ActiveHNE: Active Heterogeneous Network Embedding

Xia Chen[1] , **Guoxian Yu**[1,2,*] , **Jun Wang**[1] , **Carlotta Domeniconi**[3] , **Zhao Li**[4] , **Xiangliang Zhang**[2]
[1]College of Computer and Information Sciences, Southwest University, Chongqing, China
[2]CEMSE, King Abdullah University of Science and Technology, Thuwal, SA
[3]Department of Computer Science, George Mason University, VA, USA
[4]Alibaba Group, Hangzhou, China
{xchen, gxyu, kingjun}@swu.edu.cn, carlotta@cs.gmu.edu, lizhao.lz@alibaba-inc.com,
xiangliang.zhang@kaust.edu.sa

Abstract

Heterogeneous network embedding (HNE) is a challenging task due to the diverse node types and/or diverse relationships between nodes. Existing HNE methods are typically unsupervised. To maximize the profit of utilizing the rare and valuable supervised information in HNEs, we develop a novel Active Heterogeneous Network Embedding (ActiveHNE) framework, which includes two components: Discriminative Heterogeneous Network Embedding (DHNE) and Active Query in Heterogeneous Networks (AQHN). In DHNE, we introduce a novel semi-supervised heterogeneous network embedding method based on graph convolutional neural networks. In AQHN, we first introduce three active selection strategies based on uncertainty and representativeness, and then derive a batch selection method that assembles these strategies using a multi-armed bandit mechanism. ActiveHNE aims at improving the performance of HNE by feeding the most valuable supervision obtained by AQHN into DHNE. Experiments on public datasets demonstrate the effectiveness of ActiveHNE and its advantage on reducing the query cost.

1 Introduction

Networks are pervasive in a wide variety of real-world scenarios, ranging from popular social networks, to citation graphs and gene regulatory networks. Network embedding (NE), also known as network representation learning (NRL), enables us to capture the intrinsic information of the network data by embedding it into a low-dimensional space. Effective NE approaches can facilitate downstream network analysis tasks, such as node classification, community discovery, and link prediction [Cai *et al.*, 2017b].

Heterogeneous information networks (HINs), which involve diverse node types and/or diverse relationships between nodes, are ubiquitous in real-world scenarios [Shi *et al.*, 2017]. Although NE for homogeneous networks, with a single type of nodes and a single type of relationships has been extensively studied [Tang *et al.*, 2015; Wang *et al.*, 2016; Cai *et al.*, 2017b; Goyal and Emilio, 2018], the rich structure of HINs presents a major challenge for heterogeneous network embedding (HNE), since nodes of different types should be treated differently (Challenge 1) [Chang *et al.*, 2015; Fu *et al.*, 2017; Dong *et al.*, 2017; Shi *et al.*, 2018b; Chen *et al.*, 2018].

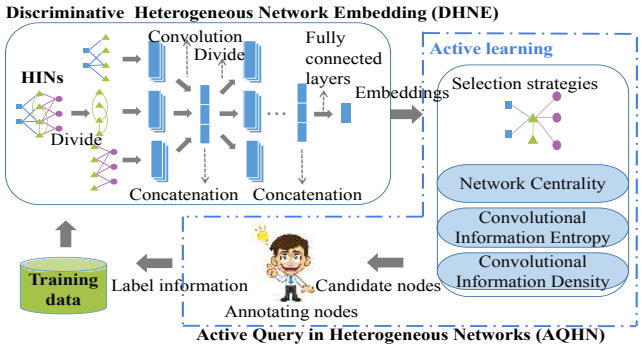


Figure 1: The architecture of ActiveHNE. ActiveHNE consists of two components: Discriminative Heterogeneous Network Embedding (DHNE) and Active Query in Heterogeneous Networks (AQHN). In each iteration, once a network embedding is obtained by DHNE, AQHN selects the most valuable nodes to be queried, and then updates DHNE with the new labels.

Most of the current HNE approaches are unsupervised. One can improve the performance of HNE by properly leveraging supervised information (Challenge 2). However, label acquisition is usually difficult and expensive due to the involvement of human experts (Challenge 3). For Challenge 3, active learning (AL), a technique widely used to acquire labels of nodes during learning, can be adopted to save cost. The selection of labeled data for model training can have significant influence on the prediction stage. AL is expected to find the most valuable nodes to label with reduced query cost [Settles, 2009]. However, since nodes in a heterogeneous network are not independently and identically distributed (*non-i.i.d.*), but connected with links, AL with networks should account for data dependency. In addition, for HINs, the different node types should also be considered.

Based on the high efficiency of graph convolution networks (GCNs) [Kipf and Welling, 2017] in utilizing label information, we propose a novel Active Heterogeneous Network

*Guoxian Yu is the corresponding author (gxyu@swu.edu.cn).

Embedding framework (called ActiveHNE) to address the above three challenges. ActiveHNE includes two components, Discriminative Heterogeneous Network Embedding (DHNE) and Active Query in Heterogeneous Networks (AQHN), as illustrated in Figure 1, and described below.

- In DHNE, we introduce a semi-supervised discriminative heterogeneous network embedding method based on graph convolutional neural networks. Since different types of nodes and relationships should be treated differently, we first decompose the original HIN into homogeneous networks and bipartite networks. For each convolutional layer, DHNE separately learns the deep semantic meanings of nodes in each obtained network, and then concatenates the output vectors of each node from all networks.
- In AQHN, besides the network centrality, we introduce two active selection strategies, namely convolutional information entropy and convolutional information density for HINs with respect to uncertainty and representativeness. In particular, these strategies take advantage of the dependency among nodes and the heterogeneity of HINs by local convolution, whose filter parameters are defined by the node importance (meassured by the number of node types of neighbors and the degree). Then, we iteratively query the most valuable batch of nodes by combining the three strategies using the multi-armed bandit mechanism [Sutton and Barto, 1998].

This work makes the following contributions. (i) We formalize the active heterogeneous network embedding problem, whose objective is to seek the most valuable nodes to query and to improve the performance of HNE using the queried labels. (ii) We present a novel heterogeneous graph convolutional neural network model for node embedding and node classification. (iii) Considering the data dependency among nodes and the heterogeneity of networks, we propose a new active learning method to select the most valuable nodes to label by leveraging local convolution and the multi-armed bandit mechanism. Our experimental study on three real-world HINs demonstrates the effectiveness of ActiveHNE on embedding HINs, and on saving the query cost.

2 Related Work

Most of the previous approaches on HNE are unsupervised [Shi *et al.*, 2018b; Chang *et al.*, 2015; Gui *et al.*, 2017]. Recently, methods have been proposed to leverage meta-paths, either specified by users or derived from additional supervision [Fu *et al.*, 2017; Dong *et al.*, 2017; Shi *et al.*, 2018a]. However, the choice of meta-paths strongly depends on the task at hands, thus limiting their ability of generalization [Shi *et al.*, 2018b]. In addition, they enrich the neighborhood of nodes, resulting in a denser network and in higher training costs [Perozzi *et al.*, 2014].

Graph neural networks (GNNs) are another widely studied approach to leverage supervision [Zhou *et al.*, 2018]. GNNs have the ability to extract multi-scale localized spatial features, and compose them to construct highly expressive representations. Among all GNN approaches, graph convolution networks (GCNs) play a central role in capturing structural dependencies [Wu *et al.*, 2019; Kipf and Welling, 2017]. A comprehensive survey of the literature shows that the majority of current GNNs are designed for homogeneous networks only. GNNs are rarely explored for heterogeneous networks [Zhang *et al.*, 2018], and they are trained based on discretionary supervision.

One can improve the embedding performance by acquiring the labels of the most valuable nodes via AL. However, AL on non-i.i.d. network data is seldom studied. In addition, the diversity of node types in HINs makes the query criterion of AL even harder to design. Although attempts have been made to improve the embedding performance by incorporating AL, they ignore the effect of classifiers' outputs on the importance of nodes and the dependence between nodes [Xin *et al.*, 2018], or neither consider the dependence between nodes, nor the heterogeneity of networks [Zhang *et al.*, 2017; Cai *et al.*, 2017a; Li *et al.*, 2018].

3 The ActiveHNE Framework

In this section, we present our Active Heterogeneous Network Embedding framework, called ActiveHNE. The architecture of ActiveHNE is given in Figure 1. ActiveHNE consists of two components: Discriminative Heterogeneous Network Embedding (DHNE) and Active Query in Heterogeneous Networks (AQHN), which are presented in the following.

3.1 Discriminative Heterogeneous Network Embedding (DHNE)

It's difficult to perform convolutions on networks due to the lack of an Euclidean representation space. In addition, HINs involve different types of nodes and relationships, each requiring its own processing, and further increasing the challenge of computing convolutions. To address this issue, we first divide the original HIN into homogeneous networks and bipartite networks (the latter involving two types of nodes). After this, for each convolutional layer in a layer-wise convolutional neural network, we separately convolve and learn the deep semantic meanings of nodes in each obtained network, and then concatenate the output vectors of each node from all networks.

Let $\{\mathcal{G}_t|t = 1, 2, \cdots, T\}$ be the collection of obtained homogeneous networks and bipartite networks, and let $\{\mathbf{A}_t|t = 1, 2, \cdots, T\}$ denote the adjacency matrices corresponding to $\{\mathcal{G}_t\}$. The spectral graph convolution theorem defines the convolution in the Fourier domain based on the normalized graph Laplacian $\mathbf{L}_t = \mathbf{I}_t - \mathbf{D}_t^{-\frac{1}{2}}\mathbf{A}_t\mathbf{D}_t^{-\frac{1}{2}} = \mathbf{D}_t^{-\frac{1}{2}}(\mathbf{D}_t - \mathbf{A}_t)\mathbf{D}_t^{-\frac{1}{2}}$, where $\mathbf{I}_t$ is the identity matrix and $\mathbf{D}_t = diag(\sum_i \mathbf{A}_t(i, j))$ denotes the degree matrix [Kipf and Welling, 2017; Wang *et al.*, 2018].

Since the nodes' degree distribution in an HIN may vary greatly, and the interaction between two connected nodes may be directed, an asymmetric matrix $\mathbf{P}_t = \mathbf{D}_t^{-1}\mathbf{A}_t$, instead of the symmetric $\mathbf{L}_t$, is more suitable to define the Fourier domain. $\mathbf{P}_t$ is the transition probability matrix.

In this paper, we separately convolve on each obtained network using the transition probability matrix $\mathbf{P}_t$ as Fourier basis. Specifically, let $\mathbf{P}_t = \mathbf{\Phi}_t\mathbf{\Lambda}_t\mathbf{\Phi}_t^{-1}$, where $\mathbf{\Lambda}_t$ and $\mathbf{\Phi}_t$ are the eigenvector matrix and the diagonal matrix of eigenvalues

of $\mathbf{P}_t$, respectively. The convolution on each obtained network is defined as follows:

$$\begin{aligned} g_{\theta_t} \star \mathbf{X}_t = g_{\theta_t}(\mathbf{P}_t)\mathbf{X}_t &= g_{\theta_t}(\mathbf{\Phi}_t \mathbf{\Lambda}_t \mathbf{\Phi}_t^{-1})\mathbf{X}_t \\ &= \mathbf{\Phi}_t g_{\theta_t}(\mathbf{\Lambda}_t)\mathbf{\Phi}_t^{-1}\mathbf{X}_t \end{aligned} \tag{1}$$

where $\mathbf{X}_t \in \mathbb{R}^{N_t \times D}$ is the input signal of the network $\mathcal{G}_t$ (N_t and D denote the number of nodes and the number of features of each node in $\mathcal{G}_t$, respectively). $g_{\theta_t} \star \mathbf{X}_t$ gives the product of the signal $\mathbf{X}_t$ with a filter g_{θ_t} in the graph Fourier domain, which denotes the output of graph convolution. $\mathbf{\Phi}_t^{-1}\mathbf{X}_t$ is the Fourier transform of signal $\mathbf{X}_t$. More details about the spectral graph convolution in the Fourier domain can be found in [Wang *et al.*, 2018].

To convolve the local neighbors of the target node, we define $g_{\theta_t}(\mathbf{\Lambda}_t)$ as a polynomial filter up to order K [Defferrard *et al.*, 2016; Zhang *et al.*, 2018] as follows:

$$g_{\theta_t}(\mathbf{\Lambda}_t) = \sum_{k=1}^{K} \theta_{tk} \mathbf{\Lambda}^k \tag{2}$$

where $\theta_t \in \mathbb{R}^K$ is a vector of polynomial coefficients. Thus, we have:

$$g_{\theta_t} \star \mathbf{X}_t = \mathbf{\Phi}_t (\sum_{k=1}^{K} \theta_{tk} \mathbf{\Lambda}_t^k) \mathbf{\Phi}_t^{-1} \mathbf{X}_t = \sum_{k=1}^{K} \theta_{tk} \mathbf{P}_t^k \mathbf{X}_t \tag{3}$$

From Eq. (3), the convolution on $\mathcal{G}_t$ only depends on the nodes that are at most K steps away from the target node. In other words, the output signals after the convolution operations are defined by a K-order approximation of localized spectral filters on networks. The filter parameters θ_{tk} can be shared over the whole network $\mathcal{G}_t$. Moreover, we generalize Eq. (3) to $D \times d$ filters for feature maps, i.e., we map the original feature dimension D to d. Thus, the convolution operation on the network $\mathcal{G}_t$ is formalized as follows:

$$\mathbf{H}_t = \sigma(\sum_{k=1}^{K} \mathbf{P}_t^k \mathbf{X}_t \mathbf{\Theta}_t) \tag{4}$$

where $\mathbf{\Theta}_t \in \mathbb{R}^{D \times d}$ and $\mathbf{H}_t \in \mathbb{R}^{N_t \times d}$ denote the matrix of filter parameters (the trainable weight matrix) and the convolved signal matrix (output signals), respectively. We use $ReLU(\cdot)$ for $\sigma(\cdot)$ as the activation function.

So far, we have performed the convolutions separately on each individual network. To leverage both the homologous and heterogeneous information of HINs for embedding, we then concatenate in order the vectors of the convoluted signals to obtain the final output signals for each node, according to the network it belongs to. For a node that is not an element of a network, we use a zero vector to represent the corresponding output signals. Let $\mathbf{Z}_t$ denote the concatenated convoluted signals of nodes in $\mathcal{G}_t$, we define the layer-wise convolution on $\mathcal{G}_t$ as follows:

$$\mathbf{H}_t^{(l)} = \sigma(\sum_{k=1}^{K} \mathbf{P}_t^k \mathbf{Z}_t^{(l)} \mathbf{\Theta}_t^{(l)}), l = 0, 1, 2, ... \tag{5}$$

where $\mathbf{Z}_t^{(l)} \in \mathbb{R}^{N_t \times T d^{(l-1)}}$, $\mathbf{\Theta}_t^{(l)} \in \mathbb{R}^{T d^{(l-1)} \times d^{(l)}}$, and $\mathbf{H}_t^{(l)} \in \mathbb{R}^{N_t \times d^{(l)}}$ denote the activations (input signals), the matrix of filter parameters (the trainable weight matrix), and the convolved signal matrix (output signals) in the l-th layer, respectively. $d^{(l)}$ is the embedding dimension of the l-th layer, and T is the number of networks. Specifically, $\mathbf{Z}_t^{(0)} = \mathbf{X}_t$ and $\mathbf{\Theta}_t^{(0)} \in \mathbb{R}^{D \times d^{(1)}}$. Eq. (5) indicates the layer-wise propagation rule in layer-wise convolutional neural networks. Although we performed the convolutions separately on each individual network, both the homologous and heterogeneous information of HINs are used for the embedding thanks to the layer-wise concatenation operators.

After β layers of convolutions and concatenations, we obtain the final output vectors of all nodes as $\mathbf{E} = \mathbf{Z}^{\beta} \in \mathbb{R}^{N \times T d^{(\beta)}}$. To obtain a discriminative embedding, we leverage supervision (i.e., label information) by adding a fully connected layer to predict the labels of nodes as follows:

$$\mathbf{F} = \sigma(\mathbf{E}\mathbf{\Theta}^{pre}) \tag{6}$$

where $\mathbf{\Theta}^{pre} \in \mathbb{R}^{T d^{(\beta)} \times C}$ is the hidden-to-output weight matrix. $\mathbf{F} \in \mathbb{R}^{N \times C}$, and $\mathbf{F}_{ic}$ stores the probability that the i-th node belongs to class c. The activation function $\sigma(\cdot)$ in the last layer is the softmax function, which is defined as $softmax(\mathbf{F}_{ic}) = \frac{exp(\mathbf{F}_{ic})}{\sum_{c'=1}^{C} exp(\mathbf{F}_{ic'})}$. Finally, the supervised loss function is defined as the cross-entropy error over all labeled nodes as follows:

$$loss = -\sum_{i=1}^{L} \sum_{c=1}^{C} \mathbf{Y}_{ic} ln \mathbf{F}_{ic} \tag{7}$$

where $\mathbf{Y} \in \{0, 1\}^{N \times C}$ stores the ground-truth labels of nodes. If the i-th node is associated with the c-th label, $\mathbf{Y}_{ic} = 1$. Otherwise, $\mathbf{Y}_{ic} = 0$. The neural network weight parameters $\mathbf{\Theta}_t^{(l)}$ and $\mathbf{\Theta}^{pre}$ are optimized using gradient descent to minimize Eq. (7). As such, Eqs. (6) and (7) enable a semi-supervised model for discriminative node embedding. The label of the i-th node can be predicted as $y_i = \arg\max_c \mathbf{F}_{ic}$.

3.2 Active Query in Heterogeneous Networks (AQHN)

In DHNE, we perform a semi-supervised heterogeneous network embedding, which requires the participation of label information. However, label acquisition is usually difficult and expensive due to the involvement of human experts. More importantly, different supervision may lead to different embedding performance. To train a more effective DHNE, we propose an active query component, AQHN, to acquire the most valuable supervision within a given budget (e.g., the allowed maximum number of queries).

Uncertainty and representativeness are widely used criteria to select samples for query in AL. Uncertainty selects the sample that the current classification model is least certain, while representativeness selects the sample that can well represent the overall input patterns of unlabeled data. Empirical studies have shown that combining the two criteria can make more efficient selection strategies [Huang *et al.*, 2014]. In the following, we first introduce three active selection strategies (Network Centrality, Convolutional Information Entropy, and Convolutional Information Density) for HINs based on uncertainty

and representativeness. Then, we propose a novel method to combine these strategies to adaptively and iteratively select the most valuable batch of nodes to query, by leveraging the multi-armed bandit mechanism [Sutton and Barto, 1998].

Selection Strategy

Network centrality (NC). NC (e.g., degree centrality and closeness centrality) [Freeman, 1978] is an effective measure to evaluate the representativeness of nodes. In this paper, we simply use degree centrality, which is defined as $\phi_{nc}(v_i) = |\mathcal{N}_i|$, to evaluate the centrality of nodes. $\mathcal{N}_i$ includes all the direct neighbors of v_i. Other measures of network centrality in HINs will be studied later.

Nodes in a HIN are non-i.i.d. and are connected by links, which reflect the dependency among nodes. Inspired by the idea of spectral graph convolution that defines the convoluted signal as a linear weighted sum of its neighbor signals, we propose two novel active strategies to select nodes for query in HINs based on a convolution of neighbors. We first define the convolution parameters (i.e., weight parameters) and then the selection strategies. Let $w_i = tanh(\frac{n_i}{N} + \frac{m_i}{V_T}) \in [0, 1)$ be the weight that quantifies the importance of node v_i. $tanh(\cdot)$ is the hyperbolic tangent function. Here n_i and m_i represent the number of neighbor nodes of v_i and the number of node types of these neighbors. N and V_T are the total number of nodes and node types in the whole network, respectively. A larger value of n_i or m_i implies that more complex information is conveyed by v_i, and thus v_i may be more important to its neighbor nodes. In the following, we use w_i as the weight parameters for convolving neighbors.

Convolutional Information Entropy (CIE). Information Entropy (IE) is a widely used metric to evaluate uncertainty. In this paper, we evaluate the uncertainty of node v_i using CIE as follows:

$$\phi_{cie}(v_i) = \sum_{v_j \in \{v_i \bigcup \mathcal{N}_i\}} w_j (-\sum_{c=1}^{C} \mathbf{F}_{jc} \log \mathbf{F}_{jc}) \qquad (8)$$

The uncertainty of v_i is a weighted sum of the uncertainties of its neighbors and itself.

Convolutional Information Density (CID). The representativeness of nodes in the embedding space is also crucial to measure the value of nodes. We apply k-means clustering on the embedding to calculate the information density (ID) of nodes, due to its high efficiency. The number of clusters for k-means is simply set to the number of class labels. CID of v_i based on its neighbors is quantified as follows:

$$\phi_{cid}(v_i) = \sum_{v_j \in \{v_i \bigcup \mathcal{N}_i\}} w_j \frac{1}{1 + dis(\mathbf{E}_j, \varphi(v_j))} \qquad (9)$$

where $dis(\cdot)$ is the distance metric (i.e., Euclidean distance) in the embedding space, $\varphi(v_i)$ is the center vector of the cluster to which v_i belongs. $\mathbf{E}_j$ is the embedding of the j-th node. $\varphi(v_j)$ and $\mathbf{E}_j$ belong to the same space. The proposed CIE and CID measure the value (uncertainty or representativeness) of a node based on the node itself and its neighbors, while IE and ID are based on the node only. Since nodes in networks are connected by links, CIE and CID are more appropriate than IE and ID. We demonstrate this in Section 4.3.

Multi-Armed Bandit for Active Node Selection

We select the most valuable nodes by leveraging the above three selection strategies. In particular, we study the batch mode setting, in which b nodes are queried in each iteration. First, we select top b nodes with the highest ϕ_{nc}, ϕ_{cie}, and ϕ_{cid} scores as the initial candidates of each selection strategy, respectively. To jointly select the most valuable b nodes from all selection strategies, one can evaluate the score of each node by using the weighted sum of scores of each strategy, where the weights capture the importance of corresponding strategies. Then, the problem of active node selection is transformed into the estimation of the importance of each strategy. But the importance of each strategy is time-sensitive and thus difficult to be specified [Cai *et al.*, 2017a]. We introduce a novel method to adaptively learn the dynamic weight parameters based on the multi-armed bandit mechanism. The well-known multi-armed bandit (MAB) problem is a simplified version of the reinforcement learning problem [Sutton and Barto, 1998], which explores what a player should do given a bandit machine with Λ arms and a budget of iterations. In each iteration, an agent plays one of the Λ arms to receive a reward. The objective is to maximize the cumulative reward. Combinatorial MAB (CMAB) [Chen *et al.*, 2013], an extension of MAB, allows to play multiple arms in each iteration.

Based on the idea of the CMAB, we can view each selection strategy as an arm, and approximate the importance of each strategy by estimating the expected reward (i.e., utility) of the corresponding arm. Let $\mathcal{C}_r^\lambda$ be the initial candidate set of arm λ in iteration r, and $\mathcal{Q}_r$ be the actually queried set of nodes in that iteration. Intuitively, the actual reward of arm λ can be defined as:

$$\mu_r(\lambda) = \psi(f_{\mathcal{L}_r \bigcup \mathcal{Q}_r^\lambda}) - \psi(f_{\mathcal{L}_r}) \qquad (10)$$

where $\mathcal{L}_r$ is the available labeled set of nodes in iteration r. $\mathcal{Q}_r^\lambda = \mathcal{C}_r^\lambda \bigcap \mathcal{Q}_r$ is the set of queried nodes that are dominated by arm λ in iteration r. $f_{\mathcal{L}_r}$ is the classifier trained on $\mathcal{L}_r$, and $\psi(f_{\mathcal{L}_r})$ is the classification performance of $f_{\mathcal{L}_r}$. We observe that $\mu_r(\lambda)$ for the current iteration can't be computed since the ground-truth of $\mathcal{Q}_r^\lambda$ is unavailable. The empirical reward is typically used to estimate the expected reward of arms. But computing the empirical $\mu_r(\lambda)$ of each arm in each iteration is very time-consuming; as such, we estimate the empirical reward of each arm using the local embedding changes of nodes caused by the arm.

We first define the local embedding changes caused by arm λ in iteration r as follows:

$$\Delta_r(\lambda) = \sum_{v_i \in \mathcal{Q}_r^\lambda} \sum_{v_j \in \mathcal{N}(v_i)} dis(\mathbf{E}_j^r, \mathbf{E}_j^{r-1}) \qquad (11)$$

where $dis(\cdot)$ is the distance metric (e.g., Euclidean distance), $\mathcal{N}(v_i)$ is the neighbors of v_i, and $\mathbf{E}_j^r$ is the node embedding of v_j in iteration r. Eq. (11) measures the empirical reward of arm λ in iteration r using the local embedding changes of nodes caused by the arm λ, which equates to the embedding changes of neighbor nodes of the nodes dominated by arm λ (or $\mathcal{Q}_r^\lambda$). This AL strategy aims to select nodes that result in the greatest change to the embeddings when their labels are available. The intuition is that one can view the magnitude of the resultant change of embeddings as the value of purchasing the labels. If

this magnitude of change is small, then the labels do not provide much new information and have a low value. To achieve a fair comparison and avoid bias, the empirical reward of arm λ in iteration r is estimated as $\hat{\mu}_r(\lambda) = \frac{\Delta_r(\lambda)}{\Delta_r(\bigcup_{\lambda=1}^{\Lambda}\lambda)}$, where $\Delta_r(\bigcup_{\lambda=1}^{\Lambda}\lambda)$ denotes the local embedding changes caused by all arms (or $\mathcal{Q}_r$). Due to the fact that the importance of each selection strategy changes over time, we use the average of the last two empirical rewards to estimate the current expected reward as follows:

$$\bar{\mu}_r(\lambda) = \frac{\hat{\mu}_{r-2}(\lambda) + \hat{\mu}_{r-1}(\lambda)}{2} \tag{12}$$

To mitigate the exploration-exploitation dilemma of CMAB, the combinatorial upper confidence bound algorithm [Chen *et al.*, 2013] estimates expected rewards based on the empirical rewards and the number of times an arm is explored. In the same way, we adjust $\bar{\mu}_r(\lambda)$ as $\tilde{\mu}_r(\lambda) = \overline{\mu}_r(\lambda) + \sqrt{\frac{3lnr}{2n_\lambda}}$, where n_λ denotes the total number of nodes queried by arm λ. This adjustment can boost the expected reward of under-explored arms to avoid dismissing a potentially optimal strategy without sufficient evidence.

After this, to avoid selecting highly controversial nodes, we estimate the expected reward of un-queried nodes $v_i \in \bigcup_{\lambda=1}^{\Lambda}\mathcal{C}_r^{\lambda}$ in iteration r using the weighted Borda count as follows:

$$\tilde{\mu}_r^*(v_i) = \sum_{\lambda=1}^{\Lambda} \tilde{\mu}_r(\lambda)(b - rank_r^{\lambda}(v_i)) \tag{13}$$

where $rank_r^{\lambda}(v_i) \in [1, b]$ is the rank order of node v_i in arm λ in iteration r (sorted in descending order of scores). Finally, the top b nodes (from $\bigcup_{\lambda=1}^{\Lambda}\mathcal{C}_r^{\lambda}$) with the highest $\tilde{\mu}_r^*(v_i)$ are selected as the query batch $\mathcal{Q}_r$ in iteration r.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate ActiveHNE on three real-world HINs extracted from DBLP[1], Cora[2], and MovieLens[3]. The extracted DBLP consists of 14K papers, 20 conferences, 14K authors, and 9K terms, with a total of 171K links. The extracted MovieLens includes 9.7K movies, 12K writers, 4.9K directors, 0.6K users, and 1.5K tags, with a total of 140K links. The extracted Cora has 25K authors, 19K papers, and 12K terms, with 146K links. We evaluate the performance of network embedding using the Accuracy of node classification task. More details of the task can be found in the supplemental file.

Baselines. we compare ActiveHNE against the following state-of-the-art methods and a variant of ActiveHNE that randomly selects nodes to query (in a kind of naive AL setting):

- **GCN** [Kipf and Welling, 2017]: a semi-supervised network embedding model, with no consideration of networks heterogeneity. To adapt GCN in AL settings, nodes are randomly selected for query in each iteration (naive AL setting).
- **metapath2vec** [Dong *et al.*, 2017] and **HHNE** [Wang *et al.*, 2019]: two unsupervised HNE methods also adapted in the naive AL setting.
- **AGE** [Cai *et al.*, 2017a] and **ANRMAB** [Li *et al.*, 2018]: two active network embedding methods without considering the dependence between nodes and the heterogeneity of networks.
- **DHNE**: a variant of ActiveHNE that randomly selects nodes to query in the naive AL setting.

For the proposed DHNE and ActiveHNE, we simply set $K = 1$ for comparative evaluation, and leave the investigation on K in the supplemental file. We train DHNE using a network with two convolutional layers and one fully connected layer as described in Section 3.1, with a maximum of 200 epochs (training iterations) using Adam. The dimensionality of the two convolutional filters is 16 and C, respectively. We use an L_2 regularization factor for all the three layers. The remaining parameters are fixed as in GCN [Kipf and Welling, 2017]. For metapath2vec and HHNE, we apply the commonly used meta-path schemes "APA" and "APCPA" on DBLP and Cora, and we use "DMTMD" and "DMUMD" on MovieLens to guide metapath-based random walks. The walk length and the number of walks per node are set to 80 and 40 as in HHNE, respectively.

Following the experimental settings in [Kipf and Welling, 2017], we randomly divide the labeled nodes into three parts: the training set (25% of the labeled nodes), the validation set (25% of the labeled nodes for hyperparameter optimization in DHNE), and the remaining as the testing set. For AL settings, the training set is used as the unlabeled pool ($\mathcal{U}$). All the comparing methods in AL settings iteratively query the labels of the selected batch of nodes from $\mathcal{U}$, and then add these queried nodes with labels into $\mathcal{L}$ (the set of labeled training nodes). For a fair comparison, we use the proposed DHNE as the basic embedding and classification method for all active learning methods (AGE and ANRMAB) in the experiments. The non-AL methods (i.e., DHNE, GCN, metapath2vec, and HHNE), randomly select the nodes to label in each iteration of AL. To evaluate the classification performance of metapath2vec and HNNE, we train a logistic regression classifier using the respective embedding of nodes. In the following, we run each method ten times and report the average results.

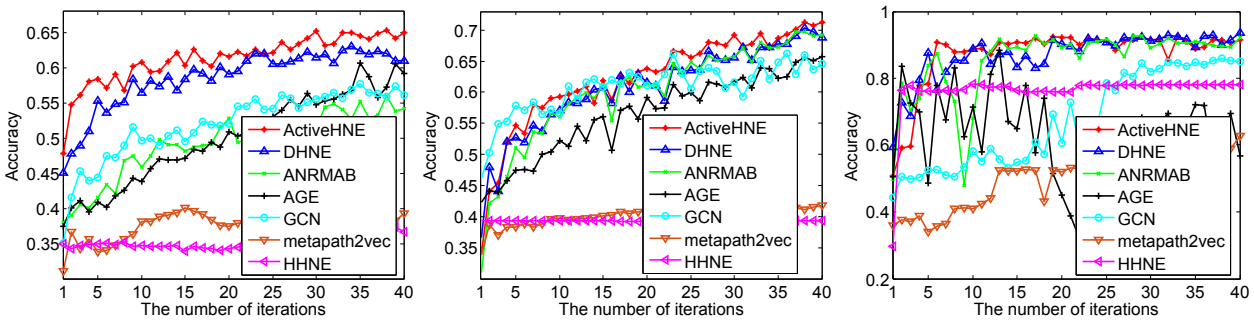
4.2 Comparison against State-of-the-art Methods

The goal of ActiveHNE is to improve classification performance with as few queried nodes as possible. Figure 2 shows the accuracy of all the comparing methods on the three datasets, as a function of the number of iterations. One iteration corresponds to b queried nodes. We set the batch size $b = 20$ for Cora and MovieLens, and $b = 5$ for DBLP, to display the difference in accuracy with respect to the number of iterations. From Figure 2, we can make the following observations:
(i) Active vs. naive-active: ActiveHNE, an active method that combines DHNE and AHQN, significantly outperforms naive-active methods (DHNE, GCN, HHNE, and metapath2vec), which randomly select nodes for query. This shows that AL is conducive to improving embeddings for classification.

[1]https://dblp.uni-trier.de/db/

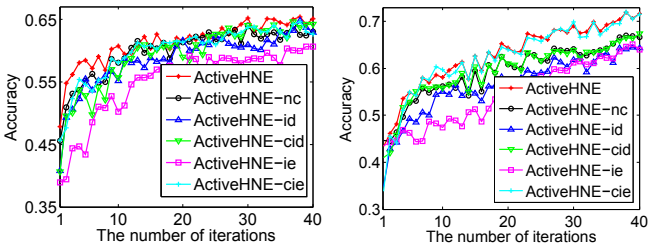
[2]http://web.cs.ucla.edu/ yzsun/data/

[3]https://grouplens.org/datasets/movielens/



(a) MovieLens (b) Cora (c) DBLP

Figure 2: Accuracy vs. number of iterations for all methods on the three datasets.



(a) MovieLens (b) Cora

Figure 3: Accuracy vs. number of iterations: ActiveHNE against its four variants on MovieLens and Cora.

(ii) ActiveHNE vs. other active methods: ActiveHNE outperforms the other AL-assisted methods (ANRMAB and AGE) on MovieLens and Cora, and has comparable performance to ANRMAB on DBLP. Since these three methods use the same embedding module and only differ on the active learning strategy, the superior performance of ActiveHNE validates the effectiveness of our designed active query strategy. ANRMAB and AGE lose to DHNE in most cases. This is because they don't consider the heterogeneity and dependency of nodes in HINs. These results demonstrate the effectiveness of our proposed AQHN for DHNE.
(iii) DHNE vs. other network embedding methods: DHNE significantly outperforms the three representative network embedding methods (GCN, HHNE, and metapath2vec), when they all use the naive-AL setting. This observation shows the superiority of DHNE in embedding HINs for nodes' classification, and it also justifies the rationality of dividing HINs into homologous networks and bipartite networks. The poor performance of HHNE and metapath2vec may be caused by the improper meta-path schemes and by the sensibility of parameters in metapath-based random walks.

4.3 Effectiveness of Individual Selection Strategy

In Section 3.2, we use three node selection strategies: NC, CIE, and CID. The latter two are our proposed novel strategies. To validate their effectiveness, we introduce five variants:

- **ActiveHNE-nc** only uses NC ϕ_{nc};
- **ActiveHNE-cie** only uses CIE ϕ_{cie} in Eq. (8);
- **ActiveHNE-ie** only uses the original information entropy $\phi_{ie}(v_i) = -\sum_{c=1}^{C} \mathbf{F}_{ic} \log \mathbf{F}_{ic}$;
- **ActiveHNE-cid** only uses CID ϕ_{cid} in Eq. (9);
- **ActiveHNE-id** only uses the original information density $\phi_{id}(v_i) = \frac{1}{1+dis(\mathbf{E}_i, \varphi(v_i))}$.

The same settings as in Figure 2 are used, and the results are shown in Figure 3. From Figure 3, we can conclude the following:
(i) ActiveHNE achieves the best accuracy among its variants. Although ActiveHNE-cie obtains an accuracy comparable to ActiveHNE on Cora, it significantly loses to ActiveHNE on MovieLens. These results demonstrate the effectiveness of ActiveHNE in combining three active selection strategies, since one single strategy cannot fit all datasets.
(ii) ActiveHNE-cie and ActiveHNE-cid achieve a better accuracy than ActiveHNE-ie and ActiveHNE-id, respectively. This result corroborates the effectiveness of our proposed CIE and CID in selecting the most uncertain and most representative nodes.

5 Conclusion

In this paper, we studied how to achieve active discriminative heterogeneous network embedding by optimally acquiring and using labels of network nodes. The proposed framework ActiveHNE extends graph convolution networks to heterogeneous networks by dividing the given network into multiple homogeneous and bipartite sub-networks, and performing convolutions on these networks. Three different query strategies are combined to query the labels of the most valuable nodes, which are fed back for the next round of discriminative network embedding. ActiveHNE achieves a superior or comparable performance to other methods. More extensive performance evaluation of ActiveHNE is given in the supplemental file. The code and supplemental file of ActiveHNE are available at http://mlda.swu.edu.cn/codes.php?name=ActiveHNE.

Acknowledgments

This work is supported by NSFC (61872300 and 61873214), Fundamental Research Funds for the Central Universities (XDJK2019B024), NSF of CQ CSTC (cstc2018jcyjAX0228 and cstc2016jcyjA0351), King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

References

[Cai *et al.*, 2017a] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *CoRR, abs/1705.05085*, 2017.

[Cai *et al.*, 2017b] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *TKDE*, 30(9):1–1, 2017.

[Chang *et al.*, 2015] Shiyu Chang, Han Wei, Jiliang Tang, Guo Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, pages 119–128, 2015.

[Chen *et al.*, 2013] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework, results and applications. In *ICML*, pages 1–9, 2013.

[Chen *et al.*, 2018] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *KDD*, pages 1177–1186, 2018.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[Dong *et al.*, 2017] Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami, Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144, 2017.

[Freeman, 1978] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978.

[Fu *et al.*, 2017] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*, pages 1797–1806, 2017.

[Goyal and Emilio, 2018] Palash Goyal and Ferrara Emilio. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151(1):78–94, 2018.

[Gui *et al.*, 2017] Huan Gui, Jialu Liu, Fangbo Tao, Jiang Meng, and Jiawei Han. Large-scale embedding learning in heterogeneous event data. In *ICDM*, pages 907–912, 2017.

[Huang *et al.*, 2014] Sheng Jun Huang, Jin Rong, and Zhi Hua Zhou. Active learning by querying informative and representative examples. *TPAMI*, 36(10):1936–1949, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, pages 1–14, 2017.

[Li *et al.*, 2018] Gao Li, Yang Hong, Zhou Chuan, Wu Jia, Pan Shirui, and Hu. Yue. Active discriminative network representation learning. In *IJCAI*, pages 2142–2148, 2018.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[Settles, 2009] Burr Settles. Active learning literature survey. *Technical report 1648, University of Wisconsin-Madison*, 2009.

[Shi *et al.*, 2017] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. A survey of heterogeneous information network analysis. *TKDE*, 29(1):17–37, 2017.

[Shi *et al.*, 2018a] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. Aspem: Embedding learning by aspects in heterogeneous information networks. In *SDM*, pages 144–152, 2018.

[Shi *et al.*, 2018b] Yu Shi, Qi Zhu, Fang Guo, Zhang Chao, and Jiawei Hang. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *KDD*, pages 2190–2199, 2018.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *TNN*, 9(5):1054–1054, 1998.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, pages 1225–1234, 2016.

[Wang *et al.*, 2018] Chu Wang, Samari Babak, and Siddiqi Kaleem. Local spectral graph convolution for point set feature learning. pages 52–66, 2018.

[Wang *et al.*, 2019] Xiao Wang, Yiding Zhang, and Chuan Shi. Hyperbolic heterogeneous information network embedding. In *AAAI*, pages 1–8, 2019.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. In *arXiv:1901.00596v1*, 2019.

[Xin *et al.*, 2018] Doris Xin, Ahmed El-Kishky, De Liao, Brandon Norick, and Jiawei Han. Active learning on heterogeneous information networks: A multi-armed bandit approach. In *ICDM*, pages 1350–1355, 2018.

[Zhang *et al.*, 2017] Ye Zhang, Matthew Lease, and Byron C. Wallace. Active discriminative text representation learning. In *AAAI*, pages 3386–3392, 2017.

[Zhang *et al.*, 2018] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi, and Yangyong Zhu. Deep collective classification in heterogeneous information networks. In *WWW*, pages 399–408, 2018.

[Zhou *et al.*, 2018] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. In *arXiv:1812.08434*, 2018.