Graph Transformer Structural Encoding Based on MCL Algorithm

Shuai Wang
South-Central Minzu University

Abstract

In this report,we propose a general structural encoding method for graph Transformer.And experimented on the OGB [1] graph classification task ogbg-ppa.

1 Introduction

As Transformer [2] has shown significant effect in NLP field, Transformer has been gradually extended to different fields. Recently, many researches have attempted to extend the Transformer-like model to the task of Non-Euclidean structure such as graph neural network. Many papers(e.g. Wu2021GraphTrans [3]) have proved that the global long-range attention mechanism is very important for graph structure tasks. Position encoding is an important part of Transformer model in different fields. However, position encoding or relative position encoding is not applicable in the Non-Euclidean structure such as graph. Therefore, we aim to propose a general structural encoding method for graph Transformer.

Inspired by the encoding of Swin Transformer [4] and Graphormer [5] , we propose a encoding method based on MCL(Markov Cluster) algorithm [6,7] (The implementation details are described in the next section), a unsupervised cluster algorithm for graphs based on simulation of (stochastic) flow in graphs. The algorithm was originally developed for clustering different clusters of graphs. Here we do not need to consider its clustering function, but only need to calculate the probability matrix, and map the matrix as a bias to add to the attention matrix. In the probability matrix, the elements in row i and column j represent the probability of arriving at node j after n steps from node i. That sounds like random-walk SE [8]. In contrast, our method can change the granularity or resolution of the clustering outcome by adjusting the hyperparameters to provide different scale information. Due to the parallel operation of matrix, our encoding method is more efficient than SPD(the distance of the shortest path) [5]. In addition, the encoding method can use the weight information of edges for weighting calculation.

We built a simple Transformer-like model for graph classification task and tested it on ogbg-ppa dataset. The experimental results show that using the above encoding method is significantly better than not using it.

2 Methods

2.1 MCL Structural Encoding

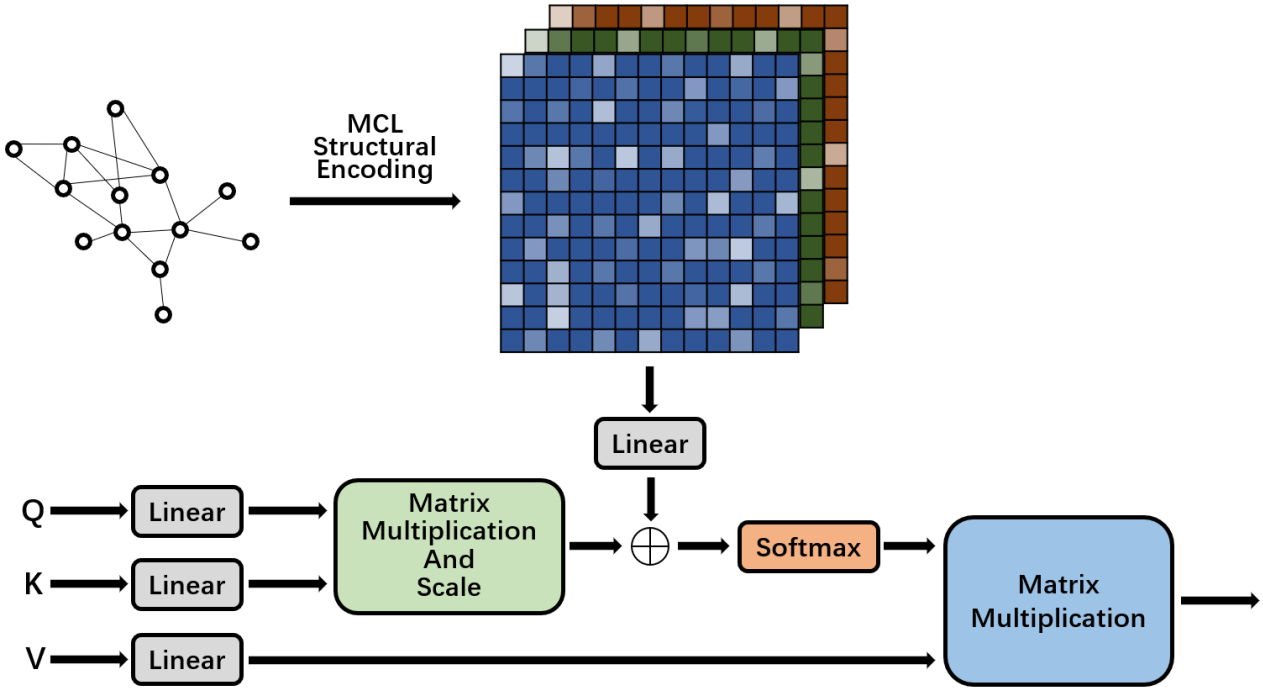

Figure 1: Illustration of MCL structural encoding.

The basic principle of MCL algorithm firstly normalizes the adjacency matrix(also indicates the initial state) to the transition matrix, then Expansion operation is performed on the transition matrix until the steady state, and finally interpretation of the states as clusterings. We only focus on getting the final state matrix (probability matrix), so simplify some steps. The core of the algorithm is expansion and inflation operation. The expansion operation is to multiply the probability matrix representing the current state with the transition matrix. This represents the new state obtained after a random walk in the graph structure. After each expansion operation need to normalization of state, as shown in **Eq.(1)**.

$$\begin{aligned} P &= NORM(A) \\ P_{ij} &= \frac{A_{ij}}{\sum_{x=0}^{N} A_{ix}} \end{aligned} \tag{1}$$

Where P_{ij} as as the $(i,\ j)$-element of the normalized state matrix, A represents the adjacency matrix or the state matrix of the previous step. N represents the number of matrix columns.

Inflation refers to the power operation of each element in the matrix. The purpose of this is to make the connection of strong neighbors stronger and the connection of weak neighbors weaker.

$$\begin{aligned} P &= INFLA(P) \\ P'_{ij} &= P^r_{ij} \end{aligned} \tag{2}$$

The complete algorithm steps are shown in **Algorithm 1** below.

Algorithm 1: Computing random walk probability matrix based on MCL algorithm

Require: adjacency matrix A, inflation parameter i, expansion intervals n;
$NORM$and $INFLA$ functions in **Eq.(1,2).**
$P_0 \leftarrow NORM(A)$
$P \leftarrow P_0$
for $k = 1\ to\ m$ **do**
    $P \leftarrow P \cdot P_0$
    if $(k\%n) == 0$ **then**
        $P \leftarrow INFLA(P)$
    end
    $P \leftarrow NORM(P)$
end
Result: probability matrix P

Finally, we map the obtained matrix linearly, convert it into a bias and add it to the attention matrix. See Figure 1 for an illustration. Denote A as the attention matrix, we have:

$$A = \frac{QK^\top}{\sqrt{d_k}} + Bias \tag{3}$$

$$Bias_{(i,j)} = Linear(P_{ij}) \tag{4}$$

Where Q, K represents the query and key projected from the previous output, $Bias$ is obtained by linear mapping each element in the probability matrix P.

2.2 Implementation Details

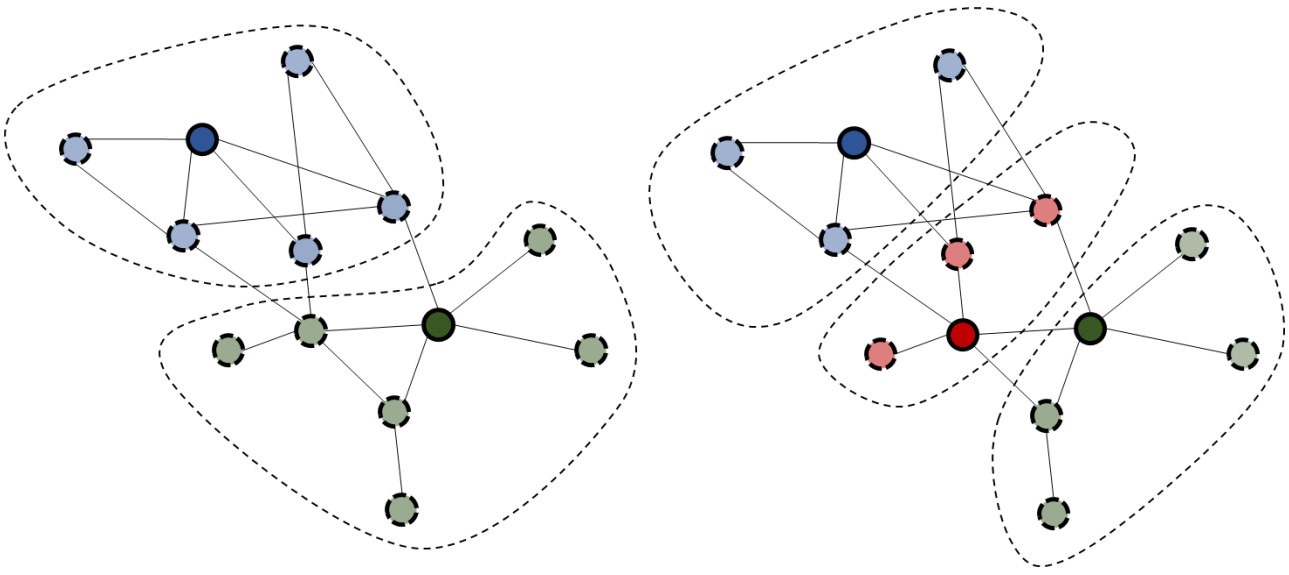

Figure 2: Inflation parameter affect the granularity of the results, resulting in a multi-scale effect.

For the next experiment, we build a lightweight Transformer-like model for graph neural network. Considering that complete Transformer without pre-training is easy to produce over-fitting on small and medium-sized datasets, we adopted a mixed approach of GCN[9] and Transformer encoder. In addition, some studies[3] have also proved that graph convolution implicitly represents the relevant information between nodes to some extent. Therefore, each layer contains a GCN module and a Transformer encoder module, the output of the GCN module as the input of the Transformer encoder. We call this architecture GC-T. Due to the addition of self-attention layer, the data needs to be aligned for the matrix parallel operation. Before inputting the model, we fill up all the nodes of the graph

structure to the maximum number of nodes, and fill the insufficient data with zero at the end. Further all BN layers in the model were replaced with LN layers, and bias for all linear layers was set to False. Post norm is used in the Transformer encoder. We characterize the module as below:

$$\begin{aligned} h''^{(l)} &= LN\left(GCN\left(h^{(l-1)}\right)\right) + h^{(l-1)} \\ h'^{(l)} &= LN\left(MSA\left(h''^{(l)}\right) + h''^{(l)}\right) \\ h^{(l)} &= LN\left(FFN\left(h'^{(l)}\right) + h'^{(l)}\right) \end{aligned} \tag{5}$$

There are many ways to embed MCL structural encoding. In addition to inputting the standard adjacency matrix, also can also input the weighted adjacency matrix. We can get different granularity probability matrix by changing Inflation parameter, so we can concatenate different granularity probability matrix to form multi-dimension structure information(As shown in Figure 2). The different embedding methods are compared in the following experimental section.

3 Experiments

In ogbg-ppa task, we trained the model proposed above. Because of the high cost of training a complete model, we extracted some data for preliminary parameter adjustment. We sampled one-tenth of the training set, one-tenth of the test set, and one-tenth of the validation set from the full dataset for preliminary comparison. The training strategy adopts warm-up and AdamW optimizer. The settings for the hyperparameters are shown in Table 1.

Table 1: Hyper-parameter of GC-T on ogbg-ppa sampled dataset

	GC-T
Layers	3
Hidden Dimension	256
FFN Dimension	256*2
Attention Heads	16
Peak Learning Rate	3e-4
Weight Decay	1e-5
Input Dropout	0.2
Attention Dropout	0.2
Warm-up Steps	4000
Pooling	mean
Epochs	110
Adam (β1, β2)	(0.9, 0.999)

We compare different embedding encoding methods, as shown in Table 2.

Table 2: Comparison results of different encodings

Methods	Train ACC	Validation ACC	Test ACC	Time Cost
GC-T	0.99812±0.00064	0.56856±0.00966	0.58090±0.01076	**1h59min**
GC-T+MCL6.0	0.99905±0.00034	**0.57661±0.00428**	**0.59496±0.00736**	2h7min
GC-T+MCL6.0(weighted)	**0.99916±0.00031**	0.57633±0.00614	0.59306±0.00406	2h53min
GC-T+MCL[3.0-6.0-9.0]	**0.99916±0.00035**	0.57173±0.00733	0.59113±0.00955	2h29min

The comparison methods in Table 2 are: GC-T without MCL encoding, GC-T with MCL encoding (inflation parameter is 6), MCL encoding method weighted on adjacency matrix (inflation parameter is 6), and multi-scale MCL encoding concatenating method (inflation parameter is 3, 6, 9). Through comparison, it is obvious that the effect of structural coding method based on MCL algorithm is improved. Among them, the second unweighted MCL encoding has the best effect, sacrificing less extra time cost and achieving the same performance improvement. We used this encoding method in the full dataset experiment. In addition, in order to alleviate overfitting phenomena and improve the generalization of the model, we also use a variety of Dropout techniques (including query key value Drop, LayerDrop) mentioned in **UniDrop**[10] in the full dataset training. Finally, our model is trained on the full ogbg-ppa dataset. The results and parameters are shown as follow.

Table 3: Hyper-parameter of GC-T on ogbg-ppa

	GC-T
Layers	3
Hidden Dimension	384
FFN Dimension	384*2
Attention Heads	16
MCL Inflation	6
Peak Learning Rate	1e-4
Weight Decay	1e-5
Input Dropout	0.2
Attention Dropout	0.5
Warm-up Steps	5000
Pooling	mean
Epochs	150
Adam (β1, β2)	(0.9, 0.999)
QKV Drop Rate	0.5
Drop Layer Rate	0.5

Table 4: Final results on ogbg-ppa

Methods	Train ACC	Validation ACC	Test ACC
GC-T+MCL(6.0)	0.999719±0.000318	0.698900±0.003674	0.743190±0.003319

References:

[1] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[2] A. Vaswani *et al.*, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun. 2017.

[3] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E.
Gonzalez, and I. Stoica, "Representing long-range context for graph neural networks with global attention," 2021.

[4] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *arXiv e-prints*, p. arXiv:2103.14030, Mar. 2021.

[5] C. Ying *et al.*, "Do Transformers Really Perform Bad for Graph Representation?" *arXiv e-prints*, p. arXiv:2106.05234, Jun. 2021.

[6] C. Stichting, M. Centrum, and S. V. Dongen, "A new cluster algorithm for graphs," *CWI (Centre for Mathematics and Computer Science)*, 1999.

[7] A. J. Enright, S. V. Dongen, and C. A. Ouzounis, "An efficient algorithm for large-scale detection of protein families," *Nucleic Acids Research*, vol. 30, no. 7, pp. 1575–1584, 2002.

[8] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph Neural Networks with Learnable Structural and Positional Representations," *arXiv e-prints*, p. arXiv:2110.07875, Oct. 2021.

[9] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv e-prints*, p. arXiv:1609.02907, Sep. 2016.

[10] Z. Wu *et al.*, "UniDrop: A Simple yet Effective Technique to Improve Transformer without Extra Cost," *arXiv e-prints*, p. arXiv:2104.04946, Apr. 2021.