Knowledge-Enhanced Top-K Recommendation in Poincaré Ball

Chen Ma [*1], **Liheng Ma** [*1], **Yingxue Zhang** [2], **Haolun Wu** [*1], **Xue Liu** [1] and **Mark Coates** [1]

[1] McGill University, [2] Huawei Noah's Ark Lab Montreal
chen.ma2@mail.mcgill.ca, liheng.ma@mail.mcgill.ca, yingxue.zhang@huawei.com,
haolun.wu@mail.mcgill.ca, xueliu@cs.mcgill.ca, mark.coates@mcgill.ca

Abstract

Personalized recommender systems are increasingly important as more content and services become available and users struggle to identify what might interest them. Thanks to the ability for providing rich information, knowledge graphs (KGs) are being incorporated to enhance the recommendation performance and interpretability. To effectively make use of the knowledge graph, we propose a recommendation model in the hyperbolic space, which facilitates the learning of the hierarchical structure of knowledge graphs. Furthermore, a hyperbolic attention network is employed to determine the relative importances of neighboring entities of a certain item. In addition, we propose an adaptive and fine-grained regularization mechanism to adaptively regularize items and their neighboring representations. Via a comparison using three real-world datasets with state-of-the-art methods, we show that the proposed model outperforms the best existing models by 2-16% in terms of NDCG@K on Top-K recommendation.

Introduction

With the rapid growth of Internet services and mobile devices, personalized recommender systems play an increasingly important role in modern society. They can reduce information overload and help satisfy diverse service demands. Such systems bring significant benefits to at least two parties. They can: (i) help users easily discover products from millions of candidates, and (ii) create opportunities for product providers to increase revenue.

To provide a more accurate and interpretable recommendation service, knowledge graphs (KGs) are being incorporated into recommender systems. A KG is a heterogeneous graph, where nodes function as entities and edges represent relations between the entities. This is an effective data structure to model relational data, e.g., two movies directed by the same director. Several recent works have integrated KGs into the recommendation model, and the approaches can be divided into two branches: path-based (Hu et al. 2018; Wang et al. 2018) and regularization-based (Zhang et al. 2016; Wang et al. 2019c). Path-based methods extract paths from

the KG that carry the high-order connectivity information and feed these paths into the predictive model. To handle the large number of paths between two nodes, researchers have either applied path selection algorithms to select prominent paths or defined meta-path patterns to constrain the paths. By contrast, regularization-based methods devise additional loss terms that capture the KG structure and use these to regularize the recommender model learning.

Although many effective models have been proposed, we argue that there are still several avenues for enhancing performance. First, previous works learn the KG representations in the Euclidean space. As has been observed in other application domains, this may not effectively capture the hierarchical structure that is known to exist within KGs (Sala et al. 2018). Second, methods like CKE (Zhang et al. 2016), CFKG (Ai et al. 2018), and RippleNet (Wang et al. 2018) do not distinguish between neighboring entities, adjusting according to their relative importance and informativeness, when learning the representation of each entity. This may lead to undesirable blurring of information from relations in the KG and an incomplete understanding of an entity. Third, all the regularization-based methods adopt a fixed hyper-parameter. We argue that the regularization degree should be adaptive, taking on different values for different entities according to the relevance and value of the information from the knowledge graph. Furthermore, different training phases may need different magnitudes of regularization power values, so the hyper-parameter values should evolve during training.

To tackle the aforementioned problems, we propose a knowledge-enhanced recommendation model in the hyperbolic space, namely *Hyper-Know*, to tackle the top-K recommendation task. In particular, we map the entity and relation embeddings of the KG as well as user and item embeddings to the Poincaré ball model. This allows us to capture the hierarchical structure in the KG. We incorporate an attention model in the hyperbolic space, and use the Einstein midpoint for aggregation, in order to form a representation of the neighborhood of each item in the knowledge graph. We then use a regularization term to encourage the representation of an item to remain close to the representation of its neighborhood (in the hyperbolic space). This transfers the relational and structural information from the knowledge graph to the recommendation model. To adaptively control

the regularization effect, we model the learning of adaptive and fine-grained regularization factors as a bilevel (inner and outer) optimization problem (Sinha, Malo, and Deb 2018). We build a proxy function to explicitly link the learning of the regularization related parameters with the outer objective function. We extensively evaluate our model on three real-world datasets, comparing it with many state-of-the-art methods using a variety of performance validation metrics. The experimental results not only demonstrate the improvements of our model over other baselines but also show the effectiveness of the proposed components.

To summarize, the major contributions of this paper are:

- To model the hierarchical structure of KG, we map the entity and relation embeddings of the KG into the Poincaré ball along with user and item embeddings. To the best of our knowledge, ours is the first work to consider knowledge-enhanced recommendation in the hyperbolic space.

- To transfer the knowledge from the KG to the recommendation model, we incorporate hyperbolic attention and use the Einstein midpoint to aggregate the neighboring entities of an item to form a neighborhood representation.

- To learn the adaptive regularization factors, we cast the learning process as a bilevel optimization problem and build a proxy function to explicitly update the regularization-related parameters.

- Experiments on three real-world datasets show that Hyper-Know significantly outperforms the state-of-the-art methods for the top-K recommendation task.

Related Work

General Recommendation

Early recommendation studies largely focused on explicit feedback (Sarwar et al. 2001; Koren 2008). The recent research focus is shifting towards implicit data (Tran et al. 2018). Collaborative filtering (CF) with implicit feedback is usually treated as a Top-K item recommendation task, where the goal is to recommend a list of items to users that users may be interested in. It is more practical and challenging (Pan et al. 2008), and accords more closely with the real-world recommendation scenario. Early works mostly rely on matrix factorization techniques (Hu, Koren, and Volinsky 2008; Rendle et al. 2009) to learn latent features of users and items. Due to their ability to learn salient representations, (deep) neural network-based methods (He et al. 2017; Sun et al. 2019; Ma, Kang, and Liu 2019) are also adopted. Autoencoder-based methods (Wu et al. 2016; Ma et al. 2018, 2019) have also been proposed for Top-K recommendation. In (Lian et al. 2018; Xue et al. 2017), deep learning techniques are used to boost the traditional matrix factorization and factorization machine methods. Recently, some methods are also conducted in the hyperbolic space. HyperML (Tran et al. 2020) conducts metric learning in the hyperbolic space and outperforms Euclidean counterparts. Feng et al. (2020) propose to tackle the next Point-of-Interest recommendation task in the hyperbolic space.

Knowledge Graph Enhanced Recommendation

Knowledge graphs (KGs) are an important means to represent side information of recommender systems and have proven to be helpful to improve the recommendation performance. For example, Zhang et al. (2016) propose to apply the TransR method (Lin et al. 2015) to learn the KG representation as well as the item embeddings in the KG. Ai et al. (2018) integrate users and items with the KG and jointly learn the recommendation and KG part. Wang et al. (2019a) propose a multi-task feature learning approach for knowledge graph enhanced recommendation, where these two parts are connected with a cross-and-compress unit to transfer knowledge and share regularization of items. Another track of research tries to perform propagation over the KG to assist in recommendation. Specifically, RippleNet (Wang et al. 2018) extends the user's interests along KG links to discover her potential interests by introducing preference propagation, which automatically propagates users' potential preferences and explores their hierarchical interests in the KG. KPRN (Wang et al. 2019d) constructs the extracted path sequence with both the entity embedding and the relation embedding. These paths are encoded with an LSTM layer and the preferences for items in each path are predicted through fully-connected layers. KGCN (Wang et al. 2019b) studies the utilization of Graph Convolutional Networks (GCNs) for computing embeddings of items via propagation among their neighbors in the KG. Recently, KGAT (Wang et al. 2019c) recursively performs propagation over the KG via a graph attention mechanism that refines entity embeddings. Several subsequent works (Chen et al. 2020; Wang et al. 2020) focus on optimizing the negative sampling procedure in knowledge-enhanced recommendation. In this paper, we report results for our proposed method using a vanilla negative sampling strategy, so that we can focus on the performance impact of the novel aspects: learning in the hyperbolic space, using hyperbolic attention with Einstein midpoint aggregation, and introducing adaptive regularization. But the advanced negative sampling strategies can also be incorporated into our proposed method to provide a further performance improvement.

Our proposed model distinguishes itself from previous models by learning knowledge-enhanced recommendation in the Poincaré ball model. In addition, we employ a hyperbolic attention model in the hyperbolic space to assign different degrees of importance to the neighboring entities of a certain item. We introduce a bilevel optimization formulation of the learning task to achieve an adaptive regularization mechanism that controls the regularization effect.

Preliminaries

Problem Formulation

The knowledge-based recommendation considered in this paper takes as inputs the user implicit feedback and the item knowledge graph. The implicit feedback is represented by a number of user-item pairs $\mathcal{D} = (u, v) \subseteq \mathcal{U} \times \mathcal{I}$, where $\mathcal{U}$ is the user set and $\mathcal{I}$ is the item set. The item knowledge graph $\mathcal{G} = (h, r, t) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ can be formulated as a set of

triples, each consisting of a relation $r \in \mathcal{R}$ and two entities $h, t \in \mathcal{E}$, referred to as the *head* and *tail* of the triple.

Then the top-K recommendation task in this paper is formulated as: given the training item set $\mathcal{S}_u$ of user u, and the non-empty test item set $\mathcal{T}_u$ (requiring that $\mathcal{S}_u \cup \mathcal{T}_u = \mathcal{D}_u$ and $\mathcal{S}_u \cap \mathcal{T}_u = \emptyset$) of user i, the model must recommend an ordered set of items $\mathcal{X}_u$ such that $|\mathcal{X}_u| \leq K$ and $\mathcal{X}_u \cap \mathcal{S}_u = \emptyset$. Then the recommendation quality is evaluated by a matching score between $\mathcal{T}_u$ and $\mathcal{X}_u$, such as Recall@K.

Hyperbolic Geometry of the Poincaré Ball

The Poincaré ball model is one of five isometric models of hyperbolic geometry (Cannon et al. 1997), which is a non-Euclidean geometry with constant negative curvature. Formally, the Poincaré ball $\left(\mathbb{B}_c^d, g^{\mathbb{B}}\right)$ of radius $1/\sqrt{c}, c > 0$ is a d-dimensional manifold $\mathbb{B}_c^d = \left\{\mathbf{x} \in \mathbb{R}^d : c\|\mathbf{x}\|^2 < 1\right\}$ equipped with the Riemannian metric $g^{\mathbb{B}}$ which is conformal to the Euclidean metric $g^{\mathbb{E}} = \mathbf{I}_d$ with the conformal factor $\lambda_{\mathbf{x}}^c = 2/\left(1 - c\|\mathbf{x}\|^2\right)$, i.e., $g^{\mathbb{B}} = (\lambda_{\mathbf{x}}^c)^2 g^{\mathbb{E}}$. The distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{B}_c^d$ is measured along a geodesic (i.e. a shortest path between the points) and is given by:

$$d_{\mathbb{B}}(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\,\|-\mathbf{x} \oplus_c \mathbf{y}\|\right), \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean norm and $\oplus_c$ represents Möbius addition (Ganea, Bécigneul, and Hofmann 2018):

$$\mathbf{x} \oplus_c \mathbf{y} = \frac{\left(1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c\|\mathbf{y}\|^2\right)\mathbf{x} + \left(1 - c\|\mathbf{x}\|^2\right)\mathbf{y}}{1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \quad (2)$$

Methodology

In this section, we introduce the proposed model, *Hyper-Know*, which integrates the knowledge graph with the recommendation task in the hyperbolic space. We first introduce the user preference learning in the hyperbolic space. Then we illustrate the hyperbolic attention mechanism that is used to distinguish items' neighboring entities in the knowledge graph. We next explain how to adaptively learn the recommendation objective and knowledge graph by a bilevel optimization formulation. Lastly we introduce the training and prediction procedure of the proposed model.

Learning User Preference

User preference modeling lies at the core of recommender systems. Recently, distance metric learning has been widely applied to measure the user preference on items, yielding substantial performance gains (Hsieh et al. 2017). In this approach, the distance $d_{\mathbb{B}}(\mathbf{u}, \mathbf{v})$ between user u and item v is used to measure the user preference on a certain item. To learn the user preference, we apply the Bayesian Personalized Ranking loss (Rendle et al. 2009) to capture the pairwise preference of a user u for an item v that the user has accessed compared to a randomly sampled item v':

$$\mathcal{L}_R(u, v, v'; \Theta) = -\ln\sigma\left(d_{\mathbb{B}}(\mathbf{u}, \mathbf{v}') - d_{\mathbb{B}}(\mathbf{u}, \mathbf{v})\right), \quad (3)$$

where $\mathbf{u}, \mathbf{v}$, and $\mathbf{v}' \in \mathbb{B}_c^d$, σ is the sigmoid function, and d is the dimension of the manifold. Θ represents the parameters of the recommender model.

Regularizing Neighboring Entities

Knowledge graphs (KGs), consisting of (head entity, relationship, tail entity) triples, are efficient data structures for representing factual knowledge and are widely used in applications such as question answering (Zhang et al. 2018). Recently, KGs have been applied in recommender systems to not only enhance the recommendation performance but also provide interpretable recommendation results.

To effectively exploit KGs in recommender systems, we treat them as relational inductive biases (Battaglia et al. 2018) between items. During the learning process, the relations in the KG can be used as regularizers; if two items link to a common entity or multiple common entities in the KG which suggests that a user might have similar preferences for the items. However, an item can link to multiple entities in the KG and the relative importance of different entities can differ greatly. Moreover, the entities can contribute in different ways to the description of the item. This motivates us to propose an attention mechanism in the Poincaré ball model.

Considering an item v, we use $\mathcal{N}_v = \{(v, r, t)|(v, r, t) \in \mathcal{G}\}$ to denote the set of neighboring triples for which v is the head entity. Then we apply a TransE-style (Bordes et al. 2013) scoring function ($\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$) to calculate the matching score between an item and its neighboring entity in $\mathcal{N}_v$:

$$\alpha(v, t) = \exp\left(-d_{\mathbb{B}}(\mathbf{v} \oplus_c \mathbf{r}, \mathbf{t})\right). \quad (4)$$

The usual way to aggregate multiple attentions in the Euclidean space is weighted midpoint aggregation. The corresponding operation in the hyperbolic space is not immediately obvious, but fortunately, the extension to hyperbolic space does exist in the form of the *Einstein midpoint*. It has a simple form in the Klein disk model (Cannon et al. 1997):

$$\mathbf{n}_v = f_{\mathbb{K} \to \mathbb{B}}\left(\sum_{t \in \mathcal{N}_v} \frac{\alpha(v, t)\gamma(\mathbf{t})}{\sum_{t'} \alpha(v, t')\gamma(\mathbf{t}')} f_{\mathbb{B} \to \mathbb{K}}(\mathbf{t})\right), \quad (5)$$

where the elements of $\gamma(\mathbf{t}) = \frac{1}{\sqrt{1 - c\|\mathbf{t}\|^2}}$ are the Lorentz factors and $f_{\mathbb{B} \to \mathbb{K}}(\cdot)$ is the function to transform the coordinates from the Poincaré ball model to the Klein disk model. The Klein model is supported on the same space as the Poincaré ball, but the same point has different coordinates in each model. Let $\mathbf{x}_{\mathbb{B}}$ and $\mathbf{x}_{\mathbb{K}}$ denote the coordinates of the same point $\mathbf{x}$ in the Poincare and Klein models correspondingly. Then the following transition formulas hold:

$$\begin{aligned} \mathbf{x}_{\mathbb{K}} &= f_{\mathbb{B} \to \mathbb{K}}(\mathbf{x}_{\mathbb{B}}) = \frac{2\mathbf{x}_{\mathbb{B}}}{1 + c\|\mathbf{x}_{\mathbb{B}}\|^2}, \\ \mathbf{x}_{\mathbb{B}} &= f_{\mathbb{K} \to \mathbb{B}}(\mathbf{x}_{\mathbb{K}}) = \frac{\mathbf{x}_{\mathbb{K}}}{1 + \sqrt{1 - c\|\mathbf{x}_{\mathbb{K}}\|^2}}. \end{aligned} \quad (6)$$

We call $\mathbf{n}_v$ in (5) the neighborhood representation of item v. During the training process we add a regularizing term that encourages the neighborhood representation $\mathbf{n}_v$ to be close to the item's representation $\mathbf{v}$. The goal is to transfer the inductive bias in KG to the item representation:

$$\mathcal{L}_K(v; \Theta) = d_{\mathbb{B}}(\mathbf{v}, \mathbf{n}_v). \quad (7)$$

Combining with the user preference learning objective $\mathcal{L}_R$, the overall knowledge-enhanced objective can be:

$$\mathcal{L}(u, v, v'; \Theta) = \mathcal{L}_R(u, v, v'; \Theta) + \beta\mathcal{L}_K(v; \Theta), \quad (8)$$

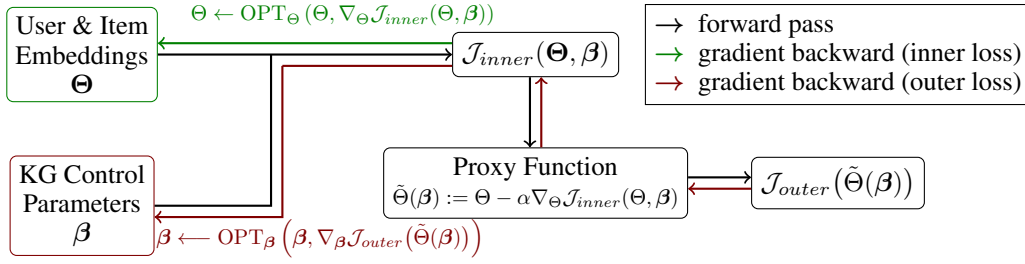where β is to balance the effect from the KG.

Figure 1: The procedure of the bilevel optimization to control the learning of fine-grained parameters $\boldsymbol{\beta}$.

Adaptive and Fine-grained Regularization

Previous works (Wang et al. 2019c, 2018; Zhang et al. 2016) that derive information from a KG in the recommender setting use a single and fixed number for β in Eq. 8 to train the overall objective. However, employing a single fixed value for β can have several drawbacks. First, different datasets may require different impact levels of regularization from KGs. Treating β as a fixed value requires extra hyperparameter searching procedure for each dataset to better realize the power of KGs. Second, different items may need different degrees of regularization. Using the same value for every item would limit the achievable performance improvement that can be derived from the KG information. Third, in different training phases, the model may need different magnitudes of regularization power.

To address the problems outlined above, we propose an adaptive regularization scheme to apply different strengths of regularization to each item and to adjust the strength throughout training. We formulate Eq. 8 as:

$$\mathcal{L}(u, v, v'; \Theta, \boldsymbol{\beta}) = \mathcal{L}_R(u, v, v'; \Theta) + \sigma(\beta_v)\mathcal{L}_K(v; \Theta), \tag{9}$$

where β_v is v-th value of $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{I}|}$ and $\sigma(\beta_v) \in (0,1)$ where $\sigma(\cdot)$ is the sigmoid function. Unfortunately, directly minimizing this objective function is not able to achieve the desired purpose of adaptively controlling the regularization. The reason is that, considering β_v explicitly appears in the loss function, constantly decreasing the value of β_v is the straightforward way to minimize the loss. As a consequence, instead of reaching optimal values for the model, all β_v will end up with very small values close to zero, leading to unsatisfactory results.

To tackle the above problem, we model the learning of recommendation models and the adaptive regularization of KG as a bilevel optimization problem (Colson, Marcotte, and Savard 2007):

$$\min_{\boldsymbol{\beta}} \mathcal{J}_{outer}\left(\Theta^*(\boldsymbol{\beta})\right) := \sum_{(u,v) \in \mathcal{D} \wedge (u,v') \notin \mathcal{D}} \mathcal{L}_R\left(u, v, v'; \Theta^*(\boldsymbol{\beta})\right)$$
$$\text{s.t. } \Theta^*(\boldsymbol{\beta}) = \operatorname*{argmin}_{\Theta} \mathcal{J}_{inner}(\Theta, \boldsymbol{\beta})$$
$$:= \sum_{(u,v) \in \mathcal{D} \wedge (u,v') \notin \mathcal{D}} \mathcal{L}(u, v, v'; \Theta, \boldsymbol{\beta}). \tag{10}$$

Here Θ contains the model parameters $\mathbf{u}$, $\mathbf{v}$, $\mathbf{r}$ and $\mathbf{t}$. The objective function $\mathcal{J}_{inner}$ attempts to minimize $\mathcal{L}$ with respect to Θ with $\boldsymbol{\beta}$ fixed. Meanwhile, the objective function $\mathcal{J}_{outer}$

optimizes $\mathcal{L}_R$ with respect to $\boldsymbol{\beta}$ through $\Theta^*(\boldsymbol{\beta})$, considering $\Theta^*(\boldsymbol{\beta})$ as a function of $\boldsymbol{\beta}$.

Algorithm 1: Iterative Training Procedure

Initialize optimizers OPT_Θ and $\text{OPT}_{\boldsymbol{\beta}}$;
while *not converged* **do**
  Θ Update (fix $\boldsymbol{\beta}^t$):
    $\Theta^{t+1} \longleftarrow \text{OPT}_\Theta\left(\Theta^t, \nabla_{\Theta^t}\mathcal{J}_{inner}(\Theta^t, \boldsymbol{\beta}^t)\right)$;
  Proxy:
    $\tilde{\Theta}^{t+1}(\boldsymbol{\beta}^t) := \Theta^t - \alpha\nabla_{\Theta^t}\mathcal{J}_{inner}(\Theta^t, \boldsymbol{\beta}^t)$;
  $\boldsymbol{\beta}$ Update (fix Θ^t):
    $\boldsymbol{\beta}^{t+1} \longleftarrow$
      $\text{OPT}_{\boldsymbol{\beta}}\left(\boldsymbol{\beta}^t, \nabla_{\boldsymbol{\beta}^t}\mathcal{J}_{outer}\left(\tilde{\Theta}^{t+1}(\boldsymbol{\beta}^t)\right)\right)$;
end

As most existing models use gradient-based methods for optimization, a simple approximation strategy with less computation is introduced as follows:

$$\nabla_{\boldsymbol{\beta}}\mathcal{J}_{outer}\left(\Theta^*(\boldsymbol{\beta})\right) \approx \nabla_{\boldsymbol{\beta}}\mathcal{J}_{outer}\left(\Theta - \alpha\nabla_\Theta\mathcal{J}_{inner}(\Theta, \boldsymbol{\beta})\right). \tag{11}$$

In this expression, α is the learning rate for one step of inner optimization. Related approximations have been validated in (Rendle 2012; Liu, Simonyan, and Yang 2019; Ma et al. 2020). Thus, we can define a proxy function to link $\boldsymbol{\beta}$ with the outer optimization:

$$\tilde{\Theta}(\boldsymbol{\beta}) := \Theta - \alpha\nabla_\Theta\mathcal{J}_{inner}(\Theta, \boldsymbol{\beta}). \tag{12}$$

For simplicity, we use two optimizers OPT_Θ and $\text{OPT}_{\boldsymbol{\beta}}$ to update Θ and $\boldsymbol{\beta}$, respectively. The iterative procedure is shown in Alg. 1.

Training and Prediction

After incorporating a parameter regularization term to avoid overfitting, the overall loss function is:

$$\min_{\boldsymbol{\beta}} \mathcal{J}_{outer}\left(\Theta^*(\boldsymbol{\beta})\right)$$
$$\text{s.t. } \Theta^*(\boldsymbol{\beta}) = \operatorname*{argmin}_{\Theta} \mathcal{J}_{inner}(\Theta, \boldsymbol{\beta}) + \lambda||\Theta||_{\text{F}}, \tag{13}$$

where λ is a hyper-parameter. When minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with backpropagation. We apply the Adam (Kingma and Ba 2015) algorithm to automatically adapt the learning rate during the learning procedure.

Recommendation Phase. For user u, we compute the distance $d_{\mathbb{B}}(\mathbf{u}, \mathbf{v})$ between the user u and each item v in the dataset. Then the items that are not in the training set and have the shortest distances are recommended to user u.

Evaluation

In this section, we first describe the experimental set-up. We then report the results of the conducted experiments and demonstrate the effectiveness of the proposed modules.

Datasets

The proposed model is evaluated on three real-world datasets from various domains with different sparsities: *Amazon-book*, *Last-FM* and *Yelp2018*, which are fully adopted from (Wang et al. 2019c). The *Amazon-book* dataset is adopted from the Amazon review dataset (He and McAuley 2016) with the *book* category, which covers a large amount of user-item interaction data, e.g., user ratings and reviews. The *Last-FM* dataset is collected from *last.fm* music website, where the tracks are viewed as the items. A subset of data from Jan. 2015 to Jun. 2015 is selected. The *Yelp2018* dataset is adopted from the 2018 edition of the *Yelp* challenge, where local businesses like restaurants and bars are viewed as the items.

All the above datasets follow the 10-core setting to ensure that each user and item have at least ten interactions. For Amazon-book and Last-FM, items are mapped into Freebase entities via title matching if there is a mapping available. For Yelp2018, the item knowledge from the local business information network (e.g., category, location, and attribute) is extracted as KG data. The data statistics after preprocessing are shown in Table 1.

For fair comparison, these three datasets in our experiments are exactly the same as those used in (Wang et al. 2019c). For each dataset, 80% of interaction data of each user is randomly selected to constitute the training set, and we treat the remaining 20% as the test set. From the training set, 10% of interactions are randomly selected as validation set to tune hyper-parameters. The experiments are executed five times and the average result is reported.

Table 1: The statistics of the datasets.

	Amazon-book	Last-FM	Yelp2018
#Users	70,679	23,566	45,919
#Items	24,915	48,123	45,538
#Interactions	847,733	3,034,796	1,185,068
#Entities	88,572	58,266	90,961
#Relations	39	9	42
#Triplets	2,557,746	464,567	1,853,704

Evaluation Metrics

We evaluate all the methods in terms of *Recall@K* and *NDCG@K*. For each user, Recall@K (R@K) indicates what percentage of her rated items emerge in the top K recommended items. NDCG@K (N@K) is the normalized discounted cumulative gain at K, which takes the position of correctly recommended items into account.

Methods Studied

To demonstrate the effectiveness of our model, we compare to the following recommendation methods:

- **FM** (Rendle 2010), a classical factorization model, which incorporates the second-order feature interactions between input features.

- **NFM** (He and Chua 2017), a state-of-the-art factorization model, which subsumes FM under a neural network.

- **CKE** (Zhang et al. 2016), a representative regularization-based method, which exploits semantic embeddings derived from TransR (Lin et al. 2015) to enhance the matrix factorization.

- **CFKG** (Ai et al. 2018), a model that applies TransE (Bordes et al. 2013) on the unified graph including users, items, entities, and relations, casting the recommendation task as the prediction of (u, Interact, i) triplets.

- **RippleNet** (Wang et al. 2018), a model that combines regularization- and path-based methods, which enrich user representations by adding those of items within paths rooted at each user.

- **GC-MC** (van den Berg, Kipf, and Welling 2017), a model designed to employ a graph convolutional network on graph-structured data. Here the model is applied on the user-item knowledge graph.

- **KGAT** (Wang et al. 2019c), a state-of-the-art KG enhanced model, which employs a graph neural network and an attention mechanism to learn from high-order graph-structured data for recommendation.

- **Hyper-Know**, the proposed model, which learns the knowledge-enhanced recommendation in the Poincaré ball and applies hyperbolic attention for distinguishing neighboring entities and bilevel optimization for adaptive regularization, respectively.

Experiment Settings

In the experiments, the latent dimension of all the models is set to 64. The parameters for all baseline methods are initialized as in the corresponding papers, and are then carefully tuned to achieve optimal performances. The learning rate is tuned amongst $[0.0001, 0.0005, 0.001, 0.005, 0.01]$, and we search for the coefficient of L2 normalization over the range $[0.0001, ..., 0.1]$. To prevent overfitting, the dropout ratio is selected from the range $[0.0, 0.1, ..., 0.9]$ for NFM, GM-MC, and KGAT. The dimension of attention network k is tested over the values $[16, 32, 64]$. Regarding NFM, the number of MLP layers is set to 1 with 64 neurons according to the original paper. For RippleNet, we set the number of hops and the memory size as 2 and 8, respectively. For KGAT, we set the depth as 3 with hidden dimension 64, 32, and 16, respectively. The network architectures of the above methods are configured to be the same as described in the original papers. For Hyper-Know, the curvature c is set to 1 and the batch size is set to 4096. The hyper-parameters are tuned on the validation set. Our experiments are conducted with PyTorch running on GPU machines (NVIDIA Tesla V100).

Table 2: The performance comparison of all methods in terms of *Recall@20* and *NDCG@20*. The best performing method is boldfaced. The underlined number is the second best performing method. * indicates the statistical significance for $p <= 0.01$ compared to the best baseline method based on the paired t-test.

	FM	NFM	CKE	CFKG	RippleNet	GC-MC	KGAT	Hyper-Know	Improv.
					Recall@20				
Amazon-book	0.1345	0.1366	0.1343	0.1142	0.1336	0.1316	<u>0.1489</u>	**0.1534***	3.23%
Last-FM	0.0778	0.0829	0.0736	0.0723	0.0791	0.0818	<u>0.0870</u>	**0.0949***	9.08%
Yelp2018	0.0627	0.0660	0.0657	0.0522	0.0664	0.0659	**0.0712**	<u>0.0683</u>	N/A
					NDCG@20				
Amazon-book	0.0886	0.0913	0.0885	0.0770	0.0910	0.0874	<u>0.1006</u>	**0.1075***	6.86%
Last-FM	0.1181	0.1214	0.1184	0.1143	0.1238	0.1253	<u>0.1325</u>	**0.1533***	16.70%
Yelp2018	0.0768	0.0810	0.0805	0.0644	0.0822	0.0790	<u>0.0867</u>	**0.0897***	3.46%

Performance Comparison

The performance comparison results are shown in Table 2.

Observations about our model. First, the proposed model—Hyper-Know, achieves the best performance for most evaluation metrics on three datasets, which illustrates the superiority of our model. Second, Hyper-Know outperforms KGAT on the Amazon-book and Last-FM datasets. Although KGAT adopts the attention model to distinguish the entity importance in the knowledge graph, it may not effectively capture the hierarchical structure between entities, which can be well-modeled by learning the entity and relation embeddings in the hyperbolic space. One possible reason why Hyper-Know does not outperform KGAT for the Recall@20 metric on the Yelp2018 dataset is that most of the entities in this KG are linked according to whether they have the same attributes, such as *HasTV*. Most of these attributes are very generic, which means that the KG provides information of limited value. As a result, much of the transfer that Hyper-Know performs from the KG to the recommendation part for the Yelp2018 dataset is likely to be noise. Third, Hyper-Know achieves better performance than GC-MC and RippleNet. Although GC-MC and RippleNet can model high-order connectivities, they fail to identify the important entities that would make a difference in recommendation. On the other hand, Hyper-Know employs an attention model in the hyperbolic space to learn the neighborhood representation of an item and transfers the knowledge from the KG to the item representation via regularization. Fourth, Hyper-Know obtains better results than CKE. One possible reason is that CKE adopts a fixed power of regularization during the whole training process. By contrast, Hyper-Know performs fine-grained regularization to regularize the item and its neighborhood. Fifth, Hyper-Know outperforms FM and NFM. One reason may be that using a distance as the scoring function can capture more fine-grained user preference.

Other observations. First, KGAT outperforms GC-MC and RippleNet. KGAT is capable of exploring the high-order connectivity in an explicit way and applies a graph attention model to aggregate the neighbors in the user-item knowledge graph in a weighted manner. Second, FM and NFM achieve better performance than CFKG and CKE in most cases. One major reason is that FM and NFM capture the second-order connectivity between users and entities, whereas CFKG and CKE model connectivity on the granu-

larity of triples, leaving high-order connectivity untouched. Third, RippleNet achieves better performance than FM. This may verify that incorporating two-hop neighboring items is of importance to enrich user representations. Fourth, NFM performs better than FM. One major reason is that NFM has stronger expressiveness, since the hidden layer allows NFM to capture the nonlinear and complex feature interactions between user, item, and entity embeddings.

Table 3: The ablation analysis. *Att* denotes the attention model, *Avg* denotes the embedding average operation, *E* denotes the Euclidean space, and *H* denotes the hyperbolic space.

Architecture	*Amazon-book*		*Last-FM*	
	R@20	N@20	R@20	N@20
(1) BPR+E	0.1017	0.0729	0.0604	0.1112
(2) BPR+H	0.1167	0.0833	0.0656	0.1191
(3) BPR+Att+E	0.1121	0.0812	0.0746	0.1319
(4) BPR+Att+H	0.1447	0.1025	0.0885	0.1453
(5) BPR+Avg+H	0.1250	0.0897	0.0775	0.1358
(6) Hyper-Know	**0.1534**	**0.1075**	**0.0949**	**0.1533**

Ablation Analysis

To verify the effectiveness of the proposed model in the Poincaré ball, the hyperbolic attention model, and the adaptive regularization mechanism, we conduct an ablation study in Table 3. This demonstrates the contribution of each module to the Hyper-Know model. In (1), we use the Euclidean distance to measure the user preference optimized by the BPR loss. In (2), we apply the distance in the Poincaré ball to measure users' preferences and optimize using Eq. 3. In (3), we integrate the TransE-style attention on the top of (1) in the Euclidean space. In (4), we add hyperbolic attention to (2). In (5), we replace the attention model in (4) with an average operation in the hyperbolic space. In (6), we present the overall Hyper-Know model to show the effectiveness of the adaptive regularization mechanism.

From the results shown in Table 3, we make the following observations. First, comparing (1) and (2), we can observe that measuring the user preference by calculating distance in the hyperbolic space achieves better performance than calculating distance in the Euclidean space. This confirms the results reported in (Tran et al. 2020). Second, from (2) and

(3), we observe that incorporating the hyperbolic attention model significantly improves the model performance. Third, in (3) and (4), we compare the performance of the attention model in both the Euclidean and hyperbolic space. From the results, we can observe that the attention model achieves better results in the hyperbolic space than in the Euclidean space. Fourth, from (1), (2), (3) and (4), we can observe that equipping the recommendation model with the KG either in the Euclidean space or hyperbolic space can improve the recommendation performance. Fifth, from (4) and (5), we observe that by distinguishing the importance of each neighbour of an item through attention, we achieve considerable improvement compared to a simple average. Comparing (4) and (6), we can observe that the adaptive regularization can provide the fine-grained regularization power.

Table 4: Training time comparison.

	CKE	CFKG	KGAT	Hyper-Know
Amazon-book	55s	22s	457s	**15s**
Last-FM	53s	27s	137s	**22s**
Yelp2018	63s	37s	352s	**20s**

Training Efficiency

In this section, we compare the training efficiency with other state-of-the-art KG-enhanced methods in terms of the training speed. We compare the time taken for one epoch of training. From the results reported in (Chen et al. 2020), these compared methods take a similar number of epochs to converge as well as our proposed method. RippleNet is not computation-efficient and takes much longer to train, we omit the comparison with RippleNet. All the experiments are conducted on a single GPU of an NVIDIA Tesla V100. All the compared methods are executed for 20 epochs and we report the average computation time, which is shown in Table 4. The training time comparison shows that Hyper-Know is more computationally efficient than other state-of-the-art methods, and the reason follows. Compared to CKE, Hyper-Know has a smaller number of learnable parameters (8.3 million v.s. 11.4 million on the Last-FM dataset). Compared to KGAT and CFKG, Hyper-Know does not incorporate the users into the KG, which makes the scale of the KG much smaller.
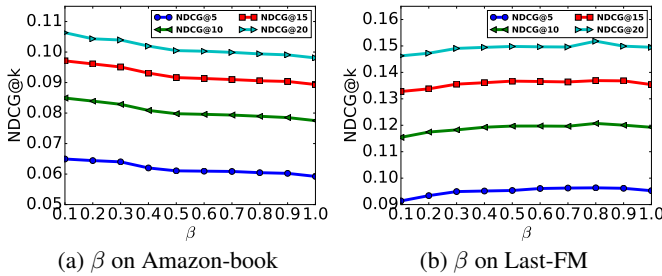

(a) β on Amazon-book   (b) β on Last-FM
Figure 2: The variation of β.

Influence of Hyper-parameters
The value of β that regularizing the item embedding with its neighborhood is an important hyper-parameter if not using the adaptive regularization mechanism. Its effect on the Amazon-book and Last-FM datasets is shown in Figure 2.

From the results in Figure 2, we observe that the value of β does affect the recommendation performance, with performance deteriorating by as much as 10 percent if a sub-optimal value is chosen. Furthermore, there is no fixed value that achieves performance that is as good as the performance obtained by using the proposed adaptive mechanism. These results demonstrate that the fine-grained and adaptive regularization benefits the recommendation task, which confirms the results reported in (Rendle 2012).
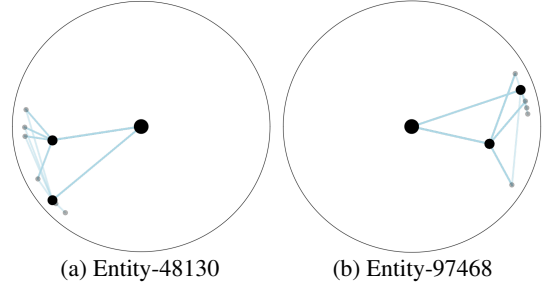

(a) Entity-48130   (b) Entity-97468
Figure 3: The embedding visualization of selected entities.

Embedding Visualization
To verify whether the learned embedding in the Poincaré ball can capture the hierarchical structure in the knowledge graph, we train Hyper-Know in the 2D space on the Last-FM dataset and visualize the entities in the 2D hyperbolic space. We randomly select two nodes and their two-hop neighbors to visualize. The visualization is shown in Figure 3. The biggest dot denotes the selected entity, the less biggest dot denotes the first-hop neighbor of the selected entity, and the smallest dot denotes the second-hop neighbor.

From Figure 3, we can observe that these three kinds of nodes may form the hierarchical patterns in Poincaré ball and the learned embeddings in the hyperbolic space can represent the hierarchical relationships.

Conclusion
In this paper, we propose a knowledge-enhanced recommendation model in the hyperbolic space (Hyper-Know) for top-K recommendation. Hyper-Know learns the user and item embeddings as well as the knowledge graph representation in the Poincaré ball model to capture the hierarchical structure in the knowledge graph. In addition, we incorporate hyperbolic attention to select the most important neighboring entities of each item. To adaptively control the regularization effect, a bilevel optimization mechanism is proposed to generate a fine-grained regularization effect between recommendation and the knowledge graph. Experimental results on three real-world datasets clearly validate the performance advantages of our model over multiple state-of-the-art methods and demonstrate the effectiveness of each of the proposed constituent modules.

References

Ai, Q.; Azizi, V.; Chen, X.; and Zhang, Y. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* .

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, H. F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR* abs/1806.01261.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.

Cannon, J. W.; Floyd, W. J.; Kenyon, R.; Parry, W. R.; et al. 1997. Hyperbolic geometry .

Chen, C.; Zhang, M.; Ma, W.; Liu, Y.; and Ma, S. 2020. Jointly Non-Sampling Learning for Knowledge Graph Enhanced Recommendation. In *SIGIR*. ACM.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An overview of bilevel optimization. *Annals OR* .

Feng, S.; Tran, L. V.; Cong, G.; Chen, L.; Li, J.; and Li, F. 2020. HME: A Hyperbolic Metric Embedding Approach for Next-POI Recommendation. In *SIGIR*. ACM.

Ganea, O.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic Neural Networks. In *NeurIPS*.

He, R.; and McAuley, J. J. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. ACM.

He, X.; and Chua, T. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. ACM.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T. 2017. Neural Collaborative Filtering. In *WWW*. ACM.

Hsieh, C.; Yang, L.; Cui, Y.; Lin, T.; Belongie, S. J.; and Estrin, D. 2017. Collaborative Metric Learning. In *WWW*. ACM.

Hu, B.; Shi, C.; Zhao, W. X.; and Yu, P. S. 2018. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In *KDD*. ACM.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. IEEE Computer Society.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proc. Int. Conf. Learning Representations*.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. ACM.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *KDD*. ACM.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. AAAI Press.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *Proc. Int. Conf. Learning Representations*.

Ma, C.; Kang, P.; and Liu, X. 2019. Hierarchical Gating Networks for Sequential Recommendation. In *KDD*. ACM.

Ma, C.; Kang, P.; Wu, B.; Wang, Q.; and Liu, X. 2019. Gated Attentive-Autoencoder for Content-Aware Recommendation. In *WSDM*. ACM.

Ma, C.; Ma, L.; Zhang, Y.; Tang, R.; Liu, X.; and Coates, M. 2020. Probabilistic Metric Learning with Adaptive Margin for Top-K Recommendation. In *KDD*. ACM.

Ma, C.; Zhang, Y.; Wang, Q.; and Liu, X. 2018. Point-of-Interest Recommendation: Exploiting Self-Attentive Autoencoders with Neighbor-Aware Influence. In *CIKM*. ACM.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R. M.; Scholz, M.; and Yang, Q. 2008. One-Class Collaborative Filtering. In *ICDM*. IEEE Computer Society.

Rendle, S. 2010. Factorization Machines. In *ICDM*. IEEE Computer Society.

Rendle, S. 2012. Learning recommender systems with adaptive regularization. In *WSDM*. ACM.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. AUAI Press.

Sala, F.; Sa, C. D.; Gu, A.; and Ré, C. 2018. Representation Tradeoffs for Hyperbolic Embeddings. In *ICML*. PMLR.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. ACM.

Sinha, A.; Malo, P.; and Deb, K. 2018. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *IEEE Trans. Evolutionary Computation* .

Sun, J.; Zhang, Y.; Ma, C.; Coates, M.; Guo, H.; Tang, R.; and He, X. 2019. Multi-graph Convolution Collaborative Filtering. In *ICDM*. IEEE.

Tran, L. V.; Tay, Y.; Zhang, S.; Cong, G.; and Li, X. 2020. HyperML: A Boosting Metric Learning Approach in Hyperbolic Space for Recommender Systems. In *WSDM*. ACM.

Tran, T.; Lee, K.; Liao, Y.; and Lee, D. 2018. Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. In *CIKM*. ACM.

van den Berg, R.; Kipf, T. N.; and Welling, M. 2017. Graph Convolutional Matrix Completion. *CoRR* abs/1706.02263.

Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; and Guo, M. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. ACM.

Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; and Guo, M. 2019a. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*. ACM.

Wang, H.; Zhao, M.; Xie, X.; Li, W.; and Guo, M. 2019b. Knowledge Graph Convolutional Networks for Recommender Systems. In *WWW*. ACM.

Wang, X.; He, X.; Cao, Y.; Liu, M.; and Chua, T. 2019c. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. ACM.

Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; and Chua, T. 2019d. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.

Wang, X.; Xu, Y.; He, X.; Cao, Y.; Wang, M.; and Chua, T. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *WWW*. ACM.

Wu, Y.; DuBois, C.; Zheng, A. X.; and Ester, M. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. ACM.

Xue, H.; Dai, X.; Zhang, J.; Huang, S.; and Chen, J. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. ijcai.org.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. ACM.

Zhang, Y.; Dai, H.; Kozareva, Z.; Smola, A. J.; and Song, L. 2018. Variational Reasoning for Question Answering With Knowledge Graph. In *AAAI*.