Binarized Attributed Network Embedding

Hong Yang †, Shirui Pan †* , Peng Zhang ‡, Ling Chen †, Defu Lian ¶, Chengqi Zhang †
† Centre for Artificial Intelligence, School of Software, FEIT, University of Technology Sydney
‡ Ant Financial; ¶ University of Electronic Science and Technology of China
{hong.yang@student., shirui.pan@, ling.chen@, chengqi.zhang@}uts.edu.au;
zhangpeng04@gmail.com; dove@uestc.edu.cn

***Abstract*—Attributed network embedding enables joint representation learning of node links and attributes. Existing attributed network embedding models are designed in continuous Euclidean spaces which often introduce data redundancy and impose challenges to storage and computation costs. To this end, we present a *Binarized Attributed Network Embedding* model (*BANE* for short) to learn binary node representation. Specifically, we define a new *Weisfeiler-Lehman proximity matrix* to capture data dependence between node links and attributes by aggregating the information of node attributes and links from neighboring nodes to a given target node in a layer-wise manner. Based on the Weisfeiler-Lehman proximity matrix, we formulate a new *Weisfiler-Lehman matrix factorization* learning function under the binary node representation constraint. The learning problem is a *mixed integer optimization* and an efficient *cyclic coordinate descent* (CCD) algorithm is used as the solution. Node classification and link prediction experiments on real-world datasets show that the proposed BANE model outperforms the state-of-the-art network embedding methods.**

***Index Terms*—Attributed network embedding, Weisfeiler-Lehman graph kernels, Learning to hash.**

I. Introduction

Attributed networks are popularly used to describe a large body of networks where both node links and attributes are observable for analysis. Applications of attributed networks range from social networks, academic citation networks, to protein-protein interaction networks.

In order to drill hidden patterns from attributed networks, network embedding models such as Deepwalk [12], node2vec [4] and LINE [17] project node links into low-dimensional vectors. Then, the projected vectors are linearly concatenated with node attribute vectors to represent the nodes for subsequent data mining tasks such as network recommendation. However, this type of network embedding falls into a two-stage learning category, where node links are vectorized independently without using any auxiliary information from node attributes. Thus, they are incapable of capturing data dependence between node links and attributes, and are often referred to as *plain network embedding*.

To enable exploitation of the dependence information between node links and attributes, *attributed network embedding* models are proposed to jointly learn from node links and attributes. The principle behind is to use node attributes as class labels to supervise structure learning from node links, or vice versa. For example, the work [21] uses textual attributes to supervise random walks on networks and derives the Text-associated DeepWalk (TADW) model. On the contrary, the work [6] reversely uses node links to supervise the factorization of attributed proximity matrices. The work [7] mutually uses node links and attributes as labels to supervise the learning from each other. As a result, attributed network embedding generally outperforms plain network embedding by considering data dependence between attributes and links.

Existing attributed network embedding models are developed in continuous Euclidean spaces. By embedding the dependence information of node attributes and links, the learned vectors may contain redundant information that degenerates computation efficiency and increases storage cost, especially when networks are very large. Imagining the task of k-nearest neighbor search to recommend top-k most similar friends in a large network of size n, assuming the latent vector is of length d, the similarity search will take time $O(n^2d)$. Thus, we prefer succinct (binary) node representation for fast node recommendation.

Binary code learning [18] can generate succinct representation. The idea is to encode high-dimensional data into a set of short binary codes with similarity preservation. Binary coding is also referred to as *hashing* which maps data to discrete Hamming space [19]. The binary codes can facilitate to represent and search of massive data because it only needs about one hundred binary bits to represent one data item, and binary computation in Hamming space is efficient by using the bit operations. Many learning-based hashing algorithms have been developed according to different scenarios, including the unsupervised methods [9], supervised methods [13], deep learning based hashing methods [14]. To the best of our knowledge, no prior studies have been focused on seeking binary representation for attributed network to preserve both network structure and node attributes.

In this work, we study the problem of binarized attributed network embedding. The key challenge is *how to aggregate the information of both node links and attributes for binary node representation learning*. Considering matrix factorization as the embedding framework, as popularly used in the previous work [6], [7], [21], we summarize challenges as follows,

- *Challenge 1*: how to design a proximity matrix to capture data dependence between node links and attributes in attributed networks. To our best knowledge, none of existing network proximity matrices encodes both node links and attributes.

*Corresponding author.

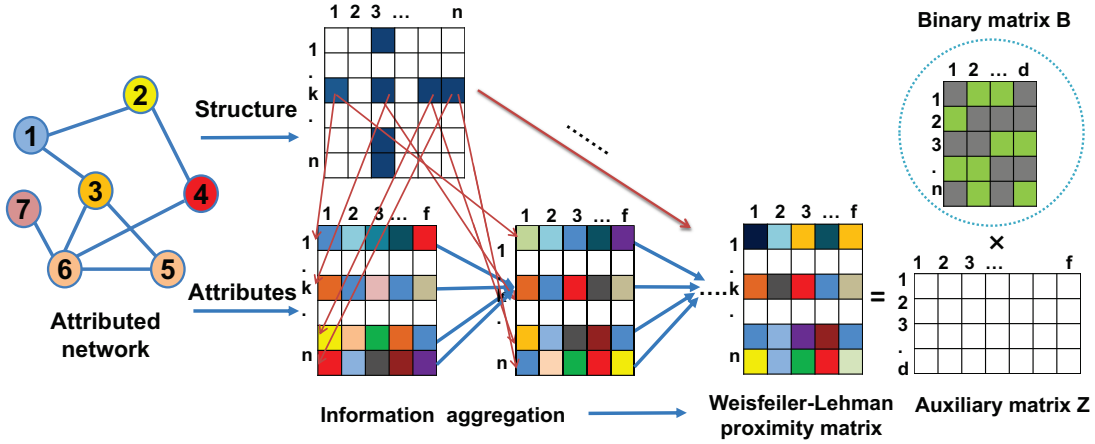

Figure 1. The conceptual framework of the *Binarized Attributed Network Embedding (BANE)*. Given an attributed network $G = \{V, E, X\}$, derive a *Weisfeiler-Lehman proximity matrix* $P = (I - \gamma\tilde{D}^{-1}\tilde{L})^k X$ by aggregating information from both structure matrix A and attribute matrix X. Factorizing matrix P into a binary node representation matrix B and an auxiliary matrix Z.

- *Challenge 2*: how to design a fast algorithm to solve the binary representation problem. Existing models for embedding attributed networks are formulated in Euclidean spaces. However, factorizing the proximity matrix under binary constraints falls into the integer programming category which requires efficient algorithms.
- *Challenge 3*: how to empirically prove the effectiveness and efficiency of the model.

To solve the above challenges, we present a new *Binarized Attributed Network Embedding* model (*BANE* for short). Inspired by the *Weisfeiler-Lehman graph kernels* [16] [8], we define a new *Weisfeiler-Lehman proximity matrix* to capture data dependence between node links and attributes. Then, based on the new proximity matrix, we formulate a *Weisfiler-Lehman matrix factorization* learning function under the binary representation constraint. The learning problem falls into the category of *mixed integer optimization* and we use an efficient *cyclic coordinate descent* (CCD) algorithm [13] as the solution. Experimental results on real-world datasets validate the performance of the proposed method. The framework of BANE is illustrated in Figure 1. The contribution of the paper is threefold:

- We first study the *binarized attributed network embedding* problem and present a new BANE model as the solution.
- We define a new *Weisfeiler-Lehman proximity matrix* to encode data dependence between node links and attributes, based on which a new *Weisfeiler-Lehman matrix factorization* is presented to learn binary representation.
- We conduct experiments to validate the performance of the proposed BANE model. The source codes are publicly available online.

II. Related Work

Attributed Network Embedding. Current network embedding methods can be categorized into *plain network embedding* [20] and *attributed network embedding* [1]. Different from plain network embedding that independently vectorizes node links without using auxiliary information from node attributes, attributed network embedding jointly models their dependence, by using node attributes as class labels to supervise the learning of node links, or vice versa. A typical attributed network embedding model is the TADW model [21] that uses textual attributes to supervise random walks on networks. Similar works include the TriDNR [11], Adversarially Regularized Graph Autoencoder (ARGA) [10], active network representation learning approach (ANRMAB) [3].

Learning to Hash. Hashing or binary coding [18] encodes high-dimensional feature vectors of documents, images and videos to compact binary codes, while preserving similarity structure in the original space. The binary codes can facilitate to represent and search of massive data because it only needs about one hundred binary bits to represent one data item, and binary computation in Hamming space is efficient by using the bit operations. Many learning-based hashing algorithms have been developed to different scenarios, including the unsupervised methods [9], supervised methods [13], deep learning based hashing methods [14]. A recent research learns discrete representation for plain networks [15]. In this paper, we will learn compact binary codes for attributed network embedding.

III. Problem Statement

An attributed network is represented as $G = \{V, E, X\}$, where $V = \{v_i\}_{i=1}^n$ denotes nodes, $E = \{e_{ij}\}_{i,j=1}^n$ denotes undirected edges, and $X = \{x_i\}_{i=1}^n \in R^{n\times f}$ denotes attribute vectors of the nodes with f the dimension of attribute vectors. In addition, the structure of network G can be derived from edges in E, denoted as an adjacency matrix A, where $A_{ij} = 1$, if $e_{ij} \in E$, otherwise, $A_{ij} = 0$. By adding a self-loop to each node in the network, we have $\tilde{A} = A + I$, where I is an identity matrix. $\tilde{D} = \text{diag}(\tilde{d}_1, ..., \tilde{d}_n)$ 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, we wish to embed each node $v_i \in V$ into a d-dimensional vector $b_i \in \{-1,+1\}^d$ in Hamming space, where b_i is the i^{th} row of matrix $B \in R^{n\times d}$. Ideally, matrix B can preserve the structure information A and the attribute information X in the original network G.

The key question is to design a proximity matrix that can jointly describe structure A and attribute X. For example, TADW [21] derives the proximity matrix by using the textual attributes X to supervise the random walk of structure A. The process can be taken as using the random walk kernel (supervised by node attributes) on graphs for node representation.

Instead of using random walk graph kernels, we use the Weisfeiler-Lehman graph kernels to generate a new proximity matrix P that encodes both node attributes in X and links in A. Specifically, we define the *Weisfeiler-Lehman proximity matrix* based on the Weisfeiler-Lehman graph kernels as follows,

Definition 1. (**Weisfeiler-Lehman Proximity Matrix**). Given a network G with adjacency matrix A and attribute matrix X, let $\tilde{D}$ 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 tradeoff parameter, and k is the number of aggregation layers.

The Weisfeiler-Lehman proximity matrix is based on the Weisfeiler-Lehman graph kernels [16] and thus naturally captures data dependence between node links and attributes. In particular, the proximity matrix has the following **properties**:

Property 1. The Weisfeiler-Lehman proximity matrix enables aggregation of node attributes and links from neighboring nodes to a target node. Parameter k controls the number of layers of neighboring nodes joining the aggregation. If $k = 1$ and $\gamma = 1$, matrix P equals the one-layer Weisfeiler-Lehman graph kernel (*Section IV.C* for details).

Property 2. The Weisfeiler-Lehman proximity matrix enables the tradeoff of node aggregation between neighboring nodes and a target node, where γ is the smoothing parameter.

IV. The Proposed Method

In this section, we first derive the learning function of *binarized Weisfeiler-Lehman matrix factorization*, and use the *Cyclic Coordinate Descent (CCD)* [13] algorithm as the solution to get binarized embedding. We further pinpoint the connection between the new proximity matrix and Weisfeiler-Lehman graph kernels.

A. Binarized Weisfeiler-Lehman Matrix Factorization

Based on Definition 1, we factorize the Weisfeiler-Lehman proximity matrix $P = (I-\gamma\tilde{D}^{-1}\tilde{L})^k X$ which jointly encodes node attributes and links into a binary node representation matrix B and an auxiliary matrix Z. Formally, the learning function of the binarized Weisfeiler-Lehman matrix factorization can be defined as follows,

$$\begin{aligned} \min_{B,Z} \ & \frac{1}{2}\|(I-\gamma\tilde{D}^{-1}\tilde{L})^k X - BZ\|_F^2 + \frac{\alpha}{2}\|Z\|_F^2, \\ s.t.: \ & B \in \{-1,+1\}^{n\times d}, Z \in R^{d\times f}, \end{aligned} \tag{1}$$

where α is a regularization parameter with respect to the auxiliary matrix Z. Due to the binary constraint with respect to matrix B, Eq.(1) is NP-hard. Next, we introduce efficient algorithms as the solution.

B. Algorithms

We propose an alternating algorithm to iteratively optimize each variable to solve the optimization problem in Eq.(1). The algorithm updates one parameter at a time and converges fast. We describe the details of the algorithm as follows,

Z-Step. Given B, solve the sub-problem with respect to Z in Eq.(1). The loss function can be written as follows,

$$\begin{aligned} \min_{Z} \ & \frac{1}{2}\|(I-\gamma\tilde{D}^{-1}\tilde{L})^k X - BZ\|_F^2 + \frac{\alpha}{2}\|Z\|_F^2, \\ & = -tr(P^T BZ) + \frac{1}{2}tr(Z^T B^T BZ) + \frac{\alpha}{2}tr(Z^T Z). \end{aligned} \tag{2}$$

Note $P = (I-\gamma\tilde{D}^{-1}\tilde{L})^k X$, and $tr(.)$ is the trace norm. By calculating the derivative of Eq.(2), we derive a closed form solution as follows,

$$Z = (B^T B + \alpha I)^{-1} B^T P. \tag{3}$$

B-Step. It is difficult to solve B due to the discrete constraint. Given Z fixed, rewrite the objective function in Eq. (1) with respect to B as follows,

$$\begin{aligned} \min_{B} \ & \frac{1}{2}\|(I-\gamma\tilde{D}^{-1}\tilde{L})^k X - BZ\|_F^2 \\ & = \frac{1}{2}tr(Z^T B^T BZ) - tr(B^T PZ^T), \\ s.t.: \ & B \in \{-1,+1\}^{n\times d}. \end{aligned} \tag{4}$$

Under the observation that *a closed-form solution for one column of B can be achieved by fixing all the other columns*, the algorithm iteratively learns one bit of B at a time.

Let b^l be the l^{th} column of B, and B' the matrix of B excluding b^l. Then, b^l is the one bit for all the n samples. Similarly, let q^l be the l^{th} column of $Q = PZ^T$, Q' the matrix of Q excluding q^l, z^l the l^{th} row of Z and Z' the matrix of Z excluding z^l. Then we obtain

$$tr(Z^T B^T BZ) = z^l Z'^T B'^T b^l + const. \tag{5}$$

Following the same logic, we obtain

$$tr(B^T Q) = (q^l)^T b^l + const. \tag{6}$$

Plugging Eqs.(5) and (6) back into Eq.(4), we obtain the optimization problem with respect to b^l as follows,

$$\begin{aligned} \min_{b^l} \ & z^l Z'^T B'^T b^l - (q^l)^T b^l \\ & = (z^l Z'^T B'^T - (q^l)^T) b^l \\ s.t.: \ & b^l \in \{-1,+1\}^{n\times 1} \end{aligned} \tag{7}$$

Algorithm 1 Binarized Attributed Network Embedding (BANE)

Input: Structure A, attribute X, dimension d, # of iterations t_1 and t_2, parameters k, γ, α
Output: Binary representation matrix B

```
Initialize Z, B randomly
Repeat until converge or reach t_1
Z-Step: Calculate Z using Eq.(3)
B-Step: Repeat until converge or reach t_2
for l = 1, ..., d do
    update b^l using Eq.(8)
end for
return matrix B
```

Eq.(7) has a closed form solution as follows,

$$b^l = sign(q^l - B'Z'(z^l)^T). \tag{8}$$

By using this method, each bit b can be computed based on the pre-learned $d-1$ bits of B'. The convergence of the alternating optimization is guaranteed theoretically, because every iteration decreases the objective function value and the objective function has a lower bound.

The details of the algorithm is given in Algorithm 1. Empirical results demonstrate that the algorithm takes a few iterations to converge. For example, in our experiments B is iteratively computed and the algorithm converges fast in about $3-10$ iterations.

C. Connection with Weisfeiler-Lehman Graph Kernels

The idea of aggregating information from neighboring nodes to a target node originated from the Weisfeiler-Lehman graph kernels [16], where the parameter k controls the layers of neighboring nodes joining the information aggregation. The original idea of the Weisfeiler-Lehman algorithm is to augment the node labels by the sorted set of node labels of their neighboring nodes, and then compress these augmented labels into new, short labels.

Theorem 1. Let $k=1$, $\gamma=1$, and $P=(I-\gamma\tilde{D}^{-1}\tilde{L})^k X$, then P is a one-layer Weisfeiler-Lehman graph kernel.

Proof. When $\gamma=1, k=1$, then $P=\tilde{D}^{-1}\tilde{A}X=P^{(1)}$. Let $h_i^{(k)}$ be the information of node v_i in the k-th iteration, and N_i be the neighbors of v_i. Define a linear aggregation function, integrating neighboring nodes' information and the target node's information under the Weisfeiler-Lehman algorithm, we can obtain the following information propagation rule,

$$h_i^{(k)} = h_i^{(k-1)} + \sum_{j\in N_i} h_j^{(k-1)}. \tag{9}$$

Such an information propagation rule can be further rewritten into a compact matrix form as follows,

$$H^{(k)} = \tilde{A}H^{(k-1)}, \tag{10}$$

where $\tilde{A}=A+I$, which adds a self-loop to each node in the network. $\tilde{D}=\text{diag}(\tilde{d}_1,...,\tilde{d}_n)$ is the degree matrix of $\tilde{A}$. Normalizing the matrix $\tilde{A}$ by its degree matrix $\tilde{D}$, we can obtain

$$H^{(k)} = \tilde{D}^{-1}\tilde{A}H^{(k-1)}. \tag{11}$$

At the beginning of the aggregation, i.e., $k=1$, $H^{(0)}=X$, then $H^{(1)}=P^{(1)}$. Thus $P^{(1)}$ is a one layer Weisfeiler-Lehman graph kernel. □

V. Experiments

In this section, we evaluate the performance of BANE on node classification and link prediction tasks. Node classification is popularly used to estimate the performance of network embedding methods. Moreover, the link prediction task is a popular testbed for evaluating model efficiency.

Table I
Dataset Description

Datasets	# Nodes	# Edges x$\|E\|$	# Attributes	# Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Wiki	2,405	17,981	4,973	19

A. Experimental Setup

Datasets. Three real-world attributed networks are used as testbed. They are popularly used in previous work [21]. Statistics of the datasets are summarized in Table I.

Baseline Methods. We compare our method with the state-of-the-art. DeepWalk and node2vec use plain network structure for embedding. TADW, HSCA and LANE use both network structure and attributes. The details are listed below,

- **DeepWalk [12]** captures contextual structure information based on random walks.
- **Node2vec** [4] performs a biased DeepWalk to explore diverse neighbors.
- **TADW [21]** learns node representations by combining attributes with structure in matrix tri-factorization.
- **HSCA [22]** adds the firstorder proximity to TADW.
- **LANE [7]** models the structural proximities in the attributed network and labels based on pairwise similarities.

Settings and Metrics. For fair comparisons, we set the embedding dimension $d=100$ for all baselines. All the parameters are set to be the default values.

For node classification, we randomly sample a portion of labeled nodes for training and use the rest for testing. The training ratios range from 10% to 90% with an increasing step of 20% for all the datasets. We use 10-fold cross validation and repeat the testing for 10 times. The performance of all the methods are evaluated in terms of Micro-F1 and Macro-F1.

For link prediction, we randomly sample 90% neighbors of each node for training and use the rest for testing. We also repeat the recommendation procedure 10 times and evaluate the performance of all the methods in terms of AUC [5], which represents the probability that a randomly selected unobserved link is more similar than a randomly selected non-existent one.

Table II
NODE CLASSIFICATION RESULTS (d=100)

		Micro-F1 (%)					Macro-F1(%)				
Datasets	Models	10%	30%	50%	70%	90%	10%	30%	50%	70%	90%
Cora	DeepWalk	63.71	73.50	78.83	80.29	81.20	61.02	71.65	77.63	79.08	79.83
	Node2vec	67.10	77.30	81.22	82.68	83.52	66.56	76.50	80.14	81.61	82.28
	TADW	81.50	84.97	85.78	86.23	86.93	79.71	83.35	84.26	84.44	85.35
	HSCA	75.21	81.25	85.10	85.97	86.38	73.42	80.10	84.01	84.41	84.82
	LANE	67.21	70.15	73.38	76.91	80.81	66.39	68.49	72.67	75.32	79.95
	BANE	**81.88**	**85.32**	**86.35**	**87.06**	**88.30**	**80.23**	**84.26**	**85.19**	**85.76**	**87.11**
Citeseer	DeepWalk	43.24	49.06	54.41	56.16	56.31	40.57	45.65	49.33	50.32	49.17
	Node2vec	48.56	55.77	62.55	63.66	63.69	46.78	53.92	58.09	59.42	60.47
	TADW	69.38	71.48	72.18	72.75	72.84	61.80	64.62	65.83	66.54	67.03
	HSCA	69.47	71.54	72.61	73.66	73.96	61.62	64.80	65.98	66.70	67.21
	LANE	53.81	60.72	61.65	63.58	67.77	50.33	57.05	58.14	60.63	63.60
	BANE	**70.24**	**72.55**	**73.78**	**74.55**	**75.08**	**62.37**	**65.73**	**67.63**	**68.44**	**69.35**
Wiki	DeepWalk	56.95	61.44	63.71	65.33	66.55	45.36	48.37	50.63	52.28	52.81
	Node2vec	57.83	62.25	63.70	65.31	66.36	45.88	49.90	50.78	52.22	52.04
	TADW	67.04	71.25	72.36	73.19	74.33	46.76	51.45	52.76	53.07	53.22
	HSCA	68.75	71.87	73.35	74.71	77.05	46.30	52.03	53.57	54.57	54.90
	LANE	62.95	69.04	70.45	72.01	73.24	46.38	50.73	52.34	54.62	55.12
	BANE	**71.41**	**77.07**	**78.91**	**79.76**	**80.49**	**46.81**	**54.83**	**56.95**	**58.43**	**58.04**

B. Node Classification Results

For all the three datasets, we reduce the dimension of node attributes to 200 by using the SVD decomposition on X. The preprocessing reduces the number of parameters in factorization. We use SVM [2] for node classification. The embedding dimension d is set to 100 and the regularization parameter α is set to 0.001.

Table II lists the results of node classification. We summarize the results as follows,

- First, BANE significantly outperforms DeepWalk and node2vec on all datasets with respect to both Micro-F1 and Macro-F1 under different training ratios from 10% to 90%. The results indicate that combining node links and attributes can substantially improve embedding accuracy.
- Second, BANE outperforms all the attributed network embedding algorithms on different datasets in terms of both Micro-F1 and Macro-F1 under different training ratios. The classification results are significantly higher than the other baseline methods by 3% on the Wiki dataset. The results indicate the effectiveness and robustness of BANE to handle both structure and attribute information.
- Third, BANE is the only binarized representation method. The results show that binary representation does not necessarily lead to accuracy loss. In fact, it may avoid the trap of over-fitting.
- Forth, BANE performs stably better than all the other benchmarks when the training ratio is low. For example, the Micro-F1 result on Wiki with 10% training reaches 0.714, which is much higher than the second highest 0.687 from HSCA. The accuracy results of most baseline methods drop rapidly when the training ratio decreases, because their node representations are noisy and inconsistent from training to testing. Instead, BANE learns jointly from node links and attributes by using high layer Weisfeiler-Lehman proximity matrix. Thus, the results of BANE contain less noise and are more robust.

Table III
LINK PREDICTION RESULTS ON THREE DATASETS

	Cora	Wiki	Citeseer
DeepWalk	83.10	80.46	80.56
Node2vec	81.59	78.91	80.24
TADW	89.77	89.86	93.80
HSCA	87.01	87.45	93.50
LANE	86.07	77.21	77.18
BANE	**93.50**	**90.90**	**95.59**

C. Link Prediction Results

Table III shows the results of link predictions on the three datasets. We randomly sample 90% neighbors of each node for training and the rest for testing. We measure the performance by AUC. The observations are listed below,

- First, we can observe that our method significantly outperforms baselines. The AUC score reaches very high value of 93.5% on Cora and 95.6% on Citeseer.
- Second, we can observe that converting real-valued numbers into binary representation can improve the link prediction accuracy. This is because the binary representation can alleviate the over-fitting problem and it is more intuitional to express the Yes/No option for recommendation. Furthermore, binary representation can replace the dot-product similarity computation with bit-wise Hamming distance. Thus, the speed can be significantly improved.

D. Binarized vs Real-valued Weisfeiler-Lehman Proximity Matrix Factorization

We also compare the original binary BANE model with its real-valued variant (BANE-r for short) by removing the binary constraint in Eq. (1). The results are reported in Table IV.

Table IV
NODE CLASSIFICATION RESULTS BETWEEN REAL-VALUED EMBEDDING AND BINARIZED EMBEDDING.

		Micro-F1					Macro-F1				
Datasets	Models	10%	30%	50%	70%	90%	10%	30%	50%	70%	90%
Cora	BANE-r	80.94	**86.70**	**87.56**	**87.87**	**89.00**	79.75	**85.64**	**86.46**	**86.61**	**87.92**
	BANE	**81.88**	85.32	86.35	87.06	88.30	80.23	84.26	85.19	85.76	87.11
Citeseer	BANE-r	67.91	**74.15**	**75.17**	**75.82**	**76.01**	61.77	**69.11**	**70.47**	**71.18**	**71.78**
	BANE	**70.24**	72.55	73.78	74.55	75.08	**62.37**	65.73	67.63	68.44	69.35
Wiki	BANE-r	63.82	71.04	74.76	75.65	77.44	**48.71**	**60.55**	**65.53**	**67.20**	**72.21**
	BANE	**71.41**	**77.07**	**78.91**	**79.76**	**80.49**	46.81	54.83	56.95	58.43	58.04

When comparing with BANE, we can observe that the real-valued embedding RANE receives slightly higher accuracy results than binary embedding on Cora and Citeseer when training ratios increase from 30% to 90%. Nevertheless, if training ratio is as low as 10%, the binary embedding of BANE beats real-valued embedding. For example, the classification Micro-F1 on the Citeseer dataset with 10% training ratio is 70.24 of BANE versus 67.91 of RANE. On the Wiki dataset, the Micro-F1 scores of BANE are higher than that of RANE at all training ratios, but the Macro-F1 scores are lower.

Remarks: The results show that the binary embedding BANE obtains competitive embedding results as real-valued embedding, especially when the training ratio is low. The reasons may be as follows.

First, binary constraints can be viewed as adding non-linear features to the linear matrix factorization, so linear classification on binary codes is equivalent to learning a nonlinear classifier on the original data. Second, the limited two values of binary codes can alleviate the possible over-fitting problem and obtain encouraging results even when the training ratio is small.

VI. CONCLUSIONS

In this paper we study a new problem of *Binarized Attributed Network Embedding* (*BANE* for short). We define a new *Weisfier-Lehman proximity matrix* to jointly encode data dependence between node links and attributes. Based on the new proximity matrix, we formulate a new binarized Weisfier-Lehman matrix factorization model to obtain binary node representation. Theoretical studies show the close connections of the new proximity matrix with Weisfier-Lehman graph kernels. Empirical results also validate the promising results compared with popular network embedding models.

In the future, we will consider to use the automated machine learning methods (AutoML) to search the best parameters for the BANE model. We wish the Weisfier-Lehman proximity matrix can precisely capture data dependence between node links and attributes for any given large networks with minimal human efforts.

ACKNOWLEDGMENT

This work is supported by an Australian Government Research Training Program Scholarship and Australian Research Council(ARC) Discovery Grant DP180100966.

REFERENCES

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

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 9(Aug):1871–1874, 2008.

[3] Li Gao, Hong Yang, Chuan Zhou, Jia Wu, Shirui Pan, and Yue Hu. Active discriminative network representation learning. In *IJCAI*, pages 2142–2148, 7 2018.

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[5] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[6] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *SDM*, pages 633–641. SIAM, 2017.

[7] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *WSDM*, pages 731–739. ACM, 2017.

[8] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[9] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[10] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, pages 2609–2615, 7 2018.

[11] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *IJCAI*, pages 1895–1901, 2016.

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

[13] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[14] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *TPAMI*, 2018.

[15] Xiaobo Shen, Shirui Pan, Weiwei Liu, Yew-Soon Ong, and Quan-Sen Sun. Discrete network embedding. In *IJCAI*, pages 3549–3555, 7 2018.

[16] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 12(Sep):2539–2561, 2011.

[17] 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.

[18] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *TPAMI*, 2017.

[19] Wei Wu, Bin Li, Ling Chen, Xingquan Zhu, and Chengqi Zhang. k-ary tree hashing for fast graph classification. *TKDE*, 2017.

[20] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *TPAMI*, 29(1):40–51, 2007.

[21] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015.

[22] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Homophily, structure, and content augmented network representation learning. In *ICDM*, pages 609–618. IEEE, 2016.