Nankai University

Undergraduate Thesis

Title：Predicting Long Non-coding RNA-Protein Interactions Based on Deep Learning Model

Student Number： 1811618

Name： Qin Su

Grade： 2018

School： College of Artificial Intelligence

Department： Department of Automation

Major Automation

Instructor Prof. Han Zhang

Completion date： May 2022

Abstract

Long non-coding RNA (lncRNA) usually refers to non-coding RNA with a length of more than 200 nucleotides. A large number of research show that lncRNA plays a key role in complex cellular processes through interactions with proteins, and is closely related to many human diseases.

Computational biologists have proposed many algorithms to predict lncRNA-protein interactions. These algorithms can be roughly divided into machine learning-based and network-based prediction methods. This paper introduces a deep learning method of predicting lncRNA-protein interaction based on graph autoencoder (GAE).

Firstly, this paper uses the GAE with graph convolution neural network to reconstruct the interaction matrix in the end-to-end framework and improves the effectiveness of GAE representation through collaborative training. Then, this paper significantly improves the performance of the model by determining hidden layer dimensions, optimizer, and the corresponding learning rate. The advantages of the proposed deep learning model are validated through comparative experiments with other computing methods. Finally, the model that uses graph attention networks, a variant of graph convolution neural network that can better integrate the correlation of vertex features, is proposed.

Starting with parameter optimization and network framework improvement, this paper improves the prediction accuracy of lncRNA-protein interaction and provides a feasible way for complex biological link prediction tasks.

Key Words： long non-coding RNA; deep learning; lncRNA-protein interaction; graph autoencoders

Chapter 1. Introduction

Section 1. Background and significance of the study

1.1.1 Research Background

Long non-coding RNA (lncRNA) generally refers to non-coding RNA transcripts longer than 200 nucleotides, which have long been considered "junk sequences" in transcripts and ignored. Due to the application of second-generation gene sequencing technology in recent years, the mystery of lncRNA has gradually been revealed, and a large amount of research data has confirmed that lncRNA plays a key role in complex cellular processes and is involved in the regulation of important life processes such as cell differentiation and individual growth and development at multiple levels, and is closely related to many major human diseases [1].

Proteins are biopolymers composed of amino acids via dehydration synthesis, which are the material basis of life and participate in almost all important biochemical reactions related to cellular activity. For example, hemoglobin can bind with oxygen and transport it to various systems in the animal body, immunoglobulin can affect the animal body's resistance to diseases, and glycoproteins can assist cells in achieving recognition of self and non-self [2].

Experimental evidence has long shown that lncRNA can participate in a series of biological processes, especially in interactions with proteins. For example, lncRNA CCAT1 promotes the proliferation and migration of triple-negative breast cancer cells by down-regulating miRNA miR-218 and activating protein ZFX [3]; lncRNA BACE1-AS can regulate mRNA BACE1, which is associated with the production of beta-amyloid protein and can lead to Alzheimer's disease [4].

In general, almost all lncRNAs need to interact with their corresponding RNA-binding proteins to exert their biological functions. At the same time, RNA-binding proteins can also interact with different types of lncRNAs to regulate different cellular processes. However, only a small number of lncRNA-protein interactions have been discovered so far. Therefore, it is crucial to explore the potential lncRNA-protein interactions for understanding and grasping the function of lncRNA. The method of experimental detection of unknown lncRNA-protein interactions is very time-consuming and expensive. The earliest proposed method can only identify low-throughput programs for one or a few lncRNAs related to proteins, as the interaction

between lncRNA and protein is many-to-many, experimental methods require multiple rounds of strict screening centered on protein and lncRNA, respectively, to detect a small amount of useful information [5].

1.1.2 Research Significance

This article mainly focuses on the problem of modeling and predicting the interactions between lncRNA and proteins using deep learning algorithms. The research significance can be summarized as follows:

Firstly, this article provides a new algorithm for predicting the interactions between lncRNA and proteins, and the experimental results have reached the most advanced level to date. This prediction algorithm provides a good tool for molecular biologists to screen and predict and therefore can study the mechanisms of complex diseases at the molecular level.

Secondly, these specific biological problems can be abstracted into a mathematical modeling process of relationship prediction. The deep learning modeling process and methods used in this article can be understood as a general process framework, which can be applied to a wider range of novel specific fields in biology.

Thirdly, the deep learning model structure designed based on interpretability in this article can provide practical examples with biological significance for deep learning researchers.

Section 2. Current Research Status

In 2011, Bellucci et al. developed a powerful and comprehensive RNA-protein binding prediction website called CatRAPID [6]. The lncRNA-protein pairs were encoded into a feature vector and scored by matrix calculation. This method is one of the earliest and most widely used lncRNA-protein prediction methods, which can predict 89% of the lncRNA-protein interactions in the NPinter database by integrating secondary structure, hydrogen bonds, and van der Waals forces (also known as intermolecular forces). Subsequently, methods for predicting lncRNA-protein interactions can be roughly divided into the following categories.

1.2.1 Machine Learning-based Approach

In predicting lncRNA-protein interactions, machine learning methods usually use the biological characteristics of lncRNAs and proteins to identify whether a protein has a potential interaction with a specific lncRNA using a supervised classifier. For example, Suresh et al. proposed a method called RPI Pred in 2015, which is based on a support

vector machine (SVM) and uses high-order 3D structure features and sequences of lncRNAs and proteins for prediction [7]; Hu et al. adopted an ensemble strategy based on XGBOOST, SVM, and Random Forest (RF) in 2018, called HLPI Ensembl [8]. However, selecting appropriate features to predict lncRNA-protein interactions is not easy, and the lack of negative samples of lncRNA-protein interactions in these machine learning-based methods may cause the performance of the supervised classifier to decline.

1.2.2 Network-based Approach

Compared with machine learning methods, network-based analysis methods do not require negative samples and are more advantageous. For example, Li et al. proposed a method called LPIHN (LncRNA-protein Interaction Prediction Based on Heterogeneous Network Model) in 2015, which constructed a heterogeneous network and implemented the Random Walk with Restart (RWR) algorithm on it [9]. To improve prediction performance, network-based methods combined with recommendation algorithms were used to infer the interactions between lncRNAs and proteins. Ge et al. proposed a method called LPBNI (LncRNA–protein Bipartite Network Inference) in 2016, which used only known lncRNA-protein interactions and implemented second-order propagation on a bipartite network [10]. Zhao et al. introduced a bipartite network method called LPI-BNPRA (Long Non-coding RNA-Protein Interaction Prediction Based on Improved Bipartite Network Recommender Algorithm) in 2018, which inferred lncRNA-protein interactions using the biased ratings of lncRNAs and proteins and aggregated hierarchical clustering [11].

These methods achieved high accuracy by implementing two rounds of resource allocation on the bipartite network. However, due to the presence of high-order correlations, the predictive effect of these studies is still insufficient and may have a negative impact on predicting lncRNA-protein interactions. To address this issue, Xie et al. proposed LPI-IBNRA (Long Non-coding RNA-Protein Interaction Prediction Based on Improved Bipartite Network Recommender Algorithm) in 2019, which eliminates second-order correlations on the bipartite network using known lncRNA-protein interactions, protein-protein interactions, and lncRNA expression similarity to improve its prediction accuracy and effectiveness [12]. Similarly, in 2020, Zhou et al. proposed an lncRNA-protein interaction prediction model based on a similarity kernel fusion method, called LPI-SKF (Predicting lncRNA-Protein Interactions Using

Similarity Kernel Fusions). For the first time, multiple similarities between lncRNAs and proteins were calculated, and their comprehensive similarity was obtained by integration. Finally, a prediction model was established using the Laplacian regularized least squares framework [13].

1.2.3 Deep learning-based approach

Theoretical research on deep learning is flourishing and its applications are becoming increasingly widespread. Compared to traditional machine learning methods, deep learning involves multiple hidden layers and nonlinear transformations, allowing for a more efficient understanding of the complex associations between lncRNAs and proteins and possessing more effective data interpretation capabilities than traditional machine learning methods [14]. This has also attracted significant attention in the field of bioinformatics, and many research groups at home and abroad have begun to apply this new data analysis method to lncRNA-protein prediction, including studies on predicting the nucleic acid binding properties of protein sequences and the sequence features of DNA or RNA binding proteins, all of which have achieved good results [15]. For example, in 2019, Wekesa et al. proposed a prediction method called PLRPIM (A Hybrid Prediction Method for Plant lncRNA-Protein Interaction), which combines deep learning and shallow machine learning methods and uses neural networks, random forests (RF), and LightGBM (Light Gradient Boosting Machine) to identify potential lncRNA-protein interactions [16].

This article introduces a method for predicting lncRNA-protein interactions using graph autoencoders and collaborative training, called LPIGAC (Predicting lncRNA-protein interactions based on graph autoencoders and collaborative training). This method enhances the robustness and accuracy of the matrix completion and label propagation processes through the use of graph autoencoders (GAE) and the correlation matrix in the end-to-end deep learning framework. Additionally, LPIGAC can improve the effectiveness of GAE representation learning by co-training two GAEs (GAE on lncRNA graph and GAE on protein graph).

LPIGAC is the first application of graph neural network methods represented by GAE in the prediction of lncRNA-protein interactions. In this paper, we will optimize the model and re-adjust relevant parameters to improve its performance, and compare it with other computational methods to demonstrate the advantages of deep learning models in this research direction. Finally, we will replace the network structure of the

graph autoencoder in this algorithm with a graph attention network model to make predictions about the interactions between lncRNAs and proteins.

Section 3. Research Content

This paper mainly studies an end-to-end deep model for predicting the interaction between proteins and lncRNAs. The main contributions are as follows:

(1) The LPIGAC algorithm was realized, with hyperparameters carefully determined and collaborative training conducted. The result of LPIGAC was compared with those of other methods, and the strengths and weaknesses of the model were analyzed.

(2) The graph attention network, which more efficiently integrates the correlation of vertex features in the model and improves the prediction accuracy of lncRNA-protein interaction, was used to replace the graph convolutional neural network in GAE.

Chapter 2. Concepts of Deep Learning

Section 1. Graph Convolutional Neural Networks

The limitations of traditional deep learning methods in dealing with non-Euclidean spatial data have led to growing interest in extending deep learning methods to graphs. With the successful contributions from various factors, researchers drew inspiration from convolutional networks, recurrent networks, and deep autoencoders to design and study Graph Neural Networks (GNN). GNNs refer to neural network structures that operate on graphs and are a type of deep learning method aimed at reasoning with data presented in graph form. GNN models include four types:

(1) Recurrent Graph Neural Networks (RecGNNs): designed for graphs with temporal features [18].

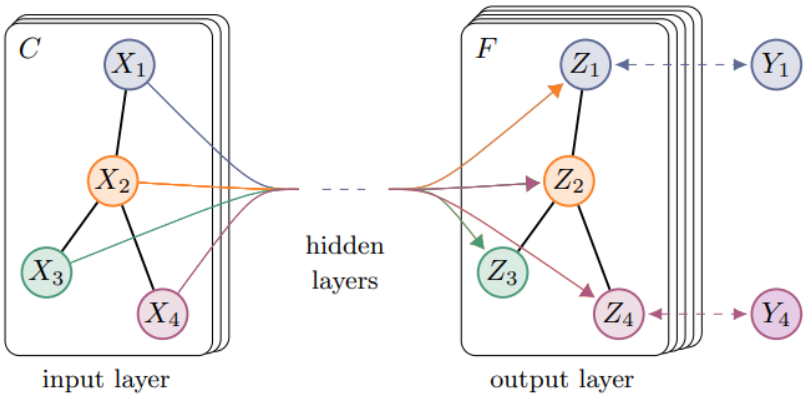
(2) Convolutional Graph Neural Networks (GCN): GCN generalizes the convolution operation from matrix data to graph data [18]. GCN uses stacked multiple graph convolution layers to extract representations of higher-level nodes and plays an important role in constructing more complex GNN models.

(3) Graph Autoencoder (GAE): an unsupervised learning framework that encodes nodes and graphs into a latent vector space and reconstructs graph data from encoded information [19]. GAE can be used for learning network representations and generating graph distributions. For network representation, GAE can learn the representation of latent nodes by reconstructing relevant information about the graph structure, such as the graph adjacency matrix. For graph generation, GAE can output the structural information of the graph either by generating the nodes and edges step by step or by outputting all the information at once.

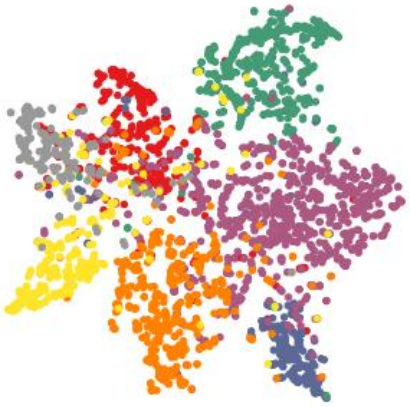
(4) Spatial-Temporal Graph Neural Networks (STGNNs): STGNNs aim to learn hidden patterns from spatial-temporal graphs, and the important idea of this method is to simultaneously consider both spatial and temporal dependencies [20].

This section focuses on the basic principles of Graph Convolutional Neural Networks and their application in predicting lncRNA-protein interaction. Before introducing specific experiments, an example of the exceptional representation power of Graph Convolutional Neural Networks can be illustrated. Fig. 2.1(a) shows a multi-layer graph convolutional neural network for semi-supervised learning with C input channels and F feature maps at the output layer. The graph structure (edges shown in black) is shared between layers, and the labels are represented by Yi. Fig. 2.1(b) shows

the visualization of the hidden layer activation of a two-layer GCN trained on the Cora dataset with 5% labeled data [21], where the colors represent the categories of documents. The strong representation power of GCNs has also attracted widespread attention in the field of bioinformatics.



(a) Graph convolutional network



(b) Hidden layer activation

Fig. 2.1 Diagram of graph convolutional network.

2.1.1 Laplace Matrix and Fourier Transform of Graphs

A graph signal differs from a discrete signal in that it is a signal defined on a node with an intrinsic correlation structure. The nature of graph signals is not only related to the intensity but also to the topology of the graph. The properties of the signals are different even if they have the same intensity on different graphs. Let the graph be $G = (V, E)$, where V is the set of all nodes in the graph, and let the length of the graph be N. The graph signal describes the mapping relationship $V \rightarrow R$, which can be expressed as a vector: $x = [x_1, x_2, \cdots, x_N]^T$, and x_i represent the signal intensity of node v_i. As shown in Fig. 2.2, the values of signal intensity of the nodes can be described by

the length of the vertical lines on the graph.

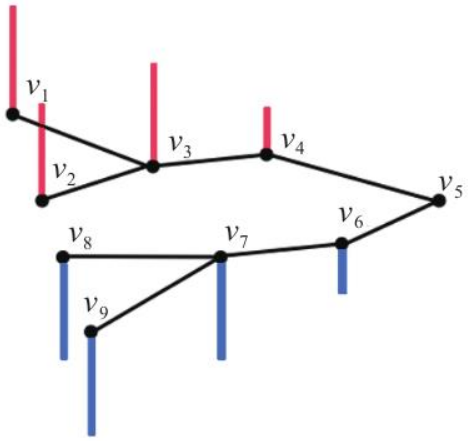


Fig. 2.2 Graph Signals.

To study the construction and characteristics of the graph signal, we must first know the concept and operation of Laplacian matrix. The definition of Laplacian matrix on graph $G = (V, E)$ is $L = D - A$, where D(Degree matrix)is the diagonal matrix representing the degree of each vertex of the graph, and the degree of a vertex can be computed by $D_{ii} = \sum_j A_{ij}$. The adjacency matrix is a common method of storing graph data in a computer, as the adjacency matrix of a graph describes the association between the vertices in the graph. Fig. 2.3 shows how the Laplacian matrix is calculated.

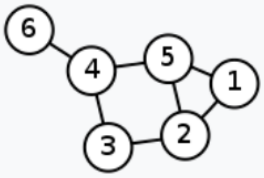
Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Fig. 2.3 Calculation of Laplace matrix.

The Fourier transform is very important in the field of signal processing. It can convert the convolution operation in the time domain into the product operation in the frequency domain, thus greatly simplifying the complexity of the calculation. The Fourier transform of the graph can be obtained analogously from the traditional Fourier transform, and the Laplace matrix is defined to correspond to the Laplace operator in the traditional Fourier transform as follows:

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{2.1}$$

Laplace operator is used widely in image processing, such as as edge detection operator, filter, etc. Its principle can be expressed as:

$$\begin{aligned}\Delta f(x,y) &= \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \\ &= [(f(x+1,y) - f(x,y) - (f(x,y) - f(x-1,y))] \\ &+[(f(x,y+1) - f(x,y)) - (f(x,y) - F(x,y-1))] \\ &= [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)\end{aligned} \tag{2.2}$$

In fact, the Laplacian operator can achieve edge detection by reflecting the difference between the central and surrounding signals. Similarly, the Laplacian matrix can also characterize the smoothness of the graph signal by describing the relationship between the center and the surrounding in the graph signal. The smoothness of the graph signal can be characterized by the total variation:

$$TV(x) = x^T Lx = \sum_{e_{ij} \in E} (x_i - x_j)^2 \tag{2.3}$$

The traditional Fourier transform converts the convolution operation in the time domain space into a multiplication operation in the frequency domain space. Similarly, the graph Fourier transform achieves the conversion of spatial domain convolution into multiplication operation in the frequency domain. As the Laplacian matrix is a real symmetric matrix, it can be orthogonally diagonalized [18]:

$$L = V\Lambda V^T = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ v_1 & v_2 & \cdots & v_N \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} \cdots & v_1 & \cdots \\ \cdots & v_2 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & v_N & \cdots \end{bmatrix} \tag{2.4}$$

where $V \in R^{N \times N}$ is an orthogonal matrix. The graph Fourier transform (GFT) can be defined as:

$$\tilde{x}_k = \sum_{i=1}^{N} V_{ki}^T x_i = \langle v_k, x \rangle \tag{2.5}$$

where $\tilde{x}_k$ denotes the Foureir coefficient of x at the kth characteristic vector. The corresponding matrix representation is:

$$\tilde{x} = V^T x, \tilde{x} \in R^N \tag{2.6}$$

Since the matrix V is orthogonal, i.e., $VV^T = I$, by left multiplication of V on Eq. (2.6) we have: $V\tilde{x} = VV^T x = Ix = x$, and therefore:

$$x = V\tilde{x}, x \in R^N \tag{2.7}$$

Accordingly, the definition of Inverse Graph Fourier Transform (IGFT) can be derived:

$$x_k = \sum_{i=1}^{N} V_{ki} \bullet \tilde{x}_i \tag{2.8}$$

The connection between the graph Fourier transform and the frequency of the graph signal can be understood by obtaining the formula for the Fourier transform of the graph signal. First, the formula for the total variance of the graph signal is transformed:

$$\begin{aligned} TV(x) &= x^T L x = x^T V \Lambda V^T x \\ &= (V\tilde{x})^T V \Lambda V^T (V\tilde{x}) \\ &= \tilde{x}^T \Lambda \tilde{x} \\ &= \sum_{k}^{N} \lambda_k \tilde{x}_k^2 \end{aligned} \tag{2.9}$$

As shown in Eq. (2.9), the total variance of the graph signal is a linear combination of the eigenvalues of the graph, whose coefficients are the squared Fourier coefficients of the graph, thus portraying the overall smoothness of the graph signal.

2.1.2 Graph Filter and Graph Convolutional Neural Network

Before presenting the definition of a graph convolutional neural network, it is necessary to introduce the graph filter, the operation in graph signal processing that can enhance or weaken the intensity of each frequency component in the spectrum of a given graph signal [18]. Let $H \in R^{N \times N}$ be the graph filter, and there exists a correspondence $H : R^N \to R^N$. With y as the output, the deifinition of the graph filter is:

$$\begin{aligned} y = Hx &= \sum_{k=1}^{N} \left(h(\lambda_k) \tilde{x}_k \right) v_k = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ v_1 & v_2 & \cdots & v_N \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \begin{bmatrix} h(\lambda_1)\tilde{x}_1 \\ h(\lambda_2)\tilde{x}_2 \\ \vdots \\ h(\lambda_N)\tilde{x}_N \end{bmatrix} \\ &= V \begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_N \end{bmatrix} \\ &= V \begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} V^T x \end{aligned} \tag{2.10}$$

From Eq. (2.10), it is known that

$$H = V\begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} V^T = V\Lambda_h V^T \tag{2.11}$$

The graph filter H frequency response matrix Λ_h, where different frequency response function $h(\lambda)$ will lead to different filtering results. Frequency response functions of arbitrary function types can be realized by Taylor expansion-polynomial approximation, thus realizing graph filters with different properties. This can be achieved with the extended Laplace matrix polynomial:

$$H = h_0 L^0 + h_1 L^1 + h_2 L^2 + \cdots + h_K L^K = \sum_{k=0}^{K} h_k L^k \tag{2.12}$$

On the basis of the graph filter the definition of the convolution operation of graph signals can be proposed. Let x_1 and x_2 be graph signals of two graphs, their convolution is:

$$\begin{aligned} x_1 * x_2 &= IGFT(GFT(x_1) \odot GFT(x_2)) \\ &= V((V^T x_1) \odot (V^T x_2)) = V(\tilde{x}_1 \odot (V^T x_2)) \\ &= V(diag(\tilde{x}_1)(V^T x_2)) \\ &= (Vdiag(\tilde{x}_1)V^T)x_2 \end{aligned} \tag{2.13}$$

Let $H_{\tilde{x}_1} = Vdiag(\tilde{x}_1)V^T$, the frequency spectrum of x_1 is the frequency response matrix of $H_{\tilde{x}_1}$, and the graph convolution operation of x_1 and x_2 is equivalent to the graph filtering operation, as shown in Eq. (2.14)

$$x_1 * x_2 = H_{\tilde{x}_1} x_2 \tag{2.14}$$

The convolution operation of graph signals is equivalent to the filtering operation of the graph, then when defining the convolutional neural network layer the frequency response matrix of the graph filter operator can all be parameterized [18]. Let the input matrix of the graph signal be X, the output matrix be X', and the graph filter be Θ, then the parameters to be learned in the filter are $\theta = [\theta_1, \theta_2, \cdots, \theta_N]$, where $\sigma(\cdot)$ is the activation function:

$$
\begin{aligned}
X^{'} &= \sigma(V\begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \ddots & \\ & & & \theta_4 \end{bmatrix} V^T X) \\
&= \sigma(V diag(\theta) V^T X) \\
&= \sigma(\Theta X)
\end{aligned} \tag{2.15}
$$

Similarly, to obtain a suitable frequency response matrix for the graph filter operator, the polynomial parameters to be learned in the graph filter can be transformed:

$$
X^{'} = \sigma\left(V\left(\sum_{k=0}^{K} \theta_k \Lambda^k \right) V^T X \right) = \sigma\left(V diag\left(\Psi\theta \right) V^T X \right) \tag{2.16}
$$

where Ψ is a van der Mond matrix. The coefficients of the Laplace matrix polynomial can be obtained by $\theta = \psi^{-1} diag^{-1}\left(\Lambda_h \right)$. In Eq. (2.16), $\theta = \left[\theta_1, \theta_2, \cdots, \theta_K \right]$ is the coefficient vector of the expanded Laplacian matrix polynomial, i.e., the core part of the graphical convolutional network layer to be learned. The complexity of the filtering relationship between the graph input matrix and the output matrix can be adjusted with K. In order to reduce the computational complexity, let $K = 1$ and $\theta_0 = \theta_1 = \theta$, such that:

$$
X^{'} = \sigma\left(\theta_0 X + \theta_1 L X \right) = \sigma\left(\theta\left(I + L \right) X \right) = \sigma\left(\theta \tilde{L} X \right) \tag{2.17}
$$

To make the training process of the convolutional network more stable , based on the regularization of the Laplacian matrix, let $\tilde{L}_{sym} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, $\tilde{A} = A + I$,and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, then $\tilde{L}_{sym}$ is the renormalized Laplacian matrix. Also, to avoid gradient disappearance or explosion when layer number of the convolutional network increases, the range of eigenvalues for $\tilde{L}_{sym}$ is $\left(-1, 1 \right]$. Further, the fitting ability of the convolutional network can be improved by applying an affine transformation to the graph input matrix using a parameterized weight matrix W , i.e.:

$$
X^{'} = \sigma\left(\tilde{L}_{sym} X W \right) \tag{2.18}
$$

Eq. (2.18) is the graph convolution layer in the graph convolution model, of which the node-level calculation is shown in Eq. (2.19):

$$
x_i = \sigma\left(\sum_{v_j \in \tilde{N}(v_i)} \tilde{L}_{sym}\left[i, j \right] \left(W x_j \right) \right) \tag{2.19}
$$

Section 2. Graph Autocoder and End-to-end Learning

The graph autoencoder with graph convolutional layers is a type of autoencoder. In this study, we use Graph Convolutional Networks (GCN) as the encoder and decoder convolutional layers to extract features and output results from the input lncRNA and protein graphs. GCN is a function that takes node features and adjacency matrices as input and node embeddings as output, and can simulate the process of label propagation [19]. By co-training two graph autoencoders on lncRNA and protein graphs, the corresponding network model can achieve better prediction performance through the representation learning of the two-layer graph convolutional encoder. The specific implementation process is detailed in the experimental section of Chapter 3.

The end-to-end learning is a representation learning method that differs from traditional machine learning models. It does not require manually extracting descriptive features, but uses raw data as input and automatically extracts features through neural networks and outputs predicted results. The model in this study has powerful representation learning capabilities and can achieve end-to-end prediction tasks through graph convolutional neural networks and reconstruction loss methods.

Section 3. Graph Attention Network

Graph attention networks (GAT) is a GNN model with an attention mechanism that can accomplish adaptive assignment of weights to different neighbors, which greatly improves the expressive power of GNN models [22].The GAT layer is:

$$z_i^{l+1} = \sum_{j \in \mathcal{N}(i)} s_{ij} W^{(l)} z_j^{(l)} \tag{2.20}$$

where s_{ij} is the attention fraction between node i and j. Assume v_i is the central node, and the weight coefficient from neighboring node v_j to v_i is e_{ij} [23]:

$$\begin{aligned} s_{ij} &= \mathrm{softmax}\left(e_{ij}\right) \\ e_{ij} &= \mathrm{LeakyRELU}\left(a^T \left[\mathrm{W}h_i \| Wh_j \right]\right) \end{aligned} \tag{2.21}$$

where a denotes the parameters to be learned in the graph attention network. The activation function is defined as:

$$\mathrm{LeakyRELU}\left(x\right) = \begin{cases} x & x \geq 0 \\ 0.2x & x < 0 \end{cases} \tag{2.22}$$

The process of applying the attention layer on the graph can be expressed as:

$$Z^{(l+1)} = Att\left(G, Z^{(l)}\right) \tag{2.23}$$

Chapeter 3. lncRNA-protein Interaction Prediction Model Based on GAE

LPIGAC is a method for predicting lncRNA-protein interaction using graph autoencoders. It uses graph autoencoders based on graph convolutional neural networks to reconstruct the association matrix and improve the effectiveness of GAE representation learning through co-training.

Section 1. Model Structure

3.1.1 Experimental Principle

The interaction information between lncRNA and protein comes from biological knowledge, and complex molecular features include the number of atoms, hydrogen bonds, evolutionary conservation scores, interaction tendencies, and electrostatic charges. Predicting lncRNA-protein interactions can be achieved by inputting sequence features. The protein sequences in the original dataset are composed of 20 amino acids, and the lncRNA sequences are composed of 4 nucleotides. By embedding these protein and lncRNA sequences, they can be transformed into effective input features. In this experiment, Word2Vec is used to learn the embedding vectors of lncRNA and protein from the sequences, where Word2Vec is an effective method for learning to embed words from natural language into vector space. BioVec[17] applies Word2Vec to the representation learning of biological sequences, including protein and nucleotide sequences. The embedded vectors in this experiment are fixed to 300.

After obtaining the embedding vector, the number of lncRNAs and proteins are denoted by *m* and *n*, respectively, and the lncRNA-protein interaction matrix is denoted by $\boldsymbol{Y} \in \boldsymbol{R}^{m \times n}$. If lncRNA i is known to be associated with protein j, then $Y(i,j) = 1$, otherwise $Y(i,j) = 0$. Predicting lncRNA-protein interactions also requires lncRNA similarity matrix $\boldsymbol{S}_l \in \boldsymbol{R}^{m \times m}$, protein similarity matrix $\boldsymbol{S}_p \in \boldsymbol{R}^{n \times n}$ and optimal score matrix $\boldsymbol{F} \in \boldsymbol{R}^{m \times m}$, where $F(i,j)1 \in [0,1]$ denotes the prediction score of the association between lncRNA i and protein j. A higher score indicates a higher probability of association between them.

3.1.2 Similarity Matrix

The similarity matrix can be considered as the adjacency matrix A of the lncRNA graph or protein graph. Each lncRNA or protein represents a node of the graph, and the graph can be constructed simply by the following method: first, sort the nodes according to the Euclidean distance between their different embedding vectors; then for each node i, choose the nearest 5 nodes other than itself; finally, assume that the set of neighboring nodes of the node is $\mathcal{N}(i)$, the matrix C satisfies if $j \in \mathcal{N}(i)$ $C_{ij} = 1$, otherwise $C_{ij} = 0$. The self-looping adjacency matrix of the graph can then be constructed as:

$$S = C^T \odot C + I \tag{3.1}$$

where $\odot$ denotes the Hadamard product. The similarity matrices of lncRNA and protein are both calculated in this way.

3.1.3 Graph Autoencoder Based on GCN

In this chapter, we propose a collaborative training approach using two graph autoencoders on the lncRNA and protein graphs, respectively. The corresponding network model can achieve better prediction results by learning representations through a two-layer graph convolutional encoder. Z_l and Z_p denote the encoder consisting of two layers of graph convolution respectively:

$$Z_l = \tanh\left(A_l \cdot \mathrm{ReLU}\left(A_l Y \Theta^{(0)}\right)\Theta^{(1)}\right) \tag{3.2}$$

$$Z_p = \tanh\left(A_p \cdot \mathrm{ReLU}\left(A_p Y \Theta^{(0)}\right)\Theta^{(1)}\right) \tag{3.3}$$

where Θ and Φ denote the weights of the graph convolutional neural network, A_l and A_p denote the normalized adjacency matrices of the lncRNA graph and protein graph. The definitions of A_l and A_p are:

$$A_p = D_p^{-1/2} S_p D_p^{-1/2} \tag{3.4}$$

$$A_l = D_l^{-1/2} S_l D_l^{-1/2} \tag{3.5}$$

in Eq. (3.4) D_l, the degree matrix of S_l, is also a diagonal matrix, which can be obtained by $D_l(i,\ i) = \sum_j S_l(i,\ j)$. Eq. (3.5) is obtained in the same way. The decoding of Z_l and Z_p is subsequently performed to obtain the score matrices $F_l \in \mathbb{R}^{m \times n}$ and $F_p \in \mathbb{R}^{m \times n}$:

$$F_l = \text{sigmoid}\left(A_l \cdot \text{ReLU}\left(A_l Z_L \Theta^{(2)}\right)\Theta^{(3)}\right) \tag{3.6}$$

$$F_p = \text{sigmoid}\left(A_p \cdot \text{ReLU}\left(A_p Z_p \Phi^{(2)}\right)\Phi^{(3)}\right) \tag{3.7}$$

GAE also uses the graph convolution layer as a decoder to reconstruct the original graph, and then obtains the reconstructed adjacency matrices $F_l \in \mathbb{R}^{m\times n}$ and $F_p \in \mathbb{R}^{m\times n}$. It is desirable for the reconstructed adjacency matrix to be as close and similar as possible to the original adjacency matrix because the adjacency matrix determines the basic structure of the graph. Therefore, during the training process of GAE, the method of minimizing reconstruction error can be used to train the graph autoencoder:

$$L_r = \alpha\left\|Y - F_l\right\|_F^2 + \left(1-\alpha\right)\left\|Y - F_p^T\right\|_F^2 \tag{3.8}$$

where $\alpha \in (0,\ 1)$ plays a role in balancing the weights of the parameters obtained in the lncRNA and protein vector spaces.

3.1.4 Collaborative Training of GAE

The defining the loss function in Eq. (3.8) indicates training the graph autoencoder on the lncRNA graph and the protein graph separately. However, it has been shown that collaborative training integrating information from both sides can improve the prediction accuracy[24]. Herein, the learning results of autoencoder representation, Z_l and Z_p, are used to define the loss fro collaborating training:

$$L_c = \frac{1}{2}\left\|Y - Z_l Z_p^T\right\|_F^2 \tag{3.9}$$

the ℓ_2 norms of Θ and Φ are added to avoid overfitting. The loss function is defined as:

$$\min_{\Theta,\Phi} L_c + \beta L_r + \lambda\left\|\Theta\right\|_F^2 + \lambda\left\|\Phi\right\|_F^2 \tag{3.10}$$

where $\lambda = 10^{-7}$, and β will be determined in the following experiments. The final result is a linear combination of F_l and F_p:

$$F = \alpha F_l + \left(1-\alpha\right)F_p^T \tag{3.11}$$

The process of the LPIGAC algorithm is shown in Fig. 3.1, where GAEl and GAEp represent graph autoencoders on the lncRNA graph and protein graph,

respectively. Algorithm 1 represents the computational process of a GAE.

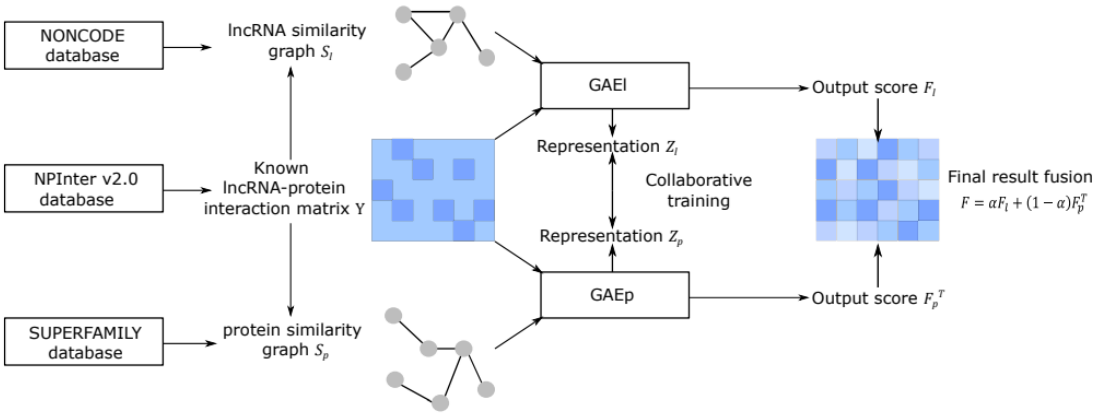


Fig. 3.1 LPIGAC

Algorithm 1 LPIGAC

Require: initial interaction matrix Y, lncRNAsequence and protein sequence, α, β

Ensure: the score matrix F

1: The embedding vectors of lncRNA and protein can be learned from the sequences by Word2Vec.

2: The similarity matrices of lncRNA and protein, S_l and S_p, are calculated separately by Eq. (3.1).

3: The adjacency matrices of the lncRNA and protein graphs, A_l and A_p, are calculated by Eqs. (3.4) and (3.5) respectively.

4: **Repeat**

5: By Eqs. (3.2) and (3.6), it can be obtained: $F_l, Z_l \leftarrow \mathrm{GAEl}(A_l, Y)$

6: By Eqs. (3.3) and (3.7), it can be obtained: $F_p, Z_p \leftarrow \mathrm{GAEl}(A_p, Y^T)$

7: Train GAEl and GAEp by optimizing the loss in Eq. (3.10).

8: **until** convergence

9: $F = \alpha F_l + (1-\alpha)F_p^T$

10: **return** F

3.1.5 Graph Autoencoder Based on GAT

The graph autoencoder is a type of autoencoder with graph convolution layers, which can simulate the propagation process of labels and achieve good results in predicting the lncRNA-protein interaction. In this section, we propose a graph autoencoder with attention layers on both the lncRNA graph and the protein graph.

Similar to the GCN-based models, we use the same dataset and graph input matrix for each lncRNA or protein, which represents a node in the lncRNA or protein graph. The encoder is composed of two graph attention layers, Z_l and Z_p:

$$Z_l = \tanh\left(\mathrm{Att}\left(G_l, \mathrm{Att}\left(G_l, Y\right)\right)\right) \tag{3.12}$$

$$Z_p = \tanh\left(\mathrm{Att}\left(G_p, \mathrm{Att}\left(G_p, Y^T\right)\right)\right) \tag{3.13}$$

The matrices $F_l \in \mathbb{R}^{m\times n}$ and $F_p \in \mathbb{R}^{m\times n}$ are obtained by decoding Z_l and Z_p, respecitively:

$$F_l = \mathrm{sigmoid}\left(\mathrm{Att}\left(G_l, \mathrm{Att}\left(G_l, Y\right)\right)\right) \tag{3.14}$$

$$F_p = \mathrm{sigmoid}\left(\mathrm{Att}\left(G_p, \mathrm{Att}\left(G_p, Y\right)\right)\right) \tag{3.15}$$

After defining the graph autoencoder of the GAT model using the above formula, other operations are the same as using the GCN model, including defining the loss function and designing the joint training model. It should be noted that to avoid overfitting, a regularization term is added to the final loss function:

$$\min_{W} L_c + \beta L_r + \lambda \left\| W \right\|_F^2 \tag{3.16}$$

Section 2. Dataset and Evaluation Indexes

The dataset used in the experiment was retrieved from the NPInter v2.0 database (http://bigdata.ibp.ac.cn/npinter), which includes 990 lncRNAs, 27 proteins, and 4158 lncRNA-protein interaction relationships between them. The lncRNA sequences can be downloaded from the NONCODE database (http://www.noncode.org/), and the protein sequences can be downloaded from the SUPERFAMILY database (http://supfam.org/).

To evaluate the predictive ability of the proposed model, the experiment used the 5-fold cross-validation method. Cross-validation can divide the dataset into training and test sets through iterations to reduce sampling bias. These data were randomly divided into five groups of the same size, with one group (fold) reserved as the test set and the other four as the training set. The results of the classifier can be represented by a confusion matrix, which contains four types of results: TP is the actual lncRNA-protein pairs predicted correctly, TN is the lncRNA-protein pairs correctly extracted from negative samples, FP is the negative samples predicted incorrectly, and FN is the case where the actual lncRNA-protein pairs are predicted as non-interacting. The confusion matrix, including the four categories of results, is shown in Table 3.1.

Table 3.1 Four types of results for the confusion matrix

Reality / Prediction	Condition Positive	Condition Negative
Outcome Positive	True Positive, TP	False Positive, FP
Outcome Negative	False Negative，FN	True Negative, TN

For the specific evaluation of the training results, the following five standard indicators can be used, including the area under the ROC curve (AUC), precision, recall, true positive rate (TPR), and false positive rate (FPR), to evaluate the average performance [25]. The larger the AUC value, the better the predictive ability of the model. The above standard indicators can be defined by these four categories of results:

$$\text{Precision: } P = \frac{TP}{TP+FP}$$

$$\text{Recall: } R = \frac{TP}{TP+FN}$$

$$\text{Sensitivity: } TPR = \frac{TP}{TP+FN}$$

$$\text{1-specificity: } FPR = \frac{FP}{FP+TN}$$

Based on the formulas for precision, recall, and specificity, ROC and PR curves can be plotted. Taking TPR as the y-axis and FPR as the x-axis, the ROC curve can be directly obtained. It can be understood from the definition of FPR and TPR that the higher the TPR, the smaller the FPR value, indicating better performance of the model. AUC value is the main criterion for measuring algorithm and model performance in this paper. Taking precision as the y-axis and recall as the x-axis, the PR curve can be obtained. It can be understood from the definition of precision and recall that the higher the precision, the higher the recall, and the more efficient the model and algorithm, as shown in Fig. 3.3, the PR curve drawn is closer to the upper right corner.

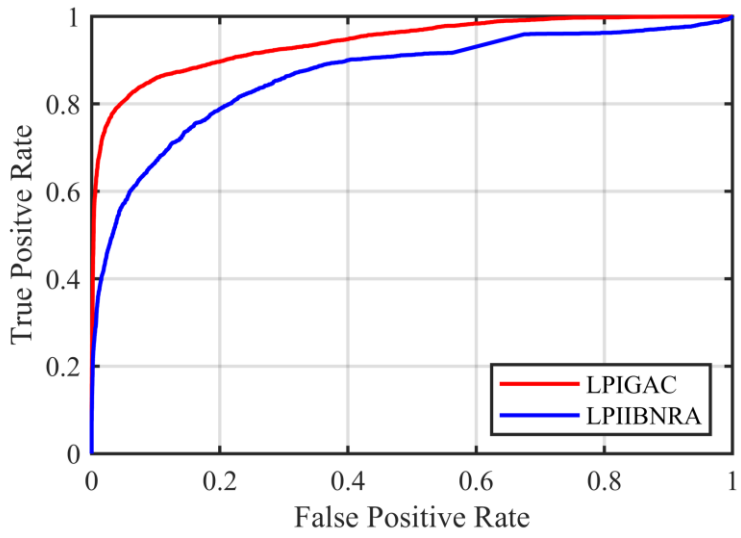


Fig. 3.2 Example of ROC curve

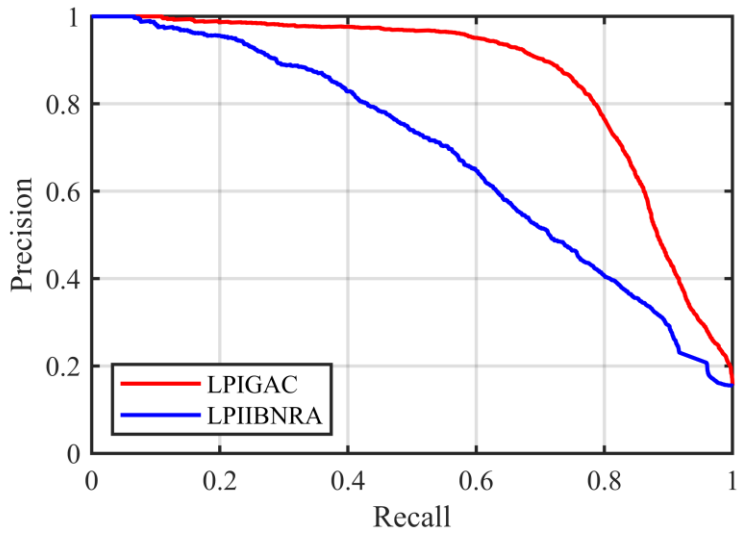


Fig. 3.3 Example of PR curve

In this section, the LPIGAC model was compared to the LPI-IBNRA model, which is a network-based analysis method used as the control group. The AUROC (Area Under the Receiver Operating Characteristic Curves) value of LPIGAC was found to be 0.932, while the AUROC value of LPI-IBNRA was 0.8660. Similarly, the AUPR (Area Under the Precision-Recall Curves) value of LPIGAC was 0.806, and the AUPR value of LPI-IBNRA was 0.6840. LPI-IBNRA is a highly precise and effective model algorithm for network-based prediction methods in recent years. However, through comparative experiments, we found that under the same conditions, the LPIGAC model based on deep learning has better average performance than LPI-IBNRA.

Chapter 4. Experiment on the Prediction of lncRNA-Protein Interaction Based on GAE

Section 1. Parameter Tuning and Model Optimization

In this chapter, the optimal hidden layer dimension, optimizer, and corresponding learning rate for the LPIGAC model were determined, and the significant improvement in model's performance was witnessed.

4.1.1 Loss Function

By comparing different loss functions in trials, it was found that using binary cross-entropy can achieve better results:

$$\mathcal{L} = -\frac{1}{N}\sum y \log \hat{y} + (1-y)\log(1-\hat{y}) \tag{4.1}$$

In the above equation, y represents the value of an element in the adjacency matrix A (0 or 1), $\hat{y}$ represents the corresponding element value (between 0 and 1) in the reconstructed adjacency matrix $\hat{A}$. By replacing the loss function in Eq. (3.8) with this loss function, the AUROC value of this model is 0.9424 and the AUPR value is 0.8591, and the overall performance is better than the original model.

4.1.2 Hidden Layer Dimension

The number of hidden layers has a significant impact on the model's performance. In this section, the performance of the model was evaluated under different hidden layer dimensions, and the results are shown in Table 4.1. The data in the table are the average results and their standard deviations obtained from five-fold cross-validation experiments.

Table 4.1 AUROC, AUPR and their standard deviations under different hidden layer dimensions

Dimension	AUROC	AUPR
25	0.9051±0.0072	0.7358±0.0244
36	0.9096±0.0097	0.7490±0.0333
49	0.9184±0.0034	0.7763±0.0078
64	0.9241±0.0040	0.7964±0.0102
81	0.9273±0.0037	0.8021±0.0085
100	0.9273±0.0037	0.8039±0.0099
121	0.9296±0.0016	0.8102±0.0036
144	0.9310±0.0021	0.8091±0.0042
169	0.9283±0.0035	0.8038±0.0140
196	0.9283±0.0029	0.8020±0.0131
225	0.9350±0.0041	0.8353±0.0145
256	0.9255±0.0030	0.8002±0.0106
289	0.9221±0.0035	0.7863±0.0049
324	0.9045±0.0119	0.7510±0.0255

To better illustrate the relationship between model results and hidden layer dimensions, Figs. 4.1 and 4.2 show the maximum values of AUROC and PR as a function of hidden layer dimensions. The performance of the model improves as the number of hidden layers increases, with results peaking at a hidden layer dimension of 225, where the AUROC value is 0.9424 and AUPR value is 0.8591. However, when the number of hidden layers is greater than 144, the results stabilize, and the performance starts to decline when the hidden layer dimension exceeds 225. Based on these observations, a hidden layer dimension of 225 is optimal for obtaining the best performance.

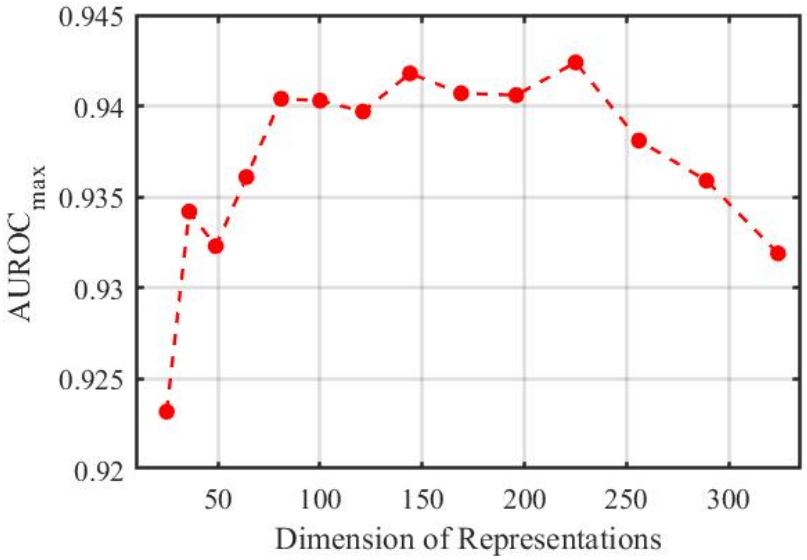


Figure 4.1 AUROC with different hidden layer dimensions

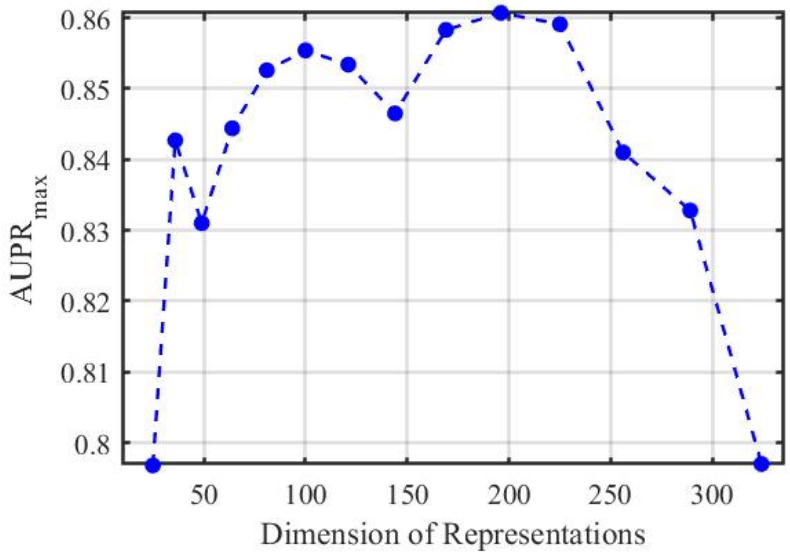


Figure 4.2 AUPR with different hidden layer dimensions

4.1.3 Optimizer and Learning Rate

To predict the relationship between biological genetic sequences using a deep learning model, the parameters of the network are continuously adjusted to transform the input matrix and fit the output through nonlinear transformations, thus obtaining the optimal solution. The optimizer is an algorithm for improving parameters, and designing and selecting an optimizer is a key research topic in deep learning. This section mainly uses several common optimizers to compare the optimization results and select the appropriate optimizer. As shown in Table 4.2, Adam, Adamax, and Adagrad algorithms achieve good optimization results with the same learning rate, while the poor performance of other optimizers is related to the setting of the learning rate and the incompatibility of the data type, which will not be further verified in this experiment.

The advantage of the AdaGrad algorithm is that it can automatically adjust the learning rate of model parameters and modify sparse data on a large scale while making small modifications to frequent parameters. However, accumulating the square of the gradient will cause the learning rate to decrease prematurely. Adam and Adamax are both deep neural network methods that can achieve adaptive learning rate. They dynamically adjust the learning rate by estimating the first and second moments of the gradient. After adjusting the learning rate of these three algorithms as shown in Table 4.3, Adam has the best overall performance after optimizing the model and is chosen as the optimizer for this model.

Overall, according to Tables 4.2 and 4.3, and the change curves of the maximum values of AUROC and AUPR with different learning rates (as shown in Fig. 4.3), the model's performance reaches its peak when the learning rate is 0.009, with an AUROC value of 0.9424 and an AUPR value of 0.8591, both better than the values before adjusting the learning rate.

Table 4.2 AUROC, AUPR and their standard deviations for different optimizers

Optimizer	AUROC	AUPR
SGD	0.5747±0.0263	0.2047±0.0244
ASGD	0.5747±0.0263	0.2047±0.0244
Rprop	0.9078±0.0115	0.7410±0.0213
Adagrad	0.9210±0.0026	0.7843±0.0043
Adadelta	0.5647±0.0282	0.1985±0.0235
RMSprop	0.8354±0.0143	0.5628±0.0112
Adam	0.9350±0.0041	0.8353±0.0145
Adamax	0.9267±0.0027	0.8055±0.0045
Momentum	0.7022±0.0089	0.3371±0.0400

Table 4.3 AUROC, AUPR and their standard deviations for different learning rates

Learning Rate	AUROC	AUPR
0.001	0.9160±0.0018	0.7976±0.0047
0.005	0.9244±0.0034	0.8076±0.0091
0.008	0.9270±0.0041	0.8145±0.0079
0.009	0.9350±0.0041	0.8353±0.0145
0.01	0.9342±0.0048	0.8120±0.0124
0.02	0.8778±0.0443	0.6591±0.1267
0.05	0.6997±0.0648	0.3864±0.0796
0.1	0.7597±0.0899	0.4379±0.0962
0.5	0.6360±0.0207	0.3831±0.0247

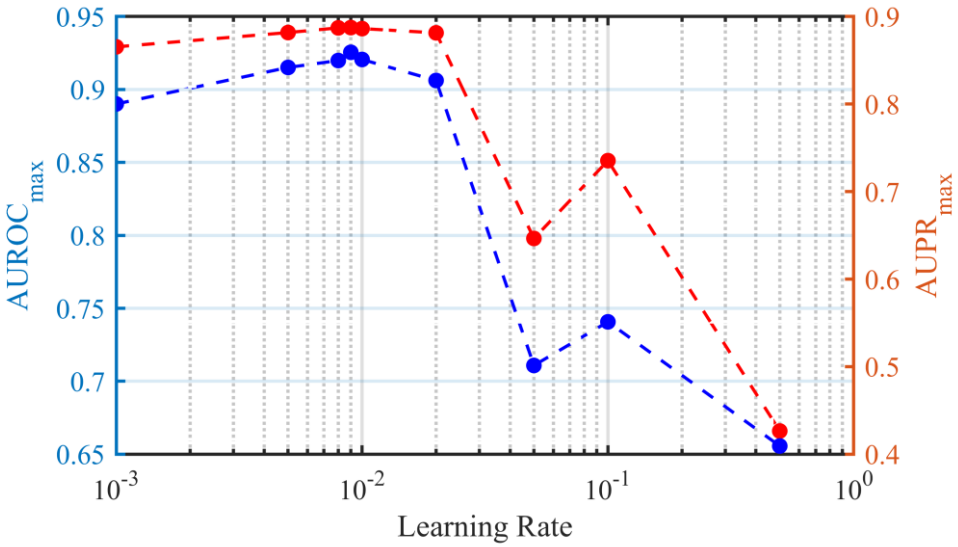


Figure 4.3 AUROC and AUPR for different learning rates

4.1.4 Collaborative Training

The GAE model's loss function for collaborative training has two parameters that need to be adjusted. The parameter α in Eq. (3.8) can balance the weight of the parameters obtained in the vector space and adjust the proportion of collaborative training between the lncRNA graph and the protein graph in the output matrix. The impacts of α and β are not independent.

In the previous content, this paper adjusted the loss function, hidden layer dimension, optimizer, and learning rate of LPIGAC, and the results of the model (AUROC=0.9424, AUPR=0.8591) were improved compared to basic LPIGAC (AUROC=0.9320, AUPR=0.8060). The previous LPIGAC set α=0.5 to uniformly fuse the information obtained from the lncRNA graph and the protein graph and simply set β=1. To further improve the performance of LPIGAC, considering the dependence

between the two hyperparameters α and β, this paper used the following grid search method to determine the optimal α and β, as shown in Figs. 4.4-4.6.

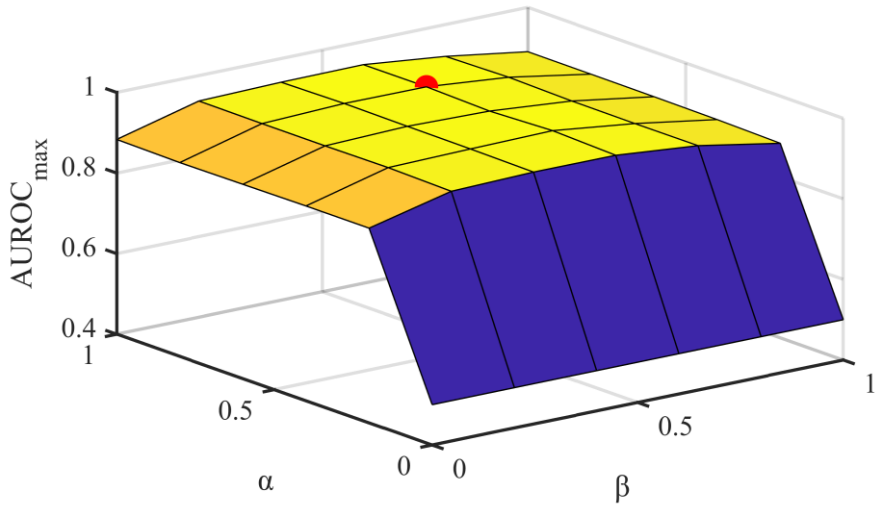


Fig. 4.4 AUROC with different α and β

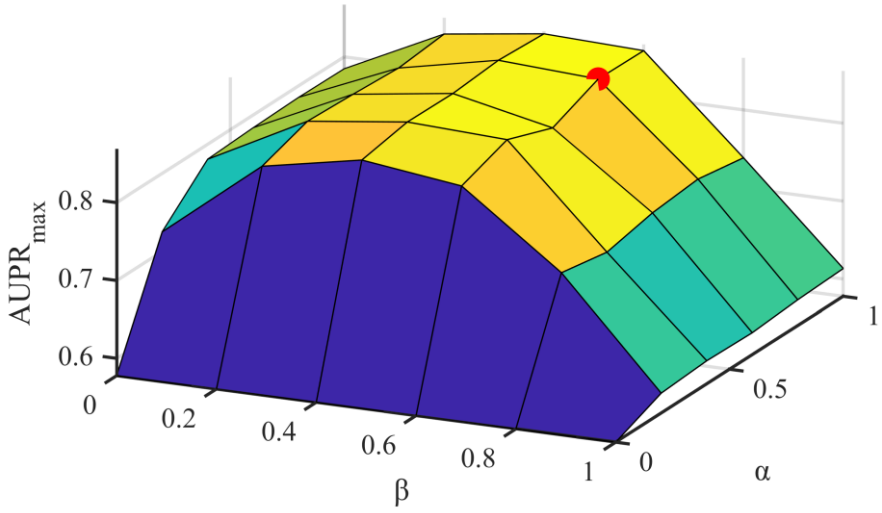


Fig. 4.5 AUPR with different α and β

In experiments the sets of α and β are: $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, $\beta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Herein, when α=1, the training is only with the lncRNA graph, and when α=0, the training is only with the protein graph. The training results show that collaborative training has better performance than using any single input. To find the best parameters more accurately, Figure 4.6 shows the trend of the sum of AUROC and AUPR with the changes of α and β. Finally, the best results are obtained

when α=0.6 and β=0.8, with AUROC=0.9424 and AUPR=0.8601.

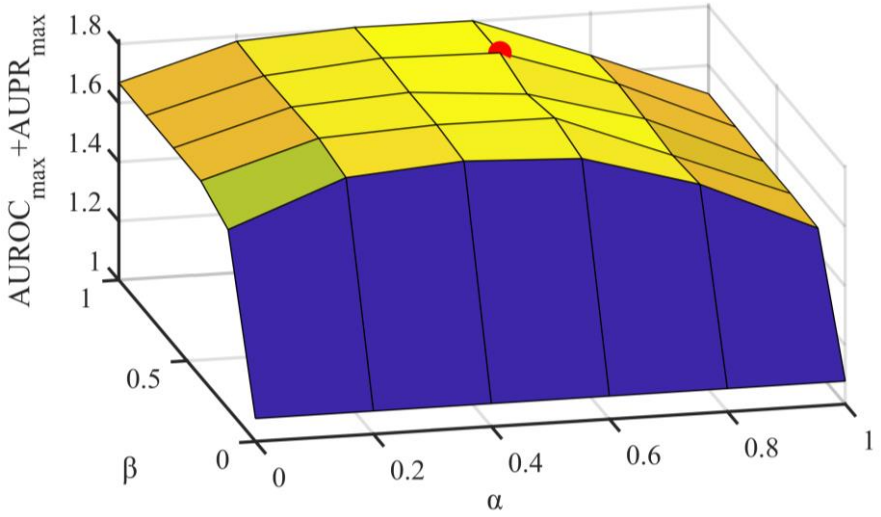


Fig. 4.6 AUROC+AUPR with different α and β

Section 2. GCN Model Comparison Experiment

In the comparative experiment, the improved LPIGAC is compared with other models, including two commonly used algorithms: Random Walk with Restart (RWR) and Collaborative Filtering (CF), as well as four cutting-edge computational methods: LPIHN, LPBNI, LPI-IBNRA, and LPI-SKF. The principles and technical backgrounds of these four methods have been briefly introduced in the introduction, and this section mainly compares their output results with LPIGAC.The experiment uses 5-fold cross-validation to evaluate performance, and Figs. 4.7 and 4.8 show the ROC and PR curves of LPIGAC and other models.

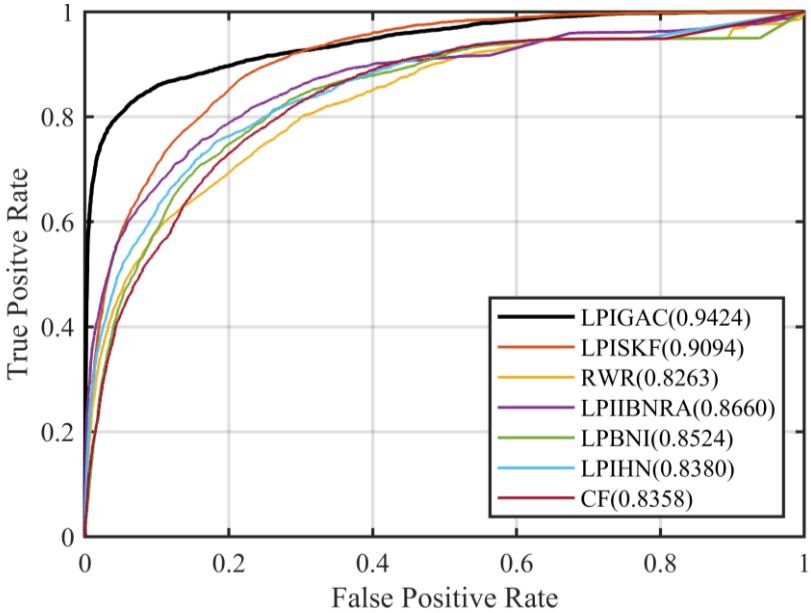


Fig. 4.7 ROC curves of different models

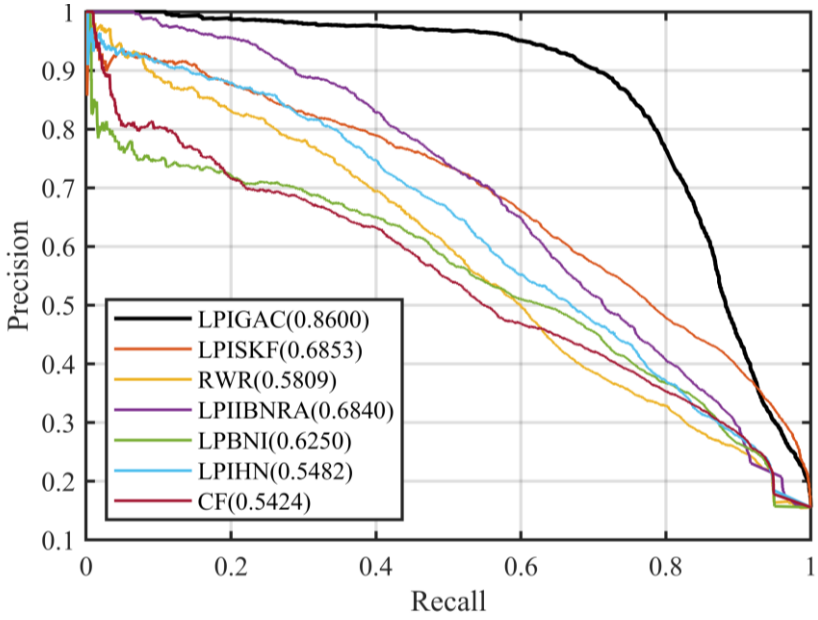


Fig. 4.8 PR curves for different models

The model threshold can then be selected based on the best F1 score, and the recall rate, precision, and F1 score can be calculated as evaluation indicators for this comparison experiment, as shown in Table 4.4. From the comprehensive ROC and PR curves and the data in the table, it can be seen that the AUROC value of the improved LPIGAC is 0.9424 and the AUPR value is 0.8601, both of which are higher than other models. At the same time, LPIGAC's best F1 score is also better than other models, demonstrating the superior performance of this model.

Table 4.4 Comparison of LPIGAC with different models

Method	AUROC	AUPR	Recall	Precision	F1-Score
LPIGAC	0.9424	0.8601	0.747	0.683	0.801
LPISKF	0.9094	0.6853	0.625	0.641	0.633
RWR	0.8263	0.5809	0.977	0.157	0.271
LPIIBNRA	0.8660	0.6840	0.599	0.651	0.624
LPIHN	0.8380	0.5482	0.668	0.479	0.558
LPBNI	0.8524	0.6250	0.633	0.533	0.579
CF	0.8358	0.5424	0.633	0.459	0.532

Section 3. Experimental Results of GAT Model

After model optimization and parameter tuning for the GAT model, the final results are shown in Figures 4.9 and 4.10, with an AUPR value of 0.8136. This result is superior to the basic LPIGAC with 0.8058, as well as to other compared models, but lower than the LPIGAC model improved and adjusted in this paper. The main reason is that there are too many parameters that need to be trained in the GAT model. Even with the parallel acceleration of the Graphics Processing Unit (GPU), it is still difficult to

overcome the problem of insufficient computing power and increase the dimension of the hidden layer to the optimal value. On the one hand, this reflects the advantage of the graph attention network in efficiently integrating the correlation of vertex features into the model. On the other hand, it also indicates that there is still room for improvement and enhancement of methods based on the graph attention mechanism.

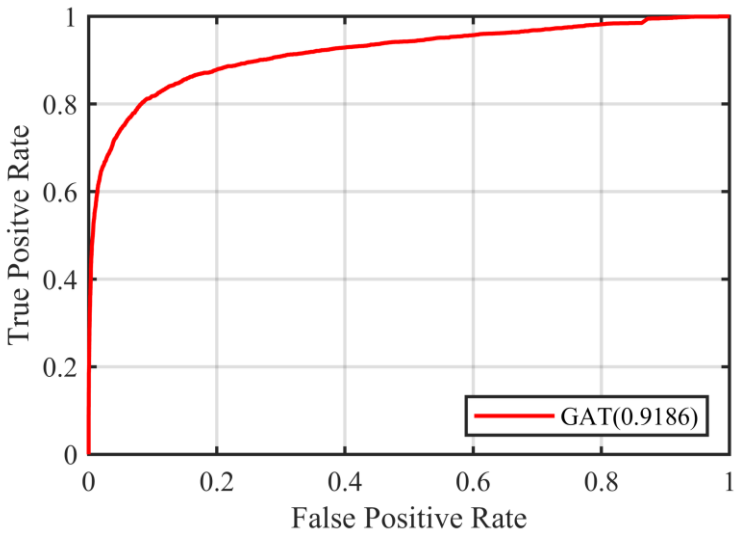


Fig. 4.9 ROC curve for GAT model

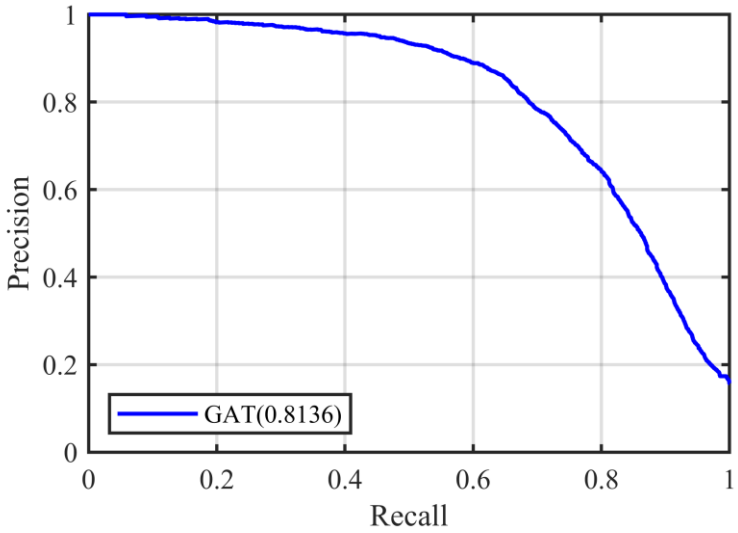


Fig. 4.10 PR curve for GAT model

Chapter 5. Summary and Outlook

Section 1. Summary

This study improved the LPIGAC algorithm, an end-to-end method for lncRNA-protein interaction prediction. GCN and GAT were used in autoencoders and collaborative training on both lncRNA and protein graphs were conducted. Model structure and hyperparameters were determined carefully to obtain the optimal performance.

This study determined the optimal hidden layer dimension, optimizer, corresponding learning rate, and the parameters of collaborative training. The model performance was significantly improved compared with the previous results - the AUROC value increased from 0.932 to 0.942 and the AUPR value increased from 0.806 to 0.86. This experimental result outperformed existing methods for lncRNA-protein association prediction, providing a universal method for link prediction tasks in other organisms. The experimental result verified that deep learning models can more effectively interpret the complex interactions between lncRNA and protein.

In addition, this work also improved the GAE model by using a graph attention network instead of the graph convolutional neural network in LPIGAC. It is shown that the graph attention network can effectively integrate the relevance of vertex features in the model.

Section 2. Analysis and Outlook

The graph convolutional neural network in GAE was replaced with a graph attention neural network, and better result was achieved after model optimization and parameter tuning, with a significantly improved AUPR value compared to the original LPIGAC. However, the overall effect was still slightly inferior to the improved LPIGAC. The main reason is that the GAT model has too many parameters to be trained. The insufficient computing capacity cannot be overcome even with the parallel acceleration of graphics processing units (GPUs), and therefore the hidden layer dimension cannot be increased to the optimal value. To address these issues and improve prediction accuracy, potential improving methods are proposed:

(1) Instead of using a single GAT layer, we can combine GCN and GAT, e.g., use two layers of GCN plus one layer of GAT in the encoder and decoder. This scheme will overcome the drawback that the results of GCN do not improve significantly with the number of layers increasing, and will avoid the high demand of computational capacity

when only GAT is used.

(2) We can also construct variational graph autoencoders (VGAE) for prediction[19]. VGAE avoids overfitting by using KL scatter and has the potential to further improve the prediction accuracy.

Deep learning, as a more advanced machine learning method, has shown remarkable data mining and analysis capabilities. With the rapid development of bioinformatics field and deep learning methods, deep learning will definitely play a greater role in the research of life science related fields.

Reference

[1] 杨峰, 易凡, 曹慧青, 梁子才, 杜权. 长链非编码 RNA 研究进展[J]. 遗传, 2014, 36(5): 456-468.

[2] 阎隆飞, 孙之荣. 蛋白质分子结构. 北京: 清华大学出版社, 1999, 2-5.

[3] Han Chunyong, Li Xuebiao, Fan Qian, Liu Guangshu, Yin Jian. CCAT1 promotes triple-negative breast cancer progression by suppressing miR-218/ZFX signaling[J]. Aging (Albany NY), 2019, 11(14).

[4] Mohammad Ali Faghihi, Farzaneh Modarresi, Ahmad M Khalil, Douglas E Wood, Barbara G Sahagan, Todd E Morgan, Caleb E Finch, Georges St. Laurent III, Paul J Kenny, Claes Wahlestedt. Expression of a noncoding RNA is elevated in Alzheimer's disease and drives rapid feed-forward regulation of beta-secretase[J]. Nature Medicine, 2008, 14(7): 723-730.

[5] Fabrizio Ferrè, Alessio Colantoni, Manuela Helmer-Citterich. Revealing protein–lncRNA interaction[J]. Briefings in Bioinformatics, 2016(1): 106-116.

[6] Bellucci, Matteo, Federico Agostini, Marianela Masin, and Gian Gaetano Tartaglia. Predicting protein associations with long noncoding RNAs. Nature Methods, 2011, 8(6): 444-445.

[7] Suresh V, Liu Liang, Adjeroh Donald, Zhou Xiaobo. RPI-Pred: predicting ncRNA-protein interaction using sequence and structural information[J]. Nucleic Acids Research, 2015, 43(3): 1370-1379.

[8] Hu Huan, Zhang Li, Ai Haixin, Zhang Hui, Fan Yetian, Zhao Qi, Liu Hongsheng. HLPI-ensemble: prediction of human lncRNA-protein interactions based on ensemble strategy[J]. RNA Biology, 2018, 15(6): 797-806.

[9] Li Ao, Ge Mengqu, Zhang Yao, Peng Chen, Wang Minghui. Predicting long noncoding RNA and protein interactions using heterogeneous network model[J]. BioMed Research International, 2015, 2015.

[10] Ge Mengqu, Li Ao, Wang Minghui. A bipartite network-based method for prediction of long non-coding RNA–protein interactions[J]. Genomics, Proteomics & Bioinformatics, 2016, 14(1): 62-71.

[11] Zhao Qi, Yu Haifan, Ming Zhong, Hu Huan, Ren Guofei, Liu Hongsheng. The bipartite network projection-recommended algorithm for predicting long non-coding RNA-protein interactions[J]. Molecular Therapy-Nucleic Acids, 2018, 13: 464-471.

[12] Xie Guobo, Wu Cuiming, Sun Yuping, Fan Zhiliang, Liu Jianghui. LPI-IBNRA: long non-coding rna-protein interaction prediction based on improved bipartite network recommender algorithm[J]. Frontiers in Genetics, 2019, 10: 343.

[13] Zhou Yuan Ke, Hu Jie, Shen Zi Ang, Zhang Wen Ya, Du Pu Feng. LPI-SKF: predicting lncRNA-protein interactions using similarity kernel fusions[J]. Frontiers in Genetics, 2020: 1554.

[14] 赵新元, 秦伟捷, 钱小红. 深度学习方法在生物质谱及蛋白质组学中的应用[J]. 生物化学与生物物理进展, 2018, 45(12): 1214-1223.

[15] 李洪顺, 于华, 宫秀军. 一种只利用序列信息预测 RNA 结合蛋白的深度学习模型[J]. 计算机研究与发展,2018,55(1):93-101.

[16] Wekesa Jael Sanyanda, Luan Yushi, Chen Ming, Meng Jun. A hybrid prediction method for

plant lncRNA-protein interaction[J]. Cells, 2019, 8(6): 521.
[17] Jin Chen, Shi Zhuangwei, Zhang Han. Predicting lncRNA-protein interactions based on graph autoencoders and collaborative training. IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2021: 38-43.
[18] 刘忠雨, 李彦霖, 周洋. 深入浅出图神经网络: GNN 原理解析. 北京: 机械工业出版社, 2019, 81-100.
[19] Ding Yulian, Tian Li Ping, Lei Xiujuan, Liao Bo, Wu Fang Xiang. Variational graph auto-encoders for miRNA-disease association prediction[J]. Methods, 2021, 192: 25-34.
[20] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, Jian Yu. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. *Proceedings of the World Wide Web Conference*, New York, NY, USA, 2020, 1082–1092.
[21] Kipf Thomas N, Welling Max. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations*, 2017.
[22] Veličković Petar, Casanova Arantxa, Liò Pietro, Cucurull Guillem, Romero Adriana, Bengio Yoshua. Graph attention networks[J]. Proceedings of the 6th International Conference on Learning Representations, 2018.
[23] 施荣华, 金鑫, 胡超. 基于图注意力网络的方面级别文本情感分析[J]. 计算机工程,2022,48(2):34-39.
[24] Zhuangwei Shi, Han Zhang, Chen Jin, Xiongwen Quan, Yanbin Yin. A representation learning model based on variational inference and graph autoencoder for predicting lncRNA-disease associations[J]. BMC Bioinformatics, 2021, 22(1).
[25] 陈卫中, 潘晓平, 宋兴勃, 倪宗瓒. ROC 曲线中最佳工作点的选择[J]. 中国卫生统计, 2006(2): 157-158.