Variation Autoencoder Based Network Representation Learning for Classification

Hang Li $^{\#1}$, **Haozheng Wang** $^{\#2}$, **Zhenglu Yang** $^{\#3}$ **Masato Odagaki** $^{\dagger 4}$
$^{\#}$College of Computer and Control Engineering, Nankai University, Tianjin, China
† Maebashi Institute of Technology, Japan
$^{1\ 2}$\{hangl,hzwang\}@mail.nankai.edu.cn ^{3}yangzl@nankai.edu.cn ^{4}odagaki@maebashiit.ac.jp

Abstract

Network representation is the basis of many applications and of extensive interest in various fields, such as information retrieval, social network analysis, and recommendation systems. Most previous methods for network representation only consider the incomplete aspects of a problem, including link structure, node information, and partial integration. The present study introduces a deep network representation model that seamlessly integrates the text information and structure of a network. The model captures highly non-linear relationships between nodes and complex features of a network by exploiting the variational autoencoder (VAE), which is a deep unsupervised generation algorithm. The representation learned with a paragraph vector model is merged with that learned with the VAE to obtain the network representation, which preserves both structure and text information. Comprehensive experiments is conducted on benchmark datasets and find that the introduced model performs better than state-of-the-art techniques.

1 Introduction

Information network representation is an important research issue because it is the basis of many applications, such as document classification in citation networks, functional label prediction in protein-protein interaction networks, and potential friend recommendations in social networks. Although there are not a few recent work proposed to study the issue (Belkin and Niyogi, 2003; Tenenbaum et al., 2001; Cao et al., 2015; Tian et al., 2014; Cao, 2016), it is still far from satisfactory because of the intrinsic difficulty. In essence, the rich and complex information (i.e., link structure and node contents) embedded in information networks poses a significant challenge in the effective representation of networks.

Network-distributed representation learning can be viewed as a problem using low-dimensional vectors to represent nodes in a network. Most network representation methods are based on a network structure. The traditional representation is based on matrix decomposition and uses eigenvectors as representation (Belkin and Niyogi, 2003; Roweis and Saul, 2000; Tenenbaum et al., 2001). Furthermore, they extend to high-order information (Cao et al., 2015). However, these methods are not applicable to large-scale networks, and although many approximate approaches have been developed to solve this problem, they are not effective enough. Some methods are based on optimization objective functions (Tang et al., 2015; Pan et al., 2016; Yang et al., 2015). Although they are suitable for large-scale network data, they adopt shallow models that are limited in terms of performance and are difficult to use to obtain highly non-linear relationships that are vital to the preservation of network structure. Inspired by deep learning techniques in natural language processing, (Perozzi et al., 2014; Grover and Leskovec, 2016) adopted several stunted random walks in networks to generate node sequences serving as sentence corpus and then applied the skip-gram model to these sequences to learn node representation. However, they cannot easily handle additional information during random walks in a network.

To capture highly non-linear structures for large-scale networks, (Tian et al., 2014; Cao, 2016) introduced an autoencoder to model training instead of using a sampling based method to generate linear sequences. Motivated by this model, we develop the variational autoencoder (VAE) (Kingma and Welling, 2014), which is a deep generation model, instead of a basic autoencoder. Most previous studies utilized only one type of information in networks. The work in (Le and Mikolov, 2014) focused on node content, and others (Grover and Leskovec, 2016; Perozzi et al., 2014) explored link structure. Although a few previous models (Pan et al., 2016; Yang et al., 2015) combined both text information and network structure, they did not preserve the complete network structure and only partially utilized node content. A straightforward method is to learn representations from text features and network structure independently, and then concatenate the two separate representations.

To address the above issues, we introduce a deep generative model to learn network representation by modeling both node content information and network structure comprehensively. First, the representation based on node content through the paragraph vector model is obtained. Then, we feed the network adjacency matrix and representation obtained into a deep generative model, the building block of which is the VAE. After stacking several layers of the VAE, the result of the first layer is chosen before decoding as the final representation. Intuitively, we can obtain the representation containing both content information and structure in a d-dimensional feature space. The experimental evaluation demonstrates the superior performance of the model on the benchmark datasets.

2 Preliminary

Notation: Let $G = (V, E, C)$ denote a given network, where $V = \{v_i\}_{i=1...N}$ is the node set and $E = \{e_{ij}\}$ is the edge set that indicates the relation of nodes. If a direct link exists between v_i and v_j then $e_{ij} = 1$; otherwise, $e_{ij} = 0$ when network is unweighted. $C = \{c_i\}$ is the set of content information. let A denote the adjacency matrix for a network, and let $x = \{e_{i,k}, ..., e_{n,k}\}$ be an adjacency vector. Our goal is to seek a low-dimensional vector $\vec{y}_j$ for each node v_i of a given network.

Autoencoder: We first provide a brief description of a basic autoencoder and the VAE. The basic autoencoder first compresses the input into a small form and then transforms it back into an approximation of the input. The encoding part aims to find the compression representation z of a given data x, and the decoding part is a reflection of the encoder used to reconstruct the original input x. The VAE (Kingma and Welling, 2014) imposes a prior distribution on the hidden layer vector of the autoencoder and re-parameterizes the network according to the parameters of the prior distribution. Through the parameterization process, the means and variance values of the input data can be learned. We extended VAE to generate two means and variances of input data, which can be considered correspond to the content and structure respectively.

3 Model Description

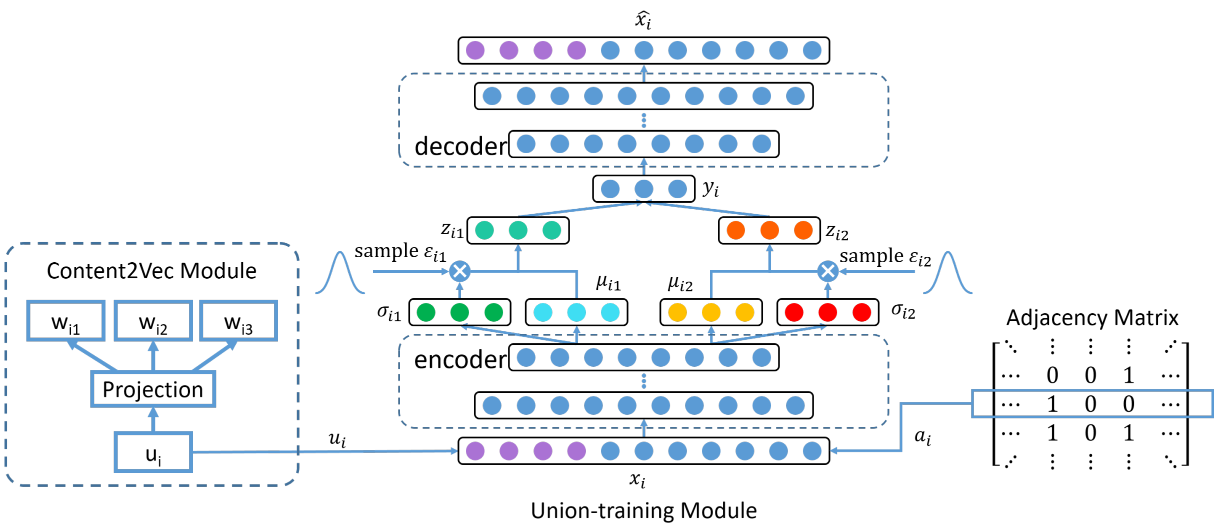
The architecture of the proposed model is shown in Fig. 1. The whole architecture consists of two main modules, namely, the content2vec module and the union training module. For an information network, such as a paper citation network, we can obtain the node link and content information (e.g., paper abstract). We learn an effective feature representation vector that preserves both structure information and node content information and can thus be applied to many tasks (e.g., paper classification).

3.1 Content2vec Module

We employ the state-of-the-art approach called doc2vec (Le and Mikolov, 2014), which utilizes text to learn vector representations of documents, as our content2vec module. Specifically, if one node contains other information (e.g., author name), we treat it as a word and merge it into the comprehensive text information (e.g., the abstract of the paper in the citation network) as the content of the node. A representation $\boldsymbol{u}_i$ that includes the node content information is obtained from this module.

3.2 Union-training Module

The union training module is the core part of our model, in which content information and structure information are integrated. The details are shown in Fig 1. The VAE is adopted as the main block. Given a network, the adjacency matrix $\boldsymbol{A}$ can be obtained. $\boldsymbol{A}$ can describe the relationship among the nodes and reflect the overall structure of the network. We extract each adjacency vector $\boldsymbol{a_i}$ and concatenate it with the corresponding $\boldsymbol{u_i}$ as the input $\boldsymbol{x}_i$ of our model. Therefore, the content and



$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 0 & 0 & 1 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 1 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Figure 1: Architecture of our model. w_i can be seen as a word of the content information, u_i is a node in the network, $\boldsymbol{u}_i$ is a representation vector learned by the Content2Vec Module, $\boldsymbol{x}_i$ is a vector of the adjacency matrix. The input of the union-training module is combination of $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, the encoder and decoder are stack full-connected layer, $\boldsymbol{\sigma}_{i1}, \boldsymbol{\sigma}_{i2}, \boldsymbol{\mu}_{i1}, \boldsymbol{\mu}_{i2}$ can be seen the mean and variance of the distribution of the content and structure data, respectively. $\boldsymbol{\varepsilon}_{i1}$ and $\boldsymbol{\varepsilon}_{i2}$ are the sample data from two Gaussian distributions.

structure information is able to be learned simultaneously.

During the encoding phase, we adapt several fully connected layers composed of multiple nonlinear mapping functions to map the input data to a highly nonlinear latent space. Therefore, given the input $\boldsymbol{x}_i$, the output h^k for the k^{th} layer is shown as follow:

$$\begin{aligned} \boldsymbol{h}^1 &= \pi(\boldsymbol{W}^1\boldsymbol{x}_i + \boldsymbol{b}^1) \\ \boldsymbol{h}^k &= \pi(\boldsymbol{W}^k\boldsymbol{h}^{k-1} + \boldsymbol{b}^k), k = 1, 2...K \end{aligned} \tag{1}$$

where π is the nonlinear activation function of each layer. The value of K varies with the data.

In the last layer of encoder, we obtain four output: $\boldsymbol{\mu}_{i1}$, $\boldsymbol{\sigma}_{i1}$ $\boldsymbol{\mu}_{i2}$ and $\boldsymbol{\sigma}_{i2}$. They can be treated as the means and variances of the distribution of content information and structure information respectively. Furthermore, we sample two values ε_{i1} and ε_{i2} from two previous distributions (e.g., Gaussian distribution). Then we can obtain the reparameterized $\boldsymbol{z}_i1$ and $\boldsymbol{z}_21$. Through concatenate $\boldsymbol{z}_i1$ and $\boldsymbol{z}_21$, content and structure information can be integrated together, $\boldsymbol{y}_i$ is the representation of the network. Nonlinear operations are not performed in this phase. Thus, the gradient descent method can be safely applied in optimization. The operations can be expressed as follows:

$$\begin{aligned} \boldsymbol{z}_{ik} &= f(\boldsymbol{\mu}_{ik}, \boldsymbol{\sigma}_{ik}, \boldsymbol{\varepsilon}_i), k = 1, 2 \\ \boldsymbol{y}_i &= Merge[\boldsymbol{z}_{i1}, \boldsymbol{z}_{i2}] \end{aligned} \tag{2}$$

where f is a linear function that can reparameterize $\boldsymbol{y}_i$, $Merge$ concatenate the two vectors together directly.

The decoding phase is a reflection of the encoder; its output $\hat{\boldsymbol{x}}_i$ should be close to the input $\boldsymbol{x}_i$. The loss function of this module that should be minimized is as follows:

$$\mathcal{L}(\boldsymbol{x}_i) = -\sum_{k=1}^{2} \mathcal{KL}(q(\boldsymbol{z}_{ik}|\boldsymbol{x}_i)||p(\boldsymbol{z}_{ik})) + \mathcal{H}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) \tag{3}$$

where KL is the KL divergence which is always used as a measure of the difference between two distributions, $\mathcal{H}$ is a cross-entropy function that is used to measure the difference between $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$.

Finally, We choose the output of the layer $\boldsymbol{y}_i$ as the final representation of each node.

4 Experiments

4.1 Experimental setup

Paper citation networks is a classical social information network. To evaluate the quality of the proposed model, we conduct three important tasks on two benchmark citation network datasets: (1)

Table 1: Macro-F1 score on Citeseer-M10 Network

%p	One-Hot	Deepwalk	Node2vec	Doc2vec	DW+D2V	TADW	TriDNR	Ours
10%	0.254	0.297	0.314	0.503	0.526	0.475	0.683	**0.889**
30%	0.321	0.334	0.331	0.536	0.615	0.488	0.744	**0.913**
50%	0.352	0.346	0.346	0.547	0.633	0.495	0.760	**0.924**
70%	0.363	0.344	0.339	0.534	0.630	0.495	0.773	**0.940**

Table 2: Macro-F1 score on DBLP Network

%p	One-Hot	Deepwalk	Node2vec	Doc2vec	DW+D2V	TADW	TriDNR	Ours
10%	0.328	0.379	0.448	0.574	0.495	0.660	0.724	**0.751**
30%	0.362	0.454	0.473	0.598	0.586	0.687	0.742	**0.753**
50%	0.371	0.459	0.475	0.604	0.614	0.697	0.747	**0.762**
70%	0.372	0.461	0.476	0.605	0.628	0.699	0.748	**0.763**

CiteSeerM10[1]. It contains 10 distinct categories with 10,310 papers and 77,218 citations. Titles are treated as the text information because no more text information is available; and (2) DBLP dataset[2]. We treat abstracts as text information and choose 4 research areas with the same setting as that of (Pan et al., 2016), which are database (SIGMOD, ICDE, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, CIKM), data mining (KDD, ICDM, SDM, PKDD, PAKDD), artificial intelligent (IJCAI, AAAI, NIPS, ICML, ECML, ACML, IJCNN, UAI, ECAI, COLT, ACL, KR), computer vision (CVPR, ICCV, ECCV, ACCV, MM, ICPR, ICIP, ICME). Therefore we get a network contains 30,422 nodes and 41,206 edges.

We compare our approach with the following methods:

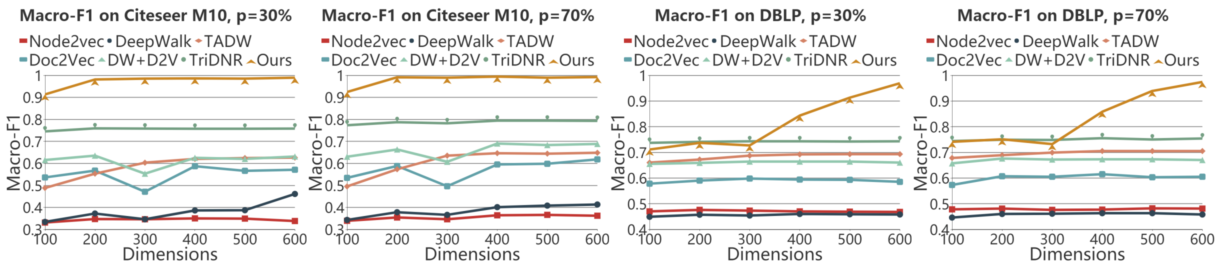
- **One-Hot** uses adjacency matrix, which carries the structure information as the high-dimension representation, and directly feed into the classifier.
- **DeepWalk** (Perozzi et al., 2014) is exploited by statistical models, which employs truncated random walks to learns nodes embedding by treating walk as the equivalent of sentences.
- **Node2vec** (Grover and Leskovec, 2016) learns the network representation by designing a biased random walk procedure which efficiently explores diverse neighborhoods.
- **Doc2vec** (Le and Mikolov, 2014) is the Paragraph Vector model that learns document representation by predicting the words appeared.
- **DW+D2V** is simply to concatenate the representation result learned by DeepWalk and Doc2vec.
- **TADW** (Yang et al., 2015) is text-based DeepWalk, which incorporates text information into network structure by matrix factorization.
- **TriDNR** (Pan et al., 2016) uses node text, label, and structure to jointly learn node representation.

4.2 Performance on Node Classification

We conduct the paper classification task on two benchmark citation networks to evaluate the performance of our method. To reduce the impact of sophisticated classifiers on the performance, we employ a linear SVM, which is a common technique used by the exiting work (Pan et al., 2016). The results are shown in Table 1 and Table 2, respectively. The reported parameters for our model are set: dimension d=100 on CiteseerM10 and d=300 on DBLP. The dimension for other algorithms is the same as ours, and the other parameters are set as their papers report, i.e., window size b=10 in DeepWalk and Node2vec, in-out parameter q=2 in Node2vec, text weight ∂=0.8 in TADW and TriDNR. We use Macro-F1 which is the same as that adopted by other algorithms to measure the classification performance. The experiments are independently conducted 10 times for each setting, and the average values are reported. The proportion of training data with labels is range from 10% to 70%.

[1]http://citeseerx.ist.psu.edu/

[2]http://arnetminer.org/citation (V4 version is used)



(a) (b) (c) (d)

Figure 2: Performance of each strategy on different training proportion p

Our model is evaluated by comparing it with seven approaches. One-Hot uses the original structure data, and its performance is poor because it is discrete and the context relation of nodes can not be captured. DeepWalk and Node2vec are structure-based methods that exhibit inferior performance mainly because they only use the shallow structure information and the network is rather sparse, while the information of the complex non-linear structure cannot be employed. The performance of Doc2vec is not as good as ours which demonstrates the effectiveness of our proposed model. TADW and TriDNR are inferior to our approach, although these two methods also consider the text and structure. Nevertheless, they cannot capture the complex non-linear structure. The reason for the superior of our method is that our model can effectively capture the inter-relationship between node content and link structure, and the intro-relationship among nodes and links, which are essential to learn the representation of networks. Furthermore, our model can capture the information of highly non-linear structure instead of the shallow structure (e.g., DeepWalk) by exploiting VAE. Moreover, our approach does not require heavy text information which is utilized by the other state-of-the-art strategies (e.g.,TriDNR). Our model exhibits consistent superior performance, and is up to 16% better than the state-of-the-art methods (i.e., the Macro-F1 score of our model is 94% when the proportion of training data with labels is 70% conducted on the Citeseer-M10 Network dataset).

4.3 Parameter Setting

A significant hyperparameter in our model is the dimension d. The performance of different methods with varying dimensions has been evaluated. The result is illustrated in Fig. 2. We obtain very good performance on the CiteSeer-M10 dataset, i.e., the Macro-F1 score is 94% and the performance tends to be stable as b becomes larger. It validates the effectiveness of our algorithm and the reason is due to the ability of our model that can capture the complex network structure and the text information. From Fig. 2, we can see that the performance gets better when d increases from 100 to 600. We think the main reason is because more information can be preserved in higher dimensional space of the datasets.

5 Conclusions

In this paper, we have introduced an effective network representation model, which comprehensively integrates the text information and the network structure. We introduced Paragraph Model as a preliminary module. And we have exploited Variational Autoencoder as the main block of our model, that could capture highly non-linear structure of the network. The comprehensive experimental evaluation on two benchmark datasets has demonstrated the effectiveness of the model.

Acknowledgement

This research is supported by the National Natural Science Foundation of China: U1636116, 11431006, Research Fund for International Young Scientists: 61650110510, Ministry of Education of Humanities and Social Science: 16YJC790123.

References

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.

Shaosheng Cao. 2016. deep neural network for learning graph representations. In *AAAI*.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep:learning graph representations with global structural information. pages 891–900.

A Grover and J Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes .

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *IJCAI*.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*.

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. 2001. A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500):2319–2323.

F. Tian, B. Gao, Q. Cui, E. Chen, and T. Y. Liu. 2014. Learning deep representations for graph clustering. *Inproceedings* .

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*.