FA-GAT: First-order Appoximation of Graph Attention Networks

Qingsong Lv
lqs19@mails.tsinghua.edu.cn

Abstract

Graph Attention Networks (GATs) (Veličković et al. 2017) utilizes pair-wise attention mechanism to aggregate information from neighbors, which is able to distinguish among different neighbors with different importance. However, pair-wise attention mechanism leads to quadratic time and space complexity. In this work, we propose FA-GAT, which is the first-order approximation of GAT. Experiments show that FA-GAT is more time and memory efficient, and keeps comparable performance to the original GAT. Moreover, under the fully-connected scenario where each pair of nodes can propogate information, FA-GAT even runs in linear complexity while GAT runs in quadratic complexity.

Related Work

This work is motivated by recent linear Transformer (Li et al. 2020), which reduces Transformer running time to linear complexity. Different from their layer normalization method, we use non-linear activation to guarantee the approximation precision.

Another recent work trying to accelerate GAT is Fast-GAT (Srinivasa et al. 2020). However, this work uses original GAT after edge pruning. Different from them, our motivation is to accelerate GAT model itself.

Method

In this section, we will clarify how we modify GAT into FA-GAT.

Suppose $\mathbf{H}^{(l)}$ is the row-stacked node representations of the l-th layer. $\mathbf{h}_i^{(l)}$ is the embedding of the node indexed by i of the l-th layer, which is also the i-th row of $\mathbf{H}^{(l)}$. The formulas of GAT are as follows (single head attention for simplicity):

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\mathbf{h}_j^{(l)}\right) \tag{1}$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i^{(l)}\|\mathbf{W}\mathbf{h}_j^{(l)}]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i^{(l)}\|\mathbf{W}\mathbf{h}_k^{(l)}]\right)\right)} \tag{2}$$

where $\mathbf{a}$ and $\mathbf{W}$ are trainable parameters, $\cdot^T$ represents transposition and $\|$ is the concatenation operation.

As we notice there are two neighboring non-linearity functions (softmax and leaky relu). we assume that leaky relu is redundant. Moreover, since $\mathbf{a}$ is trainable parameter, we can split it into two parts, so that the multiplication after concatenation can be written as addition of multiplication. Now, our modified α_{ij} can be written as:

$$\alpha_{ij} = \frac{\exp\left(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)} + \mathbf{a}_2^T\mathbf{W}\mathbf{h}_j^{(l)}\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)} + \mathbf{a}_2^T\mathbf{W}\mathbf{h}_k^{(l)}\right)} \tag{3}$$

Based on the first-order approximation of Taylor expansion $e^x \approx 1 + x$, equation 3 can be approxmated as:

$$\alpha_{ij} = \frac{1 + \mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)} + \mathbf{a}_2^T\mathbf{W}\mathbf{h}_j^{(l)}}{\sum_{k\in\mathcal{N}_i}\left(1 + \mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)} + \mathbf{a}_2^T\mathbf{W}\mathbf{h}_k^{(l)}\right)} \tag{4}$$

To guarantee the non-negativity, we introduce elu function:

$$\text{ELU}_\epsilon(x) = \begin{cases} x, & x \geq 0 \\ \epsilon(\exp(x) - 1), & x < 0 \end{cases} \tag{5}$$

where $\epsilon > 0$ is a hyper-parameter. $\text{ELU}_\epsilon(x)$ has the property that if $x \geq 0$, $\text{ELU}_\epsilon(x) = x > -\epsilon$, otherwise, $\text{ELU}_\epsilon(x) > -\epsilon$. Therefore, this function is suitable for bounding the minimum value while not drfting too much. We replace $\mathbf{a}^T\mathbf{W}\mathbf{h}$ with $\text{ELU}_\epsilon(\mathbf{a}^T\mathbf{W}\mathbf{h})$, so α_{ij} is modified as:

$$\alpha_{ij} = \frac{1 + \text{ELU}_\epsilon(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)}) + \text{ELU}_\epsilon(\mathbf{a}_2^T\mathbf{W}\mathbf{h}_j^{(l)})}{\sum_{k\in\mathcal{N}_i}\left(1 + \text{ELU}_\epsilon(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)}) + \text{ELU}_\epsilon(\mathbf{a}_2^T\mathbf{W}\mathbf{h}_k^{(l)})\right)} \tag{6}$$

Obviously, $\mathrm{ELU}_\epsilon(x) > -0.5$ when $0 < \epsilon \leq 0.5$. Consequently, $1 + \mathrm{ELU}_\epsilon(x) + \mathrm{ELU}_\epsilon(y) > 0$. Therefore, we can choose an $0 < \epsilon \leq 0.5$ to guarantee the non-negativity of approximation.

Now, based on equation 6, equation 1 can be written as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\frac{\sum_{j\in\mathcal{N}_i}\left(1 + \mathrm{ELU}_\epsilon(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)}) + \mathrm{ELU}_\epsilon(\mathbf{a}_2^T\mathbf{W}\mathbf{h}_j^{(l)})\right)\mathbf{W}\mathbf{h}_j^{(l)}}{\sum_{k\in\mathcal{N}_i}\left(1 + \mathrm{ELU}_\epsilon(\mathbf{a}_1^T\mathbf{W}\mathbf{h}_i^{(l)}) + \mathrm{ELU}_\epsilon(\mathbf{a}_2^T\mathbf{W}\mathbf{h}_k^{(l)})\right)} \right) \tag{7}$$

The above equation can be written with vectorized form. For brevity, we use $\mathbf{F}^{(l)}$ to denote $\mathbf{W}\mathbf{H}^{(l)}$, E_ϵ to denote ELU_ϵ

$$\mathbf{H}^{(l+1)} = \frac{(1 + \mathrm{E}_\epsilon(\mathbf{a}_1^T\mathbf{F}^{(l)})) * \mathbf{A}\mathbf{F}^{(l)} + \mathbf{A}(\mathrm{E}_\epsilon(\mathbf{a}_2^T\mathbf{F}^{(l)}) * \mathbf{F}^{(l)})}{(1 + \mathrm{E}_\epsilon(\mathbf{a}_1^T\mathbf{F}^{(l)})) * \mathbf{N} + \mathbf{A}\mathrm{E}_\epsilon(\mathbf{a}_2^T\mathbf{F}^{(l)})} \tag{8}$$

where $+$ and $*$ are element-wise addition and multiplication with standard broadcasting mechanism, and the omitted multiplication operations are matrix multiplications by default. $\mathbf{A}$ is the adjacency matrix of graph, and $\mathbf{N}$ is a column vector of node degree where $\mathbf{n}_i = |\mathcal{N}_i|$.

For fully-connected scenario, which means each pair of nodes may pay attention to each other, the above equation can be written as:

$$\mathbf{H}^{(l+1)} = \frac{(1 + \mathrm{E}_\epsilon(\mathbf{a}_1^T\mathbf{F}^{(l)})) * \sum \mathbf{F}^{(l)} + \mathrm{E}_\epsilon(\mathbf{a}_2^T\mathbf{F}^{(l)})^T\mathbf{F}^{(l)}}{(1 + \mathrm{E}_\epsilon(\mathbf{a}_1^T\mathbf{F}^{(l)}))|\mathcal{N}| + \sum \mathrm{E}_\epsilon(\mathbf{a}_2^T\mathbf{F}^{(l)})} \tag{9}$$

It is worth noting that equation 9 is $O(|\mathcal{N}|)$ time and memory complexity, while the original GAT under this condition is $O(|\mathcal{N}|^2)$.

Experiment

Table 1: Preliminary Results on Node Classification

Method	**Cora**
GCN (Kipf and Welling 2016)	81.4± 0.5%
GAT (Veličković et al. 2017)	83.0 ± 0.7%
FA-GAT (ours)	83.3 ± 0.8%

References

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

Li, R.; Su, J.; Duan, C.; and Zheng, S. 2020. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv preprint arXiv:2007.14902.*

Srinivasa, R. S.; Xiao, C.; Glass, L.; Romberg, J.; and Sun, J. 2020. Fast graph attention networks using effective resistance based graph sparsification. *arXiv preprint arXiv:2006.08796.*

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903.*