Efficient Learning of Entity and Predicate Embeddings for Link Prediction in Knowledge Graphs

Pasquale Minervini, Claudia d'Amato, Nicola Fanizzi, Floriana Esposito

LACAM – Department of Computer Science – Università degli Studi di Bari Aldo Moro, Italy
{firstname.lastname}@uniba.it

Abstract. *Knowledge Graphs* are a widely used formalism for representing knowledge in the Web of Data. We focus on the problem of predicting missing links in large knowledge graphs, so to discover new facts about the world. Recently, *representation learning* models that embed entities and predicates in continuous vector spaces achieved new state-of-the-art results on this problem. A major limitation in these models is that the *training process*, which consists in learning the optimal entity and predicate embeddings for a given knowledge graph, can be very computationally expensive: it may even require days of computations for large knowledge graphs. In this work, by leveraging *adaptive learning rates*, we propose a principled method for reducing the training time by an order of magnitude, while learning more accurate link prediction models. Furthermore, we employ the proposed training method for evaluating a set of novel and scalable models. Our evaluations show significant improvements over state-of-the-art link prediction methods on the WORDNET and FREEBASE datasets.

1 Introduction

Knowledge Graphs (KGs) are graph-structured Knowledge Bases (KBs), where factual knowledge about the world is represented in the form of relationships between entities. They are widely used for representing relational knowledge in a variety of domains, such as citation networks and protein interaction networks. An example of their widespread adoption is the *Linked Open Data* (LOD) Cloud, a set of interlinked KGs such as Freebase [1] and WordNet [2]. As of April 2014, the LOD Cloud was composed by 1,091 interlinked KBs, describing over 8×10^6 entities, and 188×10^6 relationships holding between them [1].

Despite their large size, many KGs are still largely incomplete. For example consider Freebase [2], a core element in the Google Knowledge Vault project [3]: 71% of the persons described in Freebase have no known place of birth, 75% of them have no known nationality, and the coverage for less frequent predicates can be even lower [3].

In this work we focus on the problem of *completing missing links* in large KGs, so to discover new facts about the world. In the literature, this problem is referred to as *link prediction* or *knowledge graph completion*, and has received a considerable attention over the last years [4,5].

[1] State of the LOD Cloud 2014: http://lod-cloud.net/

[2] Publicly available at https://developers.google.com/freebase/data

Recently, *representation learning* [6] models such as the *Translating Embeddings* model (TransE) [7] achieved new state-of-the-art link prediction results on large and Web-scale KGs [4,5]. Such models learn a unique *distributed representation*, or *embedding*, for each entity and predicate in the KG: each entity is represented by a low-dimensional continuous *embedding vector*, and each predicate is represented by an *operation* in the embedding vector space, such as a *translation* (as in [7]) or an *affine transformation* (as in [8]). We refer to these models as *embedding models*, and to the learned distributed representations as *embeddings*.

The embeddings of all entities and predicates in the KG are learned jointly: the learning process consists in minimizing a global loss functional considering the whole KG, by back-propagating the loss to the embeddings [3]. As a consequence, the learned entity and predicate embeddings retain global, structural information about the whole KG, and can be used to serve several kinds of applications. In *link prediction*, the confidence of each candidate edge can be measured as a function of the embeddings of its source entity, its target entity, and its predicate.

A major limitation in embedding models proposed so far, however, is that the learning procedure (i.e. learning the optimal embeddings of all entities and predicates in the KG) can be very time-consuming: it is based on an incremental optimization algorithm that may require days of computation to converge for large KGs [10].

In this work, we propose a novel principled method for significantly reducing the learning time in embedding models, based on *adaptive per-parameter learning rates*. Furthermore, we employ the proposed training method for evaluating a variety of novel embedding models: our evaluations achieves new state-of-the-art link prediction results on the WORDNET and FREEBASE datasets.

2 Basics

RDF Graphs The most widely used formalism for representing knowledge graphs is the W3C *Resource Description Framework* (RDF) [4], a recommended standard for representing knowledge on the Web. An RDF KB, also referred to as *RDF graph*, is a set of *RDF triples* in the form $\langle s, p, o \rangle$, where s, p and o respectively denote the *subject*, the *predicate* and the *object* of the triple: s and o are *entities*, and p is a relation type. Each triple $\langle s, p, o \rangle$ describes a statement, which is interpreted as "*A relationship p holds between entities s and o*".

Example 2.1 (Shakespeare). The statement "*William Shakespeare is an author who wrote Othello and the tragedy Hamlet*" can be expressed by the following RDF triples:

$\langle$`Shakespeare, profession, Author`$\rangle$
$\langle$`Shakespeare, author, Hamlet`$\rangle$
$\langle$`Shakespeare, author, Othello`$\rangle$
$\langle$`Hamlet, genre, Tragedy`$\rangle$

[3] In natural language processing, a similar procedure is used by the word2vec model [9] for learning an unique distributed representation for each word in a corpus of documents.

[4] `http://www.w3.org/TR/rdf11-concepts/`

An RDF graph can be viewed as a *labeled directed multigraph*, where each entity is a vertex, and each RDF triple is represented by a directed edge whose label is a predicate, and emanating from its subject vertex to its object vertex. In RDF KBs, the *Open-World Assumption* holds: a missing triple does not mean that the corresponding statement is false, but rather that its truth value is unknown (it cannot be observed). In the following, given an RDF graph G, we denote as $\mathcal{E}_G$ the set of all entities occurring as subjects or objects in G, and as $\mathcal{R}_G$ the set of all predicates occurring in G:

$$\mathcal{E}_G = \{s \mid \exists \langle s, p, o\rangle \in G\} \cup \{o \mid \exists \langle s, p, o\rangle \in G\},$$
$$\mathcal{R}_G = \{p \mid \exists \langle s, p, o\rangle \in G\}.$$

For instance, in the case of the RDF graph shown in Ex. 2.1, the sets $\mathcal{E}_G$ and $\mathcal{R}_G$ are the following: $\mathcal{E}_G = \{\texttt{Author}, \texttt{Shakespeare}, \texttt{Hamlet}, \texttt{Othello}, \texttt{Tragedy}\}$ and $\mathcal{R}_G = \{\texttt{profession}, \texttt{author}, \texttt{genre}\}$.

Furthermore, we denote as $\mathcal{S}_G = \mathcal{E}_G \times \mathcal{R}_G \times \mathcal{E}_G$ the space of *possible triples* of G, i.e. the set of all triples that can be created by using the entities and predicates in G (note that $G \subseteq \mathcal{S}_G$). We refer to all triples in G as *observed triples*, and to all triples in $\mathcal{S}_G \setminus G$ as *unobserved triples*.

Energy-Based Models Embedding models for KGs can be described in terms of *Energy-Based Models* (EBMs) [11]: EBMs are a versatile and flexible framework for modeling dependencies between variables. In the fields of *representation learning* and *deep learning* [6], EBMs are employed as building blocks for constructing hierarchical models that achieve ground-breaking results in several learning tasks.

A fundamental component in an EBM is a scalar-valued *energy function* (or *scoring function*) $E_\theta(\cdot)$, parametrized by θ, which associates a scalar *energy value* to the configuration of a set of variables. The energy of a configuration of a set of variables is inversely proportional to its probability: *more likely configurations* are associated with *lower energy values*, while *less likely configurations* are associated with *higher energy values*. Several tractable methods have been proposed for learning the parameters of an energy function [11,6]. In particular, the problem of learning the optimal parameters $\hat{\theta}$ can be cast as solving the following optimization problem [11]:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(E_\theta, \mathcal{D}),$$

where Θ is the *parameter space*, and $\mathcal{L}(\cdot)$ is a *loss functional* which measures the quality of the energy function $E_\theta(\cdot)$ on the data $\mathcal{D}$. Intuitively, the loss functional $\mathcal{L}(\cdot)$ assigns a lower loss to energy functions that associate a lower energy (corresponding to a higher probability) to correct answers, and higher energy (corresponding to a lower probability value) to all other incorrect answers.

Energy-Based Models for RDF Graphs As discussed in [12], embedding models for KGs define an *energy distribution* $E_\theta : \mathcal{S}_G \to \mathbb{R}$ over the space of possible triples $\mathcal{S}_G$. For instance, the models proposed in [8,7,13,12] are used for assigning a score $E(\langle s, p, o\rangle)$ to each triple $\langle s, p, o\rangle$ in $\mathcal{S}_G$. In a *link prediction* setting, such models are

Table 1: Energy functions used by several link prediction models, and the corresponding number of parameters: $n_e = |\mathcal{E}_G|$ and $n_r = |\mathcal{R}_G|$ respectively denote the number of entities and predicates in G, and $k, d \in \mathbb{N}$ are user-defined hyper-parameters.

Model	**Energy function** $E(\langle s, p, o\rangle)$	**Parameters Complexity**
Unstructured [12]	$\|\mathbf{e}_s - \mathbf{e}_o\|_2^2, \quad \mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$	$\mathcal{O}(n_e k)$
TransE [7]	$\|(\mathbf{e}_s + \mathbf{e}_p) - \mathbf{e}_o\|_{\{1,2\}}, \quad \mathbf{e}_p \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
SE [8]	$\|\mathbf{W}_{p,1}\mathbf{e}_s - \mathbf{W}_{p,2}\mathbf{e}_o\|_1, \quad \mathbf{W}_{p,i} \in \mathbb{R}^{k\times k}$	$\mathcal{O}\left(n_e k + n_r k^2\right)$
SME lin. [12]	$(\mathbf{W}_1\mathbf{e}_s + \mathbf{W}_2\mathbf{e}_p + \mathbf{b}_1)^T(\mathbf{W}_3\mathbf{e}_o + \mathbf{W}_4\mathbf{e}_p + \mathbf{b}_2)$ $\mathbf{e}_p \in \mathbb{R}^k, \mathbf{W}_i \in \mathbb{R}^{d\times k}, \mathbf{b}_j \in \mathbb{R}^d$	$\mathcal{O}(n_e k + n_r k + dk)$
SME bil. [12]	$\left[(\mathbf{W}_1 \times_3 \mathbf{e}_p)\mathbf{e}_s\right]^T \left[(\mathbf{W}_2 \times_3 \mathbf{e}_p)\mathbf{e}_o\right]$ $\mathbf{e}_p \in \mathbb{R}^k, \mathbf{W}_i \in \mathbb{R}^{d\times k\times k}, \mathbf{b}_j \in \mathbb{R}^d$	$\mathcal{O}\left(n_e k + n_r k + dk^2\right)$
NTN [13]	$\mathbf{u}_p^T f\left(\mathbf{e}_s^T\mathbf{W}_p\mathbf{e}_o + \mathbf{W}_{p,1}\mathbf{e}_s + \mathbf{W}_{p,2}\mathbf{e}_o + \mathbf{b}_p\right)$ $\mathbf{W}_p \in \mathbb{R}^{k\times k\times d}, \mathbf{W}_{p,i} \in \mathbb{R}^{d\times k}, \mathbf{u}_p, \mathbf{b}_p \in \mathbb{R}^d$	$\mathcal{O}\left(n_e k + n_r dk^2\right)$

used as follows. First, the optimal parameters $\hat{\theta}$ of the energy function are learned: the parameters are composed by the embeddings of all entities and predicates in the KG. Then, the energy function $E_{\hat{\theta}}(\cdot)$ is used for ranking unobserved triples: those with lower energy values have a higher probability of representing true statements, and are considered more likely candidates for a completion of the KG.

Consider the RDF graph shown in Ex. 2.1. In such a graph, we prefer learning an energy-based model that assigns a lower energy (a higher probability value) to the triple $\langle \texttt{Othello}, \texttt{genre}, \texttt{Tragedy}\rangle$, which is unobserved but represents the true statement "*Othello is a Tragedy*", and a higher energy (a lower probability value) to other unobserved triples, for example $\langle \texttt{Hamlet}, \texttt{genre}, \texttt{Author}\rangle$.

3 Energy-Based Embedding Models

Several EBMs have been proposed in the literature for addressing the problem of link prediction in KGs [8,14,7,13,12,15]. These models share a fundamental characteristic: they can be used for learning a *distributed representation* (or *embedding*) for each entity and predicate in the KG. We refer to such models as *embedding models*, and denote the distributed representation of an entity or predicate z by adding a subscript to the corresponding vector or matrix representation, as in $\mathbf{e}_z \in \mathbb{R}^k$.

Formally, let G be an RDF graph. For each entity $x \in \mathcal{E}_G$, embedding models learn a continuous vector representation $\mathbf{e}_x \in \mathbb{R}^k$, with $k \in \mathbb{N}$, called the *embedding vector* of x. Similarly, for each predicate $p \in \mathcal{R}_G$, embedding models learn an *operation* on the embedding vector space, characterized by a set of *embedding parameters*. This can be an empty set of parameters, as in the *Unstructured* model proposed in [12]; a translation vector $\mathbf{e}_p \in \mathbb{R}^k$, as in the *Translating Embeddings* model proposed in [7]; or a more complex set of parameters.

The distributed representations of all entities and predicates in G are then used for defining an *energy distribution* $E : \mathcal{S}_G \to \mathbb{R}$ over the space of possible triples of G. In particular, the energy $E(\langle s, p, o\rangle)$ of a triple $\langle s, p, o\rangle$ is defined as a function of the distributed representations of its subject s, its predicate p and its object o.

In Tab. 1, we report the energy functions adopted by several models proposed in the literature. For each model, we report the number of parameters needed for storing the distributed representations of all entities and predicates: $n_e = |\mathcal{E}_G|$ denotes the number of entities in the KG, $n_r = |\mathcal{R}_G|$ denotes the number of predicates, and $k, d \in \mathbb{N}$ are user-defined hyper-parameters. In general, if the number of parameters in a model grows *super-linearly* with the number of entities and predicates in the KG, it becomes increasingly harder for the model to scale to very large and Web-scale KGs.

The Translating Embeddings model Among the models outlined in Tab. 1, the recently proposed *Translating Embeddings* model (TransE) [7] has interesting characteristics, and recently received a considerable attention [4]:

- It achieves more accurate link prediction results than other state-of-the-art methods on several datasets.
- The number of parameters in TransE scales *linearly* in the number of entities n_e and predicates n_r in the KG: this allows TransE to potentially scale to large KGs.

The TransE model is very simple. In TransE, each entity $x \in \mathcal{E}_G$ is represented by its embedding vector $\mathbf{e}_x \in \mathbb{R}^k$, and each predicate $p \in \mathcal{R}_G$ is represented by a (vector) *translation operation* $\mathbf{e}_p \in \mathbb{R}^k$. The energy of a triple $\langle s, p, o \rangle$ is given by the L_1 or L_2 distance between $(\mathbf{e}_s + \mathbf{e}_p)$ and $\mathbf{e}_o$:

$$E(\langle s, p, o \rangle) = \|(\mathbf{e}_s + \mathbf{e}_p) - \mathbf{e}_o\|_{\{1,2\}}.$$

In TransE, all the embedding and translation vectors are learned jointly from the KG by using *Stochastic Gradient Descent*, as discussed in Sect. 4. The number of parameters needed by the TransE model for storing all the embedding and translation vectors is $(n_e k + n_r k)$, a quantity that grows *linearly* with n_e and n_r. For such a reason, TransE can potentially scale to very large and highly-relational KGs [7].

A New Set of Embedding Models In the following, we propose a set of variants of the TransE model, which preserve its scalability properties. Let $d(\mathbf{x}, \mathbf{y})$ be a *dissimilarity* function, from the following set: $d(\mathbf{x}, \mathbf{y}) \in \{\|\mathbf{x} - \mathbf{y}\|_1, \|\mathbf{x} - \mathbf{y}\|_2, -\mathbf{x}^T\mathbf{y}\}$, i.e. chosen from the L_1 and L_2 distance, and the negative inner product. We propose the following embedding models, where each is defined by the corresponding energy function $E(\cdot)$:

- TransE : $E(\langle s, p, o \rangle) = d(\mathbf{e}_s + \mathbf{e}_p, \mathbf{e}_o)$,
- TransE$^+$: $E(\langle s, p, o \rangle) = d(\mathbf{e}_s + \mathbf{e}_{p,1}, \mathbf{e}_o + \mathbf{e}_{p,2})$,
- ScalE : $E(\langle s, p, o \rangle) = d(\mathbf{e}_s \circ \mathbf{e}_p, \mathbf{e}_o)$,
- ScalE$^+$: $E(\langle s, p, o \rangle) = d(\mathbf{e}_s \circ \mathbf{e}_{p,1}, \mathbf{e}_o \circ \mathbf{e}_{p,2})$,

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$ are the embedding vectors of the entities appearing as the subject s and the object o; $\mathbf{e}_{p,\cdot} \in \mathbb{R}^k$ are the embedding parameters of the predicate p, denoting either a *translation* or a *scaling* vector; and $\circ$ denotes the Hadamard (element-wise) product, corresponding to the vector *scaling* operation. The energy function in TransE is the same used in [7], but also allows using the negative inner product as a dissimilarity measure between the (translated) subject and object embedding vectors, if it shows to

Algorithm 1 Learning the model parameters via SGD

Require: Learning rate η, Batch size n, Iterations τ
Ensure: Optimal model parameters (embeddings) $\hat{\theta}$
1: Initialize the model parameters θ_0
2: **for** $t \in \langle 1, \ldots, \tau \rangle$ **do**
3: $\quad \mathbf{e}_x \leftarrow \mathbf{e}_x / \|\mathbf{e}_x\|, \ \forall x \in \mathcal{E}_G$ {Normalize all entity embedding vectors}
4: $\quad T \leftarrow \text{SAMPLEBATCH}(G, n)$ {Sample a batch of observed and corrupted triples}
5: $\quad g_t \leftarrow \nabla \sum_{(y,\tilde{y}) \in T} \left[\gamma + E_\theta(y) - E_\theta(\tilde{y})\right]_+$ {Compute the gradient of the loss}
6: $\quad \Delta_t \leftarrow -\eta g_t$ {Compute the update to model parameters (embeddings)}
7: $\quad \theta_t \leftarrow \theta_{t-1} + \Delta_t$ {Update the model parameters}
8: **end for**
9: **return** θ_τ

improve the performance on the validation set. The TransE$^+$ model generalizes TransE by also translating the object embedding vector $\mathbf{e}_o$.

The ScalE and ScalE$^+$ models are similar to the previous two models, but replace the vector *translation* with a *scaling* operation. The rationale behind ScalE and ScalE$^+$ is the following: scaling the embedding vector of an entity can be seen as *weighting* the (latent) features of such an entity in the embedding vector space.

All proposed models share the same advantages as the TransE model: (i) the required number of parameters is $\mathcal{O}\left(n_e k + n_r k\right)$, which grows *linearly* with n_e and n_r, and (ii) the energy function and its gradient w.r.t. the embedding of entities and predicates can be computed very efficiently, using element-wise vector operations.

4 Improving the Efficiency of the Embeddings Learning Process

In [8,7,12], authors propose a method for jointly learning the embeddings of all entities and predicates in a KG G. The method relies on a *stochastic optimization process*, that iteratively updates the embeddings by reducing the energy of triples in G (observed triples) while increasing the energy of triples in $\mathcal{S}_G \setminus G$ (unobserved triples).

During the learning process, unobserved triples are randomly generated by means of a *corruption process*, which replaces either the subject or the object of each observed triple with another entity in G. More formally, given an observed triple $y \in G$, let $\mathcal{C}_G(y)$ denote the set of all corrupted triples obtained by replacing either its subject or object with another entity in G:

$$\mathcal{C}_G(\langle s, p, o \rangle) = \{\langle \tilde{s}, p, o \rangle \mid \tilde{s} \in \mathcal{E}_G\} \cup \{\langle s, p, \tilde{o} \rangle \mid \tilde{o} \in \mathcal{E}_G\}.$$

The embeddings of all entities and predicates in the KG, which compose the *model parameters*, can be learned by minimizing a *margin-based ranking loss*. More formally, learning the optimal model parameters $\hat{\theta}$, corresponding to all the entity and predicate embeddings, is equivalent to solving the following constrained minimization problem:

$$\begin{aligned} \underset{\theta \in \Theta}{\text{minimize}} \quad & \sum_{y \in G} \sum_{\tilde{y} \in \mathcal{C}_G(y)} \left[\gamma + E_\theta(y) - E_\theta(\tilde{y})\right]_+ \\ \text{subject to} \quad & \forall x \in \mathcal{E}_G : \ \|\mathbf{e}_x\| = 1, \end{aligned} \tag{1}$$

where $[x]_+ = \max\{0, x\}$, and $\gamma \geq 0$ is a hyper-parameter referred to as *margin*. The objective function in Eq. 1 (corresponding to the loss functional $\mathcal{L}(\cdot)$ discussed in Sect. 2) enforces the energy of observed triples to be lower than the energy of unobserved triples. The constraints in the optimization problem prevent the training process to trivially solve the problem by increasing the entity embedding norms.

Stochastic Gradient Descent In the literature, the constrained loss minimization problem in Eq. 1 is solved using *Stochastic Gradient Descent* (SGD) in mini-batch mode, as summarized in Alg. 1. On each iteration, the algorithm samples a batch of triples from the knowledge graph G. Batches are obtained by first randomly permuting all triples in G, partitioning them into n_b batches of the same size, and then iterating over such batches. A single pass over all triples in G is called an *epoch*. Then, for each triple y in the batch, the algorithm generates a *corrupted* triple $\tilde{y}$ uniformly sampled from $\mathcal{C}_G(y)$: this leads to a set T of observed/corrupted pairs of triples $\langle y, \tilde{y} \rangle$. The observed/corrupted triple pairs are used for computing the gradient of the objective (loss) function in Eq. 1 w.r.t. the current model parameters θ. Finally, θ is updated in the steepest descent direction of the objective function. This procedure is repeated until convergence.

The main drawback of SGD is that it requires an initial, careful tuning of the global learning rate η, which is then used for updating all model parameters, regardless of their peculiarities. However, if an entity $x \in \mathcal{E}_G$ occurs in a limited number of triples in G, the corresponding embedding vector $\mathbf{e}_x \in \mathbb{R}^k$ will be updated less often, and it will require a much longer time to be learned. For such a reason, SGD may be very time-consuming and slow to converge. For instance, it was reported in [10] that learning the optimal embeddings in TransE may require days of computation for large KGs.

A possible solution to this problem consists in associating *smaller learning rates* to parameters updated more often, such as the embedding vectors of entities appearing more frequently, and *larger learning rates* to parameters updated less often.

Adaptive Learning Rates In order to reduce the time required for learning all entity and predicate embeddings, in this work we propose leveraging *Adaptive Per-Parameter Learning Rates*. While SGD uses a global, fixed learning rate η for updating all parameters, we propose relying on methods for estimating the *optimal* learning rate for each parameter, while still being tractable for learning very large models.

We consider two highly-scalable criteria for selecting the optimal learning rates, namely the *Momentum method* [16] and *AdaGrad* [17]: they specify alternate ways of computing the parameters update Δ_t, defined in Alg. 1 on line 6.

Momentum Method The idea behind this method is to accelerate the progress along dimensions where the sign of the gradient does not change, while slowing the progress along dimensions where the sign of the gradient continues to change. The new update rule is defined as follows:

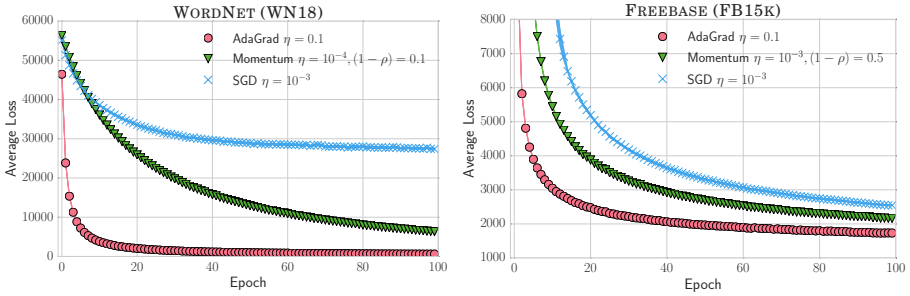
$$\Delta_t \leftarrow \rho \Delta_{t-1} - \eta_m g_t,$$

where $\eta_m \in \mathbb{R}$ is a user-defined hyper-parameter.

Table 2: Statistics for the datasets used in the **Link Prediction** and **Triple Classification** tasks.

Dataset	Entities	Predicates	Train. Triples	Valid. Triples	Test Triples
FREEBASE (FB15K) [7]	14,951	1,345	483,142	50,000	59,071
WORDNET (WN18) [7]	40,943	18	141,442	5,000	5,000

Fig. 1: Average loss (the lower, the better) across 10 TransE parameters learning tasks on the WORDNET (WN18) and FREEBASE (FB15K) datasets, using the optimal TransE settings reported in [7]. For each optimization method, we report the hyper-parameter values that achieve the lowest average loss after 100 epochs, and the corresponding average loss values.



AdaGrad This method is based on the idea that the learning rate of each parameter should grow with the inverse of gradient magnitudes. The update rule in AdaGrad is:

$$\Delta_t \leftarrow -\frac{\eta_a}{\sqrt{\sum_{j=1}^{t} g_j^2}} g_t,$$

where $\eta_a \in \mathbb{R}$ is a user-defined hyper-parameter. AdaGrad adds nearly no complexity, it has very strong convergence guarantees [17], and it has shown remarkable results on large scale learning tasks in distributed environments [18].

5 Empirical Evaluations

This section is organized as follows. In Sect. 5.1 we describe experimental settings, datasets and evaluation metrics. In Sect. 5.2, we show that *adaptive learning rates* sensibly improve both the efficiency of the learning process, and the predictive accuracy of embedding models. In Sect. 5.3, we empirically evaluate the novel embedding models proposed in Sect. 3, by training them using adaptive learning rates.

5.1 Experimental Settings

Link Prediction In these experiments, we used the metrics proposed in [7] for evaluating the *rank* of each test triple. In particular, for each test triple $\langle s, p, o \rangle$, its object

Table 3: **Link Prediction** Results: Test performance of several link prediction methods on the WORDNET and FREEBASE datasets. Results show the MEAN RANK (the lower, the better) and HITS@10 (the higher, the better) for both the RAW and the FILTERED settings [7].

Dataset	WORDNET (WN18)				FREEBASE (FB15K)			
Metric	MEAN RANK		HITS@10 (%)		MEAN RANK		HITS@10 (%)	
	RAW	FILT.	RAW	FILT.	RAW	FILT.	RAW	FILT.
Unstructured [12]	315	304	35.3	38.2	1,074	979	4.5	6.3
RESCAL [19]	1,180	1,163	37.2	52.8	828	683	28.4	44.1
SE [8]	1,011	985	68.5	80.5	273	162	28.8	39.8
SME lin. [12]	545	533	65.1	74.1	274	154	30.7	40.8
SME bil. [12]	526	509	54.7	61.3	284	158	31.3	41.3
LFM [14]	469	456	71.4	81.6	283	164	26.0	33.1
TransE [7]	263	251	75.4	89.2	243	125	34.9	47.1
TransE^A	**169**	**158**	**80.5**	**93.5**	**189**	**73**	**44.0**	**60.1**

o is replaced with every entity $\tilde{o} \in \mathcal{E}_G$ in the KG G in turn, generating a set of *corrupted* triples in the form $\langle s, p, \tilde{o} \rangle$. The energies of corrupted triples are first computed by the model, then sorted in ascending order, and used to compute the rank of the correct triple. This procedure is repeated by corrupting the subject. Aggregated over all the test triples, this procedure leads to the following two metrics: the *averaged rank*, denoted by MEAN RANK, and the *proportion of ranks not larger than* 10, denoted by HITS@10. This is referred to as the RAW setting.

In the FILTERED setting, corrupted triples that exist in either the training, validation or test set were removed before computing the rank of each triple. In both settings, a lower MEAN RANK is better, while a higher HITS@10 is better.

5.2 Evaluation of Adaptive Learning Rates

Learning Time For comparing Momentum and AdaGrad with SGD on the task of solving the optimization problem in Eq. 1, we empirically evaluated such methods on the task of learning the parameters in TransE on WN18 and FB15K, using the optimal hyper-parameter settings reported in [7]: $k = 20$, $\gamma = 2$, $d = L_1$ for WN18, and $k = 50$, $\gamma = 1$, $d = L_1$ for FB15K. Following the evaluation protocol in [20], we compared the optimization methods by using a large grid of hyper-parameters. Let $\mathcal{G}_\eta = \{10^{-6}, 10^{-5}, \ldots, 10^{1}\}$ and $\mathcal{G}_\rho = \{1-10^{-4}, 1-10^{-3}, \ldots, 1-10^{-1}, 0.5\}$. The grids of hyper-parameters considered for each of the optimization methods were the following:

- **SGD** and **AdaGrad:** rate $\eta, \eta_a \in \mathcal{G}_\eta$.
- **Momentum:** rate $\eta_m \in \mathcal{G}_\eta$, decay rate $\rho \in \mathcal{G}_\rho$.

For each possible combination of optimization method and hyper-parameter values, we performed an evaluation consisting in 10 learning tasks, each using a different random initialization of the model parameters.

Fig. 1 shows the behavior of the loss functional for each of the optimization methods, using the best hyper-parameter settings selected after 100 training epochs. We can

Table 4: **Link Prediction** Results: Test performance of the embedding models proposed in Sect. 3 on the WORDNET and FREEBASE datasets. Results show the MEAN RANK (the lower, the better) and HITS@10 (the higher, the better) [7].

Dataset	WORDNET (WN18)				FREEBASE (FB15K)			
Metric	MEAN RANK		HITS@10 (%)		MEAN RANK		HITS@10 (%)	
	RAW	FILT.	RAW	FILT.	RAW	FILT.	RAW	FILT.
TransE, from [7] SGD	263	251	75.4	89.2	243	125	34.9	47.1
TransE AdaGrad	161	150	80.5	93.5	183	63	47.9	68.2
TransE$^+$ (Sect. 3) AdaGrad	**159**	**148**	79.6	92.6	196	78	44.9	62.4
ScalE (Sect. 3) AdaGrad	187	174	82.7	94.5	194	62	**49.8**	**73.0**
ScalE$^+$ (Sect. 3) AdaGrad	298	287	**83.7**	**95.5**	**185**	**59**	50.0	71.5

immediately observe that, for both WORDNET (WN18) and FREEBASE (FB15K), AdaGrad (with rate $\eta_a = 0.1$) yields sensibly lower values of the loss functional than SGD and Momentum, even after very few iterations (< 10 epochs). The duration of each epoch was similar in all methods: each epoch took approx. **1.6 seconds** in WORDNET (WN18), and approx. **4.6 seconds** in FREEBASE (FB15K) on a single i7 CPU.

Quality of Learned Models We also measured the *quality* of models learned by AdaGrad, in terms of the MEAN RANK and HITS@10 metrics, in comparison with SGD. For this purpose, we trained TransE using AdaGrad (instead of SGD) with $\eta_a = 0.1$ for 100 epochs, denoted as TransE^A, and compared it with results obtained with TransE from the literature on *Link Prediction* tasks on the WORDNET and FREEBASE datasets. Hyper-parameters were selected according to the performance on the validation set, using the same grids of hyper-parameters used for TransE in [7] for the *Link Prediction* tasks. The results obtained by TransE^A, in comparison with state-of-the-art results reported in [7], are shown in Tab. 3. Despite the sensibly lower number of training iterations (we trained the model using AdaGrad for only 100 epochs, while in [7] TransE was trained using SGD for 1,000 epochs), TransE^A yields more accurate link prediction models (i.e. with lower MEAN RANK and higher HITS@10 values) than every other prediction model in the comparison.

5.3 Evaluation of the Proposed Embedding Models

In this section, we evaluate the embedding models inspired by TransE and proposed in Sect. 3: ScalE, TransE$^+$ and ScalE$^+$. Model hyper-parameters were selected according to the performance on the validation set. In the following experiments, we considered a wider grid of hyper-parameters: in particular, we selected the embedding vector dimension k in $\{20, 50, 100, 200, 300\}$, the *dissimilarity* function $d(\mathbf{x}, \mathbf{y})$ in $\{\|\mathbf{x} - \mathbf{y}\|_1, \|\mathbf{x} - \mathbf{y}\|_2, -\mathbf{x}^T\mathbf{y}\}$, and the margin γ in $\{1, 2, 5, 10\}$. All models were trained using AdaGrad, with $\eta_a = 0.1$, for only 100 epochs. The reduced training time enabled us to experiment with a wider range of hyper-parameters in comparison with related works in literature [7].

Results are summarized in Tab. 4. We can see that, despite their very different geometric interpretations, all of the embedding models proposed in Sect. 3 achieve sensi-

bly higher results in terms of HITS@10 in comparison with every other link prediction models outlined in Sect. 3. An explanation is that both TransE [7] and the proposed models TransE$^+$, ScalE and ScalE$^+$ have a *limited model capacity* (or complexity) in comparison with other models. For such a reason, they are *less prone to underfitting* for lack of training instances than other more expressive link prediction models, such as RESCAL [19], SME [12] and NTN [13].

In each experiment, the proposed models ScalE and ScalE$^+$ always improve over TransE in terms of HITS@10. We can clearly see that, by leveraging: (i) *Adaptive* learning rates, and (ii) The proposed embedding models ScalE and ScalE$^+$, we were able to achieve a record **95.5%** HITS@10 on WORDNET, and a **73.0%** HITS@10 on FREEBASE. These results are sensibly higher than state-of-the-art results reported in [7]. It is also remarkable that, during learning, the proposed method required a much lower learning time (100 epochs, approx. **30 minutes** on FREEBASE, on a single CPU) in comparison with [7] (1,000 epochs, and careful learning rate tuning).

A significantly lower training time – from days, as reported by [10], to minutes – can sensibly improve the applicability of embedding models for knowledge graphs in the Web of Data.

6 Conclusions

We focused on the problem of link prediction in Knowledge Graphs. Recently, *embedding models* like the TransE [7] model achieved new state-of-the-art link prediction results, while showing the potential to scale to very large KGs.

In this paper, we proposed a method for sensibly reducing the learning time in embedding models based on *adaptive learning rate selection*, and proposed a set of new models with interesting scalability properties. We extensively evaluated the proposed methods in several experiments on real world large datasets. Our results show a significant improvement over state-of-the-art link prediction methods, while significantly reducing the required training time by an order of magnitude.

The contributions in this paper sensibly improve both the effectiveness and applicability of embedding models on large and Web-scale KGs. Source code and datasets for reproducing the empirical evaluations discussed in this paper are available on-line [5].

References

1. K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008*, J. T. Wang, Ed. ACM, 2008, pp. 1247–1250.
2. G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
3. X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, S. A. Macskassy *et al.*, Eds. ACM, 2014, pp. 601–610.

[5] Source code and datasets: `https://github.com/pminervini/DeepKGC`

4. A. Bordes and E. Gabrilovich, "Constructing and mining web-scale knowledge graphs: KDD 2014 tutorial," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, 2014, p. 1967.
5. ——, "Constructing and mining web-scale knowledge graphs: WWW 2015 Tutorial," in *Proceedings of the 24th International Conference on World Wide Web, WWW 2015*, 2015, p. 1523.
6. Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.
7. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26*, C. Burges *et al.*, Eds. Curran Associates, Inc., 2013, pp. 2787–2795.
8. A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, W. Burgard *et al.*, Eds. AAAI Press, 2011.
9. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 3111–3119.
10. K. Chang, W. Yih, B. Yang, and C. Meek, "Typed tensor decomposition of knowledge bases for relation extraction," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, A. Moschitti *et al.*, Eds. ACL, 2014, pp. 1568–1579.
11. Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," in *Predicting Structured Data*, G. Bakir *et al.*, Eds. MIT Press, 2006.
12. A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.
13. R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems 26*, C. Burges *et al.*, Eds. Curran Associates, Inc., 2013, pp. 926–934.
14. R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski, "A latent factor model for highly multi-relational data," in *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, Eds. Curran Associates, Inc., 2012, pp. 3167–3175.
15. Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, C. E. Brodley *et al.*, Eds. AAAI Press, 2014, pp. 1112–1119.
16. D. E. Rumelhart, G. E. Hinton, and R. J. Wilson, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
17. J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
18. J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, Eds. Curran Associates, Inc., 2012, pp. 1223–1231.
19. M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, L. Getoor *et al.*, Eds. Omnipress, 2011, pp. 809–816.
20. T. Schaul, I. Antonoglou, and D. Silver, "Unit tests for stochastic optimization," in *International Conference on Learning Representations*, 2014.