

Where are we in embedding spaces? A Comprehensive Analysis on Network Embedding Approaches for Recommender Systems

Sixiao Zhang*
zsx57575@gmail.com
University of Technology Sydney

Hongxu Chen*[†]
hongxu.chen@uts.edu.au
University of Technology Sydney

Xiao Ming
201934855@mail.sdu.edu.cn
Shandong University

Lizhen Cui
clz@sdu.edu.cn
Shandong University

Hongzhi Yin
h.yin1@uq.edu.au
The University of Queensland

Guandong Xu[†]
guandong.xu@uts.edu.au
University of Technology Sydney

ABSTRACT

Hyperbolic space and hyperbolic embeddings are becoming a popular research field for recommender systems. However, it is not clear under what circumstances the hyperbolic space should be considered. To fill this gap, This paper provides theoretical analysis and empirical results on when and where to use hyperbolic space and hyperbolic embeddings in recommender systems. Specifically, we answer the questions that which type of models and datasets are more suited for hyperbolic space, as well as which latent size to choose. We evaluate our answers by comparing the performance of Euclidean space and hyperbolic space on different latent space models in both general item recommendation domain and social recommendation domain, with 6 widely used datasets and different latent sizes. Additionally, we propose a new metric learning based recommendation method called SCML and its hyperbolic version HSCML. We evaluate our conclusions regarding hyperbolic space on SCML and show the state-of-the-art performance of hyperbolic space by comparing HSCML with other baseline methods.

CCS CONCEPTS

• Information systems → Recommender systems.

KEYWORDS

Recommender Systems; Hyperbolic Space; Node Embeddings

1 INTRODUCTION

With the rapid development of Internet and world wide web, recommender systems play an important role in modeling user preference and providing personalized recommendation. Because of the underlying graph structure of most real-world user-item interactions, graph-based recommendation has become a popular research field. Latent space models [14] such as matrix factorization based models and metric learning based models are widely used in graph-based recommender systems. They are capable of learning user and item embeddings and utilizing the expressiveness of the latent space to capture the underlying distribution and the latent structure of the data. Most latent space models are built in Euclidean space, the

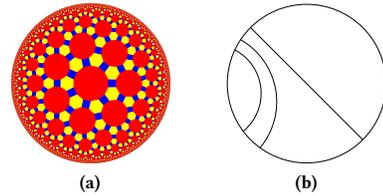


Figure 1: (a) Hyperbolic space expands exponentially¹. (b) 2D Poincaré disk and geodesics.

most straightforward latent space consistent with human cognition. However, the ability of embedding methods to model complex patterns is inherently bounded by the dimensionality of Euclidean space [21]. Euclidean embeddings suffer from a high distortion when embedding scale-free and hierarchical data [4]. The distances of users and items are not likely to be preserved when embedded into Euclidean space. It is necessary to increase the latent size in order to reduce the distortion. However, as the latent size increases, the resources needed to train and store the model also increase.

To solve this problem, a new latent space with a smaller embedding distortion is needed. Recently, hyperbolic space has been explored as a new latent space and has shown impressive performance. It outperforms Euclidean space in many domains including natural language processing [7], link prediction and node classification [4], top-n recommendation [27], etc. A key property of hyperbolic space is that it expands faster than Euclidean space. Euclidean space expands polynomially, while hyperbolic space expands exponentially. As shown in Fig. 1a, if we treat it as a circle in Euclidean space, the polygons away from the center of the circle are smaller than those close to the center. But if we treat it as a circle in hyperbolic space, then every polygon has the same size. This suggests that, to embed the same group of data, the space needed in Euclidean space is larger than hyperbolic space. In other words, the data capacity of hyperbolic space is larger than Euclidean space. Another important property of hyperbolic space is that it can preserve the hierarchies of data. Many real-world data such as texts, e-commercial networks and social networks exhibit an underlying hierarchical tree-like structure with power-law distribution [1, 21]. Meanwhile, hyperbolic space can be thought as a continuous version of trees. Therefore, such networks are consistent with hyperbolic space due to their analogous structure, and

*Both authors contributed equally to this research.

[†]Corresponding author.

can be naturally modeled by hyperbolic space with a much lower distortion compared to Euclidean space.

Although hyperbolic embeddings are gaining great attention for recommender systems nowadays [3, 4, 19, 29], it is not clear under what circumstances the hyperbolic space should be considered. A critical analysis and guidance are missing in existing literature. As a current trendy topic of hyperbolic embeddings, practitioners are indiscriminately attempting to transfer variants of existing algorithms into hyperbolic geometry in regardless of having strong motivations. Therefore, it is timely needed for a fair comparison and analysis on existing recommendation algorithm in hyperbolic space. It is unknown whether existing methods perform better in hyperbolic space compared against their original feature spaces. From dataset perspective, if hyperbolic geometry is regarded as a more suitable space for particular recommendation scenarios, we are wondering whether hyperbolic space is versatile across different datasets or is only suitable for particular datasets? Fair latitudinal comparisons are also missing in current existing works.

Our work aims to fix the gaps and provide a comprehensive analysis on hyperbolic space and hyperbolic embeddings for recommender systems. We first introduce hyperbolic space and the Poincaré ball model. Then we propose three hypotheses regarding the performance of hyperbolic space on different models, datasets, and latent sizes. Meanwhile, we propose a metric learning based social recommendation approach named Social Collaborative Metric Learning (SCML) and its hyperbolic version Hyperbolic Social Collaborative Metric Learning (HSCML). Our implementation is available at Github². We empirically validate our hypotheses on 6 benchmark datasets and 6 models from both general item recommendation domain and social recommendation domain. Finally we draw conclusions and give comments on how to use hyperbolic space. Our main contributions are:

- We provide theoretical analysis and empirical results to validate our three hypotheses: hyperbolic space is more suited for distance models than projection models; hyperbolic space is more powerful on datasets with a low density; hyperbolic space greatly outperforms Euclidean space when the latent size is small, but as the latent size increases, Euclidean space becomes comparable with hyperbolic space.
- We address the drawbacks of hyperbolic space, and give comments on when and where to use hyperbolic space.
- We propose a metric learning based social recommendation method SCML and its hyperbolic version HSCML and validate our hypotheses on them. We also show that hyperbolic space has state-of-the-art performance by comparing HSCML with other baselines on two benchmark datasets.

2 PRELIMINARIES

2.1 The Poincaré Ball Model

There are five well-known hyperbolic models: the Klein model \mathbb{K} , the Poincaré ball model \mathbb{D} , the half-plane model \mathbb{P} , the hyperboloid model \mathbb{H} , and the hemisphere model \mathbb{J} . Each of them is defined on a different domain in \mathbb{R} , and has a different Riemannian metric. Readers can refer to [2] for more detailed descriptions for them.

Among the five models, the Poincaré ball model is a very good choice for learning hyperbolic embeddings because it is well-suited for gradient-based optimization [21]. The definition domain of the Poincaré ball model with constant curvature $-c = -1$ is

$$\mathbb{D} = \{(x_1, \dots, x_n) : x_1^2 + \dots + x_n^2 < \frac{1}{c}\} \quad (1)$$

In \mathbb{R}^n , it is an n-dimensional open unit ball when $c = 1$.

The geodesics in the Poincaré ball, which corresponds to the straight lines in Euclidean space, are circles which are orthogonal to the sphere. Fig. 1b gives an illustration of a 2D Poincaré disk and its geodesics. The distance between two points \mathbf{u}, \mathbf{v} in the Poincaré ball is the length along the geodesic. With constant curvature -1, the distance is calculated by

$$d_{\mathbb{D}}(\mathbf{u}, \mathbf{v}) = \operatorname{arcosh} \left(1 + \frac{2\|\mathbf{u} - \mathbf{v}\|^2}{(1 - \|\mathbf{u}\|^2)(1 - \|\mathbf{v}\|^2)} \right) \quad (2)$$

Here $\|\cdot\|$ is the Euclidean norm, arcosh is the inverse hyperbolic cosine function. The distance is determined only by the position of \mathbf{u} and \mathbf{v} , and therefore changes smoothly with respect to \mathbf{u} and \mathbf{v} . Besides, there is only one center point in the Poincaré ball, which is exactly the origin, so it is convenient to put the root node at the origin, and the leaves will spread around the root layer by layer, capturing the hierarchical structure of the graph. Furthermore, because all the embeddings are within the unit ball, we don't need to regularize or clip the embeddings as we do in Euclidean space, meanwhile maximizing the use of the whole space. Another advantage of the Poincaré ball model is its conformality, which means that the angles are preserved when transforming Euclidean vectors to the Poincaré ball. This is useful for us to define inner product in the Poincaré ball to evaluate the performance of MF-based methods.

As for the other four hyperbolic models, the hemisphere model is used as a tool for visualising transformations between the other models, and is not often used as a model itself; the distance between two points in the Klein model is related to the two ideal points at the intersections of the unit sphere and the straight line that connects the two target points, so the distance function is not as smooth and easy to calculate as the Poincaré ball; the half-plane model uses the upper half-plane as its definition domain, and all the points that lie on the boundary of the half-plane can be treated as the center, therefore it is necessary to regularize the embeddings as they may spread all over the half-plane; the hyperboloid model also has the issue of regularization since its definition domain includes the infinity, too. Therefore, the Poincaré ball model naturally becomes a good choice for learning hyperbolic embeddings compared to the other hyperbolic models.

2.2 Latent Space Model

A key reason why hyperbolic space is suitable for recommendation tasks is that, hyperbolic space and real-world datasets both expand exponentially. This is true because we are looking at the datasets at node level. Therefore, theoretically only models that are capable of learning node embeddings are suitable for hyperbolic space. Such models are called latent space models [14]. Each node is represented by an n-dimensional embedding vector. Recommendations are made by comparing the relations between nodes. Based on the different methods that are used to calculate the relation, latent space models can be split into two categories, namely projection models and

²<https://github.com/RinneSz/Social-Collaborative-Metric-Learning>

distance models. Projection models use the inner product to model the relation. One typical example is matrix factorization. For two node embeddings \mathbf{u} and \mathbf{v} , their relation $r_{\mathbf{u}\mathbf{v}}$ is

$$r_{\mathbf{u}\mathbf{v}} = \mathbf{u}^T \mathbf{v} + b \quad (3)$$

b is the bias. Usually larger $r_{\mathbf{u}\mathbf{v}}$ means a closer relationship. The relation calculated by Eq. 3 is used for downstream pairwise ranking tasks or rating prediction tasks. Distance models use the distance between a pair of node embeddings to model their relation. Metric learning approaches are the most well-known distance models. Typically, the relation between two nodes \mathbf{u} and \mathbf{v} is

$$r_{\mathbf{u}\mathbf{v}} = d_{\mathbf{u}\mathbf{v}} \quad (4)$$

$d_{\mathbf{u}\mathbf{v}}$ is the distance between \mathbf{u} and \mathbf{v} . Two nodes are more likely to have a link or belong to the same category if they are close.

2.3 Poincare Ball Optimization and Geometry

Euclidean gradient descent method can't be directly applied to the hyperbolic space, because the value and direction of the hyperbolic gradient might be different from the Euclidean gradient. One solution is to map the hyperbolic embeddings into Euclidean embeddings, and use Euclidean gradient descent method to optimize the Euclidean embeddings, then map the updated Euclidean embeddings back into the hyperbolic embeddings. The tangent space $T_x\mathbb{D}$ of a certain point x in the hyperbolic space \mathbb{D} is often used as the target Euclidean space. The common vector operations such as matrix multiplication and addition are done on the tangent space. Thus, the hyperbolic optimization problem is simplified into a Euclidean optimization problem.

In the following part, we will introduce the necessary mathematical basis for the Poincaré ball model. **Note that the definition domain and the distance function are already defined in Eq. 1 and Eq. 2.**

Riemannian Metric. The Riemannian metric of an n -dimensional Poincaré ball at point x is

$$g_x^{\mathbb{D}} = \lambda_x^2 g^E, \text{ where } \lambda_x = \frac{2}{1 - c\|x\|^2} \quad (5)$$

$g^E = \mathbf{I}_n$ is the Euclidean metric, which is an n -dimensional identity matrix. We recover the Euclidean space when $c = 0$.

Hyperbolic Norm. The norm of a Poincaré embedding \mathbf{u} is its distance to the origin. When $c = 1$, the norm is

$$\|\mathbf{u}\|_{\mathbb{D}} = \operatorname{arcosh} \left(1 + \frac{2\|\mathbf{u}\|^2}{1 - \|\mathbf{u}\|^2} \right) \quad (6)$$

The norm is 0 when $\|\mathbf{u}\| = 0$, and goes to infinity when $\|\mathbf{u}\|$ is close to 1.

Hyperbolic Inner Product. Because the Poincaré ball model is a conformal model, for any two embeddings, their angle in the Poincaré ball is the same as their angle in Euclidean space. Therefore we define the Poincaré inner product as

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{D}} = \|\mathbf{u}\|_{\mathbb{D}} \cdot \|\mathbf{v}\|_{\mathbb{D}} \cdot \cos \langle \mathbf{u}, \mathbf{v} \rangle, \text{ where } \cos \langle \mathbf{u}, \mathbf{v} \rangle = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} \quad (7)$$

Here $\langle \cdot, \cdot \rangle$ is the Euclidean inner product.

Exponential Map & Logarithmic Map. The mapping from the tangent space $T_x\mathbb{D}$ to the hyperbolic space \mathbb{D} is called the exponential map, and the mapping from the hyperbolic space \mathbb{D} to

the tangent space $T_x\mathbb{D}$ is called the logarithmic map. Usually the origin o is chosen to be the target point x because of the simplicity and symmetry of the mapping function at o . With constant negative curvature $-c$, for $\mathbf{t} \in T_o\mathbb{D}_c$ and $\mathbf{u} \in \mathbb{D}_c$, the exponential map $\exp_o^c : T_o\mathbb{D}_c \rightarrow \mathbb{D}_c$ and the logarithmic map $\log_o^c : \mathbb{D}_c \rightarrow T_o\mathbb{D}_c$ at the origin o are defined as

$$\exp_o^c(\mathbf{t}) = \tanh(\sqrt{c}\|\mathbf{t}\|) \frac{\mathbf{t}}{\sqrt{c}\|\mathbf{t}\|}, \quad \log_o^c(\mathbf{u}) = \operatorname{artanh}(\sqrt{c}\|\mathbf{u}\|) \frac{\mathbf{u}}{\sqrt{c}\|\mathbf{u}\|} \quad (8)$$

Hyperbolic Linear Layer. In some neural network based methods, we need to change the traditional Euclidean linear layer into hyperbolic linear layer. A typical Euclidean linear layer is composed of three parts: weight multiplication, bias addition, and an activation function at its output. Suppose the weight matrix is \mathbf{W} , the bias vector is \mathbf{b} , and the activation function is σ , the Euclidean linear layer with a Euclidean input vector \mathbf{u} can be written as

$$\mathbf{y} = \sigma(\mathbf{W} \cdot \mathbf{u} + \mathbf{b}) \quad (9)$$

The corresponding hyperbolic linear layer is also composed of the above three parts. Note that, for a hyperbolic linear layer, the input \mathbf{u} should also be hyperbolic, but the weight matrix \mathbf{W} and bias \mathbf{b} should be defined on the tangent space, which means that they are Euclidean. The hyperbolic weight multiplication is defined as

$$\mathbf{W} \odot \mathbf{u} = \exp_o^c(\mathbf{W} \cdot \log_o^c(\mathbf{u})) \quad (10)$$

The hyperbolic bias addition is

$$\mathbf{u} \oplus \mathbf{b} = \exp_o^c(\log_o^c(\mathbf{u}) + \mathbf{b}) \quad (11)$$

Using Eq. 10 and Eq. 11, the hyperbolic linear layer is defined as

$$\mathbf{y} = \exp_o^c(\sigma(\log_o^c((\mathbf{W} \odot \mathbf{u}) \oplus \mathbf{b}))) \quad (12)$$

where σ is the activation function defined in Euclidean space such as ReLU or Sigmoid.

3 HYPOTHESES

In this section, we present and explain three hypotheses we make about hyperbolic space. Our experiments are designed to verify the correctness of them.

Distance models are more suited for learning hyperbolic embeddings than projection models. Since the projection models optimize the inner product and the distance models optimize the distance, the distribution of the embeddings is different after convergence. For the projection models, the relation between nodes is mainly determined by the angle. Angles of positive pairs are small and angles of negative pairs are obtuse. However, for the distance models, the relation is only determined by the distance. The positive pairs are pushed close to each other, and the negative pairs are pushed away from each other.

If we apply hyperbolic space to projection models, one issue is that, projection models usually regularize the norm of the embeddings in a limited scale, which makes it impossible to push the embeddings far from the center. So in hyperbolic space, the norm of the embedding will also be in a limited scale, making it hard to make use of the outer part of the hyperbolic space where the capacity is the largest efficiently, thus reducing the expressiveness. However, distance models do not have such limitation and can make use of the entire space. Embeddings can be pushed arbitrarily far from the origin as long as the precision allows. Another advantage of distance models is the ability to learn hierarchical information. Nodes

on a circle have the same relation with the center node, so a group of leaf nodes will be likely to spread around a root node layer by layer in the form of concentric circles. As we have mentioned that the number of leaf nodes and the capacity of the hyperbolic space are all increasing exponentially, it’s easier for distance models to learn low-distortion hyperbolic embeddings compared to projection models. Therefore, distance models should be a better choice for learning hyperbolic embeddings rather than projection models.

Hyperbolic space is more powerful compared to Euclidean space when the density of the dataset is small. Hyperbolic space is naturally capable of extracting the hierarchical information from datasets. It can use such hierarchy to help learn the node embeddings more effectively. Therefore, it can have a relatively good performance even the density is low. However, it is difficult for Euclidean space to extract the hierarchical information. Instead, Euclidean space needs more user-item pairs to help reach a comparable performance with hyperbolic space. That is to say, if the density of the dataset is small, then hyperbolic space is likely to outperform Euclidean space, but if the density is large, Euclidean space can have a comparable performance with hyperbolic space.

The performance of hyperbolic space is better than Euclidean space when the latent size is small, but as the latent size increases, the performance of Euclidean space becomes comparable with hyperbolic space. The expressiveness of hyperbolic space is much larger than Euclidean space. For example, when embedding one type of hierarchy, a 2D Poincaré disk can have arbitrary small distortion, but a 2D Euclidean space will always have some distortion [6]. So to fully capture all the hierarchical information in the dataset, the number of dimensions needed in hyperbolic space is much smaller than Euclidean space. This is our reason why hyperbolic space should outperform Euclidean space when the latent size is small. On the other side, as the latent size increases, the performance of Euclidean space and hyperbolic space both have an upper bound. Besides, they should share the same upper bound, because both of them are capable of embedding the nodes with an arbitrary low distortion with a large enough latent size. So as the latent size increases, the gap between their performances should decrease. Eventually Euclidean space will have a similar performance with hyperbolic space.

4 EVALUATING HYPERBOLIC SPACE

We evaluate the performance of hyperbolic space in two different recommendation domains, namely general item recommendation and social recommendation. The reasons why these two domains are chosen are that 1) data sparsity issues and long-tailed distributions are common in these two recommendation tasks, which are regarded as more suitable for hyperbolic embedding space. To be specific, the user-item rating distribution is usually power-law in both domains. Moreover, in social recommendation, the user-user social network distribution is also usually power-law. So hyperbolic space could be a more consistent latent space for both general item recommendation and social recommendation. A thorough evaluation on these two tasks will help following-up research avoid potential trial and errors in developing new research directions. 2) general recommendation and social recommendation are most popular tasks in recommender system research area, studying these

two classical tasks will have broader impact and value to the community. Most existing baseline approaches have only reported their performance in Euclidean space, therefore we implement them in hyperbolic space and report their performance in both Euclidean space and hyperbolic space on several benchmark datasets. We aim to verify our hypotheses proposed in section 3 empirically.

4.1 Baselines

Baseline	Domain	Task	Model
MF-BPR	general	top-n	projection
CML	general	top-n	distance
DMF	general	top-n	projection
TrustSVD	social	rating	projection
SoRec	social	rating	projection

Table 1: Baseline categories.

We conduct experiments on some of the most representative distance models and projection models in general item recommendation domain and social recommendation domain.

- **Matrix Factorization with Bayesian Personalized Ranking (MF-BPR)** [24]: a matrix factorization based method optimized by Bayesian personalized ranking loss.
- **Collaborative Metric Learning (CML)** [15]: a metric learning approach that learns a joint metric space to encode not only users’ preferences but also the user-user and item-item similarity.
- **Deep Matrix Factorization (DMF)** [30]: a novel matrix factorization model using neural network architecture to learn the node embeddings.
- **TrustSVD** [11]: a social recommendation method that extends SVD++ by incorporating social trust information to help predict user preference.
- **SoRec** [20]: a factor analysis approach based on probabilistic matrix factorization which exploits both the user-item rating matrix and the adjacency matrix of the social networks.

The domains and tasks of these baselines and whether they are distance models or projection models are described in Table 1. We use these five models because of two reasons. First, they are some of the most standard and basic models of their categories. We can compare the performance of Euclidean space and hyperbolic space with little noise or bias which might be introduced by auxiliary model components. MF-BPR is a standard matrix factorization based projection model. CML is a baseline metric learning based distance model. DMF is one of the simplest projection models that incorporate neural network structure. Another reason why we use these five models is that, the number of existing latent space models is limited. Many other popular recommendation methods such as Neural Collaborative Filtering (NeuMF) [13], Factorization Machines (FM) [23], and GraphRec [8] do not belong to latent space models. We have explained in subsection 2.2 that, theoretically, only latent space models are suitable for hyperbolic space. For those which are not latent space models, hyperbolic space is not likely to outperform Euclidean space. Even if they happen to outperform Euclidean space, it is difficult to explain the underlying reason because they do not present any hierarchical property.

When we are investigating existing social recommendation methods, we notice that most methods are projection models and are

Dataset	# Users	# Items	# Ratings	Rating Density	# Social Connections	Social Density
Movielens 1M	6040	3706	1000209	4.4684%		
Movielens 100K	943	1682	100000	6.3047%		
Automotive	2928	1835	20473	0.3810%		
Cellphones	27879	10429	194439	0.0669%		
Epinions	40163	139738	664824	0.0118%	487183	0.0302%
Ciao	10420	111520	296558	0.0255%	128797	0.1186%

Table 2: Dataset statistics.

designed for rating prediction tasks. There is a lack in top-n social recommendation with distance models. Therefore, in the next section, we will propose and evaluate a new method to fill this gap.

4.2 Datasets

We use four datasets for general item recommendation and two for social recommendation.

- **MovieLens**³: a widely adopted benchmark dataset in the application domain of recommending movies to users provided by GroupLens research. We use two configurations, namely **MovieLens 1M** and **MovieLens 100K**.
- **Amazon review data** [12]: a popular benchmark dataset that contains product reviews and metadata from Amazon. We use two subsets of it, namely **Cellphones** and **Automotive**. Both are 5-core subsets⁴.
- **Epinions**⁵: a popular dataset from a consumer review website called Epinions⁶, where users can rate items and add other users to their trust lists.
- **Ciao**⁷: another popular social recommendation dataset from social networking website Ciao⁸. It also allows users to rate items and trust other users.

Table 2 show the statistics of the datasets. Movielens 1M and Movielens 100K have a very large density, about 4% and 6%, while other four datasets are all below 0.4%. Conclusions about the influence of density can be made by comparing the performance of them.

4.3 Evaluation Metrics

Some baseline approaches are originally proposed for the top-n recommendation task, others are for the rating prediction task. We keep their tasks unchanged for the best performance. We report the hit ratio (HR) at $k = \{1, 5, 10, 15, 20\}$ and normalized discounted cumulative gain (NDCG) at $k = \{1, 5, 10, 15, 20\}$ for top-n recommendation tasks, and report the mean absolute error (MAE) and rooted mean square error (RMSE) for rating prediction tasks.

For rating prediction tasks, we leave 20% of the ratings as the validation set, and 20% as the test set, the remaining 60% as the training set. For top-n ranking tasks, we adopt the popular leave-one-out method. We leave the last item each user has interacted as the test set, and leave the penultimate as the validation set. **Note that we do not do negative sampling for both HR and NDCG**, instead we rank all the negative items, in this way we can have a more accurate and convincing result and avoid the bias introduced by the sampling [17].

³<https://grouplens.org/datasets/movielens/>

⁴<http://jmcauley.ucsd.edu/data/amazon/>

⁵http://www.trustlet.org/downloaded_epinions.html

⁶<http://www.epinions.com>

⁷<https://www.cse.msu.edu/~tangjili/datasetcode/>

⁸<http://www.ciao.co.uk>

Hit Ratio. For a *single* user, leave one item out from the user’s list of interacted items (usually the last one that the user has interacted with). Then rank the left out positive item together with all the negative items that the user has not interacted with. If the left out item is at the top n of the ranking list, then we consider this item as a hit. The hit ratio of the recommender system is then the total number of hits divided by the number of users.

Normalized Discounted Cumulative Gain. We first explain what cumulative gain (CG) and discounted cumulative gain (DCG) are. Given a list of top n recommendations, $r_i = 1$ if the i -th item has been interacted or is liked by the user; $r_i = 0$ if the i -th item has not been interacted or is disliked by the user. Thus, CG and DCG are calculated by

$$CG@n = \sum_{i=1}^n r_i, \quad DCG@n = \sum_{i=1}^n r_i \cdot \frac{1}{\log_2(i+1)} \quad (13)$$

CG measures the relevance of the recommendation list. DCG improves it by taking into consideration the ordering of the items. But DCG is not normalized so it may vary for different users. Therefore, we use the ideal discounted cumulative gain (IDCG) as the normalization constant, which is calculated by moving all the interacted items to the top of the list. This is an *ideal* list and it has the largest DCG value. Therefore, NDCG is always between 0 and 1:

$$NDCG@n = \frac{DCG@n}{IDCG@n} \quad (14)$$

Mean Absolute Error and Rooted Mean Square Error. MAE and RMSE are measures of errors between predicted ratings and observed ratings. They is calculated using the following equations:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n}, \quad RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - x_i)^2}{n}} \quad (15)$$

where y is the observed rating and x is the predicted rating.

4.3.1 Experiment Settings. Note that the target of our experiment is to compare the performance of the Euclidean space and the hyperbolic space on various latent space models, not to compare the performance of different models. For each model, the Euclidean setting and the hyperbolic setting share some basic parameters like the learning rate and batch size, we keep such parameters to be the same in both Euclidean setting and hyperbolic setting. Other latent space specified parameters such as the margin in CML is tuned individually in Euclidean setting and hyperbolic setting to reach their best performance.

We report the test scores based on the best validation scores. For each experiment, we do five independent trials and report the average result. For all methods, the learning rate is tuned among $\{0.1, 0.01, 0.001, 0.0001\}$ using Adam optimizer. The batch size of each dataset is tuned among $\{50, 500, 5000, 50000\}$ for a relatively good performance and short running time. We set hyperbolic curvature to be constant -1. For top-n recommendation tasks, we report

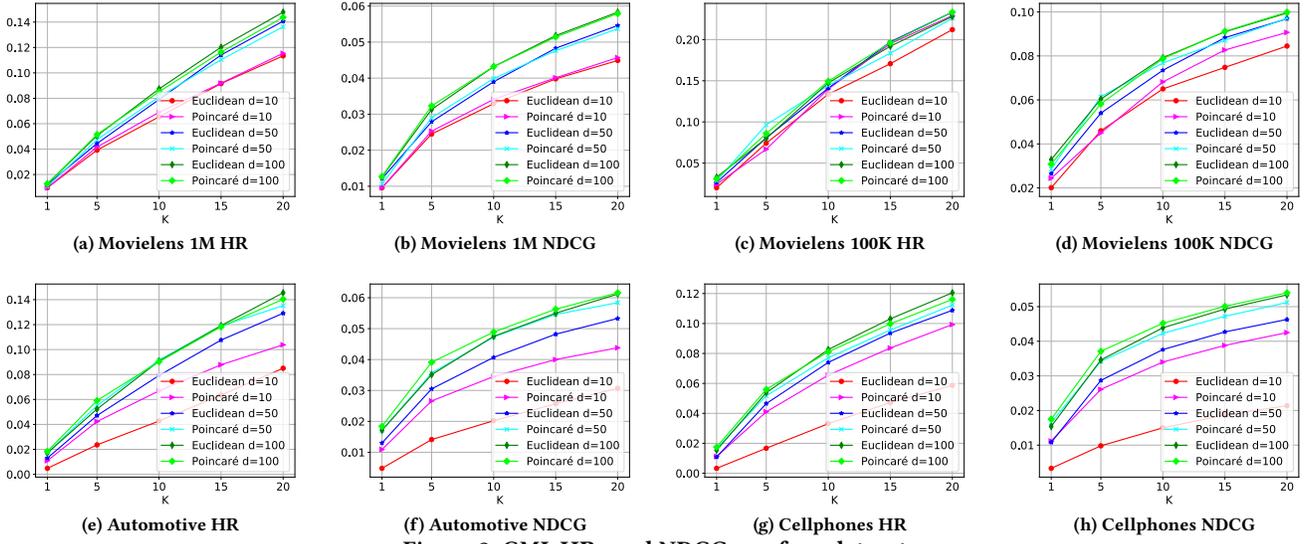


Figure 2: CML HRs and NDCGs on four datasets.

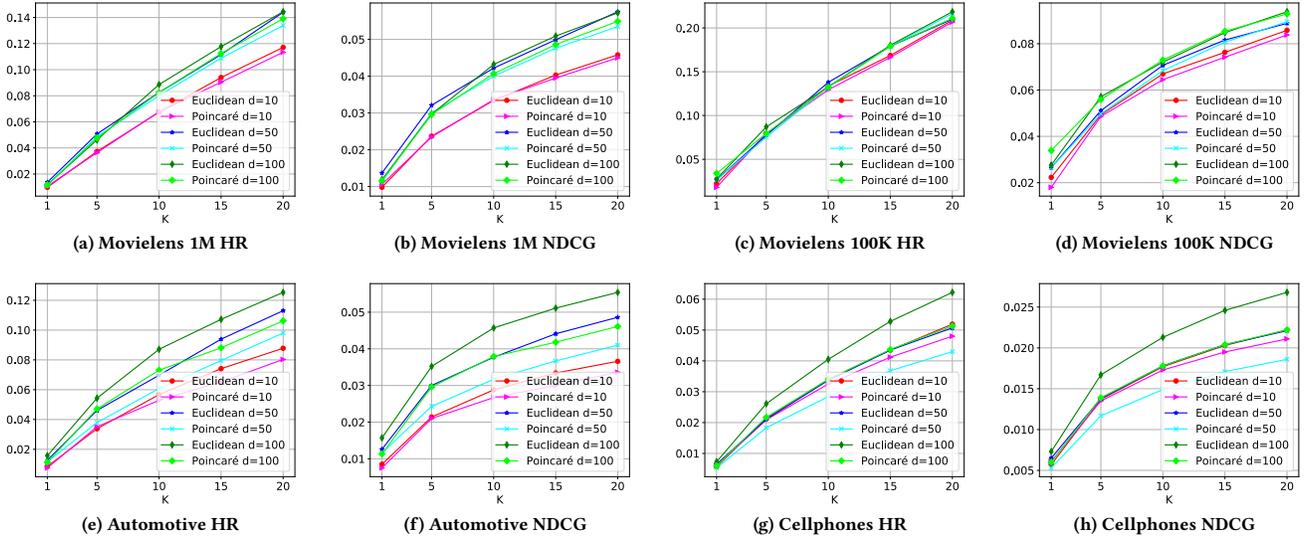


Figure 3: MF-BPR HRs and NDCGs on four datasets.

HRs and NDCGs when the latent size $d = \{10, 50, 100\}$. For rating prediction tasks, we report MAE and RMSE when the latent size $d = \{8, 16, 32, 64\}$. For CML, the margin is tuned from 0.1 to 2.0 for Euclidean space, and from 2 to 40 for hyperbolic space (the norm of the hyperbolic embedding is clipped by approximately 6 due to the precision). We keep other hyper-parameters as suggested by the original papers.

4.3.2 Results. The results of our experiments are shown from Fig. 2 to Fig. 5. Following the hypotheses we made in section 3, we first compare the performance of distance models and projection models, then compare different datasets, last we compare different latent sizes.

Distance Models vs. Projection Models. According to the results, hyperbolic space outperforms Euclidean space in distance

models such as CML. For example, from the results of CML on Automotive and Cellphones shown in Fig. 2e, Fig. 2f, Fig. 2g, and Fig. 2h, the numbers of hyperbolic space have a great improvement over the numbers of Euclidean space. One interesting thing is that for MovieLens shown in Fig. 2a, Fig. 2b, Fig. 2c, and Fig. 2d, hyperbolic space and Euclidean space seem to have a similar performance. This will be explained when we compare different datasets in the next part. For other projection models including MF-BPR in Fig. 3, DMF in Fig. 4, TrustSVD and SoRec in Fig. 5, hyperbolic space can't outperform Euclidean space in any of them. Therefore, we can draw a conclusion that distance models are more suited for hyperbolic space than projection models.

High Density Datasets vs. Low Density Datasets. The performance of Euclidean space and hyperbolic space on MovieLens

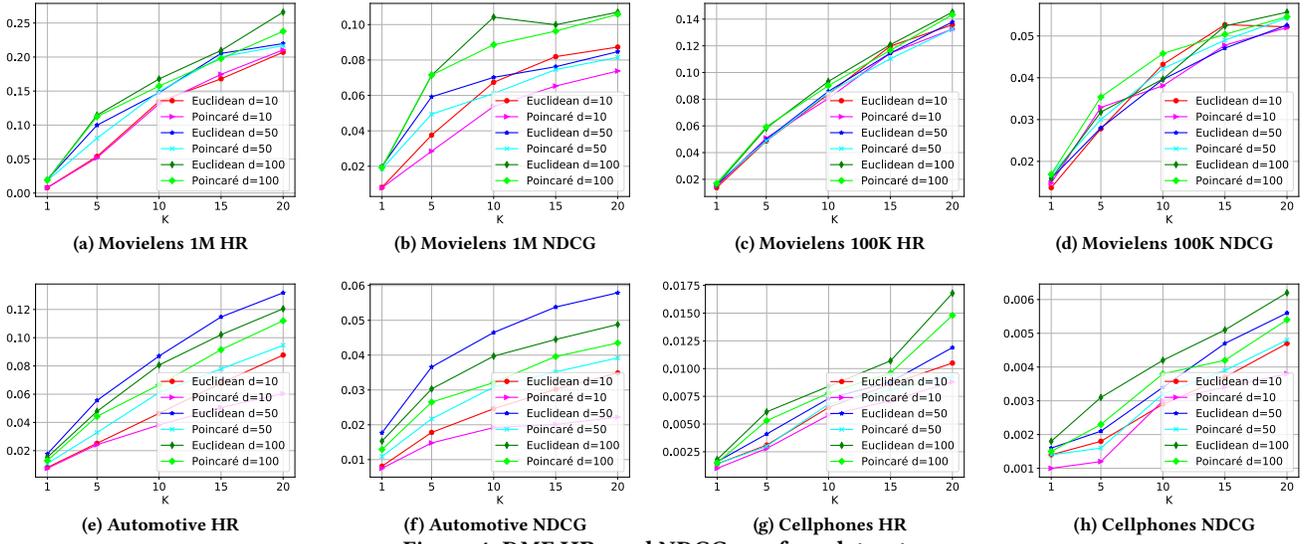


Figure 4: DMF HRs and NDCGs on four datasets.

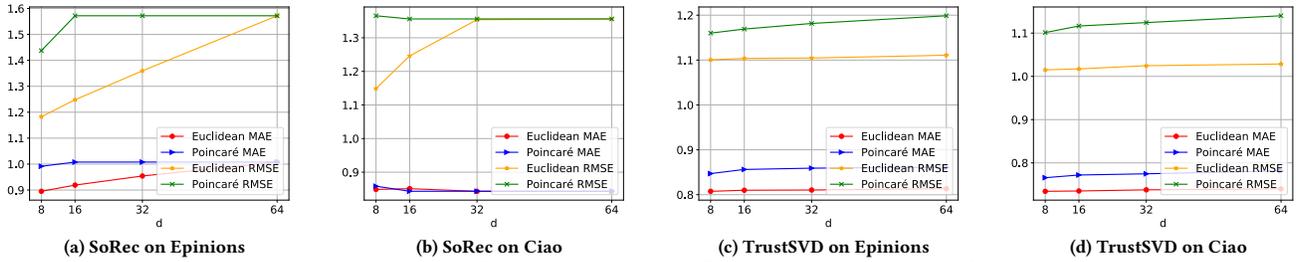


Figure 5: SoRec and TrustSVD MAEs and RMSEs on Epinions and Ciao.

1M and MovieLens 100K are much closer than Automotive and Cellphones for general item recommendation methods as shown in Fig. 2, Fig. 3, and Fig. 4. In Fig. 2, the performance of Euclidean space even becomes comparable with hyperbolic space on MovieLens. From Table 2, we can see that MovieLens 1M and MovieLens 100K have a very high density compared with other datasets. Based on the above observations and our analysis in section 3, we can draw a conclusion that datasets with a lower density can benefit more from hyperbolic space.

Influence of the Latent Size. From the results of CML on Automotive and Cellphones as shown in Fig. 2e, Fig. 2f, Fig. 2g, and Fig. 2h, we can see that the improvement of hyperbolic space over Euclidean space decreases as the latent size increases. This serves as an evidence of our hypothesis about the latent size made in section 3. We can draw a conclusion that hyperbolic space is better than Euclidean space in metric learning based approaches especially when the latent size is small, but when the latent size is large enough, Euclidean space becomes comparable with hyperbolic space.

4.3.3 Drawbacks of Hyperbolic Space. Although hyperbolic space can outperform Euclidean space, there are still some drawbacks when using hyperbolic space.

High Computational Complexity. When using hyperbolic space, the exponential map and logarithmic map are necessary whenever we apply a simple operation to the embedding such as matrix multiplication and bias addition. This makes the algorithm

computational inefficient. It does not cause much trouble in some straightforward methods which do not have much computational work such as CML and SoRec, but greatly increases the computational resources needed in some complicated approaches such as DMF which involves neural network structures.

Pay Attention to Invalid Values. Since the computation involves inverse hyperbolic functions, it is extremely easy for the denominators to be 0 or infinite. Due to the precision of the device, it is necessary to clip values wherever might become invalid.

4.3.4 Comments on Using Hyperbolic Space. Based on the above results, we can now make several comments and suggestions for using hyperbolic space and learning hyperbolic embeddings.

- Distance models are more suited for learning hyperbolic embeddings than projection models.
- If the density of the dataset is large, Euclidean space should be a better choice because it has a comparable performance with hyperbolic space and the computational complexity is low; when the density is low, hyperbolic space is more preferable because it will have a much better performance than Euclidean space.
- Choose an appropriate latent size. In most cases, hyperbolic embeddings only need a relatively small latent size to achieve good performance, which can help to save resources.

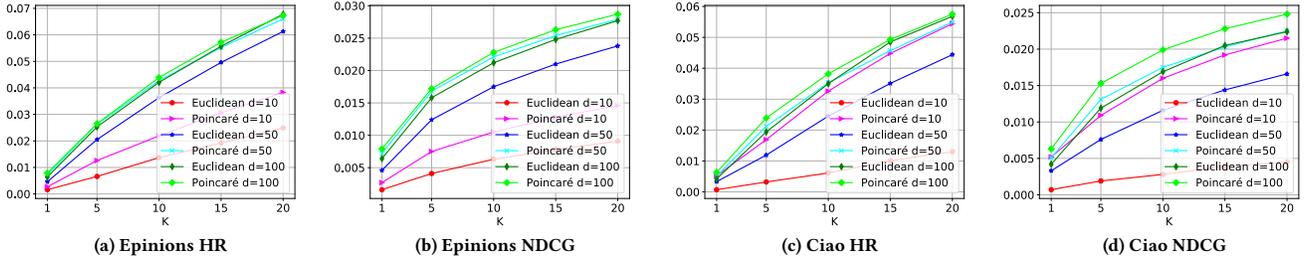


Figure 6: SCML HRs and NDCGs on Epinions and Ciao.

5 SOCIAL COLLABORATIVE METRIC LEARNING

As we explained in subsection 4.1 that there is a lack in using distance models to solve top-n social recommendation task, we here propose a new metric learning based social recommendation method for top-n recommendation called Social Collaborative Metric Learning (SCML), which is a generalized form of Collaborative Metric Learning (CML) [15]. We apply SCML in both Euclidean space and hyperbolic space using their corresponding distance functions. We call hyperbolic SCML as HSCML.

Suppose the set of user-item positive pairs is \mathcal{R} , and the set of user-user positive pairs is \mathcal{S} . For user i , sample a positive item j and a negative item k such that $(i, j) \in \mathcal{R}$, $(i, k) \notin \mathcal{R}$, and sample a positive neighbor m and a negative neighbor n such that $(i, m) \in \mathcal{S}$, $(i, n) \notin \mathcal{S}$. The item side loss \mathcal{L}_{item} and the social connection side loss \mathcal{L}_{so} are defined as

$$\mathcal{L}_{item} = \sum_{(i,j) \in \mathcal{R}} \sum_{(i,k) \notin \mathcal{R}} [m_{item} + d(i, j)^2 - d(i, k)^2]_+ \quad (16)$$

$$\mathcal{L}_{so} = \sum_{(i,m) \in \mathcal{S}} \sum_{(i,n) \notin \mathcal{S}} [m_{so} + d(i, m)^2 - d(i, n)^2]_+ \quad (17)$$

where m_{item} and m_{so} are the safety margins, $[\cdot]_+ = \max(\cdot, 0)$ is the standard hinge loss. $d(\cdot, \cdot)$ is the Euclidean distance function in SCML, and is the hyperbolic distance function in HSCML. The loss function of our model is

$$\mathcal{L} = \mathcal{L}_{item} + \lambda \mathcal{L}_{so} \quad (18)$$

where λ controls the weight of \mathcal{L}_{so} . We use \mathcal{L}_{item} to push the positive user-item pairs close to each other and push the negative user-item pairs away to each other, and use \mathcal{L}_{so} to push the socially related users close to each other. Thus, for a particular user, the items that have been interacted by his or her socially connected neighbors are also being pushed close to the user. This reflect the fact that users are more likely to interact with the items that their friends have interacted with.

6 EVALUATING SCML AND HSCML

Like what we did in section 4 when evaluating the performance of hyperbolic space, we compare the performance of SCML and HSCML. Besides, in order to show how hyperbolic space performs compared with state-of-the-art Euclidean models, we compare the performance of HSCML with Euclidean baseline approaches on Ciao and Epinions using some of the experiment results in the paper of RML-DGATs [28]. The baselines include MF-BPR [24], FISM [16], NeuMF [13], CML [15], LRML [25], SBPR [32], CUNE-BPR [31],

SAMN [5] and RML-DGATs [28]. We aim to show that, even with a simple hyperbolic distance model such as HSCML, hyperbolic space can still outperform most of the Euclidean baselines and have comparable performance with state-of-the-art methods.

6.1 Experiment Settings

First, we compare the performance of Euclidean space and hyperbolic space by comparing SCML and HSCML to evaluate the correctness of our conclusions in section 4. Negative sampling is not used. We report the HR@{1,5,10,15,20} and NDCG@{1,5,10,15,20}. We set λ to be 0.1, and m_{item} and m_{so} are tuned from 0 to 40 respectively.

Second, we compare HSCML with other baseline models. To keep consistent with the settings in [28], we set the latent size to be 128 and report HR@{10,20} & NDCG@{10,20}. Because they used negative sampling in their experiments, we also use negative sampling here. We randomly sample 999 negative samples for each user and rank them together with the test item. For HSCML, we set the λ to be 0.1. We tune m_{item} and m_{so} from 0 to 40 respectively. We repeat the experiment 10 times and report the average. Other parameters and settings are the same as we did in section 4 when evaluating the performance of hyperbolic space. The numbers of all methods except for SAMN and HSCML come from the paper of RML-DGATs [28]. We do so because the Ciao and Epinions they used are slightly different from ours, and we are unable to recover their datasets. Moreover, the implementation of RML-DGATs is not provided, so it is difficult for us to try our datasets on their model. What we can do is to run our datasets on the implementation of SAMN⁹ and report the numbers we get. Hopefully this can provide an insight on how hyperbolic space performs compared with state-of-the-art Euclidean methods.

6.2 Experimental Results

6.2.1 SCML vs. HSCML. Fig. 6 shows the results of SCML and its hyperbolic version HSCML. The conclusions in section 4 still hold here. For example, because SCML is a distance model, and the densities of Epinions and Ciao are low as shown in Table 2, so it is reasonable that the performance of hyperbolic space clearly outperforms Euclidean space on both two datasets. Moreover, as the latent size increases, the performance of Euclidean space gradually catches up with hyperbolic space, which is consistent with our conclusion regarding the influence of the latent size.

⁹<https://github.com/chenchongthu/SAMN>

Dataset	Metric	BPR	FISM	NeuMF	CML	LRML	SBPR	CUNE-BPR	RML-DGATs	SAMN	HSCML
Ciao	HR@10	0.2286	0.2076	0.2328	0.2302	0.2471	0.2364	0.2823	0.3108	<u>0.3098</u>	0.3052
	NDCG@10	0.1379	0.1207	0.1385	0.1365	0.1473	0.1446	0.1704	0.1803	0.2049	<u>0.2018</u>
	HR@20	0.3074	0.2847	0.3147	0.3129	0.3184	0.3282	0.3729	0.4157	<u>0.3863</u>	0.3844
	NDCG@20	0.1579	0.1403	0.1601	0.1563	0.1703	0.1625	0.1861	0.2066	0.2242	<u>0.2194</u>
Epinions	HR@10	0.4104	0.3842	0.4017	0.4118	0.4327	0.3749	0.4377	0.4625	0.4422	<u>0.4526</u>
	NDCG@10	0.2591	0.2421	0.2489	0.2562	0.2586	0.2436	0.2598	0.2728	<u>0.2892</u>	0.2971
	HR@20	0.5087	0.4850	0.5023	0.5051	0.5319	0.4868	<u>0.5583</u>	0.6018	0.5362	0.5481
	NDCG@20	0.2839	0.2676	0.2881	0.2972	0.3017	0.2653	0.2806	0.3080	<u>0.3129</u>	0.3189

Table 3: Top-n recommendation performance comparison. The best performance is in boldface and the second is underlined.

6.2.2 HSCML vs. baseline approaches. We compare the performance of HSCML with other baseline methods. The results are shown in Table 3. HSCML has a comparable performance with the state-of-the-art embedding methods such as SAMN and RML-DGATs, while outperforms other baseline approaches. This suggests that, hyperbolic embeddings can easily reach a comparable or better performance than existing baseline approaches with a simple distance model, whereas the state-of-the-art baselines such as SAMN and RML-DGATs are much more complicated because of the memory modules and GAT units. This demonstrates the superiority of hyperbolic space over Euclidean space. Future works to design a customized model for hyperbolic space may produce a model that outperforms all existing approaches.

7 RELATED WORK

Hyperbolic space has always been a popular research domain in Mathematics [2]. Some works have been done to explore the tree-like structure of graphs [1, 6] and the relations between hyperbolic space and hierarchical data such as languages and complex networks [18, 21]. Such works have demonstrated the consistency between real-world scale-free and hierarchical data and the hyperbolic space, providing theoretical basis for recent works which apply hyperbolic space to various tasks including link prediction, node classification, and recommendation.

Some researchers apply hyperbolic space to traditional metric learning approaches such as HyperBPR [26] and HyperML [27]. Some try to adopt hyperbolic space to neural networks and define hyperbolic neural network operations, producing powerful models such as hyperbolic neural networks [10], hyperbolic graph neural networks [19] and hyperbolic convolutional neural networks [4]. Meanwhile, [3] provides a scalable hyperbolic recommender system for industry use. [29] applies hyperbolic space to heterogeneous networks for link prediction task. [9] applies hyperbolic space to next-POI recommendation. [22] proposes a path-based recommendation approach with hyperbolic embeddings, etc.

8 CONCLUSION

In this paper, we provide a comprehensive analysis on hyperbolic space for recommender systems, comparing the performance of hyperbolic space with Euclidean space in three aspects: method, dataset, and latent size. To the best of our knowledge, this is the first work to address the advantages and disadvantages of hyperbolic space and give comments and suggestions on when and where to use it. Additionally, we propose SCML and its hyperbolic version HSCML, a distance model for social recommendation. Experiments show that hyperbolic space can easily reach a comparable or better

performance than existing Euclidean social recommendation methods with a simple distance model HSCML. A customized model for hyperbolic space may outperform all baselines in the future work.

REFERENCES

- [1] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. 2013. Tree-like structure in large social and information networks. In *ICDM*. IEEE, 1–10.
- [2] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. 1997. Hyperbolic geometry. *Flavors of geometry* 31 (1997), 59–115.
- [3] Benjamin Paul Chamberlain, Stephen R Hardwick, David R Wardrope, Fabon Dzogang, Fabio Daolio, and Saúl Vargas. 2019. Scalable hyperbolic recommender systems. *arXiv preprint arXiv:1902.08648* (2019).
- [4] Ines Chami, Zitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NIPS*. 4868–4879.
- [5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2019. Social attentional memory network: Modeling aspect-and friend-level differences in recommendation. In *WSDM*. 177–185.
- [6] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W Mahoney. 2013. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics* 9, 4 (2013), 434–491.
- [7] Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. 2018. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313* (2018).
- [8] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*. 417–426.
- [9] Shanshan Feng, Lucas Vinh Tran, Gao Cong, Lisi Chen, Jing Li, and Fan Li. 2020. HME: A Hyperbolic Metric Embedding Approach for Next-POI Recommendation. In *SIGIR*.
- [10] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *NIPS*. 5345–5355.
- [11] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, Vol. 15. 123–125.
- [12] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.
- [13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
- [14] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. 2002. Latent space approaches to social network analysis. *Journal of the American Statistical Association* 97, 460 (2002), 1090–1098.
- [15] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*. 193–201.
- [16] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD*. 659–667.
- [17] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of SIGKDD*. 1748–1757.
- [18] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010), 036106.
- [19] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. *arXiv preprint arXiv:1910.12892* (2019).
- [20] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*. 931–940.
- [21] Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS*. 6338–6347.
- [22] Nikolaos Papadis, Eleni Stai, and Vasileios Karyotis. 2017. A path-based recommendations approach for online systems via hyperbolic network embedding. In *ISCC*. IEEE.
- [23] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
- [24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* (2012).

- [25] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.
- [26] Tran Dang Quang Vinh, Yi Tay, Shuai Zhang, Gao Cong, and Xiao-Li Li. 2018. Hyperbolic recommender systems. *arXiv preprint arXiv:1809.01703* (2018).
- [27] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. HyperML: a boosting metric learning approach in hyperbolic space for recommender systems. In *WSDM*. 609–617.
- [28] Xiaodong Wang, Zhen Liu, Nana Wang, and Wentao Fan. 2020. Relational Metric Learning with Dual Graph Attention Networks for Social Recommendation. In *PAKDD*.
- [29] Xiao Wang, Yiding Zhang, and Chuan Shi. 2019. Hyperbolic heterogeneous information network embedding. In *AAAI*, Vol. 33. 5337–5344.
- [30] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*.
- [31] Chuxu Zhang, Lu Yu, Yan Wang, Chirag Shah, and Xiangliang Zhang. 2017. Collaborative User Network Embedding for Social Recommender Systems. In *SLAM*. 381–389. <https://doi.org/10.1137/1.9781611974973.43> arXiv:<https://epubs.siam.org/doi/pdf/10.1137/1.9781611974973.43>
- [32] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *CIKM*. 261–270.