Learning on Partial-Order Hypergraphs

Fuli Feng
National University of Singapore
fulifeng93@gmail.com

Xiangnan He*
National University of Singapore
xiangnanhe@gmail.com

Yiqun Liu
Tsinghua University
yiqunliu@tsinghua.edu.cn

Liqiang Nie
Shandong University
nieliqiang@gmail.com

Tat-Seng Chua
National University of Singapore
chuats@comp.nus.edu.sg

ABSTRACT

Graph-based learning methods explicitly consider the relations between two entities (*i.e.,* vertices) for learning the prediction function. They have been widely used in semi-supervised learning, manifold ranking, and clustering, among other tasks. Enhancing the expressiveness of simple graphs, hypergraphs formulate an edge as a link to multiple vertices, so as to model the higher-order relations among entities. For example, hyperedges in a hypergraph can be used to encode the similarity among vertices.

To the best of our knowledge, all existing hypergraph structures represent the hyperedge as an unordered set of vertices, without considering the possible ordering relationship among vertices. In real-world data, ordering relations commonly exist, such as in graded categorical features (*e.g.,* users' ratings on movies) and numerical features (*e.g.,* monthly income of customers). When constructing a hypergraph, ignoring such ordering relations among entities will lead to severe information loss, resulting in suboptimal performance of the subsequent learning algorithms.

In this work, we address the inherent limitation of existing hypergraphs by proposing a new data structure named *Partial-Order Hypergraph*, which specifically injects the partially ordering relations among vertices into a hyperedge. We develop regularization-based learning theories for partial-order hypergraphs, generalizing conventional hypergraph learning by incorporating logical rules that encode the partial-order relations. We apply our proposed method to two applications: university ranking from Web data and popularity prediction of online content. Extensive experiments demonstrate the superiority of our proposed partial-order hypergraphs, which consistently improve over conventional hypergraph methods.

CCS CONCEPTS

• **Information systems → Web mining**; *Web applications*; • **Computing methodologies →** *Spectral methods*; Semi-supervised learning settings; • **Applied computing →** *Education*;

*Xiangnan He is the corresponding author. This research is part of the NExT++ project, supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative.

KEYWORDS

Hypergraph, Partial-Order Hypergraph, Graph-based Learning, University Ranking, Popularity Prediction

ACM Reference Format:
Fuli Feng, Xiangnan He, Yiqun Liu, Liqiang Nie, and Tat-Seng Chua. 2018. Learning on Partial-Order Hypergraphs. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186064

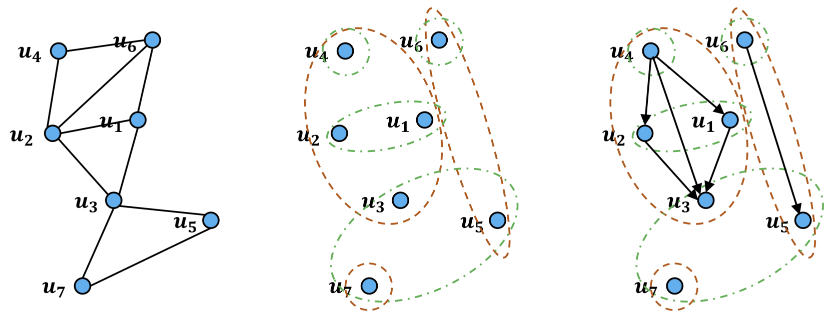
1 INTRODUCTION

Graphs naturally represent relational data and have been widely used to model the relationships between entities. Simple graphs intuitively connect two vertices (*i.e.,* entities of interest) with an edge (*i.e.,* the relationship to model), which can be either undirected or directed depending on whether the pairwise relationship between entities is symmetric. For example, given a set of entities with feature vectors, we can construct an undirected graph by forming the adjacency matrix with a similarity metric [45, 48]. The World Wide Web is a well-known instance of directed graphs, where vertices represent webpages, and edges represent hyperlinks. With such graph representations of entities and their relations, many graph-based learning methods have been developed to address various tasks, such as semi-supervised learning [11, 32, 37, 47], manifold ranking [20, 30, 31, 49], and clustering [4, 8, 40], personalized recommendation [17, 23], and so on.

Hypergraph is a generalizaton of simple graph, in which an edge (*aka.* hyperedge) can connect any number of vertices rather than just two. As such, it can model high-order relations among multiple entities that cannot be naturally represented by simple graphs. Figure 1 shows an illustrative example of using graph methods to tackle the university ranking task [9]. Each university has two features: the located city and the salary level of its graduates (Figure 1a). A simple graph can be constructed by connecting a university with its two nearest neighbors (Figure 1b); then performing a manifold ranking on the simple graph can obtain a ranked list of universities. Further, we can build a hypergraph by connecting universities with a same attribute (Figure 1c), *e.g.,* universities that are located in the same city, which is a high-order relation among universities missed by the simple graph.

In existing research, hyperedges in a hypergraph are typically formed by linking similar entities — either globally similar such as a cluster of entities that are close to each other [27, 36, 38], or locally similar such as sharing a same attribute [3, 35, 43]. However, we argue that many real-world applications need to deal with far more complex relations than similarities. One particular type is

Uni	City	Sal
u_1	c_1	s_2
u_2	c_1	s_2
u_3	c_1	s_1
u_4	c_1	s_4
u_5	c_2	s_1
u_6	c_2	s_3
u_7	c_3	s_1



(a) Data (b) Simple Graph (c) Hypergraph (d) Partial-Order Hypergraph

Figure 1: An example of using graph methods to tackle the university ranking task. (a) Input data, where each row represents a university and its features: city and salary level; for salary level, smaller index indicates higher salary (*i.e.,* $s_1 > s_2 > s_3 > s_4$). (b) A simple graph, where an edge connects a vertex and its two-nearest vertices. (c) A hypergraph, where a hyperedge connects vertices with a same attribute: either in the same city or having the same salary level. (d) A partial-order hypergraph, where the directed edges within an hyperedge represent the partially ordering relationship between vertices on the salary level.

the ordering relationship among entities, which commonly exists in graded categorical features and numerical features. Taking the university ranking task in Figure 1 as an example. Two universities u_5 and u_6 are located in the same city, while u_5 has a salary level much higher than u_6 — an evidence that u_5 might be ranked higher than u_6. Unfortunately, the hypergraph constructed in Figure 1c encodes the similarity information only, thus fails to capture the ordering information on salary. To address this limitation, an intuitive solution is to incorporate the ordering relations by adding directed edges between entities of a hyperedge, as shown in Figure 1d. It is worth noting that not every two entities within a hyperedge have an ordering relation; for example, two entities may have the same graded categorical feature (see u_1 and u_2 in Figure 1d) or the difference on the target numerical feature is not significant enough. As such, we term such generalized hypergraph with partial-order relations on vertices as a *Partial-Order Hypergraph* (POH), which is a new data structure that has never been explored before.

In this work, we formalize the concept of POHs and further develop regularization-based graph learning theories on them. We express the partial-order relations with logical rules, which can be used to encode prior domain knowledge. In the previous example of university ranking, one example of domain knowledge can be that for two universities u_i and u_j in the same city, u_i tends to be ranked higher than u_j if the salary level of u_i is higher than that of u_j. The corresponding logical rule can be written as:

$$city_{=}(u_i, u_j) \wedge salary_{>}(u_i, u_j) \longrightarrow score_{>}(u_i, u_j). \qquad (1)$$

We extend conventional hypergraph learning [38, 48] to incorporate such logical rules for an effective learning on POHs. Besides the improved accuracy, we can further enhance the interpretability of hypergraph learning. Specifically, we can interpret the learning results by verifying the logical rules, rather than relying on the *smoothness* factor only. To justify our proposed partial-order hypergraph and the learning method, we employ them to address two applications: university ranking [9] and popularity prediction [5, 16]; the two tasks are representatives of two machine learning tasks: unsupervised ranking and semi-supervised regression, respectively. Extensive results demonstrate the superiority of our proposed method, which significantly outperforms existing simple graph and hypergraph methods.

The key contributions of the paper are summarized as follows.

- We propose a novel *partial-order hypergraph* to represent the partial-order relations among vertices, which are missed by the traditional hypergraph.
- We generalize existing graph-based learning methods to partial-order hypergraphs to encode domain knowledge in the format of logical rules.
- We empirically demonstrate the effectiveness of our POH and learning method on two machine learning tasks of unsupervised ranking and semi-supervised regression.

The remainder of this paper is organized as follows. Section 2 introduces some preliminary knowledge. In Section 3, we present our proposed methods, followed by reviewing related work in Section 4. In Section 5 and 6, we employ POH learning to address the two tasks of university ranking and popularity prediction, respectively. We conclude the work in Section 7.

2 PRELIMINARIES

We introduce some notations first. We use bold capital letters (*e.g.,* $\mathbf{X}$) and bold lowercase letters (*e.g.,* $\mathbf{x}$) to denote matrices and vectors, respectively. Scalars and hyperparameters are respectively represented as normal lowercase letters (e.g., x) and Greek letters (e.g., λ). If not otherwise specified, all vectors are in a column form, and X_{ij} denotes the entry at the i-th row and the j-th column of $\mathbf{X}$.

2.1 Hypergraph

As aforementioned, the vertices and hyperedges of a hypergraph represent the entities of interest and their relations, respectively. Given N entities with their features $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$, we can construct a hypergraph with N vertices and M hyperedges, for which the structure can be represented as an incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$. Similar to the incidence matrix of a simple graph, $\mathbf{H}$ is a binary matrix, where $H_{ij} = 1$ if the i-th vertex is connected by the j-th hyperedge, otherwise $H_{ij} = 0$. There are two ways to define a hyperedge in existing work: attribute-based [3, 22, 35, 43] and neighbor-based [27, 36, 38]. An attribute-based hyperedge connects vertices with same value on one or multiple target attributes (*i.e.,* features). A neighbor-based hyperedge connects vertices nearby, based on these vertices with similarity larger than a threshold or simply using the k-nearest neighbors.

Moreover, we use the diagonal matrix $\mathbf{E} \in \mathbb{R}^{M \times M}$ to denote the degrees of hyperedges, *i.e.,* $E_{jj} = \sum_{i=1}^{N} H_{ij}$ denotes the number of vertices connected by the j-th hyperedge. Analogous to simple graph that an edge typically has a weight to model the strength of the relation, a hyperedge in hypergraphs also has a weight to denote the density of the vertices it connected. Such weights are represented as a diagonal matrix $\mathbf{W} \in \mathbb{R}^{M \times M}$. To estimate the hyperedge weight, many methods have been proposed, among which the most popular one is to use the average pairwise similarity between vertices connected by the hyperedge:

$$W_{jj} = \frac{1}{E_{jj}} \sum_{H_{ij}=1} g(\mathbf{x_i}, \mathbf{x_j}), \tag{2}$$

where g denotes the similarity function on feature vectors. In graph-based methods, one common choice for g is the radial basis function (RBF) kernel, *i.e.,* $g(\mathbf{x_i}, \mathbf{x_j}) = \exp(\frac{-\|\mathbf{x_i}-\mathbf{x_j}\|^2}{2\sigma^2})$. Given the weights for hyperedges, we can further derive the degree of a vertex i: $V_{ii} = \sum_{j=1}^{M} W_{jj} H_{ij}$, *i.e.,* the sum of weights of hyperedges that are connected with i. We use the diagonal matrix $\mathbf{V} \in \mathbb{R}^{N \times N}$ to denote the vertex degree matrix.

2.2 Learning on Hypergraphs

Graph-based learning has been applied to various machine learning tasks such as manifold ranking, semi-supervised learning, and clustering [1, 26, 44]. The general problem setting is to learn a prediction function $\hat{\mathbf{y}} = f(\mathbf{x})$, which maps an entity from the feature space to the target label space. It is usually achieved by minimizing an objective function abstracted as:

$$\Gamma = \mathcal{G} + \lambda \mathcal{L}, \tag{3}$$

where $\mathcal{L}$ is a task-specific loss function that measures the error between prediction $\hat{\mathbf{y}}$ and ground-truth $\mathbf{y}$, $\mathcal{G}$ is a graph regularization term that smooths the prediction over the graph, and λ is a hyperparameter to balance the two terms. The regularization term typically implements the *smoothness* assumption that similar vertices tend to have similar predictions. For hypergraphs, a widely used $\mathcal{G}$ is the hypergraph Laplacian term, defined as:

$$\mathcal{G} = \sum_{i=1}^{N} \sum_{j=1}^{N} \underbrace{g(\mathbf{x_i}, \mathbf{x_j}) \sum_{k=1}^{M} H_{ik} W_{kk} H_{jk}}_{\text{strength of smoothness}} \underbrace{\left\| \frac{f(\mathbf{x_i})}{\sqrt{V_{ii}}} - \frac{f(\mathbf{x_j})}{\sqrt{V_{jj}}} \right\|^2}_{\text{smoothness}}. \tag{4}$$

The regularization term operates smoothness on each pair of entities, enforcing their predictions (after normalized by their degrees) to be close to each other. The strength of smoothness is determined by the similarity over their feature vectors $g(\mathbf{x_i}, \mathbf{x_j})$ and the hyperedges that connect the two vertices. It can be equivalently written in a more concise matrix form:

$$\mathcal{G} = trace(\hat{\mathbf{Y}}(\mathbf{S} \odot \mathbf{L})\hat{\mathbf{Y}}^T), \tag{5}$$

where $\hat{\mathbf{Y}} = [\hat{\mathbf{y_1}}, \hat{\mathbf{y_2}}, \cdots, \hat{\mathbf{y_N}}]$, each element of $\mathbf{S}$ is $S_{ij} = g(\mathbf{x_i}, \mathbf{x_j})$, and $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{V}^{-1/2}(\mathbf{V} - \mathbf{H}\mathbf{W}\mathbf{H}^T)\mathbf{V}^{-1/2}$, known as the *hypergraph Laplacian matrix.* Note that $\mathbf{L}$ is typically a sparse matrix, where an entry L_{ij} is nonzero only if vertex i and j are connected by at least one hyperedge. Thus, the time complexity of calculating $\mathcal{G}$ is linear *w.r.t.* the number of nonzero entries in $\mathbf{L}$, which is far smaller than N^2.

3 PARTIAL-ORDER HYPERGRAPH

Distinct from the typical problem setting of hypergraph learning, we further associate the problem with a set of logic rules, which can be used to encode the partial-order relations between entities:

$$\left\{ p^r(\mathbf{x_i}, \mathbf{x_j}) \rightarrow q^r(f(\mathbf{x_i}), f(\mathbf{x_j})) \mid r = 1, 2, \cdots, R \right\}, \tag{6}$$

where r denotes a partial-order relation, and there can be multiple relations (in total R) for a problem. For example, in the university ranking task, we can have a partial-order relation based on salary level, number of students, research grants among other features. For each partial-order relation r, q^r is a binary function indicating whether the prediction of two entities satisfies a certain relation. For example, in a ranking/regression task, q^r can indicate whether $f(\mathbf{x}_i)$ is higher than $f(\mathbf{x}_j)$; in a classification task, q^r can indicate whether the probability of $\mathbf{x_i}$ being a class is higher than that of $\mathbf{x_j}$. The $p^r(\mathbf{x_i}, \mathbf{x_j})$ denotes whether $\mathbf{x}_i$ and $\mathbf{x}_j$ have the partial-order relation r. A partial-order relation is a pairwise relation satisfying the following basic properties on the entities connected by any hyperedge:

- Irreflexivity: not $p^r(\mathbf{x_i}, \mathbf{x_i})$.
- Asymmetry: if $p^r(\mathbf{x_i}, \mathbf{x_j})$ then not $p^r(\mathbf{x_j}, \mathbf{x_i})$.
- Transitivity: $p^r(\mathbf{x_i}, \mathbf{x_j})$ and $p^r(\mathbf{x_j}, \mathbf{x_k})$ implies $p^r(\mathbf{x_i}, \mathbf{x_k})$.

In what follows, we first present how to construct and represent a POH. We then elaborate our proposed regularized learning on POHs. Lastly, we analyze its time complexity.

3.1 Construction and Representation

To jointly represent the partial-order relations and the higher-order relations among entities, we first construct a normal hypergraph, and then use directed edges to connect vertices that have any partial-order relation. Note that it is possible that there are multiple edges between two vertices, since multiple partial-order relations are considered. Concerning the efficiency of downtream graph-based learning applications, we constrain that directed edges only exist on vertices connected by at least one hyperedge. Such a constraint guarantees that the number of directed edges constructed from a partial-order relation is no larger than the number of nonzero entries in the hypergraph Laplacian matrix. As such, a learning algorithm that enumerates all directed edges will not increase the time complexity of calculating the hypergraph Laplacian term.

As described in Section 2.1, after constructing a hypergraph, we use several matrices to represent it, such as the incidence matrix $\mathbf{H}$ and hypergraph Laplacian matrix $\mathbf{L}$. In addition to these matrices, we further introduce a partial incidence matrix $\mathbf{H^r} \in \mathbb{R}^{N \times N}$ to represent the directed edges of a partial-order relation r. As shown in Figure 2, given a partial-order relation r, we first construct a binary relation matrix $\mathbf{P^r} \in \mathbb{R}^{N \times N}$, where $P^r_{ij} = 1$ if $p^r(\mathbf{x_i}, \mathbf{x_j})$ is

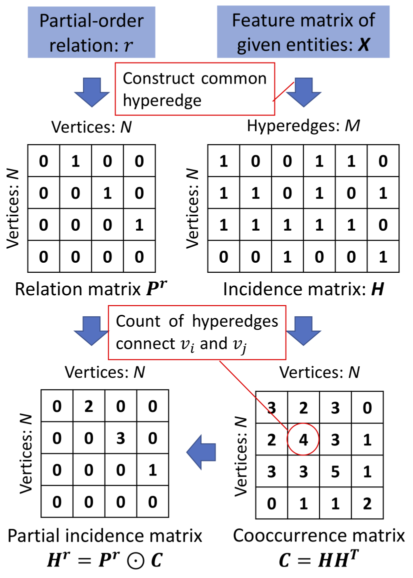

Partial-order relation: r
Feature matrix of given entities: $\boldsymbol{X}$
Construct common hyperedge
Vertices: N
Hyperedges: M
Vertices: N
Relation matrix $\boldsymbol{P}^r$
Incidence matrix: $\boldsymbol{H}$
Count of hyperedges connect v_i and v_j
Partial incidence matrix $\boldsymbol{H}^r = \boldsymbol{P}^r \odot \boldsymbol{C}$
Cooccurrence matrix $\boldsymbol{C} = \boldsymbol{H}\boldsymbol{H}^T$

Figure 2: A toy example to illustrate the construction of matrix representation of a POH. Given the feature matrix X and a partial-order relation r, we construct the incidence matrix H of the hypergraph and the binary relation matrix $\mathbf{P}^r$, respectively. We then generate the co-occurrence matrix C from H and apply element-wise product to C and $\mathbf{P}^r$ to get the partial incidence matrix $\mathbf{H}^r$.

true. Based on the incidence matrix $\mathbf{H}$ of the hypergraph, we further build a co-occurrence matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$, where each element C_{ij} denotes the number of hyperedges connecting vertex i and j. Then the partial incidence matrix $\mathbf{H}^r$ can be derived,

$$\mathbf{H}^\mathbf{r} = \mathbf{P}^\mathbf{r} \odot \mathbf{C}, \tag{7}$$

where $\odot$ is the element-wise matrix multiplication. In the partial incidence matrix, a non-zero entry H^r_{ij} means that the i-th and j-th vertices have the r-th partial-order relation, and they are simultaneously connected by H^r_{ij} hyperedges. In other words, we assign higher weights to vertex pairs that are connected by more hyperedges, accounting for the effect that vertex pairs with higher co-occurrence are more important.

3.2 Learning on Partial-Order Hypergraphs

After constructing a POH, we have several matrices to represent it, including the general ones describing a conventional hypergraph (*e.g.,* the incidence matrix $\mathbf{H}$ and edge weight matrix $\mathbf{W}$), and the specific partial incidence matrices $\{\mathbf{H}^\mathbf{r} | r = 1, 2, \cdots, R\}$ to model partial-order relations. We now consider how to extend the conventional hypergraph learning methods for POHs.

The key problem is the encoding of the partial-order relations and the corresponding logical rules into the learning framework of Equation (3). Generally speaking, there are two solutions — either injecting the rules into the predictive model (*i.e.,* the definition of $f(\mathbf{x})$), or using the rules to regularize the learning (*i.e.,* augmenting the objective function Γ). The first solution may need different ways to encode the rules for different predictive models, such as logistic regression, factorization machines [15], and neural networks [14]. As such, we opt for the second solution, aiming to provide a generic solution for POH learning. Specifically, we append an additional regularization term $\mathcal{P}$ that encodes partial-order rules to the objective function:

$$\Gamma = \mathcal{G} + \lambda \mathcal{L} + \beta \mathcal{P}, \tag{8}$$

where β is a hyperparameter to balance $\mathcal{P}$ and the other two terms. Similar to the smoothness regularizer $\mathcal{G}$, $\mathcal{P}$ should also operate on the predicted label space to regularize the learning process. We define $\mathcal{P}$ as:

$$\begin{aligned}\mathcal{P}_0 &= \sum_{r=1}^{R} a_r \mathbb{E}_{(i,j)\sim H^r_{ij} \neq 0} \left[1 - q^r(\hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{j})\right], \\ &= \sum_{r=1}^{R} \frac{a_r}{|\mathbf{H}^\mathbf{r}|} \sum_{\{i,j|H^r_{ij}\neq 0\}} 1 - q^r(\hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{j}),\end{aligned} \tag{9}$$

where $\hat{\mathbf{y}}_\mathbf{i} = f(\mathbf{x}_\mathbf{i})$ is the prediction of $\mathbf{x}_\mathbf{i}$, $|\mathbf{H}^\mathbf{r}|$ denotes the number of nonzero entries in $\mathbf{H}^\mathbf{r}$, and a_r is the hyperparameter to control the importance of the logical rule of the r-th partial-order relation. The core idea of this regularization term is to enforce the predictions of vertices that have a partial-order relation to be consistent with the corresponding rule. To be more specific, small values of $\mathcal{P}_0$ can be achieved if $q^r(\hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{j})$ is 1, meaning that $p^r(\mathbf{x}_\mathbf{i}, \mathbf{x}_\mathbf{j})$ is true (evidenced by $H^r_{ij} \neq 0$) and the rule $p^r(\mathbf{x}_\mathbf{i}, \mathbf{x}_\mathbf{j}) \longrightarrow q^r(f(\mathbf{x}_\mathbf{i}), f(\mathbf{x}_\mathbf{j}))$ is satisfied. However, this definition treats all vertex pairs of a partial-order relation equally, without considering their relative strengths. This assumption may decrease modelling fidelity for practical applications. To address this problem, we revise the regularizer as:

$$\mathcal{P}_1 = \sum_{r=1}^{R} \frac{a_r}{|\mathbf{H}^\mathbf{r}|} \sum_{\{i,j|H^r_{ij}\neq 0\}} (1 - q^r(\hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{j}))H^r_{ij}, \tag{10}$$

which incorporates H^r_{ij} as a coefficient to rescale the regularization strength of a vertex pair. With such a formulation, we enforce stronger partial-order regularization for vertex pairs that are connected by more hyperedges. Lastly, to get rid of the difficulties in discrete optimization, we replace the binary logic function q^r with a continuous function s^r that approximates it. Such approximation trick allows stable optimization and has been widely used in probabilistic soft logics [2]. This leads to the final version of our proposed partial-order regularizer:

$$\mathcal{P} = \sum_{r=1}^{R} \frac{a_r}{|\mathbf{H}^\mathbf{r}|} \sum_{\{i,j|H^r_{ij}\neq 0\}} s^r(\hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{j})H^r_{ij}, \tag{11}$$

where s^r is a self-defined function adjustable for different problems. For instance, s^r might be the subtraction between the predicted ranks $\hat{\mathbf{y}}_\mathbf{i} - \hat{\mathbf{y}}_\mathbf{j}$ in a ranking problem, while in a classification problem, s^r could be the gap between the predicted probabilities on a specific class.

3.2.1 *Optimization.*

By minimizing the objective function of Equation (8), we can achieve the prediction function that is smooth over the hypergraph and satisfies the logical rules on partial-order relations. Note that the regularization terms $\mathcal{L}$ and $\mathcal{P}$ operate on the predicted label space only and do not introduce extra model parameters. As such, the only model parameters to learn come from the predictive model $f(\mathbf{x})$. Given that f is a differentiable function (*e.g.,* logistic regression and neural networks), we can optimize the objective function with standard gradient-based methods, such as the stochastic (or batch) gradient descent [12]. Moreover, one can also directly learn $f(\mathbf{x})$ without specifying an explicit form of the predictive model. This will make the prediction function comply with the regularization to the maximum degree. We will empirically study how would this affect the prediction performance for downstream applications in Section 6.

3.3 Time Complexity Analysis

In this subsection, we analyze the time complexity of POH learning by comparing with conventional hypergraphs. As discussed in [21] and Section 2.2, the computational complexity of conventional hypergraph learning methods are $O(J)$, where J denotes the number of nonzero entries in the sparse hypergraph Laplacian matrix $\mathbf{L}$. In contrast, the additional computational cost of our POH learning comes from the construction of the partial incidence matrices $\{\mathbf{H^r} | r = 1, 2, \cdots, R\}$ and the partial-order regularization term $\mathcal{P}$. To compute $\mathbf{H^r}$, we need to obtain the co-occurrence matrix $\mathbf{C}$ first, and then evaluate the element-wise product $\mathbf{C} \odot \mathbf{P^r}$. For $\mathbf{C}$, we can achieve it by traversing all nonzero entries on the incidence matrix $\mathbf{H}$, for which the complexity is $O(J)$. Then for each nonzero element C_{ij} in $\mathbf{C}$, we check whether $\mathbf{p}^r(\mathbf{x}_i, \mathbf{x}_j)$ is true or not to obtain $\mathbf{C} \odot \mathbf{P^r}$. As such, the complexity of constructing a $\mathbf{H^r}$ is $O(J)$, and the overall complexity of constructing all R partial incidence matrices is $O(RJ)$. Similarly, the computation of $\mathcal{P}$ can be done in $O(RJ)$ time. In a real-world application, the number of partial-order relations R is typically a small number, since we need to account for the most prominent numerical or graded categorical features only. As such, the overall time complexity of our POH learning is essentially $O(J)$, which is the same as that of conventional hypergraph learning.

4 RELATED WORK

Our work is directly related to the recent work on hypergraphs, which can be seperated into two main categories: hypergraph construction and hypergraph utilization. Since proposed in [48], researchers have paid lots of attention on how to construct hypergraphs. For instances, Wang et al. leveraged a sparse representation of the entity feature space to generate hyperedges and learn the relationship among hyperedges and the connected vertices[38]. Instead of simply learning a sparse representation, Liu et al. employed an elastic net to regulate the representation learing [24]. Besides, Feng et al. jointly learned hypergraph representations of multiple hypergraphs by further encouraging the consitancy among different hypergraph representations [10]. However, none of the aforementioned work is able to incorporate the partial-order relations among entities that exist in graded categorical features and numerical features during the construction of hypergraphs. They thus lead to severe information loss and limit the expressiveness of the constructed hypergraphs.

Besides, hypergraph and hypergraph-based learning have been widely applied on many machine learning tasks, including clustering, embedding, ranking, semi-supervised classification and regression [3, 22, 27, 35, 36, 43]. For instance, the authors of [22] constructed a hypergraph to represent the correlations among news readers, news articles, topics and name entities and then ranked the news articles on the hypergraph to make recommendation for readers. In [3], the authors utilized a hypergraph to represent the relations among candidate sentences and made a graph-based extractive document summarization. Yoshida and Yuichi used a hypergraph to estimate the betweenness centrality and importance of vertices [42]. Huang et al. constrcuted a hypergraph of images and predicted the attributes of the images with hypergraph-based label propagation [19]. Tran *et al.*built a hypergraph where vertices and hyperedges respectively represent features and training samples to represent the sparse pattern of the training data [35]. Hmimida and Kanawati depicted the relation among social users, resources, and the tags of resources assigned by the users. They then employed hypergraph-based ranking to recommend candidate tags for resources [18]. These articles indicated the popularity and usability of hypergraphs and learning on hypergraphs. However they only utilized conventional hypergraphs instead of simultaneously improving the representation ability of the conventional hypergraphs and the corresponding learning methods.

5 UNIVERSITY RANKING

Following the previous work [9], we formulated university ranking as an unsupervised ranking (*i.e.,* re-ranking) problem. Given N universities with a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ and a historical ranking result $\mathbf{y} \in \mathbb{R}^N$, the target is to learn a new ranking $\mathbf{f} \in \mathbb{R}^N$. To solve the problem, we manually selected several partial-order relations and constructed a partial-order hypergraph (POH) to represent the given universities. Upon the constructed POH, we learned $\mathbf{f}$ by minimizing a ranking instantiation of the POH learning objective function. Specifically, we set the loss term in Equation (8) as $\mathcal{L} = \|\mathbf{y} - \mathbf{f}\|_F^2$, which encourages the learned ranking to be consistent and smooth with the historical one. Besides, we set the soft logic functions $\{s^r | r = 1, 2, \cdots, R\}$ as $s^r(f_i, f_j) = f_i - f_j$, *i.e.,* university i is encouraged to be ranked ahead of university j if the two universities have the selected partial-order relations. By specifying the above application-specific terms, we derived the objective function for the task,

$$\Gamma = \mathbf{f}^T \mathbf{L} \mathbf{f} + \lambda \|\mathbf{f} - \mathbf{y}\|_F^2 + \beta \sum_{r=1}^{R} \frac{a_r}{|\mathbf{H^r}|} \sum_{\{i,j | H_{ij}^r \neq 0\}} ReLU((f_i - f_j) H_{ij}^r), \quad (12)$$

where we further used the rectifier function (ReLU) [13] on the partial-order regularization term, so as to guarantee the objective function to be non-negative for stable optimization.

5.1 Experiment Settings

5.1.1 Dataset. To investigate the effectiveness of the proposed method, we employed the dataset of Chinese university ranking collected by [9]. This dataset contains 743 Chinese universities with data collected between January 1st, 2015 and May 1st, 2016. For each university, Web data from five channels were collected, including official, mass media, academic, employment, and general user channels. The official channel contains the primary information of a university, such as student quality, official activities, and development plans. In the mass media channel, they collected news reports mentioning the given university from mass media. The academic and employment channels contain university's academic status and graduate students employment statistics, respectively. The general user channel contains public impressions, attitudes, and sentiment polarities of universities shared in social media posts. To describe the universities, we also employed the rich set of features extracted by the authors of [9]. Among the 743 universities, we selected 438 top-tier ones in China[1] since they have more academic and research activities. Towards the historical ranking result $\mathbf{y}$, we merged the ranking results in 2015 from four well-known ranking systems of Chinese universities, namely, CUAA, WSL, WH, and iPIN[2]. The ranking ground-truth is generated in the same way, but based on the rankings of the four systems in the year 2016. As such, the task can be understood as using the past year's ranking and this year's features to predict the ranking of universities in this year. We normalized the constructed historical ranking result and the ground-truth into range $[0, 1]$ by scaling them with $1/N$.

5.1.2 Evaluation. We performed 5-fold cross-validation, employing three metrics to evaluate the ranking performance: mean absolute error (MAE) [41], Kendall's tau (Tau) [28], and Spearman's rank[3] (Rho) [34]. The three metrics have been widely used to evaluate pointwise, pairwise, and listwise ranking methods [25]. Note that better performance is evidenced by smaller MAE, larger Tau and Rho scores. Moreover, we carried out the student's t-test and reported the p-values where necessary.

5.1.3 Methods. We compared with following baselines[4]:

- **Simple Graph** [49]: It first constructs a simple graph to represent the universities, where the edge weight between two vertices is evaluated using the RBF kernel. We set the radius parameter σ as the median of the Euclidean distances of all pairs. The method then calculates the Laplacian matrix $\mathbf{L}$, learning $\mathbf{f}$ by minimizing the objective function $\mathbf{f}\mathbf{L}^T\mathbf{f} + \lambda\|\mathbf{y} - \mathbf{f}\|^2$. We experimented with different values of λ and reported the best performance.
- **Hypergraph** [3]: It first calculates the similarities between universities, and then constructs the hypergraph using neighbor-based methods. Specifically, the i-th hyperedge connects the k universities that are most similar to university i. The learning of $\mathbf{f}$ is performed by minimizing Equation (3). We tuned the two hyperparameters k and λ.
- **GMR** [9]: This is the state-of-the-art method for the university ranking task. It builds a simple graph from the features of each channel, modelling the relations between channels to reach a consensus ranking on all simple graphs. We used the same hyperparameter settings as reported in their paper.

We evaluate several POH methods that incorporate different partial-order relations on the same hypergraph structure of **Hypergraph**:

- **POH-Salary**: This method considers the partial-order relation on the salary feature. We encoded the logical rule $salary_>(\mathbf{x_i}, \mathbf{x_j}) \rightarrow rank_<(\mathbf{x_i}, \mathbf{x_j})$, meaning that $\mathbf{x_i}$ tends to be ranked higher than $\mathbf{x_j}$ if the salary feature of $\mathbf{x_i}$ is higher than that of $\mathbf{x_j}$.

[1]In China, universities were officially separated into three tiers by Ministry of Education (https://tinyurl.com/moe-univ-list/.).

[2] CUAA: http://www.cuaa.net/cur/. WSL: http://edu.sina.com.cn/gaokao/wushulian/. WH: http://www.nseac.com/html/168/. iPIN: https://www.wmzy.com/api/rank/schList/.

[3]Note that we omitted listwise ranking evaluation metrics like Precision@K [6], Recall@K [39], and NDCG@K [29] since they are sensitive to the selection of K

[4]Note that we omitted the comparison with **TMALL** and **GCN** mentioned in Section 6.1.3. Because [9] has shown that **TMALL** (similar to the **JL** baseline in their paper) is less effective than **GMR**; **GCN** is not fit for this unsupervised ranking task as it is designed for semi-supervised and supervised tasks [21]

Table 1: Performance comparison on university ranking.

Methods	MAE	Tau	Rho
Simple Graph	0.074 ± 9e-3	0.870 ± 2e-2	0.970 ± 8e-3
Hypergraph	0.067 ± 7e-3	0.876 ± 9e-3	0.974 ± 5e-3
GMR	0.065 ± 7e-3	0.871 ± 3e-2	0.970 ± 1e-2
POH-Salary	0.054 ± 1e-2*	0.892 ± 1e-2*	0.979 ± 5e-3*
POH-NCEE	0.055 ± 1e-2*	0.893 ± 9e-3*	0.978 ± 5e-3*
POH-All	**0.053 ± 1e-2***	**0.898 ± 1e-2****	**0.980 ± 6e-3****

$*$ and $**$ denote that the corresponding performance is significantly better (p-value < 0.05) than all baselines and all other methods, respectively.

- **POH-NCEE**: This method considers the partial-order relation on the NCEE feature, which stands for a university's admission requirement on the score of National College Entrance Examination. The logical rule to be encoded is naturally $NCEE_>(\mathbf{x_i}, \mathbf{x_j}) \rightarrow rank_<(\mathbf{x_i}, \mathbf{x_j})$, meaning universities with a higher NCEE score tend to have a better quality.
- **POH-All**: In this method, we model both partial-order relations as encoded in **POH-Salary** and **POH-NCEE**. We set the importance hyperparameters for the regularizers of the two rules as a_1 and $1 - a_1$, respectively.

5.1.4 Hyperparameter Tuning. We employed grid search to select the optimal hyperparameters for POH methods based on the results of Tau. The optimal hyperparameter setting and implementation of the compared methods can be publicly accessed[5]. For **POH-Salary** and **POH-NCEE**, we tuned one implicit (k) and two explicit hyperparameters (λ and β). To validate the strength of the proposed POH over traditional hypergraph, we set k and λ as the optimal ones of the baseline **Hypergraph**, and then searched β in the range of $[1e\text{-}4, 1e1]$. For **POH-All**, we tuned one more hyperparameter a_1, which controls the importance of logical rules and is in the range of $[0, 1]$.

Note that we have intentionally fixed k and λ to the optimal ones of **Hypergraph**, which also simplifies the tuning process. Further tuning k and λ based on the performance of POH methods can lead to even better performance (see Figure 4). Figure 3 shows the performance of **POH-All** *w.r.t.* β and a_1. This was accomplished by varying one parameter and fixing the other to the optimal value. As can be seen, our method is rather insensitive to hyperparameters around their optimal settings.

5.2 Experiment Results

5.2.1 Method Comparison. Table 1 summarizes the performance comparison on university ranking, from which we have the following observations: (1) **Hypergraph** performs better than **Simple Graph**, which verifies that considering the higher-order relations among universities is effective for the ranking task. (2) All POH-based methods outperform baselines by a large margin (*e.g.,* **POH-All** ourperforms **GMR** with an improvement of 18.46%, 3.10%, and 1.03% *w.r.t.* MAE, Tau, and Rho, respectively). It demonstrates the effectiveness of our proposed POH and regularized learning in integrating partial-order relations. (3) **POH-All** outperforms both **POH-Salary** and **POH-NCEE**. It further verifies the advantage of POH-based learning methods and reflects that jointly modelling multiple partial-order relations and rules is helpful. (4) The p-values

[5]https://github.com/hennande/Partial_Order_Hypergraph

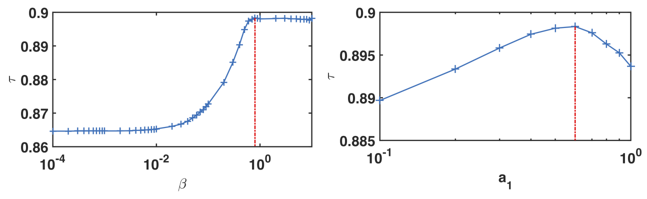


Figure 3: Procedure of tuning β and a_1 for POH-All. The red dotted line marked the optimal settings

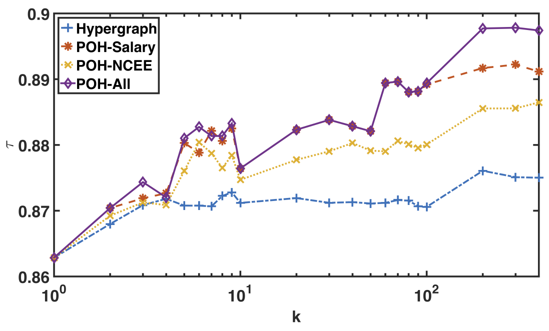


Figure 4: Performance comparison on Tau of Hypergraph and POH-based methods *w.r.t.* different k.

of student's t-test between POH-based methods and all the other methods are smaller than 0.05, indicating the significance of the performance improvements.

As we constructed hypergraphs by connecting a vertex with its k-nearest vertices, larger k makes the POH-based methods consider more vertex pairs with the given partial-order relations (these pairs would be eliminated if the vertex pair is not connected by any hyperedge). It is thus interesting to see how does the setting of k impact the performance of POH learning. Figure 4 shows the performance of Tau of **Hypergraph** and our POH-based methods on different k. Note that other hyperparameters have been fairly tuned for each setting of k. As can be seen, all POH-based methods outperform the **Hypergraph** on all settings. It demonstrates that the proposed POH learning consistently outperforms the conventional hypergraph, regardless of the underlying hypergraph structure. Moreover, all POH-based methods achieve performances better than those reported in Table 1, which shows the performance of POH methods on the optimal k of **Hypergraph** only. It reveals the potential of POH-based methods on further improvements if a better hyperparameter tuning strategy is applied.

5.2.2 Result Analysis. To understand the results better, we performed finer-grained error analysis. Given the result generated by a method, we generated an array of rank positions (integers from 1 to N) for universities, computing the absolute error on the rank position on each university.

Figure 5 depicts the distribution of the absolute rank errors as a boxplot. As can be seen, the rank error distribution of POH-based methods is more dense and centralizes at smaller medians than that of **Simple Graph** and **Hypergraph**. It provides sufficient evidence on the better ranking generated by the POH-based methods. Moreover, we find that **Simple Graph** and **Hypergraph** make errors larger than 5 on about 25% of the universities, which is rarely

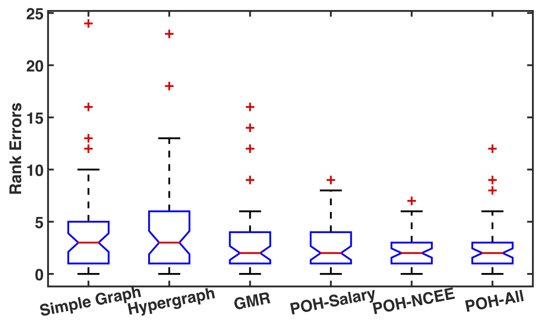


Figure 5: Distribution of absolute rank errors.

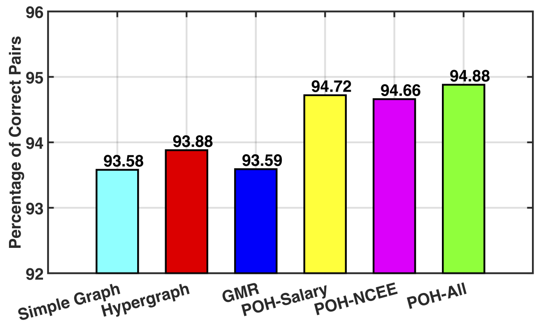


Figure 6: Percentage of correctly ranked university pairs.

seen from POH-based methods. Meanwhile, the largest error made by **Simple Graph** and **Hypergraph** is almost two times that of POH-based methods. These results demonstrate that POH-based methods are more robust, thus being more applicable in real-world applications. Among the baselines, **GMR** achieves the smallest rank error, which is comparable with **POH-Salary**. This signifies the usefulness of modelling the relations among data from different channels, which could be a future direction to be explored by POH-based methods.

Besides the investigation on pointwise rank errors, we further performed an analysis on pairwise ranks. For each method, we counted the number of university pairs that are ranked correctly, and drew the percentage of correct pairs in Figure 6. As shown, POH-based methods manage to generate ranks with correct order on 1% more university pairs than **Simple Graph**, **Hypergraph**, and **GMR**, further demonstrating the accuracy and advantage of POH-based methods. Considering that there are more than 80,000 university pairs, an improvement of 1% (correctly ranking 800+ pairs) is a significant improvement.

Finally, we studied whether the ranking generated by our method **POH-All** is consistent with the four popular Chinese university ranking systems of year 2016 (*cf.* Section 5.1.1). We evaluated the pairwise correlation (Tau) and listwise correlation (Rho) between the ranked lists of the five ranking systems (**POH-All** and the four existing systems). The results are shown in Table 2. We can see that our method achieves rather high correlations with existing ranking systems, and it correlates most with WH. This shows that our result is relatively consistent with these manually devised ranking systems, implying that our POH-based re-ranking method shall be acceptable by the general readers.

Table 2: Pairwise and listwise correlations between the ranking results of POH-All and four Chinese university ranking systems. Entries in bold denote the most correlated result to the method of the corresponding column.

	Tau					Rho				
	CUAA	**WSL**	**WH**	**iPIN**	**POH**	**CUAA**	**WSL**	**WH**	**iPIN**	**POH**
CUAA	-	0.763	0.778	0.431	0.765	-	0.903	0.903	0.573	0.886
WSL	0.763	-	0.823	0.459	0.806	**0.903**	-	0.950	0.627	0.943
WH	**0.778**	**0.823**	-	0.469	**0.832**	**0.903**	**0.950**	-	0.645	**0.961**
iPIN	0.431	0.459	0.469	-	0.569	0.573	0.627	0.645	-	0.750
POH	0.765	0.806	0.832	**0.569**	-	0.886	0.943	**0.961**	**0.750**	-

6 POPULARITY PREDICTION

Predicting the popularity of online content is a hot research topic in social media mining and has varying problem statements [16, 33]. Following the recent work on micro-video popularity prediction [5], we formulated the task as a semi-supervised regression problem. Given $N + U$ items with a feature matrix $\mathbf{X} \in \mathbb{R}^{(N+U)\times M}$ and the ground-truth popularity of the N items $\mathbf{y} \in \mathbb{R}^N$, the objective is to learn a function $\hat{y_i} = f(\mathbf{x_i})$ that maps an item from the feature space to the popularity space. To solve the problem, we first constructed a POH with partial-order relations on some important numerical features (detailed later in experiments). We then derived an instantiation of the general framework Equation (8) for the semi-supervised regression task as follows:

$$\Gamma = \hat{\mathbf{y}}^T\mathbf{L}\hat{\mathbf{y}} + \lambda\sum_{i=1}^{N}(\hat{y_i} - y_i)^2 + \beta\sum_{r=1}^{R}\frac{a_r}{|\mathbf{H^r}|}\sum_{\{i,j|H_{ij}^r\neq 0\}} ReLU((\hat{y_j} - \hat{y_i})H_{ij}^r), \tag{13}$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \cdots, \hat{y}_N, \hat{y}_{N+1}, \cdots, \hat{y}_{N+U}] \in \mathbb{R}^{N+U}$, denoting the prediction of all items (both with labels and without labels).

6.1 Experiment Settings

6.1.1 Dataset. We employed the same dataset as [5] for experiments. It contains 9,719 micro-videos collected from Vine[6], posted between July 1st and October 1st, 2015. Each micro-video has visual, audio, and textual contents, as well as the profile of the user who posted it. With these data, the authors of [5] extracted a rich set of popularity-oriented features, such as user activities, object distribution, aesthetic description, sentence embedding, and textual sentiment polarity, to represent a micro-video. To measure the popularity of a micro-video, they employed four popularity-related indicators, namely, the number of comments (*n_comments*), the number of likes (*n_likes*), the number of reposts (*n_reposts*), and the number of loops (*n_loops*); the four indicators were averagely fused ((*n_comments* + *n_likes* + *n_reposts* + *n_loops*)/4) as the popularity ground-truth for a micro-video.

6.1.2 Evaluation. We performed 10-fold cross-validation and evaluated the performance in terms of three metrics. From the regression perspective, we followed the previous work [5] and employed normalized mean square error (nMSE). Meanwhile, we utilized two ranking-oriented metrics, Tau and Rho correlation coefficients. Besides, we carried out the student's t-test and reported the p-values where necessary.

6.1.3 Methods. We compare with following baselines:

- **Simple Graph** [49]: We applied the same setting as the **Simple Graph** described in Section 5.1.3.
- **Hypergraph** [3]: We also adopted the same setting as the **Hypergraph** in Section 5.1.3.
- **TMALL** [5]: This method first calculates a simple graph Laplacian matrix with features from each modality (visual, audio, *etc.*). It then learns a common space Laplacian matrix by considering the relations among different modalities and fusing the corresponding graph Laplacian matrices. It finally performs a simple graph learning like **Simple Graph** on the common Laplacian matrix. We followed the settings as reported in their paper.
- **GCN** [21]: This is the state-of-the-art graph learning method by using graph convolutional neural networks. We replaced the log loss term in their implementation with the same mean squared loss in Equation (13) for a fair comparison. We carefully tuned four hyperparameters, namely, learning rate, dropout ratio, l_2-norm weight and hidden layer size.
- **LR-HG**: This method is similar to **Hypergraph**. Instead of directly learning $\hat{y}$, we parameterized it as a linear regression (LR) model on features. The optimization process learns the parameters of LR, which is then used to predict $\hat{y}$.

We evaluated several POH methods on the same hypergraph structure of **Hypergraph**:

- **POH-Follow**: This method considers a partial-order relation on the follower feature (*i.e.,* the number of followers of the user who posted the video). It encodes the logical rule $followers_>(\mathbf{x_i}, \mathbf{x_j}) \rightarrow popularity_>(\mathbf{x_i}, \mathbf{x_j})$, meaning that $\mathbf{x_i}$ would be more popular than $\mathbf{x_j}$ if the user of $\mathbf{x_i}$ has more followers than that of $\mathbf{x_j}$.
- **POH-Loop**: This method has the same setting as **POH-Follow**, besides that it encodes another partial-order relation on the loop feature (*i.e.,* total number of views of all videos posted by a user).
- **POH-All**: This method jointly encodes the two partial-order relations in **POH-Follow** and **POH-Loop**. We set the corresponding rule importance hyperparameters as a_1 and $1 - a_1$, respectively.
- **LR-POH**: Similar to **LR-HG**, this method parameterizes the $\hat{y}$ of **POH-All** as a linear regression model on input features.

6.1.4 Hyperparameter Tuning. We employed the same procedure as described in Section 5.1.4 to tune the hyperparameters of POH methods. Optimal hyperparameter settings of each compared method will be released together with their codes. We investigated the sensitivity of our proposed POH-based methods by taking **POH-All** as an example. Figure 7 illustrates the performance of **POH-All** while varying one hyperparameter and fixing the others with optimal values. Again, the results demonstrate that our model is not sensitive to the parameters around their optimal settings.

[6]https://vine.co/.

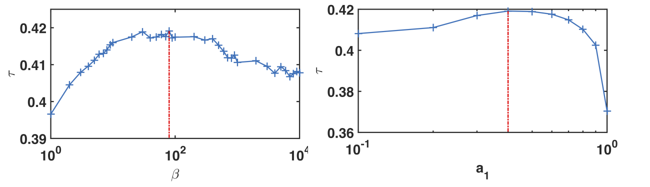


Figure 7: Procedure of tuning β and a_1 for POH-All. The red dotted line marks the optimal settings.

Table 3: Performance comparison on popularity prediction.

Methods	nMSE	Tau	Rho
Simple Graph	0.999 ± 1e-3	0.137 ± 2e-2	0.200 ± 2e-2
Hypergraph	1.000 ± 4e-5	0.165 ± 3e-2	0.240 ± 4e-2
TMALL[7]	0.979 ± 9e-3	-	-
POH-Follow	1.000 ± 4e-4	0.393 ± 3e-2*	0.562 ± 3e-2*
POH-Loop	0.997 ± 2e-3	0.376 ± 2e-2*	0.540 ± 3e-2*
POH-All	0.989 ± 9e-3	**0.419 ± 2e-2****	**0.592 ± 3e-2****
GCN	0.919 ± 6e-2	0.171 ± 2e-2	0.252 ± 3e-2
LR-HG	0.846 ± 1e-1*	0.117 ± 2e-2	0.182 ± 3e-2
LR-POH	**0.724 ± 2e-1****	0.350 ± 2e-2*	0.496 ± 3e-2*

∗ and ∗∗ denote that the corresponding performance is significantly better (p-value < 0.05) than all baselines and all other methods, respectively.

6.2 Experiment Results

6.2.1 Method Comparison. We first investigated the effectiveness of the proposed methods. Table 3 shows the performance of all the compared methods. We have the following findings:
(1) **Hypergraph** outperforms **Simple Graph** *w.r.t.* Tau and Rho, although they achieve the same performance level on nMSE. It verifies that considering the higher-order relations among videos leads to popularity prediction with more accurate relative orders.
(2) **POH-Follow** and **POH-Loop** further surpass **Hypergraph** with an average improvement of 133.03% and 129.58% on the pairwise and listwise ranking metrics; meanwhile, slight improvement is obtained on the pointwise regression metric nMSE. This indicates that considering meaningful partial-order relations is particularly helpful for better predicting the relative order of the videos.
(3) **POH-All** outperforms **POH-Follow** and **POH-Loop** with a significant average improvement on Tau (+8.97%) and Rho (+7.44%) as well as a slight improvement on nMSE. It validates that jointly considering multiple partial-order relations is useful.
(4) Comparing **Hypergraph** with **LR-HG**, we can see that better nMSE can be achieved by using LR as the predictive model, but the two ranking metrics become worse. The same situation can be observed for **POH-All** and **LR-POH**. This provides evidence that using a sophisticated model can better fit the labels and help to minimize the regression loss, however, the ranking performance may not be necessarily improved. The same finding has been observed before in popularity prediction [16] and another orthogonal application of item recommendation [7]. In our case of graph-based learning, the regularizers (for smoothness and partial-order rules) carry strong signals for learning the relative orders between vertices. However, the regularization effects might be weakened when a specialized model is used to fit the label in the meantime. We leave more detailed exploration of this hypothesis as future work.
(5) **GCN** outperforms **POH-All** *w.r.t.* nMSE, while Tau and Rho indicate that its ranking performance is worse. The lower nMSE of **GCN** can be credit to the strong representation power of the underlying neural network, which can fit the labels well. However, **GCN** may overfit the data and fail to predict the popularity ranking well without regularization on the relative orders of vertices.
(6) **LR-POH** achieves the best performance with significantly better nMSE than all the other compared methods as well as tremendously better Tau and Rho than all the baseline methods. This further demonstrates the effectiveness of our proposed POH learning.

[7]This result is directly copied from the corresponding paper since we exactly followed their experimental settings.

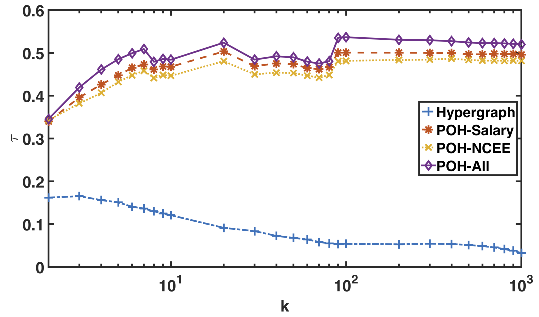


Figure 8: Performance comparison on Tau of Hypergraph and POH-based methods *w.r.t.* different k.

We further studied whether the performance improvements of the proposed POH-based methods are consistent under different hypergraph settings. We compared the optimal performance of **Hypergraph**, **POH-Follow**, **POH-Loop**, and **POH-All** under different values of k, which controls the number of videos connected by a hyperedge. As illustrated in Figure 8, all POH-based methods outperform the **Hypergraph** under all the values of k by a large margin. It is worth noting that the optimal performance of POH methods are better than that shown in Table 3 (Table 3 shows the results of POH on the optimal setting of **Hypergraph**). This is consistent with the university ranking task, which implies the potential of further improving POH learning with a better hyperparameter tuning strategy.

7 CONCLUSIONS

In this paper, we proposed a novel partial-order hypergraph that improves conventional hypergraphs by encoding the partial-order relations among vertices. We then generalized existing graph-based learning methods to partial-order hypergraphs by integrating the second-order logic rules that encode the partial-order relations; moreover, the time complexity of learning remains unchanged. Experimental results on university ranking and video popularity prediction demonstrate the effectiveness of our proposed methods.

In future, we will explore the proposed POH to address more graph-based applications. Besides, we will further improve POH learning by replacing the linear prediction function from feature to label spaces with the advanced deep neural networks. Moreover, we plan to optimize the spatial complexity of POH with discrete hashing techniques like [46]. Furthermore, we would like to investigate the automatic extraction of partial-order relations and logical rules to construct POH.

REFERENCES

[1] Charu C Aggarwal and Chandan K Reddy. 2013. *Data clustering: algorithms and applications.* CRC press.

[2] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research* (2017).

[3] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2014. Multi-document hyperedge-based ranking for text summarization. In *CIKM.* 1919–1922.

[4] Chen Chen, Cewu Lu, Qixing Huang, Qiang Yang, Dimitrios Gunopulos, and Leonidas Guibas. 2016. City-Scale Map Creation and Updating Using GPS Collections. In *SIGKDD.* 1465–1474.

[5] Jingyuan Chen, Xuemeng Song, Liqiang Nie, Xiang Wang, Hanwang Zhang, and Tat-Seng Chua. 2016. Micro tells macro: predicting the popularity of micro-videos via a transductive model. In *MM.* 898–907.

[6] Zhiyong Cheng and Jialie Shen. 2016. On effective location-aware music recommendation. *Transactions on Information System* 34, 2 (2016), 13.

[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *RecSys.* 39–46.

[8] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. In *SIGKDD.* 551–556.

[9] Fuli Feng, Liqiang Nie, Xiang Wang, Richang Hong, and Tat-Seng Chua. 2017. Computational social indicators: a case study of chinese university ranking. In *SIGIR.* 455–464.

[10] Xiaodong Feng, Sen Wu, and Wenjun Zhou. 2017. Multi-Hypergraph Consistent Sparse Coding. *Transactions on Intelligent Systems and Technology* 8, 6 (2017), 75.

[11] David F Gleich and Michael W Mahoney. 2015. Using local spectral methods to robustify graph-based learning algorithms. In *SIGKDD.* 359–368.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning.* MIT press.

[13] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* 405, 6789 (2000), 947.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR.* 770–778.

[15] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR.* 355–364.

[16] Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014. Predicting the Popularity of Web 2.0 Items Based on User Comments. In *SIGIR.* 233–242.

[17] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. 2017. Birank: Towards ranking on bipartite graphs. *Transactions on Knowledge and Data Engineering* 29, 1 (2017), 57–71.

[18] Manel Hmimida and Rushed Kanawati. 2016. A Graph-Coarsening Approach for Tag Recommendation. In *WWW.* 43–44.

[19] Sheng Huang, Mohamed Elhoseiny, Ahmed Elgammal, and Dan Yang. 2015. Learning hypergraph-regularized attribute predictors. In *CVPR.* 409–417.

[20] Jyun-Yu Jiang, Pu-Jen Cheng, and Wei Wang. 2017. Open Source Repository Recommendation in Social Coding. In *SIGIR.* 1173–1176.

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

[22] Lei Li and Tao Li. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *WSDM.* 305–314.

[23] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *WWW.* 583–592.

[24] Qingshan Liu, Yubao Sun, Cantian Wang, Tongliang Liu, and Dacheng Tao. 2017. Elastic net hypergraph learning for image clustering and semi-supervised classification. *Transactions on Image Processing* 26, 1 (2017), 452–463.

[25] Tie-Yan Liu. 2011. *Learning to rank for information retrieval.* Springer Science & Business Media.

[26] Tao Mei, Yong Rui, Shipeng Li, and Qi Tian. 2014. Multimedia search reranking: A literature survey. *Comput. Surveys* 46, 3 (2014), 38.

[27] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. 2015. Scalable large near-clique detection in large-scale networks via sampling. In *SIGKDD.* 815–824.

[28] RB Nelsen. 2001. Kendall tau metric. *Encyclopaedia of Mathematics* 3 (2001), 226–227.

[29] Liqiang Nie, Meng Wang, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Oracle in image search: a content-based approach to performance prediction. *Transactions on Information System* 30, 2 (2012), 13.

[30] Liqiang Nie, Shuicheng Yan, Meng Wang, Richang Hong, and Tat-Seng Chua. 2012. Harvesting visual concepts for image search with complex queries. In *MM.* 59–68.

[31] Adi Omari, David Carmel, Oleg Rokhlenko, and Idan Szpektor. 2016. Novelty Based Ranking of Human Answers for Community Questions. In *SIGIR.* 215–224.

[32] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, and Jiawei Han. 2015. ClusType: Effective Entity Recognition and Typing by Relation Phrase-Based Clustering. In *SIGKDD.* 995–1004.

[33] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrian, Honglin Yu, and Pascal Van Hentenryck. 2017. Expecting to Be HIP: Hawkes Intensity Processes for Social Media Popularity. In *WWW.* 735–744.

[34] Charles Spearman. 1987. The proof and measurement of association between two things. *The American journal of psychology* 100 (1987), 441–471.

[35] Kenneth Tran, Saghar Hosseini, Lin Xiao, Thomas Finley, and Mikhail Bilenko. 2015. Scaling up stochastic dual coordinate ascent. In *SIGKDD.* 1185–1194.

[36] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *WWW.* 1451–1460.

[37] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. 2016. Scalable semi-supervised learning by efficient anchor graph regularization. *Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1864–1877.

[38] Meng Wang, Xueliang Liu, and Xindong Wu. 2015. Visual Classification by ℓ_1 -Hypergraph Modeling. *TKDE* 27, 9 (2015), 2564–2574.

[39] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item silk road: Recommending items from information domains to social users. In *SIGIR.* 185–194.

[40] Xiaoqian Wang, Feiping Nie, and Heng Huang. 2016. Structured Doubly Stochastic Matrix for Graph Based Clustering: Structured Doubly Stochastic Matrix. In *SIGKDD.* 1245–1254.

[41] Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research* 30, 1 (2005), 79–82.

[42] Yuichi Yoshida. 2014. Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches. In *SIGKDD.* 1416–1425.

[43] Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon. 2015. A scalable asynchronous distributed algorithm for topic modeling. In *WWW.* 1340–1350.

[44] Rose Yu, Huida Qiu, Zhen Wen, ChingYung Lin, and Yan Liu. 2016. A survey on social media anomaly detection. *SIGKDD* 18, 1 (2016), 1–14.

[45] Dongxiang Zhang, Long Guo, Xiangnan He, Jie Shao, Sai Wu, and Heng Tao Shen. 2018. A Graph-Theoretic Fusion Framework for Unsupervised Entity Resolution. In *ICDE.*

[46] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *SIGIR.* 325–334.

[47] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *NIPS.* 321–328.

[48] Denny Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2007. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS.* 1601–1608.

[49] Denny Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. 2004. Ranking on data manifolds. In *NIPS.* 169–176.