Scalable Multiplex Network Embedding

Hongming Zhang1, Liwei Qiu2, Lingling Yi2, Yangqiu Song1

1Department of CSE, The Hong Kong University of Science and Technology.
2Tencent Technology (SZ) Co., Ltd., China.
hzhangal@cse.ust.hk, deolcaqiu@tencent.com, chrisyi@tencent.com, yqsong@cse.ust.hk

Abstract

Network embedding has been proven to be helpful for many real-world problems. In this paper, we present a scalable multiplex network embedding model to represent information of multi-type relations into a unified embedding space. To combine information of different types of relations while maintaining their distinctive properties, for each node, we propose one high-dimensional common embedding and a lower-dimensional additional embedding for each type of relation. Then multiple relations can be learned jointly based on a unified network embedding model. We conduct experiments on two tasks: link prediction and node classification using six different multiplex networks. On both tasks, our model achieved better or comparable performance compared to current state-of-the-art models with less memory use.

1 Introduction

Network or graph embedding, which uses dense vectors to represent nodes, has been well studied for many years [Archdeacon, 1996; Chung, 1997; Goyal and Ferrara, 2017]. Recently, inspired by recurrent neural networks [Goodfellow et al., 2016], network embedding has been re-investigated and developed based on random walks over graphs and stochastic gradient descent optimization, which can be applied to very large graphs, such as real social networks [Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016]. The embedding vector of nodes can be used to encode some of the topological structure of the network, and then can be used as features for downstream models. Network embedding has been proven to be helpful for network analysis tasks including link prediction, node classification, and community detection.

Multiplexity property is a common feature of networks, especially for social networks. In the social network analysis, multiplexity refers to multifaceted relationships between two people [Verbrugge, 1979]. If we generalize this idea to all kinds of network, by "multiplex network," we mean a group of networks which contains multiple kinds of relations, and each kind of the relations can create a layer of the network. Take the social network as an example. In a social network such as Facebook, users often have different kinds of interactions with each other like friendship relation, forwarding articles to each other, conversation, money transferring, etc. Each of them will create a layer of the network among all users. If we consider them as a united one, we will get a huge multiplex network. While each layer of the network can only represent one kind of interactions among users, to better understand the whole multiplex network, it is better to integrate different types of information from these networks together without sacrificing their distinctive properties.

Considering the fact that a network could be huge, in this paper, we present a scalable multiplex network embedding model to efficiently store and learn information of multi-type relations into a unified embedding space. For each node, we propose one high-dimensional common embedding vector, which is shared across all layers of the multiplex network, and we use that to build a bridge among different layers of the network. For each node, to learn its distinct property on different layers with small memory occupation, we also propose a low-dimensional additional vector for each type of relation. To align these embeddings with different dimensions together, we introduce a global transformation matrix for each layer of the network. Following DeepWalk [Perozzi et al., 2014], we use stochastic gradient descent to optimize our model. Since the global transformation matrix will be updated much more times than node vectors will be, we add an additional regularization to constrain the size of the matrix.

Our contributions can be summarized as follows.

• We formally define the problem of multiplex network embedding to handle the multiplexity property of networks.
• We propose a scalable multiplex network embedding model to represent information from multi-type relations into a unified embedding space.
• We evaluate our embedding algorithm on two network analysis tasks: link prediction and node classification. On both tasks, our model achieved better or comparable performance compared to current state-of-the-art models with less memory use.

The source code of our model is available at: github.com/HKUST-KnowComp/MNE
2 Related Work

2.1 Multiplex Network Analysis

Traditionally in social science, multiplexity has been used to characterize the multiple facets of social exchange relationships among users [Verbrugge, 1979]. The idea of multiplexity can be generalized to all kinds of network. In data mining community, people sometimes also use the term “multi-relational network” to represent the multi-type relations in social networks and verified that considering such relations in the social network can help data mining tasks such as community mining or link prediction [Cai et al., 2005; Chen et al., 2016]. In addition, in bioinformatics community, it has been shown that by integrating multiple networks, the node representation can be improved for genes functional analysis [Cho et al., 2016].

Nowadays, graph mining methods on the multiplex network are mainly targeting on specific tasks. For link prediction, traditional methods [Newman, 2001; Adamic and Adar, 2003] can be applied to multiplex network without considering the information of the typed edges. For community detection, cross-layer centrality measurement [Bródka et al., 2012] is proposed to capture the centrality of nodes in a multiplex network. Besides that, multilayered local clustering coefficient (MLCC) and cross-layer clustering coefficient (CLCC) [Kazienko et al., 2010; Lytras et al., 2010] are also proposed to describe cluster coefficient of a node in a multiplex network.

In this paper, we propose a general solution for multiplex network analysis from the perspective of network embedding.

2.2 Network Embedding

Network embedding (or sometimes called graph embedding, historically also graph drawing) has been well studied for many years [Archdeacon, 1996; Chung, 1997; Goyal and Ferrara, 2017]. It is also related to manifold learning [Huo et al., 2007] while manifold learning was usually applied to dimensionality reduction for high-dimensional data. Network embedding focuses on generating the vector representation of nodes for real networks or graphs to facilitate further analysis of networks. Traditional approaches to network embedding usually involve time-consuming computational components, such as eigenvalue decomposition to analyze the spectral property of graphs [Spielman, 2011]. However, given modern very large-scale online social networks, such approaches may be less efficient and sometimes infeasible to generate the node representations.

Recently, inspired by the development of recurrent neural networks [Goodfellow et al., 2016], particularly the efficient word embedding method, word2vec [Mikolov et al., 2013b; Mikolov et al., 2013a], many network embedding approaches have been proposed to handle large-scale social networks. For example, DeepWalk [Perozzi et al., 2014] proposes to perform random walk on the network to generate sequences of nodes and then perform skip-gram algorithm (a technique used in word2vec) on those sequences to achieve the embedding. On top of DeepWalk, Node2Vec [Grover and Leskovec, 2016] adds two parameters to control the random walk process and make it biased random walk. Some other embedding models focus on specific kinds of structures in the network. For example, LINE [Tang et al., 2015] tries to use the embedding to approximate the first-order and second-order proximities of the network.

All the models above have been proven useful on single network analysis, but they did not consider multiplexity. Recently, an embedding approach is proposed to target the multiplexity property [Liu et al., 2017]. They proposed three methods to learn one overall embedding from the multiplex network. Unlike their approach, we propose one high-dimensional common embedding and several lower-dimensional additional embeddings for each type of relations. A recent work [Li et al., 2018] called itself multi-layer network embedding. Essentially, it is a heterogeneous information network since in the definition in their paper, different layers have different type of nodes.

3 Multiplex Network Embedding

3.1 Problem Definition

We first introduce our problem and notations. Given a multiplex network, our goal is to learn a representation for each node. Suppose $N$ is the set of nodes and there are $M$ different relation types, represented as $G_1, \ldots, G_M$. For each relation based network $G_i = (N_i, E_i)$, we have $E_i \subseteq N_i \times N_i$ and $N_i \subseteq N$.

For each node $n \in N$, it has one common embedding vector, which is shared across all the relation types and denoted as $b_n \in \mathbb{R}^d$. We use that common embedding as a bridge to transfer information across different relation types. To capture the distinct property of each sub-network, for each node, we also propose a series of low-dimensional vectors $u_i^n \in \mathbb{R}^d$ for each relation type $i$. Here, $d$ is the dimension of common embedding and $s$ is the dimension of typed relation vector. To avoid our model becoming too large, we set $s \ll d \ll |N|$, where $|N|$ is the number of all nodes. To align $b_n$ and $u_i^n$, we introduce a transformation matrix $X_i \in \mathbb{R}^{s \times d}$. Then we use a new vector $v_i^n$ to represent the embedding vector of $n$ for relation type $i$:

$$v_i^n = b_n + u_i^n \cdot X_i^T u_i^n,$$  \hspace{1cm} (1)

where we use a weight $w_i$ to denote the overall importance of relation type $i$.

3.2 Optimization

For each relation type $i$ of the multiplex network, we conduct random walk to generate a sequence of nodes and then perform Skip-gram algorithm used in word2vec [Mikolov et al., 2013a] over the sequences to learn the embeddings. As shown in Eq. (1), a node’s final embedding is composed of three parts: common embedding $b_n$, relation-based embedding $u_i^n$, and transformation matrix $X_i$. We need to learn these parameters at the same time.

For any node $n$ in a random walk path relation type $i$, assuming it appears in position $j$, we will take $n_{j-c}, \ldots, n_{j-1}, n_{j+1}, \ldots, n_{j+c}$ as its neighbors, where $c$ is half of the window size. Thus, given a sequence of nodes, our objective is to minimize the following negative log-likelihood:

$$-\log P_{\theta}(n_{j-c}, \ldots, n_{j-1}, n_{j+1}, \ldots, n_{j+c} \mid n_j),$$  \hspace{1cm} (2)
which can be further factorized as $-\prod_{i=1}^{n-1} \log P_{G_i}(n_k|n_j) - \prod_{i=n}^{N} \log P(n_k|n_j)$ given the conditional independence assumption. Here we use $P_{G_i}$ to denote the parameterization for relation type $i$. For each $P_{G_i}(n_k|n_j)$, a softmax function is used to define the probability:

$$P_{G_i}(n_k | n_j) = \frac{\exp(v'_{n_k} T \cdot v_{n_j})}{\sum_{j \in \mathcal{N}_{nj}} \exp(v'_{n_k} T \cdot v_{nj})},$$

(3)

where $v'_{nj}$ represents the input word embedding of user $n_j$, and $v'_{nj}$ and $v_{nj}$ represent the parameters of context vectors shared by all relation types, which are only used during the training process. To speed up the training process, following word2vec [Mikolov et al., 2013a], we use negative sampling to approximate the objective function as:

$$E = - \log \sigma(v'_{n_k} T \cdot v_{n_j}) - \sum_{n \in \mathcal{N}_{nj}} \log \sigma(-v'_{n_k} T \cdot v_{nj}),$$

(4)

where $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic function and $\mathcal{N}_{nj}$ is the randomly sampled negative context node set for $n_j$.

Assuming the sequence is generated in relation type $i$, according to Eq. (1), in the new objective function, we can replace $v'_{nj}$ with $b_{nj} + XiT u_{nj}$. After that we employ stochastic gradient descent (SGD) to minimize the objective function. We update the context parameter vectors, node embedding vectors, and transformation matrix as follows:

$$b_{nj} := b_{nj} - \eta \frac{\partial E}{\partial b_{nj}} = b_{nj} - \eta \cdot e(n_j, n_k) \cdot v'_{nj},$$

(5)

$$v'_{nj} := v'_{nj} - \eta \frac{\partial E}{\partial v'_{nj}} = v'_{nj} - \eta \cdot e(n_j, n_k) \cdot v_{nj},$$

(6)

$$u_{nj} := u_{nj} - \eta \frac{\partial E}{\partial u_{nj}} = u_{nj} - \eta \cdot e(n_j, n_k) \cdot X^i v_{nj},$$

(7)

$$X^i := X^i - \eta \frac{\partial E}{\partial X^i} = X^i - \eta \cdot e(n_j, n_k) \cdot u_{nj} \cdot v'_{nj} T,$$

(8)

where $e(n_j, n_k) = \sigma(v'_{n_k} T \cdot v_{nj}) - t_k$ and $t_k = 1$ if $n_k$ is the neighbor of $n_j$ and $t_k = 0$ if $n_k$ is in the negative sampling set.

We initialize all the $v'_{nj}$s with zero vector and $X^i$ with zero matrix. For context parameter vector $v'_{nj}$ for all the users, we randomly initialize them. Since $X^i$’s can be updated many times when optimizing the objective function, to avoid its norm becoming too huge, we add the following constraints:

$$||X^i|| \leq r,$$

(9)

where $r$ is a controlling parameter for the Frobenius norm of matrix $||X^i||$. In our experiment, we also involve an extra combination layer and since this layer has no actual meaning, we will only update common embedding during the training on that layer.

We summarize our algorithm in Algorithm 1. As our model is a random walk based solution, assuming that we have $M$ relation types of sub-networks and $|N|$ nodes, the time complexity of our model is $O(M|N|^2)$. The memory complexity of our model is $O((d+s \cdot M)|N|).

---

**Algorithm 1 Multiplex Network Embedding**

**INPUT:** Multiplex network which is composed of sub networks $G_1, \ldots, G_M$, common embedding dimension $d$, additional embedding dimension $s$, learning rate $\eta$, size of context window $c$, number of negative sampling $k$, transformation matrix size regularization parameter $r$.

1. Initialize $b_n$, $v'_{nj}$, and $X^i$ for different nodes $n$’s and relations $i$’s.
2. For Each relation network $G_i$ do
   3. For Each path $P$ generated by random walk do
      4. For $n_j \in P$ do
         5. For $n_k \in P[j-c, j+c]$ do
            6. $N_{nj} = \text{NegativeSampling}(N, m)$
            7. $v'_{nj} = b_{nj} + X^i T u_{nj}$
            8. $N_{\text{sample}} = N_{nj} \cup n_c$
            9. For $n_k \in N_{\text{sample}}$ do
               10. $e(n_j, n_k) = \sigma(v'_{n_k} T \cdot v'_{nj}) - t_k$
               11. Update $b_{nj}, v'_{nj}, u_{nj}$, and $X^i$ based on Eqs. (5), (6), (7), and (8) respectively.
      12. End for
      13. If $||X^i|| > r$ then
         14. $X^i := \frac{r}{||X^i||} \cdot X^i$
         15. End if
      16. End for
   17. End for
3. End for

**OUTPUT:** Common embedding $b_n$, additional embeddings $u_{nj}$, and transformation matrix $X^i$ for different nodes $n$’s and relations $i$’s.

---

### 4 Experiments

#### 4.1 Datasets

We use open multiplex networks from Manlio De Domenico project as our experimental datasets. We select four multiplex networks from different network types. The selection criteria are that they cannot be too small and they cannot be too sparse, which means that they should have at least one edge per node per relation type. The details of these datasets are as follows:

- **Vickers** [Vickers and Chan, 1981]: Data collected by asking 29 seventh grade students in a school in Victoria, Australia, for whom they asked three questions and each of them creates a relation type of the network.
- **CKM** [Coleman et al., 1957]: Data collected by asking physicians in four towns in Illinois, Bloomington, Quincy, and Galesburg. They asked three questions and each of them creates a relation type of the network.
- **LAZEGA** [Lazega, 2001]: This multiplex social network consists of three kinds of relationships (Co-work, Friendship, and Advice) between partners and associates of a corporate partnership.

---

C.ELEGANS [Chen et al., 2006]: This multiplex network consists of layers corresponding to different synaptic junctions: electric (“ElectrJ”), chemical monadic (“MonoSyn”), and polyadic (“PolySyn”) among a group of neurons.

Since the sizes of the above multiplex networks are relatively small, to test the scalability and performance of our model, we also conduct experiments on two real large-scale online social media networks: Twitter and Private. The detailed information of these large scale social networks are as follows:

**Twitter:** The Twitter data used in our experiments is the higgs-twitter dataset. This dataset was used to analyze the spreading of news [De Domenico et al., 2013]. It crawled all tweets about a scientific term “Higgs boson discovery” in one week. We select the largest two sub-networks (following and retweet) to build the multiplex network.

**Private:** Our Private data is based on an online social network, where users can have friendship relations with each other and can send articles to their friends. From all of our users, we randomly chose 40,000 users who have annotated themselves affiliated with a university and we recorded all the article forward activities within that group in one month. After applying topic modeling on the content of the articles, we grouped all the articles by topics. Thus, we got 16 topics and each one of them creates a relation type of the multiplex network. If we add the base friendship relation, we have in total 17 relation types.

The statistics of these datasets are listed in Table 1.
Table 2: Link prediction based on similarities between two nodes. All the numbers are the averaged AUC score based on five-fold cross validation. The standard deviations are reported in the parentheses.

<table>
<thead>
<tr>
<th>Model</th>
<th>Vickers</th>
<th>CKM</th>
<th>LAZEGA</th>
<th>C.ELEGANS</th>
<th>Twitter</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>0.821 (0.030)</td>
<td>0.781 (0.008)</td>
<td>0.780 (0.007)</td>
<td>0.821 (0.006)</td>
<td>0.502 (0.002)</td>
<td>0.621 (0.007)</td>
</tr>
<tr>
<td>LINE</td>
<td>0.676 (0.011)</td>
<td>0.637 (0.012)</td>
<td>0.695 (0.006)</td>
<td>0.732 (0.006)</td>
<td>0.519 (0.003)</td>
<td>0.512 (0.010)</td>
</tr>
<tr>
<td>Node2Vec</td>
<td>0.821 (0.030)</td>
<td>0.781 (0.008)</td>
<td>0.780 (0.007)</td>
<td>0.820 (0.006)</td>
<td>0.504 (0.003)</td>
<td>0.644 (0.010)</td>
</tr>
<tr>
<td>PMNE (n)</td>
<td>0.810 (0.032)</td>
<td><strong>0.917</strong> (0.008)</td>
<td>0.792 (0.009)</td>
<td>0.843 (0.003)</td>
<td>0.446 (0.004)</td>
<td>0.629 (0.006)</td>
</tr>
<tr>
<td>PMNE (r)</td>
<td>0.844 (0.025)</td>
<td>0.904 (0.008)</td>
<td>0.813 (0.007)</td>
<td>0.835 (0.007)</td>
<td>0.446 (0.001)</td>
<td>0.659 (0.005)</td>
</tr>
<tr>
<td>PMNE (c)</td>
<td>0.837 (0.029)</td>
<td>0.847 (0.016)</td>
<td>0.797 (0.011)</td>
<td>0.824 (0.009)</td>
<td>0.449 (0.002)</td>
<td>0.506 (0.004)</td>
</tr>
<tr>
<td>Common Neighbor (CN)</td>
<td>0.799 (0.011)</td>
<td>0.877 (0.006)</td>
<td>0.809 (0.007)</td>
<td>0.869 (0.002)</td>
<td>0.592 (0.002)</td>
<td>0.691 (0.002)</td>
</tr>
<tr>
<td>Jaccard Coefficient (JC)</td>
<td>0.778 (0.007)</td>
<td>0.873 (0.006)</td>
<td>0.826 (0.007)</td>
<td>0.833 (0.001)</td>
<td>0.520 (0.002)</td>
<td>0.573 (0.004)</td>
</tr>
<tr>
<td>Adamic/Adar (AA)</td>
<td>0.803 (0.019)</td>
<td>0.875 (0.013)</td>
<td>0.814 (0.008)</td>
<td>0.881 (0.001)</td>
<td>0.592 (0.002)</td>
<td>0.691 (0.003)</td>
</tr>
<tr>
<td>MNE</td>
<td><strong>0.871</strong> (0.014)</td>
<td>0.900 (0.010)</td>
<td><strong>0.839</strong> (0.013)</td>
<td><strong>0.910</strong> (0.006)</td>
<td><strong>0.622</strong> (0.003)</td>
<td><strong>0.723</strong> (0.002)</td>
</tr>
</tbody>
</table>

The larger the similarity the more likely there exists a link between them. As the multiplex network has more than one relation types, we will compute AUC for each relation type first and take the average of all the relation types as the final result. For the models designed for single layer network, we will train a separate embedding for each relation type of the network and use that to predict links on the corresponding relation type, which means that they do not have information from other relation types of the network. One thing worth mentioning is that, in the two large online social network, the following/friendship network is the base network for other relation types. For example, in our Private data, one can only send articles to their friends. To make the evaluation more similar to our real task, for the AUC evaluation of these two datasets, we will randomly select the same amount of edges from the base network, which do not appear in that relation type, and the additional vectors keep the distinct property of each relation type of the network.

4.5 Parameter Sensitivity

In our algorithm, there are three parameters: the dimension of low-dimensional addition vectors $s$, the weight of additional vectors $w^s$, and the transformation matrix norm limitation $r$. To clearly show the influence of these parameters, we repeat our link prediction experiment with different parameter settings. For the small multiplex datasets, we take the average of them to see the trend.

As shown in Figure 1(a), the performance of our model is improved when we increase the dimension of the additional vector. When the dimension reaches 10, the performance reaches the top, which proves that with the help of common embedding, we can use a very low dimension (about one-tenth of the base dimension) vector to represent the property of each relation type of the multiplex network.

As shown in Figure 1(b), when we increase the weight of the additional vector, the performance of our model will arise. But after reaching to 1, the performance will start to go down slightly. This is because in our embedding training phase, we fixed the weights as one. It seems tuning this weights using cross-validation does not further improve link prediction.

As shown in Figure 1(c), if we set the regularization parameter of the transformation matrix to be too small, we may restrict the learning process of our model. When the regularization parameter is larger than 10, the performance becomes stable and the influence of our restriction is not significant.

4.6 Node classification

In this section, we conduct the experiment on node classification task to evaluate the quality of embedding captured by our model. As the CKM social network is the only multiplex network that provides the ground truth about node labels, we use it as the experimental dataset. Considering the size of
datasets being small, we conduct 2-fold cross-validation experiment on this task. The dataset labels all the researchers with their original companies. We use this feature as labels to train the classifier and evaluate our embedding. For Deepwalk, LINE, and Node2Vec, as they are designed for single network, we first train a separate embedding for each relation type and average them as the final embedding for each node. For our model, we set the weight $w_i$ of all the types to be 1.

The result is shown in Figure 2. Our model can achieve the best performance among all the embedding models. This proves that through simple average operation, our model can produce the high-quality overall representation of nodes.

![Figure 2: Node classification result.](image)

### 4.7 Scalability of Our Model

In this section, we investigate the scalability of our model with our large-scale online social network. As we all know, in real life, the social network could be huge. For example, there are about 1 billion active users on our private social network. Every day, more than 500 million relational edges are created. If we learn embedding for all of the sub-networks, the training and storage of these models could be a big challenge. Hence we propose the structure of small additional vector and transformation matrix, trying to reduce the overall model size without sacrificing the overall performance.

As shown in Figure 3, the memory used by our model is almost linear to the network size. Compared with Node2Vec and LINE, our model occupy one-tenth of the memory space without sacrificing performance. The reason is obvious. As we showed in section 3, the memory complexity of our model is $O((d + s \times M) |\mathcal{N}|)$ and the memory complexity of Deepwalk and LINE is $O((d \times M) |\mathcal{N}|)$, where $s$ is designed to be much smaller than $d$. For the PMNE(n) and PMNE(c), as they merge the multiplex network into one single network, their model is the smallest but their model loses distinct information about relational edges, which is proved important in the previous experiment.

![Figure 3: Scalability of memory use.](image)

### 5 Conclusion and Future Work

In this paper, we present a new scalable embedding model for multiplex networks. We tested our model on two tasks: link prediction and node classification. The experimental results show that our model can effectively capture the overall representation of nodes without sacrificing the distinct property of every single relation types. We also conduct scalability experiments using our social network to show that when the network is large, we only need one-tenth of the old model size to achieve better performance. In the future, we will explore the idea of multiplex network for tasks in other areas.

### 6 Acknowledgment

This paper was supported by HKUST-WeChat WHAT Lab, China 973 Fundamental R&D Program (No. 2014CB340304), and the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong.
References


[Cai et al., 2005] Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. Community mining from multi-relational networks. In PKDD, pages 445–452, 2005.


