Triplet Enhanced AutoEncoder: Model-free Discriminative Network Embedding

Yao Yang, **Haoran Chen** and **Junming Shao***
Data Mining Lab, University of Electronic Science and Technology of China
{yaoyang, haorchen}@std.uestc.edu.cn, junmshao@uestc.edu.cn

Abstract

Deep autoencoder is widely used in dimensionality reduction because of the expressive power of the neural network. Therefore, it is naturally suitable for embedding tasks, which essentially compresses high-dimensional information into a low-dimensional latent space. In terms of network representation, methods based on autoencoder such as SDNE and DNGR have achieved comparable results with the state-of-arts. However, all of them do not leverage label information, which leads to the embeddings lack the characteristic of discrimination. In this paper, we present Triplet Enhanced AutoEncoder (TEA), a new deep network embedding approach from the perspective of metric learning. Equipped with the triplet-loss constraint, the proposed approach not only allows capturing the topological structure but also preserving the discriminative information. Moreover, unlike existing discriminative embedding techniques, TEA is independent of any specific classifier, we call it the model-free property. Extensive empirical results on three public datasets (i.e, Cora, Citeseer and BlogCatalog) show that TEA is stable and achieves state-of-the-art performance compared with both supervised and unsupervised network embedding approaches on various percentages of labeled data. The source code can be obtained from https://github.com/yybeta/TEA.

1 Introduction

Real-world networks often contain billions of nodes. It is intractable to perform complex mining tasks on these original networks such as node classification, link prediction, community detection [Shao *et al.*, 2015] and recommendation. Currently, one promising solution is network representation, targeting to embed networks into a low-dimensional space (i.e., learn a vector representation for each vertex, with the goal of reconstructing the network in the learned embedding space, while the information of the original network is retained to the maximum).

*Corresponding Author

During the past decade, many network embedding methods have been proposed (e.g., [Tang *et al.*, 2015; Perozzi *et al.*, 2014; Grover and Leskovec, 2016; Wang *et al.*, 2016; Cao *et al.*, 2016; Zhou *et al.*, 2017; Velickovic *et al.*, 2018]), which can be broadly categorized as shallow model or deep model. The commonly used techniques in the shallow model include matrix factorization and random walk. Factorization-based methods such as LINE [Tang *et al.*, 2015] and GraRep [Cao *et al.*, 2015] represent the target network as a data matrix and factorize this matrix to yield low-rank embeddings. Random walk based methods like node2vec [Grover and Leskovec, 2016], learn a vector embedding by exploiting random paths to preserve the local or global network structure. For all shallow models, the major limitation is that they fail to capture the complex nonlinear network structure.

As a result, many deep models have been proposed in recent years [Perozzi *et al.*, 2014; Wang *et al.*, 2016; Cao *et al.*, 2016; Velickovic *et al.*, 2018]. For example, thanks to the powerful ability to model non-linear complex relationship, deep autoencoders are popularized in network embedding such as SDNE [Wang *et al.*, 2016] and DNGR [Cao *et al.*, 2016], which have achieved state-of-the-art performance.

Recent studies on semi-supervised learning have demonstrated that leveraging labeling information does help network representation, especially for prediction-related tasks [Tu *et al.*, 2016; Li *et al.*, 2016]. However, current works mainly focus on training a specific classifier (e.g., SVM) with labeled information, such as MMDW [Tu *et al.*, 2016]. The classifier-dependent learning framework is not stable and only performs well in node classification task with SVM. If we use the resulting embedding for node classification or clustering with other learning models, its performance is usually not satisfactory (see Tables 5 and 6 for example).

Motivated by previous works, in this paper we introduce a new deep network embedding approach, called Triplet Enhanced AutoEncoder (TEA), aiming to preserve both nonlinear network structure and node discriminative information. To this end, the label information is reformatted as a set of triplet constraints from the perspective of metric learning, and then plugged into an autoencoder architecture to learn a deep network embedding. We evaluated the performance of TEA on both node classification and clustering tasks, and the results show that our proposed model outperforms other competitive baselines, which suggest it is beneficial to both super-

vised and unsupervised tasks.

To summarize, we make the following contributions.

- We employ a metric learning strategy to utilize label information for network embedding. As a semi-supervised model, TEA not only allows capturing non-linear network structure with deep autoencoder, but also preserving the node discriminative information by imposing the triplet constraints.
- The proposed model TEA is model-free, which indicates that it is independent of any specific classifier. We are also the first who "supervise" the unsupervised autoencoder by metric learning.
- We conduct extensive experiments on real-world networks, and results show the effectiveness of the proposed TEA model.

2 Related Work

2.1 Network Embedding

In recent years, a growing number of works are proposed to find a good network representation in a low-dimensional space. As aforementioned, one strategy is to extract latent embeddings by singular value decomposition or matrix factorization [Cao *et al.*, 2015; Ou *et al.*, 2016; Tu *et al.*, 2016; Wang *et al.*, 2017; Yang *et al.*, 2015; Zhang *et al.*, 2016]. This type of methods typically work on a data matrix $\boldsymbol{A}$ (e.g., adjacent matrix) that characterizes the topological properties of a given network, and then factorizes $\boldsymbol{A} \approx \boldsymbol{U^T V}$ or $\boldsymbol{A} \approx \boldsymbol{U^T U}$ into two low-dimensional embedding matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ (i.e., the new network embedding). Since the formation process of a network is complicated and highly nonlinear, thus a linear function like matrix factorization may not be adequate to map the original network to an embedding space [Cai *et al.*, 2018].

Inspired by the word2vec [Mikolov *et al.*, 2013], there are also many random-walk-based methods emerged [Perozzi *et al.*, 2014; Grover and Leskovec, 2016; Pan *et al.*, 2016; Zhou *et al.*, 2017], which propose some implicit reduction models by gathering random walk sequences of sampled nodes throughout a network. These methods work well in practice but struggle to explain what network properties should be kept in their objective functions. Deep autoencoders are also used to learn latent embedding of $\boldsymbol{A}$ [Cao *et al.*, 2016; Wang *et al.*, 2016] in an unsupervised way, where achieve non-linear mapping with various activation functions. For these works, different objective functions like Kullback–Leibler divergence [Tang *et al.*, 2015] or Huber loss [Wei *et al.*, 2017], are applied for network embedding. In addition, some methods are proposed to apply convolutional neural networks to make use of node attributes of networks [Kipf and Welling, 2017; Hamilton *et al.*, 2017; Velickovic *et al.*, 2018].

2.2 Metric Learning

Learning a good distance metric in feature space is crucial in real-world applications. Good distance metrics are important to many computer vision tasks, such as image classification and content-based image retrieval. Distance metric learning attempt to learn metrics that keep all the data points within the same classes close, while separating all the data points from different classes far apart. Prior works explored finding a mapping function ranging from a linear transform [Weinberger and Saul, 2009; Liao *et al.*, 2015; Liu *et al.*, 2017] to complex non-linear mappings usually represented by deep neural networks [Ding *et al.*, 2015; Cheng *et al.*, 2016].

3 Method

3.1 Problem Definition

Let us begin with formally defining the problem of network representation learning. Suppose we are given a network $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of all vertices and $E = \{e_{i,j}\}_{i,j=1}^{n}$ represents the set of edges. Each edge is associated with a weight $w_{i,j} \geq 0$, $w_{i,j} = 0$ if v_i and v_j is not linked. Otherwise, for unweighted graph $w_{i,j} = 1$ and for weighted graph $w_{i,j} > 0$. We let $\boldsymbol{x}_i = [w_{i,0}, ..., w_{i,n}]^T$ as the original representation of node i, let $Y \in \mathbb{R}^{n \times k}$ be a binary matrix that collects the label of all n nodes, where k is the number of label categories.

Network embedding aims to map the graph data into a low-dimensional latent space and information of the original topology should be preserved to the maximum, where each vertex is represented as a low-dimensional vector. Namely, we aim to find a map function f: $\mathbb{R}^{|V| \times 1} \rightarrow \mathbb{R}^{d \times 1}$, $d \ll |V|$. Then for each node i, the original representation vector $\boldsymbol{x}_i$ becomes $\boldsymbol{\Phi}_i$.

3.2 DNGR Revisited

To motivate our method, we first discuss the closest related work, which is a prior deep autoencoder-based model: DNGR.

DNGR consists of three components: random surfing, positive point-wise mutual information (PPMI) calculation and stacked denoising autoencoder. Random surfing is used on the input graph to generate a probabilistic co-occurrence matrix, analogous to using the original adjacency matrix as a similarity matrix to express topological information.

PPMI matrix is transformed from the probabilistic co-occurrence matrix and used to solve the problem in word embedding that frequent words with relatively little semantic value such as meaningless stop words, these words often have a disproportionate effect on the word representations. It was proved [Cao *et al.*, 2016] to have a better-tuned context weighting strategy compared to word2vec and Glove[Pennington *et al.*, 2014]. It ensures that the following autoencoder model can capture high-order proximity more properly.

The point-wise mutual information (PMI) is calculated as follow:

$$PMI_{w,c} = \log(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)}) \tag{1}$$

Where $|D| = \sum_w \sum_c \#(w,c)$. A common approach to improving performance is to assign each negative value to 0, detailed in [Levy and Goldberg, 2014], to form the PPMI matrix

$$PPMI_{w,c} = \max(PMI_{w,c}, 0) \tag{2}$$

With random surfing and PPMI, the adjacency matrix is transformed into a PPMI matrix, and is finally fed into an autoencoder to obtain the network embedding. For the encoder, it consists of multiple non-linear functions that map the input data to the representation space. Similarly, the decoder consists of multiple non-linear functions mapping the embedding in the representation space to the reconstruction space. Formally, given an input $\boldsymbol{x}_i$, the hidden representations for each layer are shown as follows.

$$\boldsymbol{y}_i^{(1)} = \sigma(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{b}^{(1)}) \tag{3}$$

$$\boldsymbol{y}_i^{(k)} = \sigma(\boldsymbol{W}^{(k)}\boldsymbol{y}_i^{(k-1)} + \boldsymbol{b}^{(k)}), k = 2, ..., K \tag{4}$$

Here we use the LeakyRelu activation function as σ in our experiments. After obtaining $\boldsymbol{\Phi}_i = \boldsymbol{y}_i^{(K)}$, we can get the output $\hat{\boldsymbol{x}}_i$ by decoding $\boldsymbol{\Phi}_i$, i.e., reverse the calculation process of the encoder. The goal of the autoencoder is to minimize the reconstruction error between the output and the input. The loss function is given as follow.

$$\mathcal{L}_r = \sum_{i=1}^{n} ||\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i||_2^2 \tag{5}$$

In order to enhance the robustness of DNN, each input sample $\boldsymbol{x}$ in Eq. (3) is corrupted randomly by assigning some entries in the vector to 0 with a certain probability.

The stacked denoising autoencoders aids robustness of the model in the presence of noise. It also allows capturing underlying structure, which is essential to many mining tasks such as link prediction and node classification. However, as aforementioned, existing autoencoder-based network embedding fails to preserve the node discriminative information. In the following, we will introduce triplet-loss, a metric learning perspective to integrate label information in our proposed model.

3.3 Metric Learning with Triplet-loss

Metric learning aims to find a mapping function (called metric) $D : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}_{\geq 0}$ over a vector space which used to metric the distance of two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{X}$. A popular metric is Mahalanobis distance that defined as follow.

$$D_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{6}$$

Where $\boldsymbol{M}$ is a positive semidefinite matrix. Originally, this term was used to describe the quadratic forms in Gaussian distributions, the matrix $\boldsymbol{M}$ plays the role of the inverse covariance matrix. For clarity we next use the shortcut notation $D_{i,j} = D_M(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Our idea is to keep nodes with the same class to be close to each other in certain relation space, and keep vertices with different classes far apart. We define the following margin-based loss function as the objective function.

$$\mathcal{L}_{tri}(\theta) = \sum_{\substack{a,p,n \\ y_a = y_p \neq y_n}} [m + D_{a,p} - D_{a,n}]_+ \tag{7}$$

Given an anchor point $\boldsymbol{x}_a$, the projection of a positive point $\boldsymbol{x}_p$ belonging to the same class y_a is closer to the anchor's projection than that of a negative point belonging to another class y_n, by at least a margin m.

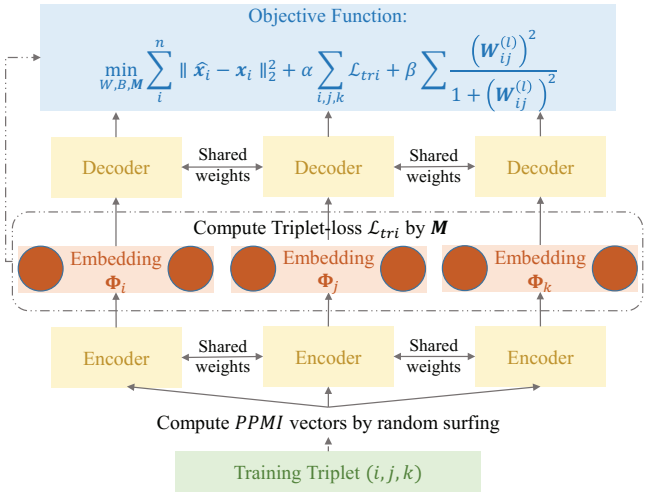


Figure 1: The framework of the Triplet Enhanced Autoencoder (TEA).

3.4 TEA: Triplet Enhanced Autoencoder

Building upon the deep autoencoder and triplet-loss [Schroff *et al.*, 2015], finally we have our Triplet Enhanced Autoencoder in Fig. 1. Here the model is trained using each triplet as a training instance. Given (i, j, k), first our model compute the PPMI vectors of them, then feed the encoder and output the embeddings $(\boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j, \boldsymbol{\Phi}_k)$. Second, we calculate the triplet-loss of the embeddings while feeding them into the following decoder. The output of the decoder as well as the triplet-loss is used to compute an objective function explained below. Three encoders and decoders share the same weights respectively. The embeddings and the network structure are updated by gradients propagated back from the output layers.

In detail, we simultaneously optimize two objectives of Eq. (5) and Eq. (7). Let $\mathcal{L}_{tea} = \mathcal{L}_r + \alpha \sum \mathcal{L}_{tri}$, where L_r is the reconstruction loss of the autoencoder and $\sum \mathcal{L}_{tri}$ is the triplet-loss. α is a key parameter that balances the weights of the two objectives. The network embedding is finally transformed into a joint optimization problem as follow.

$$\min_{W,B,\boldsymbol{M}} \mathcal{L}_{tea} = \min \mathcal{L}_r + \alpha \sum_{\substack{a,p,n \\ y_a = y_p \neq y_n}} [1 + D_{a,p} - D_{a,n}]_+ \tag{8}$$

Specifically, we amend $\mathcal{L}_r$ with a tuned regularizer as follow:

$$\mathcal{L}_r = \sum_{i=1}^{n} ||\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i||_2^2 + \beta \sum \frac{(\boldsymbol{W}_{jk}^{(l)})^2}{1 + (\boldsymbol{W}_{jk}^{(l)})^2}, \tag{9}$$

where the term $\sum \frac{(\boldsymbol{W}_{jk}^{(l)})^2}{1+(\boldsymbol{W}_{jk}^{(l)})^2}$ is called weight-elimination $\mathcal{L}2$-norm (Ridge) regularizer. The advantage of this regularizer is making the weight matrix to be sparse while reducing overfitting, and it solves the problem that $\mathcal{L}1$-norm (Lasso) regularizer cannot be differentiated [Weigend *et al.*, 1991].

As for the distance metric function in Eq. (8), we use the Mahalanobis distance function which described in Eq.(6) to measure the distance between anchor's embedding $\boldsymbol{\Phi}_i$ and another's embedding $\boldsymbol{\Phi}_j$.

Since the whole objective (8) is differentiable, we are allowed to apply mini-batch stochastic gradient descent (SGD) to optimize each $\boldsymbol{W}, \boldsymbol{b}$ and $\boldsymbol{M}$ by propagating the gradients top-down from the output layers.

3.5 Time Complexity

The most time-consuming part of our method is the computation of triplet-loss and we calculate it every batch. It is not difficult to see that the training complexity of our model is $\mathcal{O}[(B + D)Bne]$, where B is the batch-size in our training process, D is the maximum dimension of the hidden layers, n is the number of vertices and e is the number of epochs we choose to train. Parameter D is usually related to the dimension of embedding vectors but not related to the number of vertices, e is also independent with n, B is usually tuned according to the number of node categories. Thus, the overall complexity is linear to the number of vertices in the network.

4 Experiment

In this section, we assess the effectiveness of our TEA model on real-world networks. We conduct node classification as well as clustering tasks and make comparisons with baseline algorithms. Besides, we also visualize the learned representation to demonstrate its discriminative ability.

4.1 Datasets

We use the following three typical datasets for our experiments.

- **Cora**[1]: This is a citation network of academic papers. The vertexes are academic papers and the directed edges are the citation relationship between papers. It contains 2,708 machine learning papers which are categorized into 7 classes, and there are 5,429 citation links among these papers.
- **Citeseer**[2]: This is another research paper set. These papers are from 6 classes. Similar to Cora, the links are citation relationships between the documents. It contains 3,312 publications and 4,732 connections between them. Different from Cora, it contains several self-loops.
- **BlogCatalog**[3]: This is a network of social relationships between users on the BlogCatalog website. The labels of this graph are the topics specified by the uploading users. After filtering out nodes with multi-label, it contains 7,288 nodes, 131,011 edges, and 37 labels.

4.2 Baseline Methods and Evaluation Metrics

We use the following four methods as the baselines to validate the performance of our approach. The first three are learned in unsupervised ways, and the last one utilizes existing label information like our approach.

- **GraRep**: This multi-scale method generates vertex representations by explicitly computing successive powers of the random walk transition matrix, and SVD is applied to obtain a low-dimensional representation of nodes.
- **node2vec**: It adopts random walk and skip-gram model to generate network representations. By introducing the return parameter p and the in-out parameter q, node2vec combines DFS-like and BFS-like neighborhood exploration.
- **DNGR**: It firstly calculates the PPMI matrix, and then learns the representations through stacked denoising autoencoder. It is an exact computation similar to our approach in spirit, however, DNGR doesn't utilize additional labeling information.
- **MMDW**: It is a semi-supervised method which adopts the matrix factorization form of DeepWalk, to preserve the network structure and discriminative information by equipping with a support vector machine model.

In our experiment, we performed the tasks of node classification, clustering and visualization. For the node classification task, we adopted micro-F1 and macro-F1. To further demonstrate the advantage of metric learning for leveraging the label information, we evaluated the embedding results with both logistic regression and support vector machine (both implemented by LibLinear[4]). We applied K-means to the learned representations of nodes and adopted normalized mutual information (NMI) to assess the quality of the node clustering. Due to the sensitivity of K-means on the initial values, we repeated the clustering ten times, and the average results were reported here.

4.3 Parameter Settings

We propose a multi-layer deep structure in this paper and the number of layers varies with different datasets. The dimension of each layer is listed in Table 1. The hyper-parameters of α and β are tuned by using grid search on the validation set, the sensitivity is further evaluated in Section 4.5. The models were trained for 20 epochs with Adam optimizer.

We set the representation dimension to 100 for baseline algorithms, same as our model. The parameters in baselines are tuned. For GraRep, we set the maximum matrix transition step $K = 5$ for BlogCatalog, $K = 3$ for Cora and Citeseer. For node2vec, we set the window size as 10, the walk length as 80, the walk per vertex as 20, both in-out and return hyper-parameters are set to 0.25. For DNGR, we set the neural network structures the same as TEA. For MMDW, we set the gradient balance parameter $\eta = 10^{-2}$.

Dataset	#nodes in each layer
Cora	[2708-520-100]
Citeseer	[3312-570-100]
BlogCatalog	[7288-1800-450-100]

Table 1: Neural Network Structures.

[1] http://www.cs.umd.edu/~sen/lbc-proj/data/cora.tgz

[2] http://www.cs.umd.edu/~sen/lbc-proj/data/citeseer.tgz

[3] http://socialcomputing.asu.edu/datasets/BlogCatalog3

[4] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Algorithm	Mac1	Mic1	Mac2	Mic2	NMI
GraRep	77.52	78.95	77.97	79.06	42.98
node2vec	80.72	81.87	79.66	81.02	46.32
DNGR	81.13	82.09	77.91	79.32	34.20
TEA(5%)	82.13	83.06	82.22	83.27	47.67
TEA(10%)	**82.57**	**83.15**	**82.47**	**83.29**	**51.68**

Table 2: The clustering and classification performance(%) of different algorithms on the Cora data. Limited by layout, we use **Mac1** and **Mic1** to represent the metric Macro-F1 and Micro-F1 on the model Logistic-Regression respectively, **Mac2** and **Mic2** to represent the metric Macro-F1 and Micro-F1 on the model SVM respectively. The same below.

Algorithm	Mac1	Mic1	Mac2	Mic2	NMI
GraRep	43.12	52.20	44.94	53.62	16.59
node2vec	48.25	54.74	47.16	54.20	23.18
DNGR	48.93	54.20	46.98	53.14	17.41
TEA(5%)	56.89	61.79	56.73	61.95	24.70
TEA(10%)	**57.29**	**62.50**	**56.95**	**62.77**	**26.28**

Table 3: The clustering and classification performance(%) of different algorithms on the Citeseer data.

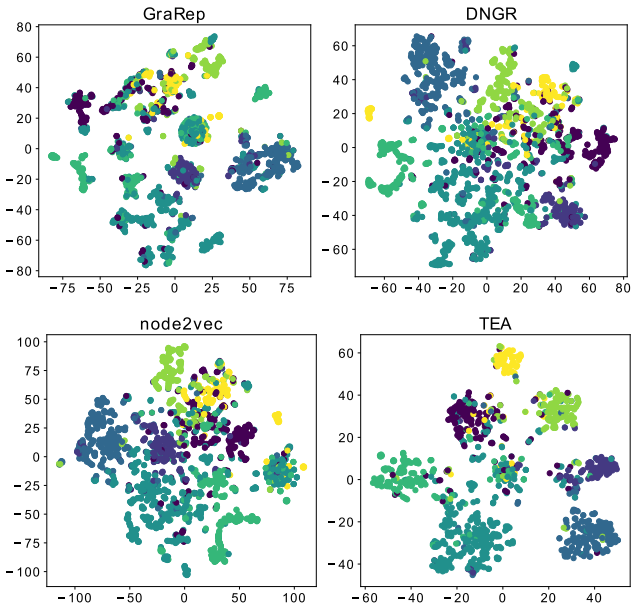


Figure 2: t-SNE 2D representations on Cora, where different colors indicate different classes. TEA has the most discriminative representation.

4.4 Results and Analysis

Here we first show our TEA can achieve considerable improvement than the unsupervised baselines with only a little label information. Afterward, we will elaborate that our TEA is model-free and outperforms other semi-supervised methods like MMDW.

Compared with Unsupervised Methods

We separate the embedding process and the evaluation process so that all the embedding output can be evaluated by more than one model. Here four-fold cross-validation is applied.

Tables 2,3 and 4 show the classification and clustering performance on different datasets. In these tables, all algorithms make full use of the topology data. TEA(5%) and TEA(10%) mean the percentages of the additional label information we use in TEA during the embedding process, and the rest of the nodes are used for evaluation. From these tables, we can observe that, compared to baselines, TEA needs only 5% labeled data to achieve an obvious improvement, especially for the SVM model. These results show that labeled information is really helpful for classification tasks.

Algorithm	Mac1	Mic1	Mac2	Mic2	NMI
GraRep	17.94	36.17	18.00	36.96	19.04
node2vec	**24.14**	38.83	23.13	38.92	20.38
DNGR	23.85	37.71	19.20	31.27	19.14
TEA(5%)	23.49	39.96	23.25	40.52	18.13
TEA(10%)	23.82	**40.69**	**23.57**	**40.70**	**20.69**

Table 4: The clustering and classification performance(%) of different algorithms on the BlogCatalog data.

To verify whether the learned representation is discriminative, we show the 2D representations of vertices using the t-SNE visualization tool in Fig. 2. In this figure, each dot represents a vertex and each color represents a category. From Fig. 2, we observe that TEA learned a better clustering and separation of the vertices. In contrast, the representations learned by node2vec tend to mix together, and GraRep and DNGR are even worse. A well-separated representation is more discriminative and easier to apply downstream network mining tasks.

Compared with MMDW

Here, in contrast to the unsupervised setting, we randomly sample a portion of vertices to adopt their label information for embedding, and use four-fold cross-validation on the rest vertices to evaluate the embedding results. We increase the labeled ratio from 20% to 80%, to compare the performance of different algorithms.

Tables 5 and 6 show the classification and clustering performance on different datasets. LR means the embedding results are evaluated under the model logistic regression, and SVM means the embedding results are evaluated with SVM.

From these tables, we can observe that TEA outperforms MMDW no matter how much labeled information is available. Although the performance of MMDW is enhanced with label information under the SVM model, it gives poor results when a logistic regression model is applied on the learned embeddings (see Tables 5 and 6). Thus it is not a good representation. These results further demonstrate that the metric learning perspective provides a good way to leverage available labeling information, and do help network embedding. More importantly, the resulting embedding is stable and independent of any specific model.

Metric	Algorithm	20%	30%	40%	50%	60%	70%	80%
Micro-F1(LR)	MMDW	32.03	30.73	30.01	30.80	31.13	31.74	33.72
	TEA	**83.30**	**83.58**	**84.74**	**83.85**	**84.96**	**84.37**	**85.90**
Macro-F1(LR)	MMDW	10.15	7.59	7.46	7.19	6.92	6.88	7.20
	TEA	**82.83**	**82.46**	**83.97**	**83.24**	**84.54**	**83.82**	**84.85**
Micro-F1(SVM)	MMDW	81.12	82.20	82.72	83.33	83.57	83.71	84.22
	TEA	**83.26**	**83.69**	**85.17**	**84.57**	**84.82**	**84.99**	**86.37**
Macro-F1(SVM)	MMDW	79.52	80.85	81.53	82.03	82.94	82.29	83.20
	TEA	**82.57**	**82.83**	**84.37**	**84.21**	**84.29**	**84.53**	**85.26**
NMI	MMDW	6.88	7.58	6.35	6.58	8.17	11.31	9.17
	TEA	**53.22**	**57.12**	**62.43**	**60.42**	**62.85**	**63.28**	**66.84**

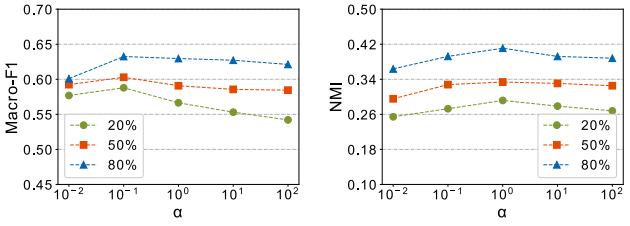
Table 5: The performance(%) of TEA and MMDW on the Cora data with different labeled ratios, ranging from 20% to 80%.

Metric	Algorithm	20%	30%	40%	50%	60%	70%	80%
Micro-F1(LR)	MMDW	34.17	36.55	39.49	40.67	42.73	30.02	28.67
	TEA	**63.17**	**63.52**	**64.43**	**64.79**	**66.50**	**67.09**	**68.19**
Macro-F1(LR)	MMDW	21.56	23.77	28.99	31.89	31.08	17.59	13.84
	TEA	**58.36**	**59.27**	**60.31**	**60.97**	**61.81**	**62.73**	**63.58**
Micro-F1(SVM)	MMDW	57.70	60.66	61.17	62.51	63.62	64.00	62.19
	TEA	**62.91**	**64.16**	**64.80**	**65.28**	**66.75**	**67.29**	**68.34**
Macro-F1(SVM)	MMDW	50.42	53.65	55.20	57.43	58.35	58.55	57.36
	TEA	**57.91**	**59.68**	**60.34**	**60.94**	**62.23**	**62.20**	**63.44**
NMI	MMDW	6.04	11.56	10.88	15.58	10.88	12.36	10.77
	TEA	**29.58**	**30.74**	**31.14**	**33.52**	**35.15**	**37.48**	**41.10**

Table 6: The performance(%) of TEA and MMDW on the Citeseer data with different labeled ratios, ranging from 20% to 80%.

4.5 Parameter Sensitivity

There is a key parameter involved in our objective function: α. As aforementioned, α is used to adjust the weights of two objectives (cf. Section 3.4). To test its sensitivity, we conduct experiments on both classification and clustering tasks. The performance of TEA against different values of α is given in Fig. 3(a) and Fig. 3(b) in terms of Macro-F1 and NMI on the Citeseer data, respectively. From the figure, we can observe that TEA has a fairly stable performance when α ranges from 10^{-2} to 10^{2}. Depending on the specific task is classification or clustering, it gains the best performance at different locations, mainly range from 10^{-1} to 10^{0}. In general, from Fig. 3, we can see that TEA achieves good performance in a wide range of parameter settings.



(a) Macro-F1 for Classification (b) NMI for Clustering

Figure 3: The parameter sensitivity analysis of TEA on both clustering and classification tasks.

5 Conclusion

In this paper, we propose a triplet-loss enhanced deep autoencoder, namely TEA, to perform network embedding via metric learning. By leveraging the labeling information with the triplet-loss, TEA allows learning the vertex representations which preserve both their network structure and node discriminative information with the model-free property. Experiments on real-world datasets show the stability of TEA and achieve state-of-the-art performance compared with both supervised and unsupervised network embedding approaches.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61403062, 61433014, 41601025), Science-Technology Foundation for Young Scientist of Sichuan Province (2016JQ0007), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (161062) and National key research and development program (2016YFB0502300).

References

[Cai *et al.*, 2018] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *TKDE*, 30(9):1616–1637, 2018.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.

[Cao *et al.*, 2016] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016.

[Cheng *et al.*, 2016] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, pages 1335–1344, 2016.

[Ding *et al.*, 2015] Shengyong Ding, Liang Lin, Guangrun Wang, and Hongyang Chao. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 48(10):2993–3003, 2015.

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

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034, 2017.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NeurIPS*, pages 2177–2185, 2014.

[Li *et al.*, 2016] Juzheng Li, Jun Zhu, and Bo Zhang. Discriminative deep random walk for network classification. In *ACL*, pages 1004–1013, 2016.

[Liao *et al.*, 2015] Shengcai Liao, Yang Hu, Xiangyu Zhu, and Stan Z Li. Person re-identification by local maximal occurrence representation and metric learning. In *CVPR*, pages 2197–2206, 2015.

[Liu *et al.*, 2017] Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. End-to-end comparative attention networks for person re-identification. *TIP*, 26(7):3492–3506, 2017.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[Ou *et al.*, 2016] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114. ACM, 2016.

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

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[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.

[Schroff *et al.*, 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[Shao *et al.*, 2015] Junming Shao, Zhichao Han, Qinli Yang, and Tao Zhou. Community detection based on distance dynamics. In *KDD*, pages 1075–1084. ACM, 2015.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[Tu *et al.*, 2016] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, Maosong Sun, et al. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, pages 3889–3895, 2016.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, pages 1225–1234. ACM, 2016.

[Wang *et al.*, 2017] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, pages 203–209, 2017.

[Wei *et al.*, 2017] Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. Cross view link prediction by learning noise-resilient representation consensus. In *WWW*, pages 1611–1619, 2017.

[Weigend *et al.*, 1991] Andreas S Weigend, David E Rumelhart, and Bernardo A Huberman. Generalization by weight-elimination with application to forecasting. In *NeurIPS*, pages 875–882, 1991.

[Weinberger and Saul, 2009] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb):207–244, 2009.

[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.

[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.

[Zhou *et al.*, 2017] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *AAAI*, pages 2942–2948, 2017.