类似 ~

Heterogeneous Graph Convolutional Networks for Bipartite Graph Embedding

Chaoyang He[*]
University of Southern California
Los Angeles, CA, USA
chaoyang.he@usc.edu

Tian Xie[*]
University of Southern California
Los Angeles, CA, USA
xiet@usc.edu

Abstract

There has been relatively little research dedicated to bipartite graph embedding. Arguably, generic graph embedding methods like node2vec and LINE can also be applied to learn bipartite graph embeddings by ignoring the vertex features. However, real-world bipartite graphs also concern the vertex features and the explicit and implicit relationship between two types of entities, which usually exhibit different properties and patterns from other types of graph-structured data. Meanwhile, convolution-based models such as GCN and R-GCN cannot apply to the bipartite graph due to the heterogeneity. To address these problems, in this work, we first propose the Heterogeneous Graph Convolutional Network (HGCN) to learn graph representation for the bipartite graph. With this basic HGCN model, we then build a three-step cascaded HGCN architecture and two output models including the decoder model and the adversarial learning model. This architecture can learn both the explicit and implicit relation between two groups in the bipartite graph. Our evaluation on two large-scale bipartite graph datasets shows that HGCN can gain better learning representation performance than other methods such as Node2Vec and DeepWalk.

1 Introduction

Graph provides a ubiquitous data structure to model interactions (i.e., edges) among entities (i.e., vertices), having been widely used in many applications such as social networks (Wang et al., 2017b), knowledge graphs (Cao et al., 2017), recommendation systems (He et al., 2015), and others (Feng et al., 2018). By far most research efforts have focused on graph embedding. In particular, they represent a vertex as a learnable embedding vector, where the proximity between vectors encodes the information about graph topology. Thus the vertex embedding can be fed into machines learning methods to address various task such as classification, ranking, link prediction, clustering, visualization and so on.

However, they only addressed homogeneous networks like social networks where vertices are the same type (Perozzi et al., 2014; Zhu et al., 2018; Liao et al., 2018; Li et al., 2017), or heterogeneous networks like knowledge graphs where vertices (and/or edges) are of different types (Chang et al., 2015; Dong et al., 2017; Xu et al., 2017). There has been relatively little research dedicated to the bipartite graph.

Although generic graph embedding methods like DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) can be applied to learn vertex embeddings for bipartite graphs by ignoring the vertex type information. But these methods are suboptimal in doing so because of several reasons: (1) Since real-world bipartite networks concern the relationship between two types of entities, explicit relations (i.e., observed links) and implicit relations (i.e., unobserved but transitive links), which usually exhibit different properties and patterns from other types of network data. For example, social network system needs to capture the relationship among users and also their relationship with chatting groups or hobby communities, E-Commerce recommender systems need to capture the collaborative filtering patterns between customers and products, and search engines need to consider the matching signals between queries and web pages. (2) The power-law distribution is a common characteristic of many real-world bipartite networks (Pongnumkul and Motohashi, 2018), but the corpus generated by a universal random walk algorithm may not preserve this property, such as the one used in DeepWalk. This may limit the information of vertices with high degree and oversample

解决幂律分布问题,

[*]equal contribution

vertices with low degree.

Our work addresses the research gap of learning graph representations for the bipartite graph with Heterogeneous Graph Convolutional Networks (HGCN). HGCN is a novel model we proposed for the bipartite graph embedding which addresses the following challenges:

1. **Large-Scale Graph**. The node/edge number is larger than one million.

2. **Heterogeneous Edge/node Features**. Edge and node contain different feature vectors. The principal difficulty is how to represent different information from different spaces into a unified embedding space.

对齐

3. **Semi-Supervised Labels or even Unsupervised Learning without Labels**. The manual labeling of nodes in a large-scale graph requires quite a lot of human resources and it is even impossible in most cases.

4. **Explicit and Implicit relation**. Real-world bipartite graph concerns the relationship between two types of vertices. Our GCN model should model the explicit and implicit relations simultaneously with a joint optimization framework.

2 Related Works

Our work is related to the neural graph embedding methods. We first review them from the perspective of network types.

The pioneer work DeepWalk (Perozzi et al., 2014) and Node2vec (Grover and Leskovec, 2016) extend the idea of Skip-gram (Mikolov et al., 2013) to model homogeneous network. However, they may not be effective to preserve both explicit and implicit relations of the graph. There are some follow-up works exploiting both 1st-order and 2nd-order proximities between vertices to embed homogeneous networks. Specifically, LINE (Tang et al., 2015) learns two separated embeddings for 1storder and 2nd-order relations; SDNE (Wang et al., 2016) and DVNE (Zhu et al., 2018) incorporates both 1st-order and 2nd-order proximities to preserve the network structure; GraRep (Cao et al., 2015) and AROPE (Zhang et al., 2018) further extends the method to capture higher-order proximities. Besides capturing high-order proximities, there are several proposals to incorporate side information into vertex embedding learning, such as vertex labels (Li et al., 2017; Huang et al., 2017), community information (Chen et al., 2016), textual content (Wang et al., 2017a), user profiles (Liao et al., 2018), and location information (Xie et al., 2016). However, these methods might be suboptimal for learning vertex representations for a bipartite network by ignoring the vertex type information.

Metapath2vec++ (Dong et al., 2017) and HNE (Chang et al., 2015) and EOE (Xu et al., 2017) are representative vertex embedding methods for heterogeneous networks. Although they can be applied to the bipartite graph which can be seen as a special type of heterogeneous networks, they are not tailored for learning on bipartite networks. Specifically, HNE aims to integrate content and linkage structures into the embedding process, and Metapath2vec++ ignores the strength of the relations between vertices and treats the explicit and implicit relations as equally. As such, they are suboptimal for vertex representation learning for a bipartite network. It is noteworthy that recent works have shown an increase of interest in generating vertex embedding by neighborhood aggregation encoders (Kipf and Welling, 2016; Schlichtkrull et al., 2017a). These methods rely on vertex features or attributes. However, they haven't considered the long-tail/power-law distribution problem.

As a ubiquitous data structure, bipartite networks have been mined for many applications, among which vertex ranking is an active research problem. For example, HITS (Kleinberg, 1999) learns to rank vertices by capturing some semantic relations within a bipartite network. Co-HITS (Deng et al., 2009) incorporates content information of vertices and the constraints on relevance into vertex ranking of the bipartite graph. BiRank (He et al., 2017) ranks vertices by taking into account both the network structure and prior knowledge. Distributed vertex representation is an alternative way to leverage signals from the bipartite graph. Unlike the ranking task, it learns a low dimensional representation of a vertex, which can be seen as the "features" of the vertex that preserves more information rather than simply a ranking score.

Latent factor model (LFM), which has been widely investigated in the field of recommender systems and semantic analysis, is the most representative model. And a typical implementation of LFM is based on matrix factorization (Rendle et al., 2009; Zhang et al., 2016; He et al., 2016). Recent advances utilize deep learning methods to learn vertex embeddings on the user-item network for recommendation (He and Chua, 2017). It is worth pointing out that these methods are tailored for the recommendation task, rather than for learning informative vertex embeddings. Moreover, they model the explicit relations in the bipartite network only, which can be improved by incorporating implicit relations as shown in (Jiang et al., 2016; Yu et al., 2018a).

Since many interesting tasks involve data cannot be simply represented in Euclidean space and instead lies in an irregular graph-like structure, for example, the social networks, biological networks or brain connections. So in order to extend neural networks to deal with arbitrarily structured graphs, early work used recursive neural networks to process data represented in graph domains as directed acyclic graphs (Frasconi et al., 1998; Sperduti and Starita, 1997). Graph Neural Networks (GNNs) were first introduced in (Gori et al., 2005; Scarselli et al., 2009) as a generalization of recursive neural networks that can directly deal with a more general class of graphs. Driven by the huge success of convolutional networks in computer vision domain, a number of methods have applied the same notation of convolution for graph data, where called graph convolutional networks (GCN) (Wu et al., 2019; Kipf and Welling, 2017). Advances in this direction are often categorized as spectral approaches and non-spectral approaches. Spectral approaches work with a spectral representation of graphs, where learn filters depend on the Laplacian eigenbasis (Bruna et al., 2013; Henaff et al., 2015). As for non-spectral approaches, (Hamilton et al., 2017; Duvenaud et al.) convolutions are directly defined on the graph, operating on groups of spatially close neighbors. In general, these methods perform a convolution by aggregating the neighbor nodes' information, so each node is able to learn a relationship between the nodes in the entire graph. Based on this property of GCN, graph auto-encoders (GAE) have been introduced to solve network embedding problem which aims to represent network vertices into a low-dimensional vector space(Cao et al.; Wang et al., 2016; Pan et al., 2018; Yu et al., 2018b). The idea is that the encoder first utilizes graph convolutional layers to embed the graph into a latent space upon which a decoder is used to reconstruct the graph structure. However, the weakness for GCN based methods is that same node attributes should be assumed. Apart from the GCN based framework, another solution is directly using stacked autoencoder to extract the hidden representations for each node. Nevertheless, the same setting also exists that different nodes should have the same attributes, which is not available in a bipartite graph. (Nassar, 2018) tried to combine GCN with the bipartite graph, where they aggregate nodes by clustering to generate a bipartite graph which can efficiently accelerate and scale the computations of GCN algorithm, but their goal is not learning representation on bipartite graph data.

3 Heterogeneous Graph Convolutional Network for Bipartite Graph

content

The property that bipartite has different feature space but two groups have strong connection motivate us to use GCNs (graph convolution networks) to learn bipartite graph embedding. Unlike purely content-based deep models, GCNs and R-GCN (Schlichtkrull et al., 2017b) leverage both content information (e.g. vertices' features) as well as graph topology structure. However in bipartite networks, the disjointed two vertices sets often have different representations and feature spaces, e.g., in the recommendation system, one vertex set is user representation, another set is item features space. This means that in most cases the bipartite graph has heterogeneous vertices, where directly aggregating the neighbors' features with itself is impossible. This drawback of existing convolution-based methods such as GCN and R-GCN motivate us to design new GCN models. The principal difficulty is how to represent different information from different spaces (features from heterogeneous vertices) into a unified embedding space.

3.1 Problem Definition

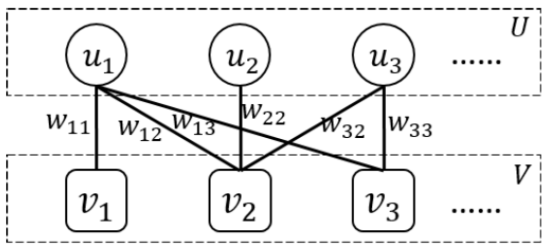


Figure 1: An Example of Bipartite Graph

Definition 1. ***(Graph)*** *Let $G = (U, V, E)$ be a bipartite network, where U and V denote the set of the two types of vertices respectively, and $E \subseteq$*

$U \times V$ defines the inter-set edges. u_i and v_j denote the i-th and j-th vertex in U and V, respectively, where $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$. So the adjacency matrix for node set U is A, which is a $M \times N$ matrix with $A_{ij} = w_ij > 0$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. The adjacency matrix for node set B is simply $A^T \in \mathbb{R}^{N \times M}$.

The two sets of nodes in bipartite graph can be associated with node attributes X^u and X^v respectively, where $X^u \in \mathbb{R}^{M \times P}$ is a feature matrix with $X^u_i \in \mathbb{R}^P$ representing the feature vector of node u_i and $X^v \in \mathbb{R}^{N \times Q}$ with $X^v_j \in \mathbb{R}^Q$ for node v_i.

The task of representation learning in heterogeneous bipartite graph data aims to map all nodes in the graph into a low-dimensional embedding space, where each node is represented as a dense embedding vector. In the embedding space, this embedding vector should preserve the following information in an unsupervised learning setting without labels:

1. Features of two different type nodes, and edge weights/features between two different type nodes.

2. The implicit relation between nodes of the same type.

3. The explicit relation between nodes of different types.

3.2 Heterogeneous Graph Convolutional Network (HGCN)

Traditionally, the convolution performed on a homogeneous graph is defined as

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)}) \tag{1}$$

where $\tilde{A} = I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, A is the adjacent matrix, D is the diagonal node degree matrix, $H^{(l)}$ and $W^{(l)}$ is the l-th layer hidden representation and weight matrix respectively. $\sigma(\cdot)$ denotes an activation function, such as ReLU($\cdot$) = max(0, $\cdot$). As for a heterogeneous bipartite graph, due to the distinct feature representation of two node sets, directly aggregating the nearby features with its own is impossible.

To deal with this problem, we propose a novel method where first merges the nearby nodes' features through a convolution operation then combine with its own by an output model. The reason for using an output model to combine the node attributes and structure embedding is that the node's

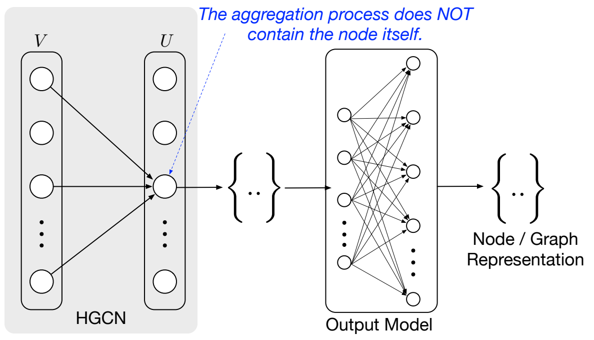

The aggregation process does NOT contain the node itself.
V
U
HGCN
$\{..\}$
Output Model
$\{..\}$
Node / Graph Representation

Figure 2: Heterogeneous Graph Convolutional Network (HGCN)

attributes is highly related to its connection with nodes in a disjoint set and their features, e.g. In our data set, black users might share the same features and have the same connection to some groups. Using an output model will incorporate both the attributes and the topological structure in a non-linear way. To be specific, the propagation model for calculating the forward-pass update of nodes in set U is:

$$H_u^{(l+1)} = \sigma(B_u H_v^{(l)} W^{(l)}) \tag{2}$$

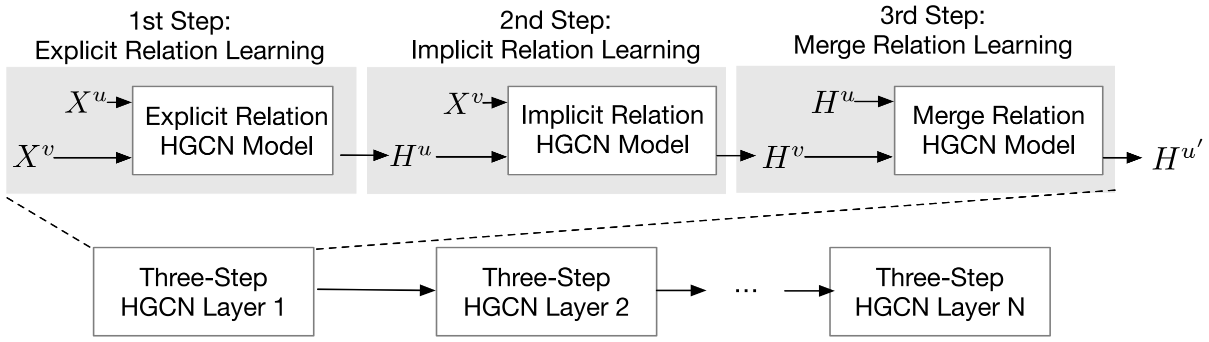
where $B_u = D_u^{-1}A$. $A \in R^{M \times N}$ is the adjacency matrix for set U. $D \in \mathbb{R}^{M \times M}$ is the diagonal degree matrix for set U, which is used for normalization. $H_v^{(l)}$ is the layer hidden representation of set V. When $l = 0$, then $H_v^{(0)} = X^v$, which is the input features of set V. Similarly, as for nodes in set V, the update formulation is:

$$H_v^{(l+1)} = \sigma(B_v H_u^{(l)} W^{(l)}) \tag{3}$$

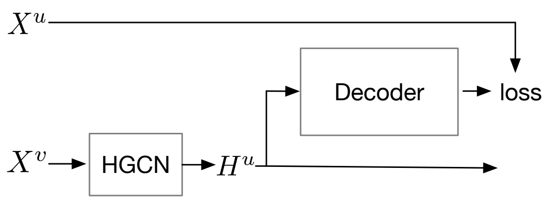
where $B_v = D_v^{-1}A^T$. $D_v \in \mathcal{R}^{N \times N}$ is the diagonal degree matrix for set V. $H_u^{(l)}$ is the layer hidden representation of set U.

Different from regular GCNs, we introduce heterogeneous transformations, i.e. first aggregating the nearby nodes' feature vectors depending on the type or feature space without itself, then aligning its own features through an output model. This can help to learn a representation which incorporates both the nodes' features and the neighbor node features and network structure information. We propose two specific output models, which will be introduced in the next sections.

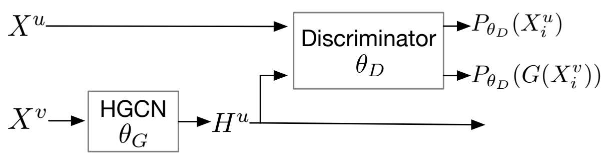
We refer to the above propagation model and the output model as a heterogeneous graph convolutional network (HGCN), as shown in Figure 2.



(a) Cascaded Heterougeous Graph Convolutional Network (HGCN) Architecture



(b) Relation HGCN model with the decoder



(c) Relation HGCN model with adversarial learning

Figure 3: Bipartite Graph Representation Learning with Cascaded HGCN Architecture

3.3 Cascaded HGCN Architecture

We designed a cascaded HGCN architecture (Figure 3(a)), which is an unsupervised learning model. The motivation is that the manual labeling of nodes in a large-scale graph requires quite a lot of human resources and it is even impossible in most cases. In addition to this, we aim to learn the node representation containing the node feature information as well as the implicit and explicit network information.

As shown in Figure 3(a), it contains three steps: the *explicit relation learning step*, the *implicit relation learning step*, and the *merge learning step*. The first step of the cascaded HGCN architecture is an explicit learning step. Its purpose is to learn nodes representation for the group U so that it includes the feature information of the group U and also includes the group V neighbor features and network topology information. The input to this first step is the feature X^u of the group U, and the feature X^v of the opposite group V, and the output is the hidden representation H^u.

We can easily find that the hidden representation H^u does not contain an indirect network topology relationship, that is, the nodes in the group U indirectly connect to other nodes in the same group through the group V. Therefore, we design the second step: the implicit relation learning step. It takes the first step's hidden representation H^u and the group V's features X^v as input. Similar to the output of step one, the output representation H^v contains the feature information of the group V and also includes its group U neighbor features and network topology information. The third step uses the explicit representation H^u and the implicit representation H^v as input, so that the output will contain both implicit and implicit representation.

We can view the three-step cascaded HGCN architecture as a network layer. As shown in Figure 3(a), we can design a multi-layer network architecture based on the three-step architecture. The training process of the cascaded HGCN architecture is step by step and then layer by layer. Its final output is $H^{u'}$ which can contain both the features of the current group and its opposite group as well as the explicit and implicit network information.

The HGCN can be regarded as a encoder-decoder model, where the encoder both encode the graph topology and the feature information from disjointed set. The decoder map this hidden representations with nodes' own features, since we assume that one node's connection is highly related to its own features. To align the similarity between the node features X_u and the hidden H_u, we design two output models. The first is the HGCN with the decoder model and the second is the HGCN with the adversarial learning model. In Figure 3(a), every step's HGCN Model can choose the HGCN with the decoder model or the HGCN with adversarial learning model. In our experiment section, we will compare the performance of these two output models.

3.4 HGCN with Decoder Model

The HGCN with Decoder Model is illustrated in Figure 3(b). It first calculates a forward-pass based on (2) to get the topological and node representation of V which is H^u. Then a decoder is applied to $H_u^{(l)}$, which tries to align $H_u^{(l)}$ with its own attributes X_u. Here we use a simple one layer neural network as our decoder:

$$\hat{X}^u = \sigma(WH^u + b) \tag{4}$$

So the loss function for set U can be defined as:

$$\mathcal{L}_u = ||\hat{X}^u - X^u||^2 \tag{5}$$

The reason use a neural network as a decoder instead of simply concatenate two representation vectors is that we would like to align the topological representation with nodes' own features in a non-linear way, so that the final representation could both embed these two side information. Therefore the decoder and loss function for set V are the same:

$$\hat{X}^v = \sigma(WH^v + b) \tag{6}$$

$$\mathcal{L} = ||\hat{X}^v - X^v||^2 \tag{7}$$

If the second step and the third step choose this decoder model as the output model, the same formula and loss function are applied but every output model has different parameters.

3.5 HGCN with Adversarial Learning

Another output model we design for each step is the HGCN with Adversarial Learning model, in which we define the learning process as a two-player game. Taking the Figure 3(c) as an example, its first game player is the HGCN block, whose purpose is to maximize the similarity between the group U's features X^u and its output H^u, the result of HGCN operation taking group V's features X^v as input. The other player is the Discriminator, whose purpose is to distinguish whether it is from the real X^u features or the H^u feature learned through HGCN. The discriminator's input comes from the real node feature X^u and the output of the HGCN H^u. The two players form an adversarial setting, each of which maximizes their abilities. This approach is in line with the work of Ganin et al. (2016), who proposed to learn latent representations invariant to the input domain, where in our case, the domain is two strongly related groups in the bipartite graph.

Discriminator objective. We define the parameters of the discriminator as θ_D. Its loss function is as follows:

$$L_D(\theta_D, \theta_G) = \sum_{i=1}^{m} \log P_{\theta_D}(Real = 1|X_i^u) + \sum_{j=1}^{n} \log P_{\theta_D}(Real = 0|G(X_j^v) \tag{8}$$

Where "real=1" indicates the discriminator's input sample is from the representation of its own features, and "real=0" indicates that the discriminator's input sample is an hidden representation generated by the HGCN.

HGCN objective. Its loss function is as follows:

$$L_G(\theta_D, \theta_G) = \sum_{i=1}^{m} \log P_{\theta_D}(Real = 0|X_i^u) + \sum_{i=1}^{n} \log P_{\theta_D}(Real = 1|HGCN(X_j^v) \tag{9}$$

To train the adversarial learning model, we follow the standard training procedure of deep adversarial Networks of Goodfellow et al. (2014). For every input sample, the discriminator and the HGCN are trained successively with stochastic gradient updates to respectively minimize L_D and L_G.

4 Experiment

Task	Classfication
Dataset Name	Tencent Social Network
$\|U\|$	1,089,436
$\|V\|$	179,853
$\|E\|$	1,979,756
$\|x_u\|$	14
$\|x_v\|$	36
labeled nodes in Y_u	6151
labeled nodes in Y_v	8315

Table 1: Data set statistic

To evaluate the representation performance, we first perform representation learning for the group U, and then based on the representation result for each node in U, a simple classifier (we use Logistic

Regression in our experiment) is used to evaluate the performance of the node representation. For the random walk-based models, we chose DeepWalk and Node2Vec as the comparison with our HGCN models; For the convolutional-based models, we compare three HGCN models we proposed this paper.

- **One Layer HGCN (1L-HGCN)**. This model first performs a single layer of HGCN on nodes in the group U to obtain hidden vectors H^u, and then concatenate the feature H^u with X^u as the output node representation.
- **Three Layer HGCN with Decoder (3L-Dec-HGCN)**. As shown in Figure (a), this model is a three layers cascade HGCN model with the Decoder model illustrated in Figure (b).
- **Three Layer HGCN with Adversarial Learning (3L-Adv-HGCN)**. As shown in Figure (a), this model is a three layers cascade HGCN model with the adversarial learning model illustrated in Figure (c).

4.1 Evaluation

Our problem is a classification problem, where in our data set is to classify whether one person or group is from black market or not. To measure and evaluate the performance of different methods, here we choose precision (Pre) and recall as our evaluation metric. The formulations are:

$$Pre = \frac{\#correctly \quad classified}{\#selected \quad test} \tag{10}$$

$$Recall = \frac{\#correctly \quad classified}{\#ground \quad truth} \tag{11}$$

The precision characterize given a sample, what's the probability make a right classification. The recall identify the percentage of ground truth that the model can found. Also, average precision (AP) is used to summarize a weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$AP = \sum_n (R_n - R_{n-1})P_n \tag{12}$$

where P_n and R_n are the precision and recall at the n-th threshold.

4.2 Experiments Results

The performance comparison is summarized in table 2. We already finished experiments of the Node2Vec and the raw feature models.

Node2Vec with Attributes			
Top	**AP**	**Prec**	**Recall**
1000	0.9760	0.9600	0.1742
5512	0.8757	0.5996	0.5996
10000	0.8206	0.3876	0.7032
41595	0.6737	0.1246	0.9401
AUC: 0.9878			

Table 2: Node2Vec result with Attributes

Node2Vec without Attributes			
Top	**AP**	**Prec**	**Recall**
1000	0.6146	0.5690	0.1032
5512	0.5356	0.4752	0.4752
10000	0.5244	0.2960	0.5370
41595	0.4555	0.0356	0.6424
AUC: 0.8469			

Table 3: Node2Vec result without Attributes

The Node2Vec (Grover and Leskovec, 2016) is a famous node embedding method, where a bias random walk is performed on the graph to capture the topological structure information. The baseline experiment is first using Node2Vec to get the node embedding of each node. Then ues a logistic regression to determine whether the user or group is from black market. Here we implement the logistic regression with and without adding the nodes' own attributes. From table 2 and 3 we can clearly see that by adding the nodes' attributes, the model can perform better in classification. So not only the graph topological structure, but also the nodes' features are important in the classification. This further motivates us to incorporate multiple types of feature vectors and the graph topology. Besides, the precision and recall show a negative correlation according to the top-k ranking, the higher the precision is, the lower the recall will be.

Note: As of the midterm deadline, we have completed the baseline evaluation of DeepWalk and Node2Vec. The 1L-HGCN model is under development. We almost finished it but haven't got the result. Other experiments are expected to be completed in the final project report. The code of this paper is maintained on GitHub. The GitHub address is: https://github.com/chaoyanghe/bipartite-graph-learning. We already added TA (GitHub ID:

Hunter-Lin) as our collaborator. Please review the code to check our progress.

5 Conclusion

Working in progress.

Acknowledgments

Working in progress.

References

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *arXiv:1312.6203 [cs]*. ArXiv: 1312.6203.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep Neural Networks for Learning Graph Representations. page 8.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1623–1633.

Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128. ACM.

Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate group information to enhance network embedding. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1901–1904. ACM.

Hongbo Deng, Michael R. Lyu, and Irwin King. 2009. A generalized co-hits algorithm and its application to bipartite graphs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248. ACM.

Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144. ACM.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gomez-Bombarelli, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. page 9.

Fuli Feng, Xiangnan He, Yiqun Liu, Liqiang Nie, and Tat-Seng Chua. 2018. Learning on partial-order hypergraphs. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1523–1532. International World Wide Web Conferences Steering Committee.

P. Frasconi, M. Gori, and A. Sperduti. 1998. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786.

M. Gori, G. Monfardini, and F. Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734, Montreal, Que., Canada. IEEE.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 855–864, San Francisco, California, USA. ACM Press.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216 [cs, stat]*. ArXiv: 1706.02216.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670. ACM.

Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364. ACM.

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

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. ACM.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. *arXiv:1506.05163 [cs]*. ArXiv: 1506.05163.

Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM.

Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. 2016. Little Is Much: Bridging Cross-Platform Behaviors through Overlapped Crowds. In *AAAI*, pages 13–19.

Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. ArXiv: 1609.02907.

Thomas N Kipf and Max Welling. 2017. SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS. page 14.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396. ACM.

Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Marcel Nassar. 2018. Hierarchical Bipartite Graph Convolution Networks. *arXiv:1812.03813 [cs, stat]*. ArXiv: 1812.03813.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2609–2615, Stockholm, Sweden. International Joint Conferences on Artificial Intelligence Organization.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Suchit Pongnumkul and Kazuyuki Motohashi. 2018. A bipartite fitness model for online music streaming services. *Physica A: Statistical Mechanics and its Applications*, 490:1125–1137.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017a. Modeling Relational Data with Graph Convolutional Networks. *arXiv:1703.06103 [cs, stat]*. ArXiv: 1703.06103.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017b. Modeling Relational Data with Graph Convolutional Networks. *arXiv:1703.06103 [cs, stat]*. ArXiv: 1703.06103.

A. Sperduti and A. Starita. 1997. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Chuan-Ju Wang, Ting-Hsiang Wang, Hsiu-Wei Yang, Bo-Sin Chang, and Ming-Feng Tsai. 2017a. ICE: item concept embedding via textual information. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 85–94. ACM.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 1225–1234, San Francisco, California, USA. ACM Press.

Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017b. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194. ACM.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *arXiv:1901.00596 [cs, stat]*. ArXiv: 1901.00596.

Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 15–24. ACM.

Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S. Yu. 2017. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 741–749. ACM.

Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. 2018a. Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. AAAI.

Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. 2018b. Learning Deep Network Representations with Adversarially Regularized Autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, pages 2663–2671, London, United Kingdom. ACM Press.

Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334. ACM.

Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2778–2786. ACM.

Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep variational network embedding in wasserstein space. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2827–2836. ACM.

A Appendices

Working in progress.

B Supplemental Material

Working in progress.