HINGCN: Heterogeneous information network revisited with GCN

Danhao Ding*, Xiang Li*, Nikos Mamoulis†, Ben Kao*
*The University of Hong Kong, Pokfulam Road, Hong Kong
†University of Ioannina, Ioannina, Greece
*{dhding2, xli2, kao}@cs.hku.hk †nikos@uoi.gr

ABSTRACT

GCN on HIN.

KEYWORDS

Semi-supervised classification; graph convolution; heterogeneous information network

ACM Reference Format:
Danhao Ding*, Xiang Li*, Nikos Mamoulis†, Ben Kao*. 2018. HINGCN: Heterogeneous information network revisited with GCN. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3178876.3185993

1 INTRODUCTION

Dan:TODO

Our main contributions are:

• We proposed a heterogeneous graph convolution algorithm that is capable of capturing edge information.
• Dan:don't know
• We conduct extensive experiments to evaluate the performance of HINGCN against 9 other classification methods. Our results show that HINGCN performs very well against the competitors. In particular, it is very robust in that it consistently performs well over all the datasets tested. Also, it outperforms others by wide margins for datasets that are highly multi-scale.

The rest of the paper is organized as follows. Section 2 mentions related works on heterogeneous graph neural networks, graph embedding and described several semi-supervised classification algorithms. Section 4 presents the HINGCN algorithm. Section 5 describes the experiments and presents experimental results. Finally, Section 6 concludes the paper.

2 RELATED WORK

2.1 Heterogeneous graph neural networks

Dan:TODO

2.2 Heterogeneous graph embedding

Dan:TODO

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04..
https://doi.org/10.1145/3178876.3185993

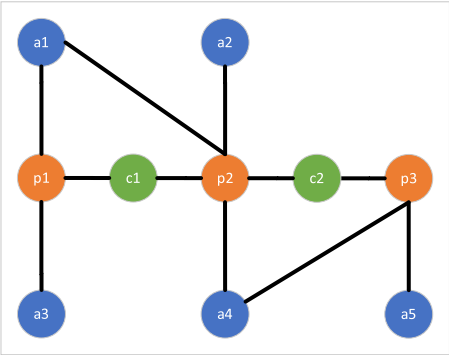

Figure 1: An example of a heterogeneous information network (DBLP). There are three types of nodes (i.e. author, paper, conference).

3 PRELIMINARY

Definition 1. **Heterogeneous Information Networks (HIN)**[5]. Let $T = \{T_1, \ldots, T_m\}$ be a set of m object types. For each type T_i, let n_i and $X_i = x_i 1, \ldots, x_{in_i}$ be the number and the set of objects of type Ti, respectively. An HIN is a graph $G = (V, E)$, where $V = \bigcup_{i=1}^{m} X_i$, and E is a set of links, each represents a binary relation between two objects in V. If $m = 1$ (i.e., there is only one object type), G reduces to a homogeneous information network.

Example 1. *In Fig. 1, we give an example of HIN. There are three types of nodes, Author, Paper and Conference. And we illustrate two types of edges: Author-Paper, Paper-Conference.*

Definition 2. **Metapath**[8]. A meta-path Φ is a path defined on a graph schema. Meta-path Φ: $T_1 \xrightarrow{R_1} \cdots \xrightarrow{R_l} T_{l+1}$ defines a composite relation $R = R_1 \circ \cdots \circ R_l$ that relates objects of type T_1 to objects of type T_{l+1}. We say a path $p = (x_1 x_2 \ldots x_{l+1})$ between x_1 and x_{l+1} in network G follows the meta-path Φ, if each object is of type T_i and each link $ei = \langle x_i x_{i+1} \rangle$ belongs to each relation R_i in Φ. We say that $p_{x_1 \rightsquigarrow x_{l+1}}$ is an instance of meta-path Φ, denoted by $p_{x_1 \rightsquigarrow x_{l+1}} \vdash \Phi$. Moreover, in this paper we say x_v is a meta-path Φ related neighbor of x_u, denoted by $x_v \in N^{\Phi}(x_u)$

Example 2. *In Fig. 1, a_3 is an APA-related neighbor of a_1, and a_5 is an APCPA-related neighbor of a_1.*

The notations throughout the rest of paper are shown in Table 1.

Table 1: Description of symbols

Symbol	Description
Φ	Meta-path
N^{Φ}	Set of neighbors with respect to a given meta-path Φ
$h^{(t)}$	Node embedding at layer t; $h^{(0)}$ is input node feature
$r^{(t)}$	Edge embedding at layer t; $r^{(0)}$ is input edge feature
$a_{i,j}^{\Phi}$	Attention of node pair (i,j) under meta-path Φ
$w_i^{\Phi_p}$	Weight of meta-path Φ_p related component of w_i
z_i	Output embedding for node i
W_F	Learnable weight matrix of fully connected layer F
σ	Sigmoid function
$\oplus$	Concatenate function
$\odot$	Hadamard (element-wise) product function

4 ALGORITHM

In this section we describe our HINGCN algorithm. Figure **??**(c) shows a flow diagram of HINGCN. Dan:TODO:signposting

4.1 Edge features as supplementary information

According to research findings of Xu et al. [9], discriminative power of a graph neural network determines its representation capability. We claim that information is loss if we only consider adjacency of target type vertices. Let's consider again the example shown in Fig. 1. In long tailed meta-path *APCPA*, a shrinking *A*-*A* adjacency loses information from intermediate *PCP* nodes. And a graph neural network using only adjacency information cannot distinguish between two instances $p_1 = (a_1p_1c_1p_2a_4)$ and $p_2 = (a_1p_2c_2p_3a_4)$. Instead, both instances are represented as edge $\langle a_1a_4\rangle$. Therefore we propose to improve expressivity of our network by adding additional edge feature to homogeneous *A*-*A* edges. These edge features are designed to effectly distinguish different intermediate nodes between meta-path instances. We propose edge features that is composed of three different components, namely path count, intermediate node embeddings and PathSim[8].

•**Path Count**. Different object pairs have different numbers of path instance between them and these numbers may reflect object similarity under a given meta-path. We propose to record count of path between objects x_u and x_v under meta-path Φ by:

$$c(x_u, x_v) = |\{p_{x_u \rightsquigarrow x_v} : p_{x_u \rightsquigarrow x_v} \vdash \Phi\}|$$

•**Intermediate object embedding**. To discriminate between different intermediate objects, we propose to use a sum of pre-trained node embedding as a component of edge feature. This node embedding can be trained from heterogeneous embedding algorithms such as [1, 2]. Given a meta-path instance $p = (x_1x_2 \dots x_{l+1})$, we calculate

$$o(x_1, x_{l+1}) = \frac{1}{(l+1)\cdot c(x_1, x_{l+1})} \sum_{i=1}^{l+1} e_i$$

where e_i is embedding of object x_i. Division of number of vertices $l+1$ and path count $c(x_u, x_v)$ is proposed to normalize intermediate object embedding to the scale of input object embeddings. Since path count is extracted and recorded as $c(x_u, x_v)$, this does no result in extra loss of information.

•**PathSim**. PathSim is proposed in [8] to measure similarity between two objects of a same type on a heterogeneous graph. Given a symmetric meta-path Φ, PathSim between x_u and x_v is:

$$s(x_u, x_v) = \frac{2 \times |\{p_{x_u \rightsquigarrow x_v} : p_{x_u \rightsquigarrow x_v} \vdash \Phi\}|}{|\{p_{x_u \rightsquigarrow x_u} : p_{x_u \rightsquigarrow x_u} \vdash \Phi\}| + |\{p_{x_v \rightsquigarrow x_v} : p_{x_v \rightsquigarrow x_v} \vdash \Phi\}|}$$

The edge feature between objects x_u and x_v under meta-path Φ is thereby concatenated as:

$$r_{\Phi}^{(0)}(x_u, x_v) = c(x_u, x_v) \oplus o(x_u, x_v) \oplus s(x_u, x_v) \tag{1}$$

4.2 Aggregating node embedding

We propose a modified version of scaled dot product attention mechanism to aggregate node embedding from its neighbor. Given a meta-path Φ, we update vertex embedding as follow. First we apply 2 layer MLPs on node embedding and its neighbor respectively for scaling purpose. The query vector of node i is

$$q_i = W_{a1}^{(t)}(\tanh(W_{a2}^{(t)} h_i^{(t)})) \tag{2}$$

And key vector $k_{i,j}$ is calculated as:

$$k_{i,j} = W_{a3}^{(t)}(\tanh(W_{a4}^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)}))) \tag{3}$$

And the value vector of neighbor j is calculated as:

$$v_{i,j}^{(t)} = W^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)}) \tag{4}$$

where edge feature $r_{i,j}^{(t)}$ is concatenated to vertex feature of the neighbors, in order to provide additional information. It is worth noticing that due to unique feature of each edge, both key and value vectors are no longer universal for different query nodes. We believe this uniqueness improves expressivity of self-attention mechanism for HIN embedding tasks.

Afterwards, the importance of a neighbor j to object i is calculated as:

$$\tilde{a}_{i,j}^{(t)} = q_i^T \cdot k_{i,j} \tag{5}$$

Then we normalize this importance coefficient by softmax function and calculate attention coefficient as:

$$a_{i,j}^{(t)} = \text{softmax}(\tilde{a}_{i,j}^{(t)}) = \frac{\exp(\tilde{a}_{i,j}^{(t)})}{\sum_{k \in N_i^{\Phi}} \exp(\tilde{a}_{i,k}^{(t)})} \tag{6}$$

Finally the output of the node aggregation layer is a weighted sum of value vectors.

$$h_i^{(t+1)} = \text{ReLU}(W_{h2} \cdot h_i^{(t)} + W_{h1} \cdot \sum_{j \in N_i^{\Phi}} a_{i,j}^{(t)} \cdot v_{i,j}^{(t)}) \tag{7}$$

Although a vertex i is always in its own meta-path neighbor, we still explicitly add term $h_i^{(t)}$ in equation 7 following GraphSAGE[3].

4.3 Aggregating edge embedding

Dan:TODO In multi-layer propagation, we propose to update embeddings of edge as well.

$$r_{i,j}^{(t+1)} = \text{ReLU}(W_{r1} \cdot (h_i^{(t)} \oplus h_j^{(t)} \oplus r_{i,j}^{(t)}) + W_{r2} \cdot r_{i,j}^{(t)}) \tag{8}$$

4.4 Dealing with over-smoothing via adaptive depth

Dan:TODO:relate this example to the example figure used in preliminary section Plain GCNs often degrades rapidly as number of layers increases [6, 7]. Stacking four or more layers into GCNs cause over-smoothing problem and features of vertices tends to converge to the same value. In heterogeneous graphs, this problem is even more severe. Let's consider again meta-path *APCPA* in Fig. 1. A two layer $a_1 a_2 a_3$ propagation on the shrinking $A - A$ homo-graph actually passes across **8** edges on the underlying heterogeneous graph, which is a terrible number of layers for plain graph convolution networks. To alleviate the over-smoothing problem, in equation 7 and 8, we have already included residual connections [4] in updating rule. And we propose to further improve the problem by creating jumping connections [10].

$$h_i^{(final)} = h_i^{(0)} \oplus h_i^{(1)} \cdots \oplus h_i^{(t)} \tag{9}$$

4.5 Aggregation across different meta-path semantics

Dan:need some intuition/justification shit Given a node i, and its set of embeddings under different semantics, a metapath aggregator is a function γ in the form of $y_i = \gamma_\Theta(\{x_{i,1}, x_{i,2}, ..x_{i,P}\})$, where $x_{i,p}$ is the embedding of node i under metapath p, and y_i is the aggregated embedding of node i in the HIN. Θ is the learnable paremeters of the aggregator. Here we propose several metapath aggregators.

• **Attention** Dan:cite The first option we consider is scaled dot-product attention introduced in Transformer.

$$\tilde{h}_i^{\Phi_p} = W_{m1} \cdot \tanh(W_{m2} \cdot h_i^{\Phi_p}) \tag{10}$$

$$\tilde{w}_i^{\Phi_p} = W_w \cdot \sigma(\tilde{h}_i^{\Phi_p}) \tag{11}$$

$$w_i^{\Phi_p} = \mathrm{softmax}(\tilde{w}_i^{\Phi_p}) = \frac{\exp(\tilde{w}_i^{\Phi_p})}{\sum_q \exp(\tilde{w}_i^{\Phi_q})} \tag{12}$$

After the fully-connected self-attention, we propose to reduce across different meta-path semantics by **Sum** operation.

$$z_i = \mathrm{ReLU}(\sum_p w_i^{\Phi_p} \cdot h_i^{\Phi_p}) \tag{13}$$

• **Gated linear unit** Inspired by forget gate mechanisms in recurrent networks (RNN) Dan:TODO:cite, we compute a soft gate between 0 (low importance) and 1 (high importance) to represent different importance to each embedding entry. We make the element-wise gated sum layer as follows:

$$o_i^{\Phi_p} = \sigma(W_o \cdot h_i^{\Phi_p}) \tag{14}$$

$$\tilde{h}_i^{\Phi_p} = \tanh(W_z \cdot h_i^{\Phi_p}) \tag{15}$$

Finally, we have

$$z_i = \sum_p o_i^{\Phi_p} \odot \tilde{h}_i^{\Phi_p} \tag{16}$$

The gated unit o_{Φ_p} with sigmoid output (ranged (0,1)) is used to control which elements are to be aggregated in the output.

• **Other aggregators** In preliminary experiments, **Concat**, **Max-Pooling** and **Mean** aggregators did not generate a better result, and we omit results of these aggregators in experiment section.

4.6 HINGCN: Semi-supervised classification on Heterogeneous Information Network via Graph Convolution Networks

After meta-path aggregation layer, the embeddings are fed to a 2-layer MLP for final output. In semi-supervised classification tasks, we minimize the Cross-Entropy loss and the model generates final prediction based on the largest output value.

$$\mathcal{L}(W) = -\sum_{l \in \mathcal{Y}_L} Y^l ln(z^l) + \lambda \parallel W \parallel_2 \tag{17}$$

$\mathcal{Y}_L$ is the set of labeled nodes in training set. W represents the weight matrices used in the layers of our HINGCN. We use L_2 normalization for parameters involved.

To this end, HINGCN is summarized in Algorithm 1. HINGCN first computes initial edge feature $r_{\Phi_p}^{(0)}$ for each edge under each meta-path. Then it applies an alternative update of node embedding and edge embedding. After that each node fuses a meta-path specific embedding by jumping connections. These embeddings are further aggregated using either meta-path level attention or gated linear units. Finally, embeddings are fed to a two layer MLP and a softmax layer for classification output.

Dan:TODO

Algorithm 1 Overall process of HINGCN

Input: Labeled type T; meta-path set $\Phi_0, \Phi_1, \ldots, \Phi_P$; an initial node representation $\{f_i, \forall x_i \in X_T\}$; number of aggregation layers L;
Output: $C = \{C_1, \ldots, C_k\}$

```
Compose edge features r^(0) according to Equation (1)
h_i^(0) ← f_i, ∀x_i ∈
for do j ← 1 do
    Calculate W = D^{-1} S, where D_ii = Σ_j S_ij
end for
Computes initial edge feature r^(0)_{Φ_p}
for do j ← 1 do
end for
Normalize each column vector x of X such that x^T x = 1
Calculate the coefficient matrix Z* by Eq. ??
Construct Z̃ = (|Z*| + |(Z*)^T|)/2
Run standard spectral clustering methods, e.g., NCuts, with Z̃ as the
similarity matrix to obtain clusters C = {C_r}_{r=1}^k
return C = {C_1, ..., C_k}
```

5 EXPERIMENT

We conducted extensive experiments to evaluate the performance of HINGCN. This section summarizes our results. We compare the various methods using three popular measures, namely, *accuracy*, *micro-F1*, and *macro-F1*. These measures evaluate clustering quality and their values range from 0 to 1, with a larger value indicating a better clustering quality.

5.1 Datasets

A summary of statistics of the datasets rae shown in table **??**.

• **DBLP**: We extract a subset of DBLP with contains 4057 authors (*A*), 14328 papers (*P*) and 20 conferences (*C*). 8789 terms of papers are aggregated to each author as feature, after tf-idf transformation. Here we select meta-paths {*APA*, *APAPA*, *APCPA*} for experiments.
• **Yelp**: A standard spectral clustering method with symmetric normalization.
• **Freebase**: A standard spectral clustering method with symmetric normalization.

5.2 Algorithms for comparison

We evaluate HINGCN and 9 other methods. To demonstrate effectiveness of our edge update mechanism and meta-path GLU layer, we also includes two variants of HINGCN.

• **Node2vec**: A standard spectral clustering method with symmetric normalization.
• **Metpath2vec**: A standard spectral clustering method with divisive normalization.
• **GCN**: A self-tuning spectral clustering method for multi-scale clusters. It uses eigenvector rotation to estimate the number of clusters. To make a fair comparison, we directly set k as in other methods.
• **GAT**: A power iteration based method which generates only one pseudo-eigenvector.
• **HAN**: A hierarchical semi-supervised graph attention network that employs node-level attention and meta-path level attention.
• **HINGCN**$_e d$: A variant of HINGCN which replace update of edge embedding by identity propagation.
• **HINGCN**$_a t$: A variant of HINGCN which uses attention mechanism for meta-path level aggregation.
• **HINGCN**: The proposed semi-supervised classification method based on heterogeneous graph convolution networks.

5.3 Experiment setup

Dan:TODO We implement HINGCN using PyTorch. We use the same learning rate 0.001. The models are trained with two NVIDIA 1080Ti GPUs using data parallelism. The batch size is 1024 for each GPU. We stack two graph aggregators, and each aggregator samples 16 neighbors for both training and testing. Each aggregator is then followed by a ReLU activation layer and a dropout layer with dropout rate 0.5. Output dimension of the layers are set to 64 and 32 respectively. For dataset without node features, we encode a 1-hot matrix as input feature. As mentioned in section Dan:TODO, we ensemble edge feature from pre-trained path-sim and node embeddings. These node embeddings are trained using Node2vec with default parameters on the underlying graph neglecting heterogeneity.

The parameters of all the baseline methods are set according to their original papers. For each method and dataset, we run the experiment 10 times and report average results.

5.4 Performance results

We apply all 10 methods on DBLP. Due to space limitations, for each category of methods, we only show the best performing one. They are, namely, NJW, PIC-k, FUSE, and ROSC.

6 CONCLUSIONS

In this paper we studied the effectiveness of semi-supervised classification methods in handling heterogeneous information network data.

Future work: in this paper we propose to use hand-crafted edge features to enhance missing information in the shrinking homogeneous graph. One possible approach would be automatic learning of edge features to provide an end-to-end solution to heterogeneous graph embedding problems.

7 ACKNOWLEDGMENTS

This research is supported by Hong Kong Research Grants Council GRF HKU 17254016.

REFERENCES

[1] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*. ACM, 135–144.

[2] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. ACM, 855–864.

[3] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[5] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. 2010. Graph Regularized Transductive Classification on Heterogeneous Information Networks. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 6321. Springer, 570–586.

[6] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. Can GCNs Go as Deep as CNNs? *CoRR* abs/1904.03751 (2019).

[7] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI Press, 3538–3545.

[8] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *PVLDB* 4, 11 (2011), 992–1003.

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*. OpenReview.net.

[10] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 5449–5458.