(GAAN)Graph Adaptive Attention Network

Anonymous CVPR submission

Paper ID *****

Abstract

Non-Euclidean data, such as social networks, and citation relationships between documents, has node information and structural information. Graph Convolutional Network(GCN) can automatically learn node features and association information between nodes. For example, social networks are very suitable for using graph data to express, such as nodes in social networks and the relationship between nodes. Users (with ID information, etc.), posts are nodes, the relationship between user A and user B is attention, and the relationship between users and posts may be published or forwarded. Through such a graph, it is possible to analyze who and what users are interested in, and further generate the recommendation system. The core ideology of the graph convolutional network is to aggregate node information by using edge information, thereby generating a new node feather. Considering the numbers and different contributions of neighbor nodes to the central node, we design the Adaptive Attention Mechanism(AAM). To further enhance the representational capability of the model, we utilize Multi-Head Graph Convolution(MHGC). Based on AAM and MHGC, we contrive the novel Graph Adaptive Attention Network (GAAN). Experiments on the CORA dataset show that the classification accuracy has achieved 85.6%.

Keywords: Non-Euclidean, GCN, Adaptive Attention Mechanism, Multi-Head Graph Convolution

1. Introduction

Many data in real life have irregular spatial structures, known as non-Euclidean data, such as social networks, recommendation systems, citation relationships between documents, transportation planning, natural language processing, etc. This type of data has both node information and structural information, which traditional deep learning networks like CNN, RNN, Transformer, etc cannot well represent. Graph Convolutional Network (GCN) [1], shown in Figure 1, is a class of deep learning models used for processing graph data, and they have made significant progress in graph data in recent years. In the real world, many complex systems can be modeled as graph structures, such as social networks, recommendation systems, protein interaction networks, etc. The nodes and edges of these graph data represent entities and their relationships, which is of great significance for understanding information transmission, node classification, graph classification, and other tasks in graph structures. However, compared with traditional regularized data such as images and text, graph data processing is more complex. Traditional Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) cannot be directly applied to graph structures because the number of nodes and connections in the graph may be dynamic. Therefore, researchers have begun exploring new graph neural network models to effectively process graph data. Graph Convolutional Networks (GCN) were proposed in this context, making an important breakthrough in graph data. The main idea of GCN is to use the neighbor information of nodes to update their representations, similar to traditional convolution operations, but on graph structures. By weighted averaging of neighboring nodes, GCN achieves information transmission and node feature updates, allowing the model to better capture the local and global structures in the graph. More broadly, Graph Convolutional Networks (GCN) are a special case of Graph Neural Networks (GNN).

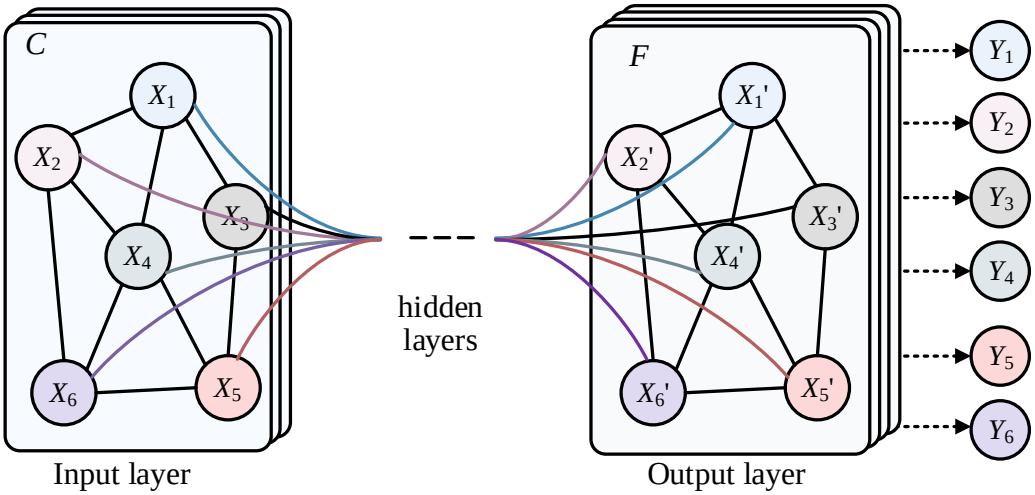

Figure 1: The structure of GCN

The previous graph convolution methods have not fully considered the number and importance of neighboring nodes. To solve the above problems, we propose the novel GAAN (Graph Adaptive Attention Network), and our main contributions are in the following two areas:

(1) To generate different weights for each neighbor node of the central node, we design the novel adaptive attention mechanism(AAM).

(2) Based on AAM, we utilize Multi-head Graph Convolution(MHGC) to model and represent features better.

2. Related Work

Graph Neural Networks (GNN) are a class of deep learning models used for processing graph data, which have made significant progress in graph data in recent years.

GCN[1] is one of the earliest proposed GNN models. GCN learns node representations by aggregating information from neighboring nodes and updating node features through weighted averaging of neighbors. It has achieved excellent performance in semi-supervised node classification tasks, making it a pioneering work in graph data. GraphSAGE[2] was proposed to address the scalability issue of GCN on large-scale graph data. GraphSAGE learns node representations by sampling neighboring nodes, enabling efficient node classification on large graphs. SGCN[3] proposes a simplified version of GCN by reducing the multi-layer convolution in GCN to a single-layer convolution, thereby reducing computational complexity. This simplified version achieves good performance on large-scale graph data. Due to the previous GNN models overlooking the importance of each neighboring node, Graph Attention Networks[4] adapt attention mechanisms to weigh neighboring nodes, allowing the model to adaptively learn the importance between nodes. This approach is of great significance in handling large graphs and node classification tasks. Graph Transformer Networks[5] propose a method for applying the Transformer architecture to graph-structured data. It introduces the concept of self-attention to graph neural networks by effectively processing the relationships between nodes and edges through dynamic attention mechanisms. The review paper[6] comprehensively introduces the development history of graph neural networks, different methods, and application areas. It summarizes and compares the development in the field of graph data analysis and the pros and cons of various models. It is an important literature for understanding the field of GNN.

GNN is an active research field that has emerged many important models and techniques. Its application scope and domains are increasingly vast, providing more innovation and progress for deep learning on graph data. From the perspective of adaptive node weights, we contrive the GAAN.

3. Methodology

The overall structure of GAAN is illustrated in Figure 2, and the specific computational process can be represented by Equation 1 to Equation 5.

$$h_{ij} = concat\left(\boldsymbol{W}\vec{h}_i, \boldsymbol{W}\vec{h}_j\right), \qquad i,j \in N \qquad (1)$$

where $\vec{h}_i$ and $\vec{h}_j$ represent the ith and jth nodes in the graph, respectively. $\boldsymbol{W}$ is the shared weight matrix to uniform the node features. h_{ij} fuses the features of the ith and jth nodes. Based on h_{ij}, we can calculate the basic AAM between the ith and jth nodes with the following Equation 2.

$$e_{ij} = \frac{a * h_{ij}}{\sqrt{D_i * D_j}} \quad (2)$$

where D_i and D_j represent the degree of *i*th and *j*th nodes, respectively. a is the weight vector to reshape h_{ij}. e_{ij} is the weight coefficient between the *i*th and *j*th nodes. Then, we adopt the softmax function to normalize e_{ij}, as shown specifically in Equation 3.

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(LeakyReLU(e_{ik}))}{\sum_{k\epsilon h_i} exp\,(LeakyReLU(e_{ik}))} \quad (3)$$

Based on the normalized weight coefficients between nodes obtained from Equations 1 to 3, we can perform the graph convolution layer computation, as shown in Equation 4.

$$\vec{h}_i' = \sigma(\sum_{j\epsilon\mathcal{N}_i} \alpha_{ij}\, \boldsymbol{W}\vec{h}_j) \quad (4)$$

Building on Equation 4, we utilize MHGC, which involves performing the computation of Equation 4 with MHGC and then averaging these results to obtain the node features of the subsequent layer. The computation process is shown in Equation 5.

$$\vec{h}_i' = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\epsilon\mathcal{N}_i} \alpha_{ij}^k\, \boldsymbol{W}^k\vec{h}_j) \quad (5)$$

Based on the aforementioned formula, we have completed the normalized weight coefficients and inter-layer computation processes for GAAN. We have designed a single hidden layer, thus constructing the GAAN with a total of two layers.

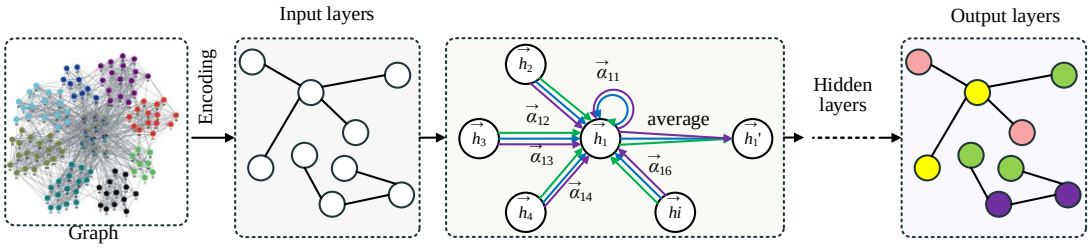

Figure 2: The structure of GAAN

4. Experiments

In this paper, we use the Cora dataset, the specific statistics of which are shown in Table 1. Cora includes a citation network of scientific publications, which we use to test the performance of GAAN.

Table 1 Summary of the datasets used in our experiments.

Dataset	Nodes	Edges	Features per node	Classes
Cora	2708	5429	1433	7

As shown in the ablation experiments in Table 2, AAM and MHGC respectively increased by 1% and 1.2%. Our GAAN has achieved an excellent performance of 85.6%.

As shown in Table 3, GCNA's accuracy is 1.9% higher than the previous GAT. Even the accuracy of GCNA without MHGC is 1.2% higher than GAT. As shown in Table 3, GCNA's accuracy is 1.9% higher than the previous GAT. Even the accuracy of GCNA without MHGC is 1.2% higher than GAT.

5. Conclusion and Future Work

We have proposed AAM and MHGC to construct GCNA, which solves the differences between neighbor nodes to the central node. Experimental results show that our method is superior in accuracy.

Over-smoothing occurs when multi-layers are stacked, leading to the features of all nodes being almost the same. However, many situations are necessary to capture the features of distant neighbors. Therefore, stacking multiple layers of GCN is inevitable. The strategy of stacking multi-layers, designed to prevent over-smoothing, is urgent.

Table 2 Ablation experimental results on Cora. 'n' and 'n_hidden' represent the number of graph convolution heads and the number of nodes in hidden layers, respectively.

	AAM	MHGC(n=8)	n_hidden			Accuracy(%)
			64	96	128	
Group1	×	×	√	×	×	**83.9**
	×	×	×	√	×	83.5
	×	×	×	×	√	83
Group2	×	√	√	×	×	84.6
	×	√	×	√	×	**85.1**
	×	√	×	×	√	84.6
Group3	√	×	√	×	×	**84.9**
	√	×	×	√	×	84.3
	√	×	×	×	√	84.6
Group4	√	√	√	×	×	84.6
	√	√	×	√	×	85.4
	√	√	×	×	√	**85.6**

Table 3 Compared experimental results

	Method	Accuracy(%)
Previous	MLP	55.1
	SemiEmb	59
	DeepWalk	67.2
	ICA	75.1
	Planetoid	75.7
	Chebyshev	81.2
	GCN	81.5
	MoNet	82.2
	GAT	83.7
GAAN (Ours)	no MHGC	84.9
	with MHGC	**85.6**

Reference

[1] Kipf, Thomas and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." ArXiv abs/1609.02907 (2016): n. pag.

[2] Hamilton, William L. et al. "Inductive Representation Learning on Large Graphs." Neural Information Processing Systems (2017).

[3] Wu, Felix et al. "Simplifying Graph Convolutional Networks." International Conference on Machine Learning (2019).

[4] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2017). Graph Attention Networks. ArXiv, abs/1710.10903.

[5] Yun, Seongjun & Jeong, Minbyul & Kim, Raehyun & Kang, Jaewoo & Kim, Hyunwoo. (2019). Graph Transformer Networks.

[6] Izadi, Mohammad Rasool, et al. "Optimization of Graph Neural Networks with Natural Gradient Descent." *2020 IEEE International Conference on Big Data (Big Data)* (2020): 171-179.