Network-Specific Variational Auto-Encoder for Embedding in Attribute Networks

Di Jin[1] , **Bingyi Li**[1] , **Pengfei Jiao**[2,1,*] , **Dongxiao He**[1] and **Weixiong Zhang**[3]

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Center of Biosafety Research and Strategy, Tianjin University, Tianjin, China
[3]Department of Computer Science and Engineering, Washington University, St. Louis, USA
{jindi, libingyi, pjiao, hedongxiao}@tju.edu.cn, weixiong.zhang@wustl.edu,

Abstract

Network embedding (NE) maps a network into a low-dimensional space while preserving intrinsic features of the network. Variational Auto-Encoder (VAE) has been actively studied for NE. These VAE-based methods typically utilize both network topologies and node semantics and treat these two types of data in the same way. However, the information of network topology and information of node semantics are orthogonal and are often from different sources; the former quantifies coupling relationships among nodes, whereas the latter represents node specific properties. Ignoring this difference affects NE. To address this issue, we develop a network-specific VAE for NE, named as NetVAE. In the encoding phase of our new approach, compression of network structures and compression of node attributes share the same encoder in order to perform co-training to achieve transfer learning and information integration. In the decoding phase, a dual decoder is introduced to reconstruct network topologies and node attributes separately. Specifically, as a part of the dual decoder, we develop a novel method based on a Gaussian mixture model and the block model to reconstruct network structures. Extensive experiments on large real-world networks demonstrate a superior performance of the new approach over the state-of-the-art methods.

1 Introduction

Network embedding (NE) is to learn a low-dimensional distributed representation for each node in a network and at the same time preserve the relationships among the nodes in the network. The learned embedding can be applied to many applications, such as network classification and clustering.

Several different NE methods have been developed [Cui *et al.*, 2018], many of which exploit network topological structures [Perozzi *et al.*, 2014; Grover and Leskovec, 2016; Wang *et al.*, 2016a]. Semantics on nodes or node attributes have also been introduced to supplement the topological information to improve the quality of NE [Zhang *et al.*, 2018a]. The existing NE methods adopt techniques from spectral optimization [Huang *et al.*, 2017], random walk [Yang *et al.*, 2015; Pan *et al.*, 2016], matrix factorization [Yang *et al.*, 2016] and Auto-Encoder (AE) [Liao *et al.*, 2018; Jin *et al.*, 2018; Tao *et al.*, 2019]. In particular, AE based methods have attracted much attention lately because it can well capture highly non-linear relationships in networks.

However, learning an embedding by incorporating network topological and node semantic information altogether is non-trivial, and the dimension of the combined data is much higher than that of network topology alone. The encoders used in AEs typically have discrete outputs and learn a function to directly map the data, making it difficult to deal with high-dimensional data. To cope with this problem, Variational Auto-Encoder (VAE) [Kingma and Welling, 2014] has been introduced by adding a priori constraint to the learning of embedding. The encoder of VAE infers a posterior distribution of continuous latent variables given the observed input, instead of learning the discrete latent variables directly as did in AE. As a result, it is more suitable for high-dimensional and complex data such as networks. This advantage also motivates to employ VAE to learn embedding in attributed networks. For example, the method in [Kipf and Welling, 2016] uses a graph convolutional network as a decoder to embed an attribute network under the VAE framework. The method in [Li *et al.*, 2017] employs the doc2vec method to learn an attribute vector representation, and then integrates it with the network topology in the VAE framework. The method in [Pan *et al.*, 2019] proposes an adversarially regularized variational graph autoencoder by considering the topological structure and node semantics in networks.

Although these VAE methods have reasonable performance, they still have serious drawbacks. They typically use the same method to map the data of network topologies and node attributes, i.e., they use the same generative mechanism, e.g. conventional neural networks, to reconstruct network topologies and node attributes. It is critical to note that network topologies and node attributes are completely different types of data. The former reflects a *coupling* relationship among nodes, whereas the latter represents *independent* node semantics. They are generally produced by completely different mechanism in nature. The existing VAE methods, as well as many NE methods, do not differentiate these two types of information, making them ineffective for many applications.

[*]Corresponding author.

To address this problem, we developed a new network-specific VAE approach, named as NetVAE, for embedding attribute networks. NetVAE uses a shared encoder to compress altogether the information of network topology and information of node attribute to derive an embedding. By using a shared encoder, the information from two distinct sources can have a joint embedding and share a joint training process so as to perform a co-training for cross learning. In the process of decoding, NetVAE uses a dual decoder with two different generative mechanisms to reconstruct separately the network topologies and node attributes. It has been argued that a conventional fully connected neural network is suitable for reconstruction of features of individual variables. We adopt this scheme to reconstruct node attributes. To reconstruct network topologies, we take advantages of the nonlinear, modular structures that the encoder implicitly extracts when compressing the input network. To this end, we first introduce a Gaussian mixture model (GMM) to make such latent modular structures explicit, and then introduce a block model [Holland *et al.*, 1983], an excellent model for network topologies, to directly exploit in the decoder such modular information for structure reconstruction. We formalize the GMM and block model as a neural network layer and integrate it into the VAE. In short, we make the following contributions.

- We developed a network-specific VAE for embedding of topological and attribute information of a network. This is the first time that these two types of information are processed differently in VAE for NE. We used the most suitable mechanisms to reconstruct network structures and node attributes separately.
- We introduced a GMM to make the nonlinear modular structures extracted by the encoder explicit so that such structures can be exploited in the decoder to improve overall embedding quality.
- We also introduced a novel network generation mechanism that combines the popular block model for networks and the GMM. The method can not only describe the inherent coupling relationship among nodes but also preserve network module structures, which helps improve the quality of embedding significantly.
- We formulated the GMM and the network generation method as an equivalent neural network layer and incorporate it in the overall VAE architecture so that an end-to-end training can be efficiently performed.

It is important to note that although our method was not originally designed for community detection, it can indeed be adopted to find communities as a by-product. Note that the dimension of the embedding does not necessarily have to be the same as the number of the actual communities in the given network, as long as the former is not smaller than the latter so that no intrinsic network property is lost after compression.

2 The Method

Let $G = (V,E)$ be an undirected attribute network with n nodes $V = \{v_1,v_2,...,v_n\}$ and e edges $E= \{e_{ij}\}$. The topological structures of G is specified by an adjacency matrix $\mathrm{A} \in R^{n\times n}$, and node semantics are represented by an attribute matrix $\mathrm{X} \in R^{n\times m}$ of maximal m attributes. Given a network, network embedding is to learn an $l \ll n$ dimensional vector $\mathrm{u}_i \in R^l$ for each node v_i based on A and X.

2.1 Overview of the Method

The NetVAE method for embedding of attribute networks has two major components, a joint encoder and a dual decoder (Figure 1). The joint encoder maps both network topologies and node attributes jointly to derive the latent variables for the embedding of the network. The dual decoder uses two distinct generative methods to reconstruct network topologies and node attributes separately (Figure 1).

In the encoder of NetVAE, we concatenate to the network topologies and node attributes in a unified representation, and then encode the combined data using a fully connected neural network to learn the parameters, i.e., the mean and standard deviation, of the normally distributed variables of the target embedding. We then synthesize the latent variables of the embedding by using the two parameters along with a Gaussian noise.

In the decoder, we adopt two different methods to regenerate network structures and node attributes. We employ a fully connected neural network to reconstruct node attributes. We introduce a new generative mechanism that explicitly exploits the nonlinear modular structures of the original network. Note that the encoder, a neural network on its own, is able to extract the nonlinear modular structures hidden in the given network in such a way that nodes belonging to the same network module tend to be grouped together in the embedding. Such modular structures can be utilized to better re-construct the network structures. To do so, we introduce a Gaussian mixture model (GMM) to make the latent modular structures explicit in the embedding space and adopt a method, based on the block model (i.e. a popular tool for describing network with communities), to exploit the modular structures for reconstruction of network structures. We then formalize the new generationve method as a neural network. All of these components are then integrated under the framework of VAE.

2.2 The Shared Encoder

We first concatenate the two types of observed data, data of network topology in the adjacency matrix A and data of node semantics in the attribute matrix X, into a unified representation $\mathrm{U} = [\mathrm{A}, \mathrm{X}]$, where the row vector u_i contains the adjacency list and attributes of node v_i. We then use a fully-connected neural network of three layers as the encoder to map each node to a nonlinear low-dimensional latent embedding space. In the first two layers of the encoder, the output of the t-th layer (for t = 1 or 2) is defined as:

$$\hat{\mathrm{h}}_i^{(t)} = f(\mathrm{W}^{(t)}\hat{\mathrm{h}}_i^{(t-1)} + \mathrm{b}^{(t)}) \tag{1}$$

where $\hat{\mathrm{h}}_i^{(0)} = \mathrm{u}_i$, $\mathrm{W}^{(t)}$ and $\mathrm{b}^{(t)}$ are the weights and bias of the layers, and $f(\cdot)$ the activation function, such as $ReLU\ (\cdot)$, for layer t.

The outputs of the last or third layer of the encoder consist of two parameters μ_i and σ_i of the normally distributed latent

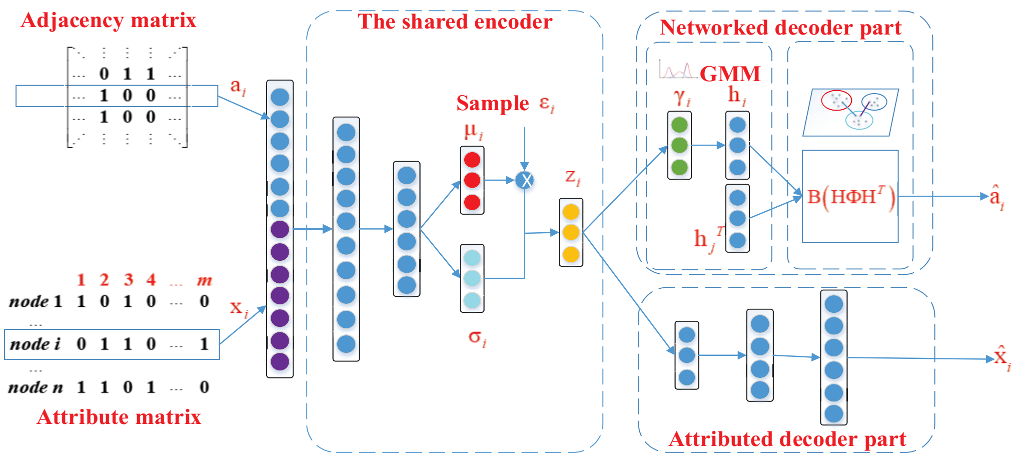

Figure 1: A sketch of NetVAE. The adjacency vector a_i and attribute vector x_i are concatenated into a unified representation u_i which is then feed into the shared encoder to derive the embedding variable z_i. The decoder has two parts, a decoder for network structures and a decoder for node attributes. The major innovation of NetVAE is in the decoder for network structures; see main text for detail.

variables of the embedding, which are calculated as follows,

$$\mu_i = W^{(\mu)}\hat{h}_i^{(2)} + b^{(\mu)}, \sigma_i = f(W^{(\sigma)}\hat{h}_i^{(2)} + b^{(\sigma)}) \quad (2)$$

where $W^{(\mu)}$ and $b^{(\mu)}$ are the weights and biases of the last layer for deriving μ_i, and $W^{(\sigma)}$ and $b^{(\sigma)}$ are that for σ_i. Following the reparameterization scheme in [Kingma and Welling, 2014], we use μ_i and σ_i as the mean and variance to synthesize the latent variable z_i of the embedding of node v_i as follows,

$$z_i = \mu_i + \sigma_i \circ \varepsilon_i \quad (3)$$

where ε_i is a noise sampled from the Gaussian distribution $N(0, I)$ with zero mean and an identity covariance matrix I, and $\circ$ denotes the element-wise multiplication. Based on this neural network structure, we map each node v_i to a latent variable z_i in the low-dimensional embedding space.

To constrain the latent variable z_i, as did in the original VAE [Kingma and Welling, 2014], we also impose a prior constraint, using the mean and variance of z_i, to ensure the target distribution of z_i to be a Gaussian distribution, i.e., we have the following constraint:

$$\begin{aligned} L_p &= D_{KL}\left(N(Z;\mu,\sigma^2)||N(Z;0,I)\right) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{l}\left(-\log\left(\left(\sigma_i^j\right)^2\right) + \left(u_i^j\right)^2 + \left(\sigma_i^j\right)^2 - 1\right) \end{aligned} \quad (4)$$

where D_{KL} is the KL divergence. Z, μ and σ are respectively the variable sets of z_i, μ_i and σ_i, l is the dimension of latent variable z_i, and u_i^j and σ_i^j are the j-th elements of μ_i and σ_i. This constraint will be added to the loss function as a regularization term.

2.3 The Dual Decoder

Different from the shared encoder, the decoder is composed of two distinct sub-decoders to reconstruct separately the network topology and node attributes.

The Decoder for Reconstructing Network Structure

This is the *core* part of the dual decoder. We first discuss the main idea and formalize it as a neural network. We then introduce a neural network variant of the Gaussian Mixture Model (GMM) to help integrate it into the VAE architecture.

The scheme for network reconstruction. Assume that there are l network modules in the given network, and the nodes in each module have the same link pattern, i.e., all nodes in the same module share the same link probability when connecting to the other nodes. (The number of modules can be also smaller than the dimension of network embedding while we make them equal here without loss of generality.) To be specific, let $\Gamma=(\gamma_{ik})_{n\times l}$ be the distributions of the module labels for the nodes in the network, where γ_{ik} denotes the probability that node v_i belongs to the k-th module, and $\Phi= (\phi_{rs})_{l\times l}$ be the distributions of the links across communities where ϕ_{rs} denotes the link probability between any two nodes in communities r and s, respectively, and $\phi_{rs} = \phi_{sr}$ for undirected networks. We first sample the module labels g_i for every node v_i from a multinomial distribution with parameters γ_i, that is:

$$g_i \sim Multinomial(\gamma_i) \quad (5)$$

where $\gamma_i=(\gamma_{ik})_{1\times l}$ denotes the probability distribution of the label of node v_i over the modules. Now consider the link between nodes v_i and v_j, which may follow a Bernoulli distribution $a_{ij} \sim Bernoulli(\phi_{g_i g_j})$ with parameters $\phi_{g_i g_j}$. Essentially, this is to generate the network links according to l types of link patterns (each of which corresponding to a module), and thus naturally describes the coupling relationship between nodes in networks considering l network communities. Note that nodes with larger degrees are more likely to be connected. We then use $d_i d_j \phi_{g_i g_j}$ to describe the link probability between nodes v_i and v_j, where d_i is the degree of v_i. We then sample each link between v_i and v_j from a Bernoulli distribution with parameters $d_i d_j \phi_{g_i g_j}$, defined as:

$$a_{ij} \sim Bernoulli(d_i d_j \phi_{g_i g_j}) \quad (6)$$

By considering the node degrees, (6) better describes the coupling relationship between nodes coming from l communities. This matches well with the well-known stochastic block model [Karrer and Newman, 2011], which could generate networks with well-defined statistical properties.

Formulating network reconstruction in neural network. To make the above network generative process suitable for integration into VAE, we formulate it as a layer of neural network. For convenience, we use a l-dimension vector h_i to represent the module g_i for node v_i, where $h_{ir} = 1$ if $r = g_i$, or 0, otherwise, forming a module indicator matrix $\mathrm{H} = (h_{ir})_{n\times l}$ for all nodes. The probability that nodes v_i and v_j are connected can then be revised to $\hat{a}_{ij}{=}d_i(\mathrm{h}_i\Phi\mathrm{h}_j^T)d_j$, which can also be written in matrix form as:

$$\hat{\mathrm{A}} = \mathrm{B}(\mathrm{H}\Phi\mathrm{H}^T) \tag{7}$$

where $\mathrm{B} = (b_{ij})_{n\times n}$ with $b_{ij} = d_i \times d_j$. Since B is a constant matrix and H can be sampled from the multinomial distribution over Γ following formula (5), we can formulate (7) as a layer of neural network by taking Φ as the weights of the layer which are to be learned in training of the neural network. We then sample the observed network topology A from a Bernoulli distribution over $\hat{\mathrm{A}}$, the likelihood of which can be defined as: $p\left(\mathrm{A}|\hat{\mathrm{A}}\right) = \prod\limits_{i<j} \hat{a}_{ij}^{a_{ij}}(1-\hat{a}_{ij})^{1-a_{ij}}$. Since maximizing this likelihood is the objective of the network generative process, we can then take the negative log-likelihood as the loss function of this layer of the neural network,

$$L_t = -\sum_{i<j}\left(a_{ij}\ln\hat{a}_{ij} + (1-a_{ij})\ln\left(1-\hat{a}_{ij}\right)\right) \tag{8}$$

Converting latent variables to module memberships. This new network structure reconstruction method must also specify the probabilities that a node is assigned to the l modules (i.e. Γ). Nevertheless, the encoder described above will instead learn a set of embedding latent variables Z. We need to convert the embedding variables into node module memberships. To this end, we introduce a neural network variation of the GMM into VAE. Given the latent variables $Z = \{\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_n\}$ and node module labels $g = \{g_1, g_2, ..., g_n\}$, the log-likelihood under the GMM model can be defined as:

$$\ln p(\mathrm{Z}|\theta) = \sum_{i=1}^{n}\ln\sum_{g_i}\pi'_k N(\mathrm{z}_i, g_i|\mu'_k, \sigma'_k) \tag{9}$$

where π'_k, μ'_k and σ'_k are the mixing coefficients, mean and covariance for the k-th module in GMM. We use θ to denote the set of all these parameters for convenience. By applying Jensen's inequality to (9), we derive a lower bound of the log likelihood as $\xi(\theta)$:

$$\xi(\theta) = \sum_{i=1}^{n}\sum_{g_i} q_i(g_i)\ln\left(\frac{\pi'_k N(\mathrm{z}_i, g_i|\mu'_k, \sigma'_k)}{q_i(g_i)}\right) \leq \ln p(\mathrm{Z}|\theta) \tag{10}$$

where the probability distribution $q_i(g_i)$ in (10) can be freely chosen. The equality holds when $q_i(g_i)$ is equal to the real posterior distribution. So we can derive it as follows:

$$q_i(g_i) = p(g_i|z_i, \theta) = \gamma_i \tag{11}$$

Ignoring the item that are not related to parameters θ and marginalizing g_i, $\xi(\theta)$ can be further written as:

$$\xi(\theta) = \sum_{i=1}^{n}\gamma_{ik}\sum_{k=1}^{c}\pi'_k N\left(z_i|\mu'_k, \sigma'_k\right) \tag{12}$$

To apply this lower bound to VAE, we introduce a layer of neural network instead of traditional Bayes rules to calculate the posterior distribution γ_i based on the latent variable z_i, that is:

$$\gamma_i = softmax(\mathrm{W}^{(\gamma)}z_i + \mathrm{b}^{(\gamma)}) \tag{13}$$

which serves as the E-step in the EM algorithm. Here, $\mathrm{W}^{(\gamma)}$ and $\mathrm{b}^{(\gamma)}$ are the weights and biases in this neural network layer. By setting the derivatives of $\xi(\theta)$ with respect to π'_k, μ'_k and σ'_k to zero, we estimate the parameters as follows:

$$\begin{gathered}\pi'_k = \sum_{i=1}^{n}\frac{\gamma_{ik}}{n}, \mu'_k = \frac{\sum_{i=1}^{n}\gamma_{ik}\mathrm{z}_i}{\sum_{i=1}^{n}\gamma_{ik}}\\ \sigma'_k = \frac{\sum_{i=1}^{n}\gamma_{ik}(\mathrm{z}_i-\mu'_k)(\mathrm{z}_i-\mu'_k)^T}{\sum_{i=1}^{n}\gamma_{ik}}\end{gathered} \tag{14}$$

which collectively serve as the M-step in the EM algorithm.

Next, to integrate this GMM model into the neural network, we follow the technique proposed in [Zong *et al.*, 2018] and add the following energy term $E(\mathrm{z}_i)$ to every node based on the parameters in (14), i.e.,

$$E(\mathrm{z}_i) = -log(\sum_{k=1}^{c}\pi'_k\frac{\exp(-\frac{1}{2}(\mathrm{z}_i-\mu'_k)^T\sigma'^{-1}_k(\mathrm{z}_i-\mu'_k))}{\sqrt{|2\pi\sigma'_k|}}) \tag{15}$$

We further calculate the average energy of all the n nodes,

$$L_c = \frac{1}{n}\sum_{i=1}^{n}E(\mathrm{z}_i) \tag{16}$$

which is added to the loss function of VAE. By minimizing the loss in training the neural network, we derive an accurate γ_i to be used for reconstructing network structures.

The Decoder for Reconstructing Node Attributes

Besides network topology, we also need to reconstruct the node attributes. To this end, we use the latent variable z_i for each node v_i as the input to a fully connected neural network to reconstruct node attributes. This is formulated in the same way as in (1) for the encoder, but with increased dimensions of the outputs layer by layer. We use the output of the last layer as the final reconstructed node attributes, denoted as $\hat{\mathrm{x}}_i$. The loss or objective function of this decoder is

$$L_a = \sum_{i=1}^{n}\ell(\hat{\mathrm{x}}_i, \mathrm{x}_i) \tag{17}$$

where ℓ is the cross entropy between $\hat{\mathrm{x}}_i$ and x_i.

2.4 The Unified Model and its Optimization

To sum up, the loss function of the whole NetVAE model is defined as:

$$L = L_t + L_a + L_p + L_c \tag{18}$$

The training process is to minimize the loss function of a given network. The first two items, L_t in (8) and L_a in

(17), are the reconstruction errors of the network structures and node attributes, respectively. The third item, L_p in (4), is a prior constraint item, which encourages the distributions of the latent variables to match the prior distributions. The last item, L_c in (16), is the average energy from introducing GMM. The NetVAE can then be optimized via stochastic gradient descent in back-propagation. The overall complexity of this method is competitive with that of the original VAE.

3 Experimental Analysis

We validated and evaluated NetVAE on two widely used applications of network node classification and node clustering (i.e., community detection). Here we also present the results and an analysis to understand why the method works.

3.1 Experiment Setup

We used seven public datasets with varying sizes (Table 1). Among these datasets, Cornell, Texas, Washington and Wisconsin (which are sub-datasets of WebKB) are webpage datasets from four universities. Citeseer is a citation network. UAI2010 contains articles information from Wikipedia pages. Pubmed is a scientific publications dataset.

Datasets	**Cornell**	**Texas**	**Waton**	**Wisin**	**Cite**	**UAI**	**Pubmed**
n	195	183	217	262	3,327	3,067	1,9717
e	304	328	446	530	4,732	45,006	44,338
m	1,588	1,498	1,578	1,623	3,698	4,973	500
c	5	5	5	5	6	19	3

Table 1: Statistics of the datasets used, where n, e, m and c denote the numbers of nodes, edges, attributes and categories. Waton, Wisin, Cite, UAI is short for Washington, Wisconsin, Citeseer, UAI2010.

We compared NetVAE with two types of network embedding methods. 1) Topology-based methods: DeepWalk [Perozzi *et al.*, 2014], Node2Vec [Grover and Leskovec, 2016] and SDNE [Wang *et al.*, 2016a]; and 2) Methods using both topological and attribute information: TADW [Yang *et al.*, 2015], VGAE [Kipf and Welling, 2016], TriDNR [Pan *et al.*, 2016], SNE [Liao *et al.*, 2018] and ARVGA [Pan *et al.*, 2019]. VGAE and ARVGA are both the VAE-based methods.

For fair comparison, we set the embedding dimension l = 64 for all methods and used the default values for the other parameters of these methods. We also used the Tensorflow deep learning tool with a learning rate of 0.001.

3.2 Node Classification

After network embedding was done, we applied the LibSVM (SVM) and LibLINEAR (LINEAR) software packages in Weka as the classifiers for all methods. We used 10-fold cross-validation to train the classifier. Then, to measure the result, we employed accuracy (AC) [Liu *et al.*, 2012] as the metric. NetVAE performs the best on 6 and 5 of the 7 networks using SVM and LINEAR respectively (Table 2). To be specific, NetVAE on average outperformed the best baseline method (i.e., TADW) by 11.20% using SVM and 6.48% using LINEAR. In particular, NetVAE performs better than the other two VAE-based methods (i.e, VGAE and ARVGA) which also use the topological and attribute information.

Packages	**Methods**	**Datasets**						
		Cornell	**Texas**	**Waton**	**Wisin**	**Cite**	**UAI**	**Pubmed**
SVM	DeepWalk	38.97	49.18	55.30	49.24	52.52	58.69	78.79
	Node2Vec	35.90	50.27	47.47	46.56	61.63	61.95	80.30
	SDNE	50.25	62.29	65.89	56.11	42.14	38.18	39.41
	SNE	48.21	57.92	54.38	59.54	44.74	41.18	78.37
	TriDNR	37.95	48.09	47.01	40.46	54.47	57.74	79.07
	ARVGA	42.56	56.28	58.99	49.26	65.10	31.30	80.64
	TADW	64.10	67.76	59.45	64.50	69.83	68.70	85.37
	VGAE	45.13	55.00	54.38	53.82	68.97	32.21	**85.42**
	NetVAE	**78.46**	**85.25**	**82.03**	**83.59**	**71.62**	**73.65**	83.53/3
LINEAR	DeepWalk	38.46	48.09	53.92	49.62	48.42	60.61	78.36
	Node2Vec	37.95	50.27	45.62	46.94	52.44	60.38	81.08
	SDNE	48.72	66.12	64.51	54.96	41.75	39.65	39.67
	SNE	45.64	59.02	55.76	59.92	44.35	29.87	77.20
	TriDNR	34.87	42.08	43.32	41.60	52.91	57.94	78.40
	ARVGA	41.54	59.02	60.37	56.11	66.71	44.28	80.59
	TADW	61.03	67.76	64.98	67.56	**72.53**	68.50	86.80
	VGAE	45.64	51.91	54.84	54.49	69.25	50.78	**87.81**
	NetVAE	**72.30**	**80.87**	**77.88**	**80.15**	69.95/2	**71.21**	82.17/3

Table 2: Comparison on node classification in terms of AC (%).

3.3 Node Clustering

Similarly, we adopted k-means for node clustering and AC and NMI (normalized mutual information) [Liu *et al.*, 2012] as the metrics. Here NMI was used because it has been widely adopted for node clustering. Our NetVAE method performs the best on 5 of the 7 networks in AC, and 6 of the 7 networks in NMI (Table 3). On average, NetVAE improved upon the best baseline (i.e., TADW) by 6.10% in AC and 16.18% in NMI. This further validate the effectiveness of NetVAE.

Metrics	**Methods**	**Datasets**						
		Cornell	**Texas**	**Waston**	**Wisin**	**Cite**	**UAI**	**Pubmed**
AC	DeepWalk	36.05	46.72	40.76	38.76	36.21	37.58	64.84
	Node2Vec	33.85	47.54	37.33	49.62	40.76	36.74	**66.76**
	SDNE	41.51	**60.02**	50.31	39.17	31.74	22.74	41.66
	SNE	43.08	41.53	48.80	55.50	31.17	30.70	66.13
	TriDNR	38.21	47.54	43.59	43.70	34.44	35.59	59.29
	ARVGA	43.59	41.48	43.66	42.81	43.50	22.95	58.76
	TADW	47.69	59.23	50.30	55.00	55.39	36.19	57.46
	VGAE	36.72	48.35	43.73	43.28	55.46	21.22	58.64
	NetVAE	**56.13**	58.85/3	**60.71**	**66.83**	**61.12**	**38.07**	62.22/4
NMI	DeepWalk	7.06	6.16	5.66	7.65	10.58	34.28	26.55
	Node2Vec	6.65	4.49	2.94	7.86	12.99	33.87	25.02
	SDNE	14.06	24.36	18.78	9.35	11.81	17.92	5.84
	SNE	11.11	12.63	17.43	23.94	7.31	26.56	**26.61**
	TriDNR	7.20	4.32	8.10	6.60	9.59	32.37	19.28
	ARVGA	10.26	7.28	12.60	11.92	22.72	22.00	18.40
	TADW	11.13	10.90	11.63	17.52	31.60	37.92	20.11
	VGAE	7.77	8.52	9.03	9.31	27.93	19.57	17.83
	NetVAE	**33.60**	**35.87**	**37.19**	**44.73**	**34.59**	**41.92**	26.15/3

Table 3: Comparison on node clustering in AC (%) and NMI (%).

Node clustering is also the so-called community detection problem. Here we further compare our method with some existing methods that are specially designed for community detection on attribute networks, since NetVAE is also community-specific. The methods that we compared include Block-LDA [Balasubramanyan and Cohen, 2011], SCI [Wang *et al.*, 2016b] and TLSC [Zhang *et al.*, 2018b]. NetVAE performs the best on 5 and 6 of the 7 networks in terms of AC and NMI respectively (Table 4). On average, NetVAE improved upon the best baseline (i.e., TLSC) by 14.28% in AC and 17.03% in NMI. The two types of comparisons validate that we can derive better embedding by taking into con-

Metrics	Methods	Datasets						
		Cornell	**Texas**	**Waston**	**Wisin**	**Cite**	**UAI**	**Pubmed**
AC	BDA	46.15	54.10	39.17	6.81	24.35	16.04	49.10
	SCI	45.64	62.30	51.15	11.44	27.98	30.94	-
	TLSC	47.69	**65.02**	51.61	13.16	35.74	29.37	61.38
	NetVAE	**56.13**	58.85/3	**60.71**	**66.83**	**61.12**	**38.07**	**62.22**
NMI	BDA	6.81	4.21	3.69	5.01	2.42	5.70	6.58
	SCI	11.44	17.84	12.37	17.03	4.87	24.80	-
	TLSC	13.16	23.92	17.63	16.65	23.16	20.68	19.63
	NetVAE	**33.60**	**35.87**	**37.19**	**44.73**	**34.59**	**41.92**	**26.15**

Table 4: Comparison on community detection in AC (%) and NMI (%). BDA is short for Block-LDA. '-' denotes run time $>$ 100 hours.

sideration network modular structures, and the learning of embedding can enhance community detection.

3.4 Why Our Approach Works

To have a deep understanding on why NetVAE works, we examine two factors that have the greatest impact on the effectiveness and performance of NetVAE.

The Dual Decoder Architecture

In order to evaluate the effectiveness of the new dual decoder architecture, we compare NetVAE with its variant OVAE that uses a shared decoder using the fully connected neural network architecture for reconstruction of both network structures and node attributes. When compared on the node clustering task, NetVAE always outperforms OVAE (Table 5), with an average improvements of 7.67% in AC and 6.52% in NMI. This result revealed that by honoring the difference between network structures and node attributes and their different contributions to forming network modules as well as treating these two types of information in separate reconstruction processes, we are able to have better results.

Metrics	Methods	Datasets						
		Cornell	**Texas**	**Waton**	**Wisin**	**Cite**	**UAI**	**Pubmed**
AC	OVAE	48.28	41.67	45.41	56.51	56.56	**39.17**	**62.63**
	NetVAE	**56.13**	**58.85**	**60.71**	**66.83**	**61.12**	38.07	62.22
NMI	OVAE	28.79	22.81	27.34	33.58	32.66	38.07	25.15
	NetVAE	**33.60**	**35.87**	**37.19**	**44.73**	**34.59**	**41.92**	**26.15**

Table 5: Comparison of NetVAE and OVAE (with shared decoder) on node clustering in terms of AC (%) and NMI (%).

Mechanism for Network Reconstruction

To analyze the factor, we compared the network structures generated from NetVAE and that from its variant OVAE with the shared decoder using neural networks. We reconstruct the network using the method in [Wang *et al.*, 2016a]. This method calculates the cosine similarities between pairs of all of the nodes in the embedding space, and ranks the node pairs in a non-increasing order of similarity. It then selects the top k node pairs as links in the reconstructed network, and uses precision@k (i.e., the proportion of the actual edges in the original network that appear in the k reconstructed edges) as the metric. Here due to space limitation we show the results on Citeseer alone (Figure 2(a)). As shown, the precision@k of NetVAE is always higher than that of OVAE. The result indicates that the network reconstruction depends on network embedding in the VAE framework and a better reconstruction

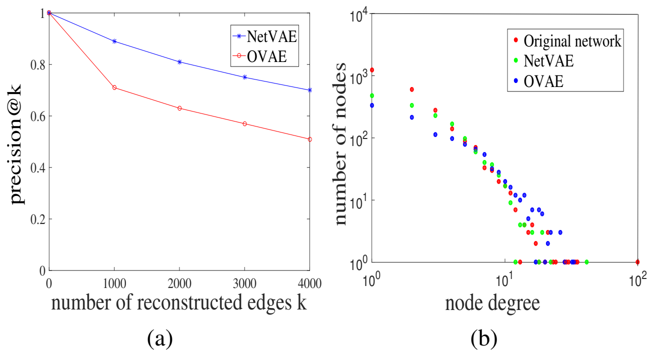

Figure 2: (a) Precision@k of NetVAE and OVAE on Citeseer. (b) Node degree distributions of the original network and the networks reconstructed by NetVAE and OVAE. The x-axes and y-axes are logarithmic, showing that they all follow power-law distributions.

depends on a better embedding. This result may also be due to the way that we take advantage of community-specific coupling relationship among nodes in structure reconstruction.

We further analyzed the effectiveness of the new network generation mechanism in NetVAE by testing if it could better preserve the statistical properties of the original network. As an example, we present the node degree distributions of the original networks and of the networks reconstructed by NetVAE and OVAE (Figure 2(b)). As shown, the node degree distributions of the network reconstructed by NetVAE match better with the original network than that of OVAE.

4 Conclusion

We developed a novel network-specific variational autoencoder, called NetVAE, for embedding of attribute networks. This is the first to distinguish these two types of information of network topology and node attributes under the VAE framework. NetVAE has three immanent features. First, it uses a shared encoder to compress both network structures and node attributes so that a co-training can be done in the model fitting process to perform transfer learning between network structures and node semantics. Second, it introduces a dual decoding scheme to take a good advantage of the difference between network structures and node semantics as in reality these two types of information are often drawn from different sources. The dual decoder has one decoder of fully connected neural network for reconstruction of node attributes and another special neural network that incorporates a block model for exploiting network module structures and incorporates module-specific node coupling relationships. Third, a Gaussian mixture model is introduced between the shared encoder and the dual decoder, which assigns nodes to network modules in the embedding space. The extensive experimental results demonstrated the superior performance of the new approach over the existing methods.

Acknowledgments

This work was supported by the Natural Science Foundation of China (61772361, 61876128).

References

[Balasubramanyan and Cohen, 2011] Ramnath Balasubramanyan and William W Cohen. Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 450–461, Vancouver,Canada, December 2011. SIAM.

[Cui *et al.*, 2018] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, page 1, June 2018.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, California, USA, August 2016. ACM.

[Holland *et al.*, 1983] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[Huang *et al.*, 2017] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 731–739, Cambridge, United Kingdom, February 2017. ACM.

[Jin *et al.*, 2018] Di Jin, Meng Ge, Liang Yang, Dongxiao He, Longbiao Wang, and Weixiong Zhang. Integrative network embedding via deep joint reconstruction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3407–3413, Stockhoim, Sweden, July 2018. Morgan Kaufmann.

[Karrer and Newman, 2011] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, January 2011.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Stat*, 1050(10):1, May 2014.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *Stat*, 1050:21, November 2016.

[Li *et al.*, 2017] Hang Li, Haozheng Wang, Zhenglu Yang, and Haochen Liu. Effective representing of information network by variational autoencoder. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2103–2109, Melbourne, Australia, August 2017. Morgan Kaufmann.

[Liao *et al.*, 2018] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270, 2018.

[Liu *et al.*, 2012] Haifeng Liu, Zhaohui Wu, Xuelong Li, Deng Cai, and Thomas S Huang. Constrained nonnegative matrix factorization for image representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1299–1311, July 2012.

[Pan *et al.*, 2016] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. *Network*, 11(9):12, February 2016.

[Pan *et al.*, 2019] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, page 1802.04407, January 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, New York, USA, August 2014. ACM.

[Tao *et al.*, 2019] Zhiqiang Tao, Hongfu Liu, Sheng Li, Zhengming Ding, and Yun Fu. Marginalized multiview ensemble clustering. *IEEE transactions on neural networks and learning systems*, pages 1–12, 2019.

[Wang *et al.*, 2016a] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, California, USA, August 2016. ACM.

[Wang *et al.*, 2016b] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. Semantic community identification in large attribute networks. In *30th AAAI Conference on Artificial Intelligence*, pages 2103–2109, Arizona, USA, February 2016. AAAI.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *24th International Joint Conference on Artificial Intelligence*, pages 731–739, Arizona, USA, June 2015. ACM.

[Yang *et al.*, 2016] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Max-margin deepwalk: Discriminative learning of network representation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3889–3895, New York, USA, July 2016. Morgan Kaufmann.

[Zhang *et al.*, 2018a] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, pages 1 – 1, June 2018.

[Zhang *et al.*, 2018b] Ge Zhang, Di Jin, Jian Gao, Pengfei Jiao, Francoise Fogelman-Soulié, and Xin Huang. Finding communities with hierarchical semantics by distinguishing general and specialized topics. In *30th AAAI Conference on Artificial Intelligence*, pages 3648–3654, Louisiana, USA, February 2018. AAAI.

[Zong *et al.*, 2018] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. February 2018.