Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5537651, 13 pages
https://doi.org/10.1155/2021/5537651

Research Article

Link Prediction and Node Classification Based on Multitask Graph Autoencoder

Shicong Chen,[1] **Deyu Yuan**,[1,2] **Shuhua Huang**,[1,2] **and Yang Chen**[3]

[1]*School of Information and Cyber Security, People's Public Security University of China, Beijing 100038, China*
[2]*Key Laboratory of Safety Precautions and Risk Assessment, Ministry of Public Security, Beijing 100038, China*
[3]*School of Public Administration, Nanjing University of Finance & Economics, Nanjing 210023, China*

Correspondence should be addressed to Deyu Yuan; yuandeyu@ppsuc.edu.cn

Received 3 February 2021; Revised 23 March 2021; Accepted 5 April 2021; Published 19 April 2021

Academic Editor: Lihua Yin

The goal of network representation learning is to extract deep-level abstraction from data features that can also be viewed as a process of transforming the high-dimensional data to low-dimensional features. Learning the mapping functions between two vector spaces is an essential problem. In this paper, we propose a new similarity index based on traditional machine learning, which integrates the concepts of common neighbor, local path, and preferential attachment. Furthermore, for applying the link prediction methods to the field of node classification, we have innovatively established an architecture named multitask graph autoencoder. Specifically, in the context of structural deep network embedding, the architecture designs a framework of high-order loss function by calculating the node similarity from multiple angles so that the model can make up for the deficiency of the second-order loss function. Through the parameter fine-tuning, the high-order loss function is introduced into the optimized autoencoder. Proved by the effective experiments, the framework is generally applicable to the majority of classical similarity indexes.

1. Introduction

Nowadays, with the explosive growth of network data, the mainstream network representation learning algorithms are gradually difficult to adapt to the intricate data types. A variety of approaches were proposed to address privacy [1] and security [2] issues. The network is the carrier of the sophisticated relationships between data. Taking social networks as an example, large websites such as Twitter and Facebook have been consistently developing for a long time so that they can possess millions of online users. The user information scale is enormous, and the network structure is rather intricated. Thus, a mass of relationships between online users are worth exploring. By capturing the structural characteristics of real-world networks, experts and scholars can deal with multiple data analysis tasks efficiently, such as community detection [3], link prediction [4, 5], and node classification [6]. The emergence of network representation learning [7, 8] technology is of vital significance to social network analysis.

In the field of link prediction based on the classical similarity index, the CN [9] index calculates the number of common neighbors to predict the potential links between node pairs. The AA [10] index imposes a penalty on lower-connected neighbors. The Jaccard [11] index measures the similarity by comparing the proximities and differences between sample sets of common neighbors. The LP [12] index introduces the influencing factor of a third-order local path to the algorithm. The Katz [13] index improves the prediction accuracy by optimizing the LP index, by which it comprehensively extends the local path to the global path.

Motivated by Natural Language Processing [14], lots of network representation learning algorithms based on the Continuous Bag-of-Word model and Random Walk have gradually appeared. Essentially, it is a network mapping technique that each node is uniquely represented in form of low-dimensional vectors. By measuring the similarities between embedding vectors, these latent representations are probably to find the potential correlations between different entities denoted by nodes. Specifically, the low-dimensional space

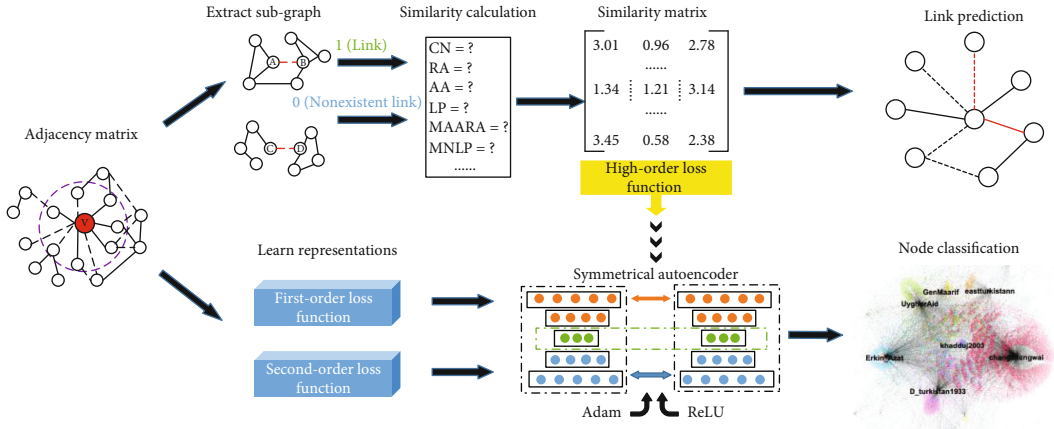

Figure 1: Architecture of our proposed multitask graph autoencoder.

can visualize the potential links in the complex network that are hard to be observed. Network representation learning not only is broadly employed to handle sophisticated social network tasks but also can be parallelized to reduce computational time.

Perozzi et al. [15] utilized the Random Walk mechanism to traverse all the network nodes deeply and preferentially. Given the initial node and walk step size, the algorithm samples a neighbor node as the next access node at random and then constitutes node access sequences of specified length in order so as to express the cooccurrence relation between nodes. After obtaining associated sampling data, the algorithm inputs sampling data into the skip-gram model for training, and the neighborhood structure of discrete nodes is then represented by vectors. Struc2vec [16] redefines node similarity from the perspective of a spatial structure. The algorithm constructs the weighted hierarchy graph by computing the node pair distances in different layers. Eventually, it leverages the generated node sequences that are structurally similar to learn network representations. Tang et al. [17] use the gradient descent method to separately optimize the first-order proximity and the second-order proximity. During the process of training, Tang et al. apply the negative sampling [18] method to decrease the time complexity.

Here, the contributions of our paper are demonstrated as follows: (1) We propose a new link prediction algorithm of mixed local neighbor and path, namely, MLNP. (2) For the deficiency of loss functions in structural deep network embedding (SDNE), our work establishes an architecture of multitask graph autoencoder (MTGAE), which designs a framework of high-order loss function from the perspective of capturing the similarity information. (3) We confirm the universal effectiveness of the loss function framework on different datasets. The specific model flow chart is shown in Figure 1.

2. Related Works

2.1. Autoencoder. As a special form of feedforward neural network, an autoencoder [19–22] is often used for dimensionality reduction feature learning in a graph embedding field. Let R^N be an N-dimensional adjacency matrix representing a graph network as input and $x_i \in R^N$ be an adjacency vector comprised of the local neighborhood structure information. The autoencoder consists of two components: the encoder $g(x_i)$: $R^N \to R^D$ and the decoder $f(y_i)$: $R^D \to R^N$. Specifically, it maps the adjacency vector to the low-dimensional embedding space composed of several nonlinear functions and acquires the approximate representation vector by effective way of compressing the graph-structured data. Then, we decode the embedding vector and represent it as the reconstruction vector $\hat{x}_i$. During the backward pass, the reconstruction loss error between the input and the output is minimized by adjusting the weight matrix cyclically. The representation vectors of latent space for different layers are computed as follows:

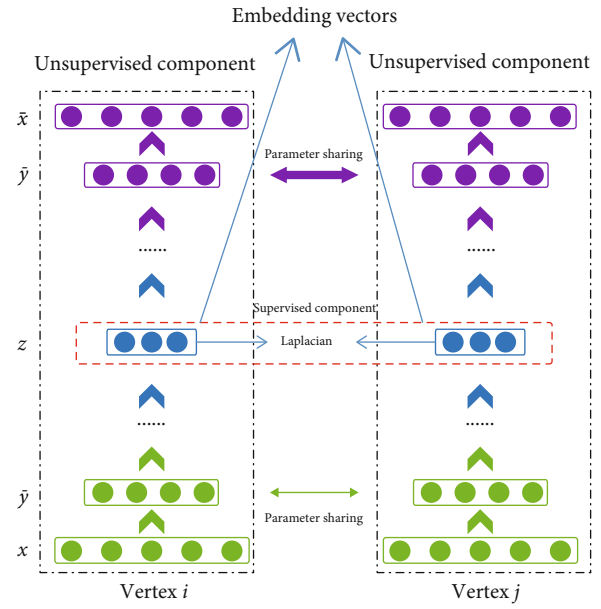

Figure 2: Traditional structural deep network embedding model.

$$y_i^{(k)} = \sigma\left(W^{(k)} y_i^{(k-1)} + b^{(k)}\right), \quad k = 2, 3, 4, \cdots, K, \tag{1}$$

Mixed local neighbor and path.
Input: edge list
Output: similarity matrix, AUC score
1: Input adjacency matrix A
2: Divide all edges into the training set and probe set
3: Construct the third-order path matrix with attenuation parameters αA^3
4: Construct optimized common neighbor matrix S_{MAARA}
5: Construct matrix based on the method of preferential attachment $S_{\mathrm{PA}} = k_x \times k_y$
6: Calculate the similarity matrix that incorporates multiple methods $S_{\mathrm{MNLP}} = S_{\mathrm{MAARA}} * S_{\mathrm{PA}} + \alpha A^3$
7: Calculate the AUC score of the MLNP index

Algorithm 1:

Multitask graph autoencoder.
Input: the network $G = (V, E)$ with adjacency matrix M, node labels, the parameters $\alpha, \beta, \gamma, \nu$
Output: network representation Y and updated parameter θ
1: Apply Adam optimizer and ReLU activation function
2: Construct the similarity matrix M
3: $X = A$
4: **Repeat**
5: Based on X, apply Equation (1) to obtain $\widehat{X}$ and $Y = Y^K$
6: $\mathrm{Loss}_{\text{high-order}} = \left\| (\widehat{X} - X) \odot M * \gamma \right\|_F^2$
7: $\mathrm{Loss}_{\mathrm{2nd}} = \left\| (\widehat{X} - X) \odot B \right\|_F^2$
8: $\mathrm{Loss}_{\mathrm{1st}} = 2\mathrm{tr}(Y^T L Y)$
9: $\mathrm{Loss}_{\mathrm{mix}} = \alpha \mathrm{Loss}_{\mathrm{1st}} + \mathrm{Loss}_{\mathrm{2nd}} + \mathrm{Loss}_{\text{high-order}} + \nu \mathrm{Loss}_{\mathrm{reg}}$
10: Use $\partial L / \partial \theta$ to backpropagate through the whole network to obtain the parameter θ
11: **Until** converge
12: Obtain the network representations $Y = Y^K$

Algorithm 2:

where $W^{(k)}$ is the weight matrix of the kth layer, $y_i^{(k-1)}$ is the $(k-1)$th layer latent vector, $b^{(k)}$ is the biases of the kth layer, and $\sigma(\cdot)$ denotes the sigmoid nonlinear activation function.

2.2. Structural Deep Network Embedding. In 2016, Wang et al. [23] put forward a structural deep network embedding model in two aspects. The first-order proximity captures local structure features of the network by judging whether nodes are linked by a direct edge [24], which can be thought of as the supervised component. Meanwhile, the second-order proximity preserves global structure features by observing the differences between the neighborhood structure of nodes, which can be regarded as the unsupervised component. Two concepts of proximity describe the characteristics of the network structure from complementary viewpoints. The SDNE model gives weights to the first-order and second-order proximity loss functions for iterative optimization, respectively. The SDNE architecture is shown in Figure 2.

The first-order loss function makes the corresponding embedding vectors of adjacent nodes $y_i^{(k)}$ and $y_j^{(k)}$ approximate in embedding spaces. The objective function is calculated as follows:

$$\mathrm{Loss}_{\mathrm{1st}} = \sum_{i,j=1}^{n} s_{i,j} \left\| y_i^{(k)} - y_j^{(k)} \right\|_2^2 = 2\mathrm{tr}\left(Y^T L Y\right), \tag{2}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace, $s_{i,j}$ is the element of the adjacency matrix, L is the Laplace vector matrix, and Y is the encoded vector matrix of the hidden layer.

Intuitively, the second-order proximity compares the neighborhood structure of node pairs, and the proximity is computed as follows:

$$\mathrm{Loss}_{\mathrm{2nd}} = \sum_{i=1}^{n} \left\| (\widehat{x}_i - x_i) \odot b_i \right\|_2^2 = \left\| \left(\widehat{X} - X\right) \odot B \right\|_F^2, \tag{3}$$

where $\widehat{x}_i - x_i$ is the reconstruction error, $\odot$ denotes the Hadamard product, and b_i is a penalty coefficient, where $b_i = \{b_{i,j}\}_{j=1}^{n}$. If $s_{i,j} = 0$, $b_{i,j} = 1$; otherwise, $b_{i,j} = \beta > 1$. Affected by the sparsity of the network, the quantity of zero elements in the adjacency matrix is far more than that of nonzero elements. We assume that the adjacency matrix is directly

addressed as the input of SDNE; it is simpler to reconstruct the zero elements. However, this is not in accordance with our previous expectations, and a reasonable solution is to impose a higher penalty coefficient β on the reconstruction error of nonzero elements. The ultimate goal of the SDNE model is to jointly optimize the proximity loss functions, and the integral loss function is shown in

$$\text{Loss}_{\text{mix}} = \alpha\text{Loss}_{1\text{st}} + \text{Loss}_{2\text{nd}} + \nu\text{Loss}_{\text{reg}}, \tag{4}$$

where Loss_{reg} denotes the regularization term to avoid overfitting. Because of the robustness of the sparse network, performances of overall optimization are hardly affected by variations of parameters α and β.

3. Proposed Link Prediction Algorithm

In this paper, we innovatively propose an MLNP link prediction algorithm that integrates methods of common neighbors, high-order path, and preferential attachment. We adjust the structural factors of the LP index by weighing prediction accuracy against computational efficiency. The calculation method is shown in

$$s_{xy} = \sum_{z\in\Gamma(x)\cap\Gamma(y)} \left(\frac{1}{\log|\Gamma(z)|}\right)^2 \times k_x \times k_y + \alpha A^3, \tag{5}$$

where A is the adjacency matrix, α is the attenuation parameter, and k_x and k_y denote the degrees of pairwise nodes. More importantly, $\Gamma(\cdot)$ means the neighbor nodes. By utilizing the MAARA matrix based on the AA index and RA [25] index, we highlight the importance of nodes with tremendous influence. In specific, the algorithm enhances the contribution of nodes with higher degree centralities to similarity and weakens the contribution of nodes with lower degree centralities to similarity. We distinguish common neighbors with different degree centralities to reflect the correlations between pairwise nodes more accurately. The node similarity is calculated as follows:

$$S_{xy} = \sum_{z\in\Gamma(x)\cap\Gamma(y)} \left(\frac{1}{\log|\Gamma(z)|}\right)^2. \tag{6}$$

According to the theory of preferential attachment [12], the probability of potential links between the central node and other neighbor nodes is directly proportional to the degree centrality of the central node. Furthermore, the likelihood one link connecting pairwise nodes v_x and v_y is also directly proportional to $k_x \times k_y$. To summarize, the Hadamard product of the reconstructed MAARA matrix and PA matrix compresses the local neighborhood information so that we can thoroughly take the properties of nodes themselves, the number, and influence of common neighbors into consideration.

The above method conducts structural optimizations for the common neighbor index and explains its superiority from the theoretical level. Inspired by the idea of the global

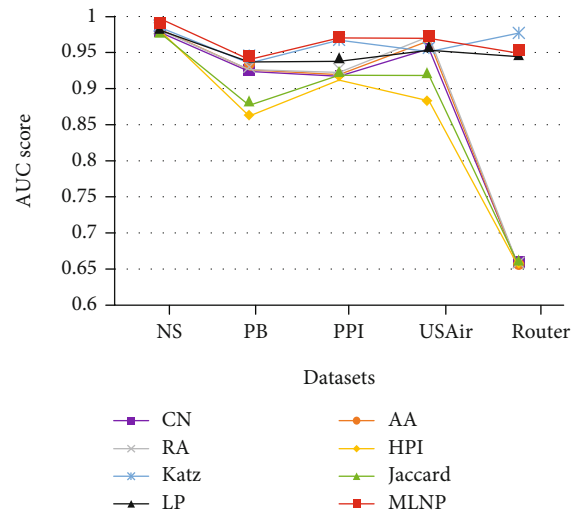

Figure 3: AUC score of different link prediction algorithms on five datasets.

Table 1: Statistics of node classification datasets.

Network	Node	Edge	Label	Average degree	Average path length
Europe-flight	399	5995	4	30.1	2.28
Brazil-flight	131	1074	4	16.4	2.71

path, as the number of intermediate nodes in local paths increases, the weight parameter of the high-order path will decay. Intuitively, the number of second-order paths is equal to the number of common neighbors that have been discussed, indicating that the weight of the third-order path is the highest. Hence, our work innovatively introduces the factor of third-order path combined with the above matrices into the ultimate similarity matrix so as to produce a substantial boost on prediction accuracies. The basic algorithm procedure is shown in Algorithm 1.

4. Multitask Graph Autoencoder

4.1. High-Order Loss Function. The deficiency in second-order proximity of the SDNE model is explained as follows: When imposing a penalty coefficient β on nonzero elements, the only criterion for measuring similarities is whether an edge exists between pairwise nodes. Factually, the properties of common neighbors, the length of paths, and even the attenuation parameters will bring about deviations in the process of computing similarities. The adjacency matrix only describes the actual condition, while the similarity matrix reveals the hidden structural similarity of the network. For instance, a couple of individuals who have more common friends are more likely to establish friendships, even though they do not get acquainted with each other before. In network topology, we can directly observe the explicit links but may

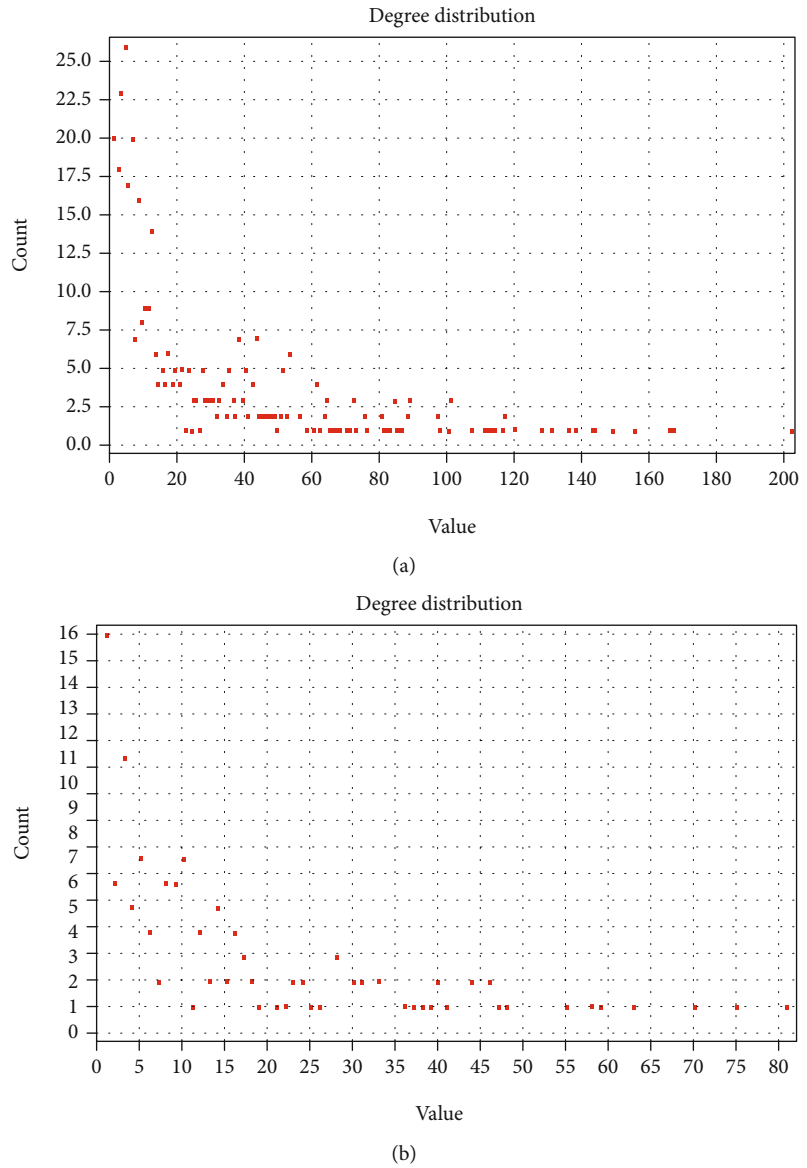

Figure 4: Degree distribution of (a) Europe-flight network and (b) Brazil-flight network.

ignore the potential links simultaneously. Thus, the idea of applying the adjacency matrix only is single that seeking the potential links inferred by the algorithm is the key to lifting the capability of our model.

The high-order proximity and second-order proximity are complementary in that they, respectively, punish matrix elements according to the explicit similarity and the hidden similarity of the network structure. By using the backpropagation algorithm, we cyclically minimize the introduced high-order loss function error. In detail, the reconstructed high-order loss function is defined as follows:

$$\text{Loss}_{\text{high-order}} = \sum_{i=1}^{n} \left\| (\widehat{x_i} - x_i) \odot M * \gamma \right\|_2^2, \tag{7}$$

where M is the similarity matrix and γ is the adjustment parameter. Parameter γ directly controls the fluctuation range of similarity and constrains the reconstruction weight. We believe that γ should be consistent with β (1-20), or the different loss functions will exhibit extreme imbalance. Our model has its advantages in addressing the tasks of link prediction and semisupervised node classification at the same time. In specific, we borrow the idea of link prediction, which takes the output similarity matrix as an intermediate product, and then, we input the processed vector matrix into a stacked autoencoder.

4.2. Optimization of Autoencoder. In our experiment, we use the Keras [26] module to implement two layers of encoder and decoder at the CPU-enabled Tensorflow [27] backend. The hidden layer dimensionality of our model architecture is fixed at N-256-128-256-N. Due to the abandonment of the deep belief network [28] structure for parameter pretraining, the SGD optimizer and Sigmoid activation function applied by the original SDNE algorithm may lead to the

cessation of training. Alternatively, our architecture attempts to apply the Adam [29] algorithm with a fixed learning rate and ReLU [30] activation function for optimization.

The Adam optimizer has the characteristics of inertia retention and environmental perception. The method of calculating a new round of gradient descent is the linear weighting of the current real gradient with the gradient used in the previous round for gradient descent. The superiority of its adaptive learning efficiency lies in overcoming the network sparsity problem effectively. Compared with the SGD optimizer which is easy to converge to the local optimum and trapped in the saddle point, the Adam optimizer is recognized for accelerating the convergence speed and maintaining the convergence stability. However, the adaptive learning rate algorithm of the Adam optimizer performs worse in the fields of object recognition and syntax component analysis. In the deep neural network (DNN), the gradient of the Sigmoid activation function is very small at a position away from point 0. During the backpropagation phase, the information loss problem caused by gradient disappearance may occur, and computation of the partial derivative involved with division may increase the time complexity of the algorithm. ReLU activation function, however, can effectively alleviate this type of vanishing gradient issue and perform well in enhancing computational efficiency. To summarize, the complete algorithm is shown in Algorithm 2.

5. Link Prediction Experiments

5.1. Datasets and Evaluation Metrics. For link prediction, we evaluate our MLNP algorithm on five classical graph-structured datasets. The fundamental information of datasets is introduced as follows.

NS [31] is a collaboration network of scientists who have published distinguished papers on the topic of complex networks. An observed link is present if there is a cooperative relationship between scientists in papers. PB [32] is an American political blog network that documents the links between blogs extracted from network websites. PPI [33] is a protein-protein interaction network. The nodes denote macromolecules of proteins, and the links indicate the interactions between a couple of proteins. USAir [34] is an aviation network of the United States that each node corresponds to a termination. If there is a direct air route between terminations, it means that there is a connection between nodes. Router is a router [35] network on the Internet, where nodes denote routers and edges directly connect the two routers for packet exchange through optical fiber or other means.

Our work adopts the most widely used AUC score to evaluate our proposed MLNP algorithm on link prediction tasks. It can be explained as the likelihood that the randomly selected test links score higher than stochastically selected nonexistent links. In contrast to the *precision@k* evaluation indicator, the AUC score overall measures the prediction accuracy. It is defined as follows:

$$\mathrm{AUC} = \frac{n' + 0.5n''}{n}. \tag{8}$$

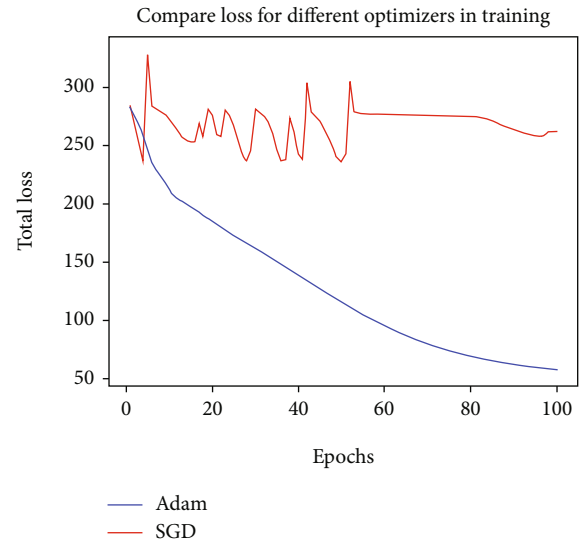

Figure 5: Comparison of loss convergence between Adam optimizer and SGD optimizer.

Table 2: Parameter settings.

Epoch	α	β	l1	l2
100	$1e-4$	5	$1e-6$	$1e-5$

Among n times of experimental comparisons, n' denotes the occurrences of missing links that score higher than nonexistent links, while n'' denotes the occurrences of having the same score.

5.2. Result Analysis. To guarantee a more fine-grained comparison, we empirically choose 90% links at random as the training set, and the remaining 10% links constitute the probe set for prediction. We summarize the consequences of link prediction for five datasets in Figure 3.

In comparison to other strong baselines [36], the experiment results explicitly show that the formulated MLNP algorithm consistently achieves the best AUC performance on three datasets {NS, PB, and PPI}, respectively, 1.24%, 0.33%, and 0.3% higher than the best baseline. Although the prediction accuracy of our method is slightly 0.24% lower than the RA index on the USAir dataset and 2.78% lower than the Katz index on the Router dataset, it remains competitive compared with the rest of the similarity indexes.

On small-scale datasets, we explicitly observe that our method outperforms all other baselines, even exceeding the Katz index based on the global path. To our surprise, the prediction accuracy reaches 99.7% on the NS dataset. However, the formulated algorithm gets worse AUC performances than the RA index and Katz index on large-scale datasets. The possible reasons are twofold. Firstly, with the increase of diameter and average path length of the network, it is far from enough that the MLNP algorithm only captures local information. Secondly, the Katz index preserves the global structures adequately by traversing the network. The experiments reveal that

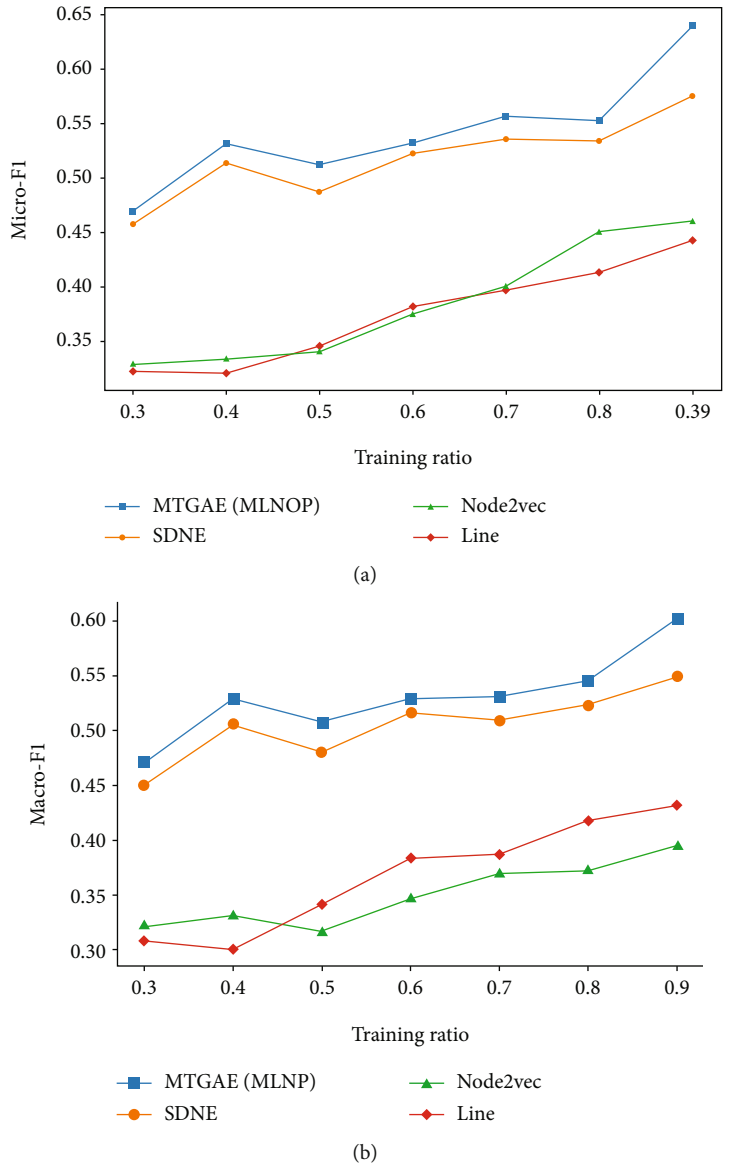

Figure 6: (a) Micro-F1 and (b) Macro-F1 of MTGAE on Europe-flight dataset.

the MLNP algorithm is quite effective for optimization of the original similarity index. We attribute the efficacy of our innovation to multiple integrated methods.

6. Node Classification Experiments

6.1. Datasets and Evaluation Metrics. We select two air transportation networks of Europe-flight and Brazil-flight to assess the effects of representations. Specifically, the dataset contents comprise nodes, links, and node labels that 399 nodes and 5995 links exist in the Europe-flight network, and 131 nodes and 1074 links exist in the Brazil-flight network. Both datasets divide node labels into four categories, and the detailed statistics of network attributes are computed in Table 1.

To ensure that the adopted similarity theory can traverse the network locally and globally, we calculate the degree distribution of nodes as well. According to the simulation consequences, although the quantity of network nodes decreases, the structure information is adversely more intact due to the relatively high link density and average node degree. The exact degree distributions of datasets are shown in Figure 4.

Empirically, we employ the current popular F1-measure indicator [367] to evaluate the quality of graph embedding representations, and the calculation method is defined as follows:

$$\begin{aligned}\text{precision} &= \frac{\sum_{A\in C}\text{TP}(A)}{\sum_{A\in C}(\text{TP}(A)+\text{FP}(A))},\\ \text{recall} &= \frac{\sum_{A\in C}\text{TP}(A)}{\sum_{A\in C}(\text{TP}(A)+\text{FN}(A))},\\ \text{F1-measure} &= \frac{2*\text{precision}*\text{recall}}{\text{precision}+\text{recall}}.\end{aligned} \tag{9}$$

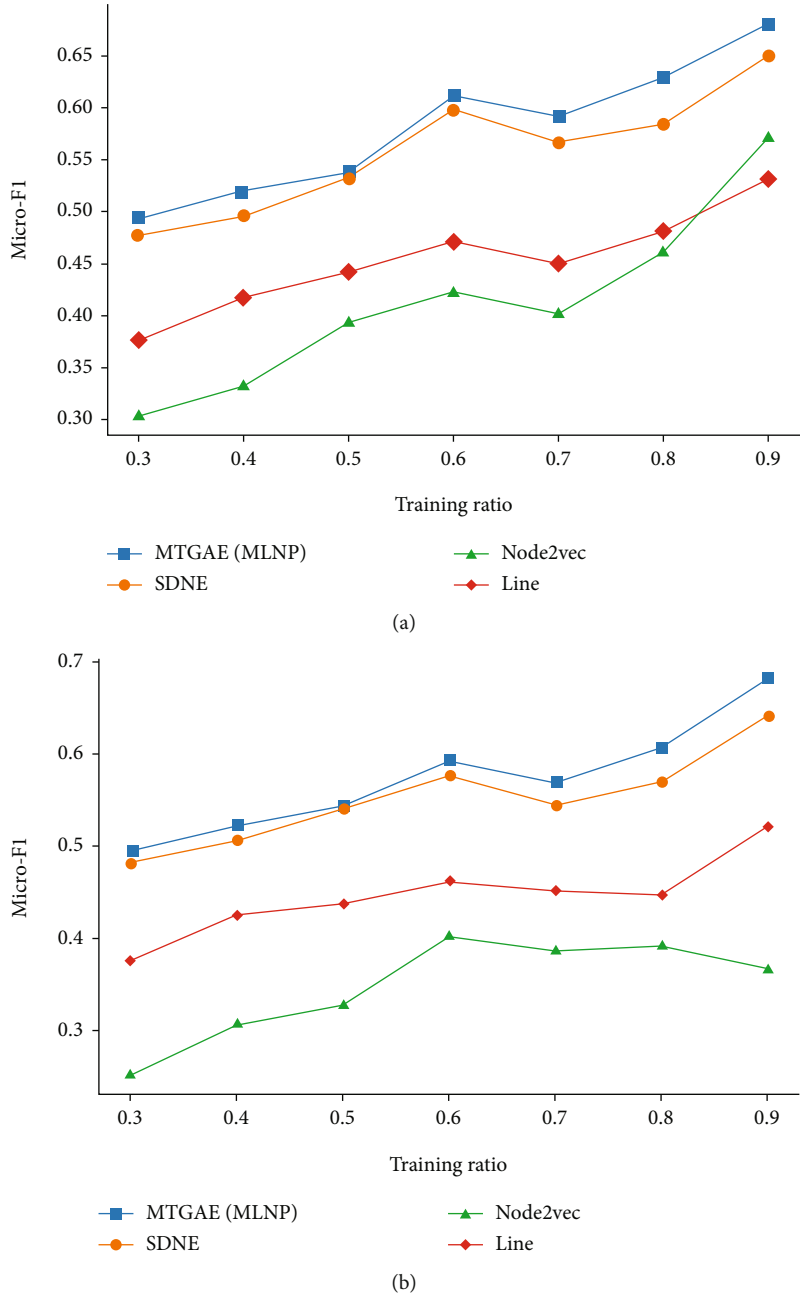

Figure 7: (a) Micro-F1 and (b) Macro-F1 of MTGAE on Brazil-flight dataset.

6.2. Loss Convergence Comparison of Optimizers. To check the loss convergence of the Adam optimizer and SGD optimizer, we apply the control variable method to perform 100 iterative training epochs on the premise of consistent model parameters. The simulation experiments of loss convergence are shown in Figure 5.

In this experiment, the results obviously reveal that the architecture combined with the Adam optimizer converges more quickly and more stably. Under the same circumstances, there is no doubt that the capability of the Adam optimizer is better compared with the SGD optimizer.

6.3. Result Analysis. We set the training batch size of our model to the total number of nodes in one network. To ensure the consistency of other model parameters, our work configures the training parameters of the MTGAE model shown in Table 2. Specifically, the weight parameters of first-order and second-order proximity should remain strictly constant. Affected by the negative effect of overfitting, the autoencoder applies the regularization to limit the weight threshold value in the fully connected neural network.

The feature learning of network structure is insufficient when we train on fewer nodes. Considering the contingent consequences that may appear, we determine to give up sampling 10% and 20% of the observed links in networks for training. Instead, when the training percentage increases from 30% to 90%, every time, we calculate the mean value of 10 experiments to compare the performances between

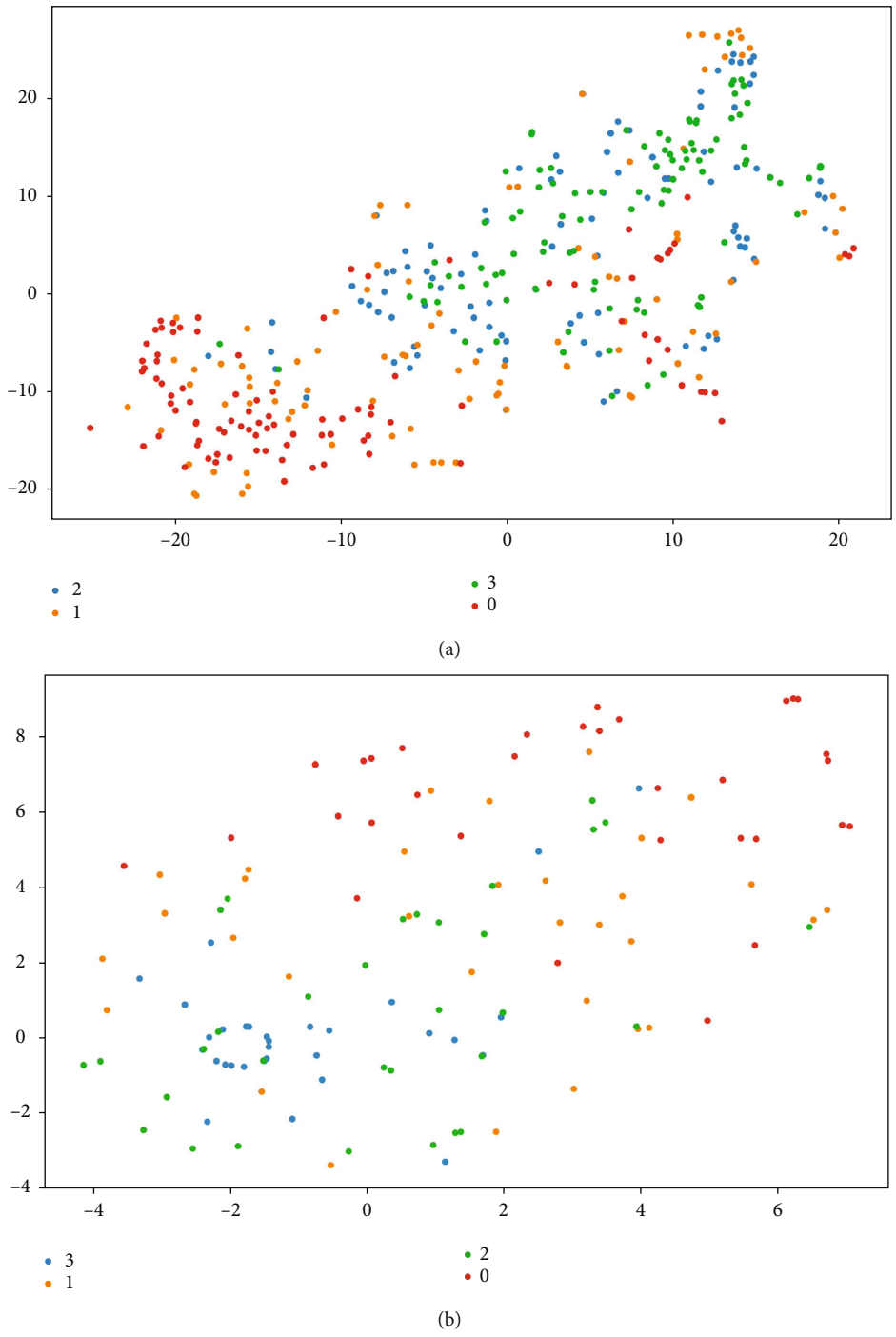

Figure 8: Visualization of (a) Europe-flight network and (b) Brazil-flight network.

the MTGAE (MLNP) algorithm and the classical SDNE algorithm. Moreover, the mainstream algorithm of Line and Node2vec [37] is chosen as benchmarks as well, and the actual consequences of node classification are shown in Figures 6 and 7.

By carefully calculating the experiment results, we discover that under different proportions of training sets, the proposed MTGAE (MLNP) model applied in Europe-flight and Brazil-flight networks boosts the average Micro-F1 by 2.42% and 2.25%, respectively, and enhances the average Macro-F1 by 2.54% and 2.21%, respectively. When the training percentage is up to 90%, it means that the algorithm completely learns the network representations, and the promotion of node classification accuracy reaches the climax, even 5%-6%. It can be seen that whatever proportion of the training set is divided by the experiment, both the Micro-F1 and Macro-F1 of our algorithm are generally higher than those of the related algorithms. We find that our algorithm

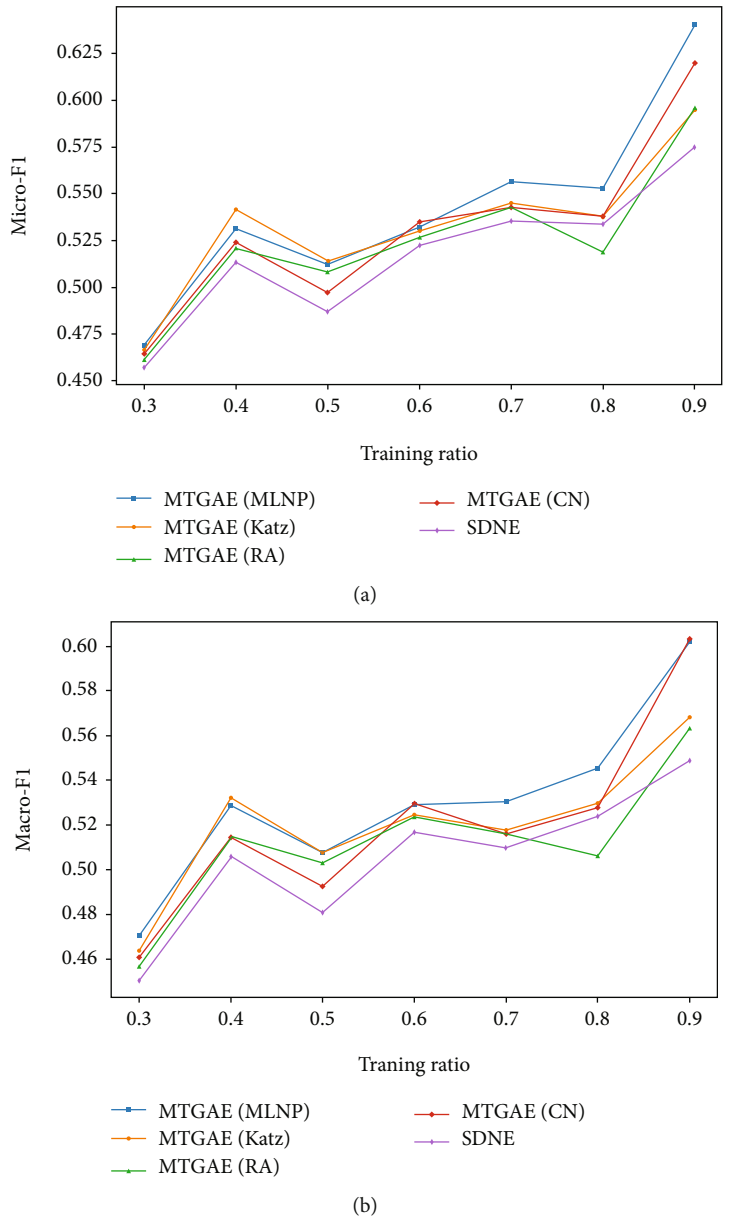

Figure 9: (a) Micro-F1 and (b) Macro-F1 of loss function framework on Europe-flight dataset.

can promote both the evaluation metrics, indicating that the introduced high-order proximity can capture the structure features better in latent spaces and achieve ideal classification effects. The visualizations of the two datasets are shown in Figure 8.

6.4. Horizontal Contrast of Loss Function Framework. In order to verify the universal validity of the high-order loss function framework, we separately adopt the same processing method as the MLNP index for the CN index, RA index, and Katz index. In two different networks, the horizontal contrasts of our experiments are shown in Figures 9 and 10.

The results reveal that no matter what kind of similarity index we introduce into the framework of the high-order loss function, the MTGAE model is superior to the SDNE model except for a couple of special cases on two datasets. Only when we randomly sample 80% of the links in the Europe-flight network and stochastically sample 50% of the links in the Brazil-flight network for training, the SDN model can behave better slightly than one or two other models. The underlying cause is the particularity of datasets. Moreover, it can be found that when we convert the MLNP index and Katz index to high-order loss functions, the improvement margin of node classification is more apparent. The accurate results are shown in Table 3. We choose the MTGAE model with the best prediction accuracy to display the specific improvement margin compared with the SDNE model. Hence, the experiment consequences demonstrate that the introduced framework of high-order loss function is generally effective in boosting the accuracy of node classification.

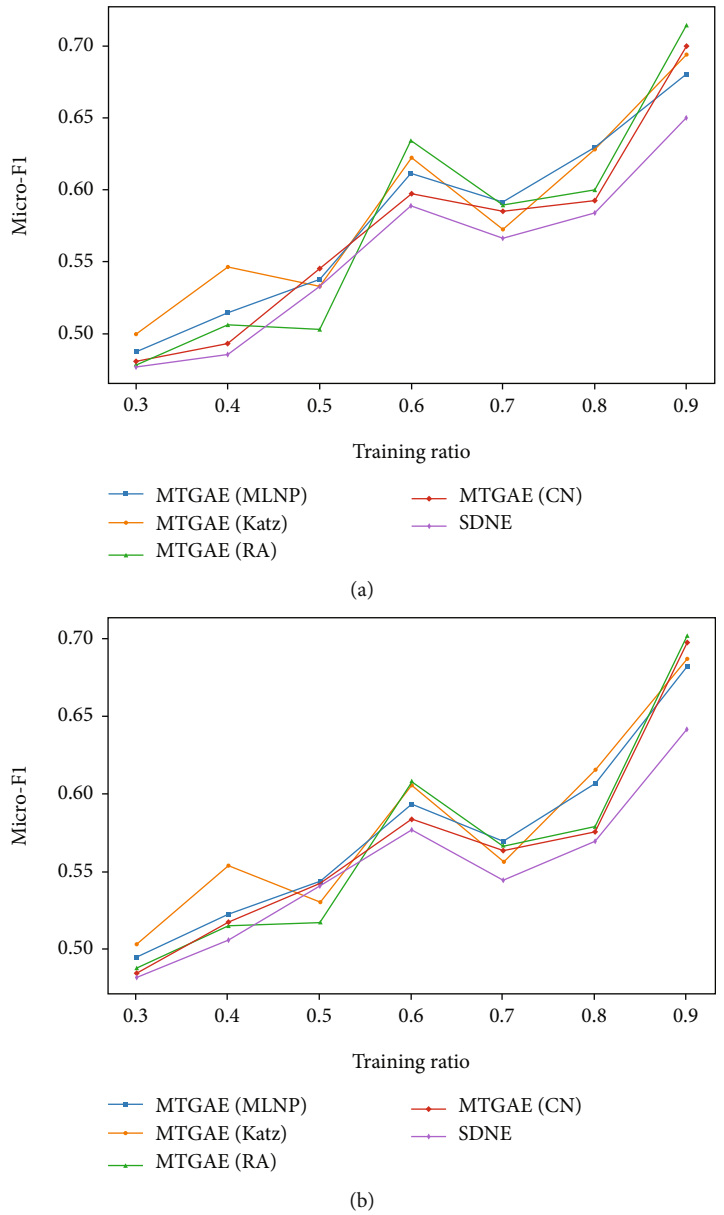

Figure 10: (a) Micro-F1 and (b) Macro-F1 of loss function framework on Europe-flight dataset.

Table 3: Performance of MTGAE model compared with SDNE.

Training ratio	Europe-flight				Brazil-flight			
	SDNE		MTGAE		SDNE		MTGAE	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
0.3	0.457	0.451	0.469	0.471	0.477	0.482	0.499	0.503
0.4	0.513	0.506	0.541	0.532	0.496	0.506	0.547	0.554
0.5	0.487	0.481	0.514	0.508	0.533	0.541	0.545	0.544
0.6	0.522	0.516	0.535	0.523	0.589	0.577	0.633	0.608
0.7	0.535	0.510	0.556	0.531	0.567	0.544	0.592	0.569
0.8	0.534	0.524	0.553	0.546	0.584	0.568	0.629	0.615
0.9	0.575	0.549	0.64	0.603	0.650	0.641	0.714	0.702

7. Conclusions

In this paper, we put forward an MLNP similarity algorithm that integrates multiple similarity theories. In addition, we establish an architecture of the MTGAE model which introduces the high-order loss function into an optimized autoencoder by preprocessing the similarity index. The extraordinary innovation of the MTGAE model is that it successfully applies the link prediction methods to the field of node classification. Specifically, the MLNP index of link prediction is used as an intermediate product to construct the high-order loss function. The above algorithms perform favorably well in both applications of link prediction and node classification. Furthermore, our work applies different similarity matrices as the high-order loss functions to verify the universal validity of the framework. The results demonstrate that our framework of high-order loss function adapts to the majority of popular similarity indexes.

With the continuous development and innovation of deep learning, numerous deep models with side information of nodes and edges emerge in an endless stream. However, some static models can no longer satisfy the needs of a broad range of practical applications. Experts and scholars have gradually turned their attention to dynamic graph embedding models. Although some professors have put forward algorithms to address the dynamic network, quite efficient methods to handle the multidimensional features still lack. The dynamic network is increasingly becoming a significant research object. Embedding the features of nodes and edges into autoencoder architecture and building dynamic evolution models are becoming significant research directions to extend graph embedding technologies. In the future, the majority of models to address the network representation learning problems have broad application prospects in such as recommender systems [38] and mobile computing [39].

Data Availability

The data used to support the findings of this study are publicly available.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61771072), Special Project of People's Public Security University of China (Grant No. 2020JWCX01), and Open Project of the Key Laboratory of the Police Internet of Things Application Technology (Ministry of Public Security of China).

References

[1] Z. Sun, L. Yin, C. Li, W. Zhang, A. Li, and Z. Tian, "The QoS and privacy trade-off of adversarial deep learning: an evolutionary game approach," *Computers & Security*, vol. 96, p. 101876, 2020.

[2] L. Yin, B. Fang, Y. Guo, Z. Sun, and Z. Tian, "Hierarchically defining Internet of Things security: from CIA to CACA," *International Journal of Distributed Sensor Networks*, vol. 16, no. 1, Article ID 1550147719899374, 2020.

[3] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics*, vol. 2008, no. 10, pp. 155–168, 2008.

[4] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2071–2080, New York, New York, USA, 2016.

[5] L. Y. Lv, C. H. JIN, and T. Zhou, "Similarity index based on local paths for link prediction of complex network," *Physical Review E*, vol. 80, no. 4, pp. 211–223, 2009.

[6] W. Li, D. Yin, D. Yuan, B. Wang, and Y. Gu, "Particle propagation model for dynamic node classification," *IEEE Access*, vol. 8, pp. 140205–140215, 2020.

[7] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: a survey," *IEEE transactions on Big Data*, vol. 6, pp. 3–28, 2018.

[8] L. Cai, Y. Xu, T. He, T. Meng, and H. Liu, "A new algorithm of DeepWalk based on probability," *Journal of Physics: Conference Series*, vol. 1069, no. 1, pp. 130–135, 2019.

[9] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.

[10] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[11] P. Jaccard, "Etude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin of the Torrey Botanical Club*, vol. 37, p. 547, 1901.

[12] T. Zhou, L. Lv, and Y. C. Zhang, "Predicting missing links via local information," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 71, no. 4, pp. 623–630, 2009.

[13] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[15] B. Perozzi, R. Alrfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 701–710, Newyork, USA, 2014.

[16] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "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, Halifax, NS, Canada, 2017.

[17] 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*, pp. 1067–1077, Florence, Italy, 2015.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[20] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layerwise training of deep networks," in *International conference on neural information processing systems*, pp. 153–160, Stony Brook University, 2006.

[21] M. Ranzato, Y. L. Boureau, and Y. Lecun, "Sparse feature learning for dep belief networks," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1185–1192, 2007.

[22] N. K. Tomas and W. Max, *Variational Graph Auto-Encoders*, vol. 28, no. 3, 2016Springer, 2016.

[23] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd international conference on knowledge discovery and data mining*, pp. 1225–1234, San Francisco, CA, USA, 2016.

[24] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[25] Q. Ou, Y. D. Jin, T. Zhou, B. H. Wang, and B. Q. Yin, "Power-law strength-degree correlation from resource-allocation dynamics on weighted networks," *Physical Review E*, vol. 75, no. 2, article 021102, 2007.

[26] N. Ketkar, "Introduction to keras," in *Deep learning with Python*, pp. 97–111, Apress, Berkeley, CA, 2017.

[27] M. Abadi, A. Agarwal, P. Barham et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015, https://github.com/tensorflow/tensorflow.

[28] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[29] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, Toronto, Canada, 2010.

[31] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, 2006.

[32] R. Ackland, *Mapping the US Political Blogosphere: Are Conservative Bloggers More Prominent*, Presentation to Blog Talk Downunder, Sydney, 2005, http://incsub.org/blogtalk/images/robertackland.pdf.

[33] C. Von Mering, R. Krause, B. Snel et al., "Comparative assessment of large-scale data sets of protein-protein interactions," *Nature*, vol. 417, no. 6887, pp. 399–403, 2002.

[34] V. Batageli and A. Mrvar, *Pajek Datasets*http://vlado.fmf.unilj.si/pub/networks/data/default.htm.

[35] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[36] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[37] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, San Francisco, CA, USA, 2016.

[38] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[39] L. Yin, "Threat-based declassification and endorsement for mobile computing," *Chinese Journal of Electronics*, vol. 28, no. 5, pp. 1041–1052, 2019.