Improving Link Prediction Accuracy of Network Embedding Algorithms via Rich Node Attribute Information

Weiwei Gu, Jinqiang Hou, and Weiyi Gu*

Abstract: Complex networks are widely used to represent an abundance of real-world relations ranging from social networks to brain networks. Inferring missing links or predicting future ones based on the currently observed network is known as the link prediction task. Recent network embedding based link prediction algorithms have demonstrated ground-breaking performance on link prediction accuracy. Those algorithms usually apply node attributes as the initial feature input to accelerate the convergence speed during the training process. However, they do not take full advantage of node feature information. In this paper, besides applying feature attributes as the initial input, we make better utilization of node attribute information by building attributable networks and plugging attributable networks into some typical link prediction algorithms and name this algorithm Attributive Graph Enhanced Embedding (AGEE). AGEE is able to automatically learn the weighting trades-off between the structure and the attributive networks. Numerical experiments show that AGEE can improve the link prediction accuracy by around 3% compared with SEAL, Variational Graph AutoEncoder (VGAE), and node2vec.

Key words: attributive network; link prediction; network embedding

1 Introduction

Networks (a.k.a. graphs) consist of entities (nodes) and their connections (links) which are a fundamental representation of many real-world relations[1]. For example, networks can be used to describe the protein-protein interaction in biology[2], the syndication investment events between venture capital institutions in economics[3−5], and the structural or functional interaction between different brain regions[6]. For WWW, social networks, and citation networks, link prediction can also help in recommending relevant pages, finding new friends, or discovering new citations[7−9].

These linkages between entities contain rich information on node properties, network structures, and network evolution. Predicting the existence of a relation, which is always abbreviated as link prediction, is a crucial task in network science not only in theory but also in practice. For networks in biology like protein-protein interaction networks, metabolic networks, and food webs, the discovery and validation of links require significant experimental effort. Instead of blindly checking all possible links, link prediction can help scientists to focus on the most likely links, which can sharply reduce the experimental cost.

The conventional link prediction methods can be divided into several groups. The approaches that make link prediction according to local similarity are based on the assumption that two nodes are more likely to be connected if they have many common neighbors[10, 11]. These approaches are fast and highly parallel since they only consider local structure. However, the biggest drawback is their low prediction accuracy, especially when the network is sparse and large. While, global

• Weiwei Gu and Jinqiang Hou are with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China. E-mail: weiweigu@mail.buct.edu.cn; 2022200847@buct.edu.cn.
• Weiyi Gu is with the School of Computing, China University of Geosciences, Wuhan 430074, China. E-mail: weiyigu2002@sina.com.
∗ To whom correspondence should be addressed.

similarity based methods use the whole network topological information to calculate the similarity between links[11−14]. Although those methods perform better on prediction accuracy, they usually suffer from high computational complexity problem which makes them unfeasible for graphs that contain million and billion of nodes. There are also some probabilistic and statistical based approaches, assuming that there is a known prior structure of the network, like a hierarchical or circle structure. However, they can not get over the problem of low accuracy. Furthermore, the conventional approaches can hardly reveal hidden information about node properties and network structures behind the linkages.

Recently, there has been a surge of algorithms that seek to make link predictions through network embedding which extracts both local and global structural information about nodes from graphs automatically. The idea behind these network embedding algorithms is to learn a mapping function that embeds nodes as points in a low-dimensional space $\mathbb{R}^d$ which encodes information from the original graph. Network embedding based methods, which are usually based on the Skip-Gram method or matrix factorization, such as DeepWalk, node2vec, LINE, and struc2vec[15−19], have achieved a much higher link prediction accuracy compared with the conventional ones. Random walk based network representation learning algorithms are task agnostic, and the learned representations are used to perform graph-based downstream machine learning tasks, such as node classification[20], node centrality measuring[21], as well as link prediction[16, 22, 23]. To start with, there is no supervised information during the training process, the representation of nodes is updated directly without considering the global structure of networks. Besides, computational complexity is the biggest bottleneck since Skip-Gram based algorithms require a large number of random walks[15, 16, 19], and they have limited expressive power since the embedding process is fixed by random walk strategies. Besides, the representations can be hardly extended for inductive learning since the embedding vectors can not be transferred from graph to graph[24].

In recent years, deep learning techniques based on graph neural networks have achieved triumphs in image processing[25] and natural language processing[26].

This stimulates the extensions of the methods on graph structures to perform link prediction tasks by converting the network structures into low-dimensional representations. Those algorithms borrow the concept of convolution from the convolutional neural network and convolve the graph directly according to the connectivity structure of the graph via the Graph Neural Networks (GNNs) architecture. Representative algorithms in this genre include Graph AutoEncoder[22], GraphSage[24], and SEAL[27]. Compared with traditional link prediction methods, GNN-based link prediction algorithms take advantage of the nodal attributes via initialized node representations with feature matrix to accelerate the convergence speed during the optimization process. Each row of the feature matrix corresponds to a node's attributes in a graph, with one column representing a feature of all nodes.

To the best of our knowledge, all the GNN-based link prediction algorithms only utilize node attributes as the initial feature input and ignore the rich nodal information contained in the feature matrix. For example, in machine learning articles, "deep learning" is a widely used keyword in an abundance of articles, and due to its extensive use in many articles, it has a limited contribution to the linkages prediction of the citation network. However, the co-occurrence of rare keywords such as "percolation" in machine learning articles may reveal some significant characters since this keyword generally only appears in some multidisciplinary papers. Feature frequency is a crucial indicator for predicting article citation, especially for finding the connection between interdisciplinary and innovative papers.

Defining and identifying feature information contained in the feature matrix provide a way to discover latent graph connections that can not be quantified by network topology. Figure 1 shows the feature co-occurrence of the Cora[28] and CiteSeer[29] networks. From Fig. 1 we find the co-occurrences of node features follow a power law distribution with some features widely existing in most of the nodes. For example, the most frequent feature in the Cora graph appeared 1083 times, which means in the Cora graph, nearly 40% of nodes take this feature as a keyword. We also discovered some keywords (attributes in the feature matrix) rarely occur, they exist only in several nodes, and the rarely appeared features contain rich

(a) Frequency of feature occurrence in Cora

(b) Frequency of feature occurrence in CiteSeer
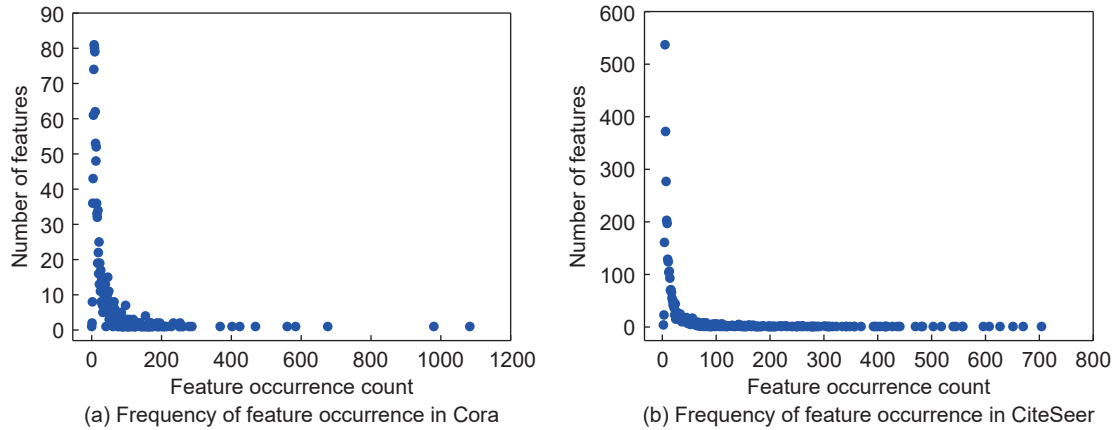
Fig. 1   Attributes distribution of the Cora and CiteSeer feature matrix with a large number of attributes occurring occasionally, while a small number of attributes contribute as features of many nodes.

information about nodal link information. How to quantify feature differences and better utilize feature information to enhance link prediction accuracy is an important link prediction research field.

In this paper, in order to make better utilization of the feature attributes to improve link prediction accuracy, we propose Attributive Graph Enhanced Embedding (AGEE). AGEE is a plug-and-play algorithm, it can be plugged into a bunch of link prediction algorithms without modifying the original link prediction architecture, and it also has the universal property, which means most of the link prediction algorithms can enhance their prediction power with the help of AGEE. AGEE first uses entropy to quantify information of each attribute and then computes the total amount of information between any node pairs. After that, AGEE sets a threshold to the total amount of information between nodes to build a feature graph. In the last step, it separately trains the feature graph and structure graph with different training algorithms to find a trade-off between the feature graph predicted probability and the structure graph to form the final structure link prediction accuracy. We validate AGEE's performance on node2vec, Variational Graph AutoEncoder (VGAE), and SEAL algorithms over several networks, and the results show that AGEE can significantly improve link prediction accuracy by around 3%.

2   Method

AGEE consists of two parts, building feature graphs according to the given feature matrices and plugging the predicted results into a variety of algorithms to improve the link prediction accuracy.

2.1   Building feature graph

Recent graph neural network based algorithms have achieved the state-of-the-art accuracy in link prediction tasks. However, those algorithms only take node feature information as the initial input of their models and ignore the rich hidden information behind them. The feature matrix of a graph is denoted as F, which is an $N \times M$ matrix, with N representing node number and M representing feature number. For example, in Fig. 1, in the Cora dataset each node has 1433 features, three of which appeared in over half of the nodes, in other words, those features are widely used as keywords, and it is difficult to infer the link existence probability between node pairs based on the information provided by those attributes. We also notice that there are a large number of rare features, which co-exist only in several nodes, and those rare features may represent field-specific and vital keywords in sub-field research areas, and the connection probability between those nodes based on those rare keywords is higher compared with the widely appeared features. Feature frequency is an important indicator to quantify features' importance and tie strength between nodes. The co-occurrence of less frequent features indicates a tight relation while the commonly appeared features represent a loose relation.

In order to distinguish attributes and quantify information contained within different features, we proposed the following Eq. (1):

$$I(m) = -\log_2(p(m)) \tag{1}$$

where $p(m)$ is the frequency that feature m appears over the total number of features across all nodes. $I(m)$ quantifies information convery by feature m. Note that

the base of the logarithm only affects the scaling, and here we take base 2 in units of bits to quantify feature information.

Feature information is also called the "surprisal" of a feature, with low occurrence probabilities corresponding to higher information, and frequently appearing features carry a small amount of information. We concatenate feature information of feature into a feature information vector $I = \{I(1), I(2), ..., I(M)\}$, $I \in \mathbb{R}^M$. As shown in Fig. 2, the left vertical axis represents the feature information bits calculated with Eq. (1), and the right vertical axis represents the frequent probability over all node features. As shown in Fig. 2, Eq. (1) gives high information to less frequently occurred features and low information to the frequently appeared features, especially for nodes that appeared greater than 1000.

Equation (1) quantifies information contained in each feature. We propose Eq. (2) to measure the feature relations between nodes.

$$W = (I \times F) \cdot (I \times F)^{\mathrm{T}} \tag{2}$$

where I is a $(1 \times M)$-dimensional vector with its component representing the feature information, F denotes the original feature matrix, and $(I \times F) \in \mathbb{R}^{N \times M}$ represents weighted feature matrix, with each element not only indicating the existence of the feature but also quantifying the feature's influence on node. In Eq. (2), we use $W \in \mathbb{R}^{N \times N}$ to represent feature relations between the node pairs with each element. $W_{i,j}$ stands for the feature similarity strength between nodes i and j.

Matrix W is densely connected with each element representing feature similarity which is computed based on the cumulative information that features contain. The higher the feature similarity, the easier it is for nodes to connect together[30]. Based on the weighted similarity matrix W, we build the feature network $G_{\text{feature}} = (V, E)$. G is an undirected and unweighted graph, where V denotes the set of vertices $|V| = N$ which is consistent with nodes in the structure graph $G_{\text{structure}}$, and $E \in V \times V$ is the set of feature links. The adjacency matrix of the feature graph is denoted as matrix A, where $A_{i,j} = 1$ if $(i, j) \in E$, otherwise $A_{i,j} = 0$, if there is no feature link between i and j.

In this paper, the feature edges are generated with the following process. We first set a benchmark value ϵ to binarize the feature weight matrix W, and we set that $A_{i,j}$ equals to 1 if $W_{i,j}$ is greater than ϵ, otherwise $A_{i,j}$ equals to 0, as shown in Eq. (3). Most networks have scale-free properties, and small ϵ values lead to dense networks while large ϵ values lead to sparse networks. In order to mimic the real density of the graph, we set ϵ to the standard value which features network density $A_{i,j}$ equal to the precomputed structure network density.

$$A_{i,j} = \begin{cases} 1, & W_{i,j} > \epsilon; \\ 0, & W_{i,j} \leqslant \epsilon \end{cases} \tag{3}$$

2.2 Plugging AGEE into link prediction algorithms

AGEE is a plugin that can be embedded into other link prediction algorithms to increase their link prediction accuracy. In this paper, we select three typical and widely used link prediction algorithms with network embedding fashions, which are node2vec[16], VGAE[22], and SEAL[27]. Node2vec mainly applies the skipGram[31] which is an unsupervised learning
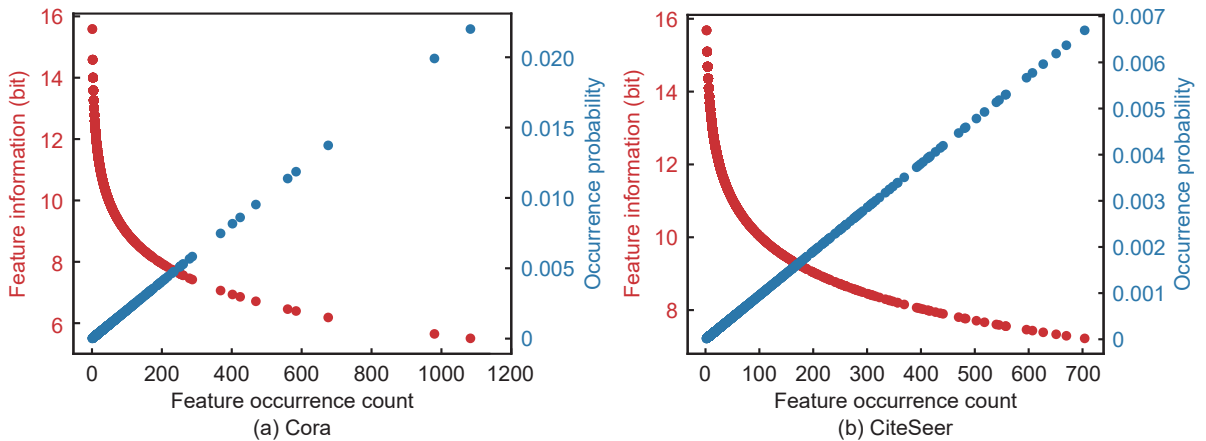
Fig. 2 Comparison of feature information quantified by Eq. (1) and feature occurrence probability of Cora and CiteSeer datasets.

technique. VGAE is mainly based on variational auto encoders[32] which is a probabilistic generative model that requires neural networks as a part of the overall structure. SEAL uses node classificational networks[20] which applies an efficient variant of convolutional neural networks that operate directly on graphs. In this section, we take the node2vec algorithm as an example and show how the AGEE helps improve its link prediction accuracy. A brief introduction to VGAE and SEAL is shown in the experiments section.

2.2.1 Learning node representations

Node2vec is one of the most popular network embedding and link prediction algorithms and is widely used to solve a variety of theoretical and practical problems in biology[33], neurosciences[34], and social science[23]. Node2vec builds on the word2vec algorithm[35] by comparing nodes in the network to "words" and a sequence of nodes explored during a biased random walk to "sentence". After the analogy process, the generated "node sentences" are fed into the Skip-Gram model to get the feature representation of nodes.

Next, we will introduce node2vec and illustrate how it learns the mapping function $f : h \leftarrow \mathbb{R}^d$ and representation of the feature graph G_{feature}. Here d is a parameter specifying the number of dimensions, and function f is a matrix that contains $|h| \times d$ parameters. For each node i in the feature graph, $N_s(i) \in h$ are a subset of network neighbors of node u. Node2vec applies the Skip-Gram architecture to optimize the following objective function which maximizes the log-probability of the observing nodes given its neighborhood $N_S(i)$ and mapping function f:

$$\max_f \sum_{i \in V} \log \Pr(N_S(i) \mid f(i)) \qquad (4)$$

To make the optimization problem manageable, we adopt the following two common assumptions: We first assume that the likelihood of observing a neighborhood node is independent of observing any other neighborhoods given the representation of the source node i.

$$\Pr(N_S(i) \mid f(i)) = \prod_{n_j \in N_S(i)} \Pr(n_j \mid f(i)) \qquad (5)$$

Influence symmetric is another assumption that assumes each node pair has a symmetric effect in the representation space. The conditional likelihood of source-neighborhood node pair can be parametrized by a dot product of their representations:

$$\Pr(n_j \mid f(i)) = \frac{\exp(f(n_j) \cdot f(i))}{\sum_{v \in V} \exp(f(v) \cdot f(i))} \qquad (6)$$

With the above assumptions, the objective function can be written as

$$\max_f \sum_{i \in V} \left[-\log Z_i + \sum_{n_j \in N_S(i)} f(n_j) \cdot f(i) \right] \qquad (7)$$

where the per-node partition function, $Z_i = \sum_{j \in V} \exp(f(i) \cdot f(j))$, is expensive to compute for large networks, and we approximate it using negative sampling[35], node2vec optimizes Formula (7) with stochastic gradient ascent over the model parameters defined on representation learning function f. In this paper, we learn the node2vec representation of the feature graph and structure graph with the implementation from node2vec.

2.2.2 Link existence probability

h^i_{feature} represents the learned node embedding of node i of feature graph G_{feature}, and for every pair of nodes i and j in G_{feature} or $G_{\text{structure}}$, we use the Hadamard product to represent the potential relations of node pairs, $e^{i,j}_{\text{feature}} = h^i_{\text{feature}} \otimes h^j_{\text{feature}}$. Note that $e^{i,j}_{\text{feature}}$ is the relation representation between nodes i and j which is also a d-dimensional vector. The above procedure is also applied to the structure graph. Link prediction between nodes i and j in the feature graph can be represented as

$$p_{i,j}(e^{i,j}_{\text{feature}}; \theta) = \frac{1}{1 + \exp(e^{i,j}_{\text{feature}} \theta)} \qquad (8)$$

where θ is a d-dimensional parameter vector, and $e^{i,j}_{\text{feature}} \theta$ is the dot product between the vectors $e^{i,j}_{\text{feature}}$ and θ. The best estimate of the entries of vector θ is obtained from the training set via logistic regression.

In this paper, we first hide γ randomly chosen edges of the structure graph to form the "positive" sample set. We then sample an equal amount of disconnected vertex pairs as the "negative" sample set. The union of the "positive" and "negative" samples form the test set and are denoted as $S_{\text{structure}}$ of the structure graph. We select 10% of the remaining edges and an equal amount of disconnected node pairs to form the validation set. The rest edges and an equal amount of disconnected node pairs form the training set. We focused our task

on predicting the existence or absence of the relations in $S_{\text{structure}}$.

2.2.3 Plugging feature predicted probability into structure graph

We train the feature graph and structure graph and plug feature predicted probability into the structure graph with a hyper-parameter α which balances the importance between the two graphs. Equation (9) is the mathematical aggregation function. In order to compute the edge existence probability between i and j of a structure graph, for example, nodes 0 and 6 in Fig. 3, α acts as the consensus coefficient between the predicted probability of the original structure graph $p_{\text{structure}}^{i,j}$ and feature graph $p_{\text{feature}}^{i,j}$. Our intuition behind the consensus operation lies in that the link between nodes 0 and 6 is difficult to predict in the original structure graph due to the absence of common neighbors. However, since in the feature graph nodes 0 and 6 have a common neighbor 5, the feature graph will give a high link probability between 0 and 6. Our experimental results show that the optimal value of α is around 0.6, which represents that node pairs in the test set of the structure graph consider 0.6 of their own predicted probability from the original structure graph and 0.4 from the feature graph.

The overall illustration of the aggregation process is shown in Fig. 3. AGEE algorithm consists of two stages. In the first stage, we build a feature graph according to rich nodal attributes and feature entropy. In the second stage, we separately train the feature graph and the original structure graph since they follow different dynamics, and we introduce a hyper-parameter α that trades off the feature and structure prediction probabilities.

$$p_{i,j} = \alpha p_{\text{structure}}^{i,j} + (1 - \alpha) p_{\text{feature}}^{i,j} \qquad (9)$$

3 Experiment

3.1 Dataset, baseline, and evaluation metric

We use three citation datasets with rich node attribute information. The Cora dataset consists of 2708 scientific publications classified into seven classes and 5429 links with each publication in the dataset being described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words as nodal attributes. The CiteSeer dataset consists of 3312 scientific publications classified into six classes. CiteSeer has 4732 links and 3703 nodal attributes by
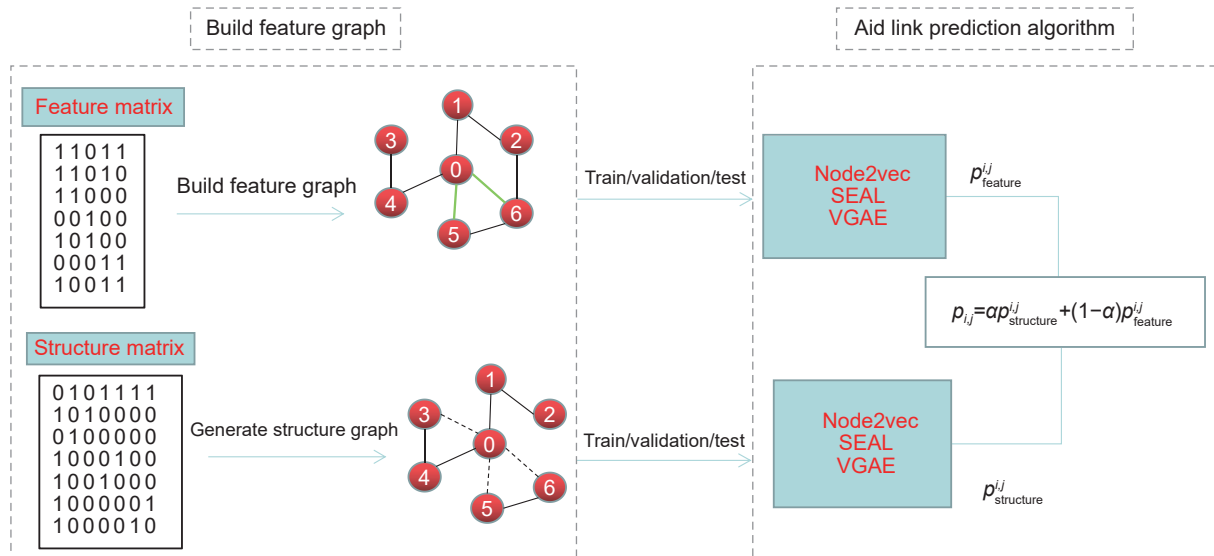
Fig. 3 Overall architecture of AGEE algorithm. We use a simple seven-node graph with five nodal features to illustrate AGEE's architecture. In the first stage, based on the feature matrix we build the feature graph according to Eqs. (1)−(3), and we also build the structure graph according to the adjacency matrix. In the structure graph, we use the dashed lines to indicate some of the representative edges that need to be predicted by link prediction algorithms and apply solid lines to represent edges used to train the link prediction algorithms. In the structure graph, there are no edges between nodes 0 and 6, 5 and 0, and in the feature graph, there are links between them. The feature graph works as a supplementary for the structure graph. In the second stage, we train the feature graph and structure graph separately and use a hyper-parameter α to find the trade-off between the structure link prediction probability and the feature link prediction probability.

0/1 values that respond to the absence/presence of the corresponding word from the dictionary. The PubMed is a larger dataset compared with Cora and CiteSeer and contains 19 717 scientific publications from the PubMed database pertaining to diabetes classified into three classes. It has 44 338 links with each publication in the dataset described by a Term Frequency-Inverse Document Frequency (TF/IDF) weighted word vector from a dictionary which consists of 500 unique words.

VGAE is a framework for unsupervised learning on graph-structured data based on the Variational AutoEncoder (VAE) architecture. VGAE improves link prediction accuracy on a number of network datasets such as Cora and CiteSeer, and it has inspired a wide range of ongoing research. VGAE is also the most promising model for network constriction tasks. The main idea of VGAE is that it encodes the input into a distribution rather than a high-dimensional vector space. Then a random sample is taken from the distribution rather than generated from the encoder directly. The loss function of VGAE consists of two parts. The first part is the variational lower bound, which measures how well the output network reconstructs the original graph. If the reconstructed network is dissimilar from the original data, then the reconstruction loss will be high. The second part works as a regularization. It is the Kullback–Leibler (KL) divergence of the approximate from the true posterior, which measures how closely the output distribution matches the latent network distribution. In VGAE, we use the default parameter setting and the PyTorch implementation from VGAE_PyTorch.

SEAL is a novel link prediction framework and has achieved state-of-the-art link prediction accuracy in a large number of small-scale networks. Instead of applying the entire network information to do the link prediction task, SEAL first extracts local enclosing subgraphs within a 2-hops neighborhood. In doing so, SEAL enables graph feature learning ability, and through node labeling operation, SEAL can better capture the hidden relationship of the nodes in subgraphs. After the subgraph extracting process, SEAL applies a GNN to replace the fully-connected neural network. During the graph convolutional process in the GNN framework, SEAL permits learning from not only subgraph structures but also latent and explicit node features, thus incorporating multiple types of information. SEAL outperforms the previous state-of-the-art method on link prediction tasks. It is the pioneer to apply node labeling and subgraph extracting operations to the link prediction task. SEAL treats link prediction as a subgraph binary classification task. It outperforms previous latent embedding-based link prediction algorithms such as VGAE and node2vec on small datasets. As for larger graphs that contain millions or billions of nodes, SEAL has a memory error problem even when the network is not big. We find SEAL fails to do the link prediction task even for networks that contain less than twenty thousand nodes such as the PubMed network. In SEAL we apply the PyTorch implementation from SEAL_PyTorch.

In this paper, we use Area Under the ROC Curve (AUC) as an evaluation metric to measure the performance of the link prediction application. AUC is computed for every test node, and the average values are reported. A higher AUC indicates better predictive performance. In this paper, for every link prediction accuracy, we repeat the link prediction procedure 10 times and get the mean average results over these repetitions.

3.2 Link prediction accuracy comparison

AGEE algorithm encodes structure network information as well as feature network information by optimizing their network topology separately and learning the best trade-off between the predicted probability of structure prediction and feature prediction. We plug AGEE into node2vec, VGAE, and SEAL algorithms. Although the main ideas and architectures behind those algorithms are different, they all follow the similar training process described below. In this paper, to do the link prediction task, we apply the repeated random sub-sampling validation by applying the following procedure 10 times. We first hide 10% of randomly chosen edges of the original structure graph. The hidden edges in the original graph are regarded as the "positive" sample set. We sample an equal amount of disconnected vertex pairs as the "negative" sample set. The union of the "positive" and "negative" sample sets form our test set. The validation set also contains 10% of randomly chosen edges of the original graph excluding the test set. The training set consists of the remaining 80% of connected node pairs and an equal number of randomly chosen disconnected node pairs. We use the training set to learn the probability of connection between pairs of nodes, use the validation

set to identify the best training epochs and trade-off weights, and use the test set to evaluate the performance of different link prediction algorithms. Besides the original node2vec, VGAE, and SEAL algorithms, we add comparisons with the following well-known link prediction algorithms.

LINE[18] minimizes a loss function to learn embedding while preserving the first- and the second-order neighbors' proximity among vertices in the graph. GANR[21] applies the node centrality network architecture to do the link prediction task and reveal the hidden structures of networks. GraphSAGE[24] learns node embedding through a general inductive framework consisting of several feature aggregators. It usually adopts a supervised node classification task as the evaluation benchmark with the assumption that a better embedding algorithm leads to higher node classification accuracy. ARVGA[36] uses a variational graph autoencoder to learn to embed and perform the link prediction as the supervised task.

From Table 1 we find that general algorithms with deep graph neural networks that have autoencoder and autodecoder frameworks such as GANR, GraphSAGE-mean, and ARVGA have higher AUC values compared with shallow neural network models such as LINE. When plugging our glue AGEE algorithm into node2vec, SEAL, and VGAE, the link prediction accuracy of all the previous algorithms is improved by around 3%. We also notice that even though node2vec has a shallow neural network, its performance is only slightly behind the deep graph neural networks based algorithms, and the combination of AGEE and node2vec algorithm outperforms all link prediction algorithms and achieves the state-of-the-art link prediction accuracy of all datasets.

3.3 Parameter sensitivity study

3.3.1 Link prediction accuracy over different training sets and robustness test

We quantify AGEE's link prediction ability by tuning the size of the training set. As shown in Fig. 4, we find

that AGEE_node2vec outperforms the original node2vec algorithm especially when the training set is small. When we use only 10% of the edges in the original network to perform the link prediction task, the node2vec's link prediction accuracy is 55%, only 5% higher than random guess 50%, however, for AGEE_node2vec the accuracy is around 68%. This result may due to that when the network structure is sparse, nodal feature network works as a complementary to provide rich link information about nodes. This experiment shows AGEE's ability to accurately predict node relations when the training set is small, compared with original link prediction algorithms.

Being able to predict missing links with a small fraction training set serves at least in the following two aspects. It first reveals the algorithm's ability to capture the hidden structure and the link principle of the network. Good link prediction algorithms can reveal the link principles among nodes even when the training set is small, and they have excellent inductive learning ability. Second, for some biological networks, such as brain connection networks, and protein-protein interaction networks, identifying links between nodes is expensive and sometimes involves bias, accurately predicting missing links or future ones without experiments saves cost.

3.3.2 Link prediction accuracy across different consensus values

The consensus value α in Eq. (9) is one of the most important hyper-parameter in AGEE plugged algorithms. It determines the prediction probability trade-off between feature prediction and structure prediction. α equals 0 representing the condition in which AGEE plugged algorithms only take feature prediction accuracy to form the final link prediction accuracy, while α equals 1 representing it equals the original node2vec algorithms. In this part, we take AGEE_node2vec as an example to test α's influence on link prediction outcomes. As shown in Fig. 5, we find α is quite robust over different datasets across

Table 1   Comparison of AGEE enhanced algorithms over link prediction task under AUC metric.

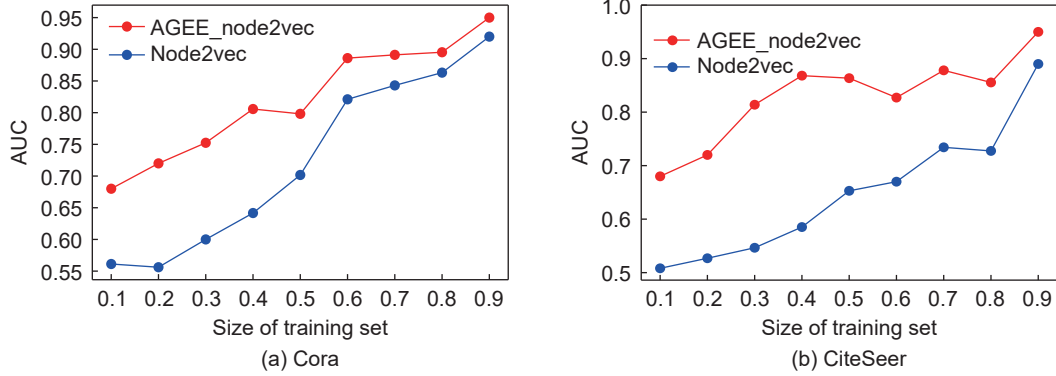| Dataset	AUC									
	GANR	LINE	GraphSAGE-mean	ARVGA	node2vec	VGAE	SEAL	AGEE_VGAE	AGEE_SEAL	AGEE_node2vec
Cora	0.93	0.76	0.89	0.93	0.92	0.91	0.90	0.93	0.93	**0.95**
CiteSeer	0.91	0.73	0.90	0.94	0.89	0.90	0.90	0.92	0.93	**0.96**
PubMed	0.95	0.84	0.91	0.95	0.94	0.93	0.94	0.96	0.95	**0.97**

Fig. 4　AUC score of link prediction accuracy comparison between original node2vec algorithm and AGEE_node2vec on a variety of training sets for Cora and CiteSeer.

different AGEE-combined link prediction algorithms. The optimal α value is around 0.6 which means when predicting structural link relations between nodes, we can take 60% of the predicted probability from structures and 40% of the probability from the nodal feature graphs. From Fig. 5, we also find among all the feature-enhanced link prediction algorithms, AGEE_node2vec outperforms AGEE_SEAL and AGEE_VGAE that are famous link prediction algorithms and are based on graph convolutional network models.

4　Conclusion

Node feature matrix usually works as the original input for graph neural networks to do the downstream classification or prediction tasks. In this paper, we extend the application of the node feature matrix by building a nodal feature graph, and with this nodal feature graph, we propose a plug-and-play AGEE model that improves the link prediction accuracy of existing embedding-based link prediction algorithms without adding extra information and increasing the

Fig. 5　AUC score of link prediction accuracy across different prediction algorithms on Cora and CiteSeer network over hyper-parameter α.

complexity of algorithms. Feature graph enhanced prediction algorithm can improve the link prediction accuracy by around 3%. We introduce a trade-off hyper-parameter α to balance the importance between feature graph predicted probability and structure graph predicted probability. Our results find the consensus value α is quite robust, and α is usually around 0.6 which means when predicting structure links, the structure graph plays a more important role compared with the feature graph.

Although the link prediction accuracy of AGEE_node2vec is quite high compared with previous link prediction algorithms, there is still room for improvement. In AGEE_node2vec the final link prediction probability is determined by a fixed α value, and all edges share the same consensus value. However, link formation follows a popularity and similarity principle and shows that nodes tend to build links with more popular nodes, nodes that have similar features, or both. The node feature matrix is a good indicator to describe the similarity between nodes, and different links are supposed to put different emphasis on feature graphs and structure graphs. Further extensions of AGEE could involve assigning edges of each node pair a personalized α value to improve link prediction accuracy in a more general way.
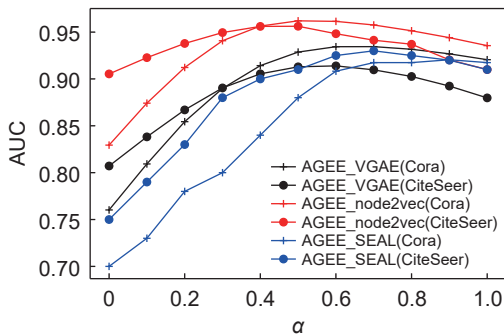
References

[1]　A. Barabási and J. Frangos, *Linked: The New Science of Networks Perseus Publishing*. Cambridge, MA, USA: Perseus Books Group, 2002.

[2]　A. Theocharidis, S. V. Dongen, A. J. Enright, and T. C. Freeman, Network visualization and analysis of gene expression data using BioLayout Express (3D), *Nat. Protoc.*, vol. 4, no. 10, pp. 1535–1550, 2009.

[3]　W. Gu, J. D. Luo, and J. Liu, Exploring small-world

network with an elite-clique: Bringing embeddedness theory into the dynamic evolution of a venture capital network, *Soc. Netw.*, vol. 57, pp. 70–81, 2019.

[4] R. Li, J. Liang, C. Cheng, X. Zhang, L. Zhao, C. Zhao, and H. E. Stanley, The evolution of k-shell in syndication networks reveals financial performance of venture capital institutions, *Soc. Netw.*, vol. 76, pp. 191–202, 2024.

[5] Q. Yao, S. Ma, J. Liang, and R. Li, Syndication network associates with specialisation and performance of venture capital firms, *J. Phys. Complex.*, vol. 4, no. 2, p. 025016, 2023.

[6] M. D. Domenico, Multilayer modeling and analysis of human brain networks, *GigaScience*, vol. 6, no. 5, p. gix004, 2017.

[7] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, Learning to construct knowledge bases from the World Wide Web, *Artif. Intell.*, vol. 118, nos. 1&2, pp. 69–113, 2000.

[8] A. Popescul and L. H. Ungar, Statistical relational learning for link prediction, presented at IJCAI Workshop on Learning Statistical Models from Relational Data, Acapulco, Mexico, 2003.

[9] D. Liben-Nowell and J. Kleinberg, The link-prediction problem for social networks, *J. Am. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[10] A. L. Barabasi and R. Albert, Emergence of scaling in random networks, *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[11] T. Zhou, L. Lü, and Y. -C. Zhang, Predicting missing links via local information, *Eur. Phys. J. B*, vol. 71, no. 4, pp. 623–630, 2009.

[12] H. Liu, Z. Hu, H. Haddadi, and H. Tian, Hidden link prediction based on node centrality and weak ties, *EPL Europhys. Lett.*, vol. 101, no. 1, p. 18004, 2013.

[13] G. Rücker, Network meta-analysis, electrical networks and graph theory, *Res. Synth. Meth.*, vol. 3, no. 4, pp. 312–324, 2012.

[14] F. Shang, B. Chen, P. Expert, L. Lü, A. Yang, H. E. Stanley, R. Lambiotte, T. S. Evans, and R. Li, Local dominance unveils clusters in networks, arXiv preprint arXiv: 2209.15497, 2022.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, DeepWalk: Online learning of social representations, in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 701–710.

[16] A. Grover and J. Leskovec, Node2vec: Scalable feature learning for networks, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

[17] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, Asymmetric transitivity preserving graph embedding, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1105–1114.

[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, LINE: Large-scale information network embedding, in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 1067–1077.

[19] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, *Struc2vec*: Learning node representations from structural identity, in *Proc. 23rd ACM SIGKDD Int. Conf.*

Knowledge Discovery and Data Mining, Halifax, Canada, 2017, pp. 385–394.

[20] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv: 1609.02907, 2016.

[21] W. Gu, F. Gao, X. Lou, and J. Zhang, Discovering latent node Information by graph attention network, *Sci. Rep.*, vol. 11, no. 1, p. 6967, 2021.

[22] T. N. Kipf and M. Welling, Variational graph auto-encoders, arXiv preprint arXiv: 1611.07308, 2016.

[23] W. Gu, L. Gong, X. Lou, and J. Zhang, The hidden flow structure and metric space of network embedding algorithms based on random walks, *Sci. Rep.*, vol. 7, no. 1, p. 13114, 2017.

[24] W. L. Hamilton, R. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1025–1035.

[25] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Las Vegas, NV, USA, 2016, pp. 770–778.

[26] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, A convolutional encoder model for neural machine translation, arXiv preprint arXiv: 1611.02344, 2016.

[27] M. Zhang and Y. Chen, Link prediction based on graph neural networks, arXiv preprint arXiv: 1802.09691, 2018.

[28] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, Automating the construction of Internet portals with machine learning, *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[29] L. Getoor, Link-based classification, in *Advanced Methods for Knowledge Discovery from Complex Data*, S. Bandyopadhyay, U. Maulik, L. B. Holder, and D. J. Cook, eds. London, UK: Springer, 2005, pp. 189–207.

[30] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguñá, and D. Krioukov, Popularity versus similarity in growing networks, *Nature*, vol. 489, no. 7417, pp. 537–540, 2012.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv: 1301.3781, 2013.

[32] D. P. Kingma and M. Welling, An introduction to variational autoencoders, *Found. Trends® Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.

[33] M. Kim, S. H. Baek, and M. Song, Relation extraction for biological pathway construction using node2vec, *BMC Bioinform.*, vol. 19, no. Suppl8, p. 206, 2018.

[34] G. Rosenthal, F. Váša, A. Griffa, P. Hagmann, E. Amico, J. Goñi, G. Avidan, and O. Sporns, Mapping higher-order relations between brain structure and function with embedded vector representations of connectomes, *Nat. Commun.*, vol. 9, no. 1, p. 2178, 2018.

[35] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Proc. 26th Int. Conf. Neural Information Processing Systems - Volume 2*, Lake Tahoe, Nevada, 2013, pp. 3111–3119.

[36] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, Adversarially regularized graph autoencoder for graph embedding, arXiv preprint arXiv: 1802.04407, 2018.
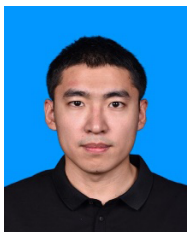
Jinqiang Hou is currently pursuing the master degree at the College of Information Science and Technology, Beijing University of Chemical Technology, China. His research interests include data mining, multilayer networks, and complex networks.

Weiyi Gu is a researcher at China University of Geosciences, China, majoring in computer science and technology. Her research interests mainly include machine learning, computer vision, and high-performance computing.

Weiwei Gu received the PhD degree from Beijing Normal University, China. She is a lecturer at Beijing University of Chemical Technology, China. From September 2018 to January 2020, she conducted joint research and training at Indiana University Bloomington, USA and The University of Chicago, USA. Her research interests include complex network representation learning, multilayer network analysis, and risk investment analysis and modeling. She has published over ten papers in SCI and SSCI journals, including *Nature Communications*, *Social Networks*, *Scientific Reports*, etc.