attr2vec: Learning Node Representations from Attributes of Nodes

Pengkun Zheng[1], Yan Wen[1(✉)], Ming Chen[2], and Geng Chen[3]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China
wenyan84@hotmail.com
[2] State Grid Shandong Electric Power Company, Qingdao Power Supply Company, Qingdao, China
[3] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao, China

Abstract. In recent years, the research in the multiple fields of representation learning has led to the emergence of many excellent Network Embedding algorithms. Here we propose *attr2vec*, a completely unsupervised algorithmic framework for learning the latent representations for nodes. In *attr2vec*, we have adopted an attribute processing method similar to GCN, that is, taking the average of the attribute of the node's neighbors as the attribute for the node. We also consider first-order neighbors and second-order neighbors separately to achieve an effect similar to multiple convolutional layers in GCN. In summary, our algorithm utilizes similar attribute processing idea of GCN, which can learn the graph topology and node attribute to generate latent representations for nodes, but implemented it with a completely unsupervised way. In some experiments on citation networks we demonstrate that our algorithm outperforms related unsupervised techniques by a significant margin.

Keywords: Network embedding · Unsupervised · Feature learning

1 Introduction

In many real-world problems, information is often organized as networks. Nodes in the network represent entities, and edges represent relationships between entities. We need to perform various prediction tasks on the network, such as visualization [1], node classification [2], and link prediction [3]. Therefore, it is necessary to accurately learn useful knowledge from the networks. Learning the network representations of network is a useful strategy: represent each node of the network with a low-dimensional vector which captures meaningful relational, structural and semantic information conveyed by the network.

Almost all recent efforts in network embedding are based on these intuitions: 1) Nodes should have similar latent representations with its neighbors. 2) Nodes that have neighbors with similar sets of nodes should have similar latent representations. Specifically, these efforts define a general concept of node's neighborhood, which is a random walk of nodes. Then transforms a network into a sample collection of linear

Q. Wu et al. (Eds.): WiSATS 2020, LNICST 358, pp. 546–555, 2021.
https://doi.org/10.1007/978-3-030-69072-4_44

sequences consisting of nodes using uniform sampling. The skip-gram model [4], a technique originally designed for learning latent representations of words in linear sequences, can also use these linear sequences of nodes as input to generate latent representations of the nodes.

These techniques for learning the representations of nodes in networks have been achieved great success in performing visualization, classification and prediction tasks. But neighborhood is a local concept usually composed of nodes that are close to each other in the network. A fundamental limitation is that network environmental similar nodes will never share the same context if their distance (hop count) in original network is larger than the skip-gram window. Thus, a pair of nodes in a network that are environmentally similar but that are far apart in distance will not have similar representations because they will not appear frequently in the same skip-gram input sequence. This is also the reason for representations such as DeepWalk [5] and node2vec [6] fail in classification tasks that depend more on structural identity.

struc2vec [7] provide an alternative methodology, one based on unsupervised learning of representations for the structural identity of nodes. It builds a multi-layer graph, making the similarity of nodes independent of their network position and labels. While struc2vec performs well in some completely structure-based classification tasks, it completely discards the neighbor similarity which contains important network topology information in real world network. Therefore, struc2vec does not perform well in general networks. However, struc2vec gives us an important revelation that we can build a new graph for random walk in order to find similar nodes that are far away from the original graph.

In short, these Network Embedding technologies generally samples the nodes in the network through specific strategies to learn the similarity of the nodes in the network, which can be regarded as a method of learning latent representation of the topology of the network. However, in the real world, the nodes in network also contain several attributes information, such as user portrait information in social networks, text information in citation networks, etc. For such information, the method based on Network Embedding usually splices the attributes to the nodes for downstream tasks.

In this paper, we propose *attr2vec*, which is a network embedding algorithm inspired by GCN [8]. GCN can directly generate node representations by learning the network topology and nodes attribute information, but it is a semi-supervised algorithm and requires a certain number of labeled nodes. Our main contribution is a completely unsupervised framework for generating latent representation for nodes by directly learning network topology and node attributes, called *attr2vec*. The key ideas of *attr2vec* are:

- Assess attributes similarity between nodes independently of their position in the network. If two nodes have similar attributes and their neighbors have similar attributes, they will have a high degree of similarity, regardless of their position in the network and the labels of their neighbors. For example, in the citation network, nodes represent articles, and the attribute of node are the known information of article, such as key words that appeared in the article. If two articles have similar attributes (such as key words), and the neighbor articles in citation network are also similar to each other, they will be similar in our access.

- Establish a multi-layer hierarchy to find nodes with similar attribute, allowing more stringent notion of similarity at higher layer. In particular, at the first level of the hierarchy, similarity between nodes depend only on their node attribute, while the similarity at the second level depends not only on their own attributes, but also on the attributes of their neighbors. This hierarchy can continue to expand until that at the top of the hierarchy the similarity between nodes depends on the entire network. As more and more layers of the hierarchy, the information obtained is getting richer, but the interference information has also increased.
- Generate random contexts for nodes by weighted random sample in the K-layer hierarchy, the random contexts for nodes are sequences of nodes that have similar attributes. Thus, nodes that frequently appear in similar contexts are more likely to have similar attributes. These contexts can be used as input to language models to learn the latent representation for nodes.

We implemented an example of *attr2vec* and confirmed its effect on several real networks through numerical experiments. We compared its performance with DeepWalk and node2vec, which are two latest learning techniques for latent representation for nodes. Our results show that *attr2vec* can make better use of the attribute information of the nodes and have better performance in classification task.

The rest of this paper is organized as follows. In Sect. 2, we briefly summarize the recent related work on learning latent representations for nodes. We present the technical details of *attr2vec* in Sect. 3. Section 4 shows experimental evaluation and comparison with other methods on classification tasks. Finally, we conclude in Sect. 5.

2 Related Work

Recent advancements in natural language processing have provided new method for learning latent representations for nodes in network. Specially, the skip-gram model is invented as a tool for learning representations for text data. It is based on the idea that words that frequently appear in similar contexts should be similar. Thus, skip-gram formulate a neighborhood preserving likelihood objective and optimize it using SGD [9] with negative sampling [10].

Inspired by the skip-gram model, recent research treats the network as sequences of nodes, the specific method is to randomly walk in the original network according to some rules to generate sequences of nodes. However, there are many feasible random walk strategies, which will result in different node sequences and different representations of nodes.

DeepWalk is the first algorithm to learn node representations using a language model. It obtains sequences of nodes by performing random walks in the original network, and then uses these sequences as input to the skip-gram model. This idea has been extended in node2vec, which propose a biased second order random walk strategy that increases flexibility when generating sequences of nodes. But one limitation is that if the distance between two nodes exceeds the window size in the skip-gram model, they will not have closely representation even if they are similar.

In order to make similar pairs of nodes that are far away in the original network appear in the same node sampling sequence, struc2vec construct a fully connected weighted hierarchical network and performs node sampling on this network (not the original network). Although struc2vec has succeeded in structure-based prediction tasks, it is not suitable for general real-world networks.

The concept of graph neural network (GNN) [11] improves the existing neural network so that it can process data represented by the network. The important difference between GNN and the previous technology is that it is semi-supervised and it can not only learn the topology of node but also the attributes of node. Experiment results show that GCN is a powerful model for data in network domain.

3 attr2vec

For the problem of learning the representation of nodes in a network, almost all recent efforts in network embedding are based on this intuition: nodes should have similar latent representations with its neighbors. However, the reason why a node exhibits property similar to its neighbors is because they are in a similar network environment, not whether they are close in distance. Moreover, in many real-world networks nodes and their neighbors do not show similarities, because sometimes they are not similar in degree although they are close in distance, or their neighbors are completely different sets of nodes. In summary, we believe that the attributes of node are the key to determining the role of node. So, we think a successful method should have these two properties:

- The similarity of the representations for nodes should depend on the similarity of the attributes of nodes. Therefore, nodes with identical attribute should have the same representations, while two nodes that have different attribute should be far apart.
- The representation of nodes should be independent of their position in the network. Thus, two nodes that have similar attribute but are far away in the network should also have similar representations.

Based on the above ideas we propose *attr2vec*, in the following part we make a detailed explanation of *attr2vec* at each step.

3.1 Generating Represent of Attributes

The first step of *attr2vec* is to generate K attribute vectors for each node. First of all, we assume that each node in the network has its own attribute. For example, in a citation network, the nodes are different articles, the edges are the citation relations between the articles. The attribute of the node are the key words in articles. It can be extracted in a unified way and represented by a series of numbers of equal length. We assume that the attribute of nodes is already in our train data. Now what we have to do is to generate K attribute vectors for each node. In particular, the k-th attribute vector of u is generated by averaging attributes of nodes that at distance exactly k from u in original network (u in a node in network).

Let $G = (V, E)$ denote the undirected, unweighted network under consideration with vertex set V and edge set E, where $n = |V|$ denotes the number of nodes in G, K is a hyper parameter and let $R_k(u)$ denote the set of nodes at distance (hop count) exactly $0 \leq k < K$ from u in G. Let *attribute(s)* denote attribute of s (s may be a node or a collection of nodes) and $attr\_vec_k(u)$ denote the k-th attribute vector of u. In particular, we define:

$$attr\_vec_k(u) = mean(attribute(R_k(u))) \tag{1}$$

where *mean(set)* means to average the target *set*. The above-described process of generating $attr\_vec_k(u)$ is shown in Fig. 1 (See Fig. 1).

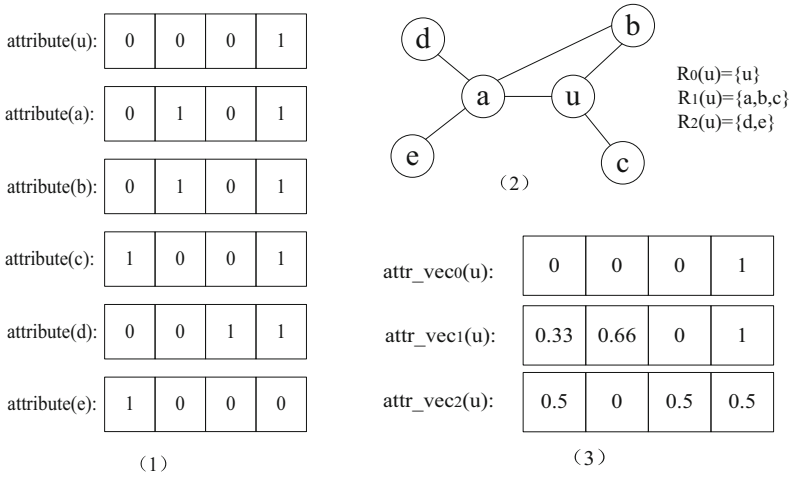

Fig. 1. Illustration of $attr\_vec_k(u)$ generation in *attr2vec*. The original attributes of nodes in network is shown in (1), and (2) is the structure of original network. Thus, we get $attr\_vec_0(u)$, $attr\_vec_1(u)$ and $attr\_vec_2(u)$ in (3).

3.2 Measuring Attribute Similarity

By comparing the $attr\_vec_k(u)$ and $attr\_vec_k(v)$ we can impose a hierarchy to measure similarity. In particular, we define:

$$D_k(u,v) = D_{k-1}(u,v) + \frac{g(attr\_vec_k(u), attr\_vec_k(v))}{\alpha_k}, k \geq 0 \tag{2}$$

where $g(attr\_vec_1, attr\_vec_2) \geq 0$ measures the distance between $attr\_vec_1$ and $attr\_vec_2$, α_k are hyper parameter for value scale adjustment to and $attr\_vec_{-1} = 0$. Note that by definition $D_k(u,v)$ is increasing and is defined only when at least one of u and v have nodes at distance k (If only u has nodes at distance k, then every element of $attr\_vec_k(v)$ is 0).

We also need to define a function g that calculates the distance between $attr\_vec_k(u)$ and $attr\_vec_k(v)$. Note that $attr\_vec_k(u)$ and $attr\_vec_k(v)$ are the same size and its elements are arbitrary decimals. We adopt Euclidean Distance to measure the distance between them, which a technique that can cope better with sequences of same sizes. We adopt the following distance function:

$$g(x,y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (3)$$

Note that when $x = y$ then $g(x, y) = 0$. Thus, the distance between two identical attribute vectors is 0.

3.3 Constructing the Context Graph

We construct a weighted multi-layer graph to save the similarity between attributes of the nodes. Each layer $k = 0, 1, \ldots, K-1$ of this weighted multi-layer graph is composed of all nodes in the network and the weighted edges between nodes. Thus, every layer is formed by a weighted undirected complete graph with node set V, and $n(n-1)$ edges. The edge weight between two nodes in a layer is given by:

$$W_k(u,v) = e^{-D_k(u,v)}, k = 0, \ldots, K-1 \quad (4)$$

Note that weights are inversely proportional to attribute distance, and assume values of $W_k(u, v)$ smaller than or equal to 1, it being equal to 1 only if $D_k(u, v) = 0$.

We connect the layers using directed edges as follows. Each vertex is connected to its corresponding vertex in the layer above and below. The edge weight between layers are as follows:

$$\begin{aligned} &W(u_k, u_{k+1}) = \log(\Gamma_k(u) + e), k = 0, \ldots, K-1 \\ &W(u_k, u_{k-1}) = 1, k = 1, \ldots, K \end{aligned} \quad (5)$$

Where $\Gamma_k(u)$ is number of edges incident to u that have weight larger than the average edge weight of the complete graph in layer k. In particular:

$$\Gamma_k(u) = \sum_{v \in V} \left(W_k(u,v) > \overline{W_k}\right)$$

$$\overline{W_k} = \sum\nolimits_{(u,v) \in} \binom{V}{2} \frac{W_k(u,v)}{\binom{n}{2}} \quad (6)$$

$\Gamma_k(u)$ represents the number of edges connected to u with a weight greater than the average weight of layer k. Note that the attribute information of farther neighbors is captured in the higher layer, so if there are many nodes similar to u in this layer, then you should go to the higher layer to find more refined similar nodes. And when going to the upper layer, there will be fewer similar nodes because there are more neighbors

participating in the calculation. Imagine that two nodes that are similar at the first layer become dissimilar after considering further neighbors, and two nodes that are not similar at the first layer are unlikely to be similar at the upper layer.

3.4 Generating Context for Nodes

In the previous work we constructed a multi-layer graph M and calculated the weights of it, M can be used to generate context for each node in the network. *Attr2vec* uses random sampling to generate context of each node, which is the sequence of nodes. In particular, we randomly sample L times on M according to the weight of the edge connected to u to generate a context of length L for u. Before each step, random sampling first determines whether need to change the layer or sample at current layer (with probability $q > 0$ the random sampling stays in the current layer, q is a hyperparameter).

Given that it will stay in the current layer, the probability of stepping from node u to node v in layer k is given by:

$$p_k(u, v) = \frac{e^{-F_k(u,v)}}{Z_k(u)} \tag{7}$$

where $Z_k(u)$ is the normalization factor for vertex u in layer k, simply given by:

$$Z_k(u) = \sum_{\substack{v \in V \\ v \neq u}} e^{-F_k(u,v)} \tag{8}$$

Note that random sampling tends to sample nodes that have similar attributes to the current node. If the attribute of two nodes are similar and the attributes of their neighbors are similar too, then they are likely to appear in the context of each other. Thus, a node is more likely to have nodes similar to it in its context, regardless of the position of the node in the original network.

3.5 Embedding with Skip-Gram

Skip-gram models have been widely used to learn word embedding and only require a collection of sentences to generate meaningful representations. Now we just need to input these sequences of node that we generated in previous section into skip-gram and the output are embeddings of nodes.

4 Experimental Evaluation

We tested our model in the following experiment, we used *attr2vec* to learn the latent representations for the nodes in the reference network, and then used these node representations for visualization and classification tasks. We also compared with the state-of-the-art techniques that learning potential representations for nodes.

4.1 Datasets

Cora: Data collected from the Collective classification in network data [12]. The dataset is a citation network dataset, it contains sparse bag-of-words keywords vectors for each document and a list of citation links between documents. The network has 2708 nodes, 5429 edges, and 7 different labels.

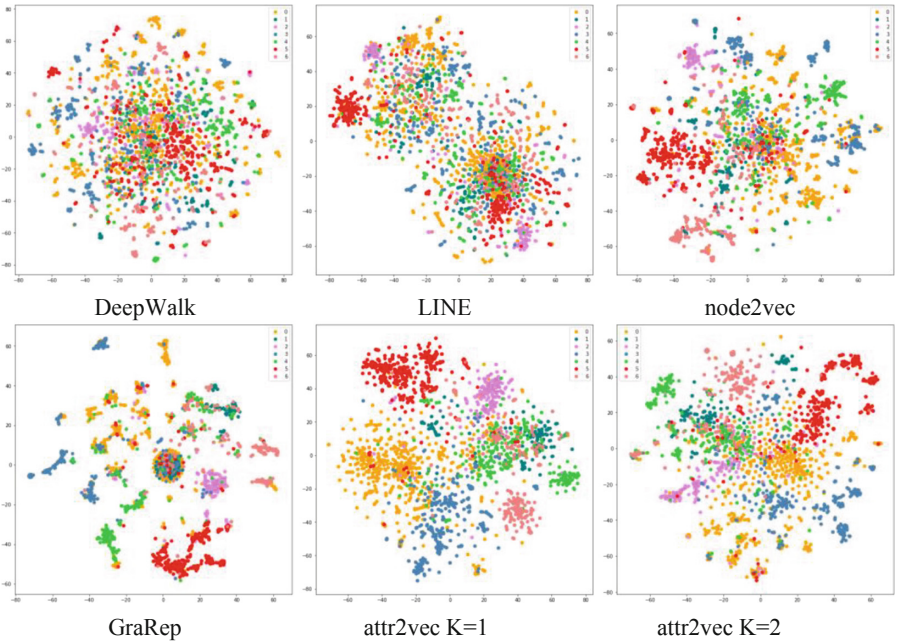

Fig. 2. Visualization of Cora network. Each point indicates one node. Color of a point indicates the label of node.

4.2 Visualization

An important application for network embedding is to visualize a network in a two-dimensional space. In this paper we visualized the representations learned from Cora network through different models. t-SNE [13] is currently the most popular method for dimension reduction and visualization of data. Therefore, we use the latent representations learned by different graph embedding models as the input to t-SNE. As a result, each node in Cora network is mapped as a two-dimensional vector. Then we can visualize each vector as a point on a two-dimensional space. For nodes labeled with different labels, we use different colors for corresponding points. Therefore, a good visualization result should be that the points with different color are separated by boundaries and the points with same color are clustered together. The visualization figure is shown in Fig. 2.

From Fig. 2, we can see that the visualization of DeepWalk and LINE [14] are not acceptable because the points with different labels are mixed with each other. For node2vec and GraRep, the clusters of points with same labels are formed. However, in the center part are still the points with different colors mixed together. Obviously, the visualization of *attr2vec* performs best in both the aspects of clustering same nodes and separate different nodes.

4.3 Classification

Another common application of latent representations for network nodes is classification. In this section, we first use *attr2vec* and some other state-of-the-art network embedding technologies to learn meaningful latent representations for Cora network, and use neural networks to divide these latent representations into training and test sets for classification tasks. The potential representation of the nodes becomes attributes, which are used to train a supervised neural network classifier (see Table 1). It can be seen that *attr2vec* achieved the best performance in both micro-f1 and macro-f1. In particularly, the performance of *attr2vec* at $K = 2$ is better than that at $K = 1$. It's a reasonable result because when $K = 2$ *attr2vec* extends the similarity of nodes itself to the neighborhood similarity of nodes, the model gets more effective information, so it can perform better. But in fact, when $K = 3$ the performance of the model does not improve, we think that's because when the model takes into account more and more distant neighbors, the noise is also increasing. So, K is not the bigger the better.

Table 1. Summary of results in terms of classification accuracy (in percent).

Method	Micro-F1	Macro-F1
DeepWalk	62.55	61.46
LINE	67.16	67.04
node2vec	73.25	72.78
GraRep	78.60	77.73
attr2vec $K = 1$	80.55	79.31
attr2vec $K = 2$	**82.94**	**81.91**

5 Conclusion

In this paper we propose *attr2vec*, a novel unsupervised network embedding algorithm that can capture network topology and node attributes. We show that attr2vec is powerful in finding similar nodes and learning the latent representation of node. Compared with the traditional DeepWalk and node2vec, it removes the limitation that it can only randomly walk on the original network and it can find similar nodes at long distances. At the same time, it expands the similarity of nodes itself to the similarity of the neighborhood, which can mine more potentially effective information. It can also learn the attribute information of nodes, instead of just learning the structure of the network, so that the nodes get a more meaningful representation.

Acknowledgments. This work was supported by the Qingdao Philosophy and Social Sciences Planning Project (QDSKL1801131), the 2018 Postgraduate Tutors'Guidance Ability Improvement Project of Shandong Province (84), the National Natural Science Foundation of China under Grant No.61701284, the Innovative Research Foundation of Qingdao under Grant No. 19–6-2–1-cg.

References

1. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(2579–2605), 85 (2008)
2. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social Network Data Analytics, pp. 115–148. Springer, Heidelberg (2011)
3. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
4. Mahoney, M.: Large text compression benchmark (2011).www.mattmahoney.net/dc/textdata
5. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In KDD (2014)
6. Grover, A., Leskovec, J.: node2vec: Scalable Feature Learning for Networks. In: ACM SIGKDD (2016)
7. Ribeiro, L.F.R., Saverese, P.H.P., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: KDD (2017)
8. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2016)
9. Bottou, L.: Stochastic gradient learning in neural networks. In: Proceedings of Neuro-Nimes 91, Nimes, France (1991)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
11. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE TNN **20**(1), 61–80 (2009)
12. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
13. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. JMLR **9**(2579–2605), 85 (2008)
14. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW (2015)