# Geometric Disentangled Collaborative Filtering

Yiding Zhang*
zyd@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chaozhuo Li*
cli@microsoft.com
Microsoft Research Asia
Beijing, China

Xing Xie
xing.xie@microsoft.com
Microsoft Research Asia
Beijing, China

Xiao Wang
xiaowang@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi†
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Yuming Liu
yumliu@microsoft.com
Microsoft
Beijing, China

Hao Sun
hasun@microsoft.com
Microsoft
Beijing, China

Liangjie Zhang
liazha@microsoft.com
Microsoft
Beijing, China

Weiwei Deng
dedeng@microsoft.com
Microsoft
Beijing, China

Qi Zhang
qizhang@microsoft.com
Microsoft
Beijing, China

## ABSTRACT

Learning informative representations of users and items from the historical interactions is crucial to collaborative filtering (CF). Existing CF approaches usually model interactions solely within the Euclidean space. However, the sophisticated user-item interactions inherently present highly non-Euclidean anatomy with various types of geometric patterns (i.e., tree-likeness and cyclic structures). The Euclidean-based models may be inadequate to fully uncover the intent factors beneath such hybrid-geometry interactions. To remedy this deficiency, in this paper, we study the novel problem of Geometric Disentangled Collaborative Filtering (GDCF), which aims to reveal and disentangle the latent intent factors across multiple geometric spaces. A novel generative GDCF model is proposed to learn geometric disentangled representations by inferring the high-level concepts associated with user intentions and various geometries. Empirically, our proposal is extensively evaluated over five real-world datasets, and the experimental results demonstrate the superiority of GDCF.

*Both authors contributed equally to this research.
†Chuan Shi is a corresponding author.


*SIGIR '22, July 11–15, 2022, Madrid, Spain.*

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531982

## CCS CONCEPTS

• **Information systems** → **Collaborative filtering**.

## KEYWORDS

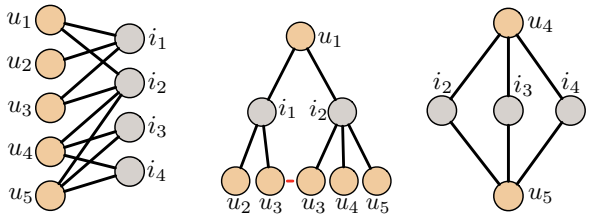
Collaborative Filtering, Disentangled Representation Learning, Non-Euclidean Geometry

**ACM Reference Format:**
Yiding Zhang, Chaozhuo Li, Xing Xie, Xiao Wang, Chuan Shi, Yuming Liu, Hao Sun, Liangjie Zhang, Weiwei Deng, and Qi Zhang. 2022. Geometric Disentangled Collaborative Filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531982

## 1 INTRODUCTION

The rapid development of information technology has facilitated an explosion of information, leading to the challenge of information overload [1]. Recommender systems mitigate the information overload by suggesting a small set of items for users to meet their personalized interests [26, 27, 47]. Basically, recommender systems aim at modeling a user's preferences based on her historical interactions (e.g., ratings and clicks) with different items, known as collaborative filtering (CF) [44], and further recommending the user some items that she might be interested in [1].

Most existing CF models follow the paradigm to first learn a set of user/item representations and then build an interaction function to make recommendations based on the learned embeddings [44]. Learning representations that precisely reflect users' preferences based chiefly on user historical behaviors, has been a central point of interest. Matrix Factorization (MF) [24] embeds users and




(a) User-item interactions. (b) Tree-likeness structures. (c) Cyclic structures.

**Figure 1: An illustration of the user-item interactions and the non-Euclidean structures.**

items as distributed vectors via matrix decomposition. Deep neural networks (DNNs) are further introduced to capture the latent preferences beneath the highly non-linear interactions [15, 47]. Recently, graph neural networks (GNNs) have demonstrated great potential in boosting recommendation performance. High-order user-item interactions are explicitly captured by the stacked GNN layers to learn expressive embeddings [14, 48].

The representation learning of most CF models is defined in the Euclidean space due to the computational simplicity [43]. However, recent researches have demonstrated that a myriad of data (e.g., data with tree-likeness or cyclic structures) exhibits the highly non-Euclidean latent anatomy [3, 25, 40, 50]. Compared to the conventional Euclidean geometry, non-Euclidean geometries (i.e., hyperbolic or spherical geometry) are more suitable for modeling data with non-Euclidean characteristics. Coincidentally, such non-Euclidean characteristics also inherently exist in the CF scenario. As shown in Figure 1(b), the high-order interactions of user $u_1$ can be naturally extended to a tree-likeness structure based on the interactions in Figure 1(a). This is reasonable as the receptive field tends to be exponentially larger in the higher orders [42]. Euclidean geometry is insufficient to obtain relatively low distortion for embedding such tree structures even using an unbounded number of dimensions [28], but this task would be surprisingly easy for hyperbolic spaces with only 2 dimensions [41]. In addition, users sharing similar preferences may interact with similar items, leading to the cyclic structures in Figure 1(c) (e.g., $u_4 \rightarrow i_2 \rightarrow u_5 \rightarrow i_3 \rightarrow u_4$). Such cyclic structures indicate the behavioral similarities between users, which are essentially the collaborative signals leveraged by CF methods [44]. Representing the cyclic structures into the Euclidean space with limited expressivity may result in inferior representations, while spherical geometry is more powerful in modeling data with cyclic structures [3, 9].

Recently, several works focus on learning user/item representations within a single type of non-Euclidean space (e.g., hyperbolic space) [36, 45, 46, 52]. However, the underlying structures of user-item interactions are more complicated than the pure trees or spheres. As shown in Figure 1(b), the tree-likeness structure and the cyclic structure (e.g., $u_1 \rightarrow i_1 \rightarrow u_3 \rightarrow i_2 \rightarrow u_1$) are nested together, leading to the hybrid geometric characteristics. From this point of view, the single geometry-based hypothesis is inadequate to capture latent intent factors beneath such hybrid structures. To reveal the latent reasons why a user interacted with an item, disentangled representation learning is introduced to learn factorized embeddings to disentangle the latent intent factors [31, 32, 49, 54]. Although existing disentangled CF models learn representations solely within the Euclidean geometry, they motivate us to disentangle the sophisticated characteristics with different geometries.

In this paper, we propose to perform CF over hybrid geometries via disentangled learning, namely Geometric Disentangled Collaborative Filtering (GDCF). Based on the basic assumption that user-item interactions are latently generated from highly sophisticated intent factors, we further presume that these intent factors should be associated with various geometries. Different intent factors are responsible for generating user-item interactions belonging to different types of geometries. However, how to learn a desirable GDCF model is still obscure. Previous disentangled CF models only focus on uncovering the intent factors, while GDCF further needs to capture the latent correlations between the intent factors and different geometries. In addition, the closeness measurements in different geometries are distinct. For example, distances in spherical spaces are finite while the distances can be infinite in Euclidean and hyperbolic spaces. Since the representations of users and items are defined within different geometries, it is essential to measure the closeness across different geometries in a principled manner.

To address the mentioned challenges, we make the first attempt to learn geometric disentangled representations and propose a novel generative GDCF model on the basis of variational autoencoders (VAEs). Specifically, the geometric disentangled representations are learned by identifying the high-level geometric concepts associated with user intentions. The concepts represent the intent factors associated with different geometries, which contribute to capturing the task-relevant correlations between latent factors and geometries. To measure the user-item similarities across multiple geometries, we further propose to project the disentangled representations from different spaces into a shared latent space, and thus a universal measurement can be leveraged to calculate their closeness. Our proposal is thoroughly evaluated on five publicly available datasets, and the experimental results demonstrate its superiority.

We summarize our main contributions as follows:

- To the best of our knowledge, we are the first to study the novel problem of geometric disentangled collaborative filtering, which is capable of capturing fine-grained geometric characteristics beneath the hybrid user-item interactions.
- We propose a novel GDCF model to learn geometric disentangled representations by identifying the high-level geometry-aware concepts associated with user intentions.
- Extensively, we evaluate GDCF on five real-world datasets. Experimental results demonstrate its superiority.

## 2 PROBLEM DEFINITION

In this section, we will formally define the studied problem. A recommendation dataset can be formulated as $\mathbf{D} = \{\mathbf{U}, \mathbf{I}, \mathbf{E}\}$, in which $\mathbf{U} = \{u_1, u_2, \cdots, u_M\}$ denotes a set of $M$ users, $\mathbf{I} = \{i_1, i_2, \cdots, i_N\}$ is the set of $N$ items and $\mathbf{E} \in \{0, 1\}^{M \times N}$ consists of the historical interactions between users and items. For convenience, we use $\mathbf{f}_u = \{i : \mathbf{E}_{u,i} = 1\}$ to represent items interacted with user $u$. Given a candidate pair $(u, i)$ consisting of a target user $u$ and a potential item $i$, we aim to learn a preference score $\mathbf{f}_{u,i} \in \{0, 1\}$ to indicate how likely this item should be recommended to the target user.

Different from existing single geometry based approaches, we aim to learn the representations for users $\{\mathbf{z}_u\}_{u=1}^N$ and items $\{\mathbf{h}_i\}_{i=1}^M$ by disentangling user embeddings with different geometries to comprehensively understand the sophisticated interactions.

## 3 PRELIMINARY

In this section, we will introduce some preliminary knowledge from the perspective of manifolds, including the basic notations and operators in three geometries (i.e., hyperbolic, Euclidean and spherical geometries). Three types of manifolds $\mathcal{M}$ are defined with the constant sectional curvature $\kappa$ [5]. The general realizations of these manifolds are the hyperboloid $\mathbb{H}_\kappa$, the Euclidean space $\mathbb{E}$, as well as the hypersphere $\mathbb{S}_\kappa$:

$$\mathcal{M}_\kappa^n = \begin{cases} \mathbb{H}_\kappa^n \colon \{\boldsymbol{x} \in \mathbb{R}^{n+1} \colon \langle \boldsymbol{x}, \boldsymbol{x} \rangle_\kappa = 1/\kappa, x_0 > 0\}, & \text{for } \kappa < 0 \\ \mathbb{E}_\kappa^n \colon \mathbb{R}^d, & \text{for } \kappa = 0 \\ \mathbb{S}_\kappa^n \colon \{\boldsymbol{x} \in \mathbb{R}^{n+1} \colon \langle \boldsymbol{x}, \boldsymbol{x} \rangle_\kappa = 1/\kappa\}, & \text{for } \kappa > 0 \end{cases} \tag{1}$$

where $\langle \cdot, \cdot \rangle_\kappa$ denotes the curvature-aware scalar product, namely $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^n x_i y_i$ for $\kappa > 0$ and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa = \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^n x_i y_i$ for $\kappa < 0$. For $\mathcal{M}_\kappa^n \in \{\mathbb{H}_\kappa^n, \mathbb{S}_\kappa^n\}$, a critical notion is **tangent space** [16], which is useful when the operators are not explicitly defined in the manifolds. The tangent space of $\mathcal{M}_\kappa^n$ at $\boldsymbol{x} \in \mathcal{M}_\kappa^n$ is defined as a $n$-dimensional vector space approximating $\mathcal{M}_\kappa^n$ around $\boldsymbol{x}$:

$$\mathcal{T}_{\boldsymbol{x}} \mathcal{M}_\kappa^n := \{\boldsymbol{v} \in \mathbb{R}^{n+1} : \langle \boldsymbol{v}, \boldsymbol{x} \rangle_\kappa = 0\}. \tag{2}$$

The mapping between manifold $\mathcal{M}_\kappa^n \in \{\mathbb{H}_\kappa^n, \mathbb{S}_\kappa^n\}$ and its tangent space $\mathcal{T}_{\boldsymbol{x}} \mathcal{M}_\kappa$ can be achieved by **exponential map** and **logarithmic map** [16]. The exponential map projects points from a subset of the tangent space $\mathcal{T}_{\boldsymbol{x}} \mathcal{M}_\kappa$ to the original manifold $\mathcal{M}_\kappa^n$, while the logarithmic map projects points in the opposite direction. For points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}_\kappa^n$, $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}} \mathcal{M}_\kappa^n$, such that $\boldsymbol{x} \neq \boldsymbol{y}$, $\boldsymbol{v} \neq \mathbf{0}$ and $\mathbf{0} = (1/\sqrt{|\kappa|}, 0, \cdots, 0)$, the exponential map $\exp_{\boldsymbol{x}}^\kappa(\cdot)$ and logarithmic map $\log_{\boldsymbol{x}}^\kappa(\cdot)$ are formally defined as follows:

$$\begin{aligned} \exp_{\boldsymbol{x}}^\kappa(\boldsymbol{v}) &= \cos_\kappa(\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa)\boldsymbol{x} + \sin_\kappa(\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa)\frac{\boldsymbol{v}}{\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa}, \\ \log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) &= \frac{\cos_\kappa^{-1}(\kappa\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa)}{\sin_\kappa\left(\cos_\kappa^{-1}(\kappa\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa)\right)}(\boldsymbol{y} - \kappa\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa \boldsymbol{x}), \end{aligned} \tag{3}$$

where $\|\boldsymbol{v}\|_\kappa = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_\kappa}$ denotes the norm of tangent vector, and the curvature-aware trigonometric functions are:

$$\sin_\kappa = \begin{cases} \sin & \text{if } \kappa > 0 \\ \sinh & \text{if } \kappa < 0 \end{cases}, \quad \cos_\kappa = \begin{cases} \cos & \text{if } \kappa > 0 \\ \cosh & \text{if } \kappa < 0 \end{cases}.$$

To connect vectors in tangent spaces, we use parallel transport $\mathrm{PT}_{\boldsymbol{x}-\boldsymbol{y}} : \mathcal{T}_{\boldsymbol{x}} \mathcal{M}_\kappa^n \to \mathcal{T}_{\boldsymbol{y}} \mathcal{M}_\kappa^n$ [16]:

$$\mathrm{PT}_{\boldsymbol{x} \to \boldsymbol{y}}^\kappa(\boldsymbol{y}) = \boldsymbol{v} - \frac{\kappa\langle \boldsymbol{y}, \boldsymbol{v} \rangle_\kappa}{1 + \kappa\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa}(\boldsymbol{x} + \boldsymbol{y}), \tag{4}$$

which ensures the transported vectors stay parallel to the connection. In addition, the geodesic distance is defined as:

$$d^\kappa(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\sqrt{|\kappa|}} \cos_\kappa^{-1}(\kappa\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa). \tag{5}$$

There are also several widely-used realizations of these spaces, i.e., Poincaré ball $\mathbb{P}_\kappa$ for hyperbolic spaces, projected hypersphere $\mathbb{D}_\kappa$ for spherical spaces. Please refer to [43] for more details. Moreover, for Euclidean space $\mathbb{E}^n$, let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{E}^n$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}} \mathbb{E}^n$, the exponential map is defined as $\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \boldsymbol{x} + \boldsymbol{v}$, and the logarithmic map is $\log_{\boldsymbol{x}}(\boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}$. The parallel transport is defined as $\mathrm{PT}_{\boldsymbol{x} \to \boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{v}$.

## 4 METHODOLOGY

In this section, we will introduce the details of the GDCF model. First, we will expound the basic model paradigm. Then, the implementations of several major components will be presented. Finally, we will discuss several key points within our proposal.

### 4.1 Mathematical Paradigm

In this subsection, we will introduce the basic mathematical paradigm of GDCF. Users may have diverse interests, and these interests can be attributed to various latent concepts. Therefore, we aim to learn a factorized representation for user $u$ composed of $K$ high-level concepts. The factorized representation of user $u$ is denoted as $\mathbf{z}_u = [\mathbf{z}_u^{(1)}, \mathbf{z}_u^{(2)}, \cdots, \mathbf{z}_u^{(K)}]$, where $\mathbf{z}_u^{(k)} \in \mathcal{M}_k$ is a $d$-dimensional concept representation living in the manifold $\mathcal{M}_k$. $\mathbf{z}_u^{(k)}$ depicts the preference of user $u$ regarding to the $k^{\text{th}}$ concept. Different from the diverse interests of users, an item usually represents a specific object, which is fixed and explicit [32]. Thus, items are represented by the $d$-dimensional embeddings $\{\mathbf{h}_i\}_{i=1}^M \in \mathbb{E}^d$ in Euclidean spaces instead of the factorized vectors.

In order to model the correlations between items and latent concepts, we design the concept assignment relations for items: $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^M$, where $\mathbf{c}_i = [c_{i,1}, c_{i,2}, \cdots, c_{i,k}]$ and $c_{i,k}$ indicates whether item $i$ belongs to concept $k$. The concept assignment of item $i$ is described by a distribution $p(c_{i,k})$, which denotes the probability of item $i$ belonging to concept $k$ and $\sum_{k=1}^K p(c_{i,k}) = 1$. The assignments among different concepts ensure that the user-item interactions can be reconstructed based on different manifolds.

Following previous works [17, 23, 31], we adopt the variational auto-encoders (VAEs) as the basic paradigms and propose a generative GDCF model. Assuming that the interactions are generated from the true world simulator using the factorized user embedding $\mathbf{z}_u$ along with the concept assignment distribution $p(\mathbf{C})$. An appropriate objective function should maximize the marginal likelihood of the user' observed data $\mathbf{f}_u$ in expectation over the distribution of user representation $\mathbf{z}_u$ and item concept assignment $\mathbf{C}$:

$$\max_\theta \ \mathbb{E}_{p_\theta(\mathbf{C})}\left[\int p_\theta(\mathbf{f}_u|\mathbf{z}_u, \mathbf{C}) p_\theta(\mathbf{z}_u) d\mathbf{z}_u\right]. \tag{6}$$

Given the historical interactions of user $u$: $\mathbf{f}_u = \{i : \mathbf{E}_{u,i} = 1\}$, the inferred posterior configurations of $\mathbf{z}_u$ and $\mathbf{C}$ can be naturally described by a probability distribution $q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C})$. Moreover, we are also interested in disentangling a user's preferences at a more granular level regarding various aspects of an item. This can be achieved by introducing a constraint over $q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C})$ and matching $q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C})$ to a unit Gaussian prior $p_\theta(\mathbf{z}_u)$. Therefore, the constrained optimization problem can be formalized as:

$$\begin{aligned} &\max_\theta \ \mathbb{E}_{\mathbf{f}_u \sim \mathcal{D}}\left[\mathbb{E}_{p_\theta(\mathbf{C})}\left[\mathbb{E}_{q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C})}\left[\log p_\theta(\mathbf{f}_u|\mathbf{z}_u, \mathbf{C})\right]\right]\right], \\ &\text{subject to } \ D_{KL}\big(q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C}) \| p_\theta(\mathbf{z}_u)\big) < \epsilon, \end{aligned} \tag{7}$$

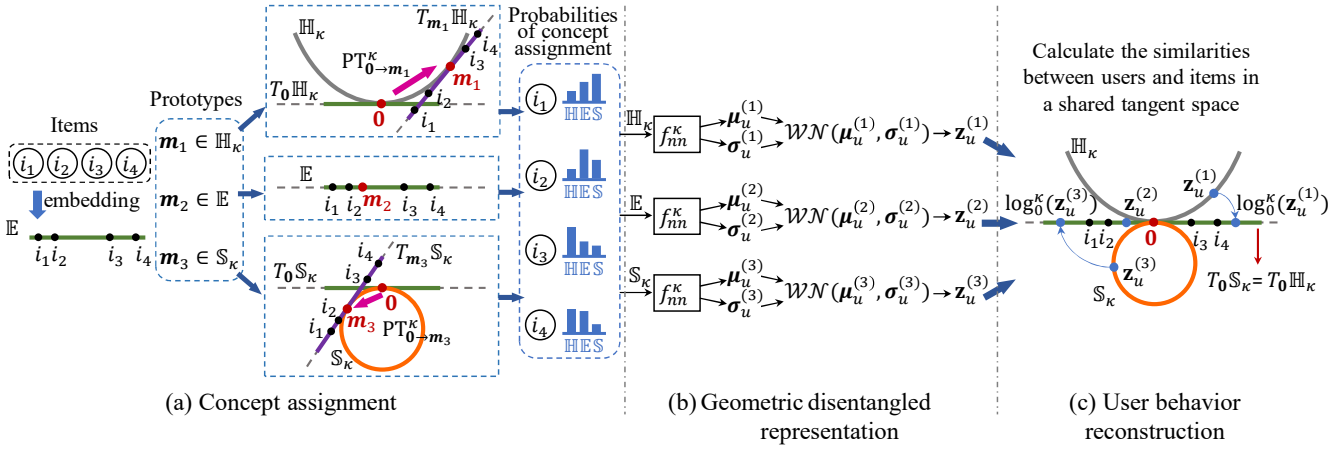



**Figure 2: The framework of GDCF.**

where $\epsilon \geq 0$ indicates the strength of the applied constraint. We rewrite Eq. (7) as a Lagrangian under the KKT conditions [19, 21]:

$$\begin{aligned}\mathcal{F}(\theta, \beta; \mathbf{f}_u, \mathbf{z}_u) = \mathbb{E}_{p_\theta(\mathbf{C})}\Big[\mathbb{E}_{q_\theta(\mathbf{z}_u|\mathbf{f}_u,\mathbf{C})}\big[\log p_\theta(\mathbf{f}_u|\mathbf{z}_u,\mathbf{C})\big] \\ - \beta\big(D_{KL}(q_\theta(\mathbf{z}_u|\mathbf{f}_u,\mathbf{C})\|p_\theta(\mathbf{z}_u)) - \epsilon\big)\Big],\end{aligned} \tag{8}$$

where the KKT multiplier $\beta \geq 0$ is the regularization coefficient. Since $\beta, \epsilon \geq 0$, according to the complementary slackness KKT condition, Eq. (8) can be re-written as [17]:

$$\begin{aligned}\mathcal{F}(\theta, \beta; \mathbf{f}_u, \mathbf{z}_u) \geq & \mathcal{L}(\theta; \mathbf{f}_u, \mathbf{z}_u, \beta) \\ = & \mathbb{E}_{p_\theta(\mathbf{C})}\Big[\mathbb{E}_{q_\theta(\mathbf{z}_u|\mathbf{f}_u,\mathbf{C})}\big[\log p_\theta(\mathbf{f}_u|\mathbf{z}_u,\mathbf{C})\big] \\ & - \beta D_{KL}(q_\theta(\mathbf{z}_u|\mathbf{f}_u,\mathbf{C})\|p_\theta(\mathbf{z}_u))\Big],\end{aligned} \tag{9}$$

which is the final optimization objective of our model.

## 4.2 Implementation

In this subsection, we will introduce the details of three major components in the implementation of the proposed GDCF model as shown in Figure 2.

*4.2.1 Concept assignment.* Concept assignment calculates the probabilities of an item assigning to different concepts. We propose a prototype-based concept assignment strategy to prevent over-parameterization and low sampling efficiency. Specifically, $K$ prototypes $\{\mathbf{m}_k\}_{k=1}^K$ are proposed to describe the anchors of the concepts across manifolds. Since items and the prototype $\mathbf{m}_k$ might come from different geometries, we need to map them into a shared space for comparison. One straightforward approach is to assume all the items are represented in the shared tangent space at $\mathbf{0}$, i.e., $\mathcal{T}_{\mathbf{0}}\mathcal{M}$, and then transform them into $\mathcal{M}_k$ via $\exp_{\mathbf{0}}^\kappa(\cdot)$. However, the transformed item representations would be identical and indistinguishable if the prototypes share the same latent space, i.e., $\mathcal{M}_k = \mathcal{M}_{k+1}$ [31]. Thus, the prototypes of items should be explicitly incorporated in the transformation process. As shown in Figure 2 (a), we assume that item representations $\{\mathbf{h}_i\}_{i=1}^M$ come from $\mathcal{T}_{\mathbf{0}}\mathcal{M}_k$, and transform them into $\mathcal{T}_{\mathbf{m}_k}\mathcal{M}_k$ via parallel transport to obtain the prototype-aware item representations:

$$\mathbf{h}'_i = \mathrm{PT}^\kappa_{\mathbf{0}\to\mathbf{m}_k}(\mathbf{h}_i). \tag{10}$$

Then, we define the similarity measurement between $\mathbf{h}'_i$ and $\mathbf{m}_k$. A common strategy is to transform $\mathbf{h}'_i$ into manifold $\mathcal{M}_k$ via $\exp^\kappa_{\mathbf{m}_k}(\cdot)$. Similarities between $\mathbf{h}'_i$ and $\mathbf{m}_k$ can be measured by their geodesic distance on $\mathcal{M}_k$. We theoretically prove this approach is equivalent to a simpler operation, i.e., computing the norm of $\mathbf{h}'_i$ on $\mathcal{T}_{\mathbf{m}_k}\mathcal{M}_k$:

Theorem 4.1. *For $\mathbf{x}, \mathbf{y} \in \mathcal{M}$ and $\mathcal{M} \in \{\mathbb{E}, \mathbb{S}_\kappa, \mathbb{D}_\kappa, \mathbb{H}_\kappa, \mathbb{P}_\kappa\}$, the geodesic distance between $\mathbf{x}$ and $\mathbf{y}$, is equal to the norm of $\log^\kappa_{\mathbf{x}}(\mathbf{y})$ in the tangent space $\mathcal{T}_{\mathbf{m}_k}\mathcal{M}_k$, i.e., $d(\mathbf{x}, \mathbf{y}) = \|\log^\kappa_{\mathbf{x}}(\mathbf{y})\|_\kappa$.*

Proof of Theorem 4.1: For Euclidean space $\mathbb{E}$, it is easy to prove that $\|\log_{\boldsymbol{x}}(\boldsymbol{y})\| = \|\boldsymbol{y} - \boldsymbol{x}\| = d_{\mathbb{E}}(\boldsymbol{x}, \boldsymbol{y})$. For the hyperboloid model, let $\boldsymbol{v} = \log^\kappa_{\boldsymbol{x}}(\boldsymbol{y})$, we have $\boldsymbol{y} = \exp^\kappa_{\boldsymbol{x}}(\boldsymbol{v})$. The distance between $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{H}_\kappa$ can be rewritten as:

$$\begin{aligned}& d_{\mathbb{H}_\kappa}(\boldsymbol{x}, \boldsymbol{y}) = d_{\mathbb{H}_\kappa}(\boldsymbol{x}, \exp^\kappa_{\boldsymbol{x}}(\boldsymbol{v})) \\ =& \frac{1}{\sqrt{-\kappa}}\cosh^{-1}\Big(\kappa\big(\langle \boldsymbol{x}, \cosh(\sqrt{-\kappa}\|\boldsymbol{v}\|_{\mathcal{L}})\boldsymbol{x} \\ & + \sinh(\sqrt{-\kappa}\|\boldsymbol{v}\|_{\mathcal{L}})\frac{\boldsymbol{v}}{\sqrt{-\kappa}\|\boldsymbol{v}\|_{\mathcal{L}}}\rangle_{\mathcal{L}}\big)\Big) \\ =& \frac{1}{\sqrt{-\kappa}}\cosh^{-1}\big(\kappa \cdot \frac{1}{\kappa}\cosh(\sqrt{-\kappa}\|\boldsymbol{v}\|_{\mathcal{L}})\big) \\ =& \|\boldsymbol{v}\|_{\mathcal{L}} = \|\log^\kappa_{\boldsymbol{x}}(\boldsymbol{y})\|_{\mathcal{L}}.\end{aligned} \tag{11}$$

This theorem also holds for the hypersphere model $\mathbb{S}_\kappa$ following above proposition. For the Poincaré ball model $\mathbb{P}_\kappa$, the distance between $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{P}_\kappa$ can be rewritten as:

$$\begin{aligned}& d_{\mathbb{P}_\kappa}(\boldsymbol{x}, \boldsymbol{y}) = d_{\mathbb{P}_\kappa}(\boldsymbol{x}, \exp^\kappa_{\boldsymbol{x}}(\boldsymbol{v})) \\ =& \frac{2}{\sqrt{-\kappa}}\tanh^{-1}\left(\sqrt{-\kappa}\tanh\left(\sqrt{-\kappa}\frac{\lambda^\kappa_{\boldsymbol{x}}\|\boldsymbol{v}\|}{2}\right)\frac{1}{\sqrt{-\kappa}}\right) \\ =& \lambda_{\boldsymbol{x}}\|\boldsymbol{v}\| = \|\log^\kappa_{\boldsymbol{x}}(\boldsymbol{y})\|^{\mathbb{P}_\kappa}_{\boldsymbol{x}}.\end{aligned} \tag{12}$$

The formulas about $\mathbb{P}_\kappa$ are referred from [43]. The proof of the projected hypersphere model $\mathbb{D}_\kappa$ is similar to the above process. □

Based on Theorem 4.1, we can measure the similarities by the norm of $\mathbf{h}_i'$ as:

$$s_{i,k} = -\|\mathbf{h}_i'\|_\kappa. \tag{13}$$

The probability of item $i$ belonging to concept $k$ can be achieved after a softmax function:

$$p(c_{i,k}) = \frac{\exp(s_{i,k}/\tau)}{\sum_{l=1}^{K}\exp(s_{i,l}/\tau)}, \tag{14}$$

where $\tau$ is a hyper-parameter to control the scale of values, and we set $\tau \in (0,1)$ to obtain a more skewed distribution.

*4.2.2 Geometric disentangled representation.* Users may have diverse interests, leading to the disentangled representations w.r.t. different concepts. The prior distribution $p_\theta(\mathbf{z}_u)$ is set to a unit Gaussian distribution. The encoder $q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C})$ learns the representations for user $u$ based on the high-level concept distributions. Assume $q_\theta(\mathbf{z}_u|\mathbf{f}_u, \mathbf{C}) = \prod_{k=1}^{K} q_\theta(\mathbf{z}_u^{(k)}|\mathbf{f}_u, \mathbf{C})$, and each $q_\theta(\mathbf{z}_u^{(k)}|\mathbf{f}_u, \mathbf{C})$ is defined as a multivariate normal distribution with a diagonal co-variance matrix [17]. Let $\mathbf{t}_u = \big[[f_{u,1}, f_{u,2}, \cdots, f_{u,M}] \odot [p(c_{1,k}), p(c_{2,k}), \cdots, p(c_{M,k})]\big] \in \mathbb{E}^M$ be the probabilities of the interacted items for user $u$ belonging to $k^{\text{th}}$ concept, where $\odot$ is the point-wise product. The mean vector $\boldsymbol{\mu}_u^{(k)} \in \mathcal{M}_k^d$ and the standard deviation $\boldsymbol{\sigma}_u^{(k)} \in \mathcal{T}_{\mathbf{0}}\mathcal{M}_k^d$ are calculated by a neural network $f_{\text{nn}}^\kappa : \mathcal{M}_k^M \to \mathcal{M}_k^{2d}$ with input $\mathbf{t}_u$:

$$\begin{aligned} (\mathbf{a}_u^{(k)}, \mathbf{b}_u^{(k)}) &= f_{\text{nn}}^\kappa\big(\exp_{\mathbf{0}}^\kappa(\mathbf{t}_u)\big) \\ \boldsymbol{\mu}_u^{(k)} &= \mathbf{a}_u^{(k)}, \boldsymbol{\sigma}_u^{(k)} = \sigma_0 \cdot e^{(-\frac{1}{2}\log_{\mathbf{0}}^\kappa(\mathbf{b}_u^{(k)}))}. \end{aligned} \tag{15}$$

Here we implement the $f_{\text{nn}}^\kappa$ as non-Euclidean neural networks [3, 10, 53]. Specifically, for the projected manifolds, i.e., $\mathbb{D}_\kappa$ and $\mathbb{P}_\kappa$, we follow the neural structures proposed in [3, 10], which leverage the $\log_{\mathbf{0}}^\kappa$ and $\exp_{\mathbf{0}}^\kappa$ to transform the $M$-dimensional representations between tangent spaces and the manifolds. The Euclidean operations are conducted over all the $M$ elements for the $M$-dimensional features. However, a $M$-dimensional feature vector in $\mathbb{H}_\kappa$ or $\mathbb{S}_\kappa$ has $M+1$ elements. If the Euclidean operations $f_{\text{nn}}$ are directly applied on all the $M+1$ dimensions, the transformed features would be out of the tangent spaces [53]. To address this challenge, we propose to conduct the Euclidean operations on the last $M$ elements. The non-Euclidean operations defined in $\mathbb{P}_\kappa$ or $\mathbb{D}_\kappa$ are equivalent to those defined in $\mathbb{H}_\kappa$ or $\mathbb{S}_\kappa$, respectively. Please refer to Appendix A.1 for the proposed operations and proofs.

As shown in Figure 2 (b), based on the obtained $\boldsymbol{\mu}_u^{(k)}$ and $\boldsymbol{\sigma}_u^{(k)}$, we leverage the wrapped normal distribution [37, 43] to build the Gaussian distribution on Riemannian manifolds to generate user representations. Specifically, to sample instances from the distribution $\mathcal{WN}(\boldsymbol{\mu}_u^{(k)}, \boldsymbol{\sigma}_u^{(k)})$, we need to first sample instances from the tangent space of the manifold $\mathcal{T}_{\mathbf{0}}\mathcal{M}_k$, i.e., $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_u) \in \mathcal{T}_{\mathbf{0}}\mathcal{M}_k$. The geometric disentangled representation $\mathbf{z}_u^{(k)}$ can be obtained via parallel transporting $\mathbf{v}_u$ from $\mathcal{T}_{\mathbf{0}}\mathcal{M}_k$ to $\mathcal{T}_{\boldsymbol{\mu}_u}\mathcal{M}_k$, and then projecting it to the manifold $\mathcal{T}_{\boldsymbol{\mu}_u}\mathcal{M}_k$ by exponential map:

$$\mathbf{z}_u^{(k)} = \exp_{\boldsymbol{\mu}_u^{(k)}}^\kappa\big(\text{PT}_{\mathbf{0}\to\boldsymbol{\mu}_u^{(k)}}(\mathbf{v}_u)\big). \tag{16}$$

*4.2.3 User behavior reconstruction.* Here we aim to predict which items are most likely to be clicked by the target user. Since the item representations live in the Euclidean space, we can view this space as a tangent space and transform the item representations $\{\mathbf{h}_i\}_{i=1}^M$ to $\mathcal{M}_k$, and then compute their similarities based on the distances. However, the distances in different spaces may have different value scales. For example, the distances in Euclidean or hyperbolic spaces could be theoretically infinite, while for spherical spaces, the distances are finite [16]. To tackle this challenge, we propose to project the user representations $\mathbf{z}_u^{(k)}$ to the tangent space $\mathcal{T}_{\mathbf{0}}\mathcal{M}_\kappa$. As shown in Figure 2 (c), the tangent spaces of these manifolds are isomorphic to Euclidean spaces. Therefore, we can leverage Euclidean measurements to calculate the similarities of different geometries. Here we only consider the angle between the representations to avoid the influence of the norm of the representations in different spaces. Therefore, the Euclidean inner product can be employed to measure the similarities between the normalized $k^{\text{th}}$ concept of user $u$ and item $i$ in the shared tangent space $\mathcal{T}_{\mathbf{0}}\mathcal{M}_\kappa$:

$$s(\mathbf{z}_u^{(k)}, \mathbf{h}_i) = \langle \frac{log_{\mathbf{0}}^\kappa(\mathbf{z}_u^{(k)})}{\|\log_{\mathbf{0}}^\kappa(\mathbf{z}_u^{(k)})\|}, \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|}\rangle. \tag{17}$$

User behaviors can be further reconstructed via the categorical distribution over the $M$ items: $p_\theta(f_{u,i}|\mathbf{z}_u, \mathbf{C}) \propto \sum_{k=1}^{K} s(\mathbf{z}_u^{(k)}, \mathbf{h}_i)$.

## 4.3 Optimization

The trainable parameters $\theta$ of GDCF include $K$ concept prototypes $\{\mathbf{m}_k \in \mathcal{M}_k^d\}_{k=1}^K$, $M$ item representations $\{\mathbf{h}_i \in \mathbb{E}^d\}_{k=1}^K$, and the parameters in the neural network $f_{\text{nn}}^\kappa$. We optimize $\theta$ to maximize the training objective in Eq. (9) via Adam [22] and RiemannianAdam [4] for Euclidean and non-Euclidean parameters, respectively. Moreover, since a $d$-dimensional embedding in manifold $\mathcal{M}_\kappa \in \{\mathbb{H}_\kappa, \mathbb{S}_\kappa\}$ has $d+1$ elements, we concatenate or remove an element "0" at the first dimension of features in the tangent spaces to ensure the vector operations can be conducted in a principal manner.

## 4.4 Discussions

In this section, we will discuss two key points, i.e., similarity measurements and probability distributions on Riemannian manifolds, as well as the relations to some related methods including disentangled VAEs and mix-curvature representation learning.

**Similarity measurement.** We leverage two types of metrics to measure the similarities: the geodesic distance for concept assignment in Eq. (13), and the Euclidean distance on the tangent spaces for user behavior reconstruction in Eq. (17). These two metrics are employed for different purposes. Concept assignment aims to infer the most likely manifold for item $i$ while the geodesic distance thrives on measuring the manifold-specific similarity. Thus, the metric used in the concept assignment is set to geodesic distance. For the behavior reconstruction, we need to measure the similarities between item representations and disentangled user representations on all the manifolds in a universal manner. The disentangled user representations are projected to a shared tangent space, where the Euclidean distance can be utilized as the closeness metric.

**Probability distribution on Riemannian manifolds.** Several approaches have been proposed to generalize the Normal distribution

**Table 1: Statistics of five datasets.**

| Dataset | MovieLens-100k | MovieLens-1M | MovieLens-20M | AliShop | LastFM |
|---|---|---|---|---|---|
| #users | 603 | 6,038 | 136,677 | 10,668 | 1,862 |
| #items | 5,697 | 3,605 | 20,108 | 20,591 | 14,795 |
| #interactions | 47,922 | 836,452 | 9,990,030 | 767,493 | 89,805 |
| #held-out users | 50 | 500 | 10,000 | 4,000 | 200 |

to Riemannian manifolds, including restricted normal, Riemannian normal, and wrapped normal [9, 13, 37]. Restricted normal distributions restrict a point of the ambient space sampled from a Gaussian distribution to the manifolds. Von Mises-Fisher distribution [9] is a kind of restricted normal, but it has only a single scalar covariance parameter and cannot parameterize covariance for different dimensions. Riemannian normal distributions [13, 34, 35] are based on geodesic distance in the manifolds. These distributions aim to maximize the entropy distributions and resemble the Gaussian distribution's properties the closest. However, the resemble process is quite computationally expensive, leading to low efficiency. Wrapped normal distributions [37, 43] first sample features from a Gaussian distribution in the tangent space and the sampled features are further transformed to Riemannian manifolds via parallel transport and exponential map. We select wrapped normal distributions in our model due to its computational efficiency. Note that we do not follow the wrapped normal distributions when calculating the probability density function. This is because the density functions are expected to be directly computed on the tangent space instead of the manifolds.

**Connections to disentangled VAEs.** VAEs have been widely exploited to learn the disentangled representations [17, 18, 20, 31]. Existing disentangled VAEs usually leverage the cosine function to measure the similarities, which ignore the norms of representations. Thus, these disentangled VAEs can be viewed as representation learning on the hypersphere [31]. The key difference between these methods and our proposal is that GDCF learns the representations on different Riemannian manifolds. Multiple Riemannian manifolds endow our proposal with powerful modeling capacity. The major difference between MacridVAE and GDCF is that MacridVAE only focuses on a single manifold (i.e., hypersphere) while GDCF is capable of disentangling representations for different types of manifolds. In addition, MacridVAE aims at learning the one-hot concept assignment relations $\mathbf{C}$, while GDCF does not require the $\mathbf{C}$ to be one-hot vectors, leading to a general solution with less restrictions.

**Connections to mix-curvature representation learning.** Mix-curvature methods [3, 11, 43] learn the representations by balancing the expressive capacities of different kinds of manifolds. Cross-manifold representations learned by existing mix-curvature methods are defined in the product spaces, i.e., $\mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_K$. GDCF is fundamentally different from these methods because the representations of GDCF on different manifolds are independent from each other, i.e., $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K$, which can preserve the uniqueness of different geometries.

**Complexity analysis.** The time complexity of GDCF is $O(kdNM)$, where $k$ is the number of concepts, $d$ is the dimension of user/item representations, $N$ and $M$ are the number of users and items, respectively. The time complexity is on par with the SOTA disentangled models like MacridVAE [31].

# 5 EXPERIMENTS

## 5.1 Experimental Setup

**Datasets**. We conduct the experiments on five real-world datasets, including three MovieLens datasets with different numbers of interactions (MovieLens-100k, MovieLens-1M and MovieLens-20M) [12], AliShop [31] and LastFM [6]. The training/validation/testing partitions are following previous works [27, 31]. The statistics of these datasets are shown in Table 1.

**Baselines**. We select the following SOTA CF models as baselines:

- NGCF [48] is a graph-based CF model to incorporate the high-order connectivity of user-item interactions.
- LightGCN [14] is a SOTA CF recommendation model based on graph convolution network.
- DGCF [49] is a disentangled CF model to learn representations for different latent user intentions via graph convolutional network.
- MacridVAE [31] is also a disentangled model, which leverages VAE to disentangle macro and micro components for user behaviors.
- HyperML [46] is a hyperbolic CF model, which explores metric learning in hyperbolic spaces for recommendation.
- H-VAE [36] is a hyperbolic VAE based CF approach, which employs a hyperbolic VAE to solve the collaborative filtering problem.
- HGCF [45] is a hyperbolic CF method, which is devised to capture the high-order correlations between users and items by using hyperbolic GNN.
- $\kappa$-GCN [3] is a mix-curvature GNN. We modify its loss to the hinge loss to solve the CF problem.

**Hyper-parameter**. We perform the hyper-parameter search on a small validation set. For the proposed GDCF, we set the dimension of embedding as 100 and tune the temperature $\tau \in [0, 0.1]$, $\beta \in [0, 100]$ in Eq. (9), the number of factors $K \in \{1, 2, 3, \ldots, 20\}$, the standard deviation of the prior $\sigma_0 \in [0.075, 0.5]$ and the number of hidden layers $l \in \{0, 1, 2, 3\}$ by grid search. We set $\kappa \in \{-1, 0, 1\}$ for hyperbolic, Euclidean and spherical geometry, respectively. Moreover, we also tune the following parameters for all methods: learning rate $\in [10^{-6}, 10^{-2}]$, dropout probability $\in [0, 1]$, $L_2$ regularization strength $\in [10^{-8}, 10^{-2}]$. We tune these hyper-parameters automatically via Optuna [2].

**Infrastructure**. We implement our model with Pytorch, and conduct the experiments with: CPU: Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz GPU: 24 × NVIDIA Tesla V100.

## 5.2 Main Results

We evaluate the performance of our approach on the five popular datasets. Here we implement GDCF with three kinds of manifolds i.e., $\mathbb{H}_\kappa$, $\mathbb{E}$, and $\mathbb{S}_\kappa$. To evaluate the effectiveness on top-K recommendation task, we adopt two popular evaluation metrics: Recall@20 and NDCG@20 [14, 48, 49]. We repeat this process 5 times and report the average scores.

**Table 2: The recommendation performance comparison. Best results are in bold and second best results are underlined.**

| Datasets | MovieLens-100k | | MovieLens-1M | | MovieLens-20M | | Alishop | | LastFM | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 |
| NGCF | 0.2437 | 0.2997 | 0.2956 | 0.2852 | 0.2979 | 0.3214 | 0.1058 | 0.1175 | 0.2681 | 0.2924 |
| LightGCN | 0.2596 | 0.3112 | 0.3115 | 0.3016 | 0.3172 | 0.3498 | 0.1573 | 0.1512 | 0.3116 | 0.3258 |
| HyperML | 0.2160 | 0.2549 | 0.2719 | 0.2839 | 0.2631 | 0.2809 | 0.1023 | 0.1061 | 0.2546 | 0.2852 |
| H-VAE | 0.2157 | 0.2706 | 0.2729 | 0.3049 | 0.2405 | 0.2737 | 0.1087 | 0.1101 | 0.2667 | 0.2795 |
| HGCF | 0.2218 | 0.2701 | 0.2844 | 0.2955 | 0.3093 | 0.3536 | 0.1115 | 0.1135 | 0.2706 | 0.2985 |
| $\kappa$-GCN | 0.1912 | 0.2408 | 0.2443 | 0.2351 | 0.2155 | 0.2419 | 0.0846 | 0.0816 | 0.2489 | 0.2582 |
| DGCF | 0.2572 | 0.3148 | 0.3252 | 0.3128 | 0.3193 | 0.3626 | 0.1556 | 0.1547 | 0.3062 | 0.3242 |
| MacridVAE | 0.2310 | 0.3095 | 0.3241 | 0.3605 | 0.3205 | 0.3965 | 0.1831 | 0.1862 | 0.3072 | 0.3213 |
| GDCF | **0.2652** | **0.3261** | **0.3314** | **0.3637** | **0.3317** | **0.4052** | **0.1874** | **0.1886** | **0.3173** | **0.3296** |

Experimental results are summarized in Table 2, and we can observe that the proposed GDCF achieves SOTA performance on the five datasets across all metrics, demonstrating the effectiveness of the geometric disentangled representation learning. GDCF outperforms the hyperbolic CF methods (e.g., HyperML, H-VAE and HGCF) by a large margin, which verifies that a single hyperbolic space may be insufficient to capture the hybrid geometric characteristics. The disentanglement-based methods (DGCF and MacridVAE) outperform other baselines in most cases, which proves the benefits of factorized representations under the CF scenario. By incorporating the geometric concepts into the disentangled representation learning, GDCF surpasses DGCF and MacridVAE over all the datasets, confirming the superior modeling capacity of our proposal.

## 5.3 Ablation Study

Since three types of geometries are incorporated in the GDCF model, we further conduct an ablation study to investigate the effectiveness of different geometry combinations. Here we leverage hyperboloid model $\mathbb{H}_\kappa$ and Poincaré ball $\mathbb{P}_\kappa$ to describe hyperbolic geometry, hypersphere model $\mathbb{S}_\kappa$ and projected hypersphere $\mathbb{D}_\kappa$ to describe spherical geometry, respectively. We evaluate the performance of GDCF with three types of geometry combinations, and the subscripts are omitted for simplicity: the single geometry $\mathbb{H}, \mathbb{P}, \mathbb{E}, \mathbb{S}, \mathbb{D}$; the combinations of two geometries $\mathbb{H}\&\mathbb{E}, \mathbb{P}\&\mathbb{E}, \mathbb{E}\&\mathbb{S}, \mathbb{E}\&\mathbb{D}, \mathbb{H}\&\mathbb{S}, \mathbb{P}\&\mathbb{D}$; and the combinations of three geometries $\mathbb{H}\&\mathbb{E}\&\mathbb{S}$, $\mathbb{P}\&\mathbb{E}\&\mathbb{D}$. The numbers of components for different geometries are set to approximately equal.

The results are reported in Table 3. The best results of all the methods are marked in bold, while the best results within each type of geometry combinations are marked with underlines. One can see that GDCF implemented with three kinds of geometries achieve the best performance in most cases, which demonstrates that all the three geometries contribute to learning expressive and meaningful representations. For the single geometry-based variations, non-Euclidean methods ($\mathbb{H}$, $\mathbb{P}$, $\mathbb{S}$ and $\mathbb{D}$) generally outperform Euclidean-based model ($\mathbb{E}$). It demonstrates that the non-Euclidean geometric structures are ubiquitous under the recommendation scenario and our proposal can effectively encode such patterns into representations. Moreover, the simplified version of GDCF with only two geometries ($\mathbb{H}\&\mathbb{E}$ and $\mathbb{H}\&\mathbb{S}$) already achieve promising performance on MovieLens datasets, revealing the effectiveness of representation learning in multiple geometries. Ablation models implemented with $\mathbb{H}$ or $\mathbb{S}$ consistently perform better than the counterparts with $\mathbb{P}$ and $\mathbb{D}$, which is reasonable as $\mathbb{P}$ and $\mathbb{D}$ suffer from numerical instability in the optimization process [7, 29, 39].

## 5.4 Universal Curvature Analysis

Previous ablation studies raise an interesting question: how to find the most effective combination of these geometries. The simplest approach is to enumerate all the possible combinations and select the candidate with best performance. This enumeration method would be optimal but impractical due to the inefficiency and low scalability [43]. Here we propose an approximate method to tackle this challenge. Specifically, we randomly initialize the curvatures of the components within $[-1, 1]$ and view the curvatures $\kappa$ of all components as trainable parameters in the training process. Moreover, the sign of curvature is also not constrained. Thus, components are able to change their geometries from hyperbolic (spherical) to spherical (hyperbolic) space to achieve the optimal solutions. Note that $\|\mathbf{0}\|_\kappa = \|(1/\sqrt{|\kappa|}, 0, \cdots, 0)^T\|_\kappa \overset{\kappa\to 0}{\longrightarrow} \pm\infty$, for the hyperboloid $\mathbb{H}_\kappa$ and hypersphere $\mathbb{S}_\kappa$ [43]. Therefore, the universal curvature variation of GDCF leverages the Poincaré ball $\mathbb{P}$ and the projected hypersphere $\mathbb{D}$ for hyperbolic and spherical geometry, respectively. GDCF with the universal curvature is denoted as $\mathbb{U}$. We are interested in comparing $\mathbb{U}$ with $\mathbb{P}\&\mathbb{D}$, since both of them leverage Poincaré ball $\mathbb{P}$ and projected hypersphere $\mathbb{D}$.

Table 4 exhibits the results. The combinations with the best performance are marked in bold. One can see that the variations with universal curvature $\mathbb{U}$ outperform $\mathbb{P}\&\mathbb{D}$ in most cases, which indicates $\mathbb{U}$ can automatically select more proper geometries. However, $\mathbb{U}$ does not converge to some specific curvatures [43], leading to the low generalization. Another limitation of $\mathbb{U}$ is that the Euclidean component cannot be incorporated as it cannot ensure $\kappa$ will converge to 0 in the training process.

## 5.5 Parameter Sensitivity Analysis

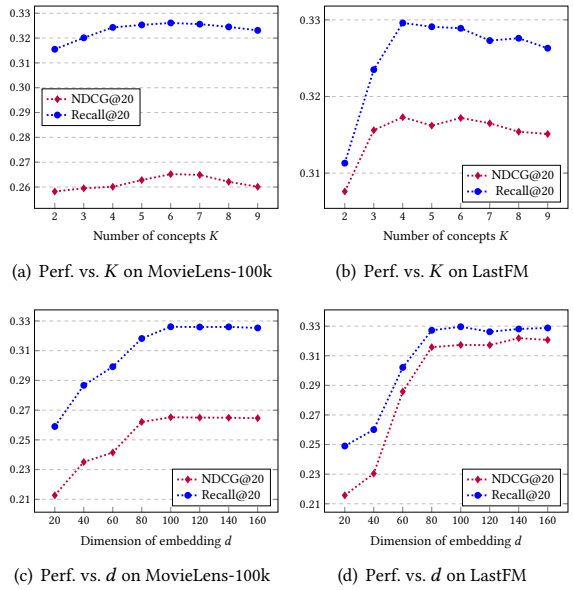
In this subsection, we will investigate the effect of two key parameters (the number of concepts $K$ and the dimension of embedding $d$) on the MovieLens-100k and LastFM datasets. Figure 3(a) and 3(b) show the results of GDCF under varying $K$. With the increase of the number of concepts $K$, the performance of GDCF first improves

**Table 3: Ablation study of the geometry combinations. Best results of all the methods are in bold, and the best results within each type of geometry combinations are underlined.**

| Datasets | MovieLens-100k | | MovieLens-1M | | MovieLens-100M | | Alishop | | LastFM | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 |
| $\mathbb{H}$ | 0.2340 | 0.3127 | 0.3236 | 0.3539 | 0.3223 | 0.3858 | 0.1870 | 0.1833 | 0.3106 | 0.3275 |
| $\mathbb{P}$ | 0.2353 | 0.3063 | 0.3245 | 0.3517 | 0.3216 | 0.3827 | 0.1862 | 0.1824 | 0.309 | 0.3225 |
| $\mathbb{E}$ | 0.2252 | 0.2965 | 0.3253 | 0.3548 | 0.3207 | 0.3851 | 0.1805 | 0.1797 | 0.3023 | 0.3172 |
| $\mathbb{S}$ | 0.2361 | 0.3141 | 0.3266 | 0.3554 | 0.3239 | 0.3892 | 0.1728 | 0.1733 | 0.3076 | 0.3212 |
| $\mathbb{D}$ | 0.2357 | 0.3046 | 0.3196 | 0.3501 | 0.3216 | 0.3864 | 0.1734 | 0.1751 | 0.3018 | 0.3176 |
| $\mathbb{H}$&$\mathbb{E}$ | 0.2620 | 0.3135 | 0.3263 | 0.3597 | 0.3263 | 0.3991 | **0.1885** | 0.1833 | 0.3147 | 0.3288 |
| $\mathbb{P}$&$\mathbb{E}$ | 0.2542 | 0.3118 | 0.3257 | 0.3540 | 0.3248 | 0.3954 | 0.1875 | 0.1826 | 0.3105 | 0.3238 |
| $\mathbb{E}$&$\mathbb{S}$ | 0.2517 | 0.3126 | 0.3261 | 0.3595 | 0.3250 | 0.3973 | 0.1798 | 0.1770 | 0.3149 | 0.3281 |
| $\mathbb{E}$&$\mathbb{D}$ | 0.2523 | 0.3231 | 0.3254 | 0.3538 | 0.3244 | 0.3922 | 0.1757 | 0.1785 | 0.3067 | 0.3192 |
| $\mathbb{H}$&$\mathbb{S}$ | 0.2582 | 0.3155 | 0.3271 | 0.3599 | 0.3295 | 0.4006 | 0.1841 | 0.1804 | 0.3141 | 0.3239 |
| $\mathbb{P}$&$\mathbb{D}$ | 0.2526 | 0.3164 | 0.3241 | 0.3562 | 0.3266 | 0.3934 | 0.1818 | 0.1794 | 0.3099 | 0.3227 |
| $\mathbb{P}$&$\mathbb{E}$&$\mathbb{D}$ | 0.2549 | 0.3196 | 0.3256 | 0.3572 | 0.3284 | 0.3986 | 0.1811 | 0.1802 | 0.3124 | 0.3276 |
| $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ | **0.2652** | **0.3261** | **0.3314** | **0.3637** | **0.3317** | **0.4052** | 0.1874 | **0.1886** | **0.3173** | **0.3296** |

**Table 4: Results of the universal curvature analysis.**

| Dataset | Metric | $\mathbb{P}$&$\mathbb{D}$ | $\mathbb{U}$ | Best |
|---|---|---|---|---|
| MovieLens-100k | NDCG@20 | 0.2526 | **0.2564** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| | Recall@20 | 0.3164 | **0.3218** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| MovieLens-1M | NDCG@20 | 0.3241 | **0.3261** | $\mathbb{H}$&$\mathbb{R}$&$\mathbb{S}$ |
| | Recall@20 | **0.3562** | 0.3557 | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| MovieLens-20M | NDCG@20 | 0.3266 | **0.3301** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| | Recall@20 | 0.3934 | **0.4049** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| AliShop | NDCG@20 | 0.1818 | **0.1821** | $\mathbb{H}$&$\mathbb{E}$ |
| | Recall@20 | 0.1794 | **0.1814** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| LastFM | NDCG@20 | 0.3099 | **0.3134** | $\mathbb{H}$&$\mathbb{E}$&$\mathbb{S}$ |
| | Recall@20 | 0.3227 | **0.3305** | $\mathbb{U}$ |




(a) Perf. vs. $K$ on MovieLens-100k (b) Perf. vs. $K$ on LastFM

(c) Perf. vs. $d$ on MovieLens-100k (d) Perf. vs. $d$ on LastFM

**Figure 3: Results of the parameter sensitivity analysis.**

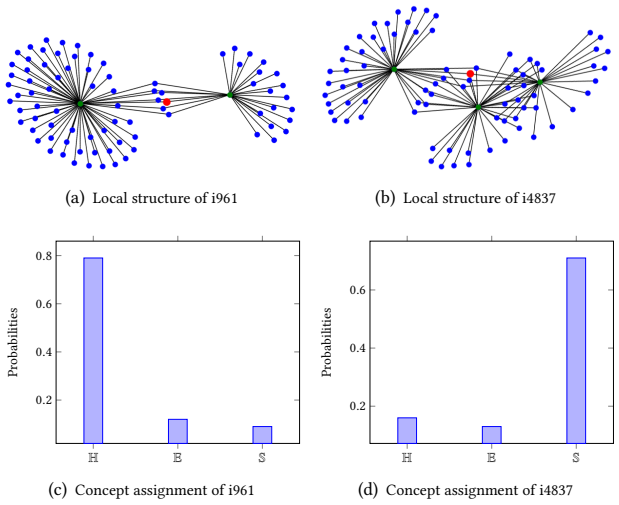
and then drops on both datasets across all metrics. This phenomenon suggests that a suitable number of concepts is needed for GDCF, while redundant concepts may introduce noise and lead to the performance decay. Also, figure 3(c) and 3(d) present the results of GDCF under varying $d$. With the increase of the dimension of embeddings $d$, model performance first improves and then keeps steady. This is reasonable as representations of proper dimensions are already capable of fully modeling the interactions, and too large dimensions may increase model complexity and effect the training efficiency.

## 5.6 Case Study

In this subsection, we will investigate whether GDCF can capture the correlations between the latent concepts and the geometric structures. To this end, we provide two cases (items with ID i961 and i4837) in MovieLens-100k dataset. Figure 4 displays the 2-hop ego-graphs centered with different items. One can see that the ego-graph of i961 is closer to the tree-likeness structures with less cycles compared to i4837. After model training, the concept assignment probabilities calculated by GDCF on different geometries are shown in Figure 4(c) and Figure 4(d). These probabilities reveal that the hyperbolic model $\mathbb{H}$ is more important in modeling tree-structures (i961) and spherical model $\mathbb{S}$ is crucial in modeling cyclical structures (i4837). This phenomenon demonstrates that our proposal is capable of effectively capturing the intrinsic correlations between the latent concepts and the geometric structures.

# 6 RELATED WORK

In this section, we will briefly summarize two related techniques: the disentangled representation learning and the non-Euclidean representation learning, along with their applications on the recommendation scenario.




(a) Local structure of i961 (b) Local structure of i4837

(c) Concept assignment of i961 (d) Concept assignment of i4837

**Figure 4: Local topological structures and concept assignments of two cases in MovieLens-100k. The center nodes, 1-hop nodes, 2-hop nodes are marked by red, green, blue circles, respectively.**

### 6.1 Disentangled Representation Learning

Disentangled representation learning aims at learning factorized representations that reveal and disentangle the latent intent factors beneath the input dataset, which has been widely used in a myriad of applications. Higgins et al. [17] propose beta-VAE to learn interpretable factorised latent representations from raw image data in an unsupervised manner. Ma et al. [30] introduce the disentangled graph convolutional network to learn disentangled node representations. Yang et al. [51] propose a factorizable graph convolutional network for explicitly disentangling the intertwined relations encoded in a graph. Recently, disentangled representation learning is introduced into the recommendation scenario to boost performance. MacridVAE [31] is designed to infer the high-level concepts associated with user intentions. Wang et al. [49] further devise a disentangled graph collaborative filtering model to encode the high-order interactions into the factorized representations. For the sequential recommendation, Ma et al. [33] introduce a sequence-to-sequence training strategy based on latent self-supervision and disentanglement. By combining the curriculum learning with the disentangled representation learning, Curriculum Disentangled Recommendation model [8] is proposed to efficiently learn disentangled representations from complex and noisy multi-feedbacks. Despite the promising performance, existing disentangled approaches learn representations solely within the Euclidean geometry and ignore the hybrid geometric characteristics of the user-item interactions.

### 6.2 Non-Euclidean Representation Learning

A myriad of data exhibits the highly non-Euclidean latent anatomy [3, 25, 40, 50], and non-Euclidean geometries (i.e., hyperbolic or spherical geometry) are more suitable for modeling data with non-Euclidean characteristics compared to Euclidean geometry. Nickel et al. [38] first leverage the hyperbolic space to learn hierarchical representations of symbolic data. HGCN [7] is an inductive hyperbolic GCN designed for hierarchical and scale-free graphs. Mixed-curvature Variational Autoencoder [43] exploits both the hyperbolic and spherical geometries to better modeling data characteristics. In the recommendation scenario, existing works usually focus on modeling interactions with hyperbolic models [46]. Feng et al. [46] propose to learn the representations of check-in activities in a hyperbolic space. Mirvakhabova et al. [36] propose an autoencoder based on hyperbolic geometry for solving collaborative filtering problem. HGCF [45] is devised to capture the high-order correlations between users and items by integrating hyperbolic geometry into graph neural networks. Most exising works focus on learning representations in a single non-Euclidean space, which may be incapable of modeling the hybrid geometric interactions as discussed in the introduction.

## 7 CONCLUSION

In this paper, we study the novel problem of geometric disentangled collaborative filtering. Different from existing single geometry-based CF models, we propose a novel GDCF model to incorporate multiple types of geometries to disentangle the sophisticated hybrid patterns of user-item interactions. Specifically, GDCF learns factorized representations that uncover and disentangle the intent factors hidden in the historical interactions, which is capable of learning expressive user/item representations. Extensive experiments are conducted over five publicly available datasets, and the experimental results demonstrate the superiority of GDCF.

## ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62172052, 62002029, 61772082).

## REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.
[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *SIGKDD*.
[3] Gregor Bachmann, Gary Bécigneul, and Octavian-Eugen Ganea. 2020. Constant Curvature Graph Convolutional Networks. In *ICML*.
[4] Gary Becigneul and Octavian-Eugen Ganea. 2019. Riemannian Adaptive Optimization Methods. In *ICLR*.
[5] Marcel Berger. 2012. *A panoramic view of Riemannian geometry*. Springer Science & Business Media.
[6] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *RecSys*. ACM.
[7] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NeurIPS*. 4869–4880.
[8] Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum Disentangled Recommendation with Noisy Multi-feedback. *NeurIPS* 34 (2021).
[9] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891* (2018).
[10] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *NeurIPS*. 5350–5360.
[11] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2018. Learning mixed-curvature representations in product spaces. In *ICLR*.

[12] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[13] Søren Hauberg. 2018. Directional statistics with the spherical normal distribution. In *FUSION*. 704–711.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[16] Sigurdur Helgason. 1979. *Differential geometry, Lie groups, and symmetric spaces*. Vol. 80. Academic press.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*. 1024–1034.

[18] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2018. Learning to decompose and disentangle representations for video prediction. In *NeurIPS*. 515–524.

[19] Kuhn HW and T AW. 1951. Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium, Berkeley: University of California Press*. 481–492.

[20] Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled Representation Learning for Non-Parallel Text Style Transfer. In *ACL*. 424–434.

[21] William Karush. 1939. *Minima of Functions of Several Variables with Inequalities as Side Conditions*. Master's thesis. Department of Mathematics, University of Chicago, Chicago, IL, USA.

[22] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR* (2015).

[23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[25] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010), 036106.

[26] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *SIGKDD*. 305–314.

[27] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.

[28] Nathan Linial, Eran London, and Yuri Rabinovich. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica* 15, 2 (1995), 215–245.

[29] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *NeurIPS*. 8228–8239.

[30] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *ICML*. PMLR, 4212–4221.

[31] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *NeurIPS*. 5711–5722.

[32] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *SIGKDD*. 483–491.

[33] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *SIGKDD*. 483–491.

[34] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. 2019. Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders. *arXiv preprint arXiv:1901.06033* (2019).

[35] Emile Mathieu and Maximilian Nickel. 2020. Riemannian Continuous Normalizing Flows. In *NeurIPS*. 2503–2515.

[36] Leyla Mirvakhabova, Evgeny Frolov, Valentin Khrulkov, Ivan Oseledets, and Alexander Tuzhilin. 2020. Performance of hyperbolic geometry models on top-N recommendation tasks. In *RecSys*. 527–532.

[37] Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. 2019. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *ICML*. 4693–4702.

[38] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*. 6338–6347.

[39] Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*. 3779–3788.

[40] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. 2021. Hyperbolic deep neural networks: A survey. *arXiv preprint arXiv:2101.04562* (2021).

[41] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. 2018. Representation tradeoffs for hyperbolic embeddings. In *ICML*. 4457–4466.

[42] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph Neural Networks for Friend Ranking in Large-scale Social Platforms. In *WWW*. 2535–2546.

[43] Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. 2019. Mixed-curvature Variational Autoencoders. In *ICLR*.

[44] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[45] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In *WWW*. 593–601.

[46] 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.

[47] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*. 1235–1244.

[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[49] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *SIGIR*. 1001–1010.

[50] Richard C Wilson, Edwin R Hancock, Elżbieta Pekalska, and Robert PW Duin. 2014. Spherical and hyperbolic embeddings of data. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2255–2269.

[51] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. 2020. Factorizable Graph Convolutional Networks. *NeurIPS* 33 (2020).

[52] Sixiao Zhang, Hongxu Chen, Xiao Ming, Lizhen Cui, Hongzhi Yin, and Guandong Xu. 2021. Where are we in embedding spaces? A Comprehensive Analysis on Network Embedding Approaches for Recommender Systems. *arXiv preprint arXiv:2105.08908* (2021).

[53] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian Graph Convolutional Networks. In *WWW*. 1249–1261.

[54] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Depeng Jin, and Yong Li. 2020. Disentangling User Interest and Conformity for Recommendation with Causal Embedding. *arXiv preprint arXiv:2006.11011* (2020).


# A SUPPLEMENTARY MATERIAL

## A.1 Neural Networks in the hypersphere model

To ensure the transformed features satisfy the spherical geometry, we define hypersphere matrix-vector multiplication to perform the hypersphere matrix-vector multiplication:

Definition A.1. **(Hypersphere matrix-vector multiplication)** *If $\mathbf{M}: \mathbb{R}^n \to \mathbb{R}^m$ is a linear map with matrix representation, given two points $\boldsymbol{x} = (x_0, \ldots, x_n) \in \mathbb{S}^n_\kappa$, $\boldsymbol{v} = (v_0, \ldots, v_n) \in \mathcal{T}_{\mathbf{0}}\mathbb{S}^n_\kappa$, we have:*

$$\mathbf{M}^{\otimes_\kappa}(\boldsymbol{x}) = \exp_{\mathbf{0}}^{\kappa}(\hat{\mathbf{M}}(\log_{\mathbf{0}}^{\kappa}(\boldsymbol{x}))), \hat{\mathbf{M}}(\boldsymbol{v}) = (0, \mathbf{M}(v_1, \ldots, v_n)). \quad (18)$$

*Let $\mathbf{M}$ be a $m \times n$ matrix, $\mathbf{M}'$ be a $l \times m$ matrix, $\mathbf{x} \in \mathbb{S}^n_\kappa$, $\mathbf{M} \otimes_\kappa \boldsymbol{x} := \mathbf{M}^{\otimes_\kappa}(\boldsymbol{x})$, we have matrix associativity as: $(\mathbf{M}'\mathbf{M}) \otimes_\kappa \boldsymbol{x} = \mathbf{M}' \otimes_\kappa (\mathbf{M} \otimes_\kappa \boldsymbol{x})$.*

The hypersphere matrix-vector multiplication also satisfies following theorem:

Theorem A.1. *Given a point in spherical space, which is represented by $\boldsymbol{x}^n_{\mathbb{S}_\kappa} \in \mathbb{S}^n_\kappa$ using hypersphere model or $\mathbf{x}^n_{\mathbb{D}_\kappa} \in \mathbb{D}^n_{\mathbb{D}_\kappa}$ using projected hypersphere ball model [3], respectively. Let $\mathbf{M}$ be a $m \times n$ matrix, hypersphere matrix-vector multiplication $\mathbf{M} \otimes_\kappa \boldsymbol{x}^n_{\mathbb{S}_\kappa}$ used in hyperboloid model is equivalent to Möbius matrix-vector multiplication $\mathbf{M} \otimes_\kappa \boldsymbol{x}^n_{\mathbb{D}_\kappa}$ used in projected hypersphere model.*

*Proof.* Let $\boldsymbol{x}^n_{\mathbb{S}_\kappa} \in \mathbb{S}^n_\kappa$, $\boldsymbol{v} = (v_0, v_1, \ldots, v_n) \in \mathcal{T}_{\mathbf{0}}\mathbb{S}^n_\kappa$, and $\mathbf{M}$ be a $m \times n$ matrix, hypersphere matrix-vector multiplication is shown as follows:

$$\begin{aligned} \mathbf{M} \otimes_\kappa \boldsymbol{x}^n_{\mathbb{S}_\kappa} &:= \mathbf{M}^{\otimes_\kappa}(\boldsymbol{x}^n_{\mathbb{S}_\kappa}) = \exp_{\mathbf{0}}^{\kappa}\left(\hat{\mathbf{M}}\big(\log_{\mathbf{0}}^{\kappa}(\boldsymbol{x}^n_{\mathbb{S}_\kappa})\big)\right) = \boldsymbol{y}^m_{\mathbb{S}_\kappa}, \\ \hat{\mathbf{M}}(\boldsymbol{v}) &= \big(0, \mathbf{M}(v_1, \ldots, v_n)\big). \end{aligned} \quad (19)$$

Let $\boldsymbol{x}^n_{\mathbb{D}_\kappa} \in \mathbb{D}^n_\kappa$, Möbius matrix-vector multiplication has the formulation as [3]:

$$\begin{aligned}&\mathbf{M} \otimes_\kappa \boldsymbol{x}^n_{\mathbb{D}_\kappa} := \mathbf{M}^{\otimes_\kappa}(\boldsymbol{x}^n_{\mathbb{D}_\kappa}) \\ =&(1/\sqrt{\kappa}) \tan\Big(\frac{\|\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|}{\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|} \tan^{-1}(\sqrt{\kappa}\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|)\Big)\frac{\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}}{\|\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|} = \boldsymbol{y}^m_{\mathbb{D}_\kappa}.\end{aligned} \tag{20}$$

For $P_{\mathbb{S}^n_\kappa \to \mathbb{D}^n_\kappa}(\boldsymbol{x}^n_{\mathbb{S}_\kappa}) = \boldsymbol{x}^n_{\mathbb{D}_\kappa}$ and a shared $m \times n$ matrix $\mathbf{M}$, we aim to prove $P_{\mathbb{S}^m_\kappa \to \mathbb{D}^m_\kappa}(\boldsymbol{y}^m_{\mathbb{S}_\kappa}) = \boldsymbol{y}^m_{\mathbb{D}_\kappa}$.

For $\boldsymbol{x}^n_{\mathbb{S}_\kappa} = (x_{0_{\mathbb{S}_\kappa}}, x_{1_{\mathbb{S}_\kappa}}, \ldots, x_{n_{\mathbb{S}_\kappa}}) \in \mathbb{S}^n_\kappa$, let $\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa} = (x_{1_{\mathbb{S}_\kappa}}, \ldots, x_{n_{\mathbb{S}_\kappa}})$, and the logarithmic map of $\boldsymbol{x}^n_{\mathbb{S}_\kappa}$ at $\mathbf{0} = (1/\sqrt{\kappa}, 0, \ldots, 0) \in \mathbb{S}^n_\kappa$ is shown as follows:

$$\log^\kappa_{\mathbf{0}}(\boldsymbol{x}^n_{\mathbb{S}_\kappa}) = \frac{\cos^{-1}(\sqrt{\kappa}x_{0_{\mathbb{S}_\kappa}})}{\sqrt{1-\kappa x^2_{0_{\mathbb{S}_\kappa}}}}(0, \hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}) = \frac{\cos^{-1}(\sqrt{\kappa}x_{0_{\mathbb{S}_\kappa}})}{\sqrt{\kappa}\|\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|}(0, \hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}). \tag{21}$$

Let $q = \frac{\cos^{-1}(\sqrt{\kappa}x_{0_{\mathbb{S}_\kappa}})}{\sqrt{\kappa}\|\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|}$, $\log^\kappa_{\mathbf{0}}(\boldsymbol{x}^n_{\mathbb{S}_\kappa}) = q(0, \hat{\boldsymbol{x}}_{\mathbb{S}_\kappa})$, so we have:

$$\hat{\mathbf{M}}\big(\log^\kappa_{\mathbf{0}}(\boldsymbol{x}^n_{\mathbb{S}_\kappa})\big) = (0, q\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}) = \boldsymbol{m}. \tag{22}$$

The hypersphere matrix-vector multiplication is given as following:

$$\mathbf{M}\otimes_\kappa \boldsymbol{x}^n_{\mathbb{S}_\kappa} = \Big(\frac{1}{\sqrt{\kappa}}\cos(\sqrt{\kappa}\|\boldsymbol{m}\|), \frac{\sin(\sqrt{\kappa}\|\boldsymbol{m}\|)q}{\sqrt{\kappa}\|\boldsymbol{m}\|}\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\Big) = \boldsymbol{y}^m_{\mathbb{S}_\kappa}. \tag{23}$$

Then we map $\boldsymbol{y}^m_{\mathbb{S}_\kappa}$ to the projected hypersphere model:

$$P_{\mathbb{S}^n_\kappa \to \mathbb{D}^n_\kappa}(\boldsymbol{y}^m_{\mathbb{S}_\kappa}) = \frac{1}{\sqrt{\kappa}}\tan\Big(\|\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|\frac{\cos^{-1}(\sqrt{\kappa}x_{0_{\mathbb{S}_\kappa}})}{2\|\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|}\Big)\frac{\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}}{\|\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|}. \tag{24}$$

Note that $\|\boldsymbol{m}\| = \sqrt{\langle \boldsymbol{m}, \boldsymbol{m}\rangle} = \|q\mathbf{M}\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|$, and $x_{0_{\mathbb{S}_\kappa}} = \sqrt{1/\kappa - \|\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|^2}$. Moreover, the point $\boldsymbol{x}^n_{\mathbb{D}_\kappa} = (x_{1_{\mathbb{D}_\kappa}}, \ldots, x_{n_{\mathbb{D}_\kappa}}) \in \mathbb{D}^n_\kappa$ can be mapped into the hypersphere model as following:

$$\begin{aligned}P_{\mathbb{D}^n_\kappa \to \mathbb{S}^n_\kappa}(\boldsymbol{x}^n_{\mathbb{D}_\kappa}) &= \frac{(1/\sqrt{\kappa} - \sqrt{\kappa}\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|^2, 2x_{1_{\mathbb{D}_\kappa}}, \ldots, 2x_{n_{\mathbb{D}_\kappa}})}{1+\kappa\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|^2} \\ &= (x_{0_{\mathbb{S}_\kappa}}, x_{1_{\mathbb{S}_\kappa}}, \ldots, x_{n_{\mathbb{S}_\kappa}}).\end{aligned} \tag{25}$$

Thus, the squared norm of $\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}$ is given as:

$$\|\hat{\boldsymbol{x}}_{\mathbb{S}_\kappa}\|^2 = \sum_{i=1}^n \Big(\frac{2}{1+\kappa\|\boldsymbol{x}_{i_{\mathbb{D}_\kappa}}\|^2}\Big)^2 (x_{i_{\mathbb{D}_\kappa}})^2 = \Big(\frac{2\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|}{1+\kappa\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|^2}\Big)^2. \tag{26}$$

By combining Eq. (24) and Eq. (26), we have:

$$\begin{aligned}&P_{\mathbb{S}^m_\kappa \to \mathbb{D}^m_\kappa}(\boldsymbol{y}^m_{\mathbb{S}_\kappa}) \\ =&(1/\sqrt{\kappa}) \tan\Big(\frac{\|\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|}{\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|} \tan^{-1}(\sqrt{\kappa}\|\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|)\Big)\frac{\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}}{\|\mathbf{M}\boldsymbol{x}^n_{\mathbb{D}_\kappa}\|} = \boldsymbol{y}^m_{\mathbb{D}_\kappa}.\end{aligned} \tag{27}$$

Therefore, hypersphere matrix-vector multiplication is equivalence to Möbius matrix-vector multiplication. □

Also, to apply **non-linear transformation** on the hypersphere model, the hypersphere pointwise non-linear activation can be derived as:

Definition A.2. **Hypersphere pointwise non-linear activation** *If $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is a pointwise non-linearity map, given two points $\mathbf{x} = (x_0, \cdots, x_n) \in \mathbb{S}^n_\kappa$ and $\mathbf{v} = (v_0, \cdots, v_n) \in \mathcal{T}_{\mathbf{0}}\mathbb{S}^n_\kappa$, the hypersphere version $\sigma_{\otimes_\kappa}$ is:*

$$\sigma^{\otimes_\kappa}(\boldsymbol{x}) = \exp^\kappa_{\mathbf{0}}(\hat{\sigma}^{\otimes_\kappa}(\log^\kappa_{\mathbf{0}}(\boldsymbol{x}))), \hat{\sigma}^{\otimes_\kappa}(\boldsymbol{v}) = (0, \sigma(v_1), \ldots, \sigma(v_n))). \tag{28}$$

The hypersphere pointwise non-linear activation has the following property:

Theorem A.2. *Given a point in hyperbolic space, it is modeled by $\mathbf{x}^{n,\beta} \in \mathbb{H}^{n,\beta}$ using hyperboloid model and $\mathbf{x}^{n,\alpha} \in \mathbb{D}^{n,\alpha}$ using Poincaré ball model, respectively. Lorentzian pointwise non-linearity $\sigma^{\otimes^\beta}(\mathbf{x}^{n,\beta})$ in the hyperboloid model is equivalent to Möbius pointwise non-linearity $\sigma^{\otimes^\alpha}(\mathbf{x}^{n,\alpha})$ in the Poincaré ball model [10], when $\sigma(\cdot)$ indicates some specific non-linear activation, e.g., Relu, leaklyRelu.*

The proof of Theorem A.2 is similar to Theorem A.1, and we omit it due to the page length limit.