Classification of Edge-dependent Labels of Nodes in Hypergraphs (ONLINE APPENDIX)

Minyoung Choe
KAIST
minyoung.choe@kaist.ac.kr

Sunwoo Kim
KAIST
kswoo97@kaist.ac.kr

Jaemin Yoo
Carnegie Mellon University
jaeminyoo@cmu.edu

Kijung Shin
KAIST
kijungs@kaist.ac.kr

ABSTRACT

This document provides supplementary information to the main paper "Classification of Edge-Dependent Labels of Nodes in Hypergraphs."

ACM Reference Format:
Minyoung Choe, Sunwoo Kim, Jaemin Yoo, and Kijung Shin. 2023. Classification of Edge-dependent Labels of Nodes in Hypergraphs (ONLINE APPENDIX). In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3580305.3599274

1 DETAILS OF HNN

Since we implemented HNN[**?**] by ourselves based on the description, we clarify the step of updating embeddings. Let $\mathbf{H} \in \mathbb{R}^{N \times M}$ denote the incidence matrix where $\mathbf{H}_{ve} = 1$ iff the vertex v is in the hyperedge e and 0 otherwise. A diagonal node degree matrix is $\mathbf{D} = diag(\mathbf{H}\mathbf{1}_M)$ where $\mathbf{1}_M$ is a M sized vector of all ones, and $D(v)$ indicates the degree of the node v. Then a diagonal hyperedge degree matrix is $\hat{\mathbf{D}} = diag(\mathbf{H}^T\mathbf{1}_N)$ where $\mathbf{1}_N$ is a N sized vector of all ones, and $\hat{D}(e)$ indicates the size of the hyperedge e.

HNN uses the random walk node transition matrix $\mathbf{R} \in \mathbb{R}^{N \times N}$ and the random walk hyperedge transition matrix $\hat{\mathbf{R}} \in \mathbb{R}^{M \times M}$ in the update. They are defined as follows,

$$\mathbf{R} = \mathbf{H}\mathbf{D}_e^{-1}(\mathbf{D}^{-1}\mathbf{H})^T \tag{1}$$

$$\hat{\mathbf{R}} = (\mathbf{D}^{-1}\mathbf{H})^T\mathbf{H}\mathbf{D}_e^{-1} \tag{2}$$

Before defining the update step, Equation 3 initializes the node embedding matrix $\mathbf{Z}^{(1)}$ by the input $\mathbf{X}$ and then initializes hyperedge embeddings $\mathbf{Y}^{(1)}$ by $\mathbf{Z}^{(1)}$.

$$\mathbf{Z}^{(1)} = \mathbf{X}, \mathbf{Y}^{(1)} = (\mathbf{D}^{-1}H)^T\mathbf{Z}^{(1)} \tag{3}$$

In the update step, the node embedding $z^{(k)}$ is first updated and then the hyperedge embedding $y^{(k)}$ is updated at layer $k+1$.

$$\tilde{y}_{e_i}^{(k)} = \sum_{e_j \in} \frac{\hat{\mathbf{R}}_{e_i e_j}}{\hat{D}(e_j)} \sum_{v_j \in e_j} \frac{\psi([z_{v_j}^{(k)} y_{e_j}^{(k)}])}{D(v_j)} \tag{4}$$

$$z_v^{(k+1)} = \sigma(\frac{1}{D(v)} \sum_{e \in \{e: v \in e, e \in\}} (\tilde{y}_e^{(k)} + y_e^{(k)})\mathbf{W}^{(k)}) \tag{5}$$

$$\tilde{z}_{v_i}^{(k+1)} = \sum_{v_j \in} \frac{\mathbf{R}_{v_i v_j}}{D(v_j)} \sum_{e \in \{e: v_j \in e, e \in\}} \frac{y_e^{(k)}}{\hat{D}(e)} \tag{6}$$

$$y_e^{(k+1)} = \sigma(\frac{1}{\hat{D}(e)} \sum_{v \in e} (\tilde{z}_v^{(k+1)} + \hat{\psi}([z_v^{(k+1)} y_e^{(k)}]))\hat{\mathbf{W}}^{(k)}) \tag{7}$$

where a function ψ takes concatenated embeddings of the node and the hyperedge so that derives a hyperedge-dependent embedding for the corresponding pair. We implement the function ψ as a single layer perceptron and the non-linear activation function σ as a rectified linear unit (Relu).

2 DETAILED DESCRIPTION OF DOWNSTREAM TASKS FOR

2.1 Ranking Aggregation

Data. We use two datasets, (1) Halo game dataset and (2) AMiner dataset with authors' H-index. The Halo 2 game dataset [**?**] consists of 5,507 users and two kinds of matches: 31,028 free-for-all matches where up to 8 players can involve in a game, and 5,093 1-v-1 matches where two members play the game. We use free-for-all matches to train and evaluate an edge-dependent node label classifier, and 1-v-1 matches as the test data for the downstream task. We build a hypergraph by regarding players as nodes and matches as hyperedges. The players are classified into their contribution levels during each match. The other dataset is extracted from the Coauthorship-AMiner dataset as following steps: (a) extracting publications (hyperedges) which venue is successfully matched with the venues in Google Scholar [1] under the four categories: Computer Graphics, Computational Linguistics, Human Computer Interaction and Databases & Information Systems, (b) sampling 10,000 publications which compose one connected component among them. We use hypergraphs with authors'orders in publications (i.e., edge-dependent labels) to train, and evaluate the result of the downstream task with authors' H-Index. We see how much is it helpful to know the author's order in publications for estimating their H-Index.

Training. In the Halo 2 game dataset, the continuous scores that the users get in each match are binned, and we use the binning index to indicate the contribution level of each player in the match, i.e., edge-dependent node labels. In detail, we normalize the scores during each match and divide the normalized scores into five bins containing 1/5 (20%) of the scores. We train our model with free-for-all matches by splitting them into train/valid sets and predict all contribution levels in free-for-all matches. In the AMiner dataset, we train our model to accurately predict the order of authors by using train and valid sets, and predict all of them including test

[1] https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599274

sets. According to the predicted order of authors, we give the contribution level of the first and last author to 2 and others to 1 as following [**?**]. Using a well-performing randomwalk based downstream algorithm [**?**] on the input hypergraph, the overall rankings of nodes are estimated by the stationary distribution of the random walks on the hypergraphs combined with predicted contribution levels. In other words, the node with a higher contribution level is visited more frequently within the hyperedge during the random walks. In the evaluation, we predict the winner, i.e. higher ranked player or higher H-Indexed author, as the one with the highest stationary probability in the node pairwise comparison.

Results. The method based on is best among three competitors. Surprisingly, it even slightly outperforms that of a hypergraph with ground-truth scores in both datasets. This shows that binning could lead to a better performance. Regarding the AMiner dataset, we assume that would outperform if it finds out the co-first author since it is possible to assign more than one label for the first author in each hyperedge while the ground truth allows only one could be the first author in each hyperedge.

2.2 Clustering

Data. We used the datasets extracted from (1) Coauthorship-DBLP and (2) Coauthorship-AMiner where authors and publications are nodes and hyperedges, respectively. From the Coauthorship-DBLP, we select hyperedges that belong to only one category among computer science, data engineering, software engineering, and theory following the dataset description [**?**] and then we choose 1000 hyperedges among them that compose one connected component. For the Coauthorship-AMiner, we use the same as in the ranking aggregation task. In both datasets, edge-dependent node labels represent the orders of authors in each publication, and three labels are used, as described in Section **??**.

Training. We train to predict the order of authors in each publication. After predicting the orders of all authors in each publication, we assign two as weights to the first and last authors and one to the others, following [**?**]. Then we apply the clustering algorithm (RDC-Spec) proposed in [**?**] on the hypergraphs with predicted edge-dependent node weights. Specifically, the stationary distribution of random walks, which proceed in consideration of node weights, is used for spectral clustering.

Results. We compare the performance of RDC-Spec on three different hypergraphs, (a) which are with ground-truth, (b) predicted, and (c) no edge-dependent node labels, respectively. The performance with labels predicted by is better than that with no labels, while that with ground-truth labels is best. Thus, predicting the orders of authors in each publication (i.e., edge-dependent node labels) is beneficial in hypergraph clustering.

2.3 Product Return Prediction

Data. We use a synthetic dataset from [**?**] for the product return prediction task. It consists of 6,000 baskets and 10,000 products, which are nodes and hyperedges, respectively. The return rate of each product is assigned by the truncated normal distribution with mean 0.5 and standard deviation 1.0. The count of products in each basket is from 1 to 3 with the probabilities 60%, 30%, and 10%. The baskets are labeled based on the count of products in them.

Training. We directly use the count of products in each basket as edge-dependent node labels and train to predict them. We then run HyperGo [**?**] on the hypergraph with predicted counts.

Results. As in the previous tasks, we compare the result of HyperGo on three different hypergraphs, which are with ground-truth, predicted, and no counts of products in each basket. The hypergraph with unknown counts indicates a situation in which you only know if the product is in a basket without knowing if there exist duplicates or not. The method with the labels predicted by was second best, outperforming that without labels. This result demonstrates the usefulness of identifying edge-dependent node labels in the considered application.