AOAM: Automatic Optimization of Adjacency Matrix for Graph Convolutional Network

Yuhang Zhang, Hongshuai Ren, Jiexia Ye, Xitong Gao, Yang Wang, Kejiang Ye, Chengzhong Xu

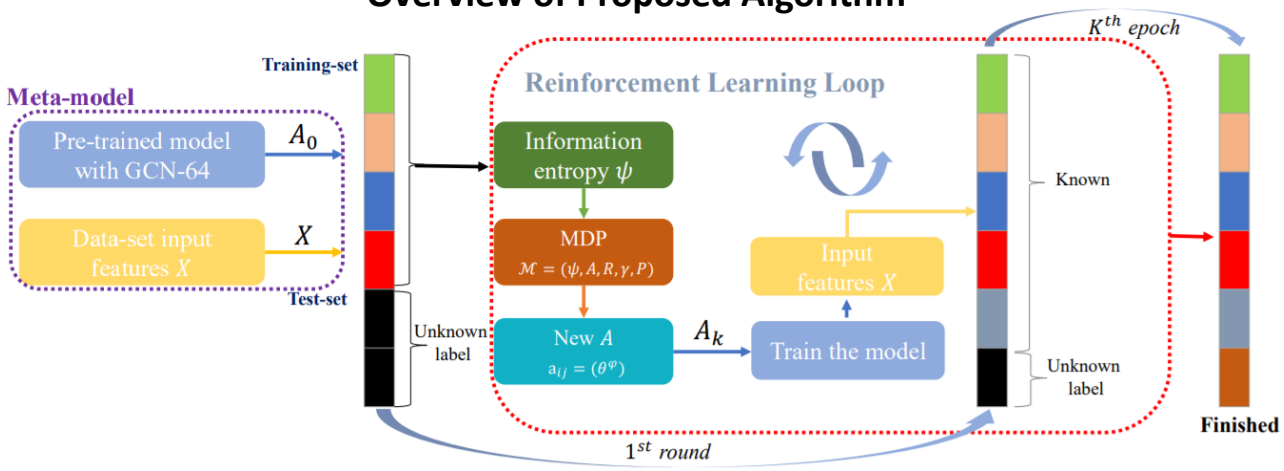
University of Chinese Academy of Sciences
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
University of Macau

1 Abstract

Graph Convolutional Network (GCN) is adopted to tackle the problem of convolution operation in non-Euclidean space. Previous works on GCN have made some progress, however, one of their limitations is that the design of Adjacency Matrix (AM) as GCN input requires domain knowledge and such process is cumbersome, tedious and error-prone. In addition, entries of a fixed Adjacency Matrix are generally designed as binary values (i.e., ones and zeros) which can not reflect the real relationship between nodes. Meanwhile, many applications require a weighted and dynamic Adjacency Matrix instead of an unweighted and fixed AM, and there are few works focusing on designing a more flexible Adjacency Matrix. To that end, we propose an end-to-end algorithm to improve the GCN performance by focusing on the Adjacency Matrix. We first provide a calculation method called *node information entropy* to update the matrix. Then, we perform the search strategy in a continuous space and introduce the Deep Deterministic Policy Gradient (DDPG) method to overcome the drawback of the discrete space search. Finally, we integrate the GCN and reinforcement learning into an end-to-end framework. Our method can automatically define the Adjacency Matrix without prior knowledge. At the same time, the proposed approach can deal with any size of the matrix and provide a better AM for network. Four popular datasets are selected to evaluate the capability of our algorithm. The method in this paper achieves the state-of-the-art performance on *Cora* and *Pubmed* datasets, with the accuracy of 84.6% and 81.6% respectively.

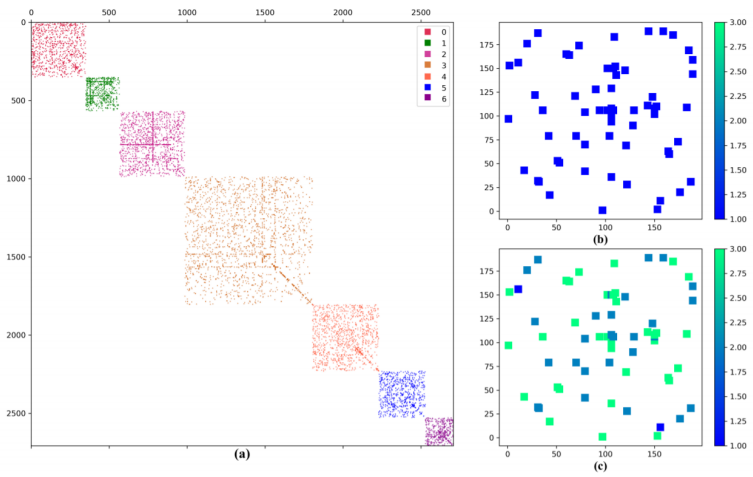
2 Method

Overview of Proposed Algorithm



Adjacency Matrix: discrete value to continuous value

Illustration of the optimized matrix. Fig. (a) is the sample distribution from Cora dataset. The blue rectangular block in Fig. (b) represents that all 200 nodes are initialized to 1. Fig. (c) is the optimized matrix by Section III-C. The horizontal and vertical coordinates represent nodes' positions, and different colors represent the new weights after network optimization.



Algorithm 1 Graph Convolutional Networks with Adjacency Modeling

Input: The nodes' features $\boldsymbol{X}$ with dimension in $N \times F$. Pre-trained GCN-64 model.
Output: Optimized matrix $\boldsymbol{A}$ and the classification results.

Calculate the entropy ψ by the pre-trained model.
Initialize the adjacency matrix $\boldsymbol{A}$.
for epoch of RL **do**
  Initialize a random process N for action exploration.
  $t = 1$
  while $\epsilon \ll \epsilon'$ (Designed error threshold) **do**
    Select action $\alpha_t = \mu(s_t|\theta^\mu) + N_t$ according to the current policy and exploration noise N_t;
    Calculate the information entropy ψ by the input meta-model;
    Use the equation (8) to update entropy ψ;
    Do $a_{ij} = \alpha_t - \psi$,
    Update adjacency matrix A by $a_{ij} = Max(a_{ij}, a_{ji})$;
    Calculate Acc_{cur} with X, A;
    $t = t + 1$
  end while
  $R = Acc_{cur} - Acc_{meta-model}$
  Update agent;
end for
return Best adjacency matrix A.

Our goal is to exploit the modeling method to optimize the value of the adjacency matrix in GCN. We first propose a method of adjacency modeling which can find a better adjacency matrix and it does improve the performance of GCN.

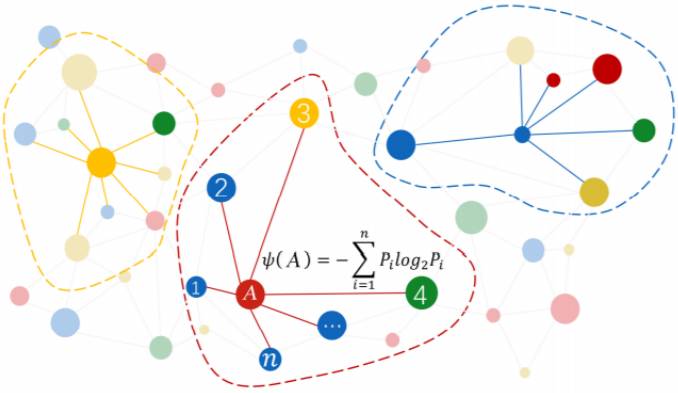
Considering the value of an adjacency matrix should not be influenced by human knowledge which is erroneous, this paper employs reinforcement learning to find the matrix value. We proposes a node entropy to complete the updating of each matrix during the reinforcement learning loop. The DDPG is used to produce a series of continuous value to fill in the new produced matrix. Finally, our method achieves the automatic calculation.

$$\begin{cases} a_{ij} = \alpha_t - \psi \\ \psi^l(n) = \sum_{m \in adj(l)} -p_m^{(l)} \log_2 p_m^{(l)}, \end{cases} \tag{10}$$

a_{ij} and a_{ji} are the same value in the adjacency matrix, how can we choose the final value to fill in the matrix? We adopt formula (11) to decide which one can be accepted in the end. This formula means that the weight between two nodes is determined by the one with the larger value.

$$a_{ij} = Max(a_{ij}, a_{ji}). \tag{11}$$

the node information entropy between nodes n



3 Experiments Results

TABLE II: Results of transductive learning.

	Cora	Citeseer	Pubmed
ICA [25]	75.10%	69.10%	73.90%
LP [26]	68.00%	45.30%	63.00%
ManiReg [27]	59.50%	60.10%	70.70%
SemiEmb [28]	59.00%	59.60%	71.70%
DeepWalk [29]	67.20%	43.20%	65.30%
Planetoid [24]	75.70%	64.70%	77.20%
Chebyshev [11]	81.20%	69.80%	74.40%
MoNet [30]	81.70%	69.90%	78.80%
GAT [2]	83.30%	72.50%	79.10%
Auto-GNN [5]	83.60%	**73.80%**	79.70%
TOGCN [21]	83.10%	72.70%	79.50%
GCN-64	81.50 %	69.05 %	79.30 %
Ours-AOAM	**84.60%**	72.10%	**81.60%**

In this section, we conduct experiments to compare our AOAM and other state-of-the-art approaches. This paper selects four benchmark datasets, which are employed on transductive learning and inductive learning tasks. We achieve or approach to the state-of-the-art performance. In the following descriptions, section IV in our paper shows the training setup and some detailed information about this method. We review the experiments and make discussions at the end.

4 Conclusion

1. In this paper, we develop an end-to-end optimal algorithm to define the input Adjacent Matrix without human intervention.
2. We find the relationship between entropy value and matrix. To that end, we are the first to propose a calculation approach (node information entropy) to update the matrix during the training process.
3. Our method achieves state-of-the-art performance on two popular datasets: Cora and Pubmed.