SNEQ: Semi-Supervised Attributed Network Embedding with Attention-Based Quantisation

Tao He,[1] Lianli Gao,[2] Jingkuan Song,[2] Xin Wang,[3] Kejie Huang,[4] Yuan-Fang Li[1]*

[1]Faculty of Information Technology, Monash University, Clayton, Victoria 3800
[2]Center for Future Media, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731
[3]College of Intelligence and Computing, Tianjin University, Jinnan, Tianjin 300350
[4]College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, Zhejiang 310007
{tao.he, yuanfang.li}@monash.edu, lianli.gao@uestc.edu.cn, jingkuan.song@gmail.com,
wangx@tju.edu.cn, huangkejie@zju.edu.cn,

Abstract

Learning accurate low-dimensional embeddings for a network is a crucial task as it facilitates many network analytics tasks. Moreover, the trained embeddings often require a significant amount of space to store, making storage and processing a challenge, especially as large-scale networks become more prevalent. In this paper, we present a novel semi-supervised network embedding and compression method, SNEQ, that is competitive with state-of-art embedding methods while being far more space- and time-efficient. SNEQ incorporates a novel quantisation method based on a self-attention layer that is trained in an end-to-end fashion, which is able to dramatically compress the size of the trained embeddings, thus reduces storage footprint and accelerates retrieval speed. Our evaluation on four real-world networks of diverse characteristics shows that SNEQ outperforms a number of state-of-the-art embedding methods in link prediction, node classification and node recommendation. Moreover, the quantised embedding shows a great advantage in terms of storage and time compared with continuous embeddings as well as hashing methods.

Introduction

Network embedding methods learn an encoder system that converts nodes into low-dimensional float-valued vectors, aiming at preserving the original network information as much as possible. It has a wide range of applications and has attracted significant attention. Many algorithms have been proposed, including DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015), Node2vec (Grover and Leskovec 2016), and NetMF (Qiu et al. 2018), among many others. Many of these methods (Perozzi, Al-Rfou, and Skiena 2014; Lian et al. 2018; Qiu et al. 2018) are trained in an unsupervised manner, completely disregarding the valuable node label information, leaving significant room for improvement on node classification performance. Graph2Gauss (G2G) (Bojchevski and Günnemann 2018) is an unsupervised method that embeds each node into a Gaussian distribution and achieves good performance in link prediction and node classification.

Label information has also been incorporated in embedding methods. HCGE (Dos Santos, Piwowarski, and Gallinari 2016) is a supervised method for node classification with good improvement. Kipf and Welling (2017) first proposed an end-to-end semi-supervised graph convolutional network (GCN) architecture via a localised first-order approximation. H-GCN (Hu et al. 2019) explored a hierarchical GCN that aggregates nodes with similar neighbourhood structure into hypernodes. Graph Attention Network (GAT) (Veličković et al. 2018) is an attention-based graph convolutional network, focusing on weighted aggregation of neighbourhood information. A major drawback of most existing graph embedding methods is the high computational cost on retrieval tasks such as node recommendation. Retrieval tasks rely on expensive distance measures on float-valued vectors such as dot product. On large networks with hundreds of thousands or millions of nodes, retrieval becomes prohibitively expensive. At the same time, it also imposes a significant storage overhead.

Embedding vectors usually contain a large amount of redundant information. Hashing and quantisation methods (Lian et al. 2018; Du and Wang 2014; Zhang et al. 2014) have been proposed to compress such learned features. The main idea of quantisation is to learn a series of *codebooks* to approximate the actual embeddings. In other words, the original embeddings are represented by their closest centres. Although quantisation and hashing methods introduce additional information loss, they significantly reduce storage footprint and retrieval time. By pre-generating a lookup table, recent quantisation methods (Cao et al. 2017a) can achieve a retrieval speed similar to hashing. With the additional codebooks, quantisation also achieves a better retrieval accuracy than hashing.

In this paper, we present SNEQ, a novel semi-supervised network embedding method combined with quantisation learned through a self-attention layer. The embedding component utilises both node labels, network proximity information, as well as node attributes. Meanwhile, our self-attention based quantisation learns a compact and accurate codebook in an end-to-end fashion. SNEQ takes all of the

*Lianli Gao, Jingkuan Song and Yuan-Fang Li are joint corresponding authors.

above properties into account and jointly optimises on them. Our experiments on four real-world datasets with diverse characteristics show that SNEQ outperforms existing state-of-the-art embedding methods on the tasks of link prediction, node classification, and node recommendation. Comparing to well-known hashing methods, SNEQ also achieves substantially better performance on these tasks while maintaining a comparable storage footprint and time cost for retrieval. In summary, our main contributions are fourfold:

- We propose a semi-supervised embedding scheme, semantic margin, to preserve the supervised information into the embedding space while maintaining the network structure information as much as possible.
- We develop an adaptive margin component, which can precisely preserve more discriminative neighbour information, compared with a fixed margin.
- We explore a self-attention-based quantisation compression method that drastically reduces the size of learned embedding and can be optimised via a continuous way.
- In our extensive experiments on link prediction, node classification, and node recommendation, our method outperforms state-of-the-art methods, including both continuous embedding methods and discrete hashing methods.

Methodology

In this section, we will present our network embedding and quantisation technique in detail. The overall architecture of SNEQ is shown in Fig. 1.

Problem definition. Let $G = (V, X, E, Y)$ represent an attributed network. $V = \{v_1, v_2, \ldots, v_N\}$ is a set of N nodes. $X \in \mathbb{R}^{N \times D}$ represents D-dimensional node attributes. $E = \{e_{i,j}\}_{i,j=1}^{N}$ denotes the edge set of the network. Specifically, $e_{i,j} = 1$ if v_i has a connection to v_j, otherwise $e_{i,j} = 0$. $Y = \{y_1, y_2, \ldots, y_N\}$ is the label set of each node. To simplify the notation, we use an adjacent matrix $A \in \mathbb{R}^{N \times N}$ to represent the edge connections of a network. Our embedding goal is to embed the nodes of the network G into low-dimensional vectors $\mathbf{Z} \in \mathbb{R}^{N \times L}$, where $L \ll D$. Simultaneously, the quantisation module is responsible for encoding the real-valued vectors $\mathbf{Z}$ into short compact codes $\mathbf{Q} \in \mathbb{R}^{N \times M}$ and a set of codebooks $\mathcal{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_M\}$, where $\mathbf{C}_i \in \mathbb{R}^{K \times L}$. M is the number of codebooks in the embedding space $\mathbf{Z}$, K is the number of centres in each codebook $\mathbf{C}_i$, where $L = M * K$.

Graph Structure Embedding via Adaptive Margin

Our method preserves the network structure information by learning and optimising an adaptive margin loss. For an attributed network G, our model takes the attributes X as the input. We use a multi-layer perceptron (MLP) to encode each node v_i, initialised with node attributes x_i for an attributes network.

$$h_i^{(1)} = f(\mathbf{w}_i^{(1)} x_i + \mathbf{b}_i^{(1)}) \tag{1}$$

where i denotes the hidden layer and $f(.)$ is the activation function. For notational convenience, the output of the last layer of the encoder is denoted by $\mathbf{Z}$.

Intuitively, in a homogeneous network, the shorter distance between two nodes is, the more similar they are. In this paper, we propose to use the shortest distance between two nodes (in terms of hops) to measure their structural similarity. Theoretically, we can regard embedding as a procedure to project the shortest distance information in the original network into the embedding space. Our motivation is how to precisely preserve the different shortest distance into the embeddings. We adopt the widely-used metric learning technique, which has been successfully applied in computer vision tasks such as person re-identification (Dong et al. 2018), face recognition (Schroff, Kalenichenko, and Philbin 2015), and hashing-based image retrieval (Song et al. 2018), whose main goal is to ensure that data points in the feature space have consistent distance with their original data distribution space. In this work, we leverage the triplet loss to learn node distances. The adaptive margin loss ℓ_a is defined as below:

$$\ell_a = \frac{1}{N} \min_{\mathbf{W}^e} \Big(\sum_{tr(a,p,n)}^{N} \max(D_{(a,p)} - D_{(a,n)} + \delta_{a,n} - \delta_{a,p}, 0) \Big) \tag{2}$$

where $\mathbf{W}^e$ denotes the parameters of the encoder; $D_{(.)}$ is the distance function to measure two nodes in the embedding space, for which we chose Euclidean distance (L_2 norm); and $\delta_{i,j}$ denotes the shortest distance from node v_i to node v_j. The function $tr(.)$ is a triplet sampling function that, given an anchor node, selects a positive node v_p and a negative node v_n. Concretely, given an anchor node v_a, v_p and v_n are sampled under the condition of $\delta_{a,p} < \delta_{a,n}$.

In Eq. 2, the adaptive margin, $\delta_{a,n} - \delta_{a,p}$, is the key term that defines the disparity between two shortest distances, that is, the anchor node to the positive node and the negative node. To model the difference between a node's neighbours, we adopt an adaptive margin to present it. Specifically, the distant neighbour is divided by a large margin while the close neighbour is clustered by a small distance. There are many different algorithms to calculate the shortest distance δ, including Dijkstra's and Floyd's algorithms. In this paper we adopt a fast strategy based on matrix multiplication (Sankowski 2005).

Semi-supervised Semantic Embedding

In many situations, a subset of the nodes in a network are given labels, and intuitively, nodes with the same label should be closer in the embedding space than otherwise. Based on this intuition, we explore a semi-supervised strategy to preserve label information into embeddings $\mathbf{Z}$. The goal is to cluster same-label nodes with a small margin but separate those with different labels by a large margin. The general idea is explained in Figure 1, where the differently labelled nodes (different colours) should be separated by large margins.

A key optimisation of our semi-supervised learning procedure is that we do not need to use all labelled nodes to train the model. This is due to the hypothesis that nodes with a small shortest distance are more likely to belong to the same class. Generally, as long as a small part of nodes

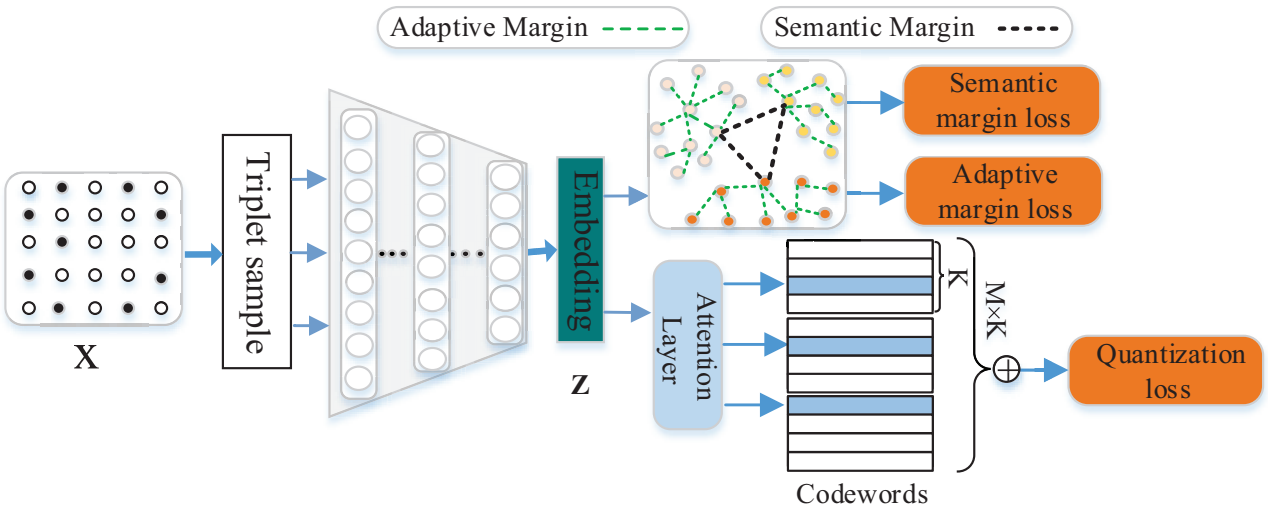


Figure 1: The high-level architecture of our framework. SNEQ consists of three main parts: (1) network structure preservation through adaptive margin learning, (2) semi-supervised semantic margin learning, and (3) self-attention-based quantisation learning. The adaptive margin loss aims at forcing the embedding space to preserve the original neighbourhood information. The semantic margin loss is responsible for separating nodes with different labels. The self-attention-based quantisation loss guarantees the reconstructed embeddings close to the original one.

in the same class are adequately separated, other nodes with small shortest distances to those separated nodes can also be separated. We only need to ensure that the semantic margin is large enough because if too small, the adaptive margin will conflict with it and degrade the performance of the classification. Instead, provided that the semantic margin is large enough, we can think that the semantic margin divides the embedding space into several discriminative subspaces, each of which stands for a community with the same class label, in which the adaptive margin is applied to preserve their neighbours structure information. The formulation of our semantic margin loss ℓ_c is given as the following:

$$\ell_c = \frac{1}{T} \min_{\mathbf{W}^e} \sum_{i,j=1}^{T} \left(D_{(i,j)} - S_{i,j}\right)^2 \tag{3}$$

where T is the sample size for semi-supervision; $D_{(.)}$ is the same distance function as in Eq. 2; and $S_{i,j}$ represents the constant semantic margin and is formulated as below:

$$S_{i,j} = \begin{cases} 0, & \text{if } y_i \cap y_j \neq \emptyset \\ M_c, & \text{otherwise} \end{cases} \tag{4}$$

where y_i is the label of node v_i and M_c is a constant defining the semantic margin.

The aim of Eq. 3 is to force the same-labelled nodes to be close each other, but nodes with different labels to be at a distance of M_c. It is worth noting that the value of T is important to node classification performance, as a large T value generally improves node classification result, due to more supervised information added. At the same time, a larger T could negatively affect graph structure learning due to the fact that some nodes which are supposed to be neighbours are separated by a large margin of M_c, leading to a degradation in link prediction performance. Hence, the choice of T is a trade-off between two competing factors: structural and semantic information. In the experiments, we test the impact of different T.

Self-attention for Deep Quantisation

As we motivated previously, our quantisation strategy aims at reducing the size of the embedding vectors. Ideally, this reduction would improve both space and time efficiency while preserving task performance. Product quantisation (PQ) (Jégou, Douze, and Schmid 2011) is a well-known quantisation technique. PQ quantises node embedding vectors $\mathbf{Z}$ into M codebooks $\mathcal{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_M\}$, and each codebook $\mathbf{C}_j$ consists of K centres(codewords), which can be seen as K subspaces in the embedding space. For each codebook $\mathbf{C}_j$, we use a 1-of-K indicator vector $\mathbf{b}_{ij} \in \{0,1\}^{1\times K}$ to assign one centre to $\mathbf{z}_i$. It is worth noting that there is only one 1 in each $\mathbf{b}_{ij}$, which represents the closest centre to $\mathbf{z}_i$, while all the other values of $\mathbf{b}_{ij}$ are 0. The general quantisation loss ℓ_q is calculated by:

$$\ell_q = \frac{1}{N} \min_{\mathbf{W}^e, \mathbf{C}} \sum_{i=1}^{N} \left\| \mathbf{z}_i - \sum_{j=1}^{M} \mathbf{b}_{ij} \mathbf{C}_j \right\|^2 \tag{5}$$

This loss function aims at reducing the reconstruction loss between the embedding vector $\mathbf{z}_i$ and its quantisation codes. A smaller ℓ_q means better reconstruction performance from the quantisation codes. Nevertheless, the optimisation of Eq. 5 becomes a discrete code search problem, that is, we need to find M binary codes $\mathbf{b}$ in order to approximate $\mathbf{z}$ as much as possible, similar to the learning to hash problem (Wang et al. 2018). The direct optimisation is an NP-hard problem, and many recent works (Li, Wang, and Kang 2016; Zhang et al. 2014) have proposed to use relaxation

strategies to approximate the learned binary codes. Thus, we can rewrite Eq. 5 in a continuous form:

$$\begin{aligned}\ell_q =& \frac{1}{N}\min_{\mathbf{W}^e,\mathbf{C}}\sum_{i=1}^{N}\left\|\mathbf{z}_i-\sum_{j=1}^{M}\mathbf{u}_{ij}\mathbf{C}_j\right\|^2 \text{ such that} \\ & \mathbf{u}_{ij}\propto\mathbf{b}_{ij},\forall j=1,2,\ldots,N \\ & \mathbf{u}_{ij}\in[0,1]^K,\forall j=1,2,\ldots,N \\ & \mathbf{b}_{ij}\in\{0,1\}^K,\|\mathbf{b}_{ij}\|_1=1,\forall j=1,2,\ldots,N\end{aligned} \tag{6}$$

The drawback of this approach is that the relaxation will introduce large quantisation error (Cao et al. 2017b; Song et al. 2018). Recently, self-attention (Zhang et al. 2018), aimed at informing the model what is important, also gains significant success in computer vision and natural language processing. Inspired by the attention mechanism, we propose a continuous optimisation method based on a self-attention mechanism. Concretely, in Eq. 6, the ideal case is that the largest number in $\mathbf{u}_{ij}$ is extremely close to 1 while all the other dimensions are close to 0. The attention mechanism is naturally suitable for this case as it amplifies the important part and lessens the unimportant. Specifically, our goal is to convert each $\mathbf{u}_{ij}$ into an approximate one-hot vector as $\mathbf{b}_{ij}$. To achieve it, we multiply the continuous vector $\mathbf{u}_{ij}$ with an attention weight vector $\mathbf{r}_{ij}$ in a pointwise manner, where $\sum_j \mathbf{r}_{ij}=1$. Then, we apply the L_1 norm to regularize the approximate results. The self-attention formation is given as follows:

$$\mathbf{u}_{ij}=\|\mathbf{u}_{ij}\otimes\mathbf{r}_{ij}\|_1 \tag{7}$$

where $\otimes$ is the Hadamard product (matrix pointwise multiplication) and $\mathbf{r}_{ij}$ is the attention weights. Consequently, Eq. 6 becomes:

$$\ell_q=\frac{1}{N}\min_{\mathbf{W}^e,\mathbf{W}^a,\mathbf{C}}\left(\sum_{i=1}^{N}\left\|\mathbf{z}_i-\sum_{j=1}^{M}\|\mathbf{u}_{ij}\otimes\mathbf{r}_{ij}\|_1\mathbf{C}_j\right\|^2\right) \tag{8}$$

where $\mathbf{W}^a$ denotes the parameters of the self-attention module. The attention weight $\mathbf{r}_{ij}$ is calculated in an attention layer by the following formulation:

$$\mathbf{r}_{ij}=\mathrm{softmax}(f(\mathbf{W}_s\mathbf{z}_{ij}+\mathbf{b}_s)) \tag{9}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are the weight and bias parameters of the self-attention layer, $\mathbf{W}^a=\mathbf{W}_s\cup\mathbf{b}_s$ for notational convenience, and $\mathbf{z}_{ij}$ denotes the j-th block when we divide $\mathbf{z}_i$ into M blocks each of which is of dimension K. Since all the parameters in Eq. 8 are continuous, we can optimise Eq. 8 by continuous optimisation methods.

Out-of-sample Extension

During the test stage, due to the fact that the approximate code $\mathbf{u}_{ij}$ is not completely equal to the one-hot vector $\mathbf{b}_{ij}$, we choose the index of the maximum element of $(\mathbf{z}_{ij}\otimes\mathbf{r}_{ij})$ as the approximate codeword and the formulation is following:

$$\mathbf{b}_{ij}=\arg\max(\mathbf{z}_{ij}\otimes\mathbf{r}_{ij}) \tag{10}$$

where the $\arg\max(.)$ function is used to set the index of the maximum element in a vector to 1 and other indices to 0.

After obtaining the quantised codes, we can reconstruct the approximate embedding vectors by adding M codewords:

$$\mathbf{z}_i^*=\sum_{j=1}^{M}\mathbf{b}_{ij}\mathbf{C}_j \tag{11}$$

where $\mathbf{b}_{ij}$ is obtained by Eq. 10. $\mathbf{z}_i^*$ can be seen as the approximation of $\mathbf{z}_i$.

Distance Computation

Similarity-based retrieval methods for quantisation (Du and Wang 2014) and hashing (Song et al. 2018) such as the approximate nearest neighbour (ANN) algorithm can be applied on $\mathbf{z}_i^*$ to significantly reduce retrieval time. For product quantisation, Symmetric Distance Computation (SDC) (Jégou, Douze, and Schmid 2011) is a powerful similarity metric that calculates the inner-product similarity between a query point $\mathbf{q}$ and a database points $\mathbf{z}_i$. The SDC computation is defined as the following:

$$SDC(\mathbf{q},\mathbf{z}_i)=\sum_{j=1}^{M}(\mathbf{b}_{qj}\mathbf{C}_j)*\mathbf{z}_i^* \tag{12}$$

In fact, in Eq. 12, many computations are repetitive as the codebooks $\mathcal{C}$ are fixed and only consist of $M*K$ codewords. Hence, we can pre-compute the similarity among those codewords and store them in a lookup table. At test time, the similarity computation can be done with $O(M)$ time complexity.

Learning

SNEQ learns network embedding and quantisation in an end-to-end architecture, which integrates adaptive margin loss (Eq. 2), semi-supervised semantic margin loss (Eq. 3), and self-attention-based quantisation loss (Eq. 8) into a joint optimisation problem:

$$\ell=\min_{\mathbf{W^e},\mathbf{W^a},\mathbf{C}}(\ell_a+\alpha\ell_c+\beta\ell_q) \tag{13}$$

where α and β are the balance parameters to trade-off the importance of each part. All the parameters in Eq. 13 are continuous and trained by mini-batch stochastic gradient descent. The update rules are formulated as follows:

$$\begin{aligned}W^e \leftarrow & W^e-\eta\times(\frac{\partial L_a}{\partial W^e}+\alpha\frac{\partial L_c}{\partial W^e}+\beta\frac{\partial L_q}{\partial W^a}\times\frac{\partial W^a}{\partial W^e}) \\ W^a \leftarrow & W^a-\eta\times(\beta\frac{\partial L_q}{\partial W^a}) \\ \mathcal{C} \leftarrow & \mathcal{C}-\eta\times(\beta\frac{\partial L_q}{\partial C})\end{aligned} \tag{14}$$

where η is the learning rate.

Experiments

Datasets. We evaluate our method on four real-world networks. Brief statistics of the datasets are shown in Table 1. It is worthing noting that cDBLP (Yang and Leskovec 2015) is

a large-scale unattributed network. We conduct experiments on the standard tasks of link prediction, node classification and node recommendation. We also evaluate the efficiency and effectiveness of quantisation.

Table 1: Brief statistics of the datasets.

G	Cora_ML	Citeseer	DBLP	cDBLP
$\|V\|$	2,995	4,230	17,716	317,080
$\|E\|$	8,416	5,358	105,734	1,049,866
Attr.	Yes	Yes	Yes	No
Labels	7	6	6	5,000

Implementation Details. Our encoder consists of two dense layers of 512 units and 128 units respectively[1]. The dimension of the network embeddings is set to 128. The learning rate η is set as 0.001, and batch size is 100. Hyperparameters α and β are not fixed but tuned in an unsupervised way similar to (Xie et al. 2018). Specifically, $\alpha = \frac{0.1}{1+e^{-(\omega\mu)}}$, where ω is a constant value 0.5, and μ is the training progress from 0 to 1, while β is set as $1.0 - \frac{1}{1+e^{-(\omega\mu)}}$. For quantisation, we set $M = 16$ and $K = 8$ by default, but also test the impact of different values of M and K, which can be seen in experiments below. The amount of labelled nodes T used in semi-supervised training (as defined in Eq. 3) is set to 10% of $|V|$ by default. For a fair comparison with learning to hash methods, these methods' dimension is set as 48 bits, equal to our quantisation code length, i.e. $48 = 16 * \log(8)$. All experiments were performed on a workstation with 256 GB memory, 32 Intel(R) Xeon(R) CPUs (E5-2620 v4 @ 2.10GHz) and 8 GeForce GTX 1080Ti GPUs. For each model a maximum approx. 100 GB of memory was allocated.

Baselines. We compared our method with some recent state-of-the-art graph embedding methods, including both semi-supervised and unsupervised methods. We divide them into two groups: continuous embeddings and discrete embeddings. The continuous group includes the following semi-supervised methods: SEANO (Liang et al. 2018), GAT (Veličković et al. 2018) and H-GCN (Hu et al. 2019), a hierarchical extension of GCN; as well as unsupervised methods: DeepWalk (DW) (Perozzi, Al-Rfou, and Skiena 2014), TADW (Yang et al. 2015), Graph2Gauss (G2G) (Bojchevski and Günnemann 2018), and ONE (Bandyopadhyay, Lokesh, and Murty 2019).

The discrete methods include SH (Datar et al. 2004), DCF (Zhang et al. 2016), NetHash (Wu et al. 2018) and BANE (Yang et al. 2018). The discrete variant (with quantisation) of our model is denoted SNEQ$^{\#}$. In addition, we also test a more efficient setting of our model, denoted SNEQ$^{\#}_{8\times16}$, with $M = 8$ and $K = 16$ bits respectively.

Link Prediction

Link prediction is a standard task to evaluate the performance of network embedding methods, aiming at measuring the preservation of the network structure. Specifically, we randomly select 5% and 10% edges as the validation and test set respectively, similar to Graph2Gauss (Bojchevski and Günnemann 2018). Following the convention, we use AUC as the performance metric.

Table 2 summarises the link prediction results of the continuous embedding methods (upper block) and of the discrete methods (lower block). N/A denotes the algorithms that did not finish within 24 hours or within 100GB memory. From the upper block of Table 2, it can be observed that our method consistently outperforms other continuous network embedding methods, except lower than G2G by 0.42 percentage points on the Cora_ML dataset but on the large dataset cDBLP, SNEQ exceeds G2G by 3.94 percentages.

In the lower block of the table, it can be seen that our quantisation embedding is substantially superior to all the others, with at least 2∼3 percentage points better than the second best method BANE. The possible reason is that as the embedding dimension is relatively low, hash-based methods suffer larger loss on structure information without an additional schema (codebooks). In contrast, with the same dimension, the code words of our quantisation are able to preserve much global information. We can also observe another advantage of our method, that SNEQ can handle large-scale networks such as cDBLP while many hashing methods, including DCF and BANE, cannot. This is because all of them depend on SVD, which incurs high memory usage.

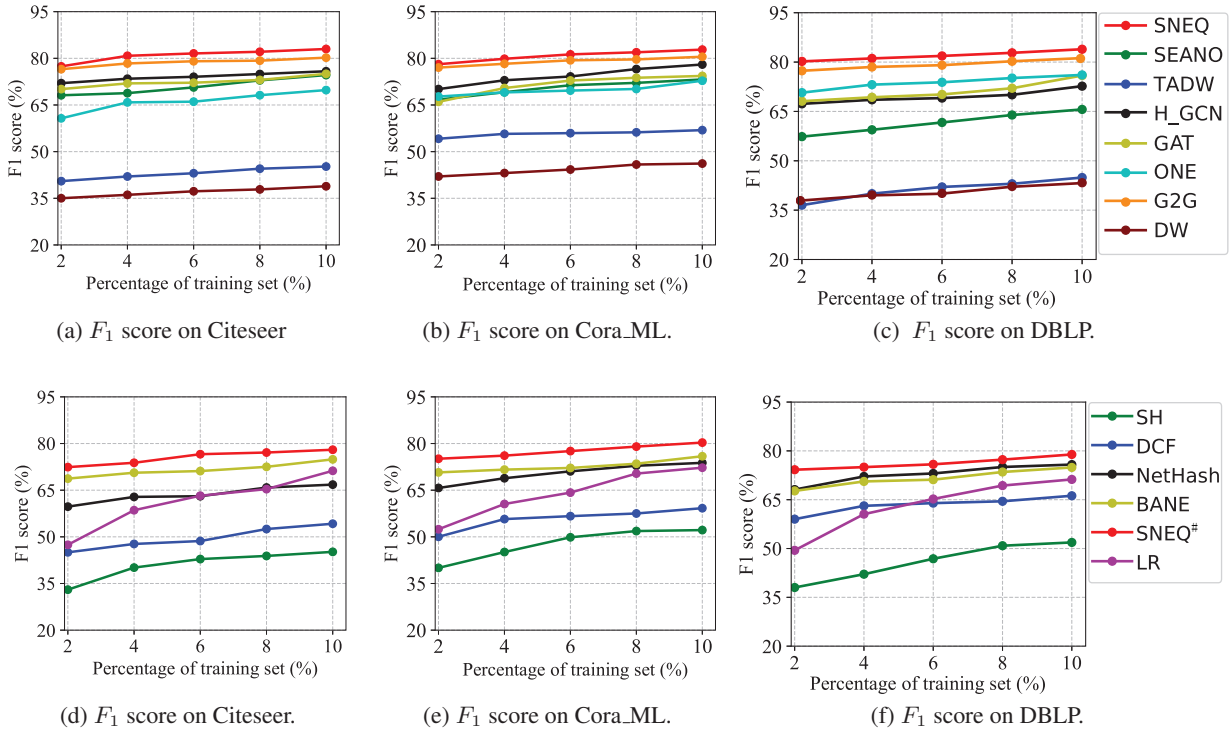
Table 2: Link prediction results of the continuous methods (top) and discrete methods (bottom). Best results in each block are **bolded**.

Models	Citeseer	Cora_ML	DBLP	cDBLP
DW	82.14	81.26	71.67	66.84
TADW	85.58	84.40	78.81	71.32
GAT	90.43	92.03	89.19	81.89
SEANO	83.36	85.25	88.43	80.15
H-GCN	90.51	91.69	89.17	82.57
ONE	93.56	92.73	93.62	N/A
G2G	96.39	**97.63**	97.71	88.18
SNEQ	**96.47**	97.21	**97.74**	**92.12**
SH	80.42	85.10	82.31	80.57
DCF	79.47	80.17	83.12	N/A
NetHash	85.14	86.84	87.61	83.18
BANE	90.09	91.05	90.63	N/A
SNEQ$^{\#}_{8\times16}$	91.35	92.85	93.15	90.79
SNEQ$^{\#}$	**92.72**	**93.18**	**93.60**	**91.24**

Node Classification

We tested our method on three attributed datasets: Citeseer, Cora_ML, and cDBLP. We use the one-vs-the-rest logistic regression as the classifier, repeat the prediction for 10 times, and report the average of Macro-F_1 and Micro-F_1 results. Figure 2 shows the results, where figures (a)∼(c) are the results of the continuous embedding methods while (d)∼(f) present the discrete results. In each experiment, a varying percentage (e.g. 2, 4, 6, 8, 10%) of nodes are sampled for training the classifier. From Figure 2, we can make the following observations:

[1]Our code is released at https://github.com/htlsn/SNEQ.



(a) F_1 score on Citeseer (b) F_1 score on Cora_ML. (c) F_1 score on DBLP.

(d) F_1 score on Citeseer. (e) F_1 score on Cora_ML. (f) F_1 score on DBLP.

Figure 2: Node classification results on the three attributed networks. The top row shows the results of continuous embeddings while the bottom row shows the results of discrete embedding results.

(1) From figures (a)∼(c), we can find that our continuous embeddings are superior to the counterparts on all of the three datasets. Specifically, SNEQ is about 1∼2 percentage points higher than the second best method G2G. H_GCN generally outperforms GAT because H_GCN performs a graph coarsening operation grouping the nodes with same neighbours into one node. However, the node refining operation in H-GCN highly depends on the initial network structure and the complexity or sparsity of networks. In contrast, our semantic margin module does not depend on the neighbourhood structure and only considers the pairwise distance between different nodes, and thus can generalise better for more complex networks.

(2) In the discrete embedding results, our quantisation shows a great advantage compared with the hashing-based methods. Specifically, SNEQ outperforms the best hashing method BANE about 2∼4 percentage points on the three datasets. The possible reason is that the short binary codes lead to large loss of information in hashing methods. Meanwhile, as node attributes are binary and can be regarded as hash codes, we also test the classification performance of the attributes via a logistic regression classifier, denoted as LR. It can be observed that LR exceeds many hash methods such as NetHash and DCF. This is because the dimensions reduce significantly from $\sim 3,000$ to 48 bits and many meaningful features are abandoned.

Besides, SH is trained by our continuous embeddings. However it suffers a hug performance degradation, compared from the continuous embeddings, of nearly 30 percentage points. We conjecture that there are two main reasons: (1) the dimension reduces to about one third, leading to information loss; and (2) the procedure of learning to hash is not an end-to-end procedure, i.e., first learning embedding vectors and then training the hash function. In contrast, SNEQ avoids these issues in two ways: (1) the codewords can preserve as much information as possible, even though the quantised code is short; and (2) the embedding and learning to quantise steps are unified in one network and jointly trained by the back propagation scheme.

Node Recommendation

In node recommendation, given a node, an embedding model ranks all nodes according to a distance measurement and recommends the closest node. This task is widely applied in social and commercial networks, specially for the recommendation scenario. Following the settings of INH-MF (Lian et al. 2018), 90% of neighbours of each node are used to train each embedding model while the remaining 10% neighbours are reserved for testing. We use NDCG@50 as the evaluation metric, and the final results are averaged over 10 runs. Table 3 shows the results of node recommendation, where we can obtain the following observations:

(1) In terms of the continuous embedding methods, our method outperforms others on the four dataset. Specifically, our method is superior to the most competitive methods H-GCG and G2G about 2∼3 percentage points. G2G adopts a rank loss to preserve neighbour information, but one difference from ours is that we learn adaptive margins for different

neighbours: more distant neighbours will result in a larger margin, i.e., a larger penalty, while closer neighbours are separated by a relative small margin. Hence, our method is more advantageous for the preservation of first-order proximity.

(2) In the second part of Table 3, our method shows even larger superiority, averagely 4∼7 percentage points higher than the best hash method NetHash. SH exhibits worst performance, and the reason is that hash codes are learned separately in a two-step way. It is worth noting that because of excessive memory usage by DCF and BANE, we did not obtain their results on cDBLP.

Table 3: Node recommendation results on the four networks, measured by NDCG@50. The upper block shows the results of original continuous embedding methods while the lower block contains results are calculated from embeddings reconstructed from discrete codes. Best results in each block are **bolded**.

Method	Citeseer	Cora_ML	DBLP	cDBLP
DW	35.15	41.22	11.42	16.62
SEANO	56.27	47.17	12.05	17.26
GAT	58.48	46.29	10.82	19.37
ONE	54.62	46.87	13.13	N/A
G2G	58.51	48.42	12.07	18.63
H-GCN	59.11	47.07	10.73	20.53
SNEQ	**61.25**	**50.40**	**14.79**	**23.62**
SH	36.54	35.39	3.56	10.32
DCF	21.64	20.05	4.74	N/A
NetHash	41.28	37.54	4.62	12.39
BANE	35.64	37.05	8.74	N/A
SNEQ$^{\#}{}_{8\times16}$	42.49	40.02	9.03	18.75
SNEQ$^{\#}$	**43.41**	**41.82**	**9.79**	**19.05**

Space and Time Efficiency of SNEQ$^{\#}$

The space and time efficiency results are shown in Table 4 and 5. Empirically, it can be observed in Table 4 that, the larger the network, the more reduction in storage quantisation brings: for cDBLP, the original embeddings are approx. 60 times larger than SNEQ$^{\#}$ (row four). It can be seen that the alternative setting of our quantisation (row three), where $M = 8$ and $K = 16$, incurs much less space footprint, and even less than that of hashing for the large networks but it is inferior on other tasks, as can be seen in Table 2 (lower block) and Table 3. Compared to hashing methods such as SH, it can be figured out that quantisation uses slightly more space as it stores M codebooks. However, the evidence from the other tasks in the previous three subsections, we can observe hashing methods' inferior performance to quantisation. In fact, the additional storage cost of quantisation over hashing is the codebooks, at approx. 64KB for SNEQ$^{\#}$. Therefore, for larger networks hashing methods' savings in storage are minuscule and negligible given modern hardware.

Table 5 shows the average retrieval time of node recommendation in milliseconds. For real-valued embeddings, we use the Euclidean distance to measure node similarity, while for quantised codes we use SDC in Equation 12 to measure their similarity. Quantisation achieves up to 70 $\times$ retrieval speedup over the real-valued embeddings. Similar to our analysis on storage, setting $M = 8$ and $K = 16$ can further reduce retrieval time by 50%, though its link prediction performance declines slightly. Hashing methods, represented by SH, are more time-efficient than our model SNEQ$^{\#}$. However, as we noted above, this is achieved with degraded task performance.

Table 4: Storage cost (MB) of different embeddings.

Method	Cora_ML	Citeseer	DBLP	cDBLP
Float	2.145	1.513	9.562	155.31
SH	**0.031**	**0.026**	0.153	2.524
SNEQ$^{\#}{}_{8\times16}$	0.082	0.077	**0.138**	**1.382**
SNEQ$^{\#}$	0.098	0.089	0.211	2.697

Table 5: Average retrieval time cost (ms) of different embeddings.

Method	Cora_ML	Citeseer	DBLP	cDBLP
Float	42.7	71.2	191.2	3,923.5
SH	0.7	1.0	2.7	34.7
SNEQ$^{\#}{}_{8\times16}$	**0.5**	**0.8**	**2.3**	**26.1**
SNEQ$^{\#}$	1.0	1.5	4.4	59.6

Conclusion

In this paper, we propose SNEQ, an end-to-end network embedding and quantisation method. SNEQ is trained in a semi-supervised manner, which simultaneously preserves network structure and semantic information into the embedding space while compressing the embeddings by self-attention-based product quantisation. Specifically, we incorporate the adaptive margin loss for preserving network structure information, semantic margin loss for semantic space learning, and self-attention based quantisation loss to learn compact codes. In standard evaluation tasks on four diverse networks, our method outperforms state-of-the-art network embedding methods. In addition, due to quantisation, SNEQ significantly reduces storage footprint and accelerates query time for node recommendation.

Acknowledgement

Xin Wang is supported by the National Natural Science Foundation of China (Grant No. 61972275).

References

Bandyopadhyay, S.; Lokesh, N.; and Murty, M. N. 2019. Outlier aware network embedding for attributed networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 12–19.

Bojchevski, A., and Günnemann, S. 2018. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR 2018*.

Cao, Y.; Long, M.; Wang, J.; and Liu, S. 2017a. Deep visual-semantic quantization for efficient image retrieval. In *CVPR 2017*, 1328–1337.

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017b. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, 5608–5617.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM.

Dong, H.; Lu, P.; Zhong, S.; Liu, C.; Ji, Y.; and Gong, S. 2018. Person re-identification by enhanced local maximal occurrence representation and generalized similarity metric learning. *Neurocomputing* 307:25–37.

Dos Santos, L.; Piwowarski, B.; and Gallinari, P. 2016. Multilabel classification on heterogeneous graphs with Gaussian embeddings. In *ECML-PKDD 2016*, 606–622. Springer.

Du, C., and Wang, J. 2014. Inner product similarity search using compositional codes. *CoRR* abs/1406.4966.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

Hu, F.; Zhu, Y.; Wu, S.; Wang, L.; and Tan, T. 2019. Hierarchical graph convolutional networks for semi-supervised node classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 4532–4539.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(1):117–128.

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

Li, W.; Wang, S.; and Kang, W. 2016. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI 2016*, 1711–1717.

Lian, D.; Zheng, K.; Zheng, V. W.; Ge, Y.; Cao, L.; Tsang, I. W.; and Xie, X. 2018. High-order proximity preserving information network hashing. In *the 24th ACM SIGKDD*, 1744–1753. ACM.

Liang, J.; Jacobs, P.; Sun, J.; and Parthasarathy, S. 2018. Semi-supervised embedding in attributed networks with outliers. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, 153–161. SIAM.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Qiu, J.; Dong, Y.; Ma, H.; Li, J.; Wang, K.; and Tang, J. 2018. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *WSDM 2018*, 459–467. ACM.

Sankowski, P. 2005. Shortest paths in matrix multiplication time. In *European Symposium on Algorithms*, 770–778. Springer.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 815–823.

Song, J.; He, T.; Gao, L.; Xu, X.; Hanjalic, A.; and Shen, H. T. 2018. Binary generative adversarial networks for image retrieval. In *AAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018*, 394–401.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-scale information network embedding. In *WWW 2015*, 1067–1077.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Wang, J.; Zhang, T.; Sebe, N.; Shen, H. T.; et al. 2018. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* 40(4):769–790.

Wu, W.; Li, B.; Chen, L.; and Zhang, C. 2018. Efficient attributed network embedding via recursive randomized hashing. In *IJCAI*, volume 18, 2861–2867.

Xie, S.; Zheng, Z.; Chen, L.; and Chen, C. 2018. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning*, 5419–5428.

Yang, J., and Leskovec, J. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42(1):181–213.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. 2015. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yang, H.; Pan, S.; Zhang, P.; Chen, L.; Lian, D.; and Zhang, C. 2018. Binarized attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, 1476–1481. IEEE.

Zhang, P.; Zhang, W.; Li, W.; and Guo, M. 2014. Supervised hashing with latent factor models. In *SIGIR'14*, 173–182.

Zhang, H.; Shen, F.; Liu, W.; He, X.; Luan, H.; and Chua, T.-S. 2016. Discrete collaborative filtering. In *ACM SIGIR '14*, 325–334. ACM.

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*.