Hypercomplex Graph Collaborative Filtering

Anchen Li
College of Computer Science and Technology, Jilin University
Changchun, China
liac20@mails.jlu.edu.cn

Bo Yang*
College of Computer Science and Technology, Jilin University
Changchun, China
ybo@jlu.edu.cn

Huan Huo
University of Technology Sydney
Sydney, Australia
Huan.Huo@uts.edu.au

Farookh Hussain
University of Technology Sydney
Sydney, Australia
Farookh.Hussain@uts.edu.au

ABSTRACT

Hypercomplex algebras are well-developed in the area of mathematics. Recently, several hypercomplex recommendation approaches have been proposed and yielded great success. However, two vital issues have not been well-considered in existing hypercomplex recommenders. First, these methods are only designed for specific and low-dimensional hypercomplex algebras (e.g., complex and quaternion algebras), ignoring the exploration and utilization of high-dimensional ones. Second, most recommenders treat every user-item interaction as an isolated data instance, without considering high-order collaborative relationships.

To bridge these gaps, in this paper, we propose a novel recommendation framework named **H**yper**C**omplex **G**raph **C**ollaborative **F**iltering (HCGCF). To study the high-dimensional hypercomplex algebras, we introduce Cayley–Dickson construction which utilizes a recursive process to define hypercomplex algebras and their mathematical operations. Based on Cayley–Dickson construction, we devise a hypercomplex graph convolution operator to learn user and item representations. Specifically, the operator models both the neighborhood summary and interaction relations with neighbors in hypercomplex spaces, effectively exploiting the high-order connectivity in the user-item bipartite graph. To the best of our knowledge, it is the first time that Cayley-Dickson construction and graph convolution techniques have been explicitly discussed and used in hypercomplex recommender systems. Compared with several state-of-the-art recommender baselines, HCGCF achieves superior performance in both click-through rate prediction and top-K recommendation on three real-world datasets.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Recommendation, Collaborative Filtering, Hypercomplex Spaces, Graph Convolutional Networks, Cayley–Dickson Construction.

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '22, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512065

ACM Reference Format:
Anchen Li, Bo Yang, Huan Huo, and Farookh Hussain. 2022. Hypercomplex Graph Collaborative Filtering. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512065

1 INTRODUCTION

Collaborative filtering (CF) has become one of the dominant techniques used in recommendation [17, 20, 25, 26, 44]. Learning high-quality user and item representations forms the crux of CF. Most CF models primarily learn representations in the real number system, ignoring the great potential of alternative algebra systems.

Recently, there is a surge in studying the recommendation models in hypercomplex spaces (non-real spaces) [7, 27, 38, 49]. In mathematics, hypercomplex algebra is defined as unital, distributive algebra, not necessarily associative or commutative [23]. Compared with traditional real-valued representations, hypercomplex representations consist of one real component and finite separate hyperimaginary components. This is in a similar spirit to multi-head approaches. Thereby, hypercomplex algebras have a richer representation capacity for modeling users and items. In addition, instead of regular (real) dot products, the hypercomplex multiplication captures latent inter-dependencies between the multiple components. Such product rules can be used in modeling the user-item relations, which encourages a more compact interaction to form the final representation. Based on these advantages, researchers propose diverse hypercomplex recommendation models which boost the performance over some real-valued recommenders [7, 49].

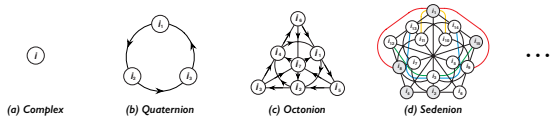


Figure 1: The mnemonics for the multiplications of the unit hypercomplex numbers. "i." is the hyperimaginary unit.

Despite effectiveness, we argue that existing hypercomplex CF approaches fall short in two aspects. First, high-dimensional hypercomplex algebras are under-explored. Current methods are designed for specific and low-dimensional hypercomplex representation learning (e.g., complex and quaternion algebras) [7, 49]. However, there exist high-dimensional numbers in hypercomplex algebra families. As shown in Figure 1[1], high-dimensional hypercomplex numbers have more hyperimaginary components and more complex multiplication rules. Intuitively, these high-dimensional

[1]The figure is obtained from [2, 31]

hypercomplex algebras are more expressive, while they are not explored and analyzed by the existing hypercomplex recommendation algorithms. Second, high-order user-item relationships are not captured. Recent studies on adopting graph convolution techniques in recommendation have shown the importance of explicitly modeling high-order user-item relations [17, 44]. Nevertheless, most hypercomplex CF recommenders exploit variants of the traditional matrix factorization (MF) framework [7, 49], which neglects such high-order user-item information.

Motivated by the aforementioned gaps, in this paper, we study the high-dimensional hypercomplex algebras for CF by utilizing GCN techniques. Indeed, it is a non-trivial task. Since the mathematical operations in hypercomplex algebras are different from those in the real number system, most well-established real-valued GCN recommenders can not be directly applied to the hypercomplex spaces. In addition, with the increase of the hypercomplex algebra dimension, mathematical operations (e.g., hypercomplex multiplication) become more intricate. How can we devise a hypercomplex GCN recommendation model in an elegant way?

To solve the above technical challenges, we propose a novel recommendation framework named HCGCF, short for **H**yper**C**omplex **G**raph **C**ollaborative **F**iltering. There are mainly two innovative points of HCGCF. First, we introduce Cayley–Dickson construction to explore high-dimensional hypercomplex spaces. Cayley–Dickson construction utilizes a recursive process to yield a sequence of hypercomplex algebras known as Cayley-Dickson algebras, each with twice the dimension of the previous one [11, 23]. Figure 1 shows the Cayley-Dickson algebra sequence. In Cayley-Dickson algebras, mathematical operations of higher-dimensional algebras are defined by the previous smaller ones, which provides an elegant algebraic formalism for studying the hypercomplex number systems. Second, based on Cayley-Dickson algebras, we devise a hypercomplex graph convolution operator to learn node representations by aggregating neighbor information in the hypercomplex spaces. The operator is equipped with a hypercomplex linear aggregator and a hypercomplex interaction aggregator. The linear aggregator uses hypercomplex sum pooling to learn a summary of the neighborhood. The interaction aggregator applies hypercomplex multiplication between the center node and neighbor embeddings to capture neighbor feature interactions. Specially, since hypercomplex multiplication in high-dimensional algebras is asymmetry, such multiplication provides a strong inductive bias for reflecting the asymmetric user-item interaction relations, i.e., users and items fundamentally belong to a separate class of entities [49].

To the best of our knowledge, this is the first work that utilizes Cayley–Dickson construction and GCN techniques for hypercomplex CF approaches. We compare HCGCF with several state-of-the-art recommenders, including real-valued methods and hypercomplex-valued methods, in three scenarios about book, movie, and music recommendation. Experimental results demonstrate that HCGCF achieves highly competitive results in both click-through rate prediction and top-K recommendation.

To summarize, our contributions are highlighted as follows:

- We propose HCGCF to study high-dimensional hypercomplex algebras and high-order user-item relations in recommendation.
- We provide an elegant approach that studies high-dimensional hypercomplex spaces by using Cayley–Dickson construction.
- Experiment results on three real-world datasets prove the efficacy of our HCGCF over several state-of-the-art baselines.

2 PRELIMINARIES

Our work investigates the notion of learning user and item representations in hypercomplex spaces. Specifically, we design our recommendation framework based on Cayley-Dickson algebras which is a representative hypercomplex number system. In this section, we cover the important background of the hypercomplex number and Cayley–Dickson algebras.

2.1 Hypercomplex Number

An n-dimensional hypercomplex number $h_x \in \mathbb{H}_n$ expanded on the real vector space has a representation in the form, as follows:

$$h_x = x_1\mathbf{i}_1 + x_2\mathbf{i}_2 + \cdots + x_n\mathbf{i}_n = \sum\nolimits_{k=1}^{n} x_k\mathbf{i}_k, \tag{1}$$

where $x_1, x_2, \cdots, x_n$ are real numbers called the components. The elements $\mathbf{i}_1, \mathbf{i}_2, \cdots, \mathbf{i}_n$ are called hyperimaginary units, where $\mathbf{i}_1 = 1$ represents the vector identity element [1, 23].

2.2 Cayley–Dickson Algebra

The Cayley-Dickson algebra $\mathcal{A}$ is a sequence of hypercomplex algebras obtained from the real numbers by the Cayley-Dickson construction [2, 11]. Higher-dimensional Cayley-Dickson algebras are obtained by doubling the previous smaller ones in the Cayley-Dickson construction [24]. Thus, the dimension of these algebras is a power of two. Specifically, such a construction procedure utilizes the m-th algebra $\mathcal{A}_m \in \mathbb{H}_{2^m}$ in the sequence to define the $(m{+}1)$-th algebra $\mathcal{A}_{m+1} \in \mathbb{H}_{2^{m+1}}$ as follows:

$$\mathcal{A}_{m+1} = \{h_a + h_b\mathbf{i}_{2^m+1}\}, \text{ and } h_a, h_b \in \mathcal{A}_m, \tag{2}$$

where $m \in \mathbb{N}$ and $\mathcal{A}_0 = \mathbb{R}$. Here $\mathbf{i}_{2^m+1} \notin \mathcal{A}_m$ is the additional hyperimaginary unit for doubling the dimension of $\mathcal{A}_m$, satisfying the following rules: $(\mathbf{i}_{2^m+1})^2 = -1$, $\mathbf{i}_1\mathbf{i}_{2^m+1} = \mathbf{i}_{2^m+1}\mathbf{i}_1$, and $\mathbf{i}_o\mathbf{i}_{2^m+1} = -\mathbf{i}_{2^m+1}\mathbf{i}_o = \mathbf{i}_{2^m+o}$ for all $o = 2, 3, \cdots, 2^m$.

The mathematical operations for Cayley-Dickson algebras are defined recursively [2, 10, 11]. For $h_x = h_a + h_b\mathbf{i}_{2^m+1} \in \mathcal{A}_{m+1}$, $h_y = h_c + h_d\mathbf{i}_{2^m+1} \in \mathcal{A}_{m+1}$; $h_a \in \mathcal{A}_m$, $h_b \in \mathcal{A}_m$, $h_c \in \mathcal{A}_m$, $h_d \in \mathcal{A}_m$; and $\gamma \in \mathbb{R}$, several widely utilized operations for Cayley-Dickson algebras are introduced as follows:

- The *addition* of h_x and h_y is defined as:

$$h_x \oplus_{\mathrm{m}+1} h_y = (h_a \oplus_{\mathrm{m}} h_c) + (h_b \oplus_{\mathrm{m}} h_d)\mathbf{i}_{2^m+1}. \tag{3}$$

  The *subtraction* follows the same principle analogously but flipping $\oplus$ with $\ominus$.
- The *conjugate* of h_x is defined as:

$$\overline{h}_x = \overline{h}_a - h_b\mathbf{i}_{2^m+1}. \tag{4}$$

  The conjugation for every $a \in \mathbb{R}$ is defined as: $\overline{a} = a$.
- The *multiplication* of h_x and h_y is defined as:

$$\begin{aligned} h_x \otimes_{\mathrm{m}+1} h_y = &\left(h_a \otimes_{\mathrm{m}} h_c \ominus_{\mathrm{m}} \overline{h}_d \otimes_{\mathrm{m}} h_b\right) \\ &+ \left(h_a \otimes_{\mathrm{m}} h_d \oplus_{\mathrm{m}} h_b \otimes_{\mathrm{m}} \overline{h}_c\right)\mathbf{i}_{2^m+1}. \end{aligned} \tag{5}$$

  When $m \geq 2$, the multiplication is asymmetry.

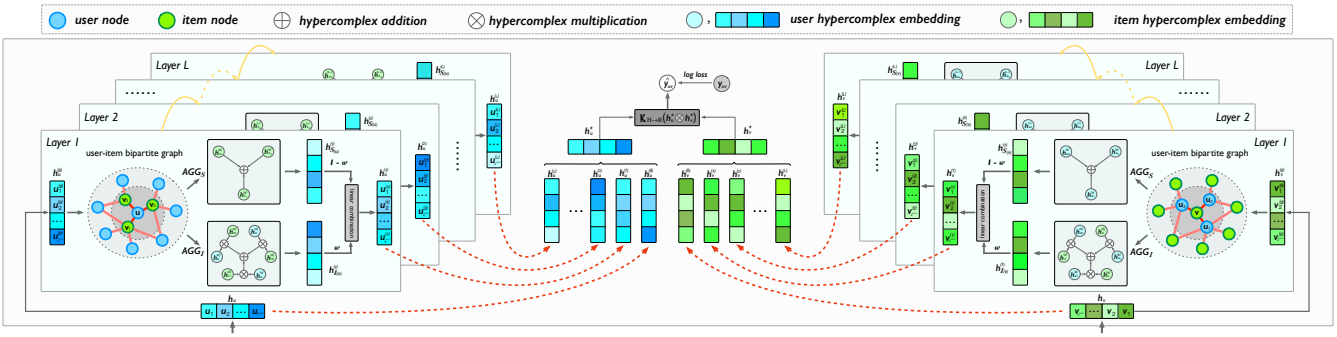


Figure 2: The overall structure of our HCGCF framework. The framework is based on Cayley–Dickson algebras.

- The *scalar multiplication* of h_x by γ is defined as:

$$\gamma h_x = \gamma h_a + \gamma h_b \mathbf{i}_{2^m+1}. \tag{6}$$

Based on this form of mathematical operations, we can use the low-dimensional algebra operations to study the high-dimensional ones recursively. We will make use of these mathematical operations of Cayley–Dickson algebras to design our method HCGCF.

3 METHODOLOGY

In this section, we first formulate the problems. Then, we elaborate on the HCGCF framework. Lastly, we discuss the optimization process of our HCGCF.

3.1 Problem Formulation

We focus on the CF task with implicit feedback. Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{V}$, the historical interaction matrix is represented by $\mathbf{Y} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{V}|}$, in which $y_{uv} = 1$ indicates that user $u \in \mathcal{U}$ has engaged with item $v \in \mathcal{V}$ and zero otherwise. The goal of our HCGCF is to use the interaction matrix $\mathbf{Y}$ to predict user u's preference for item v that he has never engaged with before.

3.2 The Proposed Model

We introduce the overview of our HCGCF framework. The framework is based on Cayley–Dickson algebras. There are three parts in HCGCF: (i) the embedding layer that initializes embeddings for users and items in the hypercomplex space; (ii) hypercomplex graph convolution layers that learn the hypercomplex embeddings by exploiting the high-order user-item relationships; and (iii) the prediction layer that generates the preference score of a user-item pair based on the refined embeddings. Figure 2 illustrates the HCGCF's architecture and we will detail each part in the following.

3.2.1 ***Embedding Layer.*** The embedding layer encodes users and items with Cayley–Dickson representations. Specifically, given a user-item pair (u, v), user u's embedding $\mathbf{h}_u \in \mathcal{A}^e_{m+1}$ and item v's embedding $\mathbf{h}_v \in \mathcal{A}^e_{m+1}$ are formulated according the definition of Cayley–Dickson algebras, as follows:

$$\mathbf{h}_u = \mathbf{a}_u + \mathbf{b}_u \mathbf{i}_{2^m+1} = \sum_{k=1}^{2^{m+1}} \mathbf{u}_k \mathbf{i}_k, \tag{7}$$

$$\mathbf{h}_v = \mathbf{c}_v + \mathbf{d}_v \mathbf{i}_{2^m+1} = \sum_{k=1}^{2^{m+1}} \mathbf{v}_k \mathbf{i}_k, \tag{8}$$

where $m \in \mathbb{N}$; $\mathbf{a}_u = \sum_{k=1}^{2^m} \mathbf{u}_k \mathbf{i}_k \in \mathcal{A}^e_m$ and $\mathbf{b}_u = \sum_{k=2^m+1}^{2^{m+1}} \mathbf{u}_k \mathbf{i}_{k-2^m} \in \mathcal{A}^e_m$ are the subalgebras of $\mathbf{h}_u$, and $\mathbf{c}_v = \sum_{k=1}^{2^m} \mathbf{v}_k \mathbf{i}_k \in \mathcal{A}^e_m$ and $\mathbf{d}_v = \sum_{k=2^m+1}^{2^{m+1}} \mathbf{v}_k \mathbf{i}_{k-2^m} \in \mathcal{A}^e_m$ are the subalgebras of $\mathbf{h}_v$; $\mathbf{u}_k \in \mathbb{R}^e$ and $\mathbf{v}_k \in \mathbb{R}^e$ are the real-valued representations that correspond to the component k; and e is the component dimension size. Details of the embedding initialization are provided in subsection 3.3.1.

3.2.2 ***Aggregation Layer.*** We design a hypercomplex graph convolution (HGC) layer that aggregates the neighborhood information to refine hypercomplex embeddings. We first illustrate the process from a single user-item pair and then provide the matrix form of the HGC layer. Since the user and item aggregation processes are similar, we take the user side as an example for illustration.

The HGC layer consists of a hypercomplex linear aggregator and a hypercomplex interaction aggregator. The linear aggregator models a summary of the neighborhood, whose design is inspired by the GCNs [21]. In addition, since the feature interaction is important for recommendation [7, 49], we carefully design an interaction aggregator to capture neighbor feature interactions.

Specifically, given user u and her first-order neighborhood $\mathcal{N}_u$, the $(l+1)$-th HGC layer is defined as follows:

$$\mathbf{h}_u^{(l+1)} = \mathcal{F}_c \left(\mathrm{AGG}_{\mathcal{S}}(\mathbf{h}_v^{(l)} | v \in \mathcal{N}_u), \mathrm{AGG}_{\mathcal{I}}(\mathbf{h}_v^{(l)} | v \in \mathcal{N}_u) \right), \tag{9}$$

where $\mathbf{h}_u^{(l)}$ and $\mathbf{h}_v^{(l)}$ are the user u's and item v's representations at the l-th layer respectively, and $\mathbf{h}_u^{(0)} = \mathbf{h}_u$ and $\mathbf{h}_v^{(0)} = \mathbf{h}_v$ are the initial representations from the embedding layer; $\mathrm{AGG}_{\mathcal{S}}$ and $\mathrm{AGG}_{\mathcal{I}}$ are the aggregation functions for the linear aggregator and the interaction aggregator, respectively; and function $\mathcal{F}_c : \mathcal{A}^e_{m+1} \times \mathcal{A}^e_{m+1} \rightarrow \mathcal{A}^e_{m+1}$ is utilized to combine the hypercomplex representations.

Following the spirit of the GCNs [21], $\mathrm{AGG}_{\mathcal{S}}$ takes the summation of the neighbors' hypercomplex representations to describe the neighborhood information, as follows:

$$\begin{aligned} \mathbf{h}_{\mathcal{S}(u)}^{(l+1)} &= \mathrm{AGG}_{\mathcal{S}}(\mathbf{h}_v^{(l)} | v \in \mathcal{N}_u) \\ &= \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_{m+1}} \frac{1}{p_{uv}} \mathbf{h}_v^{(l)} \\ &= \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_m} \frac{1}{p_{uv}} \mathbf{c}_v^{(l)} + \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_m} \frac{1}{p_{uv}} \mathbf{d}_v^{(l)} \mathbf{i}_{2^m+1} \\ &= \sum_{k=1}^{2^{m+1}} \sum_{v\in\mathcal{N}_u} \frac{1}{p_{uv}} \mathbf{v}_k^{(l)} \mathbf{i}_k, \end{aligned} \tag{10}$$

where $\sum^{\oplus_{\mathrm{m}+1}}$ is the accumulation of the hypercomplex addition for the $(m+1)$-th Cayley-Dickson algebra, and p_{uv} is a normalization constant which is set as $\sqrt{|\mathcal{N}_u||\mathcal{N}_v|}$ (symmetric normalization).

Feature interactions, which have been demonstrated to be useful for many recommendation tasks [16, 35, 44], are also very important signals to model the pairwise relational information between the center node and every neighbor. In HGC, we devise aggregator $\mathrm{AGG}_{\mathcal{I}}$ to capture neighbor feature interactions by applying the hypercomplex multiplication, as follows:

$$\begin{aligned}\mathbf{h}_{\mathcal{I}(u)}^{(l+1)} &= \mathrm{AGG}_{\mathcal{I}}(\mathbf{h}_v^{(l)}|v\in\mathcal{N}_u)\\ &= \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_{\mathrm{m}+1}} \frac{1}{p_{uv}}\mathbf{h}_u^{(l)} \otimes_{\mathrm{m}+1} \mathbf{h}_v^{(l)}\\ &= \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_{\mathrm{m}+1}} \frac{1}{p_{uv}}\left(\mathbf{a}_u^{(l)} + \mathbf{b}_u^{(l)}\mathbf{i}_{2^m+1}\right) \otimes_{\mathrm{m}+1} \left(\mathbf{c}_v^{(l)} + \mathbf{d}_v^{(l)}\mathbf{i}_{2^m+1}\right)\\ &= \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_{\mathrm{m}}} \frac{1}{p_{uv}}\left(\mathbf{a}_u^{(l)} \otimes_{\mathrm{m}} \mathbf{c}_v^{(l)} \ominus_{\mathrm{m}} \overline{\mathbf{d}}_v^{(l)} \otimes_{\mathrm{m}} \mathbf{b}_u^{(l)}\right) \qquad (11)\\ &\quad + \sum\nolimits_{v\in\mathcal{N}_u}^{\oplus_{\mathrm{m}}} \frac{1}{p_{uv}}\left(\mathbf{a}_u^{(l)} \otimes_{\mathrm{m}} \mathbf{d}_v^{(l)} \oplus_{\mathrm{m}} \mathbf{b}_u^{(l)} \otimes_{\mathrm{m}} \overline{\mathbf{c}}_v^{(l)}\right)\mathbf{i}_{2^m+1},\end{aligned}$$

where $\overline{\mathbf{c}}_v^{(l)} = \mathbf{v}_1^{(l)}\mathbf{i}_1 - \sum_{k=2}^{2^m}\mathbf{v}_k^{(l)}\mathbf{i}_k$ and $\overline{\mathbf{d}}_v^{(l)} = \mathbf{v}_{2^m+1}^{(l)}\mathbf{i}_1 - \sum_{k=2^m+2}^{2^{m+1}}\mathbf{v}_k^{(l)}\mathbf{i}_{k-2^m}$ are the conjugates of $\mathbf{c}_v^{(l)}$ and $\mathbf{d}_v^{(l)}$, respectively.

We initiate this process by defining multiplication $\otimes_1$. For two Cayley-Dickson representations $\mathbf{h}_x = \mathbf{a}_x + \mathbf{b}_x\mathbf{i} \in \mathcal{A}_1^e$ and $\mathbf{h}_y = \mathbf{c}_y + \mathbf{d}_y\mathbf{i} \in \mathcal{A}_1^e$, the multiplication between them is defined as:

$$\mathbf{h}_x \otimes_1 \mathbf{h}_y = \left(\mathbf{a}_x \odot \mathbf{c}_y - \mathbf{d}_y \odot \mathbf{b}_x\right) + \left(\mathbf{a}_x \odot \mathbf{d}_y + \mathbf{b}_x \odot \mathbf{c}_y\right)\mathbf{i}, \quad (12)$$

where $\odot$ is the element-wise multiplication between the real-valued representations. Different from the multiplication in the real number system, the hypercomplex multiplication explicitly captures latent inter-dependencies between all the components, which encourages a more compact interaction of the neighborhoods. In addition, such multiplication in high-dimensional algebras ($m \geq 2$) is asymmetry, which provides an inductive bias for reflecting the asymmetric user-item relations [49].

To combine the linear aggregation and the interaction aggregation, we adopt a linear combination scheme for function $\mathcal{F}_c$, as:

$$\mathbf{h}_u^{(l+1)} = \left((1-\omega)\cdot\mathbf{h}_{\mathcal{S}(u)}^{(l+1)}\right) \oplus_{\mathrm{m}+1} \left(\omega\cdot\mathbf{h}_{\mathcal{I}(u)}^{(l+1)}\right), \qquad (13)$$

where $\omega \in [0,1]$ is a trade-off factor.

Although the feature transformations and non-linear activations are two standard operations in the graph convolution operations, recent works found that these operations are not necessary and might degrade the performance of GCN-based CF recommenders [6, 17]. Therefore, we discard the transformation matrix and activation function in each HGC layer.

Matrix Form. To facilitate implementation, we provide the matrix form of the HGC layers. We first use $\mathbf{H}_u = \sum_{k=1}^{2^{m+1}}\mathbf{U}_k\mathbf{i}_k \in \mathcal{A}_{m+1}^{|\mathcal{U}|\times e}$ and $\mathbf{H}_v = \sum_{k=1}^{2^{m+1}}\mathbf{V}_k\mathbf{i}_k \in \mathcal{A}_{m+1}^{|\mathcal{V}|\times e}$ to denote the user and item hypercomplex embedding matrices respectively, where $\mathbf{U}_k \in \mathbb{R}^{|\mathcal{U}|\times e}$ and $\mathbf{V}_k \in \mathbb{R}^{|\mathcal{V}|\times e}$ are real-valued matrices that correspond to the component k. In the following, we show how to update hypercomplex embeddings from the l-th layer to the $(l+1)$-th layer with the matrix operations. Let $\mathbf{H}^{(0)} \in \mathcal{A}_{m+1}^{(|\mathcal{U}|+|\mathcal{V}|)\times e}$ be the 0-th layer hypercomplex embedding matrix, as follows:

$$\mathbf{H}^{(0)} = \left[\mathbf{H}_u^{(0)}, \mathbf{H}_v^{(0)}\right] = \sum_{k=1}^{2^{m+1}}\left[\mathbf{U}_k^{(0)}, \mathbf{V}_k^{(0)}\right]\mathbf{i}_k, \qquad (14)$$

where $\mathbf{H}_u^{(0)} = \mathbf{H}_u$ and $\mathbf{H}_v^{(0)} = \mathbf{H}_v$ are the 0-th layer hypercomplex embedding matrices for users and items respectively, and $[\cdot,\cdot]$ is the concatenation operation. Then we obtain the matrix equivalent form of the $(l+1)$-th HGC layer, as follows:

$$\mathbf{H}_{\mathcal{S}}^{(l+1)} = \left[\mathbf{H}_{\mathcal{S}_u}^{(l+1)}, \mathbf{H}_{\mathcal{S}_v}^{(l+1)}\right] = \sum_{k=1}^{2^{m+1}}\tilde{\mathbf{A}}\left[\mathbf{U}_k^{(l)}, \mathbf{V}_k^{(l)}\right]\mathbf{i}_k, \qquad (15)$$

$$\mathbf{H}_{\mathcal{I}}^{(l+1)} = \left[\mathbf{H}_u^{(l)} \otimes_{\mathrm{m}+1} \mathbf{H}_{\mathcal{S}_u}^{(l+1)}, \mathbf{H}_{\mathcal{S}_v}^{(l+1)} \otimes_{\mathrm{m}+1} \mathbf{H}_v^{(l)}\right], \qquad (16)$$

$$\mathbf{H}^{(l+1)} = \left((1-\omega)\cdot\mathbf{H}_{\mathcal{S}}^{(l+1)}\right) \oplus_{\mathrm{m}+1} \left(\omega\cdot\mathbf{H}_{\mathcal{I}}^{(l+1)}\right), \qquad (17)$$

where $\tilde{\mathbf{A}}$ denotes the Laplacian matrix for the user-item bipartite graph, which is defined as follows:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \text{ and } \mathbf{A} = \begin{bmatrix}\mathbf{0} & \mathbf{Y}\\ \mathbf{Y}^\top & \mathbf{0}\end{bmatrix}. \qquad (18)$$

Here, $\mathbf{D}$ is the diagonal degree matrix, $\mathbf{A}$ is the adjacency matrix, $\mathbf{Y}$ is the user-item interaction matrix, and $\mathbf{0}$ is the all-zero matrix.

Since multiplication in high-dimensional hypercomplex algebras is not commutative, we unified utilize left-multiplication ($\mathbf{h}_u^{(l)} \otimes_{\mathrm{m}+1} \mathbf{h}_v^{(l)}$) to model user-item relations. We can also unified use right-multiplication, which does not change the geometrical meaning.

3.2.3 ***Prediction Layer.*** After L layers' aggregation, we utilize the holistic connection to combine the embeddings obtained at each layer (including the initial embeddings) as user u's final representation $\mathbf{h}_u^*$ and item v's final representation $\mathbf{h}_v^*$, as follows:

$$\mathbf{h}_u^* = \frac{1}{L+1}\sum\nolimits_{l=0\cdots L}^{\oplus_{\mathrm{m}+1}}\mathbf{h}_u^{(l)} = \sum_{k=1}^{2^{m+1}}\sum_{l=0}^{L}\frac{1}{L+1}\mathbf{u}_k^{(l)}\mathbf{i}_k, \qquad (19)$$

$$\mathbf{h}_v^* = \frac{1}{L+1}\sum\nolimits_{l=0\cdots L}^{\oplus_{\mathrm{m}+1}}\mathbf{h}_v^{(l)} = \sum_{k=1}^{2^{m+1}}\sum_{l=0}^{L}\frac{1}{L+1}\mathbf{v}_k^{(l)}\mathbf{i}_k. \qquad (20)$$

By doing so, we consider different semantics in different layers to enrich the initial embeddings with the aggregation layers. Here we utilize a mean pooling as the information fusion strategy of the final representations. Note that besides this strategy, other alternatives such as max pooling and attention mechanism can also be applied.

To predict the user-item score, we first use the hypercomplex multiplication to model the interaction between the user aggregation representation $\mathbf{h}_u^*$ and the item aggregation representation $\mathbf{h}_v^*$, and then feed the interaction representation into a function $\mathcal{K}_{\mathbb{H}\to\mathbb{R}}$:

$$\hat{y}_{uv} = \mathcal{K}_{\mathbb{H}\to\mathbb{R}}(\mathbf{h}_u^* \otimes_{\mathrm{m}+1} \mathbf{h}_v^*). \qquad (21)$$

Here we implement the interaction between $\mathbf{h}_u^*$ and $\mathbf{h}_v^*$ as the multiplication. One can utilize various interaction functions, such as hypercomplex neural networks. Function $\mathcal{K}_{\mathbb{H}\to\mathbb{R}}$ is utilized to transform a Cayley–Dickson representation into a scalar. Taking

$\mathbf{h}_x \in \mathcal{A}^e_{m+1}$ as input, function $\mathcal{K}_{\mathbb{H}\to\mathbb{R}}$ is defined as follows:

$$\mathcal{K}_{\mathbb{H}\to\mathbb{R}}(\mathbf{h}_x) = \frac{1}{2^{m+1}} \sum_{k=1}^{2^{m+1}} \sigma\big(\sum_{j=1}^{e} x_{k_j}\big). \tag{22}$$

Following [49], we apply the split activation function (*sigmoid*: $\sigma(x) = 1/(1+e^{-x})$) on each component because the split approach can lead to better stability and simpler computation [29].

3.3 Model Optimization

In this subsection, we first present the parameter initialization and then discuss the loss function of HCGCF.

3.3.1 ***Initialization.*** Many schemes of initializing hypercomplex parameters are proposed [13, 29, 37]. However, in our experiments, these proposed methods performed comparable or worse than the standard initialization such as Xavier initialization [14]. Therefore, we initialize all the components (i.e., $\mathbf{H}_u : \{\mathbf{U}_k\}_{k=1}^{2^{m+1}}$, $\mathbf{H}_v : \{\mathbf{V}_k\}_{k=1}^{2^{m+1}}$) in the hypercomplex embeddings independently with Xavier method. For user u and item v, their embeddings are defined as follows:

$$\mathbf{h}_u = \sum\nolimits_{k=1}^{2^{m+1}} \mathbf{U}_k^\top u_{id}\mathbf{i}_k, \;\; \mathbf{h}_v = \sum\nolimits_{k=1}^{2^{m+1}} \mathbf{V}_k^\top v_{id}\mathbf{i}_k, \tag{23}$$

where $u_{id} \in \mathbb{R}^{|\mathcal{U}|\times 1}$ and $v_{id} \in \mathbb{R}^{|\mathcal{V}|\times 1}$ encode the ID features of user u and item v, respectively.

3.3.2 ***Loss Function.*** To learn the model parameters, we utilize the cross-entropy loss, which is widely used in CF tasks with implicit feedback [8, 19, 20], as follows:

$$\begin{aligned}\min_{\Theta} \mathcal{L} &= \mathcal{L}_{\text{Rec}} + \lambda||\Theta||_2^2 \\ &= -\sum_{(u,v,\tilde{v})\in\mathcal{D}} \Big(y_{uv}\log(\hat{y}_{uv}) + (1-y_{u\tilde{v}})\log(1-\hat{y}_{u\tilde{v}})\Big) \\ &\quad + \lambda\Big(\sum\nolimits_{k=1}^{2^{m+1}} \|\mathbf{U}_k\|_2^2 + \sum\nolimits_{k=1}^{2^{m+1}} \|\mathbf{V}_k\|_2^2\Big),\end{aligned} \tag{24}$$

where $\mathcal{L}_{\text{Rec}}$ measures the recommendation loss. $\mathcal{D}$ is the set of training triplets: $\mathcal{D} = \{(u,v,\tilde{v}) \mid u \in \mathcal{U} \wedge v \in \mathcal{N}_u \wedge \tilde{v} \in \mathcal{V}\backslash\mathcal{N}_u\}$.

The second term in Equation (24) is the L2-regularizer, where $\Theta = \{\{\mathbf{U}_k\}_{k=1}^{2^{m+1}}, \{\mathbf{V}_k\}_{k=1}^{2^{m+1}}\}$ is the parameter set and λ is the balancing factor controlling the strength of the regularization.

3.4 Complexity Analysis

In this subsection, we discuss the model size and the time complexity of HCGCF.

3.4.1 ***Model Size.*** The embedding parameters in our method HCGCF include users' embedding matrix $\mathbf{H}_u : \{\mathbf{U}_k\}_{k=1}^{2^{m+1}}$ and items' embedding matrix $\mathbf{H}_v : \{\mathbf{V}_k\}_{k=1}^{2^{m+1}}$, where $\mathbf{U}_k \in \mathbb{R}^{|\mathcal{U}|\times e}, \mathbf{V}_k \in \mathbb{R}^{|\mathcal{V}|\times e}$. In addition to the embedding parameters, HCGCF involves no other parameters to learn. Therefore, the whole trainable parameters are $\Theta = \{\mathbf{H}_u, \mathbf{H}_v\}$.

3.4.2 ***Time Complexity.*** We take applying the m-th Cayley-Dickson algebra $\mathcal{A}_m$ in HCGCF as an example to analyze the time complexity. Let E denotes the number of edges in the interaction graph, L denotes the total layers in the hypercomplex aggregation layer, e denotes the embedding size of the component, s denotes the epochs of the iterator, B denotes the batch size. The time complexity mainly comes from two parts: the aggregation layer and the prediction layer. (i) In the aggregation layer, the time complexity of linear aggregator $\text{AGG}_{\mathcal{S}}$ is $O(2^m ELes\frac{E}{B})$ and the time complexity of interaction aggregator $\text{AGG}_{\mathcal{I}}$ is $O(2^{2m} ELes\frac{E}{B})$. (ii) For the prediction layer, the time complexity is $O(2^{2m}Ees)$. In general, the time consumption of the whole training process is $O(2^{2m}Ees(1+\frac{LE}{B}+\frac{LE}{2^m B}))$. In fact, as shown in the experimental section (*cf.* Section 4.2), too large m will cause side effects and decrease the performance. Empirically, the performance is sufficiently good when $m = 3$ or 4. An important future work is to solve HCGCF's scalability by employing accelerating technologies, such as FastGCN [5].

Table 1: Statistical details of the three datasets.

dataset	# users	# items	# interactions	density
Book	12,859	22,294	598,420	0.2087%
Movie	12,953	10,785	699,469	0.5007%
Music	22,887	24,656	1,127,969	0.1998%

4 EXPERIMENTS

In this section, we first introduce the experiment setup, next compare our HCGCF with baselines and study the results, then explore the impact of HCGCF's hyper-parameters, and finally conduct visualization and case study experiments for detailed model analysis.

4.1 Experimental Setup

This subsection introduces the datasets, comparison methods, evaluation schema, and parameter settings.

4.1.1 ***Datasets.*** We conduct experiments with three datasets: Douban [32, 47] book[2] and movie[3], and KKBox[4], corresponding to book, movie, and music recommendation, respectively. Since the user-item interactions are originally in the rating format for Douban datasets, we convert the ratings to implicit feedback (denoted by "1") indicating the user has interacted with the item positively. Then, for each user, we randomly sample the same amount of negative samples (denoted by "0") as their positive ones from unrated items. Table 1 presents the statistics of the datasets.

4.1.2 ***Comparison Methods.*** To explore the high-dimensional hypercomplex algebras, we implement five instances in HCGCF based on five Cayley–Dickson algebras ($\mathcal{A}_m, m = 1, 2, 3, 4, 5$), including complex $\mathbb{C} \in \mathbb{H}_2$, quaternions $\mathbb{Q} \in \mathbb{H}_4$, octonions $\mathbb{O} \in \mathbb{H}_8$, sedenions $\mathbb{S} \in \mathbb{H}_{16}$, and trigintaduonions $\mathbb{T} \in \mathbb{H}_{32}$. To verify the effectiveness of HCGCF, we compare it with the following three group baselines, covering (i) the classical models (MF [22], DMF [45] and NeuCF [20]), (ii) the GCN-based models (GCMC [3], NGCF [44], and LightGCN [17]), and (iii) the hypercomplex-valued models (CCF [49], QCF [49], QFM [7], and QNFM [7]).

4.1.3 ***Evaluation Schema.*** We consider two application scenarios to evaluate the model performance: (i) in click-through rate (CTR) prediction, Area Under Curve (AUC) and Accuracy (ACC)

[2]Douban book: http://book.douban.com
[3]Douban movie: http://movie.douban.com
[4]KKBox: https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data

Table 2: Overall performance comparison on the three datasets. * denotes the best values among all methods.

Method	Book				Movie				Music			
	AUC	ACC	Pre	Rec	AUC	ACC	Pre	Rec	AUC	ACC	Pre	Rec
MF	0.7940	0.7205	0.0215	0.0526	0.9060	0.8323	0.0590	0.1170	0.8567	0.7828	0.0095	0.0188
DMF	0.8165	0.7456	0.0217	0.0563	0.9104	0.8375	0.0602	0.1195	0.8726	0.7987	0.0141	0.0280
NeuCF	0.8064	0.7429	0.0223	0.0584	0.9161	0.8409	0.0650	0.1220	0.8705	0.7954	0.0135	0.0273
GCMC	0.8207	0.7502	0.0247	0.0604	0.9103	0.8419	0.0643	0.1172	0.8643	0.7976	0.0124	0.0256
NGCF	0.8242	0.7513	0.0253	0.0633	0.9205	0.8474	0.0651	0.1211	0.8658	0.7994	0.0167	0.0328
LightGCN	0.8370	0.7648	0.0291	0.0706	0.9255	0.8511	0.0659	0.1238	0.8909	0.8193	0.0198	0.0390
CCF	0.8303	0.7630	0.0270	0.0669	0.9114	0.8389	0.0655	0.1247	0.8811	0.8115	0.0177	0.0367
QCF	0.8316	0.7645	0.0285	0.0691	0.9172	0.8426	0.0667	0.1294	0.8827	0.8213	0.0185	0.0385
QFM	0.8291	0.7587	0.0278	0.0685	0.9150	0.8402	0.0660	0.1278	0.8840	0.8182	0.0196	0.0419
QNFM	0.8342	0.7671	0.0282	0.0687	0.9266	0.8524	0.0675	0.1290	0.8889	0.8210	0.0204	0.0434
HCGCF($\mathbb{C}$)	0.8514	0.7784	0.0293	0.0717	0.9270	0.8564	0.0662	0.1300	0.8922	0.8211	0.0202	0.0439
HCGCF($\mathbb{Q}$)	0.8532	0.7797	0.0296	0.0724	0.9281	0.8610	0.0678	0.1312	0.8960	0.8255	0.0209	0.0451
HCGCF($\mathbb{O}$)	0.8539	0.7803	0.0301	0.0735	0.9297	0.8615	**0.0694***	**0.1346***	0.8964	0.8260	**0.0220***	**0.0472***
HCGCF($\mathbb{S}$)	**0.8552***	**0.7816***	**0.0307***	**0.0748***	0.9308	**0.8634***	0.0686	0.1323	**0.8977***	**0.8267***	0.0215	0.0459
HCGCF($\mathbb{T}$)	0.8516	0.7792	0.0290	0.0710	**0.9311***	0.8618	0.0683	0.1311	0.8955	0.8250	0.0213	0.0435

are adopted as the evaluation metrics; and (ii) in top-K recommendation, Precision@K (Pre) and Recall@K (Rec) are utilized as the evaluation metrics, where K is set as 20 by default.

4.1.4 ***Parameter Settings.*** We randomly split each dataset into training (60%), test (20%), and validation (20%) sets. All methods are optimized by the Adam optimizer with batch size of 1024. Learning rate η is tuned amongst $[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$ for all methods. We search the embedding size d (for all real-valued baselines) and component dimension size e (for all hypercomplex-valued baselines) in $[4, 8, 16, 32, 64]$. For GCN-based baselines, the depth of the GCN layer L is tested in $\{1, 2, 3, 4\}$. The layer sizes of the neural networks (e.g., the multi-layer perceptron in NeuCF and the quaternion neural network in QNFM) are tested in $\{1, 2, 3, 4\}$. For our method HCGCF, we set the depth of aggregation layer L as 2, and the trade-off factor ω as 0.5. Since the Cayley–Dickson representation has multiple components, to fix the model size for HCGCF, the component dimension size e is set as $\frac{64}{n}$ (n is the number of the components) in Section 4.2 for comparison.

4.2 Performance Comparison

Table 2 shows the performance of all models. From the results, we have the following main observations:

(i) NeuCF consistently outperforms MF. Besides, QNFM generally outperforms QFM. Such results are consistent with prior studies [7, 20], indicating the power of deep learning in recommendation.

(ii) MF, CCF, and QCF are shallow representation models, while CCF and QCF consistently outperform MF. These results indicate that utilizing hypercomplex algebras for learning user-item embeddings can enhance the recommendation performance.

(iii) CCF and HCGCF($\mathbb{C}$) are based on complex algebra, while HCGCF($\mathbb{C}$) consistently outperforms CCF; meanwhile, for the quaternion methods, HCGCF($\mathbb{Q}$) achieves better performance than QCF. These results verify the importance of modeling high-order user-item relationships in hypercomplex spaces.

(iv) Among all methods, HCGCF achieves strongly competitive performance in CTR prediction and top-K recommendation. For different HCGCF instances, we find that increasing hypercomplex algebra dimension n initially can enhance the performance (e.g., from HCGCF($\mathbb{C}$) to HCGCF($\mathbb{S}$) on the book dataset). These results show that high-dimensional hypercomplex algebras improve the recommendation performance within a certain dimension range. However, the performance drops when the hypercomplex algebra dimension n further increases (e.g., from HCGCF($\mathbb{S}$) to HCGCF($\mathbb{T}$) on the book dataset). One possible reason is that since we fix the same model size for all instances, there is a trade-off between algebra dimension n and component embedding size e. Too small e means weak representation learning ability of the component, which may reduce the performance. Empirically, with the same model size (i.e., $e = \frac{64}{n}$) for all instances, HCGCF($\mathbb{O}$) and HCGCF($\mathbb{S}$) generally achieve better performance on the three datasets.

4.3 Hyper-Parameter Sensitivity

We study the effect of three hyper-parameters in HCGCF: layer size L, component embedding size e, and trade-off factor ω. The results on the three datasets are shown in Table 3. From the results, we have the following observations:

(i) For the layer size L, we observe that when L increases from 0 to 1, the performance increases quickly for most HCGCF instances, verifying the importance of considering user-item relationships. Empirically, HCGCF achieves the best performance when $L = 1$ or $L = 2$. The results also show that high-dimensional hypercomplex algebra instances can achieve good performance without designing multi-layer structures compared with low-dimensional ones. For example, one layer is good enough for high-dimensional algebra instances HCGCF($\mathbb{S}$) and HCGCF($\mathbb{T}$).

(ii) For the component embedding size e, we observe that increasing e generally boosts the performance since a larger e can encode more information, while a too-large e increases the model size. Therefore, we need to find a proper size to balance the trade-off between the performance and the storage space.

(iii) For the trade-off factor ω, the performance increases when ω is tuned from 0.1 to the optimal value and then drops down, indicating linear aggregator $\text{AGG}_{\mathcal{S}}$ and interaction aggregator $\text{AGG}_{\mathcal{I}}$ complement each other. We also find the influences of ω are varying on different datasets, e.g., smaller ω performs better on the Douban

Table 3: AUC results of HCGCF with layer number L, component embedding size e, and trade-off factor ω.

Dataset	Book					Movie					Music				
L	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
HCGCF($\mathbb{C}$)	0.8309	0.8481	**0.8514**	0.8471	0.8376	0.9123	0.9235	**0.9270**	0.9199	0.9113	0.8815	0.8908	0.8922	**0.8969**	0.8893
HCGCF($\mathbb{Q}$)	0.8319	0.8507	**0.8532**	0.8460	0.8424	0.9168	0.9270	**0.9281**	0.9277	0.9247	0.8826	0.8945	0.8960	**0.8970**	0.8902
HCGCF($\mathbb{O}$)	0.8412	0.8519	**0.8539**	0.8457	0.8445	0.9264	**0.9308**	0.9297	0.9254	0.9246	0.8835	0.8962	**0.8964**	0.8935	0.8921
HCGCF($\mathbb{S}$)	0.8475	**0.8567**	0.8552	0.8463	0.8415	0.9275	**0.9312**	0.9308	0.9280	0.9262	0.8832	**0.8988**	0.8977	0.8928	0.8849
HCGCF($\mathbb{T}$)	0.8471	**0.8534**	0.8516	0.8466	0.8438	0.9201	**0.9327**	0.9311	0.9282	0.9237	0.8844	**0.8976**	0.8955	0.8923	0.8895
e	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
HCGCF($\mathbb{C}$)	0.8183	0.8227	0.8335	0.8478	**0.8514**	0.9147	0.9189	0.9218	0.9256	**0.9270**	0.8724	0.8778	0.8843	0.8881	**0.8922**
HCGCF($\mathbb{Q}$)	0.8360	0.8426	0.8483	0.8532	**0.8590**	0.9232	0.9247	0.9260	0.9281	**0.9303**	0.8779	0.8854	0.8936	0.8960	**0.9010**
HCGCF($\mathbb{O}$)	0.8391	0.8473	0.8539	0.8570	**0.8605**	0.9253	0.9280	0.9297	0.9309	**0.9327**	0.8827	0.8930	0.8964	0.9005	**0.9028**
HCGCF($\mathbb{S}$)	0.8475	0.8552	0.8614	**0.8639**	0.8637	0.9275	0.9308	0.9324	0.9331	**0.9335**	0.8910	0.8977	0.9012	**0.9040**	0.9033
HCGCF($\mathbb{T}$)	0.8516	0.8565	0.8666	**0.8676**	0.8658	0.9311	0.9320	0.9333	**0.9345**	0.9332	0.8955	0.8983	0.9025	0.9034	**0.9038**
ω	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
HCGCF($\mathbb{C}$)	0.8498	0.8507	**0.8514**	0.8509	0.8480	0.9264	**0.9283**	0.9270	0.9256	0.9220	0.8913	0.8918	0.8922	**0.8938**	0.8932
HCGCF($\mathbb{Q}$)	0.8500	0.8526	**0.8532**	0.8517	0.8504	0.9269	0.9278	**0.9281**	0.9262	0.9248	0.8925	0.8950	**0.8960**	0.8957	0.8939
HCGCF($\mathbb{O}$)	0.8519	**0.8542**	0.8539	0.8522	0.8511	0.9268	0.9294	**0.9297**	0.9291	0.9274	0.8944	0.8962	0.8964	**0.8983**	0.8960
HCGCF($\mathbb{S}$)	0.8531	0.8548	**0.8552**	0.8535	0.8518	0.9275	**0.9322**	0.9308	0.9300	0.9277	0.8928	0.8943	0.8977	**0.8979**	0.8965
HCGCF($\mathbb{T}$)	0.8504	**0.8523**	0.8516	0.8514	0.8497	0.9291	**0.9329**	0.9311	0.9314	0.9281	0.8916	0.8924	0.8955	**0.8960**	0.8936

datasets. Thus, factor ω would differently affect the performance in practice, which should be set carefully for different scenarios.

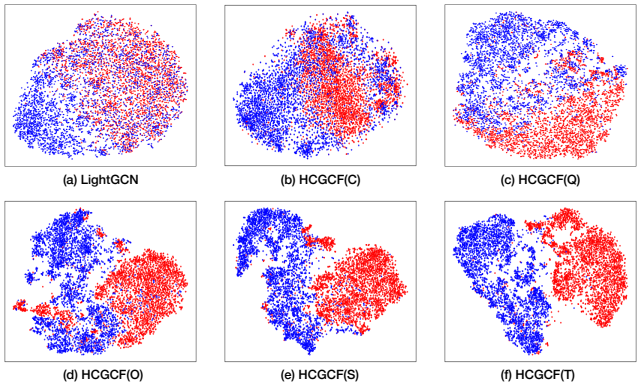


Figure 3: Visualization of node embeddings on the book dataset (red: users, blue: items).

4.4 Visualization

A nice embedding approach can provide a meaningful visualization of the network layout. To conduct a qualitative assessment of the learned embeddings, we utilize the t-SNE [40] to project the embeddings into a 2-dimensional space. For HCGCF, we concatenate all the components of the hypercomplex embeddings to have the real-valued vector inputs for the t-SNE. Figure 3 shows the visualization of the embeddings of the real-valued GCN method LightGCN and five HCGCF instances on the book dataset (we randomly select 3000 user-item pairs from the test set), where red nodes represent users and blue nodes represent items. From the figure, we find (i) Compared with LightGCN, HCGCF shows a better layout, which indicates that utilizing the hypercomplex representation is helpful to preserve the structure of the bipartite graph; (ii) from HCGCF($\mathbb{Q}$)), the separation between user and item embeddings is gradually clear, which indicates that hypercomplex multiplication in high-dimensional algebras ($m \geq 2$) provides an inductive bias in the aggregation layer for reflecting the asymmetrical user-item relations; and (iii) among different algebra instances, the user and item embeddings in high-dimensional instances (e.g., HCGCF($\mathbb{S}$) and HCGCF($\mathbb{T}$)) are better separated visually. The results show that high-dimensional hypercomplex representations are more expressive to learn the heterogeneous nature of the bipartite graph.

4.5 Case Study

In HCGCF, embeddings are sliced into multiple components and the final prediction are modeled by the hypercomplex multiplication. In this subsection, we explore how the model makes the prediction. Specifically, we show the value of each component in the prediction representation (i.e., $\mathbf{h}_u^* \otimes_{m+1} \mathbf{h}_v^*$ in Equation (21)). We take HCGCF($\mathbb{O}$) as an example to analyze our model.

We first select several user-item interaction pairs from the test set of the movie dataset to show the component values in Figure 4. We find that: (i) For different positive items of the same user, the component values are different (e.g., (u_{2391}, v_{511}) and (u_{2391}, v_{6392})). The reason may because that users have different perspectives on the items they like. Similar findings are also shown in the interaction of different users on the same item (e.g., (u_{2391}, v_{6392}) and (u_{5724}, v_{6392})); and (ii) generally, the component values are '+' for the positive items and are '-' for the negative items, which indicates each component is helpful to the final prediction.

To further provide a macro-level analysis, we present the boxplot figure of the prediction representation values in the test set in Figure 5. We observe that component values are different, indicating components have different contributions to the final prediction.

5 RELATED WORK

In this section, we review the areas which are relevant to our study.

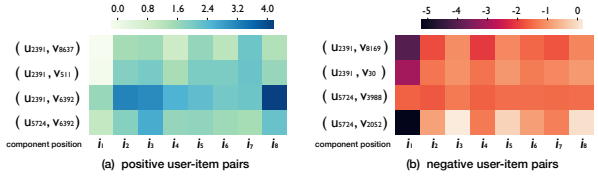


Figure 4: Case studies of component values in the prediction representation for HCGCF(𝕆) on the movie dataset.

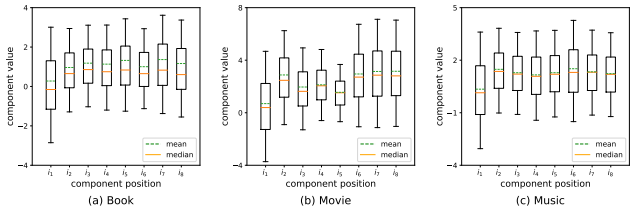


Figure 5: The component values in the prediction representation of HCGCF(𝕆) on the three datasets.

5.1 Graph-based Collaborative Filtering

From the perspective of the graph, the user-item relations can be regarded as a bipartite graph, where each user or item indicates a node and the interactions denote an edge between them [9]. In the collaborative filtering literature, considerable research efforts exploit the graph structure for modeling user-item interactions. Prior works, such as ItemRank [15] and BiRank [18], utilize propagate user preference on the bipartite graph and generate similarity scores for user-item pairs. HOP-Rec [46] is another well-known graph-based CF model, which first utilizes random walks on the graph to enrich user-item interactions and then adopts the MF framework based on the enriched interaction data to generate recommendations. Recently, graph convolution networks (GCNs) have become new state-of-the-art for graph representation [21, 41]. Inspired by the strong expressiveness of the graph convolution, extensive studies on GCN-based CF approaches have been conducted and achieved great success [3, 4, 6, 17, 33, 44]. For example, NGCF iteratively propagates user and item embeddings in the graph to model the user-item high-order connectivities. LightGCN [17] further enhances the performance of NGCF by simplifying the design of NGCF's graph convolution operations. However, all the above-mentioned graph-based CF models primarily focus on real-valued representations, which neglect the great potential of hypercomplex spaces. To bridge the gap, in this work, we explore the use of hypercomplex algebras and GCN techniques for CF tasks. Our HCGCF applies the hypercomplex graph convolution operator on the user-item bipartite graph, which distills high-order collaborative signals to refine the user and item hypercomplex embeddings.

5.2 Hypercomplex Representation Learning

Hypercomplex algebras play an important role in mathematics and physics [23]. Lately, hypercomplex representation learnings have demonstrated great promise in the field of machine learning. In graph representation learning, ComplEx [39] and RotatE [34] introduce complex-valued representations for the knowledge graph embedding task. QuatE [48] further enhances the performance of ComplEx and RotatE by using quaternion algebra. QGNN designs a graph neural network in quaternion space for graph classification and node classification tasks [28]. In computer vision, quaternion convolutional neural networks (QCNNs) obtain better performance than traditional real-valued models on image classification [30, 50]. Hypercomplex-valued extreme learning machines (HELMs) make great progress in color image auto-encoding [42, 43]. Hypercomplex representations are also useful for enhancing the performance of the transformer model for natural language processing [36].

Some recent studies use hypercomplex representation learning for recommendation [7, 12, 27, 38, 49]. For instance, CCF and QCF learn user and item representations in complex and quaternion spaces respectively, and both of them utilize the MF framework for CF tasks [49]. QFM extends factorization machines by designing a quaternion interaction pooling operation to model the feature interactions [7]. QNFM further enhances the performance of QFM by using the quaternion feed-forward network [7]. Inspired by QuatE, QKGN utilizes quaternion embeddings with the knowledge graphs to model users and items for recommendation [27]. QUALSE utilizes quaternion algebra in studying users' long-term and short-term preferences for sequential recommendation [38]. A recent work QGCN embeds users and items in quaternion space and devises a quaternion embedding propagation operation to refine representations [12]. Nevertheless, nearly all the above-mentioned hypercomplex recommenders are implemented with specific and low-dimensional hypercomplex algebras (e.g., complex and quaternion algebras). In this work, we use Cayley–Dickson construction in our HCGCF framework to explore and analyze high-dimensional hypercomplex spaces for better recommendation.

6 CONCLUSION

In this work, we propose a hypercomplex-valued GCN collaborative filtering framework HCGCF. We introduce Cayley–Dickson construction in HCGCF to explore high-dimensional hypercomplex spaces. Based on Cayley-Dickson construction, we devise a novel hypercomplex graph convolution operator to model the high-order user-item connectivity. We carry out extensive experiments on three real-world datasets. The results demonstrate our HCGCF outperforms several state-of-the-art recommendation models.

To the best of our knowledge, HCGCF is the first work to study the high-dimensional hypercomplex algebras for CF tasks by using GCN techniques. We believe the insights in this study and the promising results are enlightening for other future works to explore the hypercomplex space in solving recommendation problems.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their helpful comments. This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. 62172185 and 61876069; Jilin Province Key Scientific and Technological Research and Development Project under Grant Nos. 20180201067GX and 20180201044GX; and Jilin Province Natural Science Foundation under Grant No. 20200201036JC.

REFERENCES

[1] Daniel Alfsmann. 2006. On families of 2 N-dimensional hypercomplex algebras suitable for digital signal processing. In *EUSIPCO*. 1–4.

[2] John Baez. 2002. The octonions. *Bulletin of the american mathematical society* 39, 2 (2002), 145–205.

[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In *KDD*.

[4] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite graph embedding via mutual information maximization. In *WSDM*. 635–643.

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *ICLR*.

[6] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI*. 27–34.

[7] Tong Chen, Hongzhi Yin, Xiangliang Zhang, Zi Huang, Yang Wang, and Meng Wang. 2021. Quaternion Factorization Machines: A Lightweight Solution to Intricate Feature Interaction Modelling. *TNNLS* (2021).

[8] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation.. In *IJCAI*. 3329–3335.

[9] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *WWW*. 811–816.

[10] Craig Culbert. 2007. Cayley-Dickson algebras and loops. *Journal of Forensic Biomechanics* 1, 1 (2007), 1–17.

[11] Leonard E Dickson. 1919. On quaternions and their generalization and the history of the eight square theorem. *Annals of Mathematics* (1919), 155–171.

[12] Yaxing Fang, Pengpeng Zhao, Guanfeng Liu, Yanchi Liu, Victor S Sheng, Lei Zhao, and Xiaofang Zhou. 2021. Quaternion-Based Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2111.10536* (2021).

[13] Chase J Gaudet and Anthony S Maida. 2020. Generalizing Complex/Hypercomplex Convolutions to Vector Map Convolutions. *arXiv preprint arXiv:2009.04083* (2020).

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[15] Marco Gori, Augusto Pucci, V Roma, and I Siena. 2007. Itemrank: A random-walk based scoring algorithm for recommender engines.. In *IJCAI*. 2766–2771.

[16] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*.

[17] 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.

[18] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. 2016. Birank: Towards ranking on bipartite graphs. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 57–71.

[19] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[21] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

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

[23] Kantor I L, Kantor I L, and Solodovnikov A S. 1989. *Hypercomplex numbers: an elementary introduction to algebras*. Springer.

[24] Srdan Lazendic, Aleksandra Pizurica, and Hendrik De Bie. 2018. Hypercomplex algebras for dictionary learning. In *AGACSE*. 57–64.

[25] Anchen Li, Bo Yang, Hongxu Chen, and Guandong Xu. 2021. Hyperbolic Neural Collaborative Recommender. *arXiv preprint arXiv:2104.07414* (2021).

[26] Anchen Li, Bo Yang, Huan Huo, and Farookh Khadeer Hussain. 2021. Leveraging implicit relations for recommender systems. *Information Sciences* 579 (2021), 55–71.

[27] Zhaopeng Li, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. 2020. Quaternion-Based Knowledge Graph Network for Recommendation. In *MM*. 880–888.

[28] Tu Dinh Nguyen, Dinh Phung, et al. 2021. Quaternion graph neural networks. In *ACML*. PMLR, 236–251.

[29] Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. 2019. Quaternion recurrent neural networks. In *ICLR*.

[30] Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling human motion with quaternion-based neural networks. *IJCV* 128, 4 (2020), 855–872.

[31] Metod Saniga, Frédéric Holweck, and Petr Pracna. 2014. Cayley-Dickson algebras and finite geometry. *arXiv preprint arXiv:1405.6888* (2014).

[32] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.

[33] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor interaction aware graph convolution networks for recommendation. In *SIGIR*. 1289–1298.

[34] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

[35] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[36] Yi Tay, Aston Zhang, Luu Anh Tuan, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. 2019. Lightweight and efficient neural natural language processing with quaternion networks. In *ACL*. 1494–1503.

[37] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. 2018. Deep complex networks. In *ICLR*.

[38] Thanh Tran, Di You, and Kyumin Lee. 2020. Quaternion-based self-attentive long short-term user preference encoding for recommendation. In *CIKM*. 1455–1464.

[39] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*. 2071–2080.

[40] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[42] Guilherme Vieira and Marcos Eduardo Valle. 2020. Extreme Learning Machines on Cayley-Dickson Algebra Applied for Color Image Auto-Encoding. In *IJCNN*. 1–8.

[43] Guilherme Vieira and Marcos Eduardo Valle. 2021. A General Framework for Hypercomplex-valued Extreme Learning Machines. *arXiv preprint arXiv:2101.06166* (2021).

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

[45] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*. 3203–3209.

[46] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*. 140–144.

[47] Baolin Yi, Xiaoxuan Shen, Hai Liu, Zhaoli Zhang, Wei Zhang, Sannyuya Liu, and Naixue Xiong. 2019. Deep matrix factorization with implicit feedback embedding for recommendation system. *IEEE Transactions on Industrial Informatics* 15, 8 (2019), 4591–4601.

[48] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*. 2731–2741.

[49] Shuai Zhang, Lina Yao, Lucas Vinh Tran, Aston Zhang, and Yi Tay. 2019. Quaternion Collaborative Filtering for Recommendation. In *IJCAI*. 4313–4319.

[50] Xuanyu Zhu, Yi Xu, Hongteng Xu, and Changjian Chen. 2018. Quaternion convolutional neural networks. In *ECCV*. 631–647.