ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/325420173

ANE: Network Embedding via Adversarial Autoencoders

Conference Paper · January 2018

DOI: 10.1109/BigComp.2018.00019

CITATIONS
5

READS
166

4 authors, including:

Binbin Hu
Beijing University of Posts and Telecommunications
20 PUBLICATIONS **309** CITATIONS
SEE PROFILE

Chuan Shi
Shanghai Institutes for Biological Sciences
101 PUBLICATIONS **1,258** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Regulated Transformer Rectifier Unit Design for Boeing-787 View project

All content following this page was uploaded by Binbin Hu on 15 April 2019.

The user has requested enhancement of the downloaded file.

ANE: Network Embedding via Adversarial Autoencoders

Yang Xiao* , Ding Xiao , Binbin Hu , Chuan Shi
Beijing Key Lab of Intelligent Telecommunications Software and Multimedia
Beijing University of Posts and Telecommunications,Beijing,China 100876
Email: ynynny@sina.com, dxiao@bupt.edu.cn, hubinbin@bupt.edu.cn, shichuan@bupt.edu.cn

***Abstract*—Network embedding is an important method to learn low-dimensional representations of vertexes in network, whose goal is to capture and preserve the highly non-linear network structures. Here, we propose an Adversarial autoencoders based Network Embedding method (ANE for short), which utilizes the rencently proposed adversarial autoencoders to perform variational inference by matching the aggregated posterior of low-dimensional representations of vertexes with an arbitraray prior distribution. This framework introduces adversarial regularization to autoencoders. And it is able to attaches the latent representations of similar vertexes to each other and thus prevents the manifold fracturing problem that is typically encountered in the embeddings learnt by the autoencoders. Experiments demonstrate the effictiveness of ANE on link prediction and multi-label classification on three real-world information networks.**

***Keywords*—network embedding, adversarial autoencoders, latent representations.**

I. INTRODUCTION

Nowadays, networks are ubiquitous and a lot of applications are based on the information from networks. For example, Online shopping often needs to find similar users with common interests in the social network. Therefore, one of the important problems is how to learn latent network representations [1]. The most straightforward way is to embed network into a low-dimensional latent space, i.e. learn low-dimensional representations for each vertexes of network, which can demonstrate the structures of network and reconstruct network. Such a low-dimensional embedding is useful in a variety of applications, such as node classification [2], link prediction [3] and recommendation [4]. However the latent structures of the network is highly non-linear [5]. Therefore, one of the most important problems is that capture the underlying high non-linearity structures and preserveing both the local and global network structures. And many real-world networks are frequently so sparse that only very limited links can be used [6].

In the last few years, various network embedding methods have been proposed, such as IsoMAP [7], Laplacian Eigenmaps(LE) [8], DeepWalk [6], Node2vec [9], LINE [10] and DNGR [11]. However, It is difficult for shallow models to capture the highly non-linear network structure, whose representation capacity is insufficient. Meanwhile, some models are based on the neural language model, which does not utilize the adjacency matrix of the network directly. And the non-regularized autoencoders some method adopted typically fracture the manifold into many different domains. [12].

On the other line, adversarial training has been proved to be a great feature learning tool. In this paper we introduce adversarial training into learning network-embedding representations by training Adversarial Autoencoders (AAE) [13]. This is motivated by the recent success of GANs [14], which has been proved to have a powerful capacity to learn reusable feature representations from complex data and has achieved great success in image and text processing [15]–[17]. Adversarial Autoencoders is trained with dual objectives, a traditional reconstruction error criterion, and an adversarial training criterion that matches the aggregated posterior distribution of the latent representation of the autoencoder to an arbitrary prior distribution, which has a strong connection to VAE [18] traing.

We utilize a multi-layer autoencoders which consists of multiple non-liner functions to capture the highly non-linear network structures. The model must be able to optimize an objective which preserves both the local and global network structures. Therefore, we exploit the first-order and second-order [10] into training process simultaneously. The first-order proximity represents the similarity between the vertexes which have the observed links in the network. However, observing the first-order proximity of the network is not sufficient for preserveing the global network structures. As a complement, the model can capture the global network structures with the second-order proximity, which is determined through the shared neighborhood structures. To preserve both local and global structures in ANE, the autoencoder component of the model is used to capture the global structures by reconstructing the second-order with a traditional reconstruction error criterion. Meanwhile, the second component, which benefits from the idea of Laplacian Eigenmaps, preserves the local network structures by giving a punishment when the distance between similar vertexes is far off in the latent representation space.

However, the original autoencoders typically encounters the manifold fracturing problem. And adversarial training can alleviate the problem [13]. The adversarial training component can be viewed as regularization component which discriminatively predicts whether a sample arises from the low-dimensional representations of the network or from a sampled distribution. As a consequence, the model is able to learn representations which capture network structures well.

To demonstrate the validity of the ANE, we conduct experiments on three real-world network datasets which are from real-word applications. The effectiveness of the learned low-dimensional embeddings is evaluated within multiple tasks, including network reconstruction, link prediction and multi-label classification. The results demonstrate the representation learned by our methed can generally outperform baselines in these tasks. It shows effectiveness of the method in capturing the network sttuctures.

Organization. The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 formally defines the problem. Section 4 introduces the ANE model in details. Section 5 presents the experimental results. Finally we conclude in Section 6.

II. RELATED WORK

Our work focuses on network embedding, which aims at mapping the network or vertex data to a low-dimensional latent space where each vertex is represented as a low dimensional real vector. Several network embedding methods including DeepWalk [6], LINE [10], Node2vec [9], Deep Graph Kernels [19] and DDRW [20] have been proposed. These models are based on the neural language model. Several network embedding models are based on other neural network model. For example, DNGR [11] is based on a stacked denoising auto-encoder, and [21] adopts the convolutional neural network to learn the network feature representations. Unlike all the works described above, in this paper, we introduce the adversarial training to learn the network embedding.

The GAN is inspired by the Nash equilibrium in game theory. The learning process becomes a procedure of competition between generation model (G) and discriminant model (D) to directly shape the output distribution of the network via back-propagation. Compared with VAE, if the discriminator network is a perfect fitting, then the generator will reconstruct training distribution perfectly, and VAE has a certain bias. The GAN provides an attractive alternative to maximum likelihood techniques and has recently achieved great success on image and text processing, especially image generation. Several recent works including GAN [14], CGAN [15], LAPGAN [16] and DCGAN [17] have been proved the adversarial training has powerful ability on representation learning. Adversarial autoencoder (AAE) [13] is a variation of generative adversarial network. And AAE impose a prior distribution on the hidden code vector of the autoencoder by matching the aggregated posterior of the hidden code vector with the prior distribution, which combines the advantages of GAN and VAE and is able to alleviate the manifold fracturing problem. Therefore, we introduce AAE to perform network embedding and impose first-order and second proximity into AAE to preserve the structures of network better.

III. PROBLEM DEFINITION

In this section, we define the problem of Netwoek Embedding via Adversarial Autoencoders.

Definition 1: (Network)A network is defined as $G = (V, E)$, where V represents the set of vertexes and E is the set of edges between the vertexes. Each edge $\mathrm{e} \in E$ is an ordered pair $e = (u, v)$ and associated with weight $w_{u,v} > 0$, which indicates the strength of the relation. If G is unweighted network, $w_{u,v} = 1$. If G is weighted network, $w_{u,v} > 0$. Otherwise, $w_{u,v} = 0$, if there is no link between u and v.

In this paper, we only consider undirected networks. In order to embed a network into a low-dimensional latent space, in which each vertex is "encode" as a low-dimensional vector. The embedding must preserved the network structures. The first is capturing the network local structures, which are characterized by first-order proximity between vertexes.

Definition 2: (First-order Proximity)The first-order proximity is the pairwise proximity between two vertexes and represents network local structures. For each pair of vertexes linked by an edge, the $w_{u,v} > 0$, which is on behalf of the first-order proximity betweeen the two vertexes. Otherwise, the first-order proximity is 0.

The first-order proximity implies the similarity of two vertexes in a network if there exists an edge between the two vertexes. However, the links in real world network is always so sparse that measuring the similarity between the vertexes by first-order proximity is difficult without observed link. Therefore, we use the second-order proximity as complementary to capture the network global structures.

Definition 3: (Second-order Proximity)The second-order proximity is the similarity of the pair's neighborhood structures between two vertexes. Let $p_u = \left(w_{u,1}, \cdots w_{u,|V|}\right)$ denote the first-order proximity of vertex u with all the other vertexes in network. Consequently, the second-proximity between u and v is determined by the similarity between p_u and p_v.

Naturally, if a pair of vertexes have more common neighbors, there exist more similarities between them. And the second-order proximity is 0, if no vertex links to both u and v.

We define the network embedding with first-order and second-order proximity as follows.

Definition 4: (Network Embedding)Given a network $G = (V, E)$, network embedding aims at learning a function $f : V \rightarrow R^d$, which maps each vertex into a low-dimensional space R^d, where $d \ll |V|$. And the latent representation preserves the both first-order and second-order proximity between vertexes.

IV. ANE:NETWOEK EMBEDDING VIA ADVERSARIAL AUTOENCODERS

In this section, we first introduce the framework of ANE. Then we introduce the loss functions. At last, we give the alorithm of ANE.

A. Framework

In this paper, we propose the ANE to perform the network embedding. The framework of ANE is shown in Figure 1.

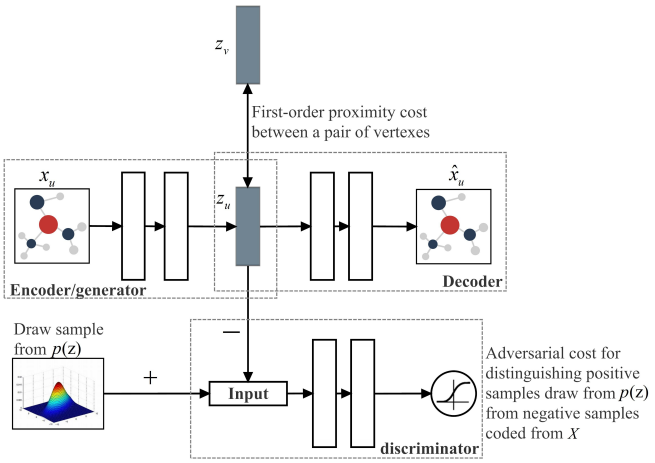


Fig. 1. The framework of ANE

To capture the highly non-linear structures of network, we use a deep model AAE with non-linear activation functiontons map the input data to a non-linear low-dimensional latent space to preserve the network structures.

The middle row is a standard autoencoder consisting of an encoder network and a decoder network. The encoder maps the input data x into a latent code z, then the decoder reconstructs the input data as $\hat{x}$. Accordingly, this component is able to capture the second-order proximity of each vertex by reconstructing its neighborhood structure, which can obtain the neighborhood of vertexes and network global structure indirectly.

The top row diagrams a component used to capture the first-order proximity of a pair of vertexes by measuring the similarities of vertexes in the low-dimensional latent space. Hence the latent representation can obtain the pairwise similarities, i.e. the local structures. By exploiting the first-order and second-order proximity simultaneously, the model can preserve both local and global network structures.

The bottom row is a discriminator network trained to discriminatively predict whether a sameple arises from the latent code of encoder or from a sampled distribution. As [13] proved, such a discriminative procedure matches the aggregated posterior distribution of the latent representation to the prior distribution, which can refine the latent representation of vertexes. As a result, ANE is able to learn a representation of network.

B. Loss Functions

In this part, we introduce the loss functions of ANE. And some of the terms and notations is defined in Table I. First, we describe the autoencoder component to preserve the second-oreder proximity.

The second-order proximity indicates the similarity of neighborhood structures between two vertexes, which means the more common neighborhood shared by a pair of vertexes, the more similar they are. Therefore, the model needs to model the all neighborhoods of each vertex. Given a network, the information of the network structures including local and global structures can be described by the adjacency matrix P. The instances $\{p_1, \cdots p_n\}$ of the P indicate the neighborhood structures of each vertex, where n is the number of vertexes. For each $p_u = \{p_{u,v}\}_{v=1}^{n}$, $p_{u,v}$ describes the first-order proximity between vertex u and v. Put P as the input, the model is able to preserve the second-order proximity.

TABLE I
TERMS AND NOTATIONS

Symbol	Definition
z	latent representation/code
$p(z)$	prior distribution imposed on z
$q(z)$	aggregated posterior distribution of z
n	number of vertexes
$P = \{p_1, \cdots p_n\}$	the adjacency matrix of the network
$X = \{x_u\}_{u=1}^{n}, \hat{X} = \{\hat{x}_u\}_{u=1}^{n}$	the input data and reconstructed data
θ	the overall parameters

The autoencoder component tries to capture the second-proximity by reconstructing the input. In detail, The encoder, which is made of multiple non-linear activation functions, maps the input data into the latent representation space. The decoder is similar to the encoder, but its process is reverse, i.e. reconstructing latent representation to reconstruction input space. Let the adjacency matrix P as the input X to encoder network, i.e. $x_u = p_u$, encoder will map instance x_u to latent representation z_u. And then decoder network will reconstruct z_u to output $\hat{x}_u$. As previously mentioned, the p_u describes the neighborhood structures of vertex u. Then the reconstruction process will capture the second-order proximity, where the similarity of latent codes reflects the similarity of neighborhood structures between vertexes.

However, there are only a small number of links in real-world network, which leads to the sparsity of network. As a result, the number of zero elements in the adjacency matrix P is much more than the elements of non-zero, which would debase the effect of the reconstruction. The autoencoder component would be inclined to reconstruct more zero elements to output $\hat{X}$. Therefore, we use the weighted Binary Cross Entropy as loss function to impose more penalty to the reconstruction error of the non-zero elements than zero elements. The objective function of each instance x_u is shown as follows:

$$L_u = -1/n \sum_{v}^{n} k_{u,v}(x_{u,v} log(\hat{x}_{u,v}) + (1 - x_{u,v}) log(1 - \hat{x}_{u,v})) \tag{1}$$

where $k_{u,v}$ means the weight coefficient of penalty imposed to the each elements. If $p_{u,v} = 0$, $k_{u,v} = 1$, else $k_{u,v} > 1$. Then the second-proximity objective function can be defined as follows:

$$L_{2nd} = \sum_{u}^{n} L_u \tag{2}$$

With this objective function, the latent representation can capture the network global structures. That is, the autoencoder component is able to map a pair of vertexes with high second-proximity near in the latent space.

However, it is insufficient to embed the network by reconstructing the second-order proximity only, which ignores the useful information of the network local structures. The model should be able to get the first-order proximity. Intuitively, a pair of vertexes linked by a certain edge has higher similarity. Thus the similarity between the two vertexes in latent representation, which denotes the first-order proximity, is constrained by the information of edge strongly. Therefore we impose a component to take advantage of the information, which borrows the idea from Laplacian Eigenmaps. The objective function is shown as below:

$$L_{1st} = \sum_{u,v=1}^{n} p_{u,v} \left\| z_u - z_v \right\|_2^2 \tag{3}$$

The objective function imposes a penalty on the pair of vertexes which have high similarity but maped far away in the latent representation space. In consequence, the latent representation can capture the network local structures.

To embed the network by capturing both first-order and second order proximity, we combine the proximity jointly by training the objective function 2 and 3 simultaneously as follow:

$$L_{ae} = L_{1st} + \alpha L_{2nd} \tag{4}$$

The standard autoencoder is a powerful tool which provides multiple non-linear function to learn a embedding of data. But non-regularized autoencoders "fracture" the manifold into many different domains which result in very different codes for similar inputs [12]. To address this problem, adversarial training is introduced into the autoencoder. As [13] proved, AAE is able to match the aggregated posterior of the latent representation with the prior distribution and impose a adversarial regularization on autoencoder component, which attaches the embedding of similar vertexes to each other and thus prevents the manifold fracturing problem. In order to do so, a discriminator network(D) is attached on the model, and the encoder network is also regarded as generator(G). Therefore, D and G play the following two-player minimax game with the value function $V(G, D)$:

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{p(z)}[logD(z)] + \mathbb{E}_{q(z)}[log(1 - D(z))] \tag{5}$$

Model training is mainly divided into two phases. In the reconstruction phase, the autoencoder updates the encoder and the decoder to minimize the L_{ae}. In the regularization phase, the adversarial network updates its discriminative network first and then updates generator network.

C. The ANE Alorithm

Now we give the algorithm of ANE. In order to encourage convergence of the minimax game, we use batchnorm in the adversarial component [22], which helps deal with training problems that arise due to poor initialization and helps gradient flow in deep models. And we use the Dropout in training, which has been proven to be an effective technique for regularization and preventing the co-adaptation of neurons [23]. Meanwhile, we use the Adagrad [24] as optimizer. The algorithm is presented in Alg. 1.

Algorithm 1: Algorithm for the AAE

initialize θ_{enc} for encoder(Enc), θ_{dec} for decoder(Dec) and θ_{dis} for discriminator(Dis).

repeat

`/* Recontruction */`

$X \leftarrow$ random mini-batch from dataset

$Z \leftarrow Enc(X)$

$L_{1st} \leftarrow \sum_{u,v=1}^{n} p_{u,v} \left\| z_u - z_v \right\|_2^2$ `// 1st-order Loss`

$\hat{X} \leftarrow Dec(Z)$

$L_{2nd} \leftarrow \sum_{u}^{n} L_u$, `// 2nd-order Loss`

$L_{ae} \leftarrow L_{1st} + \alpha L_{2nd}$

$\theta_{enc} \leftarrow \theta_{enc} - \eta \nabla_{\theta_{enc}} L_{ae}$

$\theta_{dec} \leftarrow \theta_{dec} - \eta \nabla_{\theta_{dec}} L_{ae}$

`/* Adversarial Regularization */`

$\tilde{Z} \leftarrow$ samples drawed from the prior $p(z)$

$\tilde{V} \leftarrow log(Dis(\tilde{Z})) + log(1 - Dis(Z))$

$\theta_{dis} \leftarrow \theta_{dis} + \eta \nabla_{\theta_{dis}} \tilde{V}$

$V \leftarrow log(1 - Dis(Z))$

$\theta_{enc} \leftarrow \theta_{enc} + \eta \nabla_{\theta_{enc}} V$

until *converge*;

return network embedding code $Z = \{z_u\}_{u=1}^{n}$ and parameters: θ_{enc}, θ_{dec} and θ_{dis}

V. Experiments

In this section, we will verify the effectiveness of our model by a series of expriments compared to several baseline methods.

A. Datasets

In order to evaluate the effectiveness of the latent representations generated by the model, we use three real-world networks datasets in our expriments.

- Douban[1] is a well-known social media network in China, on which users post their likes or dislikes, through ratings and comments, on movies, books, musics and so on. And users can join the some clubs such as Douban Group and Douban Location to communicate with people who share the same interests. We use two datasets with group imfor-

[1] http://www.douban.com/

TABLE II
STATISTICS OF THE DATASET

Dataset	#(V)	#(E)
DOUBAN-MOVIE	779	1366
DOUBAN-BOOK	13024	169150
YELP	14085	150532

mation, i.e. Douban-book[2] and Douban-movie[3], crawled from Douban. Thus, these datasets can be used for the multi-label classification.

- Yelp [4] is a crowd-sourced local business review and social networking site, users can submit a review of merchants and communicate with other people. The dataset is used for link-prediction because of lack of relevant information.

The detailed description can be seen in Table II.

B. Evaluation

We use different evaluation metrics for the different tasks. For reconstruction, we use the Mean Average Precision(MAP) to evaluate the performance. The MAP is defined as:

$$AP(u) = \frac{\sum_v precision@k\,(u) \cdot \Delta_u\,(k)}{|\{\Delta_u\,(k) = 1\}|}$$

$$MAP = \frac{\sum_{u \in Q} AP\,(u)}{|Q|}$$

where $u \in V$ is the vertexes, $\Delta_u\,(k) = 1$ indicates that k-th vertex of ranked vertexes list has a link with vertex u. And Q is the query set.

For the link prediction, we use the widely employed metric AUC(i.e.Area Under the ROC curve) to evaluate the performance. The metric AUC is defined as:

$$AUC = \frac{\sum_{e \in E} rank_e - \frac{N_E(1+N_E)}{2}}{N_E \cdot N_{\overline{E}}}$$

where E is the edge set, N_E and $N_{\overline{E}}$ are the numbers of non-zero elements and zero elements. And $rank_e$ indicates the rank of e by the score of prediction.

For the multi-label classification, we use $Micro-F1$ and $Macro-F1$. $Macro-F1$ gives equal weight to each class. Let l and C as the a certain label and the overall label set. Then $Macro-F1$ is defined as follow:

$$Macro - F1 = \frac{\sum_{l \in C} F1\,(l)}{|C|}$$

$Micro–F1$ gives equal weight to each instance and defined as follow:

[2]https://book.douban.com/
[3]https://movie.douban.com/
[4]http://www.yelp.com/

TABLE III
NEURAL NETWORK STRUCTURES

Dataset	#nodes od each layer
DOUBAN-MOVIE	779-256-128
DOUBAN-BOOK	13024-1024-128
YELP	14085-1024-128

$$Pr = \frac{\sum_{l \in C} TP(l)}{\sum_{l \in C} (TP(l) + FP(l))}$$

$$R = \frac{\sum_{l \in C} TP(l)}{\sum_{l \in C} (TP(l) + FN(l))}$$

$$Micro - F1 = \frac{2 \cdot Pr \cdot R}{Pr + R}$$

where TP, FP and FN are the numbers of true positives, false positive and false negatives in the instances.

C. Baseline Methods

For better evaluation of the proposed ANE method, we compare it with the following methods.

- DeepWalk [6]: DeepWalk uses uniform random walk (i.e., depth-first strategy) to sample the inputs and trains the network embedding based on skip-gram.
- LINE [10]: LINE can learn two representation vectors for each node by optimizing two carefully designed objective function that preserves the first-order proximity and second-order proximity. Then the two representations are concatenated as the final representation.
- Laplacian Eigenmaps(LE): LE generates network representations by factorizing the Laplacian matrix of the adjacency matrix. It only exploits first-order proximity of network.
- Autoencoder: Autoencoder is a self-supervised technique, which tries to copy its input to its output. As result, the hiden layer would learn a compact representation of input. We use the train procedure proposed by [25].

D. Parameter Settings

In the experiment, we use different dimensions of each layer with different datasets. The structure of the encoder networks is listed in Table III. The structure of the decoder networks is similar to encoder but reverse. And the discriminator networks are similar to decoder and replace last layer with a sigmoid layer. And the priori distributions are Gaussion Distribution typically used in AAE.

For the ANE, the hyper-parameyers of α and K are tuned by using grid search. And for LINE, the learning rate of the starting value is 0.025 and the mini-batch size is 1 of the optimizer. Then the negative samples is set as 5. For DeepWalk, we set window size as 5, walk length as 40. For AE, we just use the second proximity information in it.

TABLE IV
MAP ON DOUBAN-BOOK, MOVIE AND YELP FOR RECONSTRUCTION TASK

Algorithm	DOUBAN-MOVIE	DOUBAN-BOOK	YELP
ANE	**0.847**	**0.636**	**0.697**
LINE	0.839	0.617	0.682
DeepWalk	0.694	0.437	0.513
LE	0.432	0.212	0.281
AE	0.225	0.131	0.122

TABLE V
AUC ON DOUBAN-BOOK, MOVIE AND YELP FOR LINK PREDICTION TASK

Algorithm	DOUBAN-MOVIE	DOUBAN-BOOK	YELP
ANE	**0.853**	**0.724**	**0.829**
LINE	0.836	0.709	0.805
DeepWalk	0.804	0.673	0.712
LE	0.631	0.617	0.622
AE	0.581	0.576	0.566

E. Network Reconstruction

In this part, we evalute the capacities of different methods on the network reconstruction. The network embedding aims to preserve the network structures in low-dimensional representations. For this reason, the good network embedding method should be able to capture the structures of network with the learned embeddings, which can reconstruct the network well. We use all three datasets in this experiment. Given a network, we use different network embedding methods to get the latent codes of the networks and then use the latent codes to reconstruct the networks by predicting the links of original networks. We can evaluate the reconstruction performance of differernt methods with the training set error, because the network reconstruction is via resuming the observed links. We use the MAP as the evaluation metrics. The result is shown in Table IV.

From the result, it evidently show that our method achieves the highest MAP over the baselines on three datasets. It demonstrates that our method can capture the network structures well. Specifically, for the network of the Douban-Movie network, the MAP of the ANE can reach 0.847. It indicates that our method can reconstruct the original network almost same as the normal.

The LINE also exploits both first-order and second-order proximity to capture the local and global network structures. And the results of ANE and LINE perform better than LE and AE, which are only exploits the first-order or second-order proximity to capture the network structures. It shows that both first-order, which indicates the local structures, and second-order, which indicates global structures, play an important role in preserving the network structures. And the reason why ANE performs better than LINE may be that the adversarial training improves the capacity of the model.

F. Link Prediction

In this part, we conduct link prediction task on all three networks. We first use vertex latent representations to compose edge representations and then use them to build a classifier for predicting whether there is an edge between two vertexes. Given a pair of vertexes (u, v) linked by an edge, we use an element-wise operator to combine the vertexes latent code z_u and z_v, which is Hadamard operator and defined as follow:

$$z_{uv} = z_u * z_v$$

We randomly hide 15 percentages of existing edges of these networks and use the AUC as the evaluation metric of predicting the hidden edges. And we also randomly sample pairs of vertexes as negative samples with an equal number of hidden edges, where is no edge linking the pair of vertexes. Together with the hidden edges and non-existent edges form the dataset. The result is shown in Table V.

We observe that the performance of ANE with Hadamard operator is better than other baselines on all three network. It demonstrates that the latent representations learned by our model have a better capacity on predicting new link.

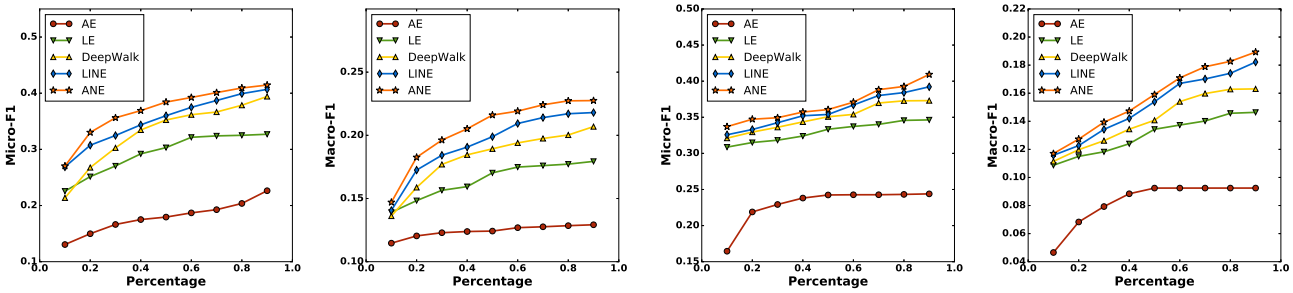
And the results of ANE and LINE also perform better than LE and AE. It proves the importance of expoliting both the first-order and second-order proximity simultaneously into the network embedding task again. The difference between results of LE and AE may come from that the first-order proximity indicates the link information directly. But the performances of ANE and LINE indicate the the second-order proximity may improve the ablity to find a new "neighbor".

G. Multi-label Classification

We use the multi-label node classification task to evaluate the quality of the latent representation learned by different models in this experiment, because of the importance of this task. Because the Groups in dataset Douban-Movie and Douban-book is excessive and some of them only have one member in the datasets we used, we remove those Groups that only has one member after getting the network embedding. The latent codes for the vertexes generated from all methods are used as features to classify each vertex into a set of labels. The one-vs-rest logistic regression classifier implemented in LibLinear package [26] is used in our experiments. we randomly sample 10% to 90% of the vertexes as the training samples and use the left vertexes to test the performance. The $Micro-F1$ and $Macro-F1$ are chosen as the evaluation metrics. The resuls are shown as follows.

In Figure 2(a) and Figure 2(b), we observe that the curve of ANE is above the curve of baselines. It shows that the latent representations of our method can achieve better performace on classification task than baselines.

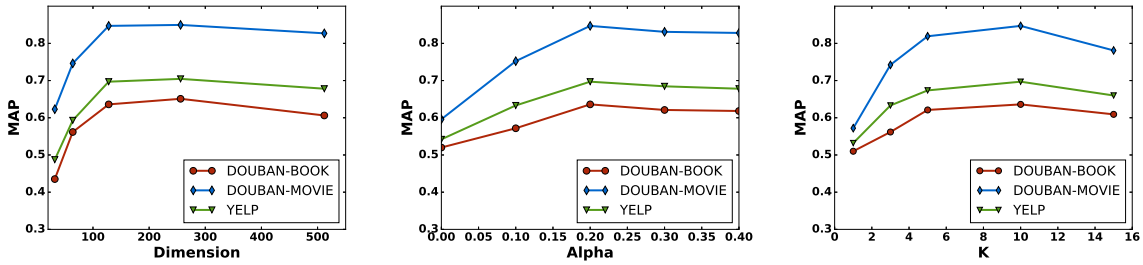
In most cases, the performance of AE is the worst among all methods. The reason may be that the zero elements are much more than non-zero elements in adjacency matrices of networks, and the traditional autoencoder gives an equal to each elements. Thus it is more prone to reconstruct the zero



(a) Micro-F1 and Macro-F1 on Douban-Movie

(b) Micro-F1 and Macro-F1 on Douban-Book

Fig. 2. Micro-F1 and Macro-F1 on Douban-Movie(a) and Douban-Book(b)



(a) dimension

(b) α

(c) K

Fig. 3. Parameter Sensitivity w.r.t. dimension, the value of α and the value of K

elements. It demonstrates that the model should pay more attention on non-zero elements.

H. Parameter Sensitivity

In this part, We explore the sensitivity of the performance w.r.t. the dimension of latent representations, the tradeoff parameter α and the overall penalty coefficient K. We report the MAP on all three datasets. The result is shown in Figure 3.

Embedding Dimension. First, we evaluate how the dimension of latent representation affects the performance of ANE. In Figure 3(a), we observe that the performances increase on all datasets at first, when the dimension of latent code increases. It indicates that more dimensions can encode more information from the network structures. However, as the dimension continuously increases, the performance starts to stop increasing slowly. The reason may be that the too many dimensions would not bring more useful imformation in code but introduces noises. Meanwhile, computational overhead has increased dramatically with the dimension increasing. All in all, it is important to find an appropriate dimension for the latent embedding representation.

Tradeoff Parameter α. Then we show how the α affects the performance in Figure 3(b). We can observe that the performance of $\alpha = 0$ is the worst. And the performance increases initially when the α increases, then it stop increasing. Because the parameter of α balances the weight of first-order and second-order proximity, the α indicates the how much impact of first-order proximity attached to model. The result demonstrates that both local and global structures of network are equally important on network embedding.

Penalty coefficient K. Finally, we evaluate how the penalty coefficient K affects the performance. The coefficient K is the reconstruction wieght of non-zero elements in the autoencoder component. The larger the K, the more the component concentrates on the non-zero elements. When $K = 1$, the model gives equal weight to each elements, the model would prone to reconstruct zero elements because of the sparsity of network. As the result shows, the performance of $K = 1$ is the worst. However, the performance decreases when the K increases to large. Because the model may reconstruct some non-existent links. It suggests that the suitable penalty to non-zero elements would improve the performance.

VI. Conclusion

In this paper, we have presented ANE for network embedding. Specifically, we introduce the adversarial autoencoder to network embedding with adversarial training, which can improve the performance of the latent representations of network by imposing an adversarial regularization. Meanwhile, we exploit the first-order and second-order proximity in the model to capture the network local and global structures. As a result, the embedding learnt by ANE will preserve the highly non-linear network structures well. Experimental results on network reconstruction, multi-label classification and link prediction show the effectiveness of ANE. In our future work,

we also plan to examine how to find an appropriate prior distribution for network embedding.

REFERENCES

[1] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*. Springer, 2011, pp. 115–148.

[3] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[4] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 283–292.

[5] D. Luo, F. Nie, H. Huang, and C. H. Ding, "Cauchy graph embedding," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 553–560.

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

[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[8] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

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

[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1067–1077.

[11] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations." in *AAAI*, 2016, pp. 1145–1152.

[12] G. Hinton. (2013) Non-linear dimensionality reduction. [Online]. Available: https://www.cs.toronto.edu/ hinton/csc2535/notes/lec11new.pdf

[13] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[15] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[16] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374.

[20] J. Li, J. Zhu, and B. Zhang, "Discriminative deep random walk for network classification." in *ACL (1)*, 2016.

[21] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2016, pp. 2014–2023.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[23] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[25] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[26] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

View publication stats