Binarized Attributed Network Embedding via Neural Networks

Hangyu Xia[1,2,3], Neng Gao[1,3 *], Jia Peng[1,3], Jingjie Mo[1,2,3], Jiong Wang[1,2,3]
[1]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences*
[2]*School of Cyber Security, University of Chinese Academy of Sciences*
[3]*Institute of Information Engineering, Chinese Academy of Sciences*
Beijing, China
{xiahangyu, gaoneng, pengjia, mojingjie, wangjiong} @iie.ac.cn

***Abstract*—Traditional attributed network embedding methods are designed to map structural and attribute information of networks jointly into a continuous Euclidean space, while recently a novel branch of them named binarized attributed network embedding has emerged to learn binary codes in Hamming space, aiming to save time and memory costs and to naturally fit node retrieval task. However, current binarized attributed network embedding methods are scarce and mostly ignore the local attribute similarity between each pair of nodes. Besides, none of them attempt to control the independency of each dimension(bit) of the learned binary representation vectors. As existing methods still need improving, we propose an unsupervised Neural-based Binarized Attributed Network Embedding (NBANE) approach. Firstly, we inherit the Weisfeiler-Lehman proximity matrix from predecessors to aggregate high-order features for each node. Secondly, we feed the aggregated features into an autoencoder with the attribute similarity penalizing term and the orthogonality term to make further dimension reduction. To solve the problem of integer optimization we adopt the relaxation-quantization method during the process of training neural networks. Empirically, we evaluate the performance of NBANE through node classification and clustering tasks on three real-world datasets and study a case on fast retrieval in academic networks. Our method achieves better performance over state-of-the-art baselines methods of various types.**

***Index Terms*—Attributed Network, Network Embedding, Weisfeiler-Lehman Proximity Matirx, Autoencoder**

I. Introduction

Attributed networks are one of the most ubiquitous data structures in real-world information systems such as social networks, academic networks [1] and knowledge graphs [2], consisting of a group of nodes associated with features linked to one another. For instance, people, personal information and their friendships can be modeled as nodes, features and links respectively. To utilize such non-structural data in different scenarios like data mining and machine learning, network embedding method, also known as network representation, is proposed to covert nodes into a latent space for down-stream tasks. In other words, distinct nodes are mapped to different low-dimensional vectors reflecting similarities among nodes, on which applications such as node classification [3], node clustering [4], link prediction [5] and node retrieval [6] can be built.

* Corresponding author.

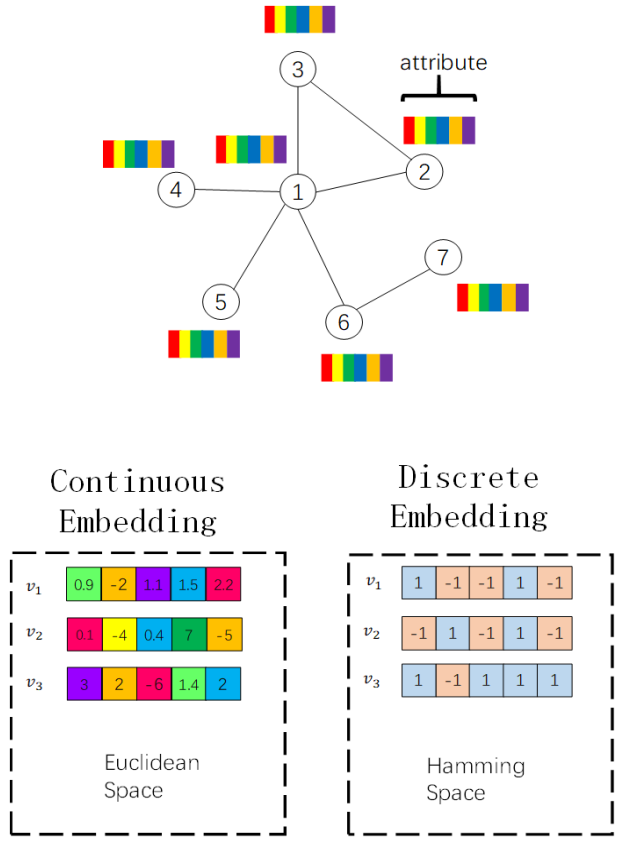


Fig. 1. Continuous embedding vs Discrete/Binarized embedding on attributed networks.

For a few years, many efforts have been made to address the problem of network embedding. Early models like DeepWalk [7], LINE [8], node2vec [9], SDNE [10] and GraRep [11] aim to learn network embedding vectors from structure information purely. Meanwhile, many models are developed to extract representations from structure and attribute information of networks jointly. For example, TADW [6], LANE [12], NANE [13], ANRL [14], GCN [15], GraphSAGE [16], GraphRNA [17] and many other classical algorithms aim to generate node embedding vectors preserving both structural relationships and attributes. However, all algorithms mentioned above output

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

real value vectors in a continuous Euclidean space, demanding great cost in computation and storage. Besides, a counter-intuitive fact is that continuous value embedding vectors can be noisy, redundant and over-fitted, which may result in bad performance in down-stream tasks.

In more recent years, building discrete especially binarized network embedding algorithms has been put on the agenda by researchers. For example, Bernoulli Embedding [18], DNE [19], BANE [20] and BinaryNE [21] find ways to map nodes into Hamming space where vectors only take value of $\{1, -1\}$ in each dimension, as shown in Fig.1. Binarized embedding is also called hashing embedding with some outstanding advantages. (1) Firstly, vectors that only take value of 1 or -1 in each dimension naturally take up rather smaller space for storage. (2) Moreover, when comparing the similarity between two nodes, it is much faster to calculate the Hamming distance with bit-wise operations than Euclidean distance with arithmetic operations between two corresponding binary codes. (3) Last but not the least, as node embedding vectors can be regarded as parameters, binary vectors produced with proper optimization are able to alleviate the over-fitting problem and to boost accuracy in many application scenarios like node classification and link prediction.

However, to the best of our knowledge, existing binarized attributed network embedding models are scarce and incomplete. Taking state-of-the-art models as example, BANE ignores local attribute similarity between two nodes when generating their embedding vectors, not able to boot node retrieval tasks. BinaryNE is more prone to capturing the global or high-order structural and attribute similarity by applying random walk techniques, not able to weigh local attribute similarity either. Besides, none of the binarized attributed network embedding models mentioned above attempt to control the independency of each bit(dimension) of vectors, resulting in the redundancy of node representations even though they are already in the form of binary codes. In a word, current algorithms still need improving and how to learn compact binary embedding vectors preserving global and local information enabling both data mining tasks and efficient node retrieval applications remains a challenge.

To modify current binarized attributed network embedding algorithms, we propose an unsupervised **N**eural-based **B**inarized **A**ttributed **N**etwork **E**mbedding framework named **NBANE**. Our model consists of two steps: (1) Aggregating nodes' attribute information from neighbors into features by using the Weisfeiler-Lehman proximity matrix; (2) Using an autoencoder to reduce the dimension of features while applying the constraints of local attribute similarity, the quantization loss , the bits independency loss to train the neural network. By taking the above elements into consideration, our model can outperform former works with both accuracy of node classification and scalability of node retrieval.

The contributions of this paper are listed as follows:

- To our best knowledge, our model is the first endeavor to introduce neural network methods into binarized attributed network embedding while simultaneously taking attribute similarity and bit othorgonality into consideration.
- We conduct experiments on three real-world datasets with node classification and node clustering tasks. By comparing the results with that of other models, the effectiveness of **NBANE** is well evaluated.
- We study a case on how to conduct fast node retrieval with nodes' binary codes, demonstrating the ability of saving time and memory of our model.

The remainder of the paper is organized as follows. In Section II, more related methodologies on network embedding and deep hashing are introduced as supplements. Preliminaries are given in Section III. We theoretically discuss our model in Section IV and empirically conduct experiments in Section V,VI. At last, conclusions based on theories and experiments are given in Section VII.

II. Related Work

A. Continuous Network Embedding

Since most network embedding methods are of this category, we mainly introduce representative ones among them. According to whether node attributes are taken into consideration, continuous network embedding algorithms fall into two categories: structure-based network embedding and attributed network embedding.

Structure-based network embedding methods work on plain networks without node contents, seeking to get nodes' representations that reflect topological similarity with a certain metric. DeepWalk [7] first employs the random walk and the Skip-Gram techniques to learn node representations preserving structural context similarity. The node2vec [9] algorithm extends DeepWalk by leveraging biased random walks. LINE [8] models the first-order proximity and the second-order proximity with the concatenation operation, followed by GraRep [11] capturing high-order proximity through matrix factorization. SDNE [10] uses a deep autoencoder to learn deep nonlinear node representations, by reconstructing node adjacent matrix for preserving the global proximity and penalizing the representation difference of connected nodes for preserving the first-order proximity.

Attributed network embedding methods learn node representations from network structure and attributes jointly. Based on the proof of DeepWalk's equivalence of matrix factorization, TADW [6] incorporates text features into its framework. LANE [12] also takes matrix factorization method on networks with node features and sparse labels to acquire representations. DANE [22] proposes dual autoencoders to deal with structure and attribute information respectively while NANE [13] uses only one autoencoder for early fusion of multi-modal information. ANRL [14] is set as a muilti-task neural network to learn embedding from hidden layers by reconstructing node attributes and node context. Besides, graph neural networks models like GCN [15], GraphSAGE [16], GAT [23] and GraphRNA [17] conduct semi-supervised end-to-end graph learning and offer node embeddings in the hidden layers naturally.

B. Discrete Network Embedding

Discrete network embedding aims to learn vectors with values restricted on a finite element set, and more typically a branch of it that merely takes value of $\{1, -1\}$ in Hamming space is called binarized network embedding. Bernoulli Network Embedding [18] generates binary codes as node presentations by conducting Bernoulli random tests. DNE [19] learns binary node representations to speed up node classification with matrix factorization. NetHash [24] is the first algorithm proposed to generate discrete node representations that encode both network structure and node content features with MinHash techniques. BANE [20] defines the Weisfeiler-Lehman proximity matrix and gets node embeddings by factorizing matrix with integer optimization, followed by LQANR [25] acquiring low-bit representations under a similar framework. BinaryNE [21] uses an attributed random walk and the continuous approximation technique to learn binary embedding vectors for nodes which can be regarded as an extension of DeepWalk.

C. Deep Hashing Methods

In other pattern recognition domains such as image, text and video, deep hashing methods have been widely adopted to extract features and compact binarized codes from high-dimensional raw data. They can be further categorized into unsupervised type and supervised type. The unsupervised deep hashing methods like DH [26] uses reconstruction autoencoder to preserve information. The supervised deep hashing methods try to utilize labels in the form of pairwise labels and triplets labels, like DPSH [27] and DSCH [28]. Deep hashing methods have great implications for the binarized network embedding task.

III. Preliminaries

In this section, we give the formal definitions and notations for the problem. Furthermore, a mathematical tool named Weisfeiler-Lehman proximity matrix is illustrated as it plays a crucial part in the proposed model.

A. Problem Definition

Given an attributed network $G = \{V, E, X\}$, $V = \{v_i\}_{i=1}^n$ denotes nodes, and $E = \{e_{ij}\}_{i,j=1}^n$ denotes undirected edges, and $X = \{x_i\}_{i=1}^n \in R^{n\times m}$ denotes attribute vectors of the nodes with m the dimension of attribute vectors. We define an adjacency matrix A on the basis of E , where $A_{ij} = 1$, if $e_{ij} \in E$, otherwise, $A_{ij} = 0$. Define $\tilde{A} = A + I$ by adding a self-loop to each node in the network, where I is an identity matrix. $\tilde{D} = diag(\tilde{d}_1, ..., \tilde{d}_n)$ is a degree matrix of $\tilde{A}$, with $\tilde{d}_i = \sum_j \tilde{a}_{ij}$ being the degree of node v_i.

Given the attributed network G, our goal is to embed each node $v_i \in V$ into a d-dimensional vector $b_i \in \{+1, -1\}^{1\times d}$in Hamming space, where b_i is the i^{th} row of matrix $B \in R^{n\times d}$. The target matrix B ought to preserve the structure information A and the attribute information X in the original network G at the same time.

B. Weisfeiler-Lehman Proximity Matrix

The Weisfeiler-Lehman proximity matrix is a useful tool to generate hand-crafted features for each node in a network. It is derived from Weisfeiler-Lehman graph kernal and has been already adopted by the former work BANE [20]. Given a network G with its adjacency matrix A and attribute matrix X, letting $\tilde{D}$ be a degree matrix of $\tilde{A}$ and $\tilde{L} = \tilde{D} - \tilde{A}$, the Weisfeiler-Lehman proximity matrix P is defined as $P = (I - \gamma\tilde{D}^{-1}\tilde{L})^k X$, where $\gamma \in [0, 1]$ is a trade-off parameter, and k is the number of aggregation layers.

C. Notations

TABLE I
MAIN NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
n	Total number of nodes in the network
m	Dimension of node attribute
d	Dimension of embedding space
γ	Trade-off parameter for the Weisfeiler-Lehman proximity matrix
k	Number of feature aggregation layer
G	Attributed network
A	Adjacent matrix for networks
E	Edges for networks
X	Attribute matrix for networks
P	Weisfeiler-Lehman proximity matrix
W_{in}, W_{out}	Weights of neural networks
S	Attribute cosine matrix
R	Real-number embedding before quantization
B	Binary embedding after quantization

IV. Our Model

In this section, we give an elaborate description of our model, including the feature aggregation part and the autoencoder part. Also, a series of loss functions for training the neural network are explained in detail.

A. Overall Architecture

Here we give a global introduction to the unsupervised **NBANE** framework, which is designed to learn binary embedding vectors for nodes in an attributed network. As shown in Fig.2, the framework is composed of two parts: the feature aggregation step and the feature transformation step. First we manually compute the Weisfeiler-Lehman proximity matrix with each row as the feature of the corresponding node, fusing both structure and attribute information. Second, we utilize an autoencoder to get embedding outcomes from its hidden layer. The following subsections depict every step of the algorithm.

B. Feature Aggregation with Weisfeiler-Lehman Proximity Matrix

Extracting nodes features preserving structural and content information is the basis of attributed network embedding. There are many existing methods for structure-attribute fusing. We inherit the Weisfeiler-Lehman proximity matrix from BANE model to achieve such a goal. Given a network G with

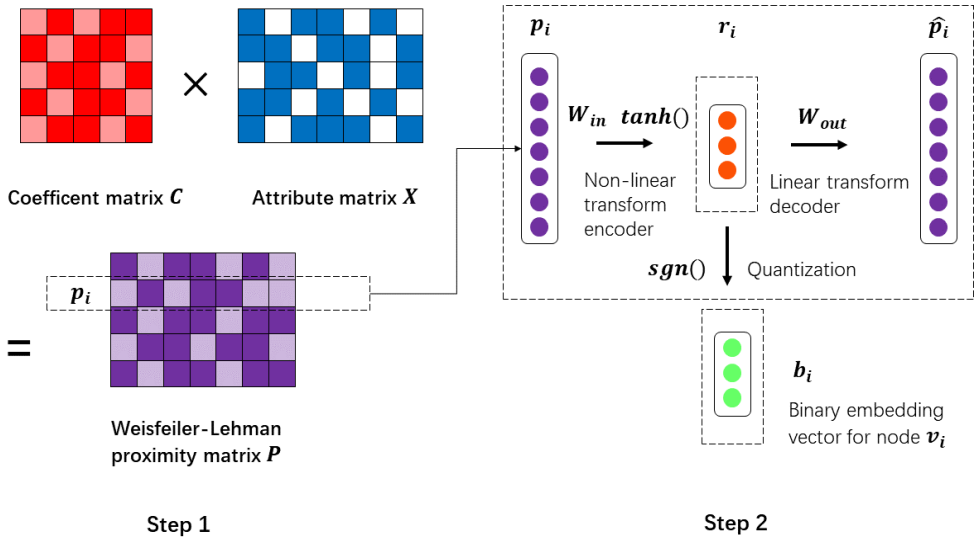


Fig. 2. Overall architecture of NBANE

its adjacent matrix A along with $\tilde{D}$, $\tilde{L}$, γ, k and attribute matrix X, we first compute the coefficient matrix

$$C = (I - \gamma \tilde{D}^{-1}\tilde{L})^k \tag{1}$$

Then Weisfeiler-Lehman proximity matrix is computed by

$$P = CX = (I - \gamma \tilde{D}^{-1}\tilde{L})^k X \in R^{n \times m} \tag{2}$$

As $P = [p_1; p_2; ...p_n]$, we take each row p_i of P as the aggregated feature of the corresponding vertex v_i. Note that the feature aggregation process is actually equivalent to linear combination of the original features $X = [x_1; x_2; ...x_n]$ with coefficient matrix $C = (I - \gamma \tilde{D}^{-1}\tilde{L})^k = \{c_{ij}\}_{n \times n}$. Then for each node v_i in the attributed network, we have the expression of its features

$$p_i = \sum_{j=1}^{n} c_{ij} x_j \tag{3}$$
$$i = 1, 2, ...n$$

From Fig.3 we can see that the hyper parameter k denotes the number of layers of neighboring nodes joining the information aggregation. On the other hand, $\gamma \in [0, 1]$ makes a trade-off between structure and attribute information. The larger γ is, the more information on structure is taken into account, and vice versa.

C. Autoencoder with Pair-Wise Attribute Similarity Restriction

By computing the Weisfeiler-Lehman proximity matrix, we encode structure and attribute information of each node v_i into the aggregated feature p_i. However, the aggregated features are still high in dimension and sparse, unable to be used as node representations. We hope that the target binarized embedding vectors for nodes have the following properties.

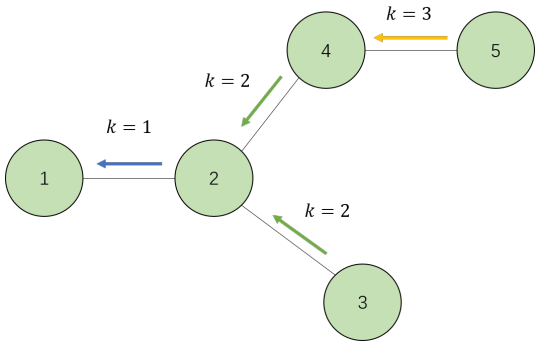


Fig. 3. Spatial explanation for feature aggregation.

- 1.Binarized embedding vectors should be low dimensional and dense.
- 2.Binarized embedding vectors should preserve structure and attribute information consistent with the aggregated features computed with the Weisfeiler-Lehman proximity matrix.
- 3.Binarized embedding vectors of two nodes with similar attribute should be close in Hamming space.

Therefore, we present an autoencoder consisting of an input layer, a hidden layer and an output layer with penalizing constraints to meet the above requests. The forward propagation consists of the encoding and the decoding process. At the encoding step, the aggregated feature p_i is input into the autoencoder to compute the hidden layer vectors as the binarized embedding result $B = \{b_i\}_{i=1}^{n} \in \{1, -1\}^{n \times d}$, and to reconstruct the output layer as follows

$$\begin{aligned} b_i &= tanh(W_{in} \cdot p_i) \\ \hat{p_i} &= W_{out} \cdot b_i \end{aligned} \tag{4}$$

where W_{in} and W_{out} are weight matrices of the neural network. Note that no bias involves the computation and the output layer does not need activation functions.

The output of the autoencoder is denoted as $\hat{P}$, namely the reconstruction layer of input data P. In order to preserve structure and attribute information in the hidden layer, we set reconstruction to guide the training of the neural network. The loss function is denoted with the Frobenius norm $||\cdot||_F$ as:

$$L_{recon} = ||(P - \hat{P}) \odot M||_F^2 \tag{5}$$

where M is the offset coefficient vector. $M_{ij} = 1$ when $P_{ij} = 0$ while $M_{ij} = \beta > 1$ when $P_{ij} \neq 0$, and $\odot$ means the Hadamard product. The offset coefficient vector is used to alleviate data sparsity of P.

In addition, we enhance the attribute similarity between each pair of nodes by adding a pair-wise penalizing term

$$\begin{aligned} L_{attr} &= \sum_{i,j=1}^{n} s_{ij} ||b_i - b_j||_H \\ & b_i, b_j \in \{1, -1\}^{1\times d} \end{aligned} \tag{6}$$

where $||\cdot||_H$ denotes the Hamming distance between two binary codes and $s_{ij} = cos(x_i, x_j)$ is the cosine similarity between attributes of two nodes v_i and v_j.

To make embedding vectors compact and bit-independent as well as to alleviate over-fitting problem, we go a step further to control the orthogonality of weights matrix W_{in} by appending term $||W_{in}W_{in}^T - I||_F^2$. The orthogonality loss is

$$L_{ortho} = ||W_{in}W_{in}^T - I||_F^2 \tag{7}$$

D. Relaxation-Quantization Method

The optimization of loss function (6) and the final joint loss function is an NP-hard problem. In order to make the problem optimizable by the gradient descendent method, we adopt the relaxation-quantization method [27]. The idea is to relax the value of b_i from $\{1, -1\}$ to real number r_i for optimization process and use the sign function $sgn(\cdot)$ to get the final binary codes, named relaxation step and quantization step respectively. Because the Hamming distance is proportional to the Euclidean distance, we replace $||\cdot||_H$ with $||\cdot||_2$. A quantization loss function is needed to make the real-number value close to its quantized result through iteration, defined as

$$L_{quant} = \sum_{i=1}^{n} ||r_i - b_i||_2^2 = \sum_{i=1}^{n} ||r_i - sgn(r_i)||_2^2 \tag{8}$$

where $sgn(\cdot)$ is the sign function with the threshold zero.

After taking all loss functions into consideration and applying relaxation-quantization method, we rewrite the joint loss function for the **NBANE** model as:

$$\begin{aligned} L &= \alpha_1 L_{recon} + \alpha_2 L_{attr} + \alpha_3 L_{ortho} + \alpha_4 L_{quant} \\ &= \alpha_1 ||(P - \hat{P}) \odot M||_F^2 + \alpha_2 \sum_{i,j=1}^{n} s_{ij} ||r_i - r_j||_2^2 \\ &+ \alpha_3 (||W_{in}W_{in}^T - I||_F^2) + \alpha_4 \sum_{i=1}^{n} ||r_i - sgn(r_i)||_2^2 \end{aligned} \tag{9}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are hyper-parameters to adjust the weights of each part. Finally we compute $b_i = sgn(r_i)$ to get the final binary embedding result for node v_i.

We apply the RMSProp optimizer to minimize the loss function in Eq.(9). Algorithm.1 shows the entire process.

Algorithm 1 NBANE model

Input:

The network G with its adjacent matrix A and attribute matrix X

Output:

Binary network embedding B

1: Compute the Weisfeiler-Lehman proximity matrix P with Eq.(2) and the attribute cosine similarity matrix S;
2: Feed each row p_i of P into the autoencoder;
3: **repeat**
4: Apply Eq.(4) to generate the hidden layer R and the reconstruct matrix $\hat{P}$;
5: Minimize Eq.(9) by RMSProp to update weights of the neural network;
6: **until** convergence
7: Acquire the hidden layer R;
8: Get the binary embedding B by computing $B = sgn(R)$;
9: **return** B

E. Relation with BANE model

Theoretically, our model can be regarded as a neural-form extension of BANE [20], which proposes and factorizes the Weisfeiler-Lehman proximity matrix P into a binary matrix $B \in \{1, -1\}^d$ and an auxiliary matrix Z with $P = BZ$. Defining Z^{-1} as the pseudo inverse matrix of Z , we can see that the embedding result of BANE is an approximate linear transform outcome of $B = PZ^{-1}$, equivalent to minimizing $||P - BZ||_F$. However, **NBANE** steps further to minimize $w_1||P - BZ||_F + w_2||S^T BS||_F$ to enhance attribute similarities between each pair of nodes. In other words, BANE is a special case of **NBANE** when $w_2 = 0$. Thus **NBANE** can cover the BANE model and improve efficiencies of various tasks.

V. Experimental Configuration

In this section, we introduce the datasets, experiment schemes and baselines that validate the effectiveness of the proposed algorithm.

A. Datasets

Three real-world attributed networks are selected for experiments:

- **Cora**. The Cora network is an academic network composed of 2,708 machine learning publications and their citation relationships. Theses publications are categorized into 7 groups. Each publication is represented by a 1,433-dimensional binary vector, with each dimension denoting the presence/absence of the corresponding word.

- **Hamilton** and **Stanford**. They are two social networks among dataset Facebook100 collected by Adam D'Angelo of Facebook. Each node contains 7 attributes: student/faculty status flag, gender, major, second major/minor, dorm/house, year, and high school, which is described by a 139-dimensional and 235-dimensional feature vector respectively, and student/faculty status flag is selected as label. Two datasets include 2,314 nodes, 96,394 edges and 11,621 nodes, 568,330 edges separately.

The statistics of three datasets are presented in Table II.

TABLE II
THE STATISTICS OF THE THREE DATASETS

Dataset	Node	Edge	Attribute
Cora	2078	5429	1433
Hamilton	2314	96394	139
Stanford	11621	568330	235

B. Evaluation Scheme

- **Node classification**. Node classification task aims to utilize a portion of network embedding vectors and their labels to train an SVM. Then it is used to predict the label of the rest of nodes .
- **Node clustering**. To test if the embedding result has spatial distribution characters, embedding vectors are put into a clustering algorithm to be grouped.
- **Node retrieval**. Given a query node, we use its binary embedding vector to compute similarities with other nodes and return the closest results. Besides, time and memory for retrieval are compared to demonstrate the efficiency of our model.

C. Baseline Methods

We compare our binarized attributed embedding model with the following baselines. Some of them are state-of-the-art models.

- **node2vec** [9]. As an extension of the classical DeepWalk model, node2vec assigns parameters p and q to make a trade-off between local and global information.
- **SDNE** [10]. SDNE is an autoencoder embedding framework with a first-order structure penalizing term to capture global and local structure similarities of nodes.
- **TADW** [6]. It is a matrix-factorization-based method to encode both structure and content information of nodes into their embedding vectors.
- **LANE** [12](unsupervised version). It fuses structural, attribute and label information together to preserve node similarity. Here we only use its unsupervised version.
- **ANRL** [14]. A state-of-the-art attributed network embedding framework that adopts an attributed random walk and recurrent neural network to learn embedding for nodes and graphs by concatenating hidden layers.
- **BANE** [20]. In essence BANE is a linear transformation to the Weisfeiler-Lehman proximity matrix of an attributed network. It iteratively optimizes variables to get binary embedding vectors for nodes.

Note that the selected baselines are of three types: (1) Unsupervised plain network embedding methods like **node2vec** and **SDNE**; (2) Unsupervised continuous attributed network embedding methods like **TADW**, **LANE** and **ANRL**; (3) Unsupervised binarized attributed network embedding methods like **BANE**. Other algorithms mentioned before like GCN, GraphSAGE and GraphRNA are excluded because they require labels to conduct semi-supervised learning.

VI. EXPERIMENTAL RESULTS

TABLE III
NODE CLASSIFICATION RESULT MICRO F1-SCORE(%)

Dataset	Train size	10%	30%	50%	70%	90%
Cora	node2vec	74.67	80.31	83.10	83.79	86.57
	SDNE	36.96	52.03	56.81	60.12	64.58
	TADW	78.33	83.09	86.43	86.12	88.56
	LANE	70.15	80.50	82.73	84.55	87.16
	ANRL	71.03	73.18	75.81	75.84	79.41
	BANE	80.34	85.55	87.74	87.26	90.03
	NBANE	**82.06**	**85.73**	**87.99**	**88.41**	**91.81**
Hamilton	node2vec	89.05	89.84	90.86	92.03	91.90
	SDNE	82.77	87.01	88.31	89.41	91.03
	TADW	89.07	91.85	92.84	95.02	95.17
	LANE	79.64	79.98	84.89	92.89	95.34
	ANRL	80.92	89.40	91.53	93.12	94.31
	BANE	88.72	91.41	92.32	92.81	92.76
	NBANE	**94.67**	**95.26**	**95.56**	**96.29**	**97.16**
Stanford	node2vec	74.69	79.91	81.43	81.78	83.23
	SDNE	63.47	63.37	63.50	63.68	63.70
	TADW	76.87	80.72	82.00	82.97	84.42
	LANE	63.47	63.55	63.70	63.93	63.75
	ANRL	80.23	80.65	81.72	81.79	82.77
	BANE	81.58	82.37	82.62	83.48	84.75
	NBANE	**82.41**	**83.20**	**83.51**	**84.15**	**85.59**

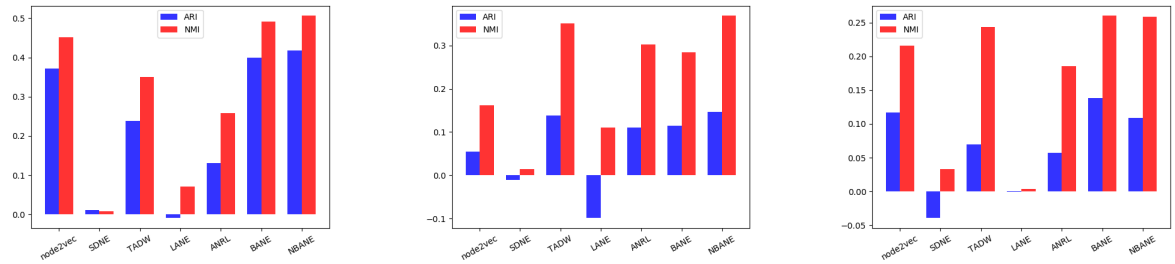
A. Node Classification Result

For the node classification task, we regard the learned node representations as samples and divide them into training set and testing set. We range the train size from 10% to 90% by taking 20% as a step and use an rbf-SVM to make classification. For fairness, we set the embedding dimension to 256 for all methods and conduct classification task under the same division of training and testing of data. We repeat the process for 5 times and report the average micro F1-score. Table.III shows the outcome of node classification.

The results can be summarized as follows. First, as a neural-based model with attribute similarity and weights orthogonality constraints, **NBANE** performs better than baselines under most settings. Second, the results show that binary representations do not necessarily lead to accuracy loss but may avoid the trap of over-fitting, a fact that has been proved by BANE and further validated by our **NBANE**.

B. Node Clustering Result

For the node clustering experiment, we utilize the K-Means++ algorithm and evaluate the clustering result with ARI(Adjusted Rand Index) and NMI(Normalized Mutual Information). We set the number of clusters same as the group

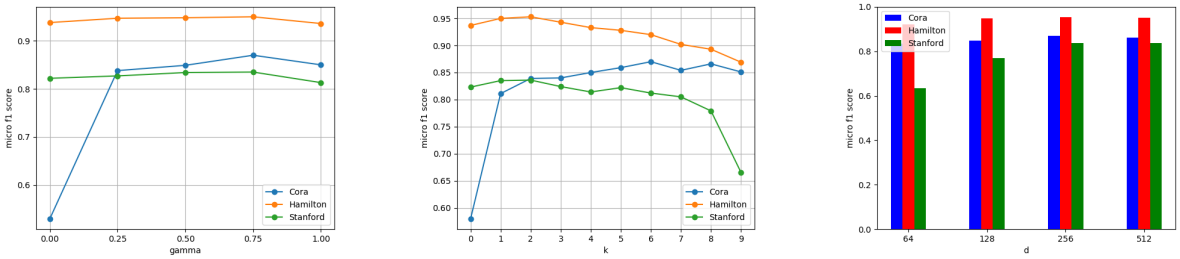


(a) Cora (b) Hamilton (c) Stanford

Fig. 4. Node clustering result on three datasets

TABLE IV
TOP-5 QUERY RESULTS ON CORA

Query: Back propagation is sensitive to initial conditions	**Runtimes**	**Memory**
TADW 1. Neural network implementation in SAS software. $\surd$ 2. Hyperplane "spin" dynamics, network plasticity and back-propagation learning. $\surd$ 3. Learning visual schemas in neural networks for object recognition and scene analysis. $\surd$ 4. Generative learning structures for generalized connectionist networks.$\surd$ 5. Neural networks and statistical models. $\surd$	79.63 ms $O(d)$ arithmetic operation	5.54M
NBANE 1. Neural network implementation in SAS software. $\surd$ 2. Hyperplane "spin" dynamics, network plasticity and back-propagation learning. $\surd$ 3. Learning visual schemas in neural networks for object recognition and scene analysis.$\surd$ 4. Visualizing high-dimensional structure with the incremental grid growing neural network.$\surd$ 5. Introduction to the theory of neural computation.$\surd$	4.27 ms $O(\tau d)$ bit operation	23.73KB



(a) γ (b) k (c) d

Fig. 5. Parameter sensitivity analysis

number and calculate the indexes 5 times to reduce occasionality. The average results are displayed in Fig.4. Obviously we can see that **NBANE** stands the best among almost all models.

C. Case Study: Fast Node Retrieval with Binary Codes

In this case study, we retrieve 5 most similar papers for a query based on binarized embedding vectors of Cora. The retrieval result is shown in Table.IV where $\surd$ marks papers with the same type of the query paper. Both models are compared under the same environment. Obviously our model saves much more time and storage than TADW [6]. The reason can be explained by computational complexity. When using continuous d-dimensional vectors to compute similarities with Euclidean norm, we need arithmetic operations with $O(d)$ time complexity. However, getting Hamming distances between each pair of nodes only requires bit operations with $O(\tau d)$ time complexity along with a coefficient $\tau \in (0, 1)$. Since bit operations are much more faster than arithmetic operations, **NBANE** saves much more time than traditional embedding methods.

D. Parameter Sensitivity

In this part we select three crucial hyper-parameters γ, k and d to explore how they influence the classification accuracies. We fix the training ratio as 50% and alter these three hyper-parameters respectively to observe how classification accuracy changes accordingly. Fig.5 shows the results.

VII. Conclusion and Future Work

In this paper we study the problem of binarized attributed network embedding with neural networks and propose **N-BANE** model. On the basis of predecessors, we borrow the Weisfier-Lehman proximity matrix to jointly encode structure and attribute information into features. Based on the aggregated features, we set up an autoencoder model along with the attribute similarity constraint, the bit independency constraint to obtain binary node representations. We also utilize the relaxation-quantization method to overcome the difficulty of integer optimization. Empirical results validate the effectiveness of our model. In the future, we will consider to design a cross-modal binarized attributed network embedding framework to learn binary representations not only for nodes but also for attributes and labels, which guarantees retrieval tasks among all kinds of information sources.

Acknowledgement

This work is supported by the National Key Research and Development Program of China.

References

[1] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[2] J. Mo, N. Gao, Y. Zhou, Y. Pei, and J. Wang, "Translation-based attributed network embedding," in *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece*, pp. 892–899, 2018.

[3] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*, pp. 115–148, 2011.

[4] E. Stattner and M. Collard, "Clustering of links and clustering of nodes: Fusion of knowledge in social networks," in *Advances in Knowledge Discovery and Management - Volume 6 [Best of EGC 2014, Rennes, France / Best of EGC 2015, Luxembourg]*, pp. 255–276, 2015.

[5] D. Liben-Nowell and J. M. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pp. 556–559, 2003.

[6] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2111–2117, 2015.

[7] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 701–710, 2014.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 1067–1077, 2015.

[9] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 855–864, 2016.

[10] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1225–1234, 2016.

[11] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pp. 891–900, 2015.

[12] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pp. 731–739, 2017.

[13] J. Mo, N. Gao, Y. Zhou, Y. Pei, and J. Wang, "NANE: attributed network embedding with local and global information," in *Web Information Systems Engineering - WISE 2018 - 19th International Conference, Dubai, United Arab Emirates, November 12-15, 2018, Proceedings, Part I*, pp. 247–261, 2018.

[14] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, "ANRL: attributed network representation learning via deep neural networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 3155–3161, 2018.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[16] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

[17] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 732–740, 2019.

[18] V. Misra and S. Bhatia, "Bernoulli embeddings for graphs," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3812–3819, 2018.

[19] X. Shen, S. Pan, W. Liu, Y. Ong, and Q. Sun, "Discrete network embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 3549–3555, 2018.

[20] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pp. 1476–1481, 2018.

[21] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Search efficient binary network embedding," *CoRR*, vol. abs/1901.04097, 2019.

[22] H. Gao and H. Huang, "Deep attributed network embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 3364–3370, 2018.

[23] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[24] W. Wu, B. Li, L. Chen, and C. Zhang, "Efficient attributed network embedding via recursive randomized hashing.," in *IJCAI*, vol. 18, pp. 2861–2867, 2018.

[25] H. Yang, S. Pan, L. Chen, C. Zhou, and P. Zhang, "Low-bit quantization for attributed network representation learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4047–4053, 2019.

[26] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 2475–2483, 2015.

[27] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 1711–1717, 2016.

[28] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.