Low-Bit Quantization for Attributed Network Representation Learning

Hong Yang\textsuperscript{1}, Shirui Pan\textsuperscript{2}, Ling Chen\textsuperscript{1}, Chuan Zhou\textsuperscript{3} and Peng Zhang\textsuperscript{4}

\textsuperscript{1}Centre for Artificial Intelligence, University of Technology Sydney, Australia
\textsuperscript{2}Faculty of Information Technology, Monash University, Australia
\textsuperscript{3}Institute of Information Engineering, Chinese Academy of Sciences, China
\textsuperscript{4}Ant Financial Services Group, Hangzhou, China

hong.yang@student.uts.edu.au, shirui.pan@monash.edu, ling.chen@uts.edu.au,
zhou.chuan@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 \textit{et al.}, 2015] uses textual attributes to supervise random walks on networks and derives the Text-associated DeepWalk (TADW) model. AANE [Huang \textit{et al.}, 2017a] uses node links to supervise the factorization of attributed proximity matrices. LANE [Huang \textit{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 \textit{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 \textit{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 Weisfeiler-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 \textit{et al.}, 2018] method, by adding the binary node representation constraint on the Weisfeiler-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: \textit{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 \textit{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 \textit{et al.}, 2016], ternary networks [Zhu \textit{et al.}, 2016] and extremely low-bit neural networks [Leng \textit{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 \textit{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) model. 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 k-hop proximity matrix, 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 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 [Sanderhaila 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 $X = \{x_i\}_{i=1}^n \in \mathbb{R}^{n \times f}$ 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 \( B \in \mathbb{R}^{n \times d} \). 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 \( P_k = (\tilde{D}^{-1/2}A)^kX \), 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 \( \alpha_k \) is the importance weight of matrix \( P_k \), matrix \( B \in \mathbb{R}^{n \times d} \) is the low-bit node representation, matrix \( Z_k \in \mathbb{R}^{d \times f} \) is an auxiliary matrix with respect to layer \( k \). Then, the learning problem can be formulated as follows,

\[
\min_{B, Z_0, ..., Z_K, \alpha} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \beta \sum_{k=0}^{K} \|Z_k\|_F^2, \quad (1)
\]

\[
\text{s.t. : } B \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n \times d},
\]

\[
\sum_{i=1}^{K} \alpha_k = 1, \alpha_k \geq 0, \quad Z_k \in \mathbb{R}^{d \times f},
\]

where \( \beta \) 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 \( \alpha \) in Eq.(1). The algorithm updates one parameter at a time and converges very fast.

\( Z_k \)-step

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

\[
\min_{Z_k} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \beta \sum_{k=0}^{K} \|Z_k\|_F^2
\]

\[
= \sum_{k=0}^{K} \alpha_k \text{tr}(Z_k^T B^T B Z_k) - \sum_{k=0}^{K} \alpha_k \text{tr}(P_k^T B Z_k) + \beta \sum_{k=0}^{K} \text{tr}(Z_k^T Z_k)
\]

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

\[
Z_k = (\alpha_k B^T B + \alpha I)^{-1} \alpha_k B^T P_k.
\]

\( B \)-step

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

\[
\min_{B} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2, \quad (4)
\]

\[
\text{s.t.: } B \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n \times d}.
\]

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,

\[
\min_{B, Q} \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \text{Ic}(Q), \quad (5)
\]

\[
\text{s.t.: } B = Q, \quad Q \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n \times d},
\]

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

\[
L_{\rho}(B, Q, \lambda) = \sum_{k=0}^{K} \alpha_k \|P_k - BZ_k\|_F^2 + \text{Ic}(Q)
\]

\[
+ \frac{\rho}{2} \|B - Q + \lambda\|_F^2 - \frac{\rho}{2} \|\lambda\|_F^2.
\]
Eq. (6) can be solved by repeating the following iterations,

\[ B^{t+1} := \arg\min_{B} L(B, Q^t, \lambda^t), \quad (7) \]

\[ Q^{t+1} := \arg\min_{Q} L(B^{t+1}, Q, \lambda^t), \quad (8) \]

\[ \lambda^{t+1} := \lambda^t + B^{t+1} - Q^{t+1}. \quad (9) \]

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

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

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

\[
B^{t+1} = \left( \sum_{k=0}^{K} \alpha_k P_k Z_k^T + \rho I \right) - \frac{1}{\lambda^t} \left( \sum_{k=0}^{K} \alpha_k Z_k Z_k^T + \rho I \right).
\]

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

\[
\min_{Q} \| Q - B^{t+1} - \lambda^t \|_F^2, \quad (12)
\]

\[
s.t.: Q \in \{-2^N, ..., -2^1, -2^0, 0, 2^0, 2^1, ..., 2^N\}^{n \times d}. \]

The optimal solution of \( Q \) is

\[
Q = \prod_{0, \pm 1, \pm 2, ..., \pm N} (B^{t+1} + \lambda^t), \quad (13)
\]

where \( \prod_{0, \pm 1, \pm 2, ..., \pm N} \) 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 \( \alpha \) and \( Z_k \).

**\( \alpha \)-Step**

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

\[
\min_{\alpha} \sum_{k=0}^{K} \alpha_k \| P_k - B Z_k \|_F^2, \quad (14)
\]

\[
s.t.: \sum_{k=0}^{K} \alpha_k = 1, \alpha_k \geq 0. \]

The optimal solution to \( \alpha \) in Eq. (14) is \( \alpha_k = 1 \) corresponding to the minimum \( \| P_k - B Z_k \|_F^2 \) 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 \( \alpha_k \leftarrow \alpha_k \) with \( r > 1 \) and obtain the Lagrange of Eq. (15) as below,

\[
L(\alpha, \eta) = \sum_{k=0}^{K} \alpha_k \| P_k - B Z_k \|_F^2 - \eta \left( \sum_{k=0}^{K} \alpha_k - 1 \right). \quad (15)
\]

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

\[
\begin{align*}
\frac{\partial L(\alpha, \eta)}{\partial \alpha_k} &= 2\alpha_k P_k - B Z_k = \eta, \\
\frac{\partial L(\alpha, \eta)}{\partial \eta} &= \sum_{k=0}^{K} \alpha_k - 1 = 0.
\end{align*}
\]

Then, \( \alpha_k \) can be solved as follows,

\[
\alpha_k = \frac{\left( \frac{1}{\| P_k - B Z_k \|_F^2} \right)^{1/(r-1)}}{\left( \sum_{k=0}^{K} \frac{1}{\| P_k - B Z_k \|_F^2} \right)^{1/(r-1)}}. \quad (17)
\]

Because \( \| P_k - B Z_k \|_F^2 \geq 0 \), 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

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Nodes</th>
<th># Edges</th>
<th># Attributes</th>
<th># Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Citeseer</td>
<td>3,327</td>
<td>4,732</td>
<td>3,703</td>
<td>6</td>
</tr>
<tr>
<td>BlogCatalog</td>
<td>5,196</td>
<td>171,743</td>
<td>8,189</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1. Dataset description
Table 2. Node classification results (d=100)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Modes</th>
<th>10%</th>
<th>30%</th>
<th>50%</th>
<th>70%</th>
<th>90%</th>
<th>10%</th>
<th>30%</th>
<th>50%</th>
<th>70%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>63.71</td>
<td>73.50</td>
<td>78.83</td>
<td>80.29</td>
<td>81.20</td>
<td>61.02</td>
<td>71.65</td>
<td>77.63</td>
<td>79.08</td>
<td>79.83</td>
<td></td>
</tr>
<tr>
<td>Node2vec</td>
<td>67.10</td>
<td>77.30</td>
<td>81.22</td>
<td>82.68</td>
<td>83.52</td>
<td>66.56</td>
<td>76.50</td>
<td>80.14</td>
<td>81.61</td>
<td>82.28</td>
<td></td>
</tr>
<tr>
<td>TADW</td>
<td>81.50</td>
<td>84.97</td>
<td>85.78</td>
<td>86.23</td>
<td>86.93</td>
<td>79.71</td>
<td>83.35</td>
<td>84.26</td>
<td>84.44</td>
<td>85.35</td>
<td></td>
</tr>
<tr>
<td>HSCA</td>
<td>75.21</td>
<td>81.25</td>
<td>85.10</td>
<td>85.97</td>
<td>86.38</td>
<td>73.42</td>
<td>80.10</td>
<td>84.01</td>
<td>84.41</td>
<td>84.82</td>
<td></td>
</tr>
<tr>
<td>LANE</td>
<td>67.21</td>
<td>70.15</td>
<td>73.38</td>
<td>76.91</td>
<td>80.81</td>
<td>66.39</td>
<td>68.49</td>
<td>72.67</td>
<td>75.32</td>
<td>79.95</td>
<td></td>
</tr>
<tr>
<td>NetHash</td>
<td>81.24</td>
<td>85.06</td>
<td>86.19</td>
<td>86.56</td>
<td>86.98</td>
<td>79.64</td>
<td>83.43</td>
<td>84.58</td>
<td>84.87</td>
<td>85.82</td>
<td></td>
</tr>
<tr>
<td>BANE</td>
<td>81.88</td>
<td>85.32</td>
<td>86.35</td>
<td>87.06</td>
<td>88.30</td>
<td>80.23</td>
<td>84.26</td>
<td>85.19</td>
<td>85.76</td>
<td>87.11</td>
<td></td>
</tr>
<tr>
<td>LQANR</td>
<td>83.00</td>
<td>85.91</td>
<td>86.74</td>
<td>87.27</td>
<td>88.27</td>
<td>82.21</td>
<td>84.79</td>
<td>85.57</td>
<td>85.95</td>
<td>86.95</td>
<td></td>
</tr>
<tr>
<td>DeepWalk</td>
<td>64.32</td>
<td>49.06</td>
<td>54.41</td>
<td>56.16</td>
<td>56.31</td>
<td>40.57</td>
<td>45.65</td>
<td>49.33</td>
<td>50.32</td>
<td>49.17</td>
<td></td>
</tr>
<tr>
<td>Node2vec</td>
<td>48.56</td>
<td>55.77</td>
<td>62.55</td>
<td>63.66</td>
<td>63.69</td>
<td>46.78</td>
<td>53.92</td>
<td>58.09</td>
<td>59.42</td>
<td>60.47</td>
<td></td>
</tr>
<tr>
<td>TADW</td>
<td>69.38</td>
<td>71.48</td>
<td>72.18</td>
<td>72.75</td>
<td>72.84</td>
<td>61.80</td>
<td>64.62</td>
<td>65.83</td>
<td>66.54</td>
<td>67.03</td>
<td></td>
</tr>
<tr>
<td>HSCA</td>
<td>69.47</td>
<td>71.54</td>
<td>72.61</td>
<td>73.66</td>
<td>73.96</td>
<td>61.62</td>
<td>64.80</td>
<td>65.98</td>
<td>66.70</td>
<td>67.21</td>
<td></td>
</tr>
<tr>
<td>LANE</td>
<td>53.81</td>
<td>60.72</td>
<td>61.65</td>
<td>63.58</td>
<td>67.77</td>
<td>50.33</td>
<td>57.05</td>
<td>58.14</td>
<td>60.63</td>
<td>63.60</td>
<td></td>
</tr>
<tr>
<td>NetHash</td>
<td>69.63</td>
<td>71.89</td>
<td>72.35</td>
<td>73.16</td>
<td>72.85</td>
<td>61.97</td>
<td>64.47</td>
<td>66.11</td>
<td>67.36</td>
<td>67.90</td>
<td></td>
</tr>
<tr>
<td>BANE</td>
<td>70.24</td>
<td>72.55</td>
<td>73.78</td>
<td>74.55</td>
<td>75.08</td>
<td>62.37</td>
<td>65.73</td>
<td>67.63</td>
<td>68.44</td>
<td>69.35</td>
<td></td>
</tr>
<tr>
<td>LQANR</td>
<td>70.41</td>
<td>72.73</td>
<td>73.80</td>
<td>74.67</td>
<td>75.20</td>
<td>62.94</td>
<td>66.11</td>
<td>67.80</td>
<td>68.72</td>
<td>69.62</td>
<td></td>
</tr>
</tbody>
</table>

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 \( \beta \) 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 continuous representation methods. The results show that low-bit representation does not necessarily lead to embedding accuracy loss compared to continuous 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.
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 BlogCatalog contains more complicated structure and attribute information than the other datasets.

**Proximity matrix order \( K \).** We test node classification with different \( K \). The results on Cora with \( K \) ranging 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 \( \alpha_k \) 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 \( \alpha_k \) 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


