VAE Supervised DNGR: Discriminative Learning of Network Representation

Man Wu

Abstract

The network can be seen everywhere in our daily life and the applications of network structure are really widespread, especially the network systems involving users. Due to the particularity of the network structure, it is challenging to deal with network relationships (finding a suitable network structure representations). I have implemented a recently proposed methods DNGR(Cao et al., 2016) to learn an effective low dimensional representation on the Wiki dataset in the previous experiments. This time, in this project, I chose to add label information (making the learning model become supervised) and use variational auto-encoder to training data, then compare different results and find the best model. I proposed a model combining the above two ideas called the Variational Auto Encoder Supervised DNGR (VAESDNGR). Various evaluation criterion has been used to assess the classification performances and the t-distributed stochastic neighbor embedding (t-SNE) method was used to make the results visualization. Through the experiments on the limited Wiki dataset, I concluded that the three improved models perform better than baseline and VAESDNGR performs best among these four.

1. Introduction

Network is the word describes the complex system and exist in our daily life everywhere. For example, the crossing roads and connected cities are traffic network; the cells and blood consist our body network; the theorems and structure make our knowledge network; kinds of people and link messages between persons are our social networks. As the dramatic development of online-social networks like Facebook, Twitter, WeChat and Weibo, the huge data scale and complex structure requires more efficient computing technique and better methods to represent it.

The most significant feature of the network is that there are the links between nodes, which means the sample nodes in the network are not independent. When you represent the networks, both the nodes information (like name, label, types) and link information should be involved. Initially, we use graph G = (V, E), where V is a vertex set representing the nodes in a network and E is an edge set representing the relationships among the nodes(Yang et al., 2015). However, traditional spectrum-based embedding, optimization, probabilistic network representation algorithms can not meet the requirement of the efficiency and accuracy in the age of big data. At that time, machine learning algorithm provides model and tools for this problem. Since the ICLR (International Conference on Learning Representations) held in 2013, representation learning has become an important problem in the world.

Network representation learning includes kinds of methods, such as Work2Vec: a group of related models which used to produce word embeddings(Mikolov et al., 2013), DeepWalk (Online Learning of Social Representations), node2vec, LINE (Large-scale information network embedding). DeepWalk is a kind of deep learning methods, using random walk conduct sequence data and get word vector by skip gram, DeepWalk performs better than others while LINE aims to embed large-scale network to low-dimensional vector space.

In the previous project, I have implemented the model called DNGR (Deep Neural Graph Representation) and used the wiki dataset in the baseline experiments and found that two hidden layers in DNGR performs best on the Wiki dataset. In this project, I will still use the previous methods to get the high-dimensional networks node embedding. The difference is that I use Variational auto-encoder here to get the highly abstract and low dimensional features for each node. In my previous related project, the learning model is unsupervised. Normally, supervised learning performs better than unsupervised ones, so this time I want to catch the idea to add label information, making it become a supervised learning model(SDNGR). This is the first thing I mainly to do, the second change is auto-encoder, I used the variational auto-encoder to the model this time (VAEDNGR). After that, I proposed a model both using supervised learning and variational auto-encoder (VAESDNGR). In the experiments, I compared the results of four models (DNGR, SDNGR, VAEDNGR, VAESDNGR) in the fixed number of hidden layers and the different proportion of training samples, the results show that the three improved models perform better than baseline and VAESDNGR performs best among these four.

The report will first describe the new methods in this project and compare them to the methods in the previous project I used. And then the entire process of my experiments will be described, including dataset, results, visualization, analysis, and discussion. After that, related work will tell you the background knowledge of network representation and some popular methods has been used widely in network embedding area, most importantly, why I choose DNGR and auto-encoder. Finally, I will discuss the future work and evaluate my project, in addition, a conclusion will be represented.

2. Methods

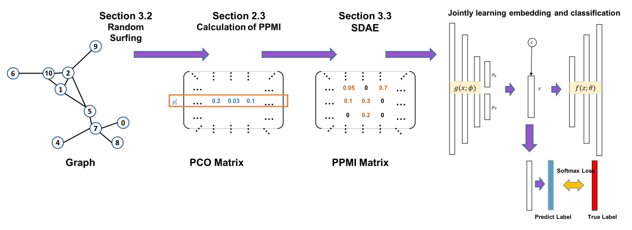

Figure 1. The network structure we used in this experiment

In this project, I still use the previous methods to get the high-dimensional networks node embedding. The difference is that I use Variational auto-encoder here to get the highly abstract and low dimensional features for each node. And then applied a Softmax layer to enforce the labels effective in such a supervised setting. The entire models structure I used here is illustrated in Figure 1. First, I used Random Surfing and PPMI to get the high-dimensional representation of network node structure (a detailed description of these methods is in my previous project: "A Study of Neural Networks for Network Representation"), and then I use the variational auto-encoder to learn highly abstract features distributions from the high dimensional features obtained by previous steps. In addition, I used a Softmax layer to dynamically refine the representation learned by variational auto-encoder. Finally, I still used the same criteria (e.g. micro, macro) in previous experiments to evaluate the network embedding quality. In particular, the evaluation is based on the performance of the task of node classification. Network embedding does have many other application scenarios, such as node clustering, link prediction, and network visualization, which I have mentioned before.

2.1. Variational Auto-encoder

2.1.1. Auto-encoder

Auto-encoder is a special kind of neural networks with some constraint on its layer, and the outputs of auto-encoder are also somewhat different from the general neural network. To begin with, auto-encoder is a kind of unsupervised learning, but some people still prefer to reckon auto-encoder as a kind of "semi-supervised" learning, or self-supervised, since its aim is to reconstruct the inputs at the end, so that it is supervised by itself. What are the constraints of this kind of neural network? That is, it has a "bottleneck", which indicates its narrow-hidden layer which has fewer units than the previous(encoder) and later(decoder) layers. A simple auto-encoder structure can be illustrated by Figure 2

To be more specific, the standard auto-encoder is formed by two processes, which are known as encoding and decoding processes. These two processes can be explicitly

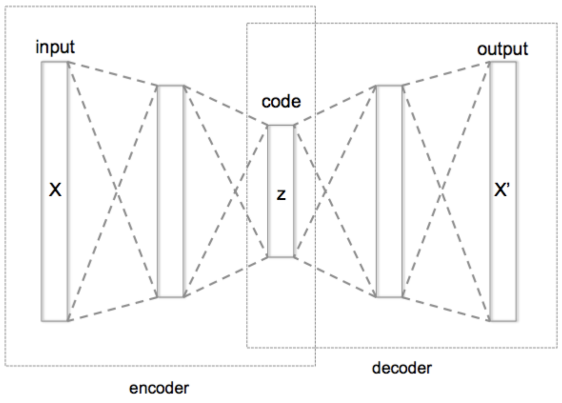

Figure 2. The structure of a simple auto-encoder network

represented by:

$$h = g^{(1)}(W^{(1)}x + b^{(1)}), \tag{1}$$

$$f = g^{(2)}(W^{(2)}x + b^{(2)}), \tag{2}$$

Here, the first equation above indicates the encoder, where the x is the inputs data and the h is the transformed version of x, which can be reckoned as a compressed one so that h is a relatively low dimensional representation of x. g is a point-wise activation function. In practice, Relu(Jarrett et al., 2009) is always preferred here, which may have something to do with the decent gradient its possess when doing back propagation through the network. The second equation shows the decoding process. The transformed values h from the previous encoding process is used as inputs to feed to this decoder. Above all, its aim is to make f, the output of the decoder, as close to x, the original inputs of auto-encoder, as possible. Therefore, the loss function of this neural network is formed by a reconstruction loss term.

Generally, auto-encoder can be used as a compression algorithm, which first takes the latent representation of the data as the compressed version of its, and then using decoding process to reconstruct the original data. Obviously, this process is lossy, the learned representation may fail to fully reconstruct the original data. In nature, it can learn features to initialize a supervised model. Besides, data denoising and dimensionality reduction for the sake of visualization of high dimensional data are two active application of auto-encoder now.

2.1.2. Kullback leibler(KL) divergence

Before fully expand the detail of VAE, I would like to first introduce a crucial concept Kullback Leibler(KL) divergence, which is a measurement that can quantify the discrepancy of two different distributions p and q. KL divergence also has another name called cross entropy, which is more commonly used. Actually, the name of "entropy" comes from information theory. It is used to describe the degree of ordering of a system. The more ordered a system is, the lower its information entropy will be, and vice versa. For a discrete random variable $X = \{x_1, x_2, ..., x_n\}$, and their corresponding possibilities are represented by $p(x_i)(i = 1, 2, ..., n)$. The entropy of random variable X is:

$$H(x) = -\sum_{i=1}^{n} p(x_i) log p(x_i) \tag{3}$$

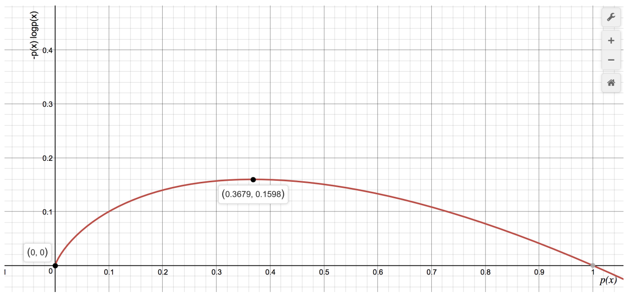

Figure 3. The values of $-p(x)logp(x)$ subject to $p(x) \in [0, 1]$

As we can see in Figure 3. Intuitively, if the random variable X has many fragment values, each of them takes a quite low possibility, and then many of them tend to contribute more to X's entropy H(X) compared to those values with a high probability. Remember here that the sum of the probabilities of different values needs to be added up to 1. Rewritten the entropy into $H(X) = E[-log(p(x))]$ may allow us to understand it from another level that why values with small possibilities will contribute more.

Suppose we have two different distributions, p(z) and q(z), the cross-entropy of p for q is defined as:

$$D_{KL}(p||q) = \int p(z) log \frac{p(z)}{q(z)} dz. \tag{4}$$

It can measure the "distance" between these two distributions to a certain extent. We should be cautious here that the KL divergence does not satisfy the symmetric criteria, which means that $D_{KL}(p||q) \neq D_{KL}(q||p)$. KL divergence is non-negative and only if two distribution p and q are completely the same and then the $D_{KL}(p||q)$ will be zero. Therefore, KL divergence can efficiently help us to quantify the information lost when using a distribution to approximate another distribution. In general, p is the true distribution, and the q indicated the approximate distribution. This is exactly why I introduce KL divergence here before I dive into the details of VAE.

2.1.3. Variational auto-encoder

In the previous project, I have only applied Auto-encoder to get the low dimensional representation of the node structure information, but the representation learned by it was not so satisfactory. So, I try to refine it and learn a better representation (encoded representation). The variational auto-encoder, which has been introduced by Diederik P Kingma, Max Welling(Kingma & Welling, 2013) in 2014, and has been extensively expanded later by many other researchers in the settings of handwritten digits, faces generating, as well as the scene segmentation for self-driving tasks, may be a reasonable and interesting attempt here.

Variational auto-encoder (VAE for short) is a variant of auto-encoder. The difference between them is that VAE learns a probabilistic distribution for each dimension in latent space rather than a deterministic value for each dimension of latent space in standard auto-encoder. Subsequently, we can sample from the latent distribution and use it to generate new inputs. Jeremy gives a very vivid example illustrating what latent variables are in the VAE. He claims that the latent variable is a compact, high-level representation of the inputs data. In their particular task setting, they found that the latent variable can be some characteristics that describe the font, such as bold and italics. They used only 40 latent dimensions to reconstruct the glyph in $64 \times 64 = 4,096$ pixels. In addition, this network is also able to generate some different fonts that have never seen in training examples. In other words, the VAE is essentially a generative model. It not only has the ability to reconstruct the original input, but also can relatively comprehensively learn the high-level features from the original data and generate new ones.

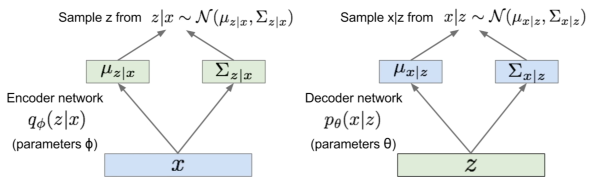

Figure 4. The encoder and decoder network of VAE

The structure of VAE is still the same as the basic auto-encoder. Figure 4 illustrates a simplified VAE structure. It consists of two main parts: encoder and decoder. The difference is that the encoder and decoder network here are probabilistic, so as that the output of the encoder network (or the latent variables) will be a series of probabilistic distribution. I followed the notations in Kingma & Welling (2013) paper, using $q(z|x^{(i)})$ to denote the encoder network, and using $p(x^{(i)}|z)$ to denote the decoder network, and their parameters are ϕ and θ respectively. It is worth mention that the encoder is also known as recognition network or inference network, while the decoder has another name called generation network. Figure 4 is the network structure. Now, our task is to maximize the log data likelihood $log(p_\theta(x^{(i)}))$ (since for auto-encoder, we want the output of decoder network as close to the inputs of encoder network as possible):

$$
\begin{aligned}
log p_\theta(x^{(i)}) &= E_{z\sim q_\phi(z|x^{(i)})}\left[log p_\theta(x^{(i)})\right] \\
&= E_z\left[log \frac{p_\theta(x^{(i)}|z)p_\theta(z)}{p_\theta(z|x^{(i)})}\right](Bayes' Rule) \\
&= E_z\left[log \frac{p_\theta(x^{(i)}|z)p_\theta(z)}{p_\theta(z|x^{(i)})}\frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right] \\
&= E_z\left[log p_\theta(x^{(i)}|z)\right] - E_z\left[log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] \\
&+ E_z\left[log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right] \\
&= E_z\left[log p_\theta(x^{(i)}|z)\right] - D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) \\
&+ D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)}))
\end{aligned}
\tag{5}
$$

Remember here that the p_θ denote the decoder network and the q_ϕ denote the encoder network. The first step in the above derivation process introduces the latent variables z into the expectation and z satisfies $q_\phi(z|x^{(i)})$. The second step is just applying Bayes' Rule, and the next two steps of derivation used the basic property of the log function and a little trick. After understanding the concept of KL divergence, we can easily write the formula of the fourth row into the fifth row above as the KL divergence form. Let's look at what these three terms exactly mean. The first term here is the log probability of $x^{(i)}$ given z for the decoder network and can be simply estimated by sampling. The second term here is the KL divergence between Gaussians for encoder and the prior of latent variables z. As for the last term above, even though the $p_\theta(z|x^{(i)})$ is intractable, but from the discussion in the previous section, we know that KL divergence is at least non-negative. So, the sum of the first two terms in equation (5) can form a lower bound of the log data likelihood that we want to maximize. That is:

$$log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi) \tag{6}$$

$\mathcal{L}(x^{(i)}, \theta, \phi)$ also called "Variational lower bound" (for short ELBO). Therefore, our goal can be converted to maximize this lower bound, which can be formulated by:

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi) \tag{7}$$

Intuitively, the first term forces the outputs of decoder $\hat{X}$ to be as close to inputs X as possible, because it makes training samples have a higher probability. It can be reckoned as a kind of reconstruction loss. The second term forces the latent variables to be close to a Gaussian distribution, which acts as a regularizing force here. Also, we can re-formulate a loss function here for this task. We introduced a hyper parameter β to balance the influence of these two terms. If beta takes a small value, then the mode will degenerate into a standard auto-encoder. Otherwise, increase the value of beta will emphasise the role of KL terms, which ensures the learned latent distribution will have a sufficient variance and make sure the latent space is smooth enough. The influence of beta on the model can be illustrated by Figure 5.

$$L(X, \hat{X}) + \beta \sum_j KL(q_j(Z|X)||\mathbf{N}(0, 1)) \tag{8}$$

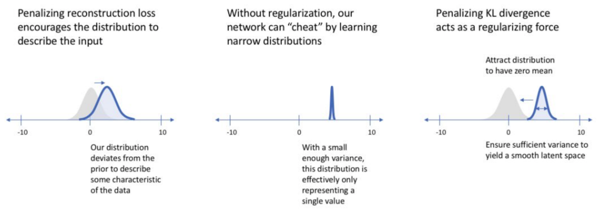

Figure 5. An illustration of the influences of β to the latent representation

Actually, this is only part of the final loss function of my entire model.

2.1.4. GENERATIVE MODEL

There are one interesting thing here. Just like the hottest Generative Adversarial Network(GANs)(Goodfellow et al., 2014) recently, the VAE is also reckoned as a generative model, because we can easily generate new samples, which are similar to the original input data that we feed to the auto-encoder model, from the latent variable distribution we learned. Therefore, I can easily have the idea that whether I can generate some new samples from the limited data, and then use these data together with the original data I collected to retraining my auto-encoder. Will that yield to a better representation? Since my original collected data is obviously not a complete set, and if I add these VAE generated data into training, that may help to compensate these insufficiencies of my data. These kinds of immature idea is just a bit of inspiration from the EM(Expectation Maximization) algorithm in Nigam et al.'s paper, perhaps without a very strict theoretical foundation here. Besides, as for a generative model, the PixelRNN/PixelCNN and GANs may have better performance in terms of the samples it generates, but they do have their backward. Such as PixelCNN is inefficient to generate sequential data, and the training process of GANs is a little bit unstable.

2.2. Softmax

Softmax Regression is a generalization of Logistic regression, which is capable of the classification tasks that have more than 2 classes. When using Softmax, we always use the one-hot encoding approach to construct the target. Suppose that there are K possible classes, and then for each data point $x^{(i)}$, if it has the label of $c \leq K$, then the target of this data point $y^{(i)}$ will be a K-length vector, with all the elements are 0, except the only c^{th} element is set to 1. It is easy to find that $y_k^{(i)}$ is just a Kronecker delta, which is defined by:

$$\delta_{i,j} = \begin{cases} 0 & if i \neq j, \\ 1 & if i = j, \end{cases} \tag{9}$$

We would like to learn a parameter W, that can output the vector. If there are only two classes, the problem can be simply solved by logistic regression. However, there are multiple classes, how can we deal with it. The idea is that we can start by using separate weights w^k for each class k, then the probability of a data point belongs to class k can be formed by $e^{(w^k)^T x}$. Besides, the sum of the probability of a data point belongs to all the classes should add up to 1. So the probabilistic can be normalized by $\sum_{k'} e^{(w^{k'})^T x}$. Then the outputs can be formalized as:

$$f_k = \frac{e^{(w^k)^T x}}{\sum_{k'} e^{(w^{k'})^T x}} \tag{10}$$

Where the f_k indicates the probabilistic of x belongs to class k. Besides, the normally used loss function in Softmax regression is the log likelihood.

2.3. Discriminative Learning

Above all, my task is to get a reasonable and satisfied network embedding from the network structure information (the relationship between network nodes, labels of network nodes) I have. Most network embedding methods known today are learned in an unsupervised way, which has not combined the labels information (sometimes easy to get) in order to get satisfied vertices embedding. In my previous project I was just learning network embedding in a traditional unsupervised way, but this time I try to integrate the label information into the low dimensional vertices embedding I learned. Therefore, the embedding I learned is a discriminative representation of vertices. Actually, it is not trivial to explore how to integrate the labels information into our embedding learning process. Tu et al. has proposed a novel method called Max-Margin Deep Walk (DDMW for short) that combined the labels information into the network representation they learned. This method draws on the idea of support vector machine(SVM), which is a popular method that has heavily used in document classification and handwriting character recognition, as well as other supervised learning settings. I did not reproduce this method due to time. However, because of the experiment setting, this is a method that worth to compare with.

3. Experiments

3.1. Experimental Setup

In order to be consistent with previous experiments, I kept the following experimental configuration unchanged:

Dataset: For consistency with previous experiments, I still use the Wiki(Sen et al., 2008) dataset. As described in previous project, the Wiki dataset provided by openNE toolkit (https://github.com/thunlp/OpenNE) for node classification task, which contains 2405 web pages from 19 categories and 17981 links among them.

Baseline (DNGR): In previous project, I have implemented and explored the network representation task of doing node classification. I modified the state-of-the-art method DNGR (deep neural networks for graph representations) as my baseline, which use the stacked auto-encoder neural network model to extract complex features. I firstly use the PPMI method to preprocess the original adjacent matrix as the input of the auto-encoder neural network. Then I use the auto-encoder neural network for training to get the best feature representation of the reduced-dimension nodes (In this project I still set the reduced nodes to 128, i.e. After training in the auto-encoder neural network, the number of the nodes decreased from 2045 to 128). In the process of training, I try to minimize the standard squared loss of the input and output, which is used to evaluate the performance of training. After training, I use half of the 128 trained nodes to do the classification task. Meanwhile, I get the score of the 'micro', 'macro', 'samples' and 'weighted', which is widely used as the evaluation of the classification performances(Tang & Liu, 2009). I tuned the parameters and tried to optimize the baseline. Finally, I found that the auto-encoder neural network with 2 hidden layers has the lowest loss and performs best in the node classification task.

Parameters: In order to be comparable to the baseline, I set the number of hidden layer of the network in my experiment to 2 and run 400 epochs for training. In addition, for the first stage of the experiments of this project, I still use 50% of the random nodes for training and the remaining 50% for testing.

However, when using t-SNE (Maaten & Hinton, 2008) to visualize the classification performance of my baseline method, the results are not very satisfactory and the nodes are not clearly classified. Thus, I want to explore more methods from different perspectives to outperform the baseline as well as to have the best representation of the Wiki network and a good classification performance.

3.2. Models and Experimental Results

I want to improve my baseline from the following two aspects:

Supervised DNGR (SDNGR) The baseline method DNGR which using the auto-encoder neural network encodes the network structure into vertex representations and is learned in unsupervised form. However, the learned representations usually lack the ability of discrimination when applied to machine learning tasks, such as node classification task. Thus, I want to overcome this challenge and make the DNGR become a supervised learning model. In the experiment, I randomly sample a portion of labeled nodes and take their representations as features for training, and the rest is used for testing. In order to be comparable to the baseline, I set the number of hidden layer of SDNGR to 2. The performance of training loss compared with the baseline is shown in Figure 6 and the comparison of classification accuracy is shown in Table 1.

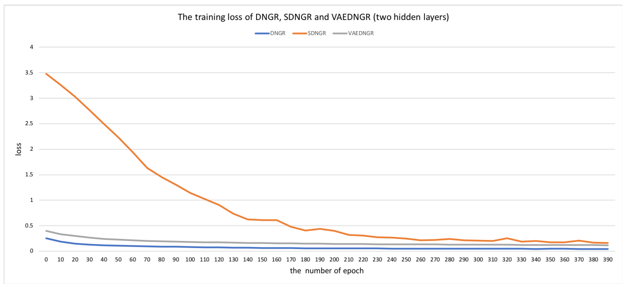

Figure 6. The training loss of DNGR, SDNGR and VAEDNGR(two hidden layers)

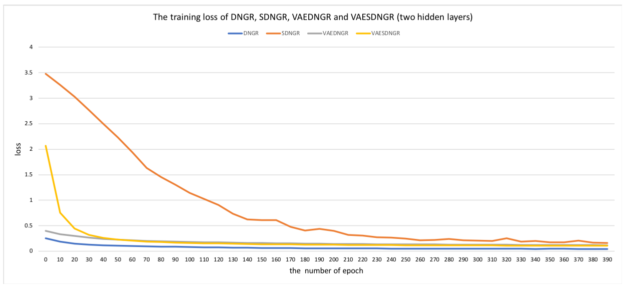

Figure 7. The training loss of DNGR, SDNGR, VAEDNGR and VAESDNGR (two hidden layers)

	Micro	Macro	Samples	Weighted
DNGR	0.6617	0.5132	0.6617	0.6533
SDNGR	0.6641	0.5323	0.6642	0.6586
VAEDNGR	0.6741	0.5381	0.6741	0.6662

Table 1. The comparison of Classification Accuracy of Three models on Five Criteria

Variational AutoEncoder DNGR (VAEDNGR) As introduced in the section of methodology, Variational AutoEncoder is one type of autoencoder, which can not only used for data dimensionality reduction or feature extraction, but is now also being extended to generate models. Based on the idea of improving the baseline, I replaced the autoencoder in the previous experiment with a variational autoencoder. The performance compared with the baseline is also shown in Figure 6 and Table 1 respectively.

Through the Figure 6, we can see that loss of each model (DNGR, SDNGR and VAEDNGR) is decreasing, indicating that the loss function of each model converges. In addition, the specific values of each model loss are not comparable because their function to calculate loss is not the same. Through Table 1, we can see that for each evaluation criterion of classification accuracy, SDNGR and VAEDNGR all outperform the baseline model DNGR.

From the data, we can see that the two methods of adding supervision information and using variational autoencoder in training are both effective, and both are better than the baseline method. Hence, based on the above ideas, I combine these two methods and propose a model of the **Variable Auto Encoder Supervised DNGR (VAESDNGR)**. I assume that this model outperforms the above two models and the baseline. I compared the new model with the baseline and the above two models, as shown in Figure 7 and Table 2. As we can see from Figure 7 and Table 2, the experimental results prove my hypothesis that the VAESDNGR model does perform best among these four models. However, I cannot arbitrarily conclude that the VAESDNGR is the best model, I still need to do other experiments.

	Micro	Macro	Samples	Weighted
DNGR	0.6617	0.5132	0.6617	0.6533
SDNGR	0.6641	0.5323	0.6642	0.6586
VAEDNGR	0.6741	0.5381	0.6741	0.6662
VAEDNGR	0.6899	0.5392	0.6899	0.6840

Table 2. The comparison of Classification Accuracy of Four models on Five Criteria

3.3. Performance on Node Classification

I vary the percentages of training samples p% from 10% to 70% and report the results of different models in Table 3. I choose the "Macro Score" from the five criteria in previous project as the criterion to compare different models. As shown in Table 3, for each method, the accuracy rate improves with the increase of the number of training nodes. Especially for the model VAESDNGR, 30% of training nodes can get relatively good accuracy. The overall performance of model VAESDNGR on node classification is better than the baseline and the other two models.

%p	DNGR	SDNGR	VAEDNGR	VAESDNGR
10	0.4280	0.3973	0.4167	0.3700
30	0.4953	0.4897	0.4859	0.5148
50	0.5132	0.5324	0.5381	0.5392
70	0.5760	0.5400	0.5566	0.5962

Table 3. the Macro Score of five different models on Wiki dataset

3.3.1. Visualization

In order to show classification task of different methods more intuitively, I use t-SNE to visualize the results (Maaten & Hinton, 2008). I compare the three new models with the baseline. The results of the visualization are shown from Figure 8 to 11. From the visual contrast of these models, we can see that the model VAESDNGR has the best performance, its points are clustered best, and the classification has clear boundaries. The second best-performing model is SDNGR. They all outperform the baseline. However, the model VAEDNGR does not outperform the baseline as I expected. From my limited understanding, I assume that the reason may be that when the hidden layer number is 2, the variational autoencoder is not as good as original autoencoder, which may be more suitable for more number of hidden layers.

However, the 2D visualization only plays a supplementary role in the model selection, and the evaluation criteria are still the score of 'micro', 'macro', 'samples' and 'weighted '. Fortunately, the experiments are still validating my hypothesis to some extent. The VAESDNGR model is superior to other models on this limited Wiki dataset, both in the evaluation score and the visualization performance.

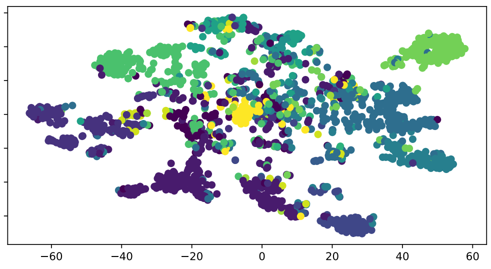

Figure 8. 2D visualization of DNGR (Baseline) (two hidden layers)

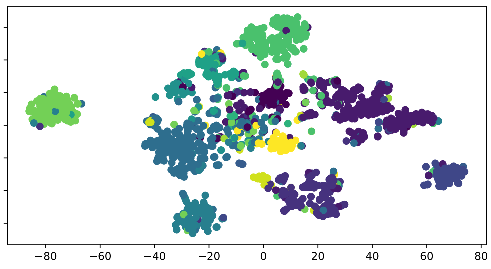

Figure 9. 2D visualization of SDNGR (two hidden layers)

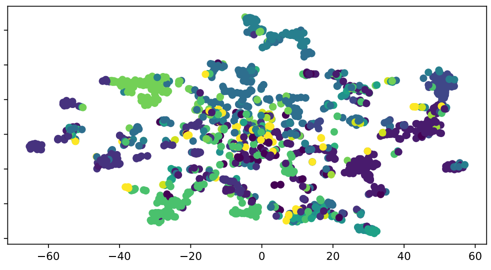

Figure 10. 2D visualization of VAEDNGR (two hidden layers)

4. Related Work

Network is an important way to present the relationship between objects, how to use and present the feature information of network is the key issue in network research.

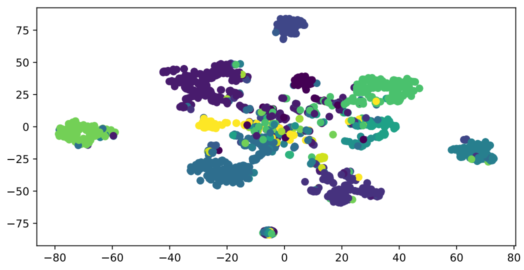

Figure 11. 2D visualization of VAESDNGR (two hidden layers)

Network representation learning (also named network embedding or graph embedding) actually aims to encode network structure into a low-dimensional space ((Tu et al., 2016)). Many models and methods have been proposed, implemented to learn efficient vertex representations. The traditional methods are based on factorization, such as: Locally Linear Embedding(Roweis & Saul, 2000), Laplace Eigenmaps(Belkin & Niyogi, 2002), and Graph Factorization(Ahmed et al., 2013).

However, these methods can not meet the requirement of the developing space of data scale and the complexity of the database. After that, the methods based on machine learning appeared. The Skip-gram model (Tu et al., 2016) proposed in 2013 employed a simple neural network model to learn distributed vectors for words; As the propose of Word2Vec, a word embedding technique in natural language processing and motivated by the Skip- Gram, DeepWalk- a deep learning model was mentioned in the paper "DeepWalk Online Learning of Social Representations" in 2014. DeepWalk has been extensively verified on various network analysis tasks and it represents network only using structure(Perozzi et al., 2014). Another typical network representation learning model needs to mention is LINE(Tang et al., 2015), which is proposed to handle large-scale networks with millions of vertices and billions of edges, also can be simply defined as large-scale data. LINE is a state-of-art algorithm for network representation based on network structure. Besides DeepWalk and LINE, there are also many optimized methods and specific method has been proposed and implemented, for example, Doc2Vec, a paragraph vectors algorithm can embed any piece of text in a distributed vector; TriDNR(Tu et al., 2016), a kind of representation algorithm which can exploit network structure, node content, label information for learning; DNRL is a variant of TriDNR, which only use label information and nodes content, ignoring the network structure information.

In previous project, I have chosen DNGR (deep neural networks for graph representations) to do the nodes classification, DNGR uses the unsupervised stacked auto-encoder neural network model to extract complex features of high dimensional input matrix ((Cao et al., 2016)).However, DeepWalk, LINE, and DNGR are all unsupervised methods as they can only utilize the network information for model learning, without considering any text information augmented with each node and label information provided by the specific task like node classification(Tu et al., 2016).At that point, I want to add label information to make it become supervised learning, and then training it and comparing it to the original ones in this project. In addition, I want to improve the auto-encoder model. So I choose Variational auto-encoder, which is a generating model proposed by Kingma&Welling in 2014(Kingma & Welling, 2013).

5. Discussion

I have implemented the Supervised DNGR model and Variable Auto-encoder (VAE) and combined the classification results in the different number of layers and proportion of samples in four models. The results show that VAESDNGR performs best and 30% of training nodes can get relatively good node classification accuracy.

Supervised learning can be divided into the discriminative model and generative model. VAE is an efficient and popular generative model and always show brilliant performance. In this project, I have implemented VAE but not use it to generate samples which not exist in the sample. Maybe I can use it to generate more samples later and make up for the lack of the sample.

Despite the generative model, I can also add nodes content information in the future. As I mentioned before, the model I used (DNGP) just utilize the network information for model learning. I have added label information for each node, so I can also add content information in the following research to test the model, how the label or content information will influence the classification performance.

In addition, I can also test the model on more different datasets to prove universality. For example, large-scale dataset, multi-label nodes dataset.

6. Conclusion

In this project, I explored and studied methods and models from two aspects to improve my baseline, which is modifying baseline to a supervised learning model and applying variational auto-encoder to the original network. The two models all outperform the baseline on the accuracy of classification. Thus, I combine these two methods and propose a model of the Variable Auto Encoder Supervised DNGR (VAESDNGR). I assume that this model outperforms the above two models and the baseline. From the experiments, the experimental results and the 2D visualization performance prove my hypothesis on the limited wiki dataset. However, I cannot arbitrarily conclude that the VAESDNGR is the best model, I still need to do other experiments such as testing the model on larger datasets to prove universality. Fortunately, the models I put forward in the experiments still exceeded the baseline method.

My project code (using Tensorflow) is available on GitHub, you can run it and the link shows there: https://github.com/Mandy001/VAE-Supervised-DNGR

References

Ahmed, Amr, Shervashidze, Nino, Narayanamurthy, Shravan, Josifovski, Vanja, and Smola, Alexander J. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 37–48. ACM, 2013.

Belkin, Mikhail and Niyogi, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591, 2002.

Cao, Shaosheng, Lu, Wei, and Xu, Qiongkai. Deep neural networks for learning graph representations. In *AAAI*, pp. 1145–1152, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Jarrett, Kevin, Kavukcuoglu, Koray, LeCun, Yann, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153. IEEE, 2009.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Nigam, Kamal, McCallum, Andrew Kachites, Thrun, Sebastian, and Mitchell, Tom. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

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

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

Sen, Prithviraj, Namata, Galileo, Bilgic, Mustafa, Getoor, Lise, Galligher, Brian, and Eliassi-Rad, Tina. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

Tang, Jian, Qu, Meng, Wang, Mingzhe, Zhang, Ming, Yan, Jun, and Mei, Qiaozhu. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

Tang, Lei and Liu, Huan. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 817–826. ACM, 2009.

Tu, Cunchao, Zhang, Weicheng, Liu, Zhiyuan, and Sun, Maosong. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, pp. 3889–3895, 2016.

Yang, Cheng, Liu, Zhiyuan, Zhao, Deli, Sun, Maosong, and Chang, Edward Y. Network representation learning with rich text information. In *IJCAI*, pp. 2111–2117, 2015.