Application of Deep Learning in Graphic areas: A Review of Network Embedding

Ziyao Li [1]

Abstract

Different Network Embedding approaches are introduced in this review. Traditional methods in embedding tasks are Principal Component Analysis and Multi-Dimensional Scaling. Isomap and Laplacian Eigenmap further expanded these methods to network spaces. *Word2vec* introduced Skip-gram in Natural Language Processing, which became a powerful tool in network analysis in *DeepWalk* and following work. Other methods dealing with heterogeneity, using MF-based methods, convolutional network methods, autoencoder methods and adversarial methods are also briefly introduced in this review.

1. Introduction

Network Embedding, or Network Representation Learning, has long been an important and ubiquitous task in graphic, or network, machine learning areas. The main task of Network Embedding is a process transforming an individual node in a graph to a vector in Euclidean Space with a fixed dimension, during which the topology information and other types of information, such as node labels, from the original network is kept. The embedded vectors of each node from the embedding task is then used as inputs in different downstream tasks, such as link prediction and node label prediction. These tasks over real-world networks can lead to various applications, including drug design, recommendation systems, and realizing all kinds of data mining purposes, all of which make the Network Embedding task crucial and popularly studied.

Primitive studies over Network Embedding task regarded this representation purpose as a dimension reduction task. Implementing traditional dimension reduction algorithms on this network-specific task without considering the subtle characteristics of real-world networks was almost intuitive. Naive attempts took the corresponding rows in the adjacent matrix of individual nodes as inputs, and then adopted dimension reduction methods such as Principal Component Analysis (PCA) or Multi-Dimensional Scaling (MDS) on these inputs. Later, more dimensional algorithms were proposed to solve the non-linearity problems, such as (Tenenbaum et al., 2000). Nonetheless, these methods are not designed specifically for network data, and only adopting Pointwise Mutual Information (PMI), i.e. the rows in the adjacent matrix, seems insufficient in describing global structures of networks.

These attempts, however, were actually not in the range of deep learning areas. The pioneer and the most impressive work leveraging neural networks in embedding tasks emerged in Natural Language Processing area-the famous *word2vec* (Mikolov et al., 2013). The simplicity to implement, the efficiency in computation, and the excellent performance immediately made this work among the most prevalent ones in coming years. Embed-

[1]SID: 1500017776; Data Science and Big Data Technology Track, Yuanpei College, Peking University..

This paper is a mid-term assignment of course Deep Learning, 2018 *of Prof. Zhihua Zhang, Peking University.*

ding soon became the hottest topic. The ideas of *word2vec* were implemented in all kinds of areas, including networks. Countless researches focusing on network embedding swarms to all kinds of conferences, including SIGKDD, ICLR, NIPS and IJCAI.

2. *Isomap*: an Origin and Intuition

Technically, *Isomap* (Tenenbaum et al., 2000) was not a specifically designed method with regard to network data. Instead, it was implemented by translating data in Euclidean space into a form of network data, and then adopting a representation technique similar to Multi-Dimensional Scaling. However, instead of keeping the original distance matrix as Multi-Dimensional Scaling did, Isomap used the sums of sequences of "short hops" between neighboring points to approximate distances between faraway points. Such distances were calculated from a shortest path algorithm, with distances between neighborhood points calculated directly (typically in a Euclidean measurement). Algorithm 1 is the original algorithm proposed in this paper.

Although seems primitive, this method performs significantly better than original dimension reduction method, especially when non-linearity, or a low-dimensional manifold emerges in the data points. Besides, it correctly shows the dimension of such manifolds. Its application in network data is intuitive however naive, since applying Isomap on a network simply equals to measure the distance between two nodes with the shortest path connecting them. Its ingeniousness in transforming Euclidean distance into network distance cease to militate. However, it was one of the most important network embedding approaches then.

(Belkin & Niyogi, 2000) adopted the very idea proposed in (Tenenbaum et al., 2000), but instead of calculating the whole distance matrix D_G, it solved a sparse eigenvalue problem over the original neighborhood distance matrix,

$$Ly = \lambda Dy$$

Algorithm 1 Isomap

Construct neighborhood graph: Define the graph G over all data points by connecting points i and j if [as measured by $d_X(i,j)$] they are closer than ϵ (ϵ-Isomap), or if i is one of the K nearest neighbors of j (K-Isomap). Set edge lengths equal to $d_X(i,j)$.

Compute shortest paths: Initialize $d_G(i,j) = d_X(i,j)$ if i,j are linked by an edge; $d_G(i,j) = \infty$ otherwise. Then for each value of $k = 1, 2, \cdots, N$ in turn, replace all entries $d_G(i,j)$ by $\min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$. The matrix of final values $D_G = \{d_G(i,j)\}$ will contain the shortest path distances between all pairs of points in G. (Floyd Algorithm)

Construct d-dimensional embedding: Let λ_p be the p-th eigenvalue (in decreasing order) of the matrix $\tau(D_G)$, and ν_p^i be the i-th component of the p-th eigenvector. Then set the p-th component of the d-dimensional coordinate vector y_i equal to $\sqrt{\lambda_p \nu_p^i}$.

where

$$\begin{aligned} D_{ii} &= \sum_j W_j i, \\ D_{ij} &= 0, \quad i \neq j, \\ L &= D - W. \end{aligned}$$

Here, W is a weight matrix defined over the distances calculated in the Isomap. It can be estimated through a Gaussian kernel or simply an 0-1 kernel. L is a Laplacian operator defined over W. Since it no longer used the Floyd Algorithm, which can be very computationally expensive as the scale of problem increases, it performed better on network data.

3. From *word2vec* to *DeepWalk* and *everything2vec*

Traditional embedding methods mentioned above cannot perfectly accomplish the Network Embedding task, since the eigenvalue problem itself is

rather complex when it comes to massive-size networks. Similar situation lies in Natural Language Processing areas, where the vocabulary can be extensively large, and sentences or phrases, in representations as arcs in graphs, are massive as well. (Mikolov et al., 2013) along with its simplicity and efficiency therefore became a milestone in embedding areas.

Word2vec (Mikolov et al., 2013) proposed two language models to embed words in their context namely Continuous Bag-of-Words (CBOW) and Skip-gram. CBOW takes the context of each words as inputs, predicting the central word, while Skip-gram takes the central word as input predicting the context. Only a single layer neural network is used to reduce the computational complexity arising in the origin NNLM (Bengio et al., 2003) model. Figure 1 shows the architectures proposed in this paper.

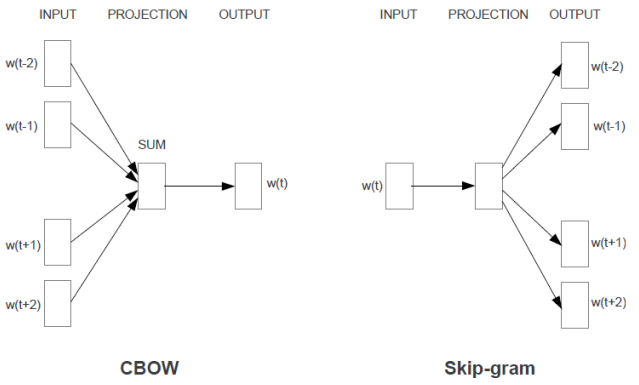

Figure 1. Architectures of CBOW and Skip-gram models proposed in *word2vec*.

Simply by removing layers and activations and a slight adjustment to modeling structure, *word2vec* became a most prevalent toolkit in NLP areas. What's more, its concepts were continuously and widely extended in all kinds of areas, one of which was *DeepWalk* (Perozzi et al., 2014). *DeepWalk* adopted the very idea in *word2vec* by analogizing nodes and their neighbors as words and their contexts. Therefore, a sentence, in its perspective, is a sequence of words generated by a process continuously accessing words and their most intimate context words, i.e. a random walk process. Such analogy is based on the observation that the distribution of number of vertex occurrences against total number of vertices in random walk is much alike to that of words. A Skip-gram model is implemented with a hierarchical softmax architecture, which transformed the whole vertex set into a randomly established binary tree and predicted the outputs layer by layer to reduce arguments. Figure 2 is the proposed structure of hierarchical softmax.

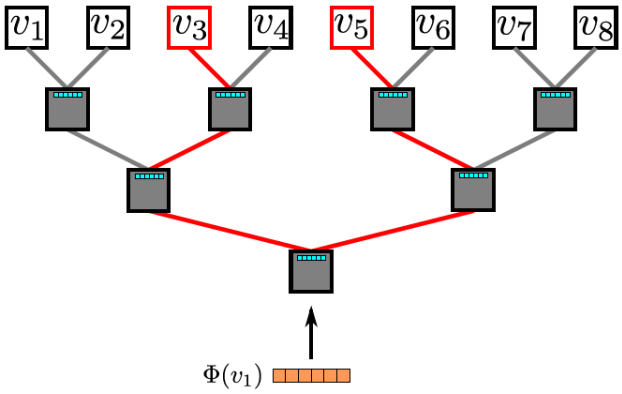

Figure 2. Architecture of hierarchical softmax proposed in *DeepWalk* to implement Skip-gram prediction.

DeepWalk itself was not much extendable for it applied a simulating method, or to say, a real random work process, to collect the sequences ("sentences"). Therefore, it was hard to combine this method to any other optimization purposes or downstream applications in one framework. The best propagation for *DeepWalk* was (Yang et al., 2015). Its primitive motivation was to combine textual information of vertices (typically regarded as labels or descriptions as input information over individual vertices) and the *DeepWalk* structure together, while its most valuable contribution is proving that *DeepWalk* is equivalent to a Matrix Factorization (MF) process. By achieving such, it conveniently combine the *DeepWalk* and textual features over vertices in one.

By describing the original *DeepWalk* process for

each vertex sequence as an optimization problem,

$$\mathcal{L} = \frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-t\le j\le t, j\neq 0}\log Pr(v_{i+j}|v_i),$$

$$Pr(v_j|v_i) = \frac{\exp(c_{v_j}^T r_{v_i})}{\sum_{v\in V} c_v^T r_{v_i}},$$

this paper further used former conclusions over Skip-gram with negative sampling (SGNS) (Levy & Goldberg, 2014) and a comparison between limited-length random walk process in *DeepWalk* and a *PageRank* (Page et al., 1999) process to derive that

$$M_{ij} = \log\frac{N(v_i, c_j)}{N(v_i)},$$

$$\frac{N(v_i, c_j)}{N(v_i)} = \frac{[e_i(A + A^2 + \cdots + A^t)]_j}{t},$$

where A is the transition matrix in *PageRank*. Furthermore, denote the embedding matrix $W = [r_{v_1}, r_{v_2}, \cdots, r_{v_{|V|}}] \in \mathbf{R}^{(k\times b)}$ for central words and $V = [c_{v_1}, c_{v_2}, \cdots, c_{v_{|V_c|}}] \in \mathbf{R}^{(k\times d)}$ for contextual words, then the original embedding problem became an optimization problem over W and H, namely

$$\min_{W,H}\sum_{i,j\in\Omega}(M_{ij} - (W^T H)_{ij})^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2).$$

Combining point-wise features, denoted by $X \in \mathbf{R}^{(p_x\times b)}$ and $Y \in \mathbf{R}^{(p_y\times d)}$, the problem can be described as

$$\min_{W,H} \sum_{i,j\in\Omega}(M_{ij} - (X^T W^T H Y)_{ij})^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2),$$

where $W \in \mathbf{R}^{(k\times p_x)}$ and $H \in \mathbf{R}^{(k\times p_y)}$, and WX, HY are calculated embeddings.

As *DeepWalk* was extended with the ability to combine other features, it soon became prevalent in various applications: (Tu et al., 2016) combine the structure of *DeepWalk* and a Support Vector Machine (SVM) design together with an intention to derive discriminative embeddings over a specific classification problem; (Grover & Leskovec, 2016) replace the original random walk process with a "biased" one, combining Breadth-First Search (BFS) and Depth-First Search (DFS) with a probability model.

4. Alternatives: Various Approaches to Network Embedding

Besides mentioned models, various approaches are conducted on the Network Embedding task as well.

4.1. Matrix Factorization

GraRep (Cao et al., 2015) calculated the first k-order proximity matrices of a network, decomposed these matrices with Single Value Decomposition (SVD) and concatenated the features attained as the final embedding. The performance of this algorithm was excellent at the cost of its computational complexity, which limited its usage to smaller-scale problems than other approximation methods.

NEU (Yang et al., 2017) proposed an iterative method by comparing advantages over Spectral Clustering, *word2vec* and *GraRep*. It updated the embedding matrices R and C with a designed policy that

$$\begin{aligned} R &= (I + \lambda A)R \\ C &= (I + \lambda A^T)C \end{aligned}$$

where A is the row-normalized adjacent matrix. A higher-order iterative method was along proposed as

$$\begin{aligned} R &= (I + \lambda_1 A + \lambda_2 A^2)R \\ C &= (I + \lambda_1 A^T + \lambda_2 (A^T)^2)C \end{aligned}$$

to collect higher-order information in the iterative process.

NetMF (Qiu et al., 2018), however, unified four current Network Embedding Algorithms together, namely *DeepWalk*, *LINE* (Tang et al., 2015b), *PTE*

(Tang et al., 2015a), and *node2vec* on the perspective of Matrix Factorization. Besides, it proposed a novel approach in Network Embedding with similar ideas of the four methods mentioned. The newly proposed method have different regards to small and large window-sizes. The formulas are too complicated to be described there. The interested can resort to the very publication.

4.2. Heterogeneity

Heterogeneity is sometimes a nature of real-world networks, such as citation networks including both authors and publications. Dealing with general heterogeneity is a very difficult problem. Several attempts were proposed, while all of them are limited and cannot be applied in a wider range.

PTE (Tang et al., 2015a) specified the problem over bipart networks containing word-word, word-document and word-label. The heterogeneity is naively solved with a simple sum of three hyper-parameterized loss over three sub-graphs, i.e.

$$
\begin{aligned}
O_{ww} &= -\sum_{(i,j)\in E_{ww}} w_{ij}\log p(v_i|v_j) \\
O_{wd} &= -\sum_{(i,j)\in E_{wd}} w_{ij}\log p(v_i|d_j) \\
O_{wl} &= -\sum_{(i,j)\in E_{wl}} w_{ij}\log p(v_i|l_j) \\
O_{pte} &= O_{ww} + O_{wd} + O_{wl}
\end{aligned}
$$

(Chang et al., 2015) adopted similar ideas in neural networks, which took an image-document network as example. By dividing the heterogeneous network into three sub-architectures sharing weights, which is shown in Figure 3. The model passably solved the problem with little novelty.

HINES (Huang & Mamoulis, 2017) took a more generalizable approach by introducing the idea *metapath*. By connecting two vertices with a metapath and a truncated proximity defined on it corresponding to the type of vertices and edges on this metapath, the model simulated a heterogeneous k-order proximity. Only the metapaths with lengths shorter than a threshold l is considered as valid. The total model is identical to typical network embedding models-a KL-divergence loss.

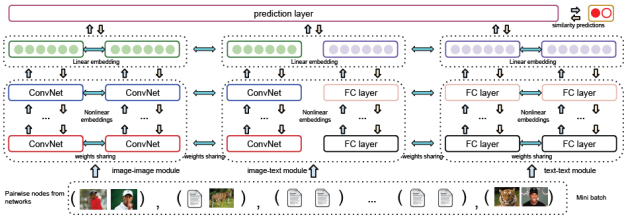

Figure 3. Architectures proposed in (Chang et al., 2015)

4.3. Other Important Work

Applications of convolutional neural networks, although rare compared to MF based models in network areas, do have some instructive meanings. (Kipf & Welling, 2017) is one of the pioneers in these attempts. It defined convolutions over graphs and implemented a semi-supervised classification method as the downstream application.

LINE (Tang et al., 2015b) was also one of the most impressive work in Network Embedding areas. It implemented not only the first-order proximity over a graph, but also a second-order proximity defined over common neighbors of vertices. Beware that this publication was earlier than all those analysis over proximity matrices of graphs, it indeed influenced coming research work significantly.

SDNE (Wang et al., 2016) adopted another dimensionality reduction tool, *Autoencoders*, in Network Embedding areas. *SDNE* took basic local features, e.g. PMI, as input and output in the autoencoder structure while monitoring the loss of its second-order proximity. The sparsity of networks are specially treated with a self-designed loss

$$
\begin{aligned}
\mathcal{L}_{2nd} &= ||(\hat{X} - X)\odot B||_F^2, \\
b_{ij} &= 1 \qquad s_{ij} = 0 \\
, b_{ij} &= \beta > 1 \qquad s_{ij} \neq 0.
\end{aligned}
$$

Figure 4 shows the architecture of *SDNE*.

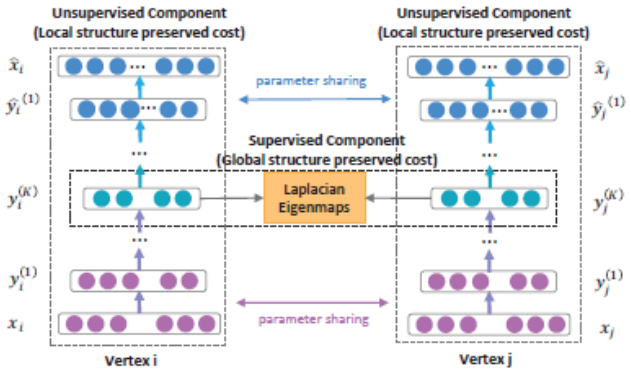

Figure 4. Architectures proposed in *SDNE*.

ANE (Dai et al., 2018) adopted the newest and hottest technique, GAN, in Network Embedding area. It trains the generator and discriminator in adversarial relations to identify the quality of the embeddings. Figure 5 is the architecture proposed in *ANE*.

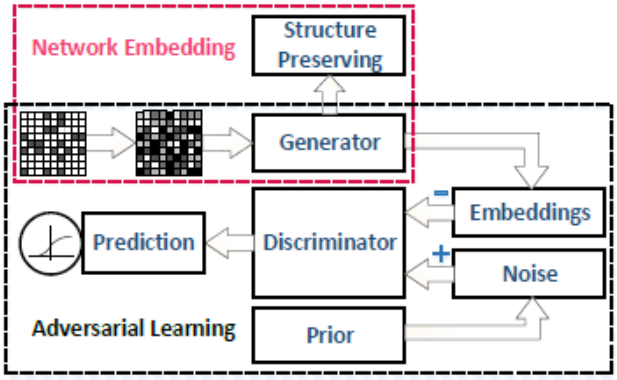

Figure 5. Architectures proposed in *ANE*.

5. Network Embedding: the Future

The future work in network embedding will probably start (or continue) focusing on dynamic networks, stream networks and very large scale networks. These are the most problematic parts in current network embedding tasks. In these tasks, as the snapshots of streams and dynamic networks can generate massive amount of redundant information, current methods seem clumsy performing on these problems. Heterogeneous networks, as mentioned above, still also have spaces for researches looking for more essential and generalizable algorithms and analysis.

References

Belkin, Mikhail and Niyogi, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of NIPS*, 2000.

Bengio, Yoshua, Ducharme, Rjean, Vincent, Pascal, and Jauvin, Christian. A neural probabilistic language model. *Journal of Machine Learning Research*, pp. 1137–1155, 2003.

Cao, Shaosheng, Lu, Wei, and Xu, Qiongkai. Grarep: Learning graph representations with global structural information. In *Proceedings of CIKM*, pp. 891–900, 2015.

Chang, Shiyu, Han, Wei, Tang, Jiliang, Qi, Guo-Jun, Aggarwal, Charu C, and Huang, Thomas S. Heterogeneous network embedding via deep architectures. In *Proceedings of KDD*, pp. 119–128, 2015.

Dai, Quanyu, Li, Qiang, Tang, Jian, and Wang, Dan. Adversarial network embedding, 2018.

Grover, Aditya and Leskovec, Jure. node2vec: Scalable feature learning for networks. In *Proceedings of KDD*, pp. 855–864, 2016.

Huang, Zhipeng and Mamoulis, Nikos. Heterogeneous information network embedding for meta path based proximity. *CoRR*, abs/1701.05291, 2017. URL http://arxiv.org/abs/1701.05291.

Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

Levy, Omer and Goldberg, Yoav. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, 2014.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *Computer Science*, 2013.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

Perozzi, Bryan, Al-Rfou, Rami, and Skiena, Steven. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, pp. 701–710, 2014.

Qiu, Jiezhong, Dong, Yuxiao, Ma, Hao, Li, Jian, Wang, Kuansan, and Tang, Jie. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of WSDM*, 2018.

Tang, Jian, Qu, Meng, and Mei, Qiaozhu. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of KDD*, pp. 1165–1174, 2015a.

Tang, Jian, Qu, Meng, Wang, Mingzhe, Zhang, Ming, Yan, Jun, and Mei, Qiaozhu. Line: Large-scale information network embedding. In *Proceedings of WWW*, pp. 1067–1077, 2015b.

Tenenbaum, Joshua B., de Silva, Vin, and Langford, John C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

Tu, Cunchao, Zhang, Weicheng, Liu, Zhiyuan, and Sun, Maosong. Max-margin deepwalk: discriminative learning of network representation. In *Proceedings of IJCAI*, 2016.

Wang, Daixin, Cui, Peng, and Zhu, Wenwu. Structural deep network embedding. In *Proceedings of KDD*, pp. 1225–1234, 2016.

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

Yang, Cheng, Sun, Maosong, Liu, Zhiyuan, and Tu, Cunchao. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of IJCAI*, 2017.