Graph Attention Networks

Junwei Deng

ATTENTION: This is a brief summary report, still I may involved some of my comments, which means it is not a classical summary report.

Abstract

In this paper, authors present their **G**raph **A**ttention **N**etworks, which develop a attentional layer which take consideration of one node's neighbors' features with different weight to get the layer output. The aim is to solve some problem cause by spectral-based GNN(such as GCN), including the efficiency, task generalization and so on, and of course improve the performance. The network is evaluated by transductive and inductive learning based on Cora, Citeseer, Pubmed and PPI dataset.

Key Word: Graph Attention Networks, GNN, GCN, Attention Mechanism, spectral-based

Definition

Spectral-based GNN:

For a undirected graph, we have

$$L = I_n - D^{-0.5}AD^{-0.5} = UAU'$$

And what we do in the graph convolution is actually

$$x * g_\theta = Ug_\theta U'x$$

Where g_\theta is the filter and x is the features of each node's concatenation.

Inductive Learning vs Transductive Learning

Learn the pattern and then inference is Inductive, learn some specific point and then use it to decide specific point is transductive.

Introduction & Contribution

In this part we know that the shortcomming this paper want to solve**(And it is the most important contribution in this paper)**

- Efficiency, the whole process of attention can be parallizable!
- Neighbors' weights to the graph nodes can be arbitrary, which provide more model capacity.
- The model can be directly used to solve the problem whose test-case has the completely unseen and unknown graph structures, while GCN can not.

Attention Layer Structure

In this section, we are talking about one attention layer through formula and image.

$$\mathrm{Input}: \mathrm{h} = \{\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_\mathrm{N}\}, \mathrm{h}_\mathrm{i} \in \mathrm{R}^\mathrm{F}$$
$$\mathrm{Output}: \mathrm{h}' = \{\mathrm{h}'_1, \mathrm{h}'_2, \ldots, \mathrm{h}'_\mathrm{N}\} \in \mathrm{R}^{\mathrm{F}'}$$

To achieve this, we need a weight matrix, or so to call the coefficient matrix in CNN and we can have the attention coefficients as following:

$$e_{i,j} = \alpha(Wh_i, Wh_j)$$

Of course we can use this attention coefficient to linearly combination and get the new output, still we find that if we just use e_i,j, **the structural information will be droped!** Because we don't include any connection information of the graph into the learning, we can recall the formula in GCN, which is

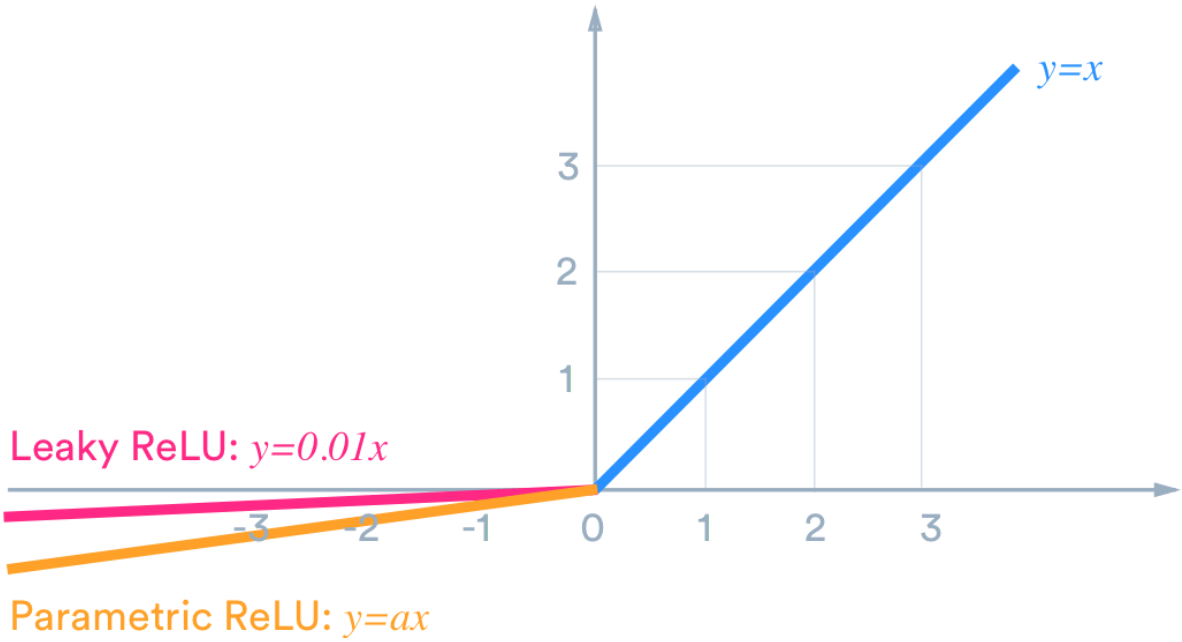
$$f(X, A), \text{where A is the adjacency matrix}$$

Then here is the most talented part, that is we can use a softmax function and limit that only the neighboring nodes can affect the node's attention coefficients.

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})}$$

where N_i is the neighborhood of node i.

Note: LeakyReLU



take two formula together we can have

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}h_i \| \mathbf{W}h_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\mathbf{a}^T \left[\mathbf{W}h_i \| \mathbf{W}h_k\right]))}$$

Where the

$$a^T \left[\mathbf{W}h_i \| \mathbf{W}h_j\right]$$

is the e_ij part and we have

$$\alpha \in R^{2F'}$$

to make sure the shape is correct.

After linearly combination, then we can use an **activate function \sigma** to make sure nonlinerity:

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}h_j\right)$$

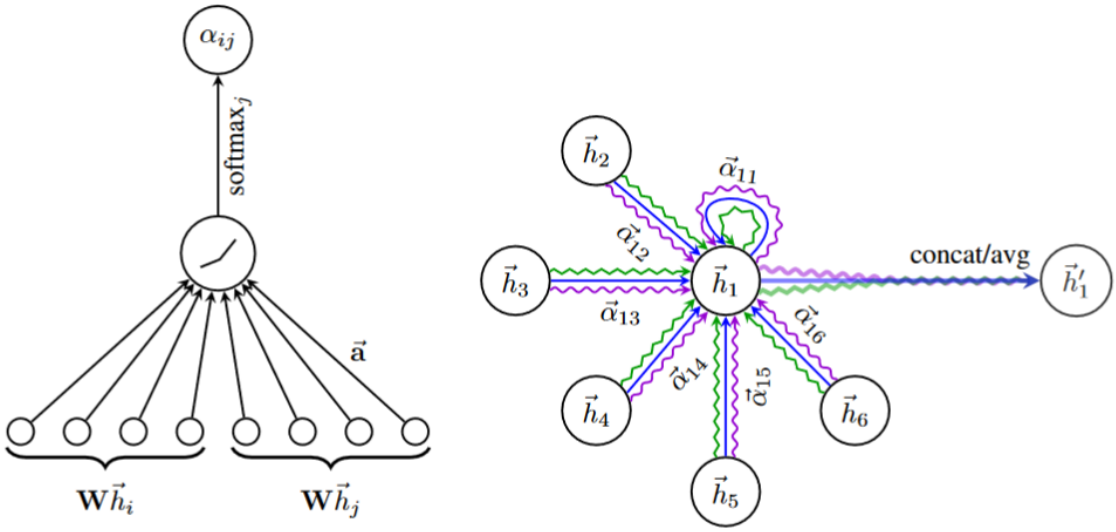
The the paper include the multi-head attention, which basically means that we can use more than one attention coefficient for one pair of nodes. And k means the kth attention coefficient and its weight matrix.

$$h_i' = ||_{k=1}^{K} \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k h_j \right)$$

Of course to make sure we have only correct output dim, the author use averaging for the final layer.

$$h_i' = \sigma \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k h_j \right)$$

The whole process is much more clear when we see the flowchart. The left is how to get attention coefficient and the right one is illustrate their combination and multiple head attention(k=3 in this case)



Contribution

- Efficient! The computation is not dependable and most of them is element add or costless operation, the can be parallelized.
- **(I don't think this one convince me)** We include different weight for nodes of a same neighborhood, so the capacity of the model is increase. (Okey, but why it worse, the conclusion is too "high-level")
- The model can be easily generalize to some unknown structure graph. It doesn't need a undirected graph, because no Laplace Matrix is calculated.
- Can access all neighbors compared to Hamilton 2017
- [Need more read] It is a special case for MoNet

Evaluation

Dataset

https://relational.fit.cvut.cz/dataset/CORA

http://csxstatic.ist.psu.edu/downloads/data

https://www.ncbi.nlm.nih.gov/pubmed/

https://snap.stanford.edu/biodata/datasets/10028/10028-PP-Miner.html

hyper-parameters are not interesting, I will not list them here, only some results, **actually you may find the performance is not the most significant contribution in this paper**.

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Contribution

The paper involve attention mechanism to GNN and improve the shrotcoming of the previous GNN, including:

- Efficiency
- Model capacity.
- Model generalization
- Model preformance

Which makes it an classical and outstanding one.