DNGR: DEEP NEURAL NETWORKS FOR LEARNING GRAPH REPRESENTATIONS

Apoorva Vinod Gorur

Features Engineering

https://github.com/apoorvavinod/DNGR

Table of Contents

1. Introduction

The introduction of word embeddings[1] by T. Mikolov has influenced many areas of study in Data Science. Graph and Network analysis has also adapted this concept of embeddings to generate latent representations for nodes in a graph. The DeepWalk [2] was the first model to implement this technique. The DNGR [3] model makes use of methodologies expressed by Levy and Goldberg [4] and Deep neural networks to generate embeddings for graph nodes.

In the following subsections, the report talks about methodologies of contemporary models and how the DNGR model utilizes matrix factorization and neural network techniques to generate embeddings for nodes in a weighted graph.

2. Contemporary models

This section talks about previous models that have had success with generating node embeddings. Section 3 uses this as reference to denote how the DNGR model differs from these models.

The DeepWalk model proposed by B. Perozzi makes use of truncated random walks and word2vec to generate node embeddings. DeepWalk generates random walks for each node by randomly sampling the node's neighbors and truncating the walk after a certain length η is reached. This procedure is repeated for each node γ times. The random walks are treated as sentences and fed to the word2vec model. The word2vec model builds a vocabulary and a co-occurrence matrix depending on the window size(w). It then constructs embeddings for each node by treating them as if they were individual words.

GENE [5] incorporates a similar strategy to DeepWalk's random walk but considers the node's group information as well. Making the group information a part of the node embeddings will enhance classification and clustering tasks. For each walk W_{vi} where V_i is the root node, GENE samples a group label 'g' from the groups that V_i belongs to and labels it to that walk. GENE also considers each group 'g' and generates walks from the set of nodes V that belong to the group 'g'. These walks are then concatenated and used for generating embeddings.

Several models have also been developed based on matrix factorization methods. These methods are used on the co-occurrence matrix generated for the graph. A PMI (Pointwise Mutual Information) matrix is computed using the co-occurrence matrix to serve as embeddings. Levy and Goldberg also propose the use of PPMI (Positive Pointwise Mutual Information) matrix and performing dimension reduction using SVD (Singular Value Decomposition) to generate embeddings. Making use of SVD gives a linear projection; the DNGR model aims to replace SVD with non-linear projection methods.

3. DNGR Model

The DNGR model computes node embeddings in 3 steps. They are discussed in the following sub-sections. DNGR uses random surfing model to construct a co-occurrence matrix. This matrix is used to compute a PPMI matrix which is then fed to a stacked denoising autoencoder to generate latent representations for the nodes.

3.1 Random Surfing

The random surfing model constructs a transition matrix A that has transition probabilities between different nodes in the graph. Each row vector of the transition matrix is of the form p_k where the j^{th} entry corresponds to the probability of the reaching the j^{th} vertex after k steps of transition. The random surfing model also considers the probability of a restart at each step. There is a probability α that the random surfing will choose the next node and a probability 1 - α that it will return to the original node and restart. This is summarized using the recurrence relation: $p_k = \alpha \cdot p_{k-1} A + (1-\alpha) \cdot p_0$ where P_0 is the initial one-hot vector of a node V_i with the i^{th} entry being 1 and other entries being 0. The transition matrix A captures the importance of node similarity with the relative distances between the nodes.

The random surfing model has advantages over the random walk method used in DeepWalk. The generation of random walks is a relatively slow process even with the use of multiple CPU threads. The DNGR model computes the co-occurrence statistics directly with the use of transition matrix. The extent of capturing node similarity information in DeepWalk is limited by the maximum length of each random walk. Nodes that are outside the range miss out during the computation of co-occurrence matrix. This is not an issue for the random surfing model of DNGR. DeepWalk has the hyper-parameters η (walk length) and γ (walks per node) which have to be configured properly for different datasets.

3.2 PPMI matrix

In Natural Language Processing, although using a co-occurrence matrix as embeddings makes sense, it isn't the best way of measuring contextual information since stop-words heavily influence the representations. A PMI (Pointwise Mutual Information) matrix computed from the co-occurrence matrix is thus used to address this issue. The effectiveness has been seen to further improve with the use of a PPMI (Positive Pointwise Mutual Information) matrix instead of PMI. For a word *w* and context *c*, the PPMI value is calculated using the formula:

$$\mathrm{PPMI}(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0\right)$$

The transition matrix A generated by the random surfing model is used to construct a Positive Pointwise Mutual Information(PPMI) matrix. For any two nodes, the PPMI values answers the question whether the two nodes tend to occur more together or more independently.

3.3 Stacked Denoising Autoencoder

Levy and Goldberg [3] propose the use of SVD for dimensionality reduction after computing PMI matrix for a text corpus. Although SVD is an effective tool for dimensionality reduction, it captures only linear relationships between the two vectors. DNGR proposes the use of neural networks instead of SVD to capture a non-linear mapping between the nodes. DNGR makes use of Stacked Denoising Autoencoders to compute low-dimensional latent representations for the nodes.

Autoencoders are trained to propagate its input through a feed forward network and reconstruct the hidden layer outputs back to the original input. This creates an effective compression procedure where the low-dimensional hidden layer outputs can be de-compressed to give a lossless reproduction of the input. Figure 1 below shows a visual representation of a stacked denoising autoencoder.

The embedding for a node in the PPMI matrix will be a row vector of dimension 1 x |V| where V is the set of all nodes in the graph. This is fed to the autoencoder to reduce the dimension up to 1 x *d* where *d* can be set to any desirable value (common values used are 64, 128, 256, 512). Denoising autoencoders are different from conventional autoencoders. The introduction of noise in the input before feeding it to the network allows the autoencoder to learn to remove noise from the input. Stacking these autoencoders together improves the efficiency of the latent representations.

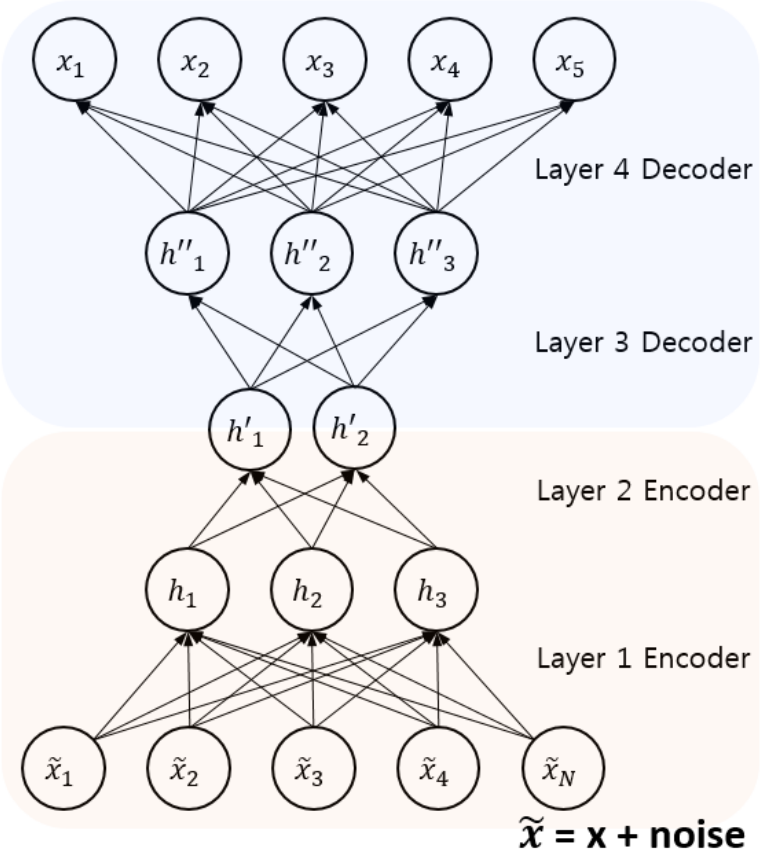

Figure 1 Stacked Denoising autoencoder. Img Source: https://wikidocs.net/3413

4. Experiments

The embeddings produced by DNGR is evaluated by computing Normalized Mutual Information (NMI) [6] for clustering tasks. A popular tool called 't-SNE'[7] is used to map the embeddings into 2D space for visualization. This will help evaluate how efficiently the DNGR model makes distinction between different types of nodes. The dataset used for the task is described in the following subsection.

4.1 Dataset

The dataset used for evaluation is the 20-NewsGroup dataset. It contains around 20,000 documents related to 20 different topics. TF-IDF vectors are computed for each document. These vectors are used as nodes in the graph and the cosine-similarity between the documents are the edges between them. The experiments are conducted on three subgraph of the whole dataset which are represented by 3NG, 6NG and 9NG. The groups contained in each graph is listed below. The evaluation uses the same convention used by Tian et al. (2014) [8]

- **3-NG**: *corp.graphics, rec.sport.baseball, talk.politics.guns*
- **6-NG**: *alt.atheism, comp.sys.mac.hardware, rec.motorcycles, rec.sport.hockey, soc.religion.christian, talk.religion.misc*
- **9-NG**: *talk.politics.mideast, talk.politics.misc, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, sci.electronics, sci.crypt, sci.med, sci.space, misc.forsale*

4.2 Results

The evaluation of the models were done with the NMI(%). Table 1 shows the performance of DNGR against other models. The NMI values for SVD, PPMI, DeepWalk and SGNS were taken from the original DNGR paper. The NMI values for the DNGR are based on my implementation. DNGR outperforms other models for the 3NG and 9NG graphs and comes close to the top for the 6NG graph.

Model	NMI (%)		
	3NG	6NG	9NG
DNGR (α = 0.98)	74.78	64.38	55.6
DNGR (α = 0.95)	74.51	65.23	54.9
DNGR (α = 1.0)	71.03	64.84	52.1
SVD (α = 0.98)	62.95	66.18	52.9
SVD (α = 0.95)	62.83	65.33	50.0
SVD (α = 1.0)	60.27	63.52	53.52
PPMI	62.84	66.52	52.3
DeepWalk (d = 128)	55.83	58.11	46.86
DeepWalk (d = 256)	56.34	58.4	46.62
DeepWalk (d = 512)	59.64	59.82	46.29
SGNS (d = 128)	64.34	62.99	48.14
SGNS (d = 256)	62.56	63.25	48.1
SGNS (d = 512)	63.97	63.16	49.04

Table 1 NMI values on 20-NewsGroup dataset

Figure 2 below shows the visualization of the generated embeddings for the 3NG graph using t-SNE tool. It shows clear distinction between the 3 groups. The 3 different colors correspond to the 3 groups present in the 3NG graph.

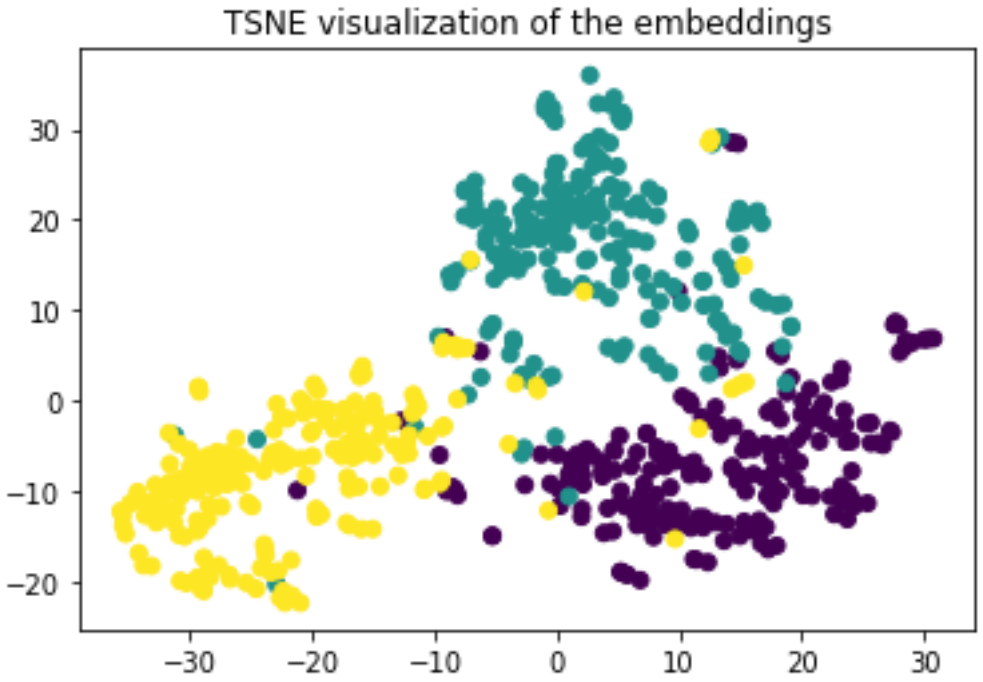

Figure 2 t-SNE visualization of embeddings for 3NG

5. Conclusion

The results reported in section 4 clearly show that DNGR performs better than its contemporaries in most cases. The original paper of DNGR performs other tasks such as word similarity and data visualization to compare performance and it can be seen that DNGR outperforms other models. The DNGR model is proof that using deep neural networks is helpful in mapping non-linear relationship in the data. Further improvements can include compatibility with heterogeneous graphs and signed graphs as well.

6. References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781, 2013.

[2] B. Perozzi, A. Rami and S. Skiena. DeepWalk: Online learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 701-710, New York, NY, USA, 2014. ACM.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1145–1152. AAAI Press, 2016.

[4] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems (NIPS)*, pages 2177–2185, 2014.

[5] Jifan Chen, Qi Zhang, and Xuanjing Huang. Incorporate group information to enhance network embedding. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1901–1904. ACM, 2016.

[6] Strehl, A.; Ghosh, J.; and Mooney, R. 2000. Impact of similarity measures on web-page clustering. In AAAI, 58–64.

[7] Van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-sne. JMLR 9(2579-2605):85.

[8] Tian, F.; Gao, B.; Cui, Q.; Chen, E.; and Liu, T.-Y. 2014. Learning deep representations for graph clustering. In AAAI.