ISSN: 2710-0588

APTM: Structurally Informative Network Representation Learning

Pingke Zhang, Jinyu Huang*

School of Sichuan University of Science and Engineering, Zigong 643000, China

Abstract

Network representation learning algorithms provide a method to map complex network data into low-dimensional real vectors, aiming to capture and preserve structural information within the network. In recent years, these algorithms have found widespread applications in tasks such as link prediction and node classification in graph data mining. In this work, we propose a novel algorithm based on an adaptive transfer probability matrix. We use a deep neural network, comprising an autoencoder, to encode and reduce the dimensionality of the generated matrix, thereby encoding the intricate structural information of the network into low-dimensional real vectors. We evaluate the algorithm's performance through node classification, and in comparison with mainstream network representation learning algorithms, our proposed algorithm demonstrates favorable results. It outperforms baseline models in terms of micro-F1 scores on three datasets: PPI, Citeseer, and Wiki.

Keywords

Network Representation Learning; Graph Data Mining; Autoencoder.

1. Introduction

In recent years, significant breakthroughs have been achieved in various fields through the application of deep learning and Convolutional Neural Networks (CNNs)[1], such as machine translation [2], natural language processing (NLP) in computer vision (CV) [3], object detection [4], and image classification [5]. Beyond traditional data types like text, audio, images, and videos, real-world data includes graph data such as protein interaction networks, citation networks, and social networks, describing a set of entities and their relationships.Network representation learning algorithms have emerged as widely used methods in graph data mining tasks [6-9]. These algorithms map complex network data into low-dimensional real vectors, capturing and preserving the structural features of the network. These low-dimensional real vectors are also referred to as node embeddings or network representations. Research indicates that these algorithms have demonstrated significant success in tasks like node classification [6-7], link prediction [8], and recommendation systems [9] for graph data.In recent years, several efficient network representation learning algorithms have been developed, such as LINE [10], DeepWalk [11], Struc2Vec [12], and SDNE [13]. The LINE model optimizes first-order and second-order similarity measures between nodes to learn node embeddings [10]. First-order similarity refers to the similarity of connected node pairs, and second-order similarity refers to the similarity of node pairs with common neighbors. DeepWalk, inspired by the Word2Vec language model, performs random walks on the network, treating the resulting sequences as sentences and nodes as words [11]. These sequences are then input into the model for training to generate node embedding vectors, capturing semantic similarity between nodes. Struc2Vec, inspired by DeepWalk, focuses on learning the topological structure features of nodes' surroundings by utilizing multi-level structural information to gradually aggregate local and global structural features [12]. SDNE utilizes a deep autoencoder model to capture both local and global structure of the graph, generating node embedding

ISSN: 2710-0588

vectors [13]. Specifically, SDNE uses the adjacency matrix of the graph to obtain local and global structure information. Vector representations in the adjacency matrix, containing information about node neighbors, are optimized for reconstruction loss to capture global structural information. Additionally, non-zero elements in the adjacency matrix, representing edges, are used to optimize distance loss between connected node pairs, enhancing the model's ability to capture local structural information.Given that the neighborhood situations of nodes in a network vary, our optimization of traditional network representation learning algorithms introduces a novel adaptive transition probability matrix generation algorithm [10]. This is combined with deep neural networks for encoding and dimensionality reduction, effectively encoding the complex structural information of the network into low-dimensional real vectors.

2. Related Work

Autoencoder [14] is an unsupervised learning model in the field of deep learning, used for feature learning, dimensionality reduction, and data reconstruction. It consists of an encoder and a decoder, aiming to encode input data into a low-dimensional representation and reconstruct it back to the original data through the decoder. The encoder and decoder are typically structurally symmetric neural networks, and during training, autoencoders learn meaningful representations of data by minimizing reconstruction errors. Although autoencoders are primarily applied in computer vision [15], researchers have utilized them for graph clustering tasks.

The core idea of the Structural Deep Network Embedding (SDNE) model is to use autoencoders to learn node embeddings, transforming the structural information of the network into low-dimensional vector representations, preserving the similarity and relationships between nodes in the embedding space. SDNE takes the adjacency matrix as input to a neural network with an encoder-decoder structure. The decoder's output layer calculates the second-order similarity loss function, preserving the second-order similarity of nodes while reducing the dimensionality of the adjacency matrix. This ensures that nodes with similar neighbors are closer in the embedding space. Simultaneously, using the output layer of the encoder, the first-order similarity loss function is computed to preserve the first-order similarity of nodes, i.e., nodes connected by an edge are considered similar. By jointly optimizing the loss functions for first-order and second-order similarity, the model generates network representations that retain both types of similarity. In general, the loss function of an autoencoder is as follows:

$$Loss = \sum_{i=1}^{n} || \overline{x}_i - x_i ||_2^2 \tag{1}$$

where $\overline{x}_i$ is the reconstructed data from the decoder, and x_i is the input data.

SDNE takes into account that real-world networks are generally sparse, meaning the adjacency matrix of a sparse network has far more zero elements than non-zero elements. Directly using the autoencoder's loss function would result in reconstructing many zero elements. Therefore, the model adds a penalty coefficient for reconstructing non-zero elements. The second-order similarity loss function of the model is defined as:

$$Loss_{2nd} = \sum_{i=1}^{n} || (\overline{x}_i - x_i) \odot b_i ||_2^2 \tag{2}$$

ISSN: 2710-0588

where b_i is a vector of the same dimension as x_i, with each element being 1 for each zero element in b_i, and β is the penalty coefficient for non-zero elements.

By using an improved autoencoder with the adjacency matrix as input and ensuring the second-order similarity loss function, vertices with similar neighborhood structures will be mapped close in the representation space. Since the neighbors around a node can reflect its global position in the network, the second-order similarity loss function is beneficial for preserving the global structure of the network. SDNE uses the first-order similarity to represent the local network structure, and the first-order similarity loss function is defined as:

$$Loss_{1st} = \sum_{i,j=1}^{n} s_{i,j} \| y_i - y_j \|_2^2 \tag{3}$$

where $s_{i,j}$ represents the value in the i-th row and j-th column of the adjacency matrix, and y_i and y_j represent the vectors obtained by encoding the nodes i and j, respectively. In the network, the presence of a link between two nodes indicates a certain first-order similarity.

SDNE optimizes the overall structure and local structure of the network by linearly weighting the two loss functions. The complete loss function of the model is as follows:

$$Loss_{mix} = \alpha \sum_{i,j=1}^{n} s_{i,j} \| y_i - y_j \|_2^2 + \sum_{i=1}^{n} \| (\overline{x}_i - x_i) \odot b_i \|_2^2 + L_{reg} \tag{4}$$

where L_{reg} is the L2 norm regularization term to prevent overfitting, and α and β are two tunable hyperparameters used to adjust the weights of the first-order and second-order similarity in the loss function. By adjusting these parameters, the model can handle diverse network structures, such as sparse networks where edge and neighbor information is limited, by appropriately increasing α and β to help the model better capture local and global features around nodes.

3. Algorithm Implementation

This can be considered an extension of SDNE. The original model takes an adjacency matrix as input, where each vector in the adjacency matrix only contains the first-order neighbors of nodes. Modeling neighbor similarity between nodes is done solely through the first-order neighbors of each node. This approach results in the loss of higher-order neighbor information around nodes. To enable the model to generate node embeddings that contain more structural features, we propose the use of an Adaptive Transition Probability Matrix (ATPM) to replace the adjacency matrix, thereby capturing a more complete node neighborhood structure.

We describe the structural characteristics of the network using the ATPM, which is an n×n symmetric matrix. In ATPM, the value of the element at the i-th row and j-th column is calculated using the formula as follows:

$$S(i,j) = \sum_{1}^{k} P(i,j)^k + \sum_{1}^{k} P(j,i)^k \tag{5}$$

where $P(i,j)^k$ represents the probability of transitioning from node i to node j after traversing k edges.

ISSN: 2710-0588

3.1. ATPM Construction Strategy

In a network, the connectivity between two nodes represents the existence of a path that can be used to travel from one node to another. If two nodes are connected, it implies that there is a series of edges or connections through which one can reach the other. Therefore, connected node pairs have transition probabilities between them.

For a node a in the network, computing the transition probabilities for all nodes connected to a would require a significant amount of computational resources and produce a lot of unnecessary probability information.

To address this, we have designed an ATPM construction strategy based on the minimum transition probability value p_min. Firstly, initialize an n×n matrix X with all elements set to zero. For a node a in the network, we employ a depth-first search to gradually explore the nodes around a and calculate the transition probability p between a and each node visited during the search. With each computed probability p, the corresponding element in matrix X is incremented by p. For instance, if there is a 1st-order transition probability from a to c, the value at the position of a-row and c-column in matrix X will increase by $P(a,c)^1$. If there is a 2nd-order transition probability $P(a,c)^2$ from a to c and if $P(a,c)^2$ is greater than p_min, then the value of $X(a,c)$ will further increase by $P(a,c)^2$.

During the depth-first search, if the transition probability from a to a higher-order neighbor falls below p_min, the search will not continue deeper. This method of constructing the transition probability matrix allows us to obtain important k-order neighbor information around nodes while reducing computational complexity. In a network, the surrounding structure of nodes can be either sparse or dense. ATPM can capture transition probabilities for more distant nodes in sparse structures and detailed first and second-order neighbor transition probabilities in dense structures. Compared to the adjacency matrix, ATPM can reflect a richer set of node neighborhood structural features.

4. Experimental Analysis:

4.1. Dataset Introduction

We used three widely used datasets, Citeseer, PPI, and Wiki, to demonstrate the effectiveness of the ATPM algorithm. Citeseer is a citation network dataset where nodes represent academic papers, edges represent citation relationships between papers, and labels denote the research field of the papers. PPI is a protein-protein interaction network dataset where nodes represent proteins, edges represent interactions between proteins, and labels represent the biological state of the proteins. Wiki is a co-occurrence word network dataset extracted from the first one million bytes of text in Wikipedia. In the Wiki dataset, nodes represent words, edges represent pairs of words that co-occur in the same sentence, and labels denote the part of speech of the words. The statistics of the datasets are as shown in Table 1.

Table 1. Network Datasets

Dataset	Number of Nodes	Number of Edges	Number of labels
Citeseer	3312	4732	6
PPI	3890	38739	50
Wiki	4777	92517	40

4.2. Experimental Evaluation Metrics

To evaluate the effectiveness of the model, we used node classification tasks to assess the quality of network representations, which is a common task for evaluating network embeddings [11-12]. We employed Micro-F1 as the evaluation metric, which is a variant of the

ISSN: 2710-0588

F1 score, primarily used for dealing with imbalanced datasets. The F1 score is the harmonic mean of precision and recall and is calculated as follows:

$$Precision = \frac{\sum_{A \in C} TP(A)}{\sum_{A \in C} (TP(A) + FP(A))}$$

$$Recall = \frac{\sum_{A \in C} TP(A)}{\sum_{A \in C} (TP(A) + FN(A))}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where $TP(A)$ represents true positive samples, which are the samples correctly predicted as positive by the model. $FP(A)$ represents false positive samples, which are negative samples incorrectly predicted as positive by the model, and $FN(A)$ represents false negative samples, which are positive samples incorrectly predicted as negative by the model. When calculating Micro-F1, all true positives, false positives, and false negatives are summed up, and then the overall precision and recall are computed to obtain the Micro-F1 score. Micro-F1 score places more emphasis on the performance of each individual sample and is less affected by class imbalance.

4.3. Experimental Results Analysis

In this study, LINE, Struc2vec, and SDNE were selected as baseline models. The parameters for each method were set according to the values proposed in their respective papers, and the dimension of the generated embedding vectors was uniformly set to 100.

Due to the introduction of the parameter p_min. In APTM to control the minimum value of transition probabilities to be obtained, the optimal values of p_min may vary for different datasets. This paper determined the optimal p_min values for each dataset through extensive experiments. The optimal p_min for the Citeseer dataset is approximately 0.005, for the PPI dataset is approximately 0.01, and for the Wiki dataset is approximately 0.02. The micro-f1 scores comparison for different algorithms on these three datasets is illustrated in Figures 1 to 3, and detailed data is presented in Tables 2 to 4.

Table 2. Micro-f1 score comparison of Wiki data set

Training set	10%	30%	50%	70%	90%
LINE	0.3655	0.3578	0.3596	0.3656	0.3737
Struc2Vec	0.2568	0.2823	0.2965	0.3044	0.3079
SDNE	0.4418	**0.4942**	**0.5057**	**0.5093**	0.5129
ATPM	**0.4668**	0.4892	0.4978	0.5073	**0.5135**

Table 3. Micro-f1 score comparison of Wiki data set

Training set	10%	30%	50%	70%	90%
LINE	0.1071	0.1274	0.1398	0.1528	0.1659
Struc2Vec	0.1372	0.1674	0.1821	0.1893	0.1977
SDNE	0.1307	0.1531	0.1719	0.1793	0.1916
ATPM	**0.1397**	**0.1731**	**0.1899**	**0.2013**	**0.2121**

ISSN: 2710-0588

Table 4. Micro-f1 score comparison of Wiki data set

Training set	10%	30%	50%	70%	90%
LINE	0.2728	0.3011	0.3078	0.3263	0.3266
Struc2Vec	0.4308	0.4686	0.4811	0.4907	0. 5011
SDNE	0.4687	0.5109	0.5215	0.5289	0.5212
ATPM	**0.4826**	**0. 5269**	**0. 5311**	**0.5394**	**0.5521**

5. Conclusion

We addressed the limitation of some network representation learning algorithms in learning global structural information, which led to poor performance of the generated node embeddings in terms of classification accuracy. To optimize this issue, we introduced an adaptive transition probability matrix to enhance the global structural information for each node, thereby generating node embeddings with more distinctive global structural features. Experimental results indicate that the improved algorithm exhibits varying degrees of enhancement in classification accuracy across three datasets from different domains, demonstrating its positive practical applicability.

Acknowledgments

Natural Science Foundation.

References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[2] Wu, Y., Schuster, M., Chen, Z., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

[3] Yu, A. W., Dohan, D., Luong, M. T., et al. (2018). QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In International Conference on Learning Representations.

[4] Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. Advances in neural information processing systems, 26.

[5] Marino, K., Salakhutdinov, R., & Gupta, A. (2017). The More You Know: Using Knowledge Graphs for Image Classification. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 20-28). IEEE Computer Society.

[6] Huang, L., & Yang, C. M. (2022). GeNI: A Network Representation Learning Method for Node Classification. Journal of Southwest University of Science and Technology (Natural Science Edition), 37(4), 63.

[7] Chen, S. C., Yuan, D. Y., Huang, S. H., et al. (2022). Node Label Classification Algorithm Based on Structural Deep Network Embedding Model. Computer Science, 49(03), 105-112.

[8] Ai, C. L., He, M., Lv, L., et al. (2023). Edge Embedding Link Prediction Algorithm Based on Comprehensive Random Walk Strategy. Journal of Yunnan University (Natural Science Edition), 45(01), 29-37. DOI:10.15888/j.cnki.csa.009068.

[9] Zhang, X. X., Tang, Y. Q., Zhao, W., et al. (2023). Personalized Learning Resource Recommendation Based on Knowledge Graph and Graph Embedding. Journal of Computer Systems & Applications, 32(05), 180-187. DOI:10.15888/j.cnki.csa.009068.

[10] Tang, J., Qu, M., Wang, M., et al. (2015). Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web (pp. 1067-1077).

ISSN: 2710-0588

[11] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).

[12] Ribeiro, L. F. R., Saverese, P. H. P., & Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 385-394).

[13] Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1225-1234).

[14] Ng, A. (2011). Sparse autoencoder. CS294A Lecture notes, 72(2011), 1-19.

[15] Lore, K. G., Akintayo, A., & Sarkar, S. (2017). LLNet: A deep autoencoder approach to natural low-light image enhancement. Pattern Recognition, 61, 650-662.