CDRec: Cayley–Dickson Recommender

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

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

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

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

ABSTRACT
In this paper, we propose a recommendation framework named Cayley–Dickson Recommender. We introduce Cayley–Dickson construction which uses a recursive process to define hypercomplex algebras and their mathematical operations. We also design a graph convolution operator to learn representations in the hypercomplex space. To the best of our knowledge, it is the first time that Cayley-Dickson construction and graph convolution techniques have been used in hypercomplex recommendation. Compared with the state-of-the-art recommendation methods, our method achieves superior performance on real-world datasets.

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

1 INTRODUCTION
Collaborative filtering (CF) is one of the dominant techniques used in recommender systems [15, 27]. 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 [3, 4, 11, 14, 38].

Recently, there is a surge in studying the recommendation models in hypercomplex spaces (non-real spaces) [6, 22, 32, 42]. In mathematics, hypercomplex algebra is defined as unital, distributive algebra, not necessarily associative or commutative [20]. 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. Based on these advantages, researchers propose diverse hypercomplex recommendation models which boost the performance over some real-valued recommenders [6, 42].

In this paper, we study the high-dimensional hypercomplex algebras for CF by utilizing GCN techniques. We propose a novel recommendation framework named CDRec, short for Cayley–Dickson Recommender. There are mainly two innovative points of CDRec. 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 [8, 20]. Second, based on Cayley-Dickson algebras, we devise a graph convolution operator to learn node representations by aggregating neighborhood information in the hypercomplex spaces.

To the best of our knowledge, this is the first work that utilizes Cayley–Dickson construction and GCN techniques for hypercomplex CF models. We compare CDRec with other 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 CDRec achieves highly competitive results and outperforms leading baselines in click-through rate prediction and top-K recommendation.

In summary, our main contributions are listed as follows:
• We propose CDRec, a hypercomplex GCN framework for recommender systems, which models user-item high-order relationships in hypercomplex spaces.
• 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 CDRec 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 $h \in \mathbb{H}_n$ expanded on the real vector space has a representation in the form, as follows:

$$h = x_1i_1 + x_2i_2 + \cdots + x_ni_n = \sum_{k=1}^{n} x_ki_k, \quad (1)$$

where $x_1, x_2, \ldots, x_n$ are real numbers called the components. The elements $i_1, i_2, \ldots, i_n$ are called hyperimaginary units, where $i_1$ represents the vector identity element $[1, 20].$

2.2 Cayley–Dickson Algebra
The Cayley-Dickson algebra $\mathbb{A}$ is a sequence of hypercomplex algebras obtained from the real numbers by the Cayley-Dickson construction [2, 8]. Higher-dimensional Cayley-Dickson algebras are obtained by doubling the previous smaller ones in the Cayley-Dickson construction [21]. Thus, the dimension of these algebras is

∗ Corresponding author.
We introduce our CDRec framework. The framework is based on Cayley–Dickson algebras. 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:

$$\mathcal{A}_{m+1} = \{h_a + h_b i_{2^m+1}\}, \quad h_a, h_b \in \mathcal{A}_m,$$  \hspace{1cm} (2)

where $m \in \mathbb{N}$ and $\mathcal{A}_0 = \mathbb{R}$. Here $i_{2^m+1} \notin \mathcal{A}_m$ is the additional hyperimaginary unit for doubling the dimension of $\mathcal{A}_m$, satisfying the following rules: $(i_{2^m+1})^2 = -1$, $i_1 i_{2^m+1} = i_{2^m+1} i_1$, and $i_0 i_{2^m+1} = -i_{2^m+1} i_0 = i_{2^m+2}$ for all $a, b = 2, 3, \ldots, 2^m$.

\section{Methodology}

In this section, we first formulate the problems. We then elaborate on our proposed CDRec model. After that, we will introduce the optimization process of CDRec. Finally, we discuss the CDRec’s relations with existing models.

\subsection{Problem Formulation}

We consider the standard implicit CF task set-up with a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$. We define $Y \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{V}|}$ as the user-item historical interaction matrix whose element $y_{uv} = 1$ if user $u \in \mathcal{U}$ is interested in item $v \in \mathcal{V}$ and zero otherwise. The goal of CDRec is to utilize the historical interaction to predict the preference probability from user $u$ to item $v$.

\subsection{The Proposed Model}

We introduce our CDRec framework. The framework is based on Cayley–Dickson algebras. There are three components in the framework: (i) an embedding layer that initializes user and item embeddings in the hypercomplex space; (ii) hypercomplex graph convolution layers that refine the hypercomplex embeddings by exploiting the high-order user-item relationships; and (iii) the prediction layer that outputs the preference score of a user-item pair based on the refined embeddings. Figure 1 illustrates the CDRec’s architecture and we describe each component in detail below.

\subsubsection{Embedding Layer}

The embedding layer takes a user and an item as inputs, and encodes them with Cayley–Dickson representations. Given a user-item pair $(u, v)$, user $u$’s embedding $h_u \in \mathcal{A}_{m+1}$ and item $v$’s embedding $h_v \in \mathcal{A}_{m+1}$ are formulated according the definition of Cayley–Dickson algebras, as follows:

$$h_u = a_u + b_u i_{2^m+1} = \sum_{k=1}^{2^m+1} u_k i_k, \quad \text{(3)}$$

$$h_v = c_v + d_v i_{2^m+1} = \sum_{k=1}^{2^m+1} v_k i_k, \quad \text{(4)}$$

where $m \in \mathbb{N}$; $a_u \in \mathcal{A}_m^c$ and $b_u \in \mathcal{A}_m^c$ are the subalgebras of $h_u$, and $c_v \in \mathcal{A}_m^e$ and $d_v \in \mathcal{A}_m^e$ are the subalgebras of $h_v$; $u_k \in \mathbb{R}^e$ and $v_k \in \mathbb{R}^e$ 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 supplement A.1.

\subsubsection{Aggregation Layer}

We design a hypercomplex graph convolution (HGC) layer that aggregates the neighborhood information to refine the hypercomplex embeddings of users and items. Since the user and item aggregation processes are symmetric, 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 [16]. In addition, since the feature interaction is important for recommendation [6, 42], we carefully design an interaction aggregator to capture neighbor feature interactions.

Specifically, given user $u$ and her first-order neighborhood $N_u$, the $(l+1)$-th HGC layer is defined as follows:

$$h_{u}^{(l+1)} = \mathcal{F}_c \left( \text{AGG}_S (h_v^{(l)} | v \in N_u), \text{AGG}_F (h_v^{(l)} | v \in N_u) \right), \quad \text{(5)}$$

where $h_u^{(l)}$ and $h_v^{(l)}$ are the user $u$’s and item $v$’s representations at the $l$-th layer respectively, and $h_u^{(0)} = h_u$ and $h_v^{(0)} = h_v$ are the initial representations from the embedding layer; AGG$_S$ and AGG$_F$ are the aggregation functions for the linear aggregator and the interaction aggregator, respectively; and function $\mathcal{F}_c$ is utilized to combine the representations from two aggregators.

Following the spirit of the GCNs [16], AGG$_S$ takes the summation of the neighbors’ hypercomplex representations to describe
the neighborhood information, as follows:

\[ h_{S(u)}^{(l+1)} = \text{AGG}_S(h_v^{(l)} | v \in N_u) = \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} h_v^{(l)} \]

\[ = \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} c_v^{(l)} + \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} d_v^{(l)} i_{2m+1} \]

\[ = \sum_{k=1}^{2m+1} \frac{1}{P_{uv}} v_k^{(l)} k_i. \]

where \( \sum \theta_m \) 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 \( \sqrt{|N_u||N_v|} \) (symmetric normalization).

Feature interactions, which have been demonstrated to be useful for many recommendation tasks \([10, 30, 38]\), are also very important signals to model the pairwise relational information between the center node and every neighbor. In HGC, we devise aggregator \( \text{AGG}_I \) to capture neighbor feature interactions by applying the hypercomplex multiplication, as follows:

\[ h_{I(u)}^{(l+1)} = \text{AGG}_I(h_v^{(l)} | v \in N_u) = \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} (h_v^{(l)} \otimes h_v^{(l)}) \]

\[ = \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} (a_v^{(l)} + b_v^{(l)} i_{2m+1}) \otimes (a_v^{(l)} + d_v^{(l)} i_{2m+1}) \]

\[ = \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} (a_v^{(l)} \otimes c_v^{(l)} \otimes d_v^{(l)} \otimes h_v^{(l)}) \]

\[ + \sum \frac{\theta_m}{\sum \theta_m} \frac{1}{P_{uv}} (a_v^{(l)} \otimes d_v^{(l)} \otimes h_v^{(l)} \otimes c_v^{(l)} \otimes i_{2m+1}, \]

where \( c_v^{(l)} = v_1^{(l)} i_1 - \sum_{k=2}^{2m} v_k^{(l)} i_k \) and \( d_v^{(l)} = v_1^{(l)} i_1 - \sum_{k=2}^{2m+1} v_k^{(l)} i_k \) are the conjugates of \( c_v^{(l)} \) and \( d_v^{(l)} \), respectively.

Different from the multiplication in the real number system, the hypercomplex multiplication explicitly captures latent interdependencies between all the components. Such multiplication rules encourage a more compact interaction of the neighborhoods. We provide a detailed analysis of \( \text{AGG}_I \) in supplement D to show the component interaction ability of \( \text{AGG}_I \).

To combine the linear aggregation and the interaction aggregation, we adopt a linear combination scheme, as follows:

\[ h_u^{(l+1)} = (1 - \omega) \cdot h_{S(u)}^{(l+1)} \otimes_{m+1} \omega \cdot (h_{I(u)}^{(l+1)}) \]

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 recommenders \([5, 11]\). Therefore, we discard the transformation matrix and activation function in each HGC layer.

### 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 \( h_u^* \), as follows:

\[ h_u^* = \frac{1}{L + 1} \sum_{l=0}^{L} h_u^{(l)} = \sum_{k=1}^{L} \frac{1}{L + 1} v_k h_u^{(l)} \]

Similarly, item \( v \)’s final representation \( h_v^* \) is formulated as:

\[ h_v^* = \frac{1}{L + 1} \sum_{l=0}^{L} h_v^{(l)} = \sum_{k=1}^{L} \frac{1}{L + 1} v_k h_v^{(l)} \]

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 \( h_u^* \) and the item aggregation representation \( h_v^* \) and then feed the interaction representation into a function \( \mathcal{K}_{\mathbb{H} \rightarrow \mathbb{R}} \):

\[ \hat{y}_{uv} = \mathcal{K}_{\mathbb{H} \rightarrow \mathbb{R}}(h_u^* \otimes h_v^*). \]

Here we implement the interaction between \( h_u^* \) and \( h_v^* \) as the multiplication. One can utilize various interaction functions, such as hypercomplex neural networks. Function \( \mathcal{K}_{\mathbb{H} \rightarrow \mathbb{R}} \) is utilized to transform a Cayley–Dickson representation into a scalar. Taking \( h_x \in \mathcal{A}_{m+1} \) as input, function \( \mathcal{K}_{\mathbb{H} \rightarrow \mathbb{R}} \) is defined as follows:

\[ \mathcal{K}_{\mathbb{H} \rightarrow \mathbb{R}}(h_x) = \frac{2^{m+1}}{2^{m+1}} \sum_{k=1}^{2^{m+1}} \sigma \left( \sum_{j=1}^{2} \sigma'(x_j) \right). \]

Following \([24, 42]\), we apply the split activation function (specifically, sigmoid: \( \sigma(x) = 1/(1 + e^{-x}) \)) on each component because the split approach can lead to better stability and simpler computation.

### 3.3 Model Optimization

#### 3.3.1 Loss Function

To learn the model parameters, we utilize the cross-entropy loss, which is widely used in CF tasks with implicit feedbacks \([7, 13, 14]\), as follows:

\[ \min_{\Theta} \mathcal{L} = \mathcal{L}_{\text{Rec}} + \lambda ||\Theta||_2^2 \]

\[ = - \sum_{(u,v,\hat{v}) \in D} (y_{uv} \log(\hat{y}_{uv}) + (1 - y_{uv}) \log(1 - \hat{y}_{uv})) \]

\[ + \lambda \left( \sum_{k=1}^{2^{m+1}} ||U_k||_2^2 + \sum_{k=1}^{2^{m+1}} ||V_k||_2^2 \right), \]

where \( \mathcal{L}_{\text{Rec}} \) measures the loss for recommendation. \( D \) is the set of training triplets, which is defined as:

\[ D = \{ (u,v,\hat{v}) | u \in \mathcal{U} \land v \in \mathcal{N}_u \land \hat{v} \in \mathcal{V} \setminus \mathcal{N}_u \}. \]

The second term in Equation (13) is the L2-regularizer, where \( \Theta = \{ (U_k), (V_k) \} \) is the parameter set and \( \lambda \) is the balancing factor controlling the strength of the regularization. Details of the training procedure pseudocode and complexity analysis can be found in supplements A.2 and B due to the space limitation.

### 3.4 Discussions

In the subsection, we discuss the relationships between CDRec and the related models.
### 4.2 Performance Comparison

Table 2 shows the performance of all compared methods in CTR prediction and top-K recommendation. From the results, we have the following main observations:

(i) DMF and NeuCF consistently outperform MF. Besides, QNFM outperforms QFM in most cases. Such results are consistent with prior studies [6, 14], indicating the power of deep learning techniques in recommender systems.

(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 CDRec(C) are based on complex algebra, while CRec(C) consistently outperforms CCF; meanwhile, for the quaternion methods, QGNN and CDRec(Q) generally achieve better performance than QCF. These results verify the importance of modeling high-order user-item relationships in hypercomplex spaces. We further study the effect of model depth L for CDRec in Section 4.3.

(iv) Among different CDRec instances, we find that increasing hypercomplex algebra dimension n initially can enhance the performance (e.g., from CDRec(C) to CDRec(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 CDRec(S) to CDRec(T) on the book dataset). One possible reason is that since we fix the same model size for all CDRec instances, there is a trade-off between hypercomplex 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 for all CDRec instances (i.e., e = 64), CDRec(C) and CDRec(S) generally achieve better performance.

(v) Among all methods, our model CDRec achieves strongly competitive performance in CTR prediction and top-K recommendation on the three datasets. In particular, take CDRec(S) as an example, its relative improvements over the best baseline w.r.t. Precision@20 are 5.10%, 3.30%, and 12.63% in book, movie, and music respectively, which demonstrates the high effectiveness of CDRec. We also conduct one-sample t-tests for two strong CDRec instances: CDRec(C) and CDRec(S). The results indicate that the improvements of our methods are statistically significant.
Table 2: Comparisons of methods on the three datasets. Best baselines are underlined. * denotes the statistical significance for \( p < 0.05 \) (CDRec(\( C \)) vs the best baseline, CDRec(\( S \)) vs the best baseline).

| Method      | AUC    | ACC    | Pre  | Rec  | AUC    | ACC    | Pre  | Rec  |
|-------------|--------|--------|------|------|--------|--------|------|------|
| MF          | 0.7940 | 0.7205 | 0.0215 | 0.0526 | 0.9055 | 0.8321 | 0.0583 | 0.1130 |
| DMF         | 0.8165 | 0.7456 | 0.0217 | 0.0563 | 0.9097 | 0.8361 | 0.0585 | 0.1162 |
| NeuCF       | 0.8064 | 0.7429 | 0.0223 | 0.0584 | 0.9161 | 0.8409 | 0.0650 | 0.1220 |
| GMCN        | 0.8207 | 0.7502 | 0.0247 | 0.0604 | 0.9103 | 0.8419 | 0.0643 | 0.1172 |
| NGCF        | 0.8242 | 0.7513 | 0.0253 | 0.0633 | 0.9205 | 0.8474 | 0.0651 | 0.1211 |
| LightGCN    | 0.8370 | 0.7648 | 0.0291 | 0.0706 | 0.9255 | 0.8511 | 0.0659 | 0.1238 |
| BiGI        | 0.8333 | 0.7615 | 0.0276 | 0.0649 | 0.9262 | 0.8556 | 0.0658 | 0.1149 |
| CCF         | 0.8297 | 0.7641 | 0.0259 | 0.0648 | 0.9114 | 0.8389 | 0.0655 | 0.1247 |
| QCF         | 0.8311 | 0.7643 | 0.0263 | 0.0672 | 0.9166 | 0.8421 | 0.0663 | 0.1321 |
| QFM         | 0.8284 | 0.7541 | 0.0265 | 0.0630 | 0.9147 | 0.8398 | 0.0666 | 0.1194 |
| QNFM        | 0.8339 | 0.7667 | 0.0257 | 0.0640 | 0.9260 | 0.8519 | 0.0663 | 0.1234 |
| CDRec(\( C \)) | 0.8514 | 0.7784 | 0.0295 | 0.0720 | 0.9270 | 0.8564 | 0.0664 | 0.1325 |
| CDRec(\( S \)) | 0.8532 | 0.7797 | 0.0297 | 0.0726 | 0.9281 | 0.8610 | 0.0677 | 0.1333 |
| CDRec(\( T \)) | 0.8530 | 0.7797 | 0.0297 | 0.0726 | 0.9264 | 0.9308 | 0.2927 | 0.9294 |
| CDRec(\( C \)) | 0.8507 | 0.8532 | 0.8460 | 0.8424 | 0.9166 | 0.9270 | 0.9281 | 0.9277 | 0.9247 |
| CDRec(\( S \)) | 0.8516 | 0.8554 | 0.8539 | 0.8539 | 0.9253 | 0.9280 | 0.9297 | 0.9309 | 0.9327 |
| CDRec(\( T \)) | 0.8517 | 0.8532 | 0.8539 | 0.8539 | 0.9273 | 0.9312 | 0.9308 | 0.9280 | 0.9262 |
| CDRec(\( C \)) | 0.8523 | 0.8566 | 0.8566 | 0.8566 | 0.9201 | 0.9327 | 0.9226 | 0.9237 | 0.9345 |

Table 3: AUC result of our methods with different layer numbers \( L \).

| Dataset | Book        | Movie      | Music       |
|---------|-------------|------------|-------------|
|        | 0 1 2 3 4  | 0 1 2 3 4  | 0 1 2 3 4  |
| CDRec(\( C \)) | 0.8297 0.8480 | 0.8514 0.8471 | 0.9114 0.9233 |
| CDRec(\( Q \)) | 0.8311 0.8507 | 0.8532 0.8460 | 0.9166 0.9270 |
| CDRec(\( T \)) | 0.8408 0.8519 | 0.8539 0.8457 | 0.9264 0.9308 |
| CDRec(\( C \)) | 0.8470 0.8567 | 0.8552 0.8463 | 0.9273 0.9312 |
| CDRec(\( S \)) | 0.8468 0.8554 | 0.8556 0.8466 | 0.9201 0.9327 |

Table 4: AUC result of our methods with different component dimension sizes \( e \).

| Dataset | Book        | Movie      | Music       |
|---------|-------------|------------|-------------|
|        | 2 4 8 16 32 | 2 4 8 16 32 | 2 4 8 16 32 |
| CDRec(\( C \)) | 0.8136 0.8222 | 0.8335 0.8478 | 0.9145 0.9189 |
| CDRec(\( Q \)) | 0.8360 0.8426 | 0.8483 0.8532 | 0.9232 0.9247 |
| CDRec(\( T \)) | 0.8391 0.8473 | 0.8539 0.8570 | 0.9253 0.9280 |
| CDRec(\( C \)) | 0.8470 0.8552 | 0.8614 0.8639 | 0.9275 0.9308 |
| CDRec(\( S \)) | 0.8516 0.8565 | 0.8676 0.8666 | 0.9311 0.9350 |

Table 5: AUC result of our methods with different trade-off factors \( \omega \).

| Dataset | Book        | Movie      | Music       |
|---------|-------------|------------|-------------|
|        | 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 |
| CDRec(\( C \)) | 0.8698 0.8507 | 0.8514 0.8509 | 0.9264 0.9283 |
| CDRec(\( Q \)) | 0.8500 0.8526 | 0.8532 0.8517 | 0.9269 0.9278 |
| CDRec(\( T \)) | 0.8519 0.8542 | 0.8539 0.8539 | 0.9268 0.9294 |
| CDRec(\( S \)) | 0.8530 0.8548 | 0.8552 0.8553 | 0.9277 0.9320 |
| CDRec(\( C \)) | 0.8540 0.8523 | 0.8516 0.8541 | 0.9291 0.9329 |

4.3 Hyper-Parameter Sensitivity

We explore the impact of three hyper-parameters in CDRec: layer size \( L \), component embedding size \( e \), and trade-off factor \( \omega \). The results on the three datasets are shown in Tables 3, 4, and 5, respectively. 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 all CDRec instances on the three datasets, verifying the importance of considering user-item relationships. Empirically, CDRec achieves the best performance when \( L = 1 \) or \( L = 2 \), respectively. 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 CDRec(\( S \)) and CDRec(\( T \)).

(ii) For the component embedding size \( e \), we observe that increasing \( e \) generally boosts the performance since a larger \( e \) can encode
more information of users and items, while a too-large \( e \) increases the model size. Therefore, we need to find a proper embedding size to balance the trade-off between the performance and the storage space. In addition, we observe that CDRec has strong tolerance for the component embedding size selection, which means selecting embedding size within a certain range has little effect on results.

(iii) For the trade-off factor \( \omega \), the performance increases when \( \omega \) is tuned from 0.1 to the optimal value and then drops down. The results indicate that linear aggregator AGG\(_{S}\) and interaction aggregator AGG\(_{F}\) complement each other. Separately analyzing the performance on the Douban (book and movie) and KKBox (music) datasets, we find that the influences of \( \omega \) are varying, e.g., smaller \( \omega \) performs better on the Douban datasets. Therefore, factor \( \omega \) would effect the performance differently in practice, which should be set carefully for different scenarios.

4.4 Visualization

A good embedding approach should provide a meaningful visualization of the network layout. To conduct a qualitative assessment of the learned embeddings, we use the t-SNE [34] to visualize the embeddings. For our CDRec, we concatenate all the components of the hypercomplex embeddings to have the real-valued vector inputs for the t-SNE. Figure 2 shows the visualization of the embeddings of the real-valued GCN method LightGCN and five CDRec instances on the book dataset (due to space limitation, 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 have the following findings: (i) Compared with LightGCN, CDRec shows a better layout for the user-item bipartite graph, which indicates that utilizing the hypercomplex representation is helpful to preserve the structure of the bipartite graph. (ii) Among different algebra instances in CDRec, the user and item embeddings of CDRec(S) and CDRec(T) are better separated visually. The results indicate that high-dimensional hypercomplex representation can better learn the heterogeneous nature of the user-item bipartite graph.

5 RELATED WORK

In this section, we provide a brief overview of two areas that are highly relevant to our work.

5.1 Graph-based Collaborative Filtering

From the perspective of graph, the user-item interactions can be regarded as a bipartite graph, where each user or item indicates a node and the interactions denote an edge between them. In the collaborative filtering literature, considerable research efforts exploit the graph structure for modeling user-item interactions. Prior works, such as ItemRank [9] and BiRank [12], borrow the idea of label propagation to propagate user preference on the interaction graph and derive similarity scores for user-item pairs. HOP-Rec [40] 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 [16, 35]. Motivated by the strength of the graph convolution, extensive studies on GCN-based recommenders have been conducted and achieved great success [3–5, 11, 28, 38]. For example, NGCF iteratively propagates user and item embeddings in the graph to model the user-item high-order connectivities. LightGCN [11] 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 CDRec applies the hypercomplex graph convolution operator on the user-item graph, which distills high-order collaborative signals to refine the user and item hypercomplex embeddings.

5.2 Hypercomplex Representation Learning

Hypercomplex algebras over the real field play an important role in modern mathematics and physics [20]. Lately, representation learnings in hypercomplex spaces have demonstrated great promise in the field of machine learning. In the knowledge graph embedding task, ComplEx [33] and RotatE [29] introduce complex-valued representations. QuatE [41] further enhances their performance by using the quaternion algebra. In computer vision, quaternion convolutional neural networks (QCNNs) obtain better performance than traditional real-valued models on image classification [25, 43]. Hypercomplex-valued extreme learning machines (HELMs) make great progress in color image auto-encoding [36, 37]. Hypercomplex representations are also useful for enhancing the performance of the transformer model for natural language processing [31].

Some recent studies use the hypercomplex representation learning for recommendation [6, 22, 32, 42]. For instance, CCF and QCF learn user and item representations in complex and quaternion spaces respectively, and both of them leverage the MF framework for CF tasks [42]. QFM and QNF extend factorization machines by quaternions [6]. Inspired by QuatE, QKGN utilizes quaternion embeddings with the knowledge graphs to model users and items for recommendation [22]. QUALSE utilizes the quaternion space in modeling both users’ long-term and short-term interests for sequential recommendation [32]. Nevertheless, these recommendation models are only designed for specific and low-dimensional hypercomplex spaces (e.g., complex and quaternion algebras) and do not explore high-dimensional algebras. In this paper, we propose
a hypercomplex GCN framework, which studies high-dimensional hypercomplex spaces by exploiting Cayley–Dickson algebras.

6 CONCLUSION AND FUTURE WORK

In this work, we propose a hypercomplex-valued GCN collaborative filtering framework CDRec. To explore high-dimensional hypercomplex spaces, we introduce Cayley–Dickson construction which provides an elegant algebraic formalism for defining hypercomplex algebras and their mathematical operations. Based on Cayley–Dickson construction, we devise a novel hypercomplex graph convolution layer to model the high-order connectivity in the user-item bipartite graph. Extensive experiments on three real-world datasets show that our CDRec outperforms many state-of-the-art baselines in CTR prediction and top-K recommendation.

To the best of our knowledge, CDRec is the first work to study the high-dimensional hypercomplex algebras for CF tasks by using GCN techniques. For future work, we will (i) integrate side information into CDRec such as social networks and knowledge graphs to further enhance the performance; (ii) try to generate recommendation explanations for comprehending the user behaviors and item attributes; and (iii) investigate other advanced hypercomplex systems (e.g., split type hypercomplex algebras [19] and dual type hypercomplex algebras [26]) on the recommendation tasks.

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, Xirun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite graph embedding via mutual information maximization. In WSDM. 635–643.
[5] 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.
[6] 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. arXiv preprint arXiv:2104.01716 (2021).
[7] Weyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation. In IJCAI. 3329–3335.
[8] Leonard E Dickson. 1919. On quaternions and their generalization and the history of the eight square theorem. Annals of Mathematics (1919), 155–171.
[9] Marco Gori, Augusto Pucci, V Roma, and I Siena. 2007. Itemrank: A random-walk based scoring algorithm for recommender engines. In IJCAI. 2766–2771.
[10] HuiFeng Guo, Ruiming Tang, Yuning Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In IJCAI.
[11] 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.
[12] 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.
[13] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguo Li, 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.
[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In WWW. 173–182.
[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In ICDM. 263–272.
[16] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In ICLR.
[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer (2009), 30–37.
[18] Wald Lischke and Steffen Rendle. 2020. On sampled metrics for item recommendation. In KDD. 1748–1757.
[19] Levent Kula and Yusuf Tayli. 2007. Split quaternions and rotations in semi Euclidean space E 4 2. Journal of the Korean Mathematical Society 44, 6 (2007). 1313–1327.
[20] Kantor I L, Kantor I I L, and Solodovnikov A S. 1989. Hypercomplex numbers: an elementary introduction to algebras. Springer.
[21] Srdan Lazendic, Aleksandra Pizurica, and Hendrik De Bie. 2018. Hypercomplex numbers for dictionary learning. In AGACSE. 57–64.
[22] Zhaopeng Li, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. 2020. Quaternion-Based Knowledge Graph Network for Recommendation. In MM. 880–888.
[23] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2020. Quaternion Graph Neural Networks. In NeurIPS.
[24] Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. 2019. Quaternion recurrent neural networks. In ICLR.
[25] Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling human motion with quaternion-based neural networks. IJCV 128, 4 (2020), 855–872.
[26] E Pennestrì and R Stefanelli. 2007. Linear algebra and numerical algorithms using dual numbers. Multidisciplinary System Dynamics 18, 3 (2007), 323–344.
[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In WWW. 285–295.
[28] 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. 1209–1209.
[29] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by rotational rotation in complex space. In ICLR.
[30] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In WSDM. 565–573.
[31] Yi Tay, Aston Zhang, Luan 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.

[32] Thanh Tran, Di You, and Kyumin Lee. 2020. Quaternion-based self-attentive long short-term user preference encoding for recommendation. In CIKM. 1494–1503.

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

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

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

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

[37] Guilherme Vieira and Marcos Eduardo Valle. 2021. A General Framework for Hypercomplex-valued Extreme Learning Machines. arXiv preprint arXiv:2103.06166 (2021).

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

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

[40] 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.

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

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

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