Course contrastive recommendation algorithm based on hypergraph convolution

Yanlie Zheng [1,*,†], Xueying Li [1,†] and Qingxia Shen [1,†]

[1] *Fiberhome Telecommunication Technologies Co.,LTD, Wuhan 430068, China*

Abstract

With its powerful modeling ability of real data, graph-based convolutional recommendation algorithms have become an important tool for providing personalized services to users. However, the existing graph-based recommendation algorithms mainly face the following problems: 1. Simple graph-based neural network models are difficult to this complex interaction relationship between user-items and their interaction high order.2. Personalized recommendation systems rely on users to generate their own data, which makes it difficult to obtain effective labeling information.3. Interaction noise. User-item interaction is saturated with noise interference. In order to overcome these problems and improve the performance of personalized recommendation, in this paper, we design a course comparison recommendation method based on enhanced hypergraph convolution (DHSL-Cu). Specifically, first, we design the introduction of a dual hypergraph convolutional network to capture the higher-order relationships between users and items and the potential interaction information under different interaction types. Second, we design a momentum-driven twin-based architecture to optimize the target network using a momentum-based parameter update strategy. Finally, a negative sample selection mechanism based on course learning is designed as our training strategy. Finally, through extensive experiments on real-world datasets and parameter analysis, we demonstrate that our proposed DHSL-Cu model can efficiently improve recommendation services.

Keywords

Hypergraph, Contrastive Learning, Recommendation, Course Learning

1. Introduction

In recent years, due to the powerful learning capability demonstrated by graph convolutional networks in the field of non-Euclidean data, many researchers have introduced them into the recommendation domain [1,2]. They consider users and items in the recommendation domain as nodes of an item, and the network relationships formed between nodes as the interactions between users and items, and learn the embedded

CITI'2024: 2nd International Workshop on Computer Information Technologies in Industry 4.0, June 12–14, 2024, Ternopil, Ukraine

* Corresponding author.

† These authors contributed equally.

zhengyanlie@ fiberhome.com (Y. Zheng); xli@ fiberhome.com (X. Li); qshen@ fiberhome.com (Q. Shen)

0000-0003-3412-1639 (Y. Zheng); 0009-0008-2733-3565 (X. Li); 0009-0000-1600-4717 (Q. Shen)

information of users and items embedded in the graph structure through convolution [3]. Yin et al. proposed to demonstrate the potential of the graph CNN approach in Pinterest's recommendation task [4]. GC-MC [5] and NGCF [5] have constructed bipartite graphs of user-item interactions from user-item interaction data in order to learn user preferences by utilizing the user-item graph structure. Compared to the above simple graph models in recommendation, HyperGCN [6] combines hypergraph with graph neural network, introduces the concept of hypergraph convolution [7], and propagates the information of nodes with the same interactions through hyperedge aggregation, which is able to efficiently capture the higher-order interaction information between nodes.

Although the above work has made some progress, it is still subject to the following 2 limitations: 1) the personalized recommendation system relies on users to generate their own data, which makes it difficult to obtain effective labeling information [8]. 2) It is well known that GCN-based recommendation models are susceptible to the noise of user-item interactions [9].

For this reason, in this paper, in order to alleviate the shortcomings in existing methods, we propose a course comparison recommendation method based on enhanced hypergraph convolution (DHSL-Cu). Specifically, firstly, in order to capture the complex interactions between users and programs, we introduce a dual hypergraph convolutional model to serve as an encoder to capture the higher-order relationships between user programs and the potential interaction information under different interaction types. Second, we design a momentum-driven twin architecture to optimize the target network using a momentum-based parameter update strategy. Next, we introduce the idea of course learning and design a negative sample selection mechanism based on course learning as our training strategy. Through extensive experiments on real-world datasets as well as parameter analysis, the effectiveness and superiority of the DHSL-Cu model are demonstrated.

2. Model

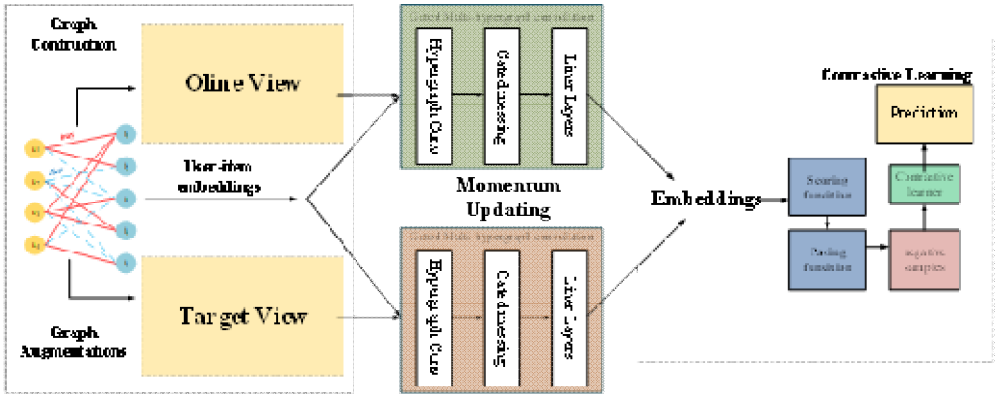

Figure 1: Overall model framework

Figure 1 illustrates our model for DHSL-Cu. As shown, our model consists of three main core modules:

Hypergraph Convolution Module: it is used to construct two-view and user-item hypergraph sets. In this module, we adopt graph augmentation to obtain new views of user-item interactions. After that, for two views, we construct user hypergraph set G_u and item hypergraph set G_{i} for online view and user hypergraph set $G_u^{'}$ and item hypergraph set $G_i^{'}$ for target view based on the interaction information of users and items. Then, similar to the literature [10], the characterization of users and items is carried out based on the convolution operation.

Twin network-based course comparison learning module:

1. Momentum-driven twin network based module: in this module both online network and target network use the gated dual hypergraph based convolutional model as the network structure. Meanwhile, we use the momentum update mechanism to optimize the target network's representation learning module during the parameter training process, which aims to utilize the characteristics of the momentum update mechanism to encode historical information in the target network, thus guiding the online network to learn to explore richer and more effective feature representations. At the same time, the asymmetry of the comparison network is enhanced by this parameter updating method thus alleviating the possible training collapse problem.
2. Negative sampling training strategy based on course learning, meanwhile, in order to solve the problem of too much randomness of negative sampling in contrast learning, we introduce the method of course learning. In the negative sampling process first through the Scoring function $S(\square)$, we sort the negative samples from easy to hard. In addition, a Pacing function $P(\square)$ is used to control the negative samples into the training process. Then, the mutual and consistency information between the two views is maximized, thus enhancing the user-item representation learning. Finally the user-item feature representation learned through the whole model is used as the final user-item representation for subsequent prediction tasks.

2.1. Formalization

Let a multipartite bipartite graph $\Psi = (U, I, E)$ have two different types of node sets (user set U and item set I) consisting of edge sets $E = \left\{E^1 \bigcup E^2 \cdots \bigcup E^K\right\}$, where E^j denotes the jth type of edge. For example, Amazon data user behavior log can be represented as, a multiple bipartite graph containing two kinds of nodes (users, items) and two kinds of edges (clicks, queries).

Input: Firstly, the original data is used to perform the corresponding data division and processing, according to the interaction behavior between users and items, for the embedding initialization of user-item nodes, to obtain the user-item initial feature matrix $X = \{X_U, X_I\}$, where $X_U \in \square^{N \times F}$, $X_I \in \square^{M \times F}$ denote the feature representations of the user set and the item set, respectively, and F is the feature dimensionality, N , M are the

number of users and the number of items. The user eigenvalues X_U and item eigenvalues X_I are used as inputs to the model in terms of hypergraph sets.

Output: By feeding the corresponding data and structures to our model, the model learns the final user-item representation for the next prediction recommendation task.

2.2. Hypergraph Building Blocks

We base our approach on two different enhancement methods to enhance the topological and attribute information of the graphs in order to obtain a new view of the user-item interaction. For both views we construct the user hypergraph set G_U for the online view and the user hypergraph set $G_U^{'}$ for the target view based on the user set U and the user hypergraph set G_I for the original view and the project hypergraph set $G_I^{'}$ for the target view based on the project set I . Among them, by combining the edge modification policy [11] is used to construct the online view, and we use the GD policy [12] for constructing the target view.

We transform both views resulting from the above graph enhancement operations into two isomorphic hypergraph sets. It is shown below:

$$E = mG_U = \left\{ g_{U,base}, g_{U,1} \cdots, g_{U,k} \right\}, G_I = \left\{ g_{I,base}, g_{I,1} \cdots, g_{I,k} \right\}, \tag{3}$$

where $g_{U,j} = \left\{ U, \mathcal{E}_{U,j} \right\}$ and $g_{I,j} = \left\{ I, \mathcal{E}_{I,j} \right\}$, where $\mathcal{E}_{U,j}$ and $\mathcal{E}_{I,j}$ denote the hyperedges in the hypergraphs $g_{U,j}$ and $g_{I,j}$, respectively. Note that all hypergraphs in G_U share the same set of user nodes U while all hypergraphs in G_I share the same set of project nodes I . For each project node $i \in I$, a hyperedge $\mathcal{E}_{U,j}$ is introduced in the hypergraph $g_{U,j}$ connecting $\left\{ u \mid u \in U, (u,i) \in E^j \right\}$, i.e., all user nodes in the set of user nodes U that are directly connected to the project node i through the interaction type E^j. Similarly, for a user node $u \in U$, a hyperedge $\mathcal{E}_{I,j}$ is introduced in $g_{I,j}$ that connects $\left\{ i \mid i \in I, (u,i) \in E^j \right\}$ i.e., all project nodes in the set of project nodes I that are directly connected to the user node u through the interaction type E^j. Note that two special hypergraphs $g_{U,base} \in G_U$ and $g_{I,base} \in G_I$ are defined as $g\left(U, \bigcup_{j=1}^{k} \mathcal{E}_{U,j} \right)$ and $g\left(V, \bigcup_{j=1}^{k} \mathcal{E}_{I,j} \right)$, i.e., the hypergraph is the one consisting of all interaction types between user items.

2.3. Dual Hypergraph Convolution Module

Dual Hypergraph Convolution Module mainly utilizes two chi-sub hypergraphs G_U and G_I in online view Γ or target view Γ' to obtain the potential features of users and items behind different interaction types by aggregating and propagating the feature

representations of users and items through the hyperedge aggregation and propagation of corresponding hypergraphs constructed on the basis of different interaction types in two hypergraphs through convolution operation. The specific process is as follows:

First, we introduce an association matrix H in the hypergraph to describe the relationship between nodes and hyperedges, given the hypergraph $g_{U,j} = \{U, \varepsilon_{U,j}\}$, where $j \in \{base, 1, \ldots, k\}$ denotes the different interaction types between user-items, and k is the number of interaction types between user-items, and defines the association matrix of $g_{U,j}$ as:

$$H_{U,j}(u,e) = \begin{cases} 1 & u \in e \bigcap e \in \varepsilon_{U,j} \\ 0 & otherwise \end{cases} \tag{4}$$

where $H_{U,j} \in \mathbb{R}^{|U| \times |\varepsilon_{U,j}|}$, $\varepsilon_{U,j}$ denotes the set of hyperedges in the hypergraph $g_{U,j}$, where $j \in \{base, 1, \ldots, k\}$, By the same definition, we define the cross association matrix for the hypergraph $g_{I,j}$. Let the diagonal matrices $D_{U,j}(u,u) \in \mathbb{R}^{|U| \times |U|}$ and $B_{U,j} \in \mathbb{R}^{|\varepsilon_{U,j}| \times |\varepsilon_{U,j}|}$ denote the node degree matrix and the hyperedge degree matrix, respectively. Where $D_{U,j}(u,u) = \sum_{e \in \varepsilon_{U,j}} H_{U,j}(u,e)$ and $B_{U,j}(e,e) = \sum_{u \in U} H_{U,j}(u,e)$. We utilize the hypergraph spectral convolution operator to learn the embedding of each hypergraph in the model, and the hypergraph convolution operator is denoted as:

$$X^{l+1} = \sigma(HWH^T \cdot X^l P^l) \tag{5}$$

Then, we use the normalized hypergraph convolution operator and define the hypergraph convolution operator for $g_{U,j}$ as:

$$X_{U,j}^{l+1} = \sigma(D_{U,j}^{-1} H_{U,j} W_U B_{U,j}^{-1} H_{U,j}^T \cdot X_{U,j}^l P_{U,j}^l) \tag{6}$$

where σ denotes the nonlinear activation function, $X_{U,j}^l \in \mathbb{R}^{|U| \times dF_l}$ denotes the user features in layer l, $W_U \in \mathbb{R}^{|I| \times |I|}$ is a unitary matrix, $P_{U,j}^l \in \mathbb{R}^{F_l \times F_{l+1}}$ is a learnable transformation matrix, and F_l and F_{l+1} denote the embedding dimensions in layers l and $l+1$. For the dual isomorphic hypergraph sets G_U and G_I, we independently learn the node features from each isomorphic hypergraph $g_{U,j}$ and $g_{I,j}$. As a result, we obtain node representations for users and projects based on different interaction types: $\{X_{U,base}, X_{U,1}, \ldots, X_{U,k}\}$ and $\{X_{I,base}, X_{I,1}, \ldots, X_{I,k}\}$.

Finally based on each layer of hypergraph convolution operator, after *t* rounds of iterations, we can get the embeddings of users and items under different types of interaction data, defined as $X_U^t = \{X_{U,base}^t, X_{U,1}^t, \ldots, X_{U,k}^t\}$, $X_{U,j}^t \in R^{|u| \times F_t}$,

$X_I^t = \left\{ X_{I,base}^t, X_{I,1}^t, \ldots, X_{I,k}^t \right\}$, $X_{I,j}^t \in R^{|I| \times F_t}$. where F_t is the final embedding dimension and k is the number of edge types. Finally, the feature values learned from multiple interaction types are concatenated together to obtain the final embedding result as follows:

$$\overline{X}_U = X_U^t \cdot W_U + b_U, \overline{X}_I = X_I^t \cdot W_I + b_I \tag{7}$$

where $W_U, W_I \in \square^{(k+1)*F_t \times F_t}$ and $b_U, b_I \in \square^{F_t}$ are trainable parameters. $\overline{X}_U \in \square^{|U| \times F_t}$ and $\overline{X}_I \in \square^{|I| \times F_t}$. This allows us to obtain the final embedding $Z = \left\{ \overline{X}_U, \overline{X}_I \right\}$ of users and items in the online graph.

2.4. Momentum-driven course comparison based module

2.4.1. Momentum-driven twinning-based network structure

In this module we construct a momentum-driven twin network-based architecture as our backbone, consisting of two identical network structures - the online network and the target network - which use the same encoder - the hypergraph convolutional module. Through the momentum optimization-based twin network feature, while the online network encodes the node features, the historical training information is retained in the target network, and then the mutual information between the two view representations is maximized in the course-based comparative learning module, which guides the online network to learn to explore richer and more effective feature representations.

The details are as follows: the target network uses the momentum update mechanism in the way of updating the parameters. The purpose is to make the target graph retain part of the historical information in the optimization process through the characteristics of the momentum update mechanism, and guide the online graph to learn richer and more effective feature representations in the comparison learning session. The iterative process of its model parameters is as follows:

$$\tilde{\mathcal{A}}^t = a \cdot \tilde{\mathcal{A}}^{t-1} + (1-a) \cdot \mathcal{A}^t \tag{8}$$

where a is an adjustable parameter controlling the degree of temporary information retention, and $\mathcal{A}^t$ and $\tilde{\mathcal{A}}^t$ denote the learnable parameters of the encoder at round t for the online network and target network, respectively. Thus, the embedding of users and items in the target graph is obtained $Z' = \left\{ \overline{X}'_U, \overline{X}'_I \right\}$.

2.4.2. Negative Sampling Module Based on Course Learning

Aiming at the problems arising from the negative sampling strategies of existing comparison learning papers, we design a novel negative sampling method based on course learning. The main idea is to sort the negative samples according to the difficulty during the training period, and introduce different negative samples into the training at different stages according to the difference in the difficulty of the negative samples.

Similar to the literature [39], firstly, for the embedding result of any node in the online graph, c negative embeddings $\{z_c'\}$ with different difficulties can be found in the target graph. meanwhile, a scoring function $S(\square)$ is defined to map the different negative embeddings into the target graph. that maps different negative embeddings to a numerical score $S(z_c')$ to measure this difficulty. Further, the scoring function is set to $sim(z_v, z_c')$ to fully measure the difficulty of negative samples. The formula is as follows:

$$S\left(z_c{}'\right) = \frac{|\, z_v \cdot z_c' \,|}{|\, z_v \,||\, z_c' \,|} \tag{9}$$

$$S(z_c') = z_v \cdot z_c' \tag{10}$$

$$S\left(z_c'\right) = sim\left(z_v, z_c'\right) = (z_v - z_c')^T \Sigma^{-1} (z_v - z_c') \tag{11}$$

where Σ is the covariance matrix of multidimensional random variables.

In line with the literature [12], after we obtain the fraction $S(z_c')$ of each individual negative embedding z_c' in the memory bank, we use the pacing function to schedule how to introduce negative samples into the training process. The pacing function $P(t)$ specifies the size of the memory bank to be used at each step t . The memory bank for t consists of the $P(t)$ lowest scoring samples. Negative sample batches are sampled uniformly from this set. We denote the complete memory bank size by C and the total number of training steps by T . The formula is as follows:

$$P(t) = \left[1 + .1 \log\left(\frac{t}{T} + e^{-10}\right)\right] \cdot K \tag{12}$$

$$P(t) = \left(\mathrm{t}/T\right)^{\phi} \cdot K \tag{13}$$

where ϕ is a smoothing parameter used to control the speed of the training process. $\phi = 1/2, 1, 2$ denote the root, linear and quadratic pacing functions, respectively.

2.4.3. Contrastive loss

To maximize the consistency between the online view and the target view. We use noise to estimate the contrast loss. Specifically, we define a "memory bank" Q , which contains each positive pair $\{z_v, z_v'\}$, i.e., the same node in both views, and C negative samples embedded in $\{z_c'\}_{c=1}^{C}$, the different nodes in both views, and then we use the similarity measure function $sim(\square,\square)$ to compute the positive pair $\{z_v, z_v'\}$ and negative pair $\{z_v, z_c'\}$, based on which the loss function is as follows:

$$l_{NCE} = -\log \frac{\exp\left(sim\left(z_v, z_v'\right)/\tau\right)}{\exp(sim(z_v, z_v')/\tau) + \sum_{c=1}^{C}\left(\exp(sim(z_v, z_c')/\tau)\right)} \tag{14}$$

where τ denotes the temperature parameter. To simplify the calculation, we use dot product as the similarity measure function.

2.5. Model Optimization

We first use the edges that already exist in the interaction data as positive edges and extract some non-existing edges as negative edges. Finally, the loss function is designed by maximizing the positive edge probability and minimizing the negative edge probability. The loss function is defined as:

$$l_{\mathrm{r}} = \sum_{(u,i)\in E}\left[\lambda \cdot \log\sigma(Z_u^T Z_i) + (1-\lambda)\cdot \sum_{j=1}^{n}\left(\mathrm{E}_{u_j \square P(u)} \log(1-\sigma(Z_u^T Z_{u_j})) + \mathrm{E}_{i_j \square P(i)} \log(1-\sigma(Z_i^T Z_{i_j}))\right)\right] \quad (15)$$

where σ is the sigmoid activation function, λ is the weight parameter in order to balance the importance of positive and negative samples, $P(u)$ defines the distribution of u candidate nodes, and n is the number of negative samples, the existing edges in the multipartite bipartite graph are used as positive samples, and for each positive sample of edge (u,i) , n negative edges are randomly sampled from node u and node i .

Ultimately, we unify the representation learning module for the main recommendation task and the self-supervised comparison learning module for auxiliary enhancement into an overall learning framework. Formally, the final learning objective is defined as:

$$l = l_r + \beta l_{NCE} \quad (16)$$

where β is a variable factor that controls the self-supervised contrast learning task. Finally we train our model using the Adam algorithm.

3. Experimental Comparison and Analysis

3.1. Datasets

To evaluate the performance of our model, we conducted experiments on two real-world datasets, Amazon [12] and Alibaba[12]. The relevant attribute statistics of the two datasets are shown in Table 1.

Table 1
Statistics of the data set

Datasets	User	Item	Interaction	Interaction types	Density
Amazon	3781	5749	60658	2	0.279%
Alibaba	1869	13349	27036	3	0.108%

3.2. Experimental Setup

Consistent with the literature [13], we use AUROC, AUPRC, Precision and Recall as evaluation metrics. Meanwhile, we randomly select 60% of the edges as the training set

and the rest of the edges as the test set. The whole process is repeated five times to obtain different random samples for the training and test sets. The mean and standard deviation values of the classification evaluation metrics are reported.

3.3. Experimental Design

In order to comprehensively test the performance of the DHSL-Cu model proposed in this paper, we test it from different perspectives as follows.

1. comparison with mainstream advanced algorithms: in order to assess the effectiveness of the DHSL-Cu model, this paper compares it with eight mainstream advanced algorithms.
2. ablation analysis: the contribution of each component is analyzed.
3. parameter sensitivity analysis: the influence degree of momentum-driven update weight ratio, learning rate, and self-supervised comparison learning factor is analyzed.

3.4. Baseline Algorithms

To evaluate the performance of the DHSL-Cu model, we compare it with the following existing mainstream state-of-the-art recommendation algorithms.

1. GraphSAGE [13]: proposes a generalized induction architecture that uses local neighbor sampling of nodes and aggregated features to generate embeddings of nodes.
2. GCN [13] uses spectral graph convolution operators to learn local graph network structures and node features in order to achieve semi-supervised learning directly on graph structure data.
3. GAT [13] uses masked self-attention to assign different weights to each node and its neighboring nodes based on their features, eliminating the need to use a pre-constructed graph.
4. HGNN [13] designed the hyperedge convolution operation to deal with data correlation during representation learning. By this method, the hyperedge convolution operation can be effectively utilized to capture the implicit layer representation of higher order data structures.
5. HyperGCN [13] is a new method of GCN training for semi-supervised learning of hypergraphs based on hypergraph theory.
6. DualHGCN [13] a self-supervised Dual Hypergraph Convolutional Network (DualHGCN) model that transforms a multilayer two-part graph network into two sets of its hypergraph sets.
7. SGL [13] designed three types of data augmentation based on different perspectives to complement/supervise the recommendation task with self-supervised signals on user-item graphs.
8. HCCF [13] designed hypergraph structure learning module and cross view hypergraph contrast coding model based on contrast learning to learn better user representations by characterizing both local and global collaborative relationships in joint embedding space.

3.5. Analysis of experimental results

3.5.1. Comparison with baseline algorithm

Table 2
Comparison of overall performance in auroc, auprc, precision and recall

Methods	Amazon				Alibaba			
	auroc	auprc	precision	recall	auroc	auprc	precision	recall
GraphSAGE	66.99	69.39	63.47	52.74	66.49	60.36	63.47	52.74
GCN	64.93	77.45	69.46	71.53	56.87	77.66	69.46	71.53
GAT	66.7	70.16	63.34	51.39	55.38	54.49	63.34	51.39
HGNN	80.14	82.94	78.54	69.51	69.64	73.5	78.54	69.51
HyperGCN	68.42	73.78	67.12	61.61	61.38	65.21	67.12	61.61
DualHGCN	83.46	88.69	85.63	76.39	84.57	86.02	85.63	76.39
HCCF	94.27	95.32	91.56	91.67	90.13	89.32	90.27	82.13
DHSL-Cu	**96.68**	**97.89**	**93.7**	**94.68**	**93.57**	**91.76**	**91.76**	**84.94**

The results of all algorithm experiments on both datasets are shown in Table 2. From the experimental results of the four evaluation metrics AUROC, AUPRC, Precision and Recall, we can conclude that:

1. GraphSAGE, GCN and GAT perform poorly on the two datasets, which may be due to the fact that these embedding methods based on simple homogeneous graphs have a weak interaction representation and do not deal well with non-planar relationships between nodes compared to hypergraph convolution.
2. DualHGCN outperforms HGNN and HyperGCN, probably because HyperGCN and HGNN are both based on the same type of interaction information, and when confronted with complex user-item interactions under different types, it is difficult to capture the potential higher-order information of the user and the item based on different types of interactions. While DualHGCN effectively captures the interaction information between different interaction types by designing the information transfer mechanism between hypergraphs, the overall performance of DualHGCN is weaker than that of HCCF, probably because HCCF is designed with a hypergraph-based comparison learning model, which efficiently utilizes the local-to-global cross-view supervision information.

In conclusion, DHSL-Cu outperforms the other benchmarks. This may be related to the following two main reasons: 1) DHSL-Cu can effectively incorporate historical training information by designing a momentum-driven target network structure, and the parameter updating method of momentum update with gradient vanishing can effectively alleviate the training collapse problem during the self-supervised learning process. 2) During the training process, we use negative sampling based on course learning, which is different from the general random sampling method, and it can effectively utilize local-to-global cross-view supervision information. The negative sampling method based on

course learning is different from the general random sampling method, which can effectively introduce different training phases according to different sample characteristics, and effectively improve the generalization ability and prediction accuracy of the model.

3.5.2. Ablation Analysis

In order to verify the different effects on the algorithm brought by the gating-based dual hypergraph convolution module, the momentum-driven twin network structure, and the negative sampling strategy based on course learning, we conducted experiments and comparative analysis. The experimental results are shown in Table. 3, where DHSL-Cu-Init denotes the feature initialization module only, DHSL-Cu-H introduces the gating-based dual hypergraph convolution module after the feature priming module, and DHSL-Cu-M introduces the momentum-driven twin-network-based module and comparative learning on top of DHSL-Cu-H. DHSL-Cu-Both, i.e., the introduction of negative sampling strategy after the introduction of the complete DHSL-Cu model.

Table 3
Comparative analysis of algorithmic impact

	auroc		auprc		precision		recall	
	amazon	alibaba	amazon	alibaba	amazon	alibaba	amazon	alibaba
Init	59.50	72.37	64.77	59.00	63.48	48.73	57.36%	62.75
H	83.46	87.59	89.95	82.20	85.63	69.81	82.30	79.92
M	92.99	89.33	94.54	86.47	90.69	86.47	91.07	81.31
Both	96.68	93.59	97.89	91.76	93.70	91.76	94.98	84.94

As shown in Table. 3, DHSL-Cu-Init performs the worst. DHSL-Cu-H better and accurately shows the effectiveness and efficiency of the gating-based dual hypergraph mechanism than it. The improvement in the performance of DHSL-Cu-M and DHSL-Cu-Both illustrates the power of the self-supervised contrast learning framework. Among them, the best performance of DHSL-Cu-Both shows that the introduction of negative sampling strategy can effectively improve and enhance the learning performance of contrast learning, and DHSL-Cu and its variants outperform the Alibaba dataset on the Amazon dataset. The possible reason could be that the Amazon dataset is denser, which helps to capture more effective data during various types of user-item interactions.

3.5.3. Parameter sensitivity analysis

This section focuses on the effect of the factor β setting for self-supervised learning on the model performance.

As shown in Tables. 4, 5, the model performance at contrast learning factor less than 0.5 is improves with increasing contrast learning factor and reaches the optimal point for all four evaluation metrics on the $beta = 0.5$ two datasets, after which the performance decreases with increasing contrast learning factor. This may be a result of learning loss

pairs that are too high in the gradient conflict between the prediction and comparison tasks during training.

Table 4
Results of different contrasting learning factors on AUROC and AUPRC metrics

beta	AUROC		AUPRC	
	Amazon	Alibaba	Amazon	Alibaba
0.1	96.39	92.29	97.74	90.91
0.2	96.39	92.67	97.70	90.94
0.3	96.44	92.56	97.76	90.56
0.4	96.53	93.36	97.79	90.53
0.5	96.64	93.57	97.86	91.44
0.6	96.50	93.24	97.77	90.78
0.7	96.51	93.09	97.74	90.40
0.8	96.43	93.07	97.77	90.89
0.9	96.49	93.15	97.74	90.53

Table 5
Results of different comparison learning factors on Precision and Recall metrics

beta	Precision		Recall	
	Amazon	Alibaba	Amazon	Alibaba
0.1	93.12	88.85	94.25	82.03
0.2	93.27	89.24	94.35	81.86
0.3	92.79	88.83	94.43	82.73
0.4	93.39	89.15	94.44	83.48
0.5	93.53	89.47	94.48	84.93
0.6	93.34	88.48	94.44	83.45
0.7	93.33	88.31	94.21	82.14
0.8	93.17	87.59	94.31	82.23
0.9	93.23	88.91	94.21	81.57

4. Conclusion

We propose a new self-supervised learning method DHSL-Cu. Specifically, we first generate two hypergraph views based on a two-part graph network of users and items after 2 different graph enhancement strategies. For the target network in the twin network we use a momentum-based parameter update mechanism. The slow moving target network is made to encode the online network history observations. A course comparison framework based on a negative sample selection mechanism for course learning is also designed. Finally, the effectiveness and superiority of the proposed DHSL-Cu model is well demonstrated through extensive experiments on real-world datasets as well as parameter analysis.

References

[1] Xie F, Zhang Y, Przystupa K, Kochan O. A Knowledge Graph Embedding Based Service Recommendation Method for Service-Based System Development [J]. Electronics, 2023, 12(13): 2935.

[2] Xu X, Przystupa K, Kochan O. Social Recommendation Algorithm Based on Self-Supervised Hypergraph Attention [J]. Electronics, 2023, 12(4): 906.

[3] Saito Y, Yaginuma S, Nishino Y, et al. Unbiased recommender learning from missing-not-at-random implicit feedback[C]//Proceedings of the 13th International Conference on Web Search and Data Mining. 2020: 501-509.

[4] Zhu D, Zhang Z, Cui P, et al. Robust graph convolutional networks against adversarial attacks[C]//Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 1399-1407.

[5] Wang X, He X, Wang M, et al. Neural graph collaborative filtering[C]//Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 2019: 165-174.

[6] Yadati N, Nimishakavi M, Yadav P, et al. Hypergcn: A new method for training graph convolutional networks on hypergraphs[C]//Advances in neural information processing systems, 2019, 32.

[7] Jiang K, Zhang C, Wei B, Li Z, Kochan O. Fault diagnosis of RV reducer based on denoising time–frequency attention neural network [J]. Expert Systems with Applications, 2024, 238: 121762.

[8] Yanqiao Zhu, Yichen Xu, Feng Yu, et al., Graph Contrastive Learning with Adaptive Augmentation[C]// Proceedings of the Web Conference 2021, pp.2069–2080.

[9] Feng Y, You H, Zhang Z, et al. Hypergraph neural networks[C]//Proceedings of the AAAI conference on artificial intelligence, 2019, 33(01): 3558-3565.

[10] Bian Shuqing, Zhao Xin Wayne, Zhou Kun, et al., Contrastive Curriculum Learning for Sequential User Behavior Modeling via Data Augmentation[C]// Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp.3737–3746.

[11] Qin Xiuyuan, Yuan Huanhuan, Zhao Pengpeng, et al., Meta-optimized Contrastive Learning for Sequential Recommendation[C]// Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp 89–98.

[12] Xue H, Yang L, Rajan V, et al. Multiplex bipartite network embedding using dual hypergraph convolutional networks[C]//Proceedings of the Web Conference 2021. 2021: 1649-1660.

[13] Wang Shoujin, Hu Liang, Wang Yan, et al., Graph Learning based Recommender Systems: A Review[C]// Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 2021,pp. 4644-4652.