p)^{1/p}$. Unless otherwise specified, we assume $p = 2$. For each $(h, r, t) \in K$, we define a set of *corrupted* triples as

$$K^{\dagger}_{(h,r,t)} = \left\{(h, r, t^{\dagger}) : t^{\dagger} \in E\right\} \cup \left\{(h^{\dagger}, r, t) : h^{\dagger} \in E\right\} \setminus K.$$

That is, $K^{\dagger}_{(h,r,t)}$ contains triples where either h or t is replaced by $h^{\dagger} \in E$ or $t^{\dagger} \in E$ so that the new triple is not in K. For ease of notation we will often write corrupted triples as $(h^{\dagger}, r, t^{\dagger}) \in K^{\dagger}_{(h,r,t)}$ (where both $h^{\dagger}$ and $t^{\dagger}$ appear), but it is always the case that precisely one of h or t is replaced, not both.

A link prediction model's *score function* is the function that distinguishes positive triples from corrupted triples. If f is a score function, then a well-performing model satisfies $f(h, r, t) \leq f(h^{\dagger}, r, t^{\dagger})$ when $(h, r, t) \in K$ and $(h^{\dagger}, r, t^{\dagger}) \in K^{\dagger}_{(h,r,t)}$.

TransE Our approach is inspired by the "translational model", TransE, introduced in (Bordes et al., 2013). Despite its age and simplicity, TransE exhibits performance that is often competitive with state-of-the-art techniques, and our SVM-based approach shares several key features with it. The goal of TransE is to make $\|h + r - t\|$ small whenever $(h, r, t) \in K$ and large otherwise. To do this, one minimizes the loss function

$$L_{\text{TransE}} \coloneqq \sum_{\substack{(h,r,t) \in K \\ (h^{\dagger}, r, t^{\dagger}) \in K^{\dagger}_{(h,r,t)}}} \tau(h, r, t, h^{\dagger}, t^{\dagger}),$$

where

$$\tau(h, r, t, h^{\dagger}, t^{\dagger}) \coloneqq \left(\gamma + \|h + r - t\| - \left\|h^{\dagger} + r - t^{\dagger}\right\|\right)_+,$$

$\gamma \in \mathbb{R}$ is a fixed margin, the norm is either 1-norm or the 2-norm and $(y)_+ \coloneqq \max\{y, 0\}$. The score function for TransE appears directly in the function L_{TransE}, and is $\|h + r - t\|$. Note that L_{TransE} is not convex due to the term $-\left\|h^{\dagger} + r - t^{\dagger}\right\|$ in the definition of τ.

Support Vector Machines A standard linear SVM, trained on data $x_i \in \mathbb{R}^k$ with labels $y_i = \pm 1$ for $i = 1, \ldots, N$, solves the linearly constrained convex optimization problem,

$$\min_{\substack{w \in \mathbb{R}^k \\ \zeta \in \mathbb{R}^N \\ \gamma \in \mathbb{R}}} C \|\zeta\|^2 + \|w\|^2$$
$$\text{subject to} \quad y_i \left(w' x_i - \gamma\right) \geq 1 - \zeta_i, (1 \leq i \leq N)$$
$$\zeta \geq 0,$$

where $C > 0$ is a regularization parameter. This constrained program may be rewritten as an equivalent unconstrained problem,

$$\min_{w, \gamma} C \left\|(\mathbf{1} - Y(Xw - \gamma \mathbf{1}))_+\right\|^2 + \|w\|^2,$$

where Y is the diagonal matrix with $Y_{ii} = y_i$ and row i of X is x_i' and $\mathbf{1} = (1, 1, 1, \ldots, 1)$.

Given an optimal solution w of this SVM, the function $\phi : x \mapsto w'x - \gamma$ defines a hyperplane that separates $\mathbb{R}^k$ into two half-spaces, $\{x : w'x - \gamma > 0\}$ and $\{x : w'x - \gamma < 0\}$. If $y_i(w'x_i + \gamma) > 0$, for all $i = 1, \ldots, N$, then the data used to train the SVM are linearly separable The function ϕ is the decision function of the SVM. Classification error is measured by $\|\zeta\|^2$. The term

$\|w\|^2$ maximizes the margin between the bounding planes $w'x = \gamma \pm 1$ (Mangasarian, 2006).

One attractive feature of casting a problem as an SVM is the wide availability of high quality SVM software packages. Such packages are numerically robust, and are capable of rapidly solving problems with massive scale.

3 Formulation

We now describe how to express the link prediction problem as a convex variant of TransE, and then we show how to write this convex problem as an SVM. Recall that our goal is to identify a score function $f : E \times R \times E \to \mathbb{R}$ such that

$$f(h + r - t) \leq f(h^\dagger + r - t^\dagger) \tag{1}$$

for all $(h, r, t) \in K$ and $(h^\dagger, r, t^\dagger) \in K^\dagger_{(h,r,t)}$. We will assume that f has the form,

$$f(h + r - t) = \|M(h + r - t)\|, \tag{2}$$

where $M \in R^{k\times k}$ is some unknown matrix. Finding M is essentially the main goal of TransE. We now show how, modulo technical artifacts, one may *define* the matrix M (and hence f) as the optimal solution to an SVM.

Substituting the right-hand side of (2) into (1), our problem becomes finding a matrix $M \in \mathbb{R}^{k\times k}$ such that

$$\|M(h + r - t)\| \leq \left\|M(h^\dagger + r - t^\dagger)\right\| \tag{3}$$

whenever $(h, r, t) \in K$ and $(h^\dagger, r, t^\dagger) \in K^\dagger_{(h,r,t)}$.

If we define $B \coloneqq M'M$, then our problem is equivalent to seeking a positive semi-definite $B \in \mathbb{R}^{k\times k}$ with

$$\begin{aligned}(h + r - t)'B(h + r - t) \leq \\ (h^\dagger + r - t^\dagger)'B(h^\dagger + r - t^\dagger)\end{aligned} \tag{4}$$

It is possible to represent the positive semi-definite condition on B as a convex constraint, but optimizing a loss function with this additional constraint is incompatible with an SVM formulation. Since our main goal is to create a score function, we omit the positive definite constraint from our program.

The inequality expressed in (4) is *quadratic* in the problem data, but *linear* in the unknown, B. Indeed, each side of the inequality (4) expresses a quadratic polynomial in the entries of h, r and t, while both expressions are linear in the entries of B.

To make this more explicit, we introduce notation. Let $\operatorname{vec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ denote the invertible linear map that "flattens" an $n \times n$ matrix into a n^2-dimensional vector. Then

$$x'Ay = \left(\operatorname{vec} xy'\right)' \operatorname{vec} A \tag{5}$$

for any $x, y \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$. When A is a symmetric matrix and $x = y$, then (5) can be adapted to apply to the upper-triangular portions of A and xx'.

Just as vec maps square matrices to vectors, one may define symvec that maps symmetric matrices in $\mathbb{R}^{n\times n}$ to vectors in $\mathbb{R}^{n(n+1)/2}$, and which satisfies

$$x'Ax = \left(\operatorname{symvec} xx'\right)' \operatorname{symvec} A$$

For brevity, we define

$$\mathrm{S}(h, r, t) \coloneqq \operatorname{symvec} (h + r - t)(h + r - t)' \tag{6}$$

With notation in hand, initialize the embeddings of h,r and t to be one-hot encoded. Our task is to find $w \in \mathbb{R}^{k(k+1)/2}$ that satisfies

$$w'\mathrm{S}(h, r, t) \leq w'\mathrm{S}(h^\dagger, r, t^\dagger),$$

for all $(h, r, t) \in K$ and $(h^\dagger, r, t^\dagger), \in K^\dagger_{(h,r,t)}$. To guarantee that the feasible set of this program is nonempty, we introduce non-negative slack variables ζ_i and minimize the residual.

This construction is similar in nature to the well-known "pair-wise transform" (Herbrich et al., 1999) where the ordinal ranking problem its transformed into a two-class classification problem by considering the difference of all comparable elements.

If we define $x_{(h,r,t),(h^\dagger,r,t^\dagger)} \coloneqq \mathrm{S}(h^\dagger, r, t^\dagger) - \mathrm{S}(h, r, t)$, then this allows us to concisely express a convex program that is identical to the SVM formulation described above, namely

$$\begin{aligned}\min_{w\in\mathbb{R}^{k(k+1)/2},\zeta} \quad & C\|\zeta\|^2 + \|w\|^2 \\ \text{subject to} \quad & w'x_\eta \geq 1 - \zeta_\eta, \\ & \qquad \left(\eta \in K_{(h,r,t)} \times K^\dagger_{(h,r,t)}\right) \\ & \zeta \geq 0.\end{aligned}$$

In practice, the term in the objective

$$\|\zeta\|^2 = \sum_i \zeta_i^2$$

is replaced with the weighted sum,

$$\sum_i \rho_i \zeta_i^2,$$

where $\rho \geq 0$ is chosen to normalize for problem parameters. Also, rather than include every $(h^\dagger, r, t^\dagger)$ for a given (h, r, t), a fixed integer N is chosen to specify the number of negative triples per positive triple. Our model has two parameters that impact problem performance: integer N and $C > 0$, which is the standard SVM regularization parameter. Performance generally improves as N increases, and selecting $C = 1$ is, we find empirically, a reasonable default. The normalization weighting we choose is uniform, so that each entry of the weight ρ is $1/N$. Finally, since the SVM solver that we applied to this problem requires both $+1$ and -1 labels be present in the data, we define label y_η as a randomly-selected choice from $\{+1, -1\}$, and then we scale x_η by y_η. This step has no impact on the optimal solution.

Our proposed model has several distinguishing features:

- The program data is extremely sparse, with perhaps 0.0001% or fewer nonzero entries in the problem data.
- Our model aims to define a *symmetric* matrix using the coefficients of a standard SVM. We gain symmetry by virtue of how our problem is constructed. The matrix is usually left implicitly defined, and we have no need to reconstruct it explicitly.
- While SVMs are typically employed for the purpose of binary classification, in our application, we are interested in the *score* attributed to triples, not the classification ability.
- Our problem has just two parameters: an integer N that affects the program size, and a regularization parameter, $C > 0$.
- The program, despite its unusually size and structure, can be solved very quickly, often in under a minute for quite large programs.

input $(h, r, t) \in K$, $N \in \mathbb{N}$
init $X = []$, $y = []$
while $\text{len}(X) < N$ **do**
  Choose $\epsilon \in \{\pm 1\}$ with probability 1/2
  Choose $h^\dagger \in E$ s.t. $(h^\dagger, r, t) \notin K$
  Append $\epsilon \left(S(h^\dagger, r, t) - S(h, r, t)\right)$ to X
  Append ϵ to y
end while
return X, y

Algorithm 1: Building N samples for the SVM by *corrupting* the head entity.

4 Implementation

We solve our programs using the LinearSVC class from scikit-learn (Pedregosa et al., 2011), which uses liblinear (Fan et al., 2008), NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). In particular we rely heavily on SciPy for its sparse array structures. The primary bottleneck is a one-time cost to build the set of negative triples which, fortunately, is an embarrassingly parallel workload. To reduce the cost in time to build the problem, we leverage the multiprocessing capability of the DataLoader class of PyTorch (Paszke et al., 2019).

One of the two main loops used to construct the data fed into the SVM is presented in Algorithm 1. The loop shown selects the negative triples where the head of a given positive is replaced. A separate, and nearly identical loop, builds the tail-corrupted triples.

The returned values of Algorithm 1 are problem data X with labels y. Since the labels are selected with balanced probability, the expected bias of the optimal solution is 0, and the SVM solver is instructed to set the intercept to 0.

A naive implementation of the program does not scale due to large memory requirements. Several tricks are employed when constructing the SVM data and evaluating metrics.

Each relation and entity is 1-hot encoded as a vector in $k := |E| + |R|$ dimensions, and we seek a symmetric matrix that acts on k-dimensional embeddings; our problem dimension is $k(k+1)/2$. Even for modest values of k, the problems passed to SVM solvers can be quite large (see Table 1). Indeed, the included liblinear Python interface experienced integer overflow errors; however, using liblinear via scikit-learn performed adequately on a single-threaded standard CPU-based architecture that we used to compute the metrics for this pa-

per.[1] While the problems are quite large they are extremely sparse. Indeed, independent of N, there are at most 12 nonzero entries in each row of X, and the memory requirements for the program are $O(N\,|K|)$.

Table 1: Problem size for $N = 100$. Problems are embedded in very high dimension, but the memory requirements are modest due to the extreme sparsity.

	FB15K	WN18	PRM45K
k	16296	40961	2884
ROWS OF X	106 MIL	29.3 MIL	7.26 MIL
COLUMNS OF X	132 MIL	8.39 BIL	4.16 MIL
COMPRESSED	1.4G	406M	84M
UNCOMPRESSED	7.5G	2.0G	332M

We now detail an indexing trick which leverages this sparsity and can be used to improve evaluation of metrics by several orders of magnitude over a naive direct evaluation.

If B is a symmetric matrix, and $w := \operatorname{vec} B$, then the entry of w that corresponds to entry $i \leq j$ of the matrix is

$$\text{ij2idx}(i,j) := \frac{k(k-1)}{2} - \frac{(k-i)(k-i-1)}{2} + j$$

If x and y are one-hot encoded and supported on indices i_x and i_y, then:

$$x'By = B_{i_x i_y} = B_{i_y i_x}$$

where the second equality holds by the assumption that B is symmetric. Then:

$$(h+r-t)'B(h+r-t) = B_{i_h i_h} + B_{i_r i_r} + B_{i_t i_t} + 2B_{i_h i_r} - 2B_{i_h i_t} - 2B_{i_r i_t} \tag{7}$$

This has an equivalent form using the function S, which was defined in (6). In this form that uses S, there is a space-savings gain of 1/2. Additionally, one has the guarantee that the matrix B, when reconstructed from the coefficients of the SVM, is symmetric.

The feature $x := h + r - t$ has at most three nonzero entries and is itself sparse. In practice, it is expensive to evaluate the outer product xx' and project onto the upper-triangular portion millions of times. Instead we simply evaluate the quadratic form directly using (7); at most 6 entries with fixed values are required: three on-diagonal entries and three off-diagonal entries.

With such large investment for performance gains, one may ask whether it is possible to simply take advantage of GPU hardware, and the well-established gains that can be had beyond standard CPU processors. Unfortunately, widely-available GPUs do not effectively leverage sparse data structures for improved performance. Indeed, commercial linear and quadratic solvers, which are commonly employed to solve sparse problems, do not support GPU hardware (Glockner, 2020).

5 Experiments

We use three knowledge graph datasets to benchmark our implementation; two are the standard link prediction benchmark datasets Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995), and one is a new knowledge graph we introduce. We report performance on all three. For Freebase (FB15k) and WordNet (WN18) we use the versions distributed in the software package released by the authors of RotatE (Sun et al., 2019). We now briefly describe the data sets.

WN18 WordNet is a knowledge graph that provides lexical relationships between words (Glorot et al., 2013). It is designed to be machine readable, while being a usable dictionary and thesaurus.

FB15k Freebase is a collaboratively-constructed knowledge graph, containing general facts about the world (Bollacker et al., 2008). The FB15k dataset here is the same subset that was introduced in (Bordes et al., 2013).

PRM45k We introduce the product reference manual knowledge graph. It contains taxonomies, articles, frequently-asked-questions and glossary terms found in an insurance company operating manual intended to be referenced by insurance agents. The entities and relationships come from different source systems, and there is some overlap/redundancy, making this a good candidate for the link prediction task as there are many missing relationships. We have included a release of this dataset with our supplementary materials and plan to make a version of the dataset publicly available at http://anonymous.url. Summary statistics about these data sets appear in Table 2.

Hyperparameters We recommend searching over $N = 10, 50, 100$ and $C = 0.5, 1, 2, 4, 8, 16,$

[1] Despite the large size of the problem, we did not find that floating point errors interfered with overall results. We attribute this to the sparsity and modest parameter values.

Table 2: Benchmark data sets.

	FB15K	WN18	PRM45K
RELATIONS	1,345	18	6
ENTITIES	14,951	40,943	2878
TRIPLES	592,213	151,442	45,384

though we report on several other values of N. We find that performance increases in N and that performance is U-shaped in C, with a small neighborhood of values where all reported metrics are close to optimal. As a rule, fit time increases with the regularization parameter C. Table 3 describes performance and build time as a function of N.

Evaluation metrics The metrics we report here are standard metrics and have been used to compare link prediction model performance for nearly a decade (Bordes et al., 2011). Given a positive triple $(h, r, t) \in K$ we evaluate the score function applied to this triple, and then we evaluate the score function applied to all negative triples $(h^\dagger, r, t^\dagger) \in K^\dagger_{(h,r,t)}$. The scores are sorted in decreasing order and the rank of (h, r, t) is recorded. This process is repeated for all positive triples in whatever data we are evaluating on (e.g. the hold-out validation or test sets). The ranks of the positive triples across the data set are aggregated into mean rank (MR), mean reciprocal rank (MRR), and hits at k (H@k).

We note that while SVMs also have an accuracy score which measures the accuracy of the fitted model as a binary classifier, in the context of our knowledge graph link prediction problem this score is not relevant, and we do not report it.

6 Results

Since the model we use is very similar to that of TransE, we expect similar performance characteristics. Table 3 shows that performance gains taper off rather quickly as N grows.

Table 4 displays timing results for our SVM-based method. It also includes self-reported results of RotatE and estimates based on HolE.

Timings report build times using a 96 CPU core processor on a machine running Ubuntu 18.04. As problem build is highly parallelizable, times are inversely-proportional to the number of available cores. Since the SVM solver we use is single-threaded, the additional cores are not necessary for training; only sufficient memory is necessary. Thus, it makes sense to build the program with a compute-optimized configuration and solve the problem with a memory-optimized configuration. Multiple values of C may be fit simultaneously, provided local memory is sufficient.

Table 5 compares metrics of various state-of-the-art methods to ours. The TransE model is noteworthy for being both simple and easy to train (Nickel et al., 2016). We were unable to find reported training times of TransE, so we report here results using the TransE code that was released by RotatE authors. We allowed the model to run for 50,000 epochs on a NVIDIA Tesla K80 GPU using same parameters the authors reported. Note that 50,000 is significantly less than 150,000 used by RotatE authors. The FB15k model was stopped early by hand, and the WN18 model was still in the process of converging when the maximum iterations were reached, so the value of $2+$ is a lower bound.

In the TransE framework, the choice of solver and parameter settings (Minervini et al., 2015) has a dramatic impact on solver performance. Hyper-parameter complexity is in contrast to our use of the default parameter settings of the SVM solver, which we use without modification.

7 Related work

Support vector machines Support vector machines have been around for decades, and have proved remarkably versatile. Our use is not standard, for we are not interested in the capability of the SVM as a binary classifier. Traditionally, SVMs are a supervised learning method useful for classification problems. However, an early and very different approach where the distance to the margin is used as a ranking function is (Joachims, 2002). Our approach is the same in spirit: we are using the coefficients of the SVM to yield an implicit symmetric matrix which we use as a score function. SVMs have been used for the problem of link prediction in (Al Hasan et al., 2006). In that work, the authors treat link prediction as a binary classification problem, and SVMs are one of several binary classifiers they evaluated.

The SVM solver that we use, which was distributed in (Pedregosa et al., 2011), is single-threaded. This, in turn, calls the lower-level library (Chang and Lin, 2011), which is tailored to large-scale and very sparse support vector problems. The developers of the Liblinear solver have invested significant effort over many years to iden-

Table 3: Metrics for $C = 1.0$ for FB15k and PRM45k, and $C = 16.0$ for WN18. The metrics here are reported on the same machine used for Table 5. Information from the rows $N = 50$ and $N = 100$ are duplicated in Table 4 and Table 5 for convenience.

N	FB15K				WN18				PRM45K			
	FIT (SEC)	BUILD (SEC)	MRR	H@10	FIT (SEC)	BUILD (SEC)	MRR	H@10	FIT (SEC)	BUILD (SEC)	MRR	H@10
1	21	21	0.320	0.493	12	6	0.179	0.262	<1	2	0.275	0.44
2	58	28	0.398	0.614	116	8	0.368	0.704	<1	3	0.333	0.516
5	82	53	0.484	0.695	214	15	0.491	0.922	2	7	0.500	0.719
10	115	93	0.530	0.738	273	27	0.500	0.934	1	4	0.437	0.647
50	328	467	0.617	0.799	465	125	0.526	0.942	8	33	0.552	0.758
100	497	902	0.632	0.806	607	250	0.540	0.946	15	58	0.555	0.770

Table 4: Timing. Parameters for SVM are $N = 100$ both data sets, $C = 1.0$ for FB15k and $C = 16.0$ for WN18. The Liblinear software is single-threaded, so the primary constraint is memory. The estimate for HolE is based on 11s per epoch on WN18, and an estimated 200 to 500 epochs required for convergence.

MODEL	FB15K		WN18		PRM45K	
	FIT	BUILD	FIT	BUILD	FIT	BUILD
TRANSE	1 HR		2+ HRS		1 HR	
ROTATE	9 HRS		4 HRS		1 HR	
HOLE			1+ HRS (EST)			
SVM(N=50)	6 MIN	8 MIN	8 MIN	2 MIN	8 SEC	33 SEC
SVM(N=100)	9 MIN	15 MIN	10 MIN	4 MIN	15 SEC	1 MIN

tify and then leverage heuristics that improve convergence times. We note that during November 2020, a multi-core version of this lower-level library was released, and the authors claim substantial reduction in fitting times may be possible.

The soft-margin support vector network that we use was introduced in (Cortes and Vapnik, 1995). The idea of representing data quadratically and training an SVM to derive a symmetric matrix was applied in (Rosales and Fung, 2006) to remove the need to choose a distance function by hand and instead *define* a distance function as the solution to a convex program. In that work, the distance function was the solution to a linear program, and extracting a distance from the solution was made possible by enforcing side condition that guaranteed the resulting set of coefficients satisfied diagonal dominance. This allowed the reconstructed matrix to be positive semidefinite, and the optimal distance could then be extracted using a matrix square root, found through a Cholesky decomposition. In this work, we are representing data quadratically, but we do not have the side condition. A similar, but not quite equivalent idea, is to fit an SVM using a quadratic kernel.

Our SVM program is both sparse and convex. We note that very recent work (Tsai and El Ghaoui, 2020) in the space of extractive text summarization also leverages sparsity and convexity to efficiently solve an optimization program. In our program, when both terms in the objective are the 1-norm, the SVM may be written as a linear program. This is attractive, for linear programs have the theoretical guarantee of a polynomial-time solution.

Knowledge graph link prediction Relational learning and knowledge graph link prediction is a large field of machine learning research. One early model is RESCAL, which is a tensor decomposition method (Nickel et al., 2011) for learning relations in a knowledge graph. The idea of the model is to factorize a tensor into a core tensor multiplied by a matrix along each dimension. The core idea adapts the earlier DEDICOM (Harshman et al., 1982) tensor factorization scheme. This method was notable for scalability and improved training times.

Subsequent to TransE, which was a landmark evolution in the performance of link prediction, there has been much research and analysis into the structure of links in knowledge graphs (Trouillon et al., 2016; Toutanova and Chen, 2015; Guu et al., 2015). Relation paths, which are a compound structure, have also been used to extend the TransE model (Lin et al., 2015). Since its introduction, and

Table 5: Results for TransE(1) are from (Bordes et al., 2013), TransE(2) and HolE are from (Nickel et al., 2016), and RotatE are from (Sun et al., 2019). Parameters for SVM are $C = 1.0$ for FB15k and $C = 16.0$ for WN18. We compute and report metrics for the filtered setting, as described in (Bordes et al., 2013).

Model	FB15k				WN18				PRM45k			
	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1
TransE(1)		0.471				0.892						
TransE(2)	0.463	0.749	0.578	0.297	0.495	0.943	0.888	0.113	0.518	0.680	0.569	0.428
HolE	0.524	0.739	0.613	0.402	0.938	0.949	0.945	0.930				
RotatE	0.797	0.884	0.830	0.746	0.949	0.959	0.952	0.944	0.518	0.675	0.565	0.432
SVM(N=50)	0.617	0.799	0.676	0.520	0.526	0.942	0.757	0.271	0.553	0.758	0.632	0.431
SVM(N=100)	0.632	0.806	0.690	0.538	0.539	0.944	0.766	0.289	0.554	0.770	0.634	0.433

partally as a result of this work, limitations of the TransE data model were identifed (Nickel et al., 2016). We expect that the known limitations that impact TransE also affect our approach. Despite these limitations, TransE's performance remains competitive on many real-world and benchmark data sets.

Recently, RotatE (Sun et al., 2019) sought to address a variety of the limitations known about TransE. RotatE embeds entities and relations in complex space and treats relations as rotations in the complex plane; the model achieves improved raw performance on standard benchmarks. It is believed that the additional freedom granted by the complex geometry contributes to the success of RotatE at learning graph structure. The authors also introduce *negative adversarial sampling* for link prediction. The idea of using negative sampling has proven quite effective for both learning knowledge graph embeddings (Trouillon et al., 2016) and word embeddings (Mikolov et al., 2013).

Case-based reasoning for knowledge graph completion was introduced in (Das et al., 2020). The authors show that a k-nearest neighbor (KNN) based approach for can be both efficient and scalable for this application.

8 Conclusion and future work

We demonstrate feasibility of using off-the-shelf SVM libraries to address the knowledge graph link-prediction problem. We present results on standard benchmarks that achieve parity with baseline models, while total wall clock time is measured in minutes rather than hours. The model structure we introduce is closely related to a well-established model with good performance characteristics, and it is elementary. The model we introduce has very few parameters to tune, and default solver settings typically yield satisfactory performance.

With the increasing focus on the large time, cost and energy consumption required to train leading machine learning models, it seems reasonable to ask whether casting nonconvex programs as convex programs could provide a path towards more efficient model training without significant sacrifice in performance.

One benefit of our convex formulation is the ability to add constraints to the program. Future directions could introduce rules about the knowledge graph using this method, as proposed in (Fung et al., 2002). Simple "if P then Q" first order logic propositions, where P and Q are linear predicates (equal, larger that, smaller than) can be represented by polyhedral sets of the form $Ax \leq b$. In this way, conditional information in the form of knowledge sets can be considered during the training process. One could also aim to adapt the model to yield proper embeddings by adding additional constraints such as diagonal dominance to the matrix B rather than just scores.

Another direction we would like to explore is how we can introduce nonlinear data models so our problem does not suffer from the same expressivity limitations as TransE. It may be possible to adapt a model derived from RotatE, for instance.

Finally, the model that we present scales quadratically in the number of entities and relations, and the dimensionality of the representation is a function of the problem. It would be useful to find a representation that has good performance without the quadratic growth in dimension.

References

Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. 2006. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collab-

oratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 301–306. AAAI Press.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Rajarshi Das, Ameya Godbole, Nicholas Monath, Manzil Zaheer, and Andrew McCallum. 2020. Probabilistic case-based reasoning for open-world knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4752–4765, Online. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

G. Fung, O. Mangasarian, and J. Shavlik. 2002. Knowledge-based support vector machine classifiers. In *NIPS*.

Greg Glockner. 2020. Does gurobi support gpus? https://support.gurobi.com/hc/en-us/articles/360012237852-Does-Gurobi-support-GPUs-.

Xavier Glorot, Antoine Bordes, J. Weston, and Yoshua Bengio. 2013. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94:233–259.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, Vancouver, Canada. Association for Computational Linguistics.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Richard A Harshman, Paul E Green, Yoram Wind, and Margaret E Lundy. 1982. A model for the analysis of asymmetric data in marketing research. *Marketing Science*, 1(2):205–242.

R. Herbrich, T. Graepel, and K. Obermayer. 1999. Support vector learning for ordinal regression. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 1, pages 97–102 vol.1.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 133–142, New York, NY, USA. Association for Computing Machinery.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Olvi L. Mangasarian. 2006. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *J. Mach. Learn. Res.*, 7:1517–1530.

G. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41.

Pasquale Minervini, Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. 2015. Scalable learning of entity and predicate embeddings for knowledge graph completion. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 162–167. IEEE.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1955–1961. AAAI Press.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 809–816.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rómer Rosales and Glenn Fung. 2006. Learning sparse metrics via linear programming. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 367–373, New York, NY, USA. Association for Computing Machinery.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. International Conference on Machine Learning (ICML).

Alicia Tsai and Laurent El Ghaoui. 2020. Sparse optimization for unsupervised extractive summarization of long documents with the frank-wolfe algorithm. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 54–62, Online. Association for Computational Linguistics.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1271–1279, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada. Association for Computational Linguistics.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 353–362, New York, NY, USA. Association for Computing Machinery.