Hypergraph Clustering Network for Interaction Data

Tianchi Yang[†] Luhao Zhang[‡] Cheng Yang[†]
Chuan Shi[†] Maodi Hu[†] Tao Li[†] Dong Wang[†]
[†] Beijing University of Posts and Telecommunications, Beijing, China
{yangtianchi,shichuan}@bupt.edu.cn, albertyang33@gmail.com
[‡] Meituan, Beijing, China
{zhangluhao,litao19,wangdong07}@meituan.com, Londeehu@gmail.com

Abstract

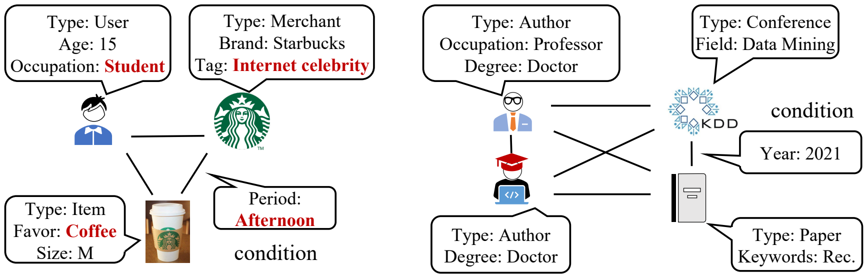
With the rapid development of social media, the data generated from interaction, which is an action collaboratively done by several objects under a certain condition, have grown exponentially. It triggers an urgent need to cluster interaction data for implicit patterns benefiting downstream tasks. Although clustering methods have been extensively studied for a long time, they mainly focus on node/object clustering, which breaks the intrinsic collaborative associations among the involved objects and conditions, thus limiting clustering performance of interaction data. To tackle this issue, we propose a novel Hypergraph CLustering network for Interaction Data (HyCLID), which not only clusters whole interactions rather than individual objects in interactions, but also exploits correlation among attributes of objects and conditions. Specifically, we first construct an attributed hypergraph to model interactions. Then, we propose a novel rethinking-based hypergraph neural network to learn the representation of interactions, which employs a novel attentive routing-based rethinking mechanism to capture the correlations among multiple object attributes and condition attributes. Furthermore, a novel adaptive mini-batch method is designed for the large scale of interaction data. Experimental results demonstrate the effectiveness of our methods for clustering the interactions and the practical value of the discovered interaction patterns.

1 Introduction

With the rapid development of social media, the volume of interaction data[1] has grown exponentially to an overwhelming scale [13, 30]. Generally, an interaction is an action collaboratively done by several objects under a certain condition [30]. For example, as illustrated in Figure 1, “a user ordered a cup of latte at Starbucks in the afternoon” in food delivery systems, and “two researchers collaborated to publish a paper in WSDM in 2022” in academic networks, etc. In detail, Interaction 1 involves a condition about period and three objects including a user, a merchant and a product, each of which carries several attributes, which indicates the characteristic of interaction data is that the interaction involves multiple objects and the rich correlations among the attributes of objects and conditions. As interaction data play an increasingly important role in daily life, analyzing them becomes a priority.

Copyright 2022 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] The term interaction data is similar to the term relational data, which focuses on the data that there are pair-wise relations *between* objects, while we introduce the term interaction data in this paper for distinction to focus on the relations among multiple objects.



Interaction 1: a user ordered a cup of latte at Starbucks in the afternoon

Interaction 2: two researchers collaborated to publish a paper in KDD in 2021

Figure 1: Examples of interactions from Meituan Waimai[2] and academic networks.

As a fundamental task in data mining, clustering on interaction data can discover the underlying patterns beneficial to downstream tasks. For instance, in the above example, the user preference can be found by clustering his interactions, enabling the platform to provide better recommendation services. Traditional clustering focuses on object features, such as K-means [6, 32], etc. Besides, graph-based clustering, which treats objects as nodes in a graph and leverages structural information of objects for clustering the objects, has also been widely explored, e.g., spectral clustering [16]. Recently, some researchers pay attention to attributed graph clustering, which jointly considers node attributes and graph structure [28]. A noticeable trend is the wide adoption of deep learning in clustering [15]. For example, Graph Neural Network (GNN) based clustering methods [1, 21, 18, 29] can effectively learn object representations for clustering via message passing mechanism. However, we argue that existing methods could not fully cope with the interaction data in the real world. Specifically, in interaction data, multiple objects as well as interaction conditions collaboratively form an indivisible semantic unit. Nonetheless, these methods focus on clustering individual nodes/objects, which breaks the intrinsic collaborative associations among the involved objects and conditions, thus limiting their ability to discover the underlying interaction patterns. Take Interaction 1 in Figure 1 as an example, if we ignore the associations among user, merchant and product, one may discover a one-sided pattern that “this user loves coffee”. This pattern will possibly lead to an inappropriate recommendation of high-end coffee to this student far beyond his consumption level.

Therefore, we propose to cluster interactions instead of objects, where each interaction is regarded as a basic unit to be clustered. Nevertheless, this is not a trivial task due to the following reasons: (1) *How to effectively model interaction data involving multiple objects?* As mentioned before, an interaction is an indivisible semantic unit. Nevertheless, a majority of existing solutions employ graph structure to model the relations between objects, which will break the collaborative association among multiple objects into several pairwise sub-relations. Hence they are insufficient to model interaction data in the real world. (2) *How to capture the correlations among the attributes of objects and conditions?* The occurrence of an interaction is mainly owing to the complex correlation among rich attributes of its involved objects and conditions. For instance, Interaction 1 in Figure 1 is mainly derived by the correlation among “student, Internet celebrity, coffee, afternoon” (marked in red). Since this correlation is among multiple attributes, capturing them is a combinatorial optimization problem. To avoid combinatorial explosion, some recent methods [19, 24] adopt self-attentive layers to capture the pairwise correlations between two features, called second-order features. Then, the higher-order features can be approximated by stacking multiple self-attentive layers with residual connections. However, the errors in the bottom layers will inevitably propagate to the following layers. Even worse, such error propagation will be enlarged as the number of layers increases, thus possibly leading to the omissions of some potential high-order features. (3) *How to handle the large scale of interaction data during clustering?* The scale of interaction data in real applications is always large, thus requiring clustering in a batch fashion. Nonetheless, the data distributions

[2] Meituan Waimai is a food delivery platform. https://waimai.meituan.com

of different batches are usually not the same, which probably causes data imbalance so that the clustering error may largely fluctuate or even increase [26]. Unfortunately, existing methods [23, 21] cannot automatically make adaptive adjustments for different batches, thereby leading to sub-optimal clustering performance.

To tackle the aforementioned issues, we propose a novel Hypergraph CLustering network for Interaction Data (HyCLID). Specifically, to model interactions that involve multiple objects and conditions, we construct an attributed hypergraph for interactions. As the hyperedges can connect an arbitrary number of objects, we model each interaction as a hyperedge connecting several nodes that represent its involved objects. Then, we propose a novel rethinking-based hypergraph neural network to capture the complex correlations (high-order features) among object attributes and conditions for embedding the interactions (hyperedges). Besides, to avoid omissions of potential high-order features, we design a novel attentive routing-based rethinking mechanism to review the capturing process and correct the errors in the bottom layers. Finally, we propose an adaptive mini-batch clustering method based on the learned interaction representations to perform deep clustering on large-scale interaction data, which employs an adaptive batch standardization to remove the impact of different data distributions of different batches. The main contributions are summarized as follows:

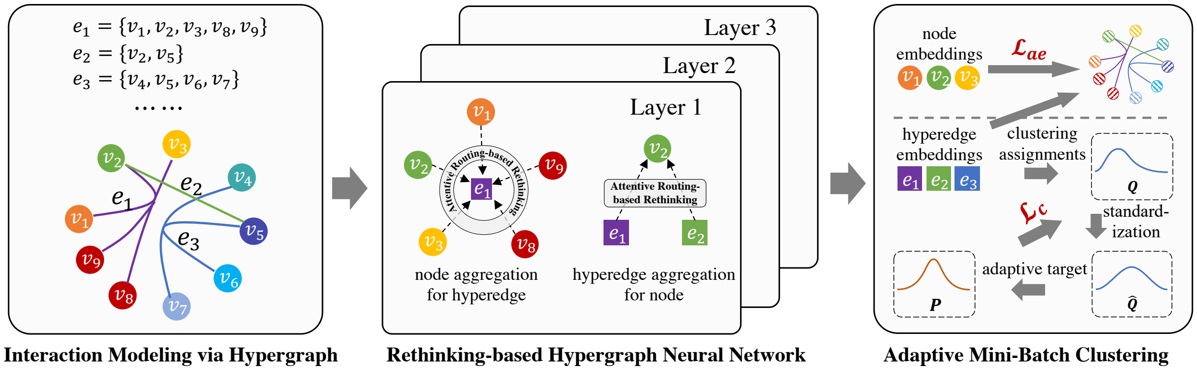
- We propose a hypergraph clustering network for interaction data, namely HyCLID. With interactions modeled as an attributed hypergraph, HyCLID designs a novel rethinking-based hypergraph neural network to learn the representation of interactions.
- Besides, we propose an adaptive mini-batch clustering method in the face of the large scale of the interaction data, which performs deep clustering in a mini-batch fashion based on the learned interaction representations.
- Experiments show that HyCLID significantly outperforms state-of-the-art methods across public and industrial datasets for clustering interactions. Furthermore, experiments on recommendation demonstrate the practical value of the cluster patterns discovered by our method in industrial applications.

2 Related Work

In this section, we first introduce the related clustering methods, and then discuss the hypergraph approaches for interaction data.

Clustering, as one of the fundamental tasks of data mining, is widely used for interaction data analysis. Traditional clustering studies are feature-based. They can target a certain type of interacting objects, and encode them as vectors based on their features [7] followed by traditional clustering methods such as K-means [6]. Besides, graph-based clustering is also a representative kind of clustering method, which treats objects as nodes in a graph and leverages structural information of objects for clustering the objects, e.g., spectral clustering [16]. Recently, attributed graph clustering has attracted a mass of attention, which further considers the attributes of nodes [28]. A noticeable trend is the wide adoption of deep learning in clustering [15]. For example, Graph Neural Network (GNN) [12, 11] based clustering methods can effectively learn representations based on the graph structure and node attributes for clustering [1, 21, 2, 18, 29]. However, modeling the interaction data using the graph structure, which can only model the pairwise relations, will inevitably lose some information since an interaction usually involves more than two objects [27].

Hypergraphs, as generalizations of graphs, can model complex and extensive information and be more suitable for interaction data [33, 25, 20, 8]. How to develop hypergraph structure based solutions for interaction data attracts increasing attention. [34] puts forward the concept of learning with hypergraphs, which can be used in clustering, classification, and embedding and achieves performance beyond the ordinary graphs. Hypergraph neural networks (HGNN) proposed in [3], which are similar to the GCNs in graphs [12], extend the convolution operation to the process of hypergraph learning. [35] proposes an unsupervised method to conduct both subspace



$e_1 = \{v_1, v_2, v_3, v_8, v_9\}$
$e_2 = \{v_2, v_5\}$
$e_3 = \{v_4, v_5, v_6, v_7\}$
……
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9
e_1 e_2 e_3
Layer 3
Layer 2
Layer 1
Attentive Routing-based Rethinking
node aggregation for hyperedge
hyperedge aggregation for node
node embeddings
$\mathcal{L}_{ae}$
hyperedge embeddings
clustering assignments
Q
standard-ization
$\mathcal{L}_c$
adaptive target
P
$\hat{Q}$

Interaction Modeling via Hypergraph **Rethinking-based Hypergraph Neural Network** **Adaptive Mini-Batch Clustering**

Figure 2: Illustration of HyCLID (object attributes and interaction conditions are omitted for easy to understand).

learning and feature selection, but its complex optimization details do not make it perform better than the above graph methods for clustering tasks. A recent study thereby proposes an end-to-end clustering method based on hypergraph neural network [8]. However, these methods only target the individual node/object, which breaks the intrinsic collaborative associations among the involved objects and conditions. The related works for clustering the interactions still are little exploration.

3 Methodology

First of all, we present an overview of the proposed HyCLID. The basic idea is that, as illustrated in Figure 2, with modeling interactions via attributed hypergraph, we propose a novel rethinking-based hypergraph neural network to learn the representations of interactions for clustering. Specifically, to model the interactions that involve multiple objects and conditions, we construct an attributed hypergraph, where each interaction is modeled as a hyperedge connecting several nodes that represent its involved objects. Besides, the attributes of objects and conditions are introduced as node features and hyperedge features, respectively. Then, a novel rethinking-based hypergraph neural network is proposed to aggregate node and hyperedge features to obtain interaction and object representations. During feature aggregation, we design a novel attentive routing-based rethinking mechanism, which can capture high-order features and then rethink to check for omissions and errors. Finally, in the face of the large scale of the interaction data, we develop an adaptive mini-batched clustering method to cluster in a mini-batch fashion based on the learned interaction representations. Moreover, it employs an adaptive batch standardization to remove the impact of different data distributions of different batches.

3.1 Interaction Modeling via Hypergraph

As mentioned above, an interaction usually involves multiple attributed objects and the corresponding interaction conditions such as temporal-spatial contexts, etc. We formalize the *interactions* as follows.

Definition 1: Interaction. An interaction is an action collaboratively done by several objects under a certain condition. Formally, given the set of objects $\mathcal{V}$, object attributes $A^{\mathcal{V}}$, and interaction conditions $A^{\mathcal{E}}$, an interaction is defined as $e =< \mathcal{V}_e, A_e^{\mathcal{V}}, A_e^{\mathcal{E}} >$, including all involved objects $\mathcal{V}_e = \{v_1, \cdots, v_{n_e} | v_i \in \mathcal{V}\}$ with their attributes $A_e^{\mathcal{V}} = \{a_1, \cdots, a_{n_e} | a_i \subset A^{\mathcal{V}}\}$, and interaction conditions $A_e^{\mathcal{E}} \subset A^{\mathcal{E}}$.

Take Interaction 1 in Figure 1 as an example, "a fifteen-year-old student ordered a middle-size latte at the Internet celebrity store Starbucks in the afternoon". In this example, the interaction *order* involves the objects including a user (with occupation attribute *student* and age attribute *fifteen*), a merchant (with attribute *Starbucks* and *Internet celebrity*) and an item *coffee* (with attribute *latte* and *middle-size*). Besides, this interaction has a condition *afternoon*. In order to avoid the association breaking in traditional graph modeling, which splits the collaborative association among multiple objects into several pairwise sub-relations, we model the interactions as an attributed hypergraph.

In detail, each of the interactions is modeled as a hyperedge connecting several nodes that represent its involved objects. Moreover, object attributes are the features of corresponding nodes, and condition attributes are the features of the hyperedges. For example, as illustrated in left of Figure 2, for an interaction e_1 involving 5 objects v_1, v_2, v_3, v_8, v_9, we build a hyperedge to connect them. Besides, we attach their object attributes $a_i (i = 1, 2, 3, 8, 9)$ to the node features. Then the condition attributes $\breve{A}^{\mathcal{E}}_{e_1}$, such as temporal-spatial contexts, are attached to the hyperedge features, since they should be seen as the attributes of interactions rather than any object. Therefore, such a hyperedge and its connecting nodes together with their features can represent an instance of interactions. The constructed attributed hypergraph is formalized as follows.

Definition 2: Attributed Hypergraph for Interactions. The hypergraph for interactions is an attributed hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}_V, \boldsymbol{X}_E)$, where the node set $\mathcal{V}$ represents the set of all involved objects, the hyperedge set $\mathcal{E}$ represents all the interactions, $\boldsymbol{X}_V$ and $\boldsymbol{X}_E$ represent the feature matrices of nodes and hyperedges, respectively.

The i-th row of $\boldsymbol{X}_V \in \mathbb{R}^{|\mathcal{V}| \times F_V}$ represents the feature indicators to the attributes $a_i \subset A^{\mathcal{V}}$ of object $v_i \in \mathcal{V}$. Similarly, the j-th row of $\boldsymbol{X}_E \in \mathbb{R}^{|\mathcal{E}| \times F_E}$ represents the feature indicators to the condition attributes $A^{\mathcal{E}}_{e_j} \subset A^{\mathcal{E}}$ of interaction $e_j \in \mathcal{E}$. F_V and F_E are the max value across the attribute number of each object and condition, respectively.

Based on the above modeling, the clustering task of interactions in this paper can be formalized as the clustering on hyperedges $\mathcal{E}$ in the constructed attributed hypergraph, which is different from other clustering methods that mainly concern about clustering of objects $\mathcal{V}$ in interactions [8].

3.2 Rethinking-based Hypergraph Neural Network

To learn the representation of interactions, we propose a novel rethinking-based hypergraph neural network, where an attentive routing-based rethinking mechanism is designed to capture high-order features and avoid omissions. It can automatically identify the correlations of features and form some meaningful combinations of high-order features. Then, it rethinks the identification process several times to prevent omissions and correct errors. Consequently, the network can learn effective representation of interactions.

3.2.1 Input Layer.

In order to facilitate capturing high-order features, following [19], we initial the representations of each feature from hyperedges or nodes as vectors, thus the representations of hyperedges and nodes are formed as feature matrices. Formally, the initial d-dimensional representations of hyperedge $e_i \in \mathcal{E}$ and node $n_j \in \mathcal{N}$ are $\boldsymbol{e}_i^{(0)} \in \mathbb{R}^{F_E \times d}$ and $\boldsymbol{v}_j^{(0)} \in \mathbb{R}^{F_V \times d}$, where each row of the feature matrix denotes a specific attribute of the node/hyperedge.

3.2.2 Layer-wise Aggregation.

For the hypergraph $\mathcal{G}$, the incidence matrix is defined as $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ with entries $h(v, e) = 1$ if $v \in e$ and 0 otherwise. Denote the diagonal matrices of the edge degrees, the node degrees and the pre-defined weights of hyperedges (default is 1) as $\boldsymbol{D}_e$, $\boldsymbol{D}_v$ and $\boldsymbol{W}$, respectively. The spectral hypergraph convolution for node embedding [3, 8] is formulated as

$$\boldsymbol{Y} = \boldsymbol{D}_v^{-1/2} \boldsymbol{H} \boldsymbol{W} \boldsymbol{D}_e^{-1} \boldsymbol{H}^\top \boldsymbol{D}_v^{-1/2} \boldsymbol{X}_V \boldsymbol{\Theta}, \tag{32}$$

where $\mathbf{\Theta}$ is the trainable filter parameters and $\boldsymbol{X}_V$ is the input node embeddings. It can be seen that in this hypergraph convolution formula, neither can the features of the hyperedge be exploited, nor can the representation of the hyperedges (interactions) be obtained.

To solve these problems, we analyze this formula intuitively and find that this form can be also understood as the following information flow: The node embeddings are first aggregated into hyperedge embeddings by multiplying matrix $\boldsymbol{H}^\top$. Then the hyperedge information is passed back to the nodes, achieved by multiplying $\boldsymbol{H}$. We can formalize these two information aggregation processes as a two-stage aggregation rule from layer l to layer $l+1$ as follows.

$$\boldsymbol{e}_i^{(l+1)} = \text{aggr}(\boldsymbol{e}_i^{(l)}, \{\boldsymbol{v}_j^{(l)} \mid v_j \in e_i\}), \tag{33}$$

$$\boldsymbol{v}_j^{(l+1)} = \text{aggr}(\boldsymbol{v}_j^{(l)}, \{\boldsymbol{e}_i^{(l+1)} \mid v_j \in e_i\}). \tag{34}$$

For the aggregation function $\text{aggr}(\cdot)$, we first concatenate all its inputs into an whole feature matrix $X = \text{concat}(\boldsymbol{e}_i^{(l)}, \{\boldsymbol{v}_j^{(l)} | v_j \in e_i\})$ or $X = \text{concat}(\boldsymbol{v}_j^{(l)}, \{\boldsymbol{e}_i^{(l+1)} | v_j \in e_i\})$, then we propose the following module to capture the high-order features based on X before obtaining the node/hyperedge embeddings. Please note that the size of X is not fixed. The number of its rows depends on the number and size of the concatenated feature matrices of hyperedges and nodes. For example, the size of the whole feature matrix for Eq. 33 is $(F_E + |e_i| \cdot F_V) \times d$. Denoting the calculation of this module as function $\text{ARR}(\cdot)$ for short, the aggregation function can be formalized as $\text{aggr}(\cdot) = \text{ARR}(\text{concat}(\cdot))$.

3.2.3 Attentive Routing-based Rethinking Mechanism

As introduced before, the interactions often arise from the intrinsic correlations among several related attributes. Motivated by previous studies [19, 24], the correlations among p attributes are formalized as a p-order features. As shown in Figure 3, we adopt the framework of multiple self-attentive layers to capture high-order features. Furthermore, in order to prevent omissions of potential high-order features, we need to review the capturing process to check for omissions and errors. Consequently, we introduce a routing mechanism and modify the self-attentive layers to rethink and readjust the captured high-order features. The details are introduced as follows.

Following [19, 24], we adopt the framework of self-attention to learn high-order features due to its superiority for saving computational and storage. However, it needs further improvement to support rethinking. Specifically, we introduce a rethinking vector $\hat{z}$, the attention score between feature x_m and x_k, i.e., the m-th and k-th rows of the input feature matrix X, is calculated as,

$$\varphi(\hat{z}; x_m, x_k) = \text{sum}\left(\hat{z} \odot W_{\text{Q}} x_m \odot W_{\text{K}} x_k\right), \tag{35}$$

where $\odot$ represents element-wise product, i.e., Hadamard product. W_{Q} and W_{K} denote the transformations for Query vector and Key vector in attention. Then the group of second-order features related to feature x_m can be combined together through the framework of self-attention as follows.

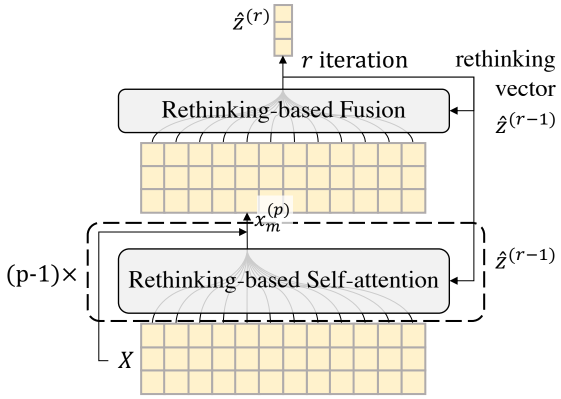


Figure 3: Illustration of the attentive routing-based rethinking mechanism.

$$\alpha_{m,k} = \exp\left(\varphi(\hat{z}; x_m, x_k)\right) / \sum\nolimits_{j=1} \exp\left(\varphi(\hat{z}; x_m, x_j)\right), \tag{36}$$

$$x_m^{(2)} = x_m + \sum\nolimits_{k=1} \alpha_{m,k}(W_{\text{V}} e_k), \tag{37}$$

where W_{V} denotes the transformation for Value vector in attention. Next, a higher-order feature $x_m^{(p)}$ related to x_m can be captured by stacking $p-1$ layers of such self-attention with residue connections. Eventually, the final combination $\hat{z}^{(1)}$ of feature of order p can be obtained by attentive fusion of $x_m^{(p)}, m = 1, 2, \cdots,$

$$\hat{z}^{(1)} = \sum\nolimits_m \text{SoftMax}\left((\hat{z} \odot \boldsymbol{x}_m^{(p)})^\top \cdot \boldsymbol{a}\right) \cdot \boldsymbol{x}_m^{(p)}\ , \tag{38}$$

where $\boldsymbol{a}$ is the vector of attention parameters.

Furthermore, to prevent omissions of potential high-order features, we introduce a routing mechanism to rethink and readjust the above capturing process. Specifically, we update the rethinking vector by the new obtained $\hat{z}^{(1)}$, and repeat the above procedure several times until routing-by-agreement, which is called attentive routing-based rethinking mechanism. Therefore, each repetition of the above process, i.e., each iteration of the routing mechanism, refines the captured high-level features. Initially, let $\hat{z} = \text{mean}_m(W_{\mathrm{V}} x_m)$.

The output high-order feature after repeating r times is denoted as $\hat{z}^{(r)}$, which is the output vector of this module. Formally, given an input feature matrix X, we have $\text{ARR}(X) = \hat{z}^{(r)}$. Then, the Eq. (33) can be rewritten by $\boldsymbol{e}_i^{(l+1)} = \text{ARR}(X)$, where X is obtained by concatenating the embedding matrices of a target hyperedge and its connecting nodes.

3.2.4 Interaction & Object Embedding.

Since each layer of propagation represents a specific order[2] of relations, we sum the embeddings from each layer as the final representation both for hyperedges (interactions) and nodes (objects). Formally, we have $\boldsymbol{e}_i = \sum_l^L \boldsymbol{e}_i^{(l)}$ and $\boldsymbol{v}_j = \sum_l^L \boldsymbol{v}_j^{(l)}$, where L denotes the number of aggregation layers.

3.3 Adaptive Mini-batch Clustering

For clustering in a mini-batch fashion, the different data distributions of batches may lead to clustering errors [26], thus making clustering difficult to discover the implicit patterns of interaction data. Therefore, enlightened by existing deep clustering methods [23], we propose an adaptive mini-batch clustering method, which first removes the impact of different data distributions via an adaptive batch standardization and then performs deep clustering on interactions.

Specifically, after the above modules, we have obtained the representations of interactions. Enlightened by Clustering Assignment Hardening [23], we take Student's t-distribution as a kernel to measure the similarity between points and centroids. Then the soft assignment matrix Q is formulated as follows:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\nu)^{-\frac{\nu+1}{2}}}, \tag{39}$$

where z_i is the embedding of the i^{th} interaction, μ_j is the j^{th} cluster centroid, and ν is the degrees of freedom of the Student's t-distribution, e.g. $\nu = 1$.

Due to the randomness of batch generating process, the assignment distribution Q can be very unstable. It severely limits the robustness of clustering, which in turn brings difficulties to discovering patterns in interaction data. To tackle this issue, we first square the Q distribution to speed up training and then batch standardize it to obtain $\hat{Q}$, which removes the influence of batch quality on the clustering performance. Formally,

$$\widehat{q}_{iu} = (q_{iu}^2 - \mu_u)\,/\,(\sqrt{\sigma_u^2 + \epsilon})\,, \tag{40}$$

$$\text{where}\quad \mu_u = \frac{1}{m}\sum\nolimits_i q_{iu}^2 \quad\text{and}\quad \sigma_u^2 = \frac{1}{m}\sum\nolimits_i \left(q_{iu}^2 - \mu_u\right)^2. \tag{41}$$

[2] Please note that the order of relations is the path length between neighbors and the target node, while the order of features is the number of features combined into high-order feature.

Table 16: Statistics of datasets.

Dataset	# Nodes						# Hyperedges	# Attributes	# Categories	Graph Density
ACM	Paper	4,025	Author	7,167	Field	60	4,025	1,902	3	$\sim 10^{-4}$
IMDB	Movie	4,661	Director	5,841	Actor	2,270	4,661	1,256	3	$\sim 10^{-4}$
MT-S	User	19,623	POI	6,489	Item	17,564	20,000	901	5	$\sim 10^{-5}$
MT-L	User	991,914	POI	17,120	Item	389,292	2,000,000	901	5	$\sim 10^{-6}$

Note that this standardization operation makes $\hat{Q}$ appear negative numbers. It means that we can just erase the low-probability cluster assignment via ReLU function, which contributes to to reduce the difficulty of clustering training and improve performance. Furthermore, we introduce two batch adaptive trainable parameters γ, β to enhance its adaptive ability as follows.

$$p_{iu} = \frac{\mathrm{ReLU}(\gamma_u \widehat{q}_{iu} + \beta_u)}{\sum_s \mathrm{ReLU}(\gamma_s \widehat{q}_{is} + \beta_s)}\ , \qquad s.t. \sum\nolimits_u \gamma_u = 1, \tag{42}$$

where γ represents the relative stability between different clusters (small value means that the samples of one cluster are prone to be modified to another cluster during training), and β defines the aforementioned low-probability cluster assignment. Consequently, the adaptive target distribution $P = [p_{iu}]$ forces assignments to have stricter probabilities (closer to 0 or 1). Meanwhile, we not only preserve the stability of the Q distribution by $\hat{Q}$, but also increase its adaptive capacity. Finally, KL-divergence is applied to let the raw assignments Q approach the target distribution P. which can be minimized for the aforementioned Q and P via neural network training. It emphasizes data points assigned with batch-adaptive high confidence.

3.4 Model Training

We apply an auto-encoder structure for self-supervised training of our HyCLID. Specifically, we reconstruct the incidence matrix H of the constructed hypergraph for interactions with a contrastive loss:

$$\mathcal{L}_{ae} = \frac{1}{2}\sum_{i,j} y_{ij} d_{ij}^2 + (1 - y_{ij})\max(0, m - d_{ij})^2, \quad \text{where} \;\; d_{ij} = \|\boldsymbol{e}_i - \boldsymbol{v}_j\|. \tag{43}$$

y_{ij} denotes the existence of a relationship between hyperedge i and node j, and m is the margin hyper-parameter, e.g., $m = 1$.

Finally, we jointly optimize the auto-encoder structure and deep clustering so that the total objective function is defined as $\mathcal{L} = \mathcal{L}_c + \gamma \mathcal{L}_{ae}$, where $\gamma \geqslant 0$ is a coefficient that controls the balance in between.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

Our proposed HyCLID is evaluated on the following four datasets, and the statistics of these datasets are shown in Table 16.

- **ACM.** The ACM dataset[3] contains three types of nodes: paper, author and field. We treat a publishing as an interaction, and hence build a hypergraph where each hyperedge connects a paper and all its corresponding authors and fields. Considering there are no labels on the hyperedges, we treat the category of the published paper as the clustering ground truth since each paper has a one-to-one correspondence to a hyperedge.

[3] https://data.dgl.ai/dataset/ACM.mat

- **IMDB.** The IMDB dataset [31] contains three types of nodes: movie, actors and directors. Similarly, we treat each movie as an interaction and construct a hypergraph where each hyperedge connects a movie, its actors and directors. The category of the movie is regarded as the clustering ground truth.
- **MT-L** and **MT-S.** We also build two real-world datasets from the food delivery industry, i.e., Meituan Waimai platform. One contains millions of orders of user purchases of foods in Beijing District within 30 days, denoting as MT-L. Each purchase order is treated as an interaction and is tagged with one of 5 purchase scenes. We construct an attributed hypergraph where each hyperedge represents an order, connecting a user node, a restaurant node and several item nodes. The features of nodes are their attributes (e.g., user profiles for user node), while the features of hyperedges (orders) are the interaction conditions, i.e., temporal-spatial contexts, since these attributes do not belong to a single node. Most baseline methods cannot support data of such scale, so we extracted a smaller dataset from it, denoted as MT-S.

4.1.2 Evaluation Metrics

Following [8], we adopt three popular metrics to assess the quality of the clustering results: clustering accuracy (ACC), normalized mutual information (NMI) and adjusted Rand index (ARI). For all the experiments, we repeat them 10 times and report the averaged results and standard deviations.

4.1.3 Baselines

We compare our proposed method HyCLID with the following three groups of methods. Traditional Methods: *K-means* [6], *node2vec* [4] and *HERec* [17]. Attributed Graph-based Methods: *SDCN* [1], *HAN* [22], *HGT* [9] and *AdaGAE* [14]. Hypergraph-based Methods: *HGNN* [3] and *AHGAE* [8]. Note that for the methods considering graph structure, for ACM and IMDB datasets, we directly use the widely applied graph structure [9]. For the MT-S dataset, we transfer the hypergraph into a graph, i.e., we introduce a summary node to replace each of hyperedges and link the summary nodes with the corresponding nodes that are originally connected by hyperedges. For metapath-based methods, i.e., HERec and HAN, we select all the possible symmetric length-2 meta-paths.

4.1.4 Implementation Detail

We implement the proposed method based on Tensorflow[4]. For our method, we set the dimension of attribute embeddings as $d = 64$ for the public datasets ACM and IMDB for fair comparison and $d = 16$ for MT-S/L for saving memory. For simplicity, we set the number of aggregation layers $L = 1$, the layer number of rethinking-based self-attention $p = 1$ and the rethinking iterations $r = 2$. For all baselines, we set their hidden dimensions as 64 and set the number of layers as their suggested value (usually equal to 2). For model training, we simply set $\gamma = 1$, and apply Adam [10] to optimize with the learning rate as 0.005 for ACM and IMDB and 0.001 for MT-S/L. The batch size is set 2048. All the experiments are performed in NVIDIA Tesla P40 Cluster. To facilitate related research, we will release our implementation to the public once the paper gets accepted.

4.2 Analysis of Clustering Results

Table 17 shows the clustering results on the three small-scale datasets. We have the following observations:

As shown, for each metric, our proposed HyCLID achieves the best results significantly. In particular, compared with the best results of the baselines, our approach achieves a significant improvement of 6.3% on ACC, 17.9% on NMI, 15.9% on ARI averagely. The reason is that HyCLID successfully makes full use of all the information in interactions, including node attributes, hyperedge attributes and hyper-relations. In detail, attributed

[4] https://www.tensorflow.org/

Table 17: Clustering results on datasets ACM, IMDB and MT-S (mean±std in percent). The best and second best results are bold and underlined, respectively. Symbol "-" represents unavailable results due to out-of-memory.

Method	ACM			IMDB			MT-S		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means	63.59 ± 0.27	36.89 ± 0.37	28.91 ± 0.42	37.01 ± 0.10	0.88 ± 0.06	1.43 ± 0.05	54.95 ± 8.08	56.36 ± 6.69	50.08 ± 7.83
node2vec	65.25 ± 0.33	38.49 ± 0.12	31.23 ± 0.88	47.87 ± 0.33	5.55 ± 0.53	<u>5.89 ± 0.64</u>	45.09 ± 6.70	23.19 ± 5.60	18.65 ± 7.78
HERec	51.41 ± 1.82	20.05 ± 1.37	20.82 ± 1.50	46.45 ± 0.62	5.32 ± 0.79	5.68 ± 0.69	39.76 ± 9.72	20.57 ± 2.04	18.32 ± 3.11
SDCN	67.79 ± 0.87	41.77 ± 0.80	37.69 ± 0.60	43.87 ± 0.35	3.37 ± 0.19	2.77 ± 0.21	73.56 ± 9.27	66.34 ± 8.02	62.80 ± 7.83
HAN	69.33 ± 0.38	44.17 ± 0.58	38.94 ± 0.36	42.43 ± 0.32	2.90 ± 0.37	2.89 ± 0.29	55.71 ± 5.96	54.09 ± 6.41	48.34 ± 8.25
HGT	75.31 ± 1.17	46.98 ± 2.28	43.06 ± 1.63	<u>48.07 ± 0.02</u>	<u>5.62 ± 0.06</u>	5.23 ± 0.05	55.22 ± 8.13	57.38 ± 7.62	51.11 ± 9.27
AdaGAE	<u>78.32 ± 3.76</u>	<u>50.94 ± 3.12</u>	<u>54.03 ± 4.09</u>	47.12 ± 3.17	5.07 ± 1.86	4.58 ± 1.73	-	-	-
HGNN	66.35 ± 2.29	31.17 ± 4.05	27.72 ± 5.32	44.11 ± 0.11	2.05 ± 0.01	1.31 ± 0.04	80.75 ± 4.32	76.70 ± 5.83	68.46 ± 3.91
AHGAE	75.76 ± 1.83	46.53 ± 2.98	41.28 ± 3.64	40.05 ± 1.54	1.73 ± 1.35	1.58 ± 1.07	<u>84.66 ± 8.01</u>	<u>79.16 ± 8.98</u>	<u>71.53 ± 9.23</u>
HyCLID	**83.63 ± 1.23**	**57.29 ± 0.76**	**58.45 ± 1.91**	**50.28 ± 0.39**	**7.43 ± 0.63**	**7.13 ± 1.02**	**91.05 ± 8.61**	**86.30 ± 6.24**	**84.70 ± 9.44**
Impr. (%)	6.78	12.46	8.18	4.60	32.20	21.05	7.55	9.02	18.41

graph-based methods can generally achieve better performance than traditional methods, especially heterogeneous GNN-based end-to-end methods (HGT), which confirms the important role of object attributes and structures among objects in the interaction data. Unexpectedly, hypergraph-based methods (HGNN, AHGAE) have not achieved competitive performance on these two datasets, while our method HyCLID successfully performs the best. We hold that this is because these hypergraph-based methods cannot capture high-order correlations (as our rethinking-based self-attention) nor be adaptive during clustering, which makes their hypergraph neural networks fail to fully utilize the information in the interaction data. Conversely, the hypergraph-based baselines achieve outstanding performance on the MT-S dataset. It suggests the necessity of attributed hypergraph modeling for real-world interaction data, which involves multiple attributed objects and a certain interaction condition. Besides, the fact that our HyCLID further performs better confirms again the effectiveness of our proposed attentive routing-based rethinking mechanism to capture high-order features and the adaptive mini-batch clustering for training.

Moreover, we also conduct a comparison on the MT-L dataset. Since most baselines cannot support running on such a large-scale dataset, we only employ mini-batch K-Means for comparison. As shown in Table 18, our proposed HyCLID consistently performs better than the baseline, which demonstrates the effectiveness of our proposed HyCLID for large-scale data.

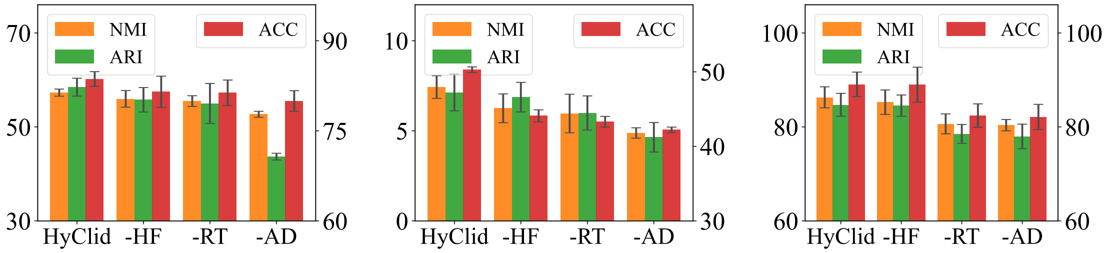
Table 18: Clustering results on dataset MT-L.

Method	ACC	NMI	ARI
Mini-batch K-means	45.82 ± 3.75	24.48 ± 2.32	18.30 ± 3.88
HyCLID	**48.60 ± 3.98**	**28.68 ± 4.92**	**24.50 ± 5.95**

4.3 Ablation Study

In this subsection, for simplicity, we compare our HyCLID with 3 variants on the datasets ACM, IMDB and MT-S to validate the design of each module. Specifically, *-HF* is a variant that removes the part to capture high-order features (i.e., set $p = 0$) based on the complete model. The variant *-RT* represents that the module attentive routing mechanism for rethinking is removed, i.e., the vanilla self-attention is adopted for replacement. *-AD* denotes that the adaptive mini-batch clustering is further removed and replaced by traditional deep clustering, i.e., clustering assignment hardening [23].

As reported in Figure 4, the clustering performance always decreases with the removal of each module. This phenomenon demonstrates the effectiveness of each of the proposed modules. In detail, compare the best performance achieved by our whole model, *-HF* performs worse, which verifies the existence of the correlations



(a) ACM (b) IMDB (c) MT-S

Figure 4: Clustering results of different variants.

among multiple attributes from the involved objects and conditions and confirms the effectiveness of our rethinking based self-attention to capture these correlations. Next, the fact that *-RT* performs more poorly shows the necessity of our rethinking module for avoiding the omissions of some potential high-orders. Furthermore, when the adaptive mini-batch clustering is further substituted by the traditional clustering assignment hardening [23], *-AD* performs the worst. On the one hand, it shows that problems such as data imbalance make the clustering significantly worse, and on the other hand, it demonstrates that our proposed adaptive module can alleviate this problem very well.

4.4 Case Study for Rethinking Mechanism

In this subsection, we dissect how our attentive routing-based rethinking mechanism works. As shown in Figure 5, the heat maps visualize the attention weights of rethinking-based fusion (top) and rethinking-based self-attention (bottom) under two iterations of rethinking (three pairs of heat maps from left to right represent $r = 0, 1, 2$, respectively).

A phenomenon can be observed that, the self-attention assigns relatively high weights to the second-order feature between *coffee* and *afternoon*, while the fusion attention also assigns high weights to *coffee* and *afternoon*. Then, the weights are further growing with the iterations of rethinking. On the contrary, the high weight, assigned to the second-order feature between *lower consumption level* and *afternoon*, is gradually reduced. It validates the effectiveness of our attentive routing-based rethinking mechanism. In detail, it not only encourages the important features or higher-order features through iterative rethinking, but also corrects the overestimated ones.

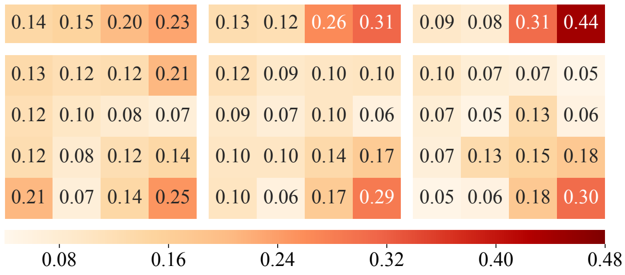


Figure 5: Heat map visualization for attentive routing-based rethinking mechanism. The rows/columns represent four selected attributes from an interaction case belonging to category "afternoon tea" of dataset MT-S, i.e., *lower consumption level*, *student*, *coffee*, *afternoon*, respectively.

4.5 Offline Experiment on Recommendation

In this subsection, we conduct an offline experiment on recommendation to show the applied usage and practical value of our method for food delivery industrial applications, i.e., Meituan Waimai platform. Specifically, we divide the MT-L dataset into training/test subsets, i.e., the orders of the first 27 days are for training while those of the last 3 days are for testing. Then we train our model on the training subset of MT-L and then infer the clustering assignments on the test subset. Finally, these learned and inferred clustering assignments are introduced into context-aware recommendation models, e.g., DeepFM [5] and AutoInt [19], as contexts for CTR prediction, where the dataset partition is the same as above. For comparisons, "-vanilla" represents that no clustering assignments are introduced, and "+HyCLID" represents that our HyCLID is adopted for assignments.

We report the results on several metrics (including HR, Recall, Precision and NDCG) in Table 19. As we can see, for both DeepFM and AutoInt, "+HyCLID" outperforms "-vanilla", indicating that the clustering results of interactions are conducive for downstream applications and demonstrates the necessity of our attributed hypergraph modeling of interactions and the effectiveness of our proposed hypergraph clustering network for clustering interactions.

Table 19: Top-3 recommendation performance.

Method	HR	Recall	Precision	NDCG
DeepFM-vanilla	0.5268	0.4950	0.1849	0.4135
DeepFM+HyCLID	**0.5386**	**0.5056**	**0.1868**	**0.4186**
AutoInt-vanilla	0.5120	0.4811	0.1786	0.4021
AutoInt+HyCLID	**0.5465**	**0.5142**	**0.1906**	**0.4236**

5 Conclusion

In this paper, we propose a novel clustering task on interaction data to discover more comprehensive cluster patterns, where the interactions are regarded as basic units to be clustered instead of objects. To this end, we propose a novel Hypergraph CLustering network for Interaction Data, namely HyCLID. Specifically, we propose to model the interactions via attributed hypergraph, and then propose a novel rethinking-based hypergraph neural network to learn the representation of interactions. It designs a novel attentive routing-based rethinking mechanism to capture the correlations among multiple attributes of objects and conditions involved in interactions. Besides, we develop an adaptive mini-batch clustering method to deal with the large scale of interaction data, which can further make adaptive adjustments for different data distributions of batches. Extensive experiments verify the effectiveness of our HyCLID on both public datasets and real industrial datasets. Moreover, offline experiments on recommendation show the practical value of our discovered cluster patterns for industrial applications.

References

[1] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural Deep Clustering Network. In WWW, 2020.

[2] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. One2Multi Graph Autoencoder for Multi-view Graph Clustering. In WWW, pages 3070–3076. ACM, April 2020.

[3] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. In AAAI, pages 3558–3565, July 2019.

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In SIGKDD, pages 855–864. Association for Computing Machinery, 2016.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In IJCAI, pages 1725–1731, 2017.

[6] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. Applied Statistics, 28(1):100, 1979.

[7] G. E. Hinton. Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786):504–507, July 2006.

[8] Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, Yining Zhao, Chenggang Yan, Jian Yin, and Yue Gao. Adaptive Hypergraph Auto-Encoder for Relational Data Clustering. TKDE, 2021.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous Graph Transformer. In WWW, pages 2704–2710. ACM, 2020.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[11] Thomas N Kipf and Max Welling. Variational Graph Auto-Encoders. In NIPS Workshop on Bayesian Deep Learning, 2016.

[12] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR. OpenReview.net, 2017.

[13] Feifei Kou, Junping Du, Yijiang He, and Lingfei Ye. Social network search based on semantic analysis and learning. CAAI Transactions on Intelligence Technology, 1(4):293–302, October 2016.

[14] Xuelong Li, Hongyuan Zhang, and Rui Zhang. Adaptive Graph Auto-Encoder for General Data Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[15] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. IEEE Access, 6:39501–39514, 2018.

[16] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. NIPS, 14, 2001.

[17] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. IEEE TKDE, 31(2):357–370, 2018.

[18] Hanyu Song, Peizhao Li, and Hongfu Liu. Deep Clustering based Fair Outlier Detection. In SIGKDD 2021, June 2021.

[19] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In CIKM, pages 1161–1170. ACM, 2019.

[20] Loc Hoang Tran and Linh Hoang Tran. Directed hypergraph neural network. arXiv, August 2020.

[21] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In IJCAI, August 2019.

[22] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous Graph Attention Network. In WWW, 2019.

[23] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In ICML, volume 48, pages 478–487. PMLR, June 2016.

[24] Yichen Xu, Yanqiao Zhu, Feng Yu, Qiang Liu, and Shu Wu. Disentangled Self-Attentive Neural Networks for Click-Through Rate Prediction. In CIKM, pages 3553–3557. ACM, 2021.

[25] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In NeurIPS, pages 1511–1522. 2019.

[26] Yilin Yan, Min Chen, Mei-Ling Shyu, and Shu-Ching Chen. Deep Learning for Imbalanced Multimedia Data Classification. In IEEE ISM, pages 483–488, 2015.

[27] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Hypergraph learning with line expansion. arXiv preprint arXiv:2005.04843, 2020.

[28] Renchi Yang, Jieming Shi, Yin Yang, Keke Huang, Shiqi Zhang, and Xiaokui Xiao. Effective and Scalable Clustering on Massive Attributed Graphs. In WWW. ACM, April 2021.

[29] Shuiqiao Yang, Sunny Verma, Borui Cai, Jiaojiao Jiang, Kun Yu, Fang Chen, and Shui Yu. Variational Co-embedding Learning for Attributed Network Clustering. arXiv, 2021.

[30] Tianchi Yang, Cheng Yang, Luhao Zhang, Chuan Shi, Maodi Hu, Huaijun Liu, Tao Li, and Dong Wang. Co-clustering Interactions via Attentive Hypergraph Neural Network. In SIGIR, pages 859–869. ACM, July 2022.

[31] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph Transformer Networks. In NIPS, volume 32. Curran Associates, Inc., 2019.

[32] Rui Zhang, Hongyuan Zhang, and Xuelong Li. Maximum Joint Probability With Multiple Representations for Clustering. IEEE TNNLS, pages 1–11, 2021.

[33] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In ICLR, 2020.

[34] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with Hypergraphs: Clustering, Classification, and Embedding. In NeurIPS, pages 1601–1608, 2006.

[35] Xiaofeng Zhu, Yonghua Zhu, Shichao Zhang, Rongyao Hu, and Wei He. Adaptive Hypergraph Learning for Unsupervised Feature Selection. In IJCAI, 2017.