


FHCF: Fully-Hyperbolic Symmetric Graph Learning for Collaborative Filtering

Guanghui Zhu()^{*}, Wang Lu, Chunfeng Yuan, and Yihua Huang

State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, China

{zgh, cfyuan, yhuang}@nju.edu.cn, luwang@smail.nju.edu.cn

Abstract. Hyperbolic graph neural networks (GNNs) have a natural advantage in modeling scale-free networks, which is a perfect fit for recommendation systems. However, existing recommendation systems based on hyperbolic GNNs often use the tangent space to achieve graph convolution on hyperbolic manifolds, which is inferior because the tangent space is only a local approximation of the manifold. In this paper, we propose a fully-hyperbolic symmetric graph neural network for collaborative filtering, named FHCF. First, to get rid of the limitation of the tangent space-based aggregation, we propose a fully-hyperbolic node aggregation operation that directly aggregates neighbors in the hyperbolic space by computing the Lorentzian centroid based on the squared Lorentzian distance. Then, to exploit the capacity of the hyperbolic space, we propose a hyperbolic space and node activity-aware push strategy to drive the node embeddings away from the origin to learn better embeddings in a wider space. Finally, to better train the model in the hyperbolic space, we further propose a symmetric hyperbolic ranking learning method, aiming at keeping negative items away from users and positive items. Extensive experimental results on three public datasets reveal that FHCF is effective and outperforms existing hyperbolic collaborative filtering models.

Keywords: Recommender system · Collaborative filtering · Hyperbolic space · Graph neural network.

1 Introduction

Recommender systems are widely used in many web applications to select items that can match users' preferences from a large number of candidate items. The core method behind recommender systems is collaborative filtering (CF)[22], which can produce effective recommendations from implicit feedback (e.g., clicks, views). Earlier typical work in CF such as matrix factorization (MF) [8] projects each user (item) into an embedding vector. More recently, due to the powerful representation learning ability, graph neural network (GNN) has been introduced to boost the performance of CF. Such collaborative filtering methods apply GNN [7, 27] to learn effective node representations from the user-item interaction graph and achieve state-of-the-art performance for recommendation.

Although GNN-based CF models work well, most of them are based on Euclidean space. However, in many domains such as recommender system, the data obeys a power-law distribution, i.e., the popular items and active users are the minority. The data with a power-law distribution has the property of an approximate tree hierarchy, and we can use Gromov's δ -hyperbolicity [6], a concept from group theory, to measure how tree-like a graph is. The smaller δ is, the more hyperbolic the graph dataset is, and $\delta = 0$ for the tree [2, 19]. Moreover, [19, 20] pointed out that it is difficult to provide a low-distortion embedding for tree-structured data in Euclidean space, even though we use an infinite embedding dimension. Fortunately, hyperbolic spaces as continuous versions of trees can be well embedded in the data with such structures.

Hyperbolic spaces, therefore, have recently received increasing attention from researchers. However, the hyperbolic graph neural networks used in the recommendation task are based on the transformation between the hyperbolic and tangent spaces to achieve neighbor aggregation [23, 28, 29]. Specifically, they first project the embedding from the hyperbolic space (\mathcal{H}^n) into the tangent space ($\mathcal{T}_{\mathbf{O}}\mathcal{H}^n$) at the origin \mathbf{O} ¹ using the logarithmic ($\log_{\mathbf{O}}(\cdot)$) transformation in the Lorentzian model, then use the GNN to aggregate the neighbor information in the tangent space, and finally use the exponential ($\exp_{\mathbf{O}}(\cdot)$) transformation to project the embedding from the tangent space back into the hyperbolic space. Figure 1(a) illustrates the process. However, we find two limitations to the above space transformation-based approach:

- **Approximation error:** First, the tangent space is a local approximation of the hyperbolic space [4]. Second, ideally, the neighbor aggregation operation for node i should be performed in the tangent space ($\mathcal{T}_i\mathcal{H}^n$) at node i . Nevertheless, this method is practically infeasible due to huge memory and computation overhead. Thus, existing methods perform aggregation operation in the tangent space ($\mathcal{T}_{\mathbf{O}}\mathcal{H}^n$) at the origin \mathbf{O} for all nodes. Overall, the mixture of tangent and hyperbolic spaces can hinder the model performance [3].
- **Model collapse:** Graph neural networks are usually multilayered. For layers $l-1$ and l , the exponential transformation of layer $l-1$ and the logarithmic transformation of layer l cancel each other out², causing the model to collapse to only one logarithmic transformation at the input, and one exponential transformation at the output [16]. As a result, the model degenerates into the plain Euclidean GNN approach.

Since the above limitations are caused by using the tangent space in neighbor aggregation, a more effective method that can perform neighbor aggregation directly in the hyperbolic space should be designed. However, there exists two challenges. First, the aggregation operation (e.g., Mean) in the Euclidean space cannot be used in the hyperbolic space. We need to design novel aggregation

¹ The origin \mathbf{O} is different in different hyperbolic models, for example in the Lorentzian model $\mathbf{O}_{\mathcal{H}} = (1, 0, \dots, 0)$ and in the Poincaré model $\mathbf{O}_{\mathcal{B}} = (0, \dots, 0)$

² $h = \log_{\mathbf{O}}^l(\exp_{\mathbf{O}}^{l-1}(h))$

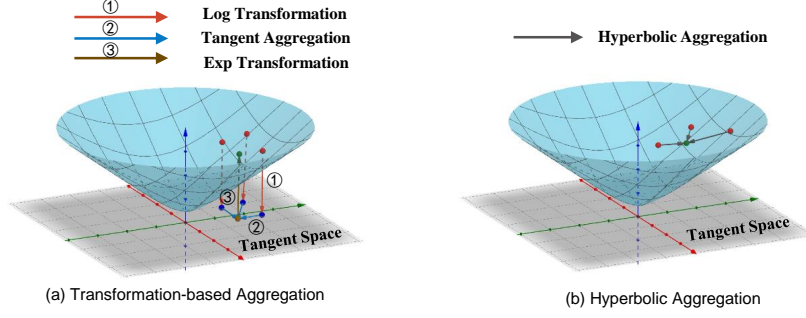


Fig. 1. (a) transformation-based neighbor aggregation; (b) the proposed direct hyperbolic aggregation without the tangent space.

operation specifically for the hyperbolic space. Second, the growth of the hyperbolic space is exponentially explosive (e.g., the perimeter is $2\pi \sinh r$ and the area is $2\pi(\cosh r - 1)$ ³ in the Poincaré disk [19], where r is the distance to the origin \mathcal{O}), leading to a larger space capacity the farther away from the origin \mathcal{O} ($r \rightarrow 1$). Exploiting the capacity in the hyperbolic space helps to improve the quality of node embeddings [29]. However, [23] recently pointed that forcing all nodes far away the origin \mathcal{O} is not appropriate. Nodes with high activity should be close to the origin \mathcal{O} and vice versa. Therefore, we should utilize the capacity of the hyperbolic space effectively when performing graph convolution.

To address these challenges, in this paper, we propose a fully-hyperbolic symmetric graph neural network for collaborative filtering, named FHCF. First, to get rid of the limitation of the tangent space-based aggregation, we propose a fully-hyperbolic node aggregation operation that directly aggregates neighbors in the hyperbolic space by computing the Lorentzian centroid based on the squared Lorentzian distance [12]. Figure 1 illustrates the difference between the existing transformation-based aggregation and the direct hyperbolic aggregation.

Second, to better utilize the space capacity in the hyperbolic space, we propose a hyperbolic space and node activity-aware push strategy to drive the node embeddings away from the origin \mathcal{O} to learn better embedding in a wider space. Different from the regularization approach used in [29] to force nodes to move away from the origin \mathcal{O} indefinitely, we leverage the good mathematical properties of the Poincaré model to push different nodes in the hyperbolic space at different distances while making their spatial growth consistent. Moreover, the proposed push strategy takes into account the node degree, ensuring that more active nodes push less distance, while others do the opposite.

Finally, we propose a symmetric hyperbolic ranking learning (SHRL) method to optimize the model. Compared with existing hyperbolic learning methods, SHRL has two benefits. First, there is no need to consider the margin setting in hyperbolic margin ranking learning. Second, SHRL is capable of keeping negative item away from users as well as positive items. In summary, our contributions can be summarized as follows:

³ $\sinh r = \frac{1}{2}(e^r - e^{-r})$, $\cosh r = \frac{1}{2}(e^r + e^{-r})$

- **Novel Framework.** We propose a novel fully-hyperbolic graph collaborative filtering framework that does not require the additional tangent space to complete the graph convolution process.
- **Novel Method.** To achieve fully-hyperbolic node aggregation, we propose a Lorentzian centroid based aggregation operation. To exploit the capacity of the hyperbolic space, we further propose a hyperbolic space and node activity-aware push strategy, which pushes nodes away from the origin \mathbf{O} with suitable distances to learn better embeddings.
- **Novel Learning Strategy.** To better train the model in the hyperbolic space, we propose a symmetric hyperbolic ranking learning method, aiming at keeping negative items away from users and positive items.
- **SOTA Results.** Extensive experimental results on three datasets reveal that FHCF is effective and outperforms the existing hyperbolic collaborative filtering models.

2 Related Work

2.1 GNN-based Recommendation Models

GNNs have achieved great success in recommendation tasks [7]. Particularly, GCN [11], as the most prevalent variant of GNNs, further fuels the development of graph neural recommendation models like NGCF [8] and LightGCN [7]. These GCN-driven models all follow a paradigm that aggregates information from neighbors layer by layer [7]. Among these models, LightGCN is the most popular due to its simple structure and excellent performance.

Most GNN-based recommendation models embed nodes in the Euclidean space, but it is difficult to provide low-distortion embeddings, although we can use an infinite embedding dimension [19, 20]. Recently, [23, 28, 29] proposed to use hyperbolic graph neural networks [2, 16] for recommendation and achieved significant results in various embedding dimensions.

2.2 Hyperbolic Neural Networks for Recommender Systems

HGNN [16] and HGCN [2] perform neighbor aggregation with the help of local tangent spaces. However, some researchers believe that this way of training in mixed spaces hinders the model training [3], and they start to explore the neighbor aggregation directly in the hyperbolic space. Typical models include LGCN [30], H2H-GCN [4], and HYBONET [3].

Recently, several researchers have started to use hyperbolic spaces for recommendations. Knowledge graph based recommender systems, HAKG [5] models the relationships in the knowledge graph into hyperbolic space to better utilize the knowledge graph for recommendation. HSR [13] performs social recommendation in a hyperbolic space to model high quality user and item embeddings. In the field of collaborative filtering, HGCF [23] can be regarded as a hyperbolic version of LightGCN [7]. HRCF [29] adds hyperbolic regularity to HGCF. The recent work HICF [28] proposes the hyperbolic adaptive margin and the hyperbolic hard negative sampling strategy.

3 PRELIMINARY

3.1 Hyperbolic Geometry

A Riemannian manifold (\mathcal{M}, g) is a differentiable manifold \mathcal{M} equipped with a metric tensor g . It can be locally approximated to a linear Euclidean space at an arbitrary point $\mathbf{x} \in \mathcal{M}$, and the approximated space is termed as a tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Hyperbolic spaces are smooth Riemannian manifolds with a constant negative curvature [4]. The hyperbolic space can be modeled using isometric models, such as the Lorentzian model, the Klein model, the Jemisphere model, the Poincaré ball model, and the Poincaré half-space model [20]. In this paper, we choose the Lorentzian model due to its numerical stability, when compared to the Poincaré model where the instability arises from the fraction [2].

Lorentzian model Formally, an n -dimensional Lorentzian model is the Riemannian manifold $\mathcal{L}^n = (\mathcal{H}^n, g_{\mathcal{L}})$, $g_{\mathcal{L}} = \text{diag}(-1, 1, \dots, 1)$ is the Riemannian metric tensor and \mathcal{H}^n is a point set satisfying:

$$\mathcal{H}^n = \{\mathbf{x} = (x_0, x_1, \dots, x_n) \in \mathcal{R}^{n+1} | \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}^4 \quad (1)$$

Lorentzian Inner Product $\langle \cdot, \cdot \rangle_{\mathcal{L}}: \mathcal{R}^{n+1} \times \mathcal{R}^{n+1} \rightarrow \mathcal{R}$ represents the Lorentzian inner product of point \mathbf{x} and point \mathbf{y} :

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^n x_i y_i \quad (2)$$

It is worth noting that $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -1$ iff $\mathbf{x} = \mathbf{y}$. Otherwise, $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} < -1$ for all pairs $(\mathbf{x}, \mathbf{y}) \in (\mathcal{H}^n)^2$.

Lorentzian distance: The distance for any pair of points \mathbf{x} and $\mathbf{y} \in \mathcal{H}^n$ is given by:

$$d_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \text{arcosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}) \quad (3)$$

Tangent Space: The tangent space centered at the point $\mathbf{x} \in \mathcal{H}^n$ in the Lorentzian manifold is then defined by:

$$\mathcal{T}_{\mathbf{x}}\mathcal{H}^n = \{\mathbf{v} \in \mathcal{R}^{n+1} | \langle \mathbf{v}, \mathbf{x} \rangle_{\mathcal{L}} = 0\} \quad (4)$$

Exponential and Logarithmic Maps: Mapping between the tangent space and the hyperbolic space is done by exponential and logarithmic maps. Given $\mathbf{x} \in \mathcal{H}^n$ and a tangent vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{H}^n$, the exponential map $\exp_{\mathbf{x}}: \mathcal{T}_{\mathbf{x}}\mathcal{H}^n \rightarrow \mathcal{H}^n$ ($\mathbf{v} \neq \mathbf{0}$):

$$\exp_{\mathbf{x}}(\mathbf{v}) = \cosh(\|\mathbf{v}\|_{\mathcal{L}})\mathbf{x} + \sinh(\|\mathbf{v}\|_{\mathcal{L}})\frac{\mathbf{v}}{\|\mathbf{v}\|_{\mathcal{L}}} \quad (5)$$

Where $\|\mathbf{v}\|_{\mathcal{L}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathcal{L}}}$, and the logarithmic map is the reverse map $\log_{\mathbf{x}}: \mathcal{H}^n \rightarrow \mathcal{T}_{\mathbf{x}}\mathcal{H}^n$ ($\mathbf{x} \neq \mathbf{y}$):

$$\log_{\mathbf{x}}(\mathbf{y}) = d_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) \frac{\mathbf{y} + \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \mathbf{x}}{\|\mathbf{y} + \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \mathbf{x}\|_{\mathcal{L}}} \quad (6)$$

⁴ For convenience of presentation, we set the curvature to -1.

Poincaré model Since we only need to use the mathematical properties of the Poincaré model in this paper, we briefly describe it here, see [20] for details. The Poincaré model is given by projecting each point of \mathcal{H}^n onto the hyperplane $x_0 = 0$, using the rays emanating from $(-1, 0, \dots, 0)$. The Poincaré model \mathcal{B} is a manifold equipped with a Riemannian metric $g_{\mathcal{B}}$. This metric is conformal to the Euclidean metric g_E with the conformal factor $\lambda_{\mathbf{x}} = \frac{2}{1-\|\mathbf{x}\|^2}$, $g_{\mathcal{B}} = \lambda_{\mathbf{x}} g_E$ and $g_E = (1, 1, \dots, 1)$. Formally, an n dimensional Poincaré unit ball (manifold) is:

$$\mathcal{B}^n = \{\mathbf{x} \in \mathcal{R}^n \mid \|\mathbf{x}\| < 1\} \quad (7)$$

where $\|\cdot\|$ denotes the Euclidean norm. Distances in \mathcal{B}^n are measured via the following function:

$$d_{\mathcal{B}(\mathbf{x}, \mathbf{y})} = \text{arcosh}\left(1 + 2 \frac{\|\mathbf{x} - \mathbf{y}\|^2}{(1 - \|\mathbf{x}\|^2)(1 - \|\mathbf{y}\|^2)}\right) \quad (8)$$

The points in the Poincaré model and the Lorentzian model can be converted to each other. The time complexity of this conversion operation is $\mathbf{O}(n)$ for each n -dimensional node embedding, and its time overhead is negligible compared to other operations.

$$\begin{aligned} p_{\mathcal{L} \rightarrow \mathcal{B}}(\mathbf{x}) &= p_{\mathcal{L} \rightarrow \mathcal{B}}(x_0, x_1, \dots, x_n) = \frac{(x_1, \dots, x_n)}{x_0 + 1} \\ p_{\mathcal{B} \rightarrow \mathcal{L}}(\mathbf{x}) &= p_{\mathcal{B} \rightarrow \mathcal{L}}(x_1, \dots, x_n) = \frac{(1 + \sum_{i=1}^n x_i^2, 2x_1, \dots, 2x_n)}{1 - \sum_{i=1}^n x_i^2} \end{aligned} \quad (9)$$

3.2 Recommendation Task

We define the user set $\mathcal{U} = \{u\}$ and the item set $\mathcal{I} = \{i\}$, the observed implicit feedback matrix is denoted as $\mathcal{R} \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$, where each entry $\mathcal{R}_{u,i} = 1$ if there exists an interaction between user u and item i , otherwise $\mathcal{R}_{u,i} = 0$.

Let \mathcal{G} denote the user-item interaction graph:

$$\mathcal{G} = \begin{pmatrix} 0 & \mathcal{R} \\ \mathcal{R}^T & 0 \end{pmatrix} \quad (10)$$

Let $\mathcal{N}_{\mathcal{G}}(i)$ denote the set of first-order neighbors of node i in graph \mathcal{G} . Following [22], given the user-item interactions matrix \mathcal{R} , the goal of collaborative filtering is to predict the unobserved interactions in \mathcal{R} . i.e., the probability of a targeted user u clicking or purchasing an unobserved candidate item i .

4 The Proposed Methodology

Figure 2 shows the overall framework of the proposed FHCF. We first initialize the embeddings of all users and items in the hyperbolic space. Then, in each GNN layer, we perform fully-hyperbolic node aggregation and activity-aware node push in the hyperbolic space. Next, the layer aggregation is performed, which aggregates the embeddings of each GNN layer to get the final embedding. Finally, a symmetric hyperbolic ranking learning method is proposed.

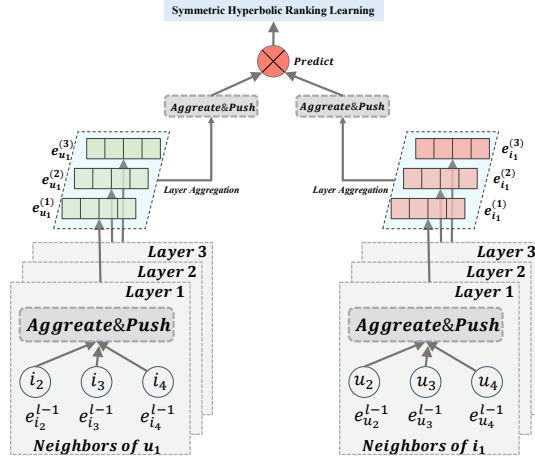


Fig. 2. The overall framework of FHCF.

4.1 Hyperbolic Embedding Initialization

Follow [23], we first randomly initialize the embedding in the tangent space at the origin $\mathbf{O}_{\mathcal{H}} = (1, 0, \dots, 0) \in \mathcal{H}^n$:

$$\boldsymbol{\theta}'_u = [0; \boldsymbol{\theta}_u] \quad \boldsymbol{\theta}'_i = [0; \boldsymbol{\theta}_i] \quad (11)$$

where $[\cdot]$ denotes concatenation. Both $\boldsymbol{\theta}'_u$ and $\boldsymbol{\theta}'_i$ satisfy $\langle \boldsymbol{\theta}'_u, \mathbf{O}_{\mathcal{H}} \rangle_{\mathcal{L}} = 0$ and $\langle \boldsymbol{\theta}'_i, \mathbf{O}_{\mathcal{H}} \rangle_{\mathcal{L}} = 0$ and therefore belong to $\mathcal{T}_{\mathbf{O}_{\mathcal{H}}} \mathcal{H}^n$. Then, we project the initialized embedding in the tangent space $(\boldsymbol{\theta}'_u, \boldsymbol{\theta}'_i)$ into the hyperbolic space via the exponential map $\exp_{\mathbf{O}_{\mathcal{H}}}(\cdot) : \mathcal{T}_{\mathbf{O}_{\mathcal{H}}} \mathcal{H}^n \rightarrow \mathcal{H}^n$. The user and item embeddings in \mathcal{H}^n are as follows:

$$e_u^0 = \exp_{\mathbf{O}_{\mathcal{H}}}(\boldsymbol{\theta}'_u) \quad e_i^0 = \exp_{\mathbf{O}_{\mathcal{H}}}(\boldsymbol{\theta}'_i) \quad (12)$$

4.2 Fully-Hyperbolic Node Aggregation

To better understand the proposed model based on fully-hyperbolic node aggregation, we first introduce the following definition and theorem.

Definition 1. (Squared Lorentzian distance) The squared Lorentzian distance [21] is defined for all pair $\mathbf{x} \in \mathcal{H}^n, \mathbf{y} \in \mathcal{H}^n$ as:

$$d_{\mathcal{L}}^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_{\mathcal{L}}^2 = -2 - 2\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \quad (13)$$

It satisfies all the axioms of a distance metric except the triangle inequality.

Theorem 1. (Centroid with the squared Lorentzian distance) The expression for the point $\mathbf{u} \in \mathcal{H}^n$ that minimizes the problem $\min_{\mathbf{u} \in \mathcal{H}^n} \sum_{i=1}^n v_i d_{\mathcal{L}}^2(\mathbf{x}_i, \mathbf{u})$ where $\forall i, \mathbf{x}_i \in \mathcal{H}^n, v_i \geq 0, \sum_{i=1}^n v_i > 0$ is given by:

$$\mathbf{u} = \frac{\sum_{i=1}^n v_i \mathbf{x}_i}{\|\sum_{i=1}^n v_i \mathbf{x}_i\|_{\mathcal{L}}} \quad (14)$$

where $\|\mathbf{a}\|_{\mathcal{L}} = \sqrt{\|\mathbf{a}\|_{\mathcal{L}}^2}$, the detailed proof can be found in [12].

According to the above theorem and definition, we leverage the Lorentzian centroid of the neighboring nodes based on the square Lorentzian distance in the hyperbolic space to perform node aggregation. Specifically, let v represent a user or item node and \mathcal{N}_v is the set of first-order neighbors of node v . We replace v_i in Eq.(14) with the normalization coefficient in GNN and x_i with the node embedding of each neighbor. The node aggregation operation in the hyperbolic space can be expressed as follows:

$$h_v^l = f_{agg}(\mathcal{N}_v \cup \{v\}) = \frac{\sum_{j \in \mathcal{N}_v \cup \{v\}} \frac{1}{|\mathcal{N}_v \cup \{v\}|} e_j^{l-1}}{\|\sum_{j \in \mathcal{N}_v \cup \{v\}} \frac{1}{|\mathcal{N}_v \cup \{v\}|} e_j^{l-1}\|_{\mathcal{L}}} \quad (15)$$

Other choices for the aggregation in the Lorentzian model include Fréchet mean [10] and Einstein midpoint [25]. The main reason why we adopt the Lorentzian centroid is twofold. First, the Lorentzian centroid is fast to compute and can be seen as Frechet mean in the pseudo-hyperbolic space [12]. The Fréchet mean is the classical generalization of Euclidean mean. However, it offers no closed-form solution. Solving the Fréchet mean requires iterative computation [17], which significantly slows down the training and inference, making it impossible to generalize to deep and large models. Second, the Lorentzian centroid has good theoretical property, which can minimize the sum of squared Lorentzian distance. The computation of the Einstein midpoint requires transformation between the Lorentzian and Klein models, bringing in numerical instability. Also, whether the Einstein midpoint in the Klein model has the geometric interpretation in the Lorentzian model remains to be investigated.

4.3 Hyperbolic Space and Node Activity-Aware Push

According to [29], pushing nodes away from the origin \mathbf{O} is very effective for the recommendation task, but it is not as convenient to push nodes away directly in the hyperbolic space as in the Euclidean space. For example, we can push nodes away from the origin by multiplying them by a factor ($\mathbf{e}_1 \Rightarrow \gamma \mathbf{e}_1$) in the Euclidean space, but this is infeasible in the hyperbolic space due to the following problems. First, the new node is not in the hyperbolic space (i.e., $\langle \gamma \mathbf{e}_1, \gamma \mathbf{e}_1 \rangle_{\mathcal{L}} \neq -1$). Second, imposing the same scale (γ) on all nodes does not take into account the fact that the spatial capacity grows exponentially with radius r when the nodes are in different positions of the hyperbolic space. This is different from the Euclidean space where the spatial capacity grows polynomially. As a result, the spatial growth of nodes located at hyperbolic boundaries is exponentially larger than that of nodes located near the origin. Giving each node the same scale (γ) does not fit in the hyperbolic space.

It is well known that the Poincaré model has good mathematical properties. On one hand, in the Poincaré model, we only need to restrict the distance between the node and the origin \mathbf{O} to satisfy the definition of the hyperbolic space. On the other hand, the hyperbolic volume of the shell at a point with distance r from \mathbf{O} is at most $Ae^{r(n-1)}$, where A is a positive constant [18].

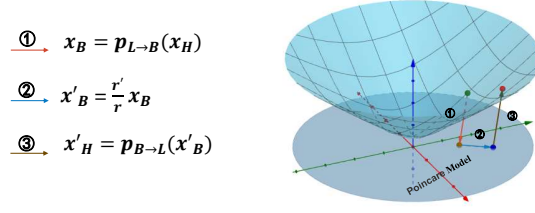


Fig. 3. Hyperbolic space and node activity-aware push strategy.

Therefore, based on the above mathematical properties, we can push the nodes away from the original with appropriate distances while satisfying the definition of the hyperbolic space.

To prevent the space growth of different nodes in the hyperbolic space from being too disparate, we expect the space growth generated by each node to be the same after pushing. To this end, we employ the Poincaré model to push the node at distance r from the origin \mathbf{O} to a point at distance r' from \mathbf{O} , making the space capacity grow d' .

$$Ae^{r'(n-1)} - Ae^{r(n-1)} = d' \Rightarrow e^{r'(n-1)} - e^{r(n-1)} = d \Rightarrow r' = \frac{1}{n-1} \log(d + e^{r(n-1)}) \quad (16)$$

where $d = \frac{d'}{A}$. By using Eq.(16), we can push different nodes in the hyperbolic space at different distances while making their spatial growth consistent. Thus, to exploit the capacity of the hyperbolic space in each GNN layer, we are able to push the nodes in the Lorentzian model as far away from the origin \mathbf{O} as possible.

However, [23] demonstrated that forcing all nodes far away the origin \mathbf{O} is not appropriate. The more active nodes should be close to the origin \mathbf{O} , which is self-evident with the property that the hyperbolic space can model hierarchical relations [19]. Therefore, to accelerate the model training and make full use of the prior knowledge of the data, we propose a node activity-aware push method. Specifically, it is expected that the more active nodes push less distance, while others do the opposite.

In Eq.(16), we know that $e^{r(n-1)}$ exhibits an exponential relationship with respect to the embedding dimension n , leading to a large value. Thus, d must be particularly large to have an effect on r . Therefore, to make better use of the activity information of the nodes, we rewrite the above formula, where \mathcal{D} represents the degree of the node, and γ is a hyperparameter to control the weight of node degree.

$$r' = \frac{1}{n-1} \log(e^{r(n-1)}(1 + d^*)) = r + \frac{1}{n-1} \log(1 + d^*) \Rightarrow d^* = \frac{\gamma}{\log(\mathcal{D})} \quad (17)$$

With the theoretical basis for pushing the nodes in the Poincaré model, we next give how to push the nodes in the Lorentzian model. Let $\mathbf{x}_{\mathcal{H}} \in \mathcal{H}^n$ the points in the Lorentzian model, $\mathbf{x}_{\mathcal{B}} \in \mathcal{B}^n$ denote the points in the Poincaré model.

Figure 3 illustrates the hyperbolic space and node activity-aware push process. We leverage the mathematical properties of the Poincaré model by first

projecting points in the Lorentzian model into the Poincaré model, then moving these points along the radial direction in the Poincaré model so that their radii grow from r to r' , and finally projecting them back into the Lorentzian model. To ensure that the node remains in the hyperbolic space after the push, we constrain the radius r' in the Poincaré model so that the node is always in the hyperbolic space. After hyperbolic space and node activity-aware push, the node embedding is given by Eq.(18).

$$e_v^l = f_{push}(h_v^l) \quad (18)$$

4.4 Symmetric Hyperbolic Ranking Learning

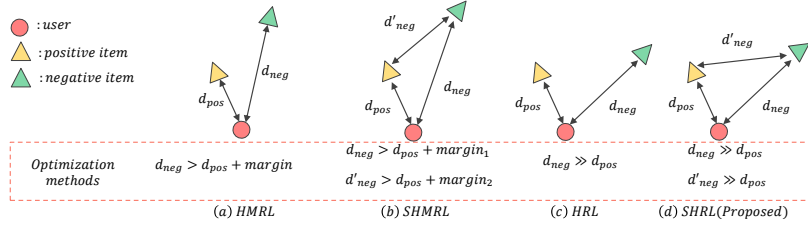


Fig. 4. Illustration of different metric learning in the hyperbolic space.

After the propagation of L layers, we get $\{e_v^1, e_v^2, \dots, e_v^L\}$. To integrate all the layers for better recommendation, we use $f_{agg}(\cdot)$ to aggregate them. Finally, $f_{push}(\cdot)$ is used to push the embedding in the hyperbolic space.

$$e_v = f_{push}(f_{agg}(\{e_v^1, e_v^2, \dots, e_v^L\})) \quad (19)$$

The hyperbolic margin ranking learning (HMLR) is commonly used in [23, 28, 29]. Because the margin of HMLR is very difficult to set, we usually need to keep adjusting its value during the model training. Therefore, we leverage hyperbolic ranking learning (HRL) to optimize the model, which takes the following form:

$$\mathcal{L}_D = \sum_{(u,i) \in \mathcal{B}} \log \frac{\exp(-d_{\mathcal{L}}^2(e_u, e_i)/\tau)}{\sum_{j \in \mathcal{N}(u) \cup \{i\}} \exp(-d_{\mathcal{L}}^2(e_u, e_j)/\tau)} \quad (20)$$

where $d_{\mathcal{L}}^2(\cdot)$ represents the squared Lorentzian distance, τ is the temperature coefficient, \mathcal{B} represents the set of user-item interaction pairs in a batch, and $\mathcal{N}(u)$ is the set of negative samples for each interaction pair, i.e., $\mathcal{N}(u) = \{j | \mathcal{R}_{u,j} = 0\}$.

Based on HRL, we can make the distance between the user and positive items much smaller than the distance between the user and negative items, which allows us to avoid the hassle of HMRL requiring constant margin adjustment during the training process. However, both HRL and HMRL only consider pushing the negative items away from the user during model training, but this may push them near the positive items, which is obviously inappropriate.

Inspired by [14], we propose a symmetric hyperbolic ranking learning (SHRL) method. The loss function \mathcal{L}_S of SHRL is shown in Eq. (21). Figure 4 shows the difference between the proposed SHRL and other hyperbolic learning methods including HMRL and HRL.

$$\mathcal{L}_S = \sum_{(u,i) \in \mathcal{B}} \log \frac{\exp(-d_{\mathcal{L}}^2(e_i, e_u)/\tau)}{\sum_{j \in \mathcal{N}(u) \cup \{u\}} \exp(-d_{\mathcal{L}}^2(e_i, e_j)/\tau)} \quad (21)$$

In summary, the overall loss of FHCF is composed of \mathcal{L}_D and \mathcal{L}_S :

$$\mathcal{L} = \mathcal{L}_D + \lambda_1 \mathcal{L}_S + \lambda_2 \|\Theta\|_2 \quad (22)$$

where λ_1 determines the weight of hyperbolic symmetry ranking learning, and λ_2 controls the regularization term. Θ denotes the parameters of the GNN model.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We use three real-world datasets to evaluate FHCF, i.e., Amazon-CD, Amazon-Book, and Yelp2020. These datasets vary in domains, scale, and density. Following [23, 28, 29], we randomly select 80 % of interactions as training data and 20 % of interactions for performance comparison. We uniformly sample N negative items for each positive instance to generate the training set.

Baselines. We compare FHCF with the 10 baseline models, including matrix factorization-based WRMF [9], variant autoencoder-based VAECF [15], metric learning-based models such as LRML [24], SML [14], GNN-based models such as NGCF [27], LightGCN [7], hyperbolic GNN-based models such as HGCF [23], HRCF [29], LGCF [26], HICF [28].

Evaluation Metrics. To evaluate the performance of top-K recommendation, we adopt two widely used metrics Recall@K and NDCG@K, where K is set to 10 and 20 for consistency. Following [23], we adopt the full-ranking strategy, which ranks all candidate items that the user has not interacted with.

Implementation Details. To keep the comparison fair, we closely follow the settings of HGCF [23]. More specifically, the embedding size is set to 50 and the total number of training epochs are fixed as 500. We utilize the Riemannian SGD [1] to learn the network parameters. RSGD mimics the stochastic gradient descent optimization while taking into account the geometry of the hyperbolic manifold [1]. We tune the hyperparameters λ_1 in [0.1,1], λ_2 in [1e-5,1e-3], τ in [0.1,0.5], N in [10,200], and γ in [0.1,0.6], where N is the number of negative samples. For the baseline settings, we refer to [23].

5.2 Overall Performance Comparison

Table 1 shows the overall performance comparison. The proposed FHCF outperforms existing models based on the Euclidean space and the hyperbolic space on

Table 1. Performance comparison between different collaborative filtering models. The best result is **bolded** and the runner-up is underlined. * denotes statistically significant improvement (measured by t-test with p -value < 0.005) over the best baselines.

Datasets	Metric	WRMF	VAECF	LRML	SML	NGCF	LightGCN	HGCF	HRCF	LGCF	HICF	FHCF
Amazon-CD	Recall@10	0.0863	0.0786	0.0502	0.0475	0.0758	0.0929	0.0962	0.1003	0.0996	<u>0.1079</u>	0.1108*
	NDCG@10	0.0651	0.0615	0.0405	0.0361	0.0591	0.0726	0.0751	0.0785	0.0780	<u>0.0848</u>	0.0872*
	Recall@20	0.1313	0.1155	0.0771	0.0734	0.1150	0.1404	0.1455	0.1503	0.1503	<u>0.1586</u>	0.1640*
	NDCG@20	0.0817	0.0752	0.0492	0.0456	0.0718	0.0881	0.0909	0.0947	0.0945	<u>0.1010</u>	0.1045*
Amazon-Book	Recall@10	0.0623	0.0740	0.0522	0.0479	0.0658	0.0799	0.0867	0.0900	0.0899	<u>0.0965</u>	0.1025*
	NDCG@10	0.0563	0.0716	0.0515	0.0422	0.0655	0.0780	0.0869	0.0902	0.0906	<u>0.0978</u>	0.1040*
	Recall@20	0.0919	0.1066	0.0834	0.0768	0.1050	0.1248	0.1318	0.1364	0.1360	<u>0.1449</u>	0.1526*
	NDCG@20	0.0730	0.0878	0.0626	0.055	0.0791	0.0938	0.1022	0.1060	0.1063	<u>0.1142</u>	0.1208*
Yelp2020	Recall@10	0.0470	0.0429	0.0326	0.0319	0.0458	0.0522	0.0527	0.0537	<u>0.0573</u>	0.0570	0.0606*
	NDCG@10	0.0372	0.0353	0.0287	0.0255	0.0405	0.0461	0.0458	0.0468	0.0485	<u>0.0502</u>	0.0534*
	Recall@20	0.0793	0.0706	0.0562	0.0544	0.0764	0.0866	0.0884	0.0898	0.0946	<u>0.0948</u>	0.1001*
	NDCG@20	0.0506	0.0469	0.0369	0.0347	0.0513	0.0582	0.0585	0.0594	0.0612	<u>0.0633</u>	0.0671*

Table 2. Ablation study in FHCF.

Dataset	Metric	w/o HSP	w/o HA	w/o SHRL	FHCF
Amazon-CD	Recall@10	0.0655	0.0992	0.1079	0.1108
	NDCG@10	0.0477	0.0789	0.0852	0.0872
	Recall@20	0.1078	0.1480	0.1608	0.1640
	NDCG@20	0.0617	0.0947	0.1023	0.1045
Yelp2020	Recall@10	0.0212	0.0533	0.0576	0.0606
	NDCG@10	0.0180	0.0471	0.0509	0.0534
	Recall@20	0.0399	0.0894	0.0964	0.1001
	NDCG@20	0.0248	0.0596	0.0642	0.0671

all datasets, verifying the effectiveness of FHCF. Among them, NGCF [8] uses GNNs to learn higher-order information of nodes, which outperforms traditional MF and metric-based models, but is less effective than LightGCN [7] due to the complex linear transformations and activation functions. These models are all based on the Euclidean space. HGCF [23], HRCF [29], and HICF [28] exploit the nature of hyperbolic space to improve model performance. Unlike existing hyperbolic graph learning methods, FHCF proposes a fully-hyperbolic symmetric graph neural network and can achieve the best performance.

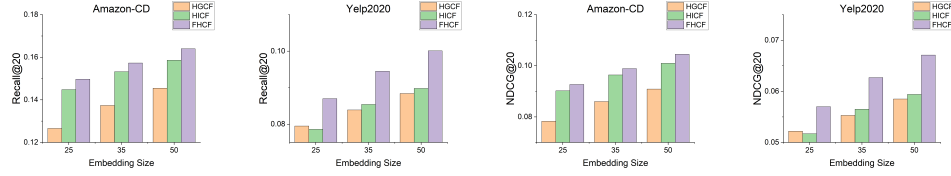
5.3 Ablation Study of FHCF

We conduct an ablation study to evaluate the effectiveness of each component in FHCF. We remove Hyperbolic Space and Node Activity-Aware Push (**HSP**), Symmetry Hyperbolic Ranking Learning (**SHRL**), and Hyperbolic Aggregation (**HA**) separately. When SHRL is removed, we use HRL to train the model. And when HA is removed, we use the tangent space for neighbor aggregation.

Table 2 shows that removing either module leads to degradation of the model performance, indicating the effectiveness of each component. In particular, the performance drops dramatically when removing HSP, which illustrates the importance of moving all nodes away from the origin \mathbf{O} in the hyperbolic space. Notably, when removing SHRL, our results are similar to those of HICF. HICF achieves performance improvement with its proposed hard negative sampling

Table 3. Performance comparison between different hyperbolic learning methods.

Dataset	Metric	HMRL	SHMRL	HRL	SHRL
Amazon-CD	Recall@10	0.0962	0.1001	0.1079	0.1108
	NDCG@10	0.0760	0.0798	0.0852	0.0872
	Recall@20	0.1470	0.1519	0.1608	0.1640
	NDCG@20	0.0925	0.0963	0.1023	0.1045
Yelp2020	Recall@10	0.0316	0.0408	0.0576	0.0606
	NDCG@10	0.0285	0.0367	0.0509	0.0534
	Recall@20	0.0548	0.0679	0.0964	0.1001
	NDCG@20	0.0367	0.046	0.0642	0.0671

**Fig. 5.** Performance comparison with different embedding sizes.

and adaptive margin. In this paper, we propose a completely different method that performs fully-hyperbolic node aggregation to get better performance.

5.4 Study on Hyperbolic Learning Methods

We further compare the impacts of different hyperbolic learning methods based on FHCF, including hyperbolic margin ranking learning (HMRL), symmetric hyperbolic margin ranking learning (SHMRL), hyperbolic ranking learning (HRL), and symmetric hyperbolic ranking learning (SHRL). Table 3 shows that the proposed SHRL outperforms other hyperbolic learning methods on Amazon-CD and Yelp2020, indicating the effectiveness of SHRL.

5.5 Generalization w.r.t. Embedding sizes

The smaller embedding size reduces the computation burden but degrades the performance. Figure 5 shows that FHCF consistently outperforms HGCF and HICF for all scenarios and the advantage is even more obvious in the case of Amazon-CD with the smallest dimension size 25.

5.6 Hyperparameter Sensitivity Analysis

Impact of τ . The temperature τ plays an important role in the loss function of FHCF (i.e., Eq.(20) and Eq.(21)). We vary τ in the range of 0.1 to 0.5. Figure 6 shows that too low or too high values of τ can lead to poor performance. In general, temperatures around 0.2 can lead to good recommended performance.

Impact of γ . γ defined in Eq.(17) controls how far the node is from the origin O . We vary γ in the range of 0.1 to 0.6. When γ is small, the nodes are still

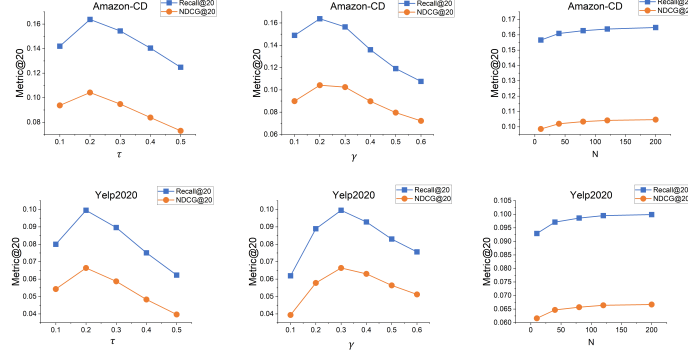


Fig. 6. Impact of τ , γ , N (Recall@20 and NDCG@20).

near the origin \mathbf{O} and thus FHCF cannot make good use of the capacity of the hyperbolic space. However, the model performance decreases when γ is too large. The optimal γ is related to the size of the dataset, because the larger the dataset, the larger the number of nodes, and the larger the required space capacity.

Impact of N . Figure 6 shows that the model performance increases with the number of negative samples, but the performance improvement is not significant. Considering the model training efficiency, we choose 100 as a compromise value.

5.7 Convergence Speed w.r.t Training Epochs

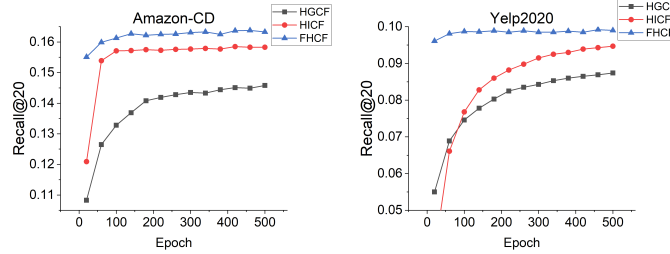


Fig. 7. Training convergence speed (Recall@20).

Following [23, 28, 29], we explore the convergence rate of FHCF during the training. As shown in Figure 7, FHCF repeatedly outperforms the baseline models in all epochs and can achieve the highest performance in fewer epochs, indicating that FHCF can significantly speed up the training process.

6 CONCLUSION

In this paper, we proposed a fully-hyperbolic symmetric graph neural network for collaborative filtering, named FHCF. To better exploit the capacity of hy-

perbolic space, we proposed a hyperbolic space and node activity-aware push strategy to drive the nodes to the suitable space. Finally, we proposed a symmetric hyperbolic ranking learning method to optimize the model, which pushes negative items away from the users and positive items at the same time. Extensive experimental results on three datasets reveal that FHCF is effective and outperforms the existing hyperbolic collaborative filtering methods.

Acknowledgement. This work was supported by the Frontier Technology R&D Program of Jiangsu Province (#BF2024005), Nanjing Science and Technology Program (#202304016), and the Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China.

References

1. Bonnabel, S.: Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control* **58**(9), 2217–2229 (2013)
2. Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems* **32** (2019)
3. Chen, W., Han, X., Lin, Y., Zhao, H., Liu, Z., Li, P., Sun, M., Zhou, J.: Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686* (2021)
4. Dai, J., Wu, Y., Gao, Z., Jia, Y.: A hyperbolic-to-hyperbolic graph convolutional network. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 154–163 (2021)
5. Du, Y., Zhu, X., Chen, L., Zheng, B., Gao, Y.: Hakg: Hierarchy-aware knowledge gated network for recommendation. *arXiv preprint arXiv:2204.04959* (2022)
6. Gromov, M.: *Hyperbolic groups*. Springer (1987)
7. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: Simplifying and powering graph convolution network for recommendation. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. pp. 639–648 (2020)
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: *Proceedings of the 26th international conference on world wide web*. pp. 173–182 (2017)
9. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: *2008 Eighth IEEE international conference on data mining*. pp. 263–272. Ieee (2008)
10. Karcher, H.: Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics* **30**(5), 509–541 (1977)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016)
12. Law, M., Liao, R., Snell, J., Zemel, R.: Lorentzian distance learning for hyperbolic representations. In: *International Conference on Machine Learning*. pp. 3672–3681. PMLR (2019)
13. Li, A., Yang, B., Hussain, F.K., Huo, H.: Hsr: hyperbolic social recommender. *Information Sciences* **585**, 275–288 (2022)
14. Li, M., Zhang, S., Zhu, F., Qian, W., Zang, L., Han, J., Hu, S.: Symmetric metric learning with adaptive margin for recommendation. In: *Proceedings of the AAAI conference on artificial intelligence*. vol. 34, pp. 4634–4641 (2020)

15. Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 world wide web conference. pp. 689–698 (2018)
16. Liu, Q., Nickel, M., Kiela, D.: Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems* **32** (2019)
17. Lou, A., Katsman, I., Jiang, Q., Belongie, S., Lim, S.N., De Sa, C.: Differentiating through the fréchet mean. In: International Conference on Machine Learning. pp. 6393–6403. PMLR (2020)
18. Lyons, R., Peres, Y.: Probability on trees and networks, vol. 42. Cambridge University Press (2017)
19. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems* **30** (2017)
20. Peng, W., Varanka, T., Mostafa, A., Shi, H., Zhao, G.: Hyperbolic deep neural networks: A survey. *arXiv preprint arXiv:2101.04562* (2021)
21. Ratcliffe, J.G., Axler, S., Ribet, K.: Foundations of hyperbolic manifolds, vol. 149. Springer (1994)
22. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The adaptive web, pp. 291–324. Springer (2007)
23. Sun, J., Cheng, Z., Zuberi, S., Pérez, F., Volkovs, M.: Hgcf: Hyperbolic graph convolution networks for collaborative filtering. In: Proceedings of the Web Conference 2021. pp. 593–601 (2021)
24. Tay, Y., Anh Tuan, L., Hui, S.C.: Latent relational metric learning via memory-based attention for collaborative ranking. In: Proceedings of the 2018 world wide web conference. pp. 729–739 (2018)
25. Ungar, A.A.: Analytic hyperbolic geometry: Mathematical foundations and applications. World Scientific (2005)
26. Wang, L., Hu, F., Wu, S., Wang, L.: Fully hyperbolic graph convolution network for recommendation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 3483–3487 (2021)
27. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. pp. 165–174 (2019)
28. Yang, M., Li, Z., Zhou, M., Liu, J., King, I.: Hicf: Hyperbolic informative collaborative filtering. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 2212–2221 (2022)
29. Yang, M., Zhou, M., Liu, J., Lian, D., King, I.: Hrcf: Enhancing collaborative filtering via hyperbolic geometric regularization. In: Proceedings of the ACM Web Conference 2022. pp. 2462–2471 (2022)
30. Zhang, Y., Wang, X., Shi, C., Liu, N., Song, G.: Lorentzian graph convolutional networks. In: Proceedings of the Web Conference 2021. pp. 1249–1261 (2021)