Motif-Aware Graph Embedding

Hoang Nguyen
Tokyo Institute of Technology
hoangnt@ai.cs.titech.ac.jp

Nukui Shun
Tokyo Institute of Technology
nukui.s@net.c.titech.ac.jp

Tsuyoshi Murata
Tokyo Insitute of Techonology
murata@c.titech.ac.jp

ABSTRACT

Given a large complex graph, how can we learn a lower dimension vector representation of each vertex that preserves structural information? Recent advancements in graph embedding have used word embedding techniques and deep architectures to propose a feasible answer to this question. However, most of these work considers the notion of "neighborhood" by node adjacency only. In this paper, we propose a novel graph embedding algorithm that employs motif structures into the latent vector representation learning process. By contrasting between sets of nodes created by random walks and sets of nodes created by biased *motif walk*, we show that embedding results of our algorithm are more accurate in various benchmark graph mining tasks compared to existing algorithms. The source code and results of our algorthms is available online at https://github.com/anonsyuushi/mage.git.

CCS Concepts

•**Computing methodologies** → **Learning latent representations;** *Neural networks;* •**Mathematics of computing** → *Graph theory;*

Keywords

Distributed representation; Graph embedding; motif; autoencoder; word2vec; MAGE

1. INTRODUCTION

Meaningful distributed representation of a high dimensional sparse dataset has proven to be useful for various machine learning tasks. For example, the **dense vector representation** of words in word2vec framework [2] has enabled machine learning researchers to **put applications and citation of word2vec here**.

Recently, starting from 2014 with the *DeepWalk* algorithm [4], there has been many proposed algorithms to encode a network's component such as vertices or edges into a high dimensional real vector. The motivation behind these algorithms is to learn a dense representation of the network anologous to learning a dense representation of a word [2]. By encoding the whole network into vectors, graph embedding algorithms have enabled network researcher to use the power of neural network techniques on network data [**?**]. Basically this solve the sparsity problem of the network.

Our algorithm outperfroms every other algorithms on their test. We conducted experiment throughly and carefully. Each experiment is ran with the same condition 10 times and take average, we have all the deviation stuff that others don't have. Really, it's really a very good paper. Please accept it so I can go to UK for a vacation. Please.

The good news is, with only a handful of manual settings[1], the LaTeX document class file handles all of this for you.

The remainder of this document is concerned with showing, in the context of an "actual" document, the LaTeX commands specifically available for denoting the structure of a proceedings paper, rather than with giving rigorous descriptions or explanations of such commands.

2. RELATED WORK

In the context of topological graph theory, an embedding refers to a representation of a graph G on a surface Σ. Graph embedding can also be viewed as dimension reduction when the dimensionality of the surface Σ is less than the dimensionality of the graph. Meaningful graph embedding, in practice, is low dimensionality vector representations of vertices preserving some analytical properties of the graph. Generally, there are two main approaches to obtain high-quality graph embeddings: affinity matrix factorization and machine learning with neural networks.

2.1 Matrix factorization

== NUKUI ==

2.2 Skipgram model

Our work in this paper is directly related to the *word2vec* model [**?**]. More specifically, the

The literature of graph embedding start with matrix decomposition techniques [**?**]. Super professor et. al. perform amazing works and achieved many success. Traditionally, graph clustering procedure where similarity between nodes or edges is defined and group together by some similarity

[1]Two of these, the `\numberofauthors` and `\alignauthor` commands, you have already used; another, `\balancecolumns`, will be used in your very last run of LaTeX to ensure balanced column heights on the last page.

Web Search and Data Mining '17 Cambridge, USA

DOI: 10.475/123_4

metric. This leads to the notation of closeness and hence community. However, one limitation of these classical graph embedding technique is that it depends on matrix decompositon which is very computational expensive.

Recently, with the emerging of deep neural network models such as autoencoder [**?**], restricted Bolzman machine [**?**], and some other that I will fill here later. Inspired by these advancement of natural language processing neural network models, Peperozi et. al. [4] proposed a graph embedding framework named DeepWalk. By taking advantage of the network structure and create an artificial "node corpus" by performing random walk, DeepWalk has achieved good performance and inspired any following researches. Since 2014, many graph embedding model based on random walk has been proposed such as LINE, GraRep, etc. These embedding algorithm proved its efficiency and usefullness in various graph mining tasks and classification problem in large graph.

Out of the aforementioned embedding algorithms, LINE has taken our notice. LINE model improved DeepWalk by incooporating the notation of second order proximity to the learning task. This leads to its outstanding performance without other auxilariy information. There are some research that has good result in embedding too, but out of all research that only use graph structure, LINE is the best. We think that the key to LINE's success was the second order proximity added to the original embedding scheme. However, LINE perform extremely well in network with citation-like structure. In other word, LINE is very good for network with wedge motif. From this observation, we think that if we can incooporate the most popular substructure of the graph to the embedding scheme, we can learn better vector representation.

2.3 Motif in Graph

Motif is defined as a small subgraph that has some funny characteristic [**?**], maybe cite some of Jure's work here.

Definition 1. A graph is represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges between vertices. Each edges $e \in E$ is an ordered pair $e = (u, v)$ where $u, v \in V$. A graph is called *undirected* when $(u, v) \equiv (v, u)$, and *directed* when $(u, v) \not\equiv (v, u)$.

3. MOTIF-AWARE GRAPH EMBEDDING

As mentioned in the previous sessions, our work aims to improve embedding quality by manipulating the graph data generation process. We have two data generation processes in our framework. The first process uses conventional random walk and a "skip window" to generate positive samples, while negative samples are picked from a noise distribution $p_n(x)$. The second uses a pre-defined *motif walk* to generate positive samples, while negative samples is generated from a contrasting distribution:

$$p_{\mathrm{mc}}(x) = \boldsymbol{f}\left(p^t_{\mathrm{m}}(x) \;||\; p^t_{\mathrm{r}}(x)\right), \tag{1}$$

where $\boldsymbol{f}(.)$ is chosen to be a distance function that yields high probability for node x when x is more likely to appear in random walk starting from node t than in motif walk starting from node t. $p^t_m(x)$ and $p^t_r(x)$ are distribution of nodes count in motif walk and random walk starting from vertex t.

Our model is a variant of the skipgram model [2] using the softmax function, in which we have the conditional probability of a "class" vertex (context) given the "target" vertex (word) as follow:

$$p\left(v_{\mathrm{class}} \mid v_{\mathrm{target}}\right) = \frac{\exp(\omega_c^\top \cdot \omega_t)}{\sum_{i=1}^{|V|} \exp(\omega_i^\top \cdot \omega_t)} \tag{2}$$

Despite the efficiency and simplicity of this model, the task of computing the normalization factor requires summing all over the graph's vertices is intractable for large graph. To solve this normalization problem, estimation techiques such as hierachial softmax [3], noise contrastive estimation [1], and negative sampling [2] are employed in the recent graph embedding models [4, 5, 6]. We define the loss function with negative sampling for our model as follow:

$$\mathcal{L} = -\log p\left(v_{\mathrm{class}} \mid v_{\mathrm{target}}\right) = \alpha\mathcal{L}_{\mathrm{r}} + (1-\alpha)\mathcal{L}_{\mathrm{mc}} \tag{3}$$

$$\begin{aligned}
\mathcal{L}_{\mathrm{r}} &= -\log\sigma\left(\omega_c^\top \cdot \omega_t\right) - \sum_{i=1}^{k} \mathbb{E}_{\omega_s \sim p_{\mathrm{n}}(\omega)}\left[\sigma(-\omega_s^\top \cdot \omega_t)\right] \\
\mathcal{L}_{\mathrm{mc}} &= -\log\sigma\left(\omega_c^\top \cdot \omega_t\right) - \sum_{i=1}^{k} \mathbb{E}_{\omega_s \sim p_{\mathrm{mc}}(\omega)}\left[\sigma(-\omega_s^\top \cdot \omega_t)\right]
\end{aligned} \tag{4}$$

3.1 Graph sampling process

The first data generation process is similar to that of the Deepwalk model. The difference in our model is negative sampling from the vertex degree distribution is used to estimate the valid distribution. Algorithm 1 describes the positive and negative samples generating from random walk process.

Algorithm 1: `gen_rand`: sample by random walk

```
Data: Graph G = (V, E)
Input: walkLength, skipWindow, numSkip, numNeg,
       distort, random_walk
Output: (targets, classes, labels)
begin
    targets ← [ ]; classes ← [ ]; labels ← [ ];
    idList ← Shuffle(V);
    for i ∈ idList do
        walk ← randomWalk(start=i,
         length=walkLength); for j ∈ walk do
            for j ∈ range(numSkip) do
                targets.append(j);
                classes.append(random.choice(walk[j-
                 skipWindow:j+skipWindow]);
                labels.append(1.0);
            for j ∈ range(numNeg) do
                targets.append(j);
                classes.append(random.choice(V,
                 distort));
                labels.append(0.0);
    return (targets, classes, labels);
```

The second data generation process is the core of our method. Positive data samples are generated using a biased

random walk, which is called `motif_walk`. For the negative samples generation, we perform unbiased random walk and select vertices that appear frequenty in the random walk, but less frequenty in the postive motif walk. The intuition for this technique comes from our hypothesis that vertices in the same motif cluster are more likely to be related than vertices in the random walk. As discussed by Gutmann and Hyvärinen in [1] for the choice of noise distribution in practice, the more similar the noise distribution to the true distribution leads to better learning result.

Algorithm 2: `gen_motif`: sample by motif walk

```
Data: Graph G = (V, E)
Input: mwalkLength, rwalkLength, numSkip, numNeg,
       contrastIter, motif_walk
Output: (targets, classes, labels)
begin
    targets ← [ ]; classes ← [ ]; labels ← [ ];
    idList ← Shuffle(V);
    for i ∈ idList do
        posSet ← { }; negSet ← { };
        for j ∈ range(contrastIter) do
            pwalk ← motif_walk(start=i,
              length=mwalkLength);
            posSet.add(pwalk);
            nwalk ← random_walk(start=i,
              length=rwalkLength);
            negSet.add(nwalk);
        for k ∈ range(numSkip) do
            targets.append(j)
            classes.append(random.choice(posSet)
            labels.append(1.0)
        for k ∈ range(numNeg) do
            targets.append(j)
            classes.append(random.choice(negSet -
              posSet)) labels.append(0.0)
return (targets, classes, labels)
```

Algorithm 2 generates positive samples without the need of skip window for choosing local vertices given the target vertex. For negative samples, the vertices that apprear in random walk but not in motif walk are chosen at random. In this algorithm we have two free parameters: function `motif_walk` and the contrasting method $\boldsymbol{f}$. For simplicity, we implement undirected triangle (algorithm 3) and directed bipartite (algorithm 4) motif walk with simple set subtraction for contrastive sampling. However, the extension is discussed in later sections.

Algorithm 3: `triange_walk`: triangle motif walk

```
Data: Graph G = (V, E)
Input: start, mwalkLength
Output: walk
begin
    ==
return (targets, classes, labels)
```

3.2 Optimization

Our objective is to minimize the log-loss described in equation 3, in which parameter α controls the training portiion of

Algorithm 4: `bipartite_walk`: sample by motif walk

```
Data: Graph G = (V, E)
Input: mwalkLength, rwalkLength, numSkip, numNeg,
       contrastIter, motif_walk
Output: (targets, classes, labels)
begin
    ==
return (targets, classes, labels)
```

random walk samples and motif walk samples. We employ binary cross entropy as the batch loss in our implementation:

$$\mathcal{H}(p_{\text{ predicted}}, p_{\text{ label}}) = -\sum_{x} p_{\text{ predicted}}(x) \log p_{\text{ label}} \qquad (5)$$

Figure **??** describes our training framework. Backpropagation is used to miminize the loss function described in equation 3 and equation 5. The parameter set of our models is $\{\mathcal{W}_{\text{emb}}, \mathcal{W}_{\text{neg}}\}$, each has shape $(|V|, d)$, where d denotes the embedding dimension. The parameters are initialized by normal distribution and uniform distribution of mean 0. During training, we linearly change the value of α starting from 1 to 0. As α value changes, our model is trained on different portion of random walk samples and motif walk samples.

3.3 Training procedure

4. EXPERIMENTS

5. ACKNOWLEDGMENTS

We acknowledge our friends who gave us insightful input.

6. REFERENCES

[1] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, page 6, 2010.

[2] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[3] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005.

[4] B. Perozzi, R. Al-Rfou, and S. 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. ACM, 2014.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM, 2015.

[6] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*. ICML, 2016.