DMAIS Lab Seminar
<Graph Attention Networks>

2024-07-09

presenter : Moon Soo Ho

Data Mining And Intelligence Systems (DMAIS) Lab
School of Computer Science and Engineering
Chung-Ang University

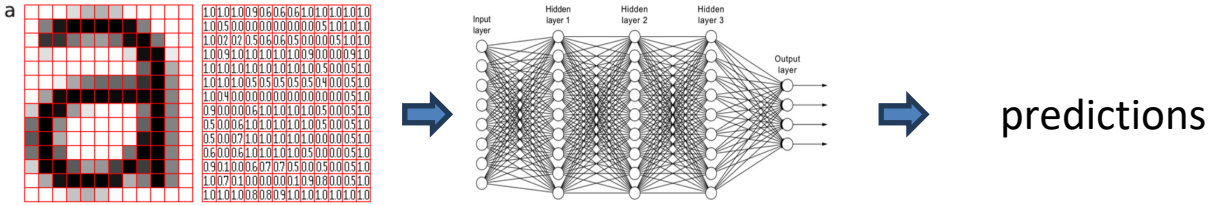
INDEX

- ☐ Main interest
- ☐ Why attention?
- ☐ Architecture
- ☐ Performance
- ☐ Conclusion

Main interest

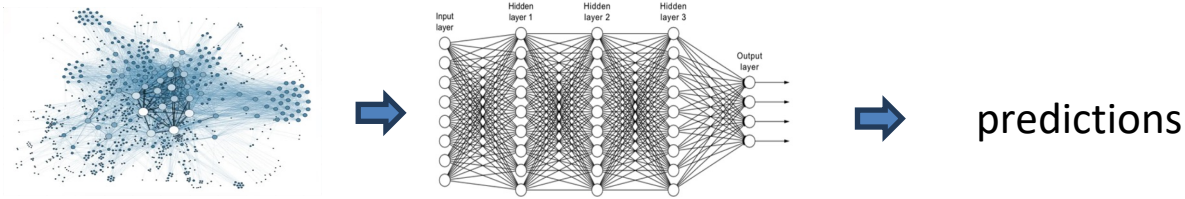
Grid sturcutre

- Feeding grid like structure to DNN was highly succesful(CNN, RNN etc.).



Graph sturcutre

- Graph structures doesn’t have self-explanatory relations with other nodes compared to grid structures.
- **How are we going to feed graph to DNN?**

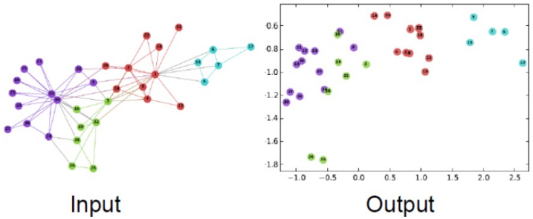


Main interest

What we usually want to achieve with (graph + DNN)

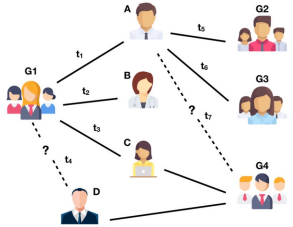
- **Node embedding(Most valuable interest)**

  Representing a high dimensional node to low dimension node.
  Transforming to low dimension benefits comparing nodes and visualization.



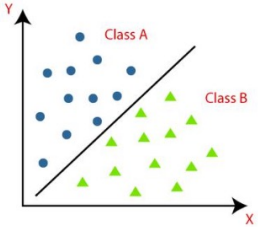
- **Link prediction**

  When a newly observed node is detected, the model can predict which node it's most likley to have edges with.



- **Classification**

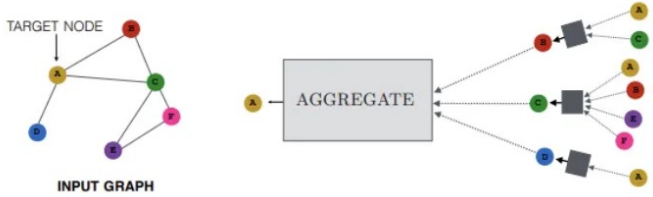
  Classifies nodes to given labels.



Why attention?

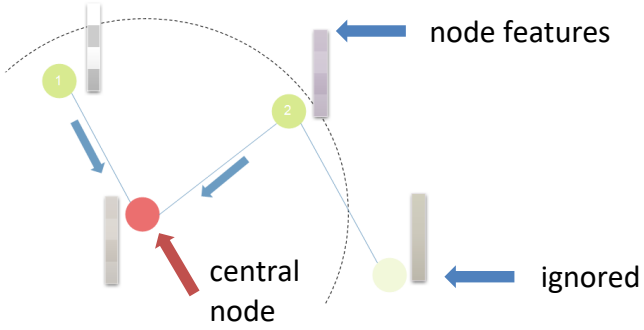
What we inherit from previous models

- Previous works such as **GNN, GCN, GraphSAGE** collected information for embedding via neighborhood nodes.



- By GCN, it's prooven that observing neighbors from 1-hop distance(first order neighbors) is most accurate.

- Also all the nodes that are qulified for aggregation has equal contribution to the central node embedding.

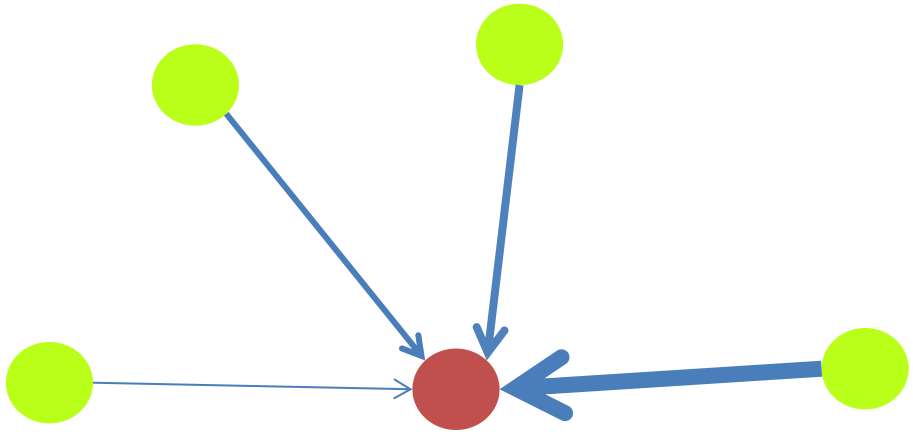


Why attention?

Different importance in neighborhood nodes.

- There is no evidence that all the 1-hop distance neighborhood nodes have equal impact to central node.
- Some messages are relatively stronger or weaker!

If the importance of neighbors to the red central node is higher in a clock-wise order...



Architecture

Our objective

- Embedd nodes in the graph(**h**) to other dimension(F -> F'). The new embedded node is denoted with '.

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F \qquad \mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

- We embed a single node by multiplying it with a parameterized weight matrix(**W**).

$$\mathbf{W} \in \mathbb{R}^{F' \times F}$$

Connection of two nodes create attention coefficient

- Note that e_{ij} means node j's attention to node i($e_{ij} \neq e_{ji}$). a is the attention function

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

- We then normalize all the neighborhood coefficients via softmax. $k \in \mathcal{N}_i$ denotes every neighbor of node i and including node i. (self loop is added to all the nodes since a node with no connection will be impossible to embed without it)

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

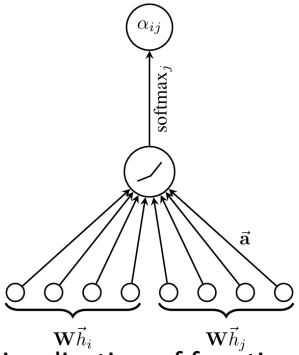
Architecture

Inside of attention function(a)

- Get's two embedded nodes as input and outputs a single real number(this is the attention coefficient).

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$$

- The way it works, concatenates(||) two nodes and feeds it to a single layer forward propagation with $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$.



visualization of function a

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

final operation of determining attention coefficient α_{ij}

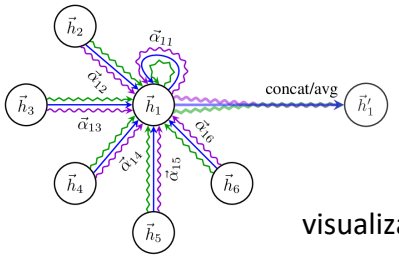
Architecture

The result

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

- However we can still make good use of “multi-head attention”, a technique that allows the model to catch diverse meanings of a single graph and at the same time normalizing it.

- Multi-head attention is simply using K machanisms and learn parameters in different environments



visualization of multi-head attention

Architecture

Two approaches to multi-head

- Concatenate all the neighborhood nodes to a single KF' size vector.

$$\vec{h}_i' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

$\alpha_{ij}^k, \boldsymbol{W}^k$: attention coefficient and weight matrix at k th head environment

This approach is not desirable since the embedding dimension might get too big than we wanted to.

- Averageing all the neighborhood embedding vectors

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

By averaging and then applying the activation function, we can obtain F' sized ebmedding vector.

Thus we select the averaging technique.

Performance

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 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

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)	**0.973** ± 0.002

- **Transductive learning on 3 benchmark graphs**

used metric : mean classification accuracy (with standard deviation)

- **Inductive learning with Protein-Protein interaction dataset**

used metric : F1 score

Conclusion

GAT is...

- Computationaly efficient : $O(|V|FF' + |E|F')$

  Time complexity linearly increases by the number of nodes and deges. No heavy operations are required(such as eigendecomposion).

- Capable of both transductive and inductive learning :

  Some previous models(GCN) were only capable to transductive learning which is costly when an unseen node appears in the process.

- Gives different attention to nodes :

  Has promising accuracy on datasets.

Networks Are Ubiquitous:
Building a Better World through Network Knowledge!

Contact: Sooho Moon (Email: moonwalk725@cau.ac.kr)