Low-Bit Quantization for Attributed Network Representation Learning

Hong Yang[1], **Shirui Pan**[2], **Ling Chen**[1], **Chuan Zhou**[3] and **Peng Zhang**[4]
[1]Centre for Artificial Intelligence, University of Technology Sydney, Australia
[2]Faculty of Information Technology, Monash University, Australia
[3]Institute of Information Engineering, Chinese Academy of Sciences, China
[4]Ant Financial Services Group, Hangzhou, China
hong.yang@student.uts.edu.au, shirui.pan@monash.edu, ling.chen@uts.edu.au,
zhouchuan@iie.ac.cn, hanyi.zp@antfin.com

Abstract

Attributed network embedding plays an important role in transferring network data into compact vectors for effective network analysis. Existing attributed network embedding models are designed either in continuous Euclidean spaces which introduce data redundancy or in binary coding spaces which incur significant loss of representation accuracy. To this end, we present a new Low-Bit Quantization for Attributed Network Representation Learning model (LQANR for short) that can learn compact node representations with low bit-width values while preserving high representation accuracy. Specifically, we formulate a new representation learning function based on matrix factorization that can jointly learn the low-bit node representations and the layer aggregation weights under the low-bit quantization constraint. Because the new learning function falls into the category of mixed integer optimization, we propose an efficient mixed-integer based alternating direction method of multipliers (ADMM) algorithm as the solution. Experiments on real-world node classification and link prediction tasks validate the promising results of the proposed LQANR model.

1 Introduction

Attributed networks are popularly used to describe a large body of networks where both node links and attributes are obtainable for analysis [Le and Lauw, 2014]. To discover latent patterns from attributed networks, *attributed network representation learning* or *attributed network embedding* models are proposed to jointly learn node representations from both node links and attributes. For example, the work [Yang *et al.*, 2015] uses textual attributes to supervise random walks on networks and derives the Text-associated DeepWalk (TADW) model. AANE [Huang *et al.*, 2017a] uses node links to supervise the factorization of attributed proximity matrices. LANE [Huang *et al.*, 2017b] mutually uses node links and attributes as labels to supervise the learning from each other. *However*, these attributed network embedding models are developed in continuous Euclidean spaces, where the learned representation vectors usually contain redundant information that degenerates computation efficiency and increases storage cost, especially when networks are large.

To reduce the size of network representation and speed up the inference on a big network, considerable efforts have been put on generating succinct network representation with low-bit representations. A pioneer work [Shen *et al.*, 2018b] is to learn binary graph embedding where a short binary code is learnt for each node through factorizing the graph similarity matrix by imposing the Hamming similarity constraint on each pair of nodes. Based on this work, a binary attributed network embedding method [Yang *et al.*, 2018] is proposed that extends the binary network embedding to binary attributed network embedding, where a new Weisfeiler-Lehman proximity matrix is used to capture data dependence between node links and attributes and a new Weisfiler-Lehman matrix factorization learning function is proposed by adding an extra binary node representation constraint.

Despite the success of reducing network representation size, the strict binary representation constraint imposed on the learning function often suffer from uncontrollable accuracy loss on test sets. For example, the binary attributed network embedding [Yang *et al.*, 2018] method, by adding the binary node representation constraint on the Weisfiler-Lehman matrix factorization, the node representation on the test set of Citeseer drops at least 2% compared with its real-valued variants without the binary constraint, especially when there are plenty of training examples. Based on the observation, it is natural to raise the question: *how to design a compact yet flexible attributed network representation model that can balance representation accuracy and representation size ?*

On the other hand, from the perspective of low-bit quantization for convolutional neural networks, there has been a series of methods [Wang *et al.*, 2017] proposed to reduce the size of network parameters while preserving high prediction accuracy. Low-bit compression of deep neural networks such as training binary neural networks with weights constrained to +1 and -1 [Courbariaux *et al.*, 2016], ternary networks [Zhu *et al.*, 2016] and extremely low-bit neural networks [Leng *et al.*, 2017] have been popularly studied recently. Compared to full-precision models, these compressed models are sparse and much smaller, which can potentially be accelerated with customized circuits and deployed to mobile devices. In particular, the early work [Denil *et al.*, 2013] pointed out that network weights have a significant redun-

dancy, and proposed to reduce the number of parameters by exploiting the linear structure of network, which motivated a line of low-rank matrix and tensor factorization based compression algorithms. The achievement in low-rank matrix and tensor factorization based compression motivates to learn low-bit quantization for attributed network embedding based on matrix factorization.

In this paper, we study the problem of flexible and compact attributed network representation and present a new *Low-Bit Quantization for Attributed Network Representation Learning* model (*LQANR* for short). Specifically, we use a *k-hop proximity matrix* to capture data dependence by propagating node attribute information from k-layer neighboring nodes to a given target node. Based on the k-hop proximity matrix, we formulate a new matrix factorization function to learn the low-bit node representation, where the weights of different layers of neighboring nodes are learnable instead of manually setting. The learning function falls into the category of mixed integer optimization and we propose an efficient *mixed-integer based alternating direction method of multipliers* (ADMM) algorithm for solution. Experimental results on real-world datasets validate the performance of the proposed LQANR method. The framework of LQANR is illustrated in Figure 1. The contribution can be summarized as follows,

- We first study the *low-bit attributed network embedding* problem and present a new LQANR model as the solution. LQANR can learn compact representation with stably high accuracy.
- We propose a new *mixed-integer based alternating direction method of multipliers (ADMM)* algorithm to efficiently learn the low-bit node representation and the layer-wise aggregation weights.
- We conduct experiments to validate the performance of the proposed LQANR model.

2 Related Work

Attributed network embedding. Current network embedding methods can be categorized into *plain network embedding* [Yan *et al.*, 2007] and *attributed network embedding* [Chang *et al.*, 2015]. 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 [Yang *et al.*, 2015] that uses textual attributes to supervise random walks on networks. Similar works include AANE [Huang *et al.*, 2017a], LANE [Huang *et al.*, 2017b], and DANE model [Li *et al.*, 2017].

Low-bit quantization for compression. Quantization methods including hashing are used to encode real-valued data to low-bit discrete data while preserving similarity structure in the original space [Wang *et al.*, 2017]. The low-bit codes can facilitate to represent and search of massive data because it only needs about one hundred bits to represent one data item, and computation in Hamming space is efficient by

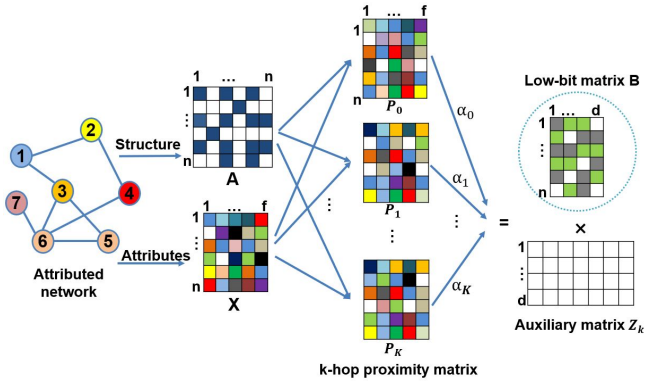


Figure 1. The conceptual framework of *Low-Bit Quantization for Attributed Network Representation Learning model (LQANR)*. Given an attributed network $G = \{V, E, X\}$, LQANR first derives the k-hop proximity matrix $P_k = (\tilde{D}^{-1}\tilde{A})^k X$ by aggregating information from both the structure matrix A and the attribute matrix X. By learning the layer-wise aggregation weights $\{\alpha_0, \alpha_1, \cdots, \alpha_K\}$ for matrices $\{P_0, P_1, \cdots, P_k\}$, LQANR jointly factorizes the k matrices into a low-bit node representation matrix B and a set of auxiliary matrices Z_k, as shown in Eq.(1).

using the bit operations. Most Hashing methods use one single bit $-1/+1$ to quantize each projected dimension, such as the spectral hashing [Liu *et al.*, 2014], supervised discrete hashing [Shen *et al.*, 2015] and deep learning based hashing methods [Shen *et al.*, 2018a]. There are also works quantizing real-valued data to multiple-bit codes using Hashing method, such as double bit quantization[Kong and Li, 2012], q-bit Manhattan quantization [Kong *et al.*, 2012] and Variable Bit Quantization [Moran *et al.*, 2013]. Recently discrete network embedding approach is proposed to learn binary codes for plain network [Shen *et al.*, 2018b], and randomized hashing method [Wu *et al.*, 2018] and binarized network embedding [Yang *et al.*, 2018] are proposed for compressing embedding for attributed networks. In this paper, we learn compact low-bit codes for attributed network embedding.

Graph signal processing. Graph signal processing [Sandryhaila and Moura, 2013] extends the signals and filters in traditional time and image based signal processing to irregular graph domains. In the spatial domain, smooth graph signals denote that neighboring nodes tend to have similar values. In the spectral domain, the smoothness of graph signals are typically called bandlimitedness. To create graph filters, functions of graph-shift operators or adjacency matrices are based on either rules or learning functions. The polynomial form of adjacency and the Laplacian matrix are used as graph filters, due to the capability of capturing local structure of graphs and ease of implementation [Ortega *et al.*, 2018].

3 Problem Statement

An attributed network can be 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 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 an existing attributed network G, we need to embed each node $v_i \in V$ into a d-dimensional low-bit vector $b_i \in \{-2^N, ..., -2^1, -1, 0, 1, ..., -2^N\}^d$, where N is an integer which determines the bit-width. b_i is the i^{th} row of matrix $B \in R^{n\times d}$. matrix B should preserve the structure information A and the attribute information X in G.

How to design a proximity matrix that can jointly describe structure A and attribute X is the most important question. This can be considered as a problem of designing a graph signal processing filter [Sandryhaila and Moura, 2013]. We consider attribute X as the graph input signal on each node and use graph adjacency matrix to create a k-hop graph filter. Then, a proximity matrix P_k is generated as the output signal. Different k-hop matrix P_k captures different layer-wise information from the neighboring nodes. We can learn the weights of P_k based on the specific applications.

Concretely, given a network G with adjacency matrix A and attribute matrix X, let $\tilde{D}$ be a degree matrix of $\tilde{A}$, the k-hop proximity matrix P_k is defined as $P_k = (\tilde{D}^{-1}\tilde{A})^k X$, where k is the number of aggregation layers.

4 The LQANR Method

In this section, we first formulate the learning function of *Low-Bit Quantization for Attributed Network Representation Learning model* (LQANR) based on matrix factorization under the low-bit quantization constraint. Then, we propose an efficient *mixed-integer based alternating direction method of multipliers algorithm* as the solution.

4.1 Formulation

Given a target node, the k-hop proximity matrix P_k aggregates attribute and structure information from its k-layer neighboring nodes. Then, P_k can be factorized to generate a low-dimensional representation of the target node. Moreover, considering that neighboring nodes located at different layers contribute differently to the target node, the learning function needs to capture the aggregation importance weights in a layer-wise manner.

The representation learning function can be formulated by simultaneously learning the low-bit node representation and the layer-wise aggregation weights. Assume that α_k is the importance weight of matrix P_k, matrix $B \in R^{n\times d}$ is the low-bit node representation, matrix $Z_k \in R^{d\times f}$ is an auxiliary matrix with respect to layer k. Then, the learning problem can be formulated as follows,

$$\begin{aligned} \min_{B,Z_0,...,Z_K,\alpha} \ & \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \beta \sum_{k=0}^{K} \|Z_k\|_F^2, \qquad (1) \\ s.t.: \ & B \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n\times d}, \\ & \sum_{i=1}^{K} \alpha_k = 1, \alpha_k \geq 0, \ \ Z_k \in R^{d\times f}, \end{aligned}$$

where β is a regularization parameter with respect to auxiliary matrices Z_k, K is the total number of layers we consider in the model. A large layer number K may cause over-smoothing when learning node attributes, while a small K cannot fully take advantage of network information. Due to the integer constraint over the representation matrix B, Eq.(1) is hard to solve which requires efficient algorithms.

4.2 Optimization

In this part, we present an efficient algorithm to iteratively optimize variables Z_k, B and α in Eq.(1). The algorithm updates one parameter at a time and converges very fast.

Z_k-step

Given B and α fixed, solve the sub-problem with respect to Z_k in Eq.(1). The loss function becomes,

$$\begin{aligned} & \min_{Z_k} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \beta \sum_{k=0}^{K} \|Z_k\|_F^2 \qquad (2) \\ =& \sum_{k=0}^{K} \alpha_k tr(Z_k^T B^T B Z_k) - \sum_{k=0}^{K} \alpha_k tr(P_k^T B Z_k) + \beta \sum_{k=0}^{K} tr(Z_k^T Z_k) \end{aligned}$$

where $tr(.)$ is a trace norm. By calculating the derivative of Eq.(2), we derive a closed form solution as follows,

$$Z_k = (\alpha_k B^T B + \alpha I)^{-1} \alpha_k B^T P_k. \qquad (3)$$

B-step

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

$$\begin{aligned} & \min_{B} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2, \qquad (4) \\ & s.t.: \ B \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n\times d}. \end{aligned}$$

Due to the discrete constraint, the optimization problem above is NP-hard.

Here, We introduce an auxiliary variable Q to decouple the parameters in the objective and the discrete constraint. The idea is largely motivated by the successful application of ADMM in mixed integer programs [Leng *et al.*, 2017]. Then, the objective function in Eq.(4) can be written as,

$$\begin{aligned} & \min_{B,Q} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + Ic(Q), \qquad (5) \\ & s.t. : B = Q, \ Q \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n\times d}, \end{aligned}$$

where I_c is defined as an indicator function. $I_C(Q) = 0$ if $Q \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}$; otherwisse, $I_C(Q) = +\infty$. The augmented Lagrange of Eq.(5), for parameter $\rho > 0$, can be formulated as,

$$\begin{aligned} L_\rho(B, Q, \lambda) = & \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + Ic(Q) \qquad (6) \\ & + \frac{\rho}{2} \|B - Q + \lambda\|_F^2 - \frac{\rho}{2} \|\lambda\|_F^2. \end{aligned}$$

Eq.(6) can be solved by repeating the following iterations,

$$B^{t+1} := \arg\min_B L_\rho(B, Q^t, \lambda^t), \tag{7}$$

$$Q^{t+1} := \arg\min_Q L_\rho(B^{t+1}, Q, \lambda^t), \tag{8}$$

$$\lambda^{t+1} := \lambda^t + B^{t+1} - Q^{t+1}. \tag{9}$$

Benefit from the decoupling of ADMM, Eq.(7) is an unconstrained objective function. We can easily calculate the gradient with respect to matrix B,

$$\frac{\partial L_\rho(B, Q^t, \lambda^t)}{\partial B} = \sum_{k=0}^{K} 2\alpha_k B Z_k Z_k^T - \sum_{k=0}^{K} 2\alpha_k P_k Z_k^T + \rho(B - Q^t + \lambda^t). \tag{10}$$

The closed form solution is given in Eq.(11), where I is an identity matrix.

$$B^{t+1} = (\sum_{k=0}^{K} \alpha_k P_k Z_k^T + \rho Q^t - \rho\lambda^t)^{-1}(\sum_{k=0}^{K} \alpha_k Z_k Z_k^T + \rho I). \tag{11}$$

In order to solve Q, Eq.(6) can be rewritten as,

$$\min_Q \ \|Q - B^{t+1} - \lambda^t\|_F^2, \tag{12}$$
$$s.t.: \ Q \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n\times d}.$$

The optimal solution of Q is

$$Q = \prod_{0,\pm1,\pm2,..,\pm N} (B^{t+1} + \lambda^t), \tag{13}$$

where $\prod_{0,\pm1,\pm2,..,\pm N}$ denotes the projection of $(B^{t+1} + \lambda^t)$ with respect to the discrete set. After either the predefined iterations or the convergence of ADMM, Q is assigned to B and the algorithm continues to update α and Z_k.

α-Step

Given Z_k and B fixed, rewrite the objective function in Eq. (1) with respect to α as follows,

$$\min_\alpha \ \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2, \tag{14}$$
$$s.t.: \sum_{k=0}^{K} \alpha_k = 1, \alpha_k \geq 0.$$

The optimal solution to α in Eq.(14) is $\alpha_k = 1$ corresponding to the minimum $\|P_k - BZ_k\|_F^2$ and $\alpha_k = 0$ otherwise. This solution means that only one order of P_k is finally selected. However, the solution of a single order does not meet our objective on exploring the complementary property of multiple orders to get a better embedding.

Alternatively, we use a trick based on the work [Xia *et al.*, 2010] to avoid the single order solution. We set $\alpha_k^r \leftarrow \alpha_k$ with $r > 1$ and obtain the Lagrange of Eq.(15) as below,

$$L(\alpha, \eta) = \sum_{k=0}^{K} \alpha_k^r \|P_k - BZ_k\|_F^2 - \eta(\sum_{k=0}^{K} \alpha_k - 1). \tag{15}$$

By setting the derivative of $L(\alpha, \eta)$ with respect to α_k and η to zero, we obtain

$$\begin{cases} \frac{\partial L(\alpha,\eta)}{\partial \alpha_k} & = \gamma\alpha_k^{r-1}(P_k - BC_i) - \eta = 0, \\ \frac{\partial L(\alpha,\eta)}{\partial \lambda} & = \sum_{k=0}^{K} \alpha_k - 1 = 0. \end{cases} \tag{16}$$

Algorithm 1 The LQANR model

Input: Structure A, attribute X, dimension d, # of iterations t_1 and t_2, parameters K, β, ρ, r, N
Output: Low-bit representation matrix B

```
Initialize Z_k, B, λ randomly
Repeat until converge or reach t_1
  Z_k-Step: Calculate Z_k using Eq.(3)
  B-Step: Repeat until converge or reach t_2 (ADMM)
    Update B using Eq.(10)
    Update Q using Eq.(12)
    Update λ using Eq.(9)
  // After ADMM, let B=Q
  α-Step: Calculate α using Eq.(16)
return matrix B
```

Then, α_k can be solved as follows,

$$\alpha_k = \frac{(1/\|P_k - BZ_k\|_F^2)^{1/(r-1)}}{(\sum_{k=0}^{K} 1/\|P_k - BZ_k\|_F^2)^{1/(r-1)}}. \tag{17}$$

Because $\|P_k - BZ_k\|_F^2 \geq 0$, we have $\alpha_k \geq 0$ naturally.

The details of the algorithm is given in Algorithm 1. Empirical studies show that the algorithm takes a few iterations to converge. For example, in our experiments B is iteratively computed and converges around $2 - 10$ iterations.

5 Experiments

In this section, we evaluate the performance of LQANR on node classification and link prediction tasks.

5.1 Experimental Setup

Datasets. Three real-world attributed networks are used as testbed. They are popularly used in many network embedding works such as [Yang *et al.*, 2015; Huang *et al.*, 2017b]. We summarize the statistics of datasets in Table 1.

Baseline methods. We compare our method with several popular network embedding methods. DeepWalk [Perozzi *et al.*, 2014] and node2vec [Grover and Leskovec, 2016] use plain network structure for embedding. TADW [Yang *et al.*, 2015], HSCA [Zhang *et al.*, 2016] and LANE [Huang *et al.*, 2017b] use both network structure and attributes. We also compare with the most recent NetHash [Wu *et al.*, 2018] and BANE [Yang *et al.*, 2018] which employs the hashing technique to learn binary network embedding.

Settings and metrics. We set the embedding dimension $d = 100$ for all baselines. All the parameters are set to be the default values. For node classification experiment, we use SVM [Fan *et al.*, 2008] as the classifier. The training ratios range from 10% to 90% for all the datasets. We use 10-fold cross validation. The performance are evaluated in terms of

Datasets	# Nodes	# Edges	# Attributes	# Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
BlogCatalog	5,196	171,743	8,189	6

Table 1. Dataset description

		Micro-F1 (%)					Macro-F1(%)				
Datasets	**Modes**	**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
	NetHash	81.24	85.06	86.19	86.56	86.98	79.64	83.43	84.58	84.87	85.82
	BANE	81.88	85.32	86.35	87.06	**88.30**	80.23	84.26	85.19	85.76	**87.11**
	LQANR	**83.00**	**85.91**	**86.74**	**87.27**	88.21	**81.79**	**84.79**	**85.57**	**85.95**	86.95
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
	NetHash	69.63	71.89	72.35	73.16	72.85	61.97	64.47	66.11	67.36	67.90
	BANE	70.24	72.55	73.78	74.55	75.08	62.37	65.73	67.63	68.44	69.35
	LQANR	**70.41**	**72.73**	**73.80**	**74.67**	**75.20**	**62.94**	**66.11**	**67.80**	**68.72**	**69.62**
BlogCatalog	DeepWalk	69.58	78.24	79.37	80.78	81.12	68.65	76.85	78.46	80.01	80.54
	Node2vec	72.43	79.05	82.36	83.40	84.95	71.54	77.27	80.81	80.95	82.03
	TADW	82.50	86.56	87.72	89.20	89.78	82.29	86.35	87.60	89.04	89.53
	HSCA	82.10	85.89	87.64	89.01	89.47	81.56	85.36	87.02	88.43	89.11
	LANE	85.23	88.56	89.64	89.89	90.08	85.85	88.27	89.03	89.59	89.95
	NetHash	81.59	86.43	87.66	88.90	89.25	82.03	86.24	87.14	88.39	89.01
	BANE	86.21	89.04	89.55	89.85	89.88	85.71	88.74	89.30	89.55	89.59
	LQANR	**86.24**	**89.29**	**89.95**	**90.44**	**90.75**	**85.91**	**89.10**	**89.79**	**90.31**	**90.55**

Table 2. Node classification results (d=100)

Micro-F1 and Macro-F1. For link prediction task, we randomly sample 90% neighbors of each node for training and use the rest for testing. We repeat 10 times and evaluate in terms of AUC [Hanley and McNeil, 1982].

5.2 Node Classification Results

The embedding dimension d is set to 100, the bit-width used for each node's representation is $\{-1, 0, +1\}$, and the regularization parameter β is set to 0.001. We summarize the results in Table 2 as follows:

- First, LQANR outperforms not only DeepWalk and Node2vec which use pure network structure information, but also the methods which use both network structure and attributes information on Cora, Citeseer and Blogcatalog in terms of Micro-F1 and Macro-F1 under different training ratios. The results validate the effectiveness and robustness of LQANR. The reason is that we use a k-hop proximity matrix to capture data dependence by propagating node attribute information from k-layer neighboring nodes to a given target node.
- Second, LQANR outperforms real-valued or called continues representation methods. The results show that low-bit representation does not necessarily lead to embedding accuracy loss compared to continues embedding. By forcing discrete constraint to the learning function, we add the non-linear factors to the matrix factorization objective function. This operation may help to avoid over-fitting problem.
- Third, LQANR outperforms NetHash and BANE which can only generate binary representation. Our method can obtain any low-bit embedding not constraint to binary, which is more flexible and accurate to capture attributed network information.

5.3 Link Prediction Results

Table 3 shows the results of link prediction. The bit-width is $\{-1, +1\}$. We summarize the observations as follows:

- First, LQANR significantly outperforms baselines on the Cora and Citeseer datasets. The AUC score reaches as high as 93.85% on Cora and 96.51% on Citeseer.
- Second, LQANR outperforms baselines using real-valued representations. Converting real-valued numbers into low-bit discrete representation can improve the link prediction accuracy because the low-bit representation can alleviate the over-fitting problem and it is preferable to deliver Yes or No for recommendation.

5.4 Speedup of LQANR

In addition to the superior performance of LQANR with respect to AUC, low-bit representation also accelerates link prediction speed by replacing the dot-product similarity computation with bit-wise Hamming distance. The speedup of 100 and 200 dimension nearest search via hamming distance compared to dot-product is shown in Figure 2. The results show that large-sized networks gain significant speedup.

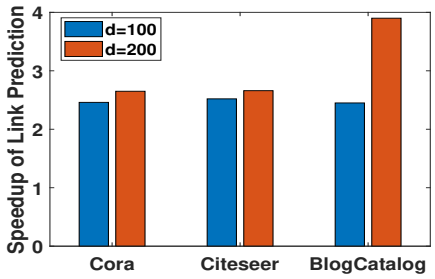


Figure 2. Speedup of link prediction

Models	Cora	Citeseer	BlogCatalog
Node2vec	81.59	80.24	60.31
TADW	89.77	93.80	60.4
HSCA	87.01	93.50	60.35
LANE	86.07	77.18	58.97
NetHash	90.35	95.03	61.05
BANE	93.50	95.59	61.44
LQANR	**93.85**	**96.51**	**61.66**

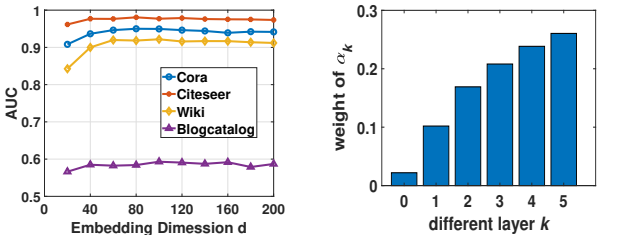
Table 3. Link prediction results on the three datasets

5.5 Parameter Study

In this part, we test four parameters, bit-width decided by N, embedding dimension d, proximity matrix maximum order K and weights of P_k impacted by r .

Bit-width decided by N. We study different kinds of bit-width for discrete node embedding. We test binary quantization, ternary quantization, one-bit shift quantization and two-bits shift quantization. The node classification result on the Cora dataset with respect to different bit-width values is shown in Table 4. We can observe that the classification accuracy increases with bit-width. For example, the Micro-F1 score increases from 80.16 when B is represented by {-1,1} to 83.51 when B is represented by {-4,-2,...,2,4}. In our experiments, we set B as {-1,0,1} if not specially mentioned.

Node embedding dimension d. We test the embedding dimension d from 20 to 200 with a stepsize of 20. The link prediction results are shown in Figure 3(a). We can observe that the performance of network embedding improves with d increasing from 20 to 100, and then remains stable when code length continuously increases. On the BlogCatalog dataset, the link prediction results are the lowest. This is because BlogCatolog contains more complicated structure and attribute information than the other datasets.



(a) Link Prediction *w.r.t.* d (b) Weights of α_k on Citeseer

Figure 3. Parameter studies *w.r.t.* embedding d and α_k

Bit-width	10%	30%	50%	70%	90%
(1,-1)	80.16	84.13	84.9	85.38	86.33
(-1,0,1)	83.00	85.91	86.74	87.27	87.70
(-2,-1,0,1,2)	83.40	86.46	87.22	87.68	88.33
(-4, ..., 4)	83.51	86.53	87.34	87.70	88.85

Table 4. Node classification (Micro-F1) *w.r.t.* bit-width

Order K	10%	30%	50%	70%	90%
K=2	82.29	85.46	85.97	86.12	86.37
K=3	82.69	85.58	86.44	86.90	87.56
K=4	82.67	85.62	86.62	**87.36**	87.44
K=5	**83.00**	**85.91**	**86.74**	87.27	87.70
K=6	82.63	85.70	86.43	87.03	**88.11**

Table 5. Node classification (Micro-F1) *w.r.t.* K on Cora

Proximity matrix order K. We test node classification with different K. The results on Cora with K arranging from 2 to 6 are shown in Table 5. It shows that $K = 5$ is the best choice for Cora in many cases. The reason is that when K is too large, it can cause over-smoothing for node attributes. However, small K cannot fully propagate node attribute information in networks.

Weights of P_k impacted by r. Different k-hop P_k matrices capture different steps of neighboring node attributes. The layer-wise weights α_k are impacted by r. We test on different datasets and find the best node classification results with the best parameter r. r is usually between 1 and 10 for the tested datasets. We plot the distribution of α_k on Citeseer with $K = 5$ and $r = 1.6$. From Figure 3(b), we can observe that the higher order P_k contributes heavier weights, which means combining more layers leads to better results.

6 Conclusions

In this paper we study a new problem of *Low-Bit Quantization for Attributed Network Representation Learning* (*LQANR*). We use a *k-hop proximity matrix* to jointly encode data dependence between node links and attributes. Based on the k-hop proximity matrix, we formulate a new learning function that simultaneously learns the low-bit node representation and the layer-wise aggregation weights. Experimental results validate the performance of the proposed model. In the future, we will combine graph signal processing methods with low-bit neural network compression methods so as to learn more concise network representations.

Acknowledgements

This project was supported by ARC Discovery Grant DP180100966, NSFC (No. 61872360) of China and an Australian Government Research Training Program Scholarship.

References

[Chang *et al.*, 2015] 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.

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[Denil *et al.*, 2013] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[Fan *et al.*, 2008] 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.

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

[Hanley and McNeil, 1982] 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.

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

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

[Kong and Li, 2012] Weihao Kong and Wu Jun Li. Double-bit quantization for hashing. In *AAAI*, pages 634–640, 2012.

[Kong *et al.*, 2012] Weihao Kong, Wu-Jun Li, and Minyi Guo. Manhattan hashing for large-scale image retrieval. In *SIGIR*, pages 45–54. ACM, 2012.

[Le and Lauw, 2014] Tuan MV Le and Hady W Lauw. Probabilistic latent document network embedding. In *ICDM*, pages 270–279. IEEE, 2014.

[Leng *et al.*, 2017] Cong Leng, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *AAAI*, 2017.

[Li *et al.*, 2017] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *CIKM*, pages 387–396. ACM, 2017.

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

[Moran *et al.*, 2013] Sean Moran, Victor Lavrenko, and Miles Osborne. Variable bit quantisation for lsh. In *ACL*, pages 753–758, 2013.

[Ortega *et al.*, 2018] Antonio Ortega, Pascal Frossard, Jelena Kovacevic, Jos M. F. Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

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

[Sandryhaila and Moura, 2013] Aliaksei Sandryhaila and Jos M. F. Moura. Discrete signal processing on graphs. *TSP*, 61(7):1644–1656, 2013.

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

[Shen *et al.*, 2018a] 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.

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

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

[Wu *et al.*, 2018] Wei Wu, Bin Li, Ling Chen, and Chengqi Zhang. Efficient Attributed Network Embedding via Recursive Randomized Hashing. In *IJCAI-18*, pages 2861–2867, 2018.

[Xia *et al.*, 2010] Tian Xia, Dacheng Tao, Tao Mei, and Yongdong Zhang. Multiview spectral embedding. *TSMC-B*, 40(6):1438–1446, 2010.

[Yan *et al.*, 2007] 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.

[Yang *et al.*, 2015] 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.

[Yang *et al.*, 2018] Hong Yang, Shirui Pan, Peng Zhang, Ling Chen, Defu Lian, and Chengqi Zhang. Binarized attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1476–1481. IEEE, 2018.

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

[Zhu *et al.*, 2016] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.