Graph Neural Networks

Qingyue Zhang, Xi’an Jiaotong University
zhangqingyue2019@stu.xjtu.edu.cn

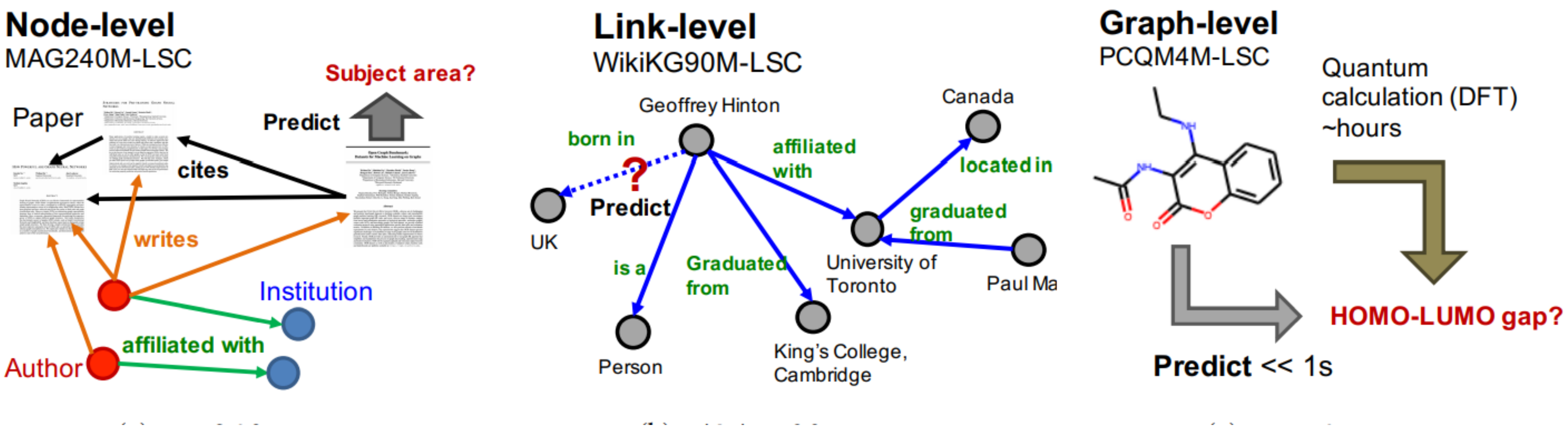
February 7th, 2022

DL trend

- CNN and deep NNs
- Word embeddings, LSTM and GRUs
- Attention, pre-trained language models
- Graph neural networks
- Few-shot learning and contrastive learning

Table of Contents

Deep Learning on Graphs Intro



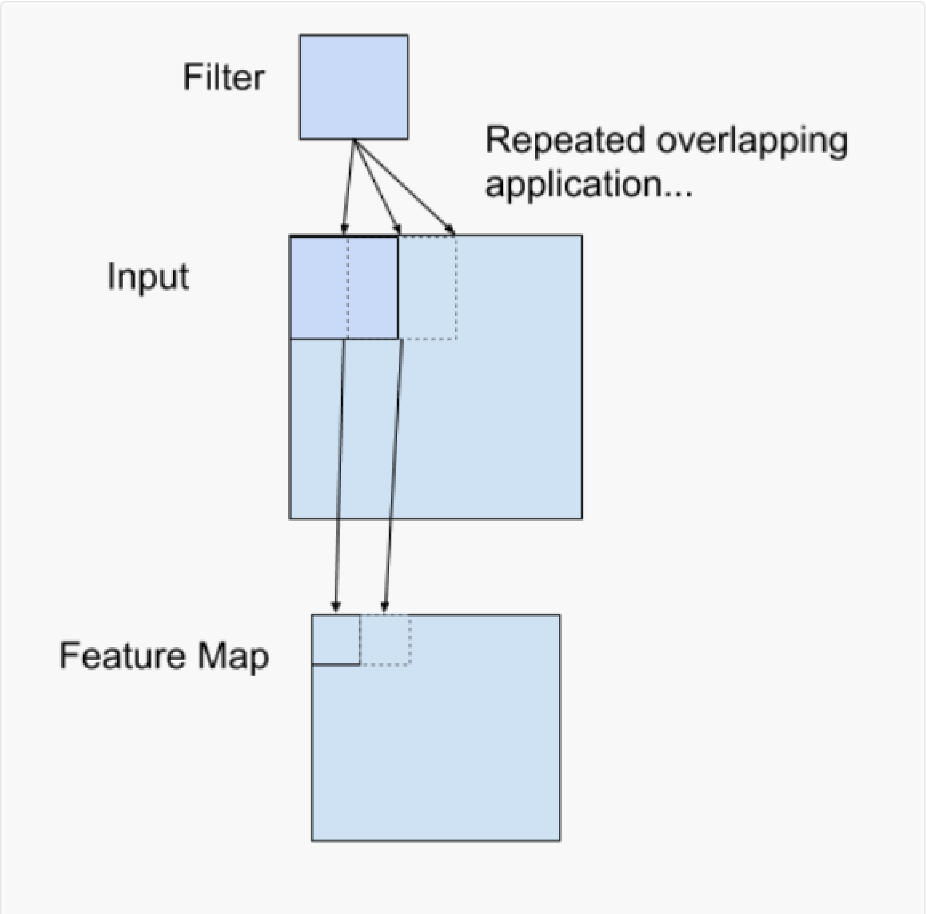
Node Encoding

- Shallow Encoding
  - One vector for each node, as learned parameters
  - Drawback: O(|V|), transductive, no node features
  - Q: Is word2vec shallow encoding?

- Deep Encoding
  - A series of NN to produce node embedding
  - Param # independent of |V|, inductive, incorporate node features

Table of Contents

Convolution



Single Node Standpoint Convolution

$$h_v^{(l)} = \sigma(W^{(l-1)} \sum_{u \in N(v) \cup \{v\}} \frac{h_u^{(l-1)}}{|N(v) \cup \{v\}|})$$

$$h_v^{(0)} = f_v, z_v = h_v^{(K)}, l = 1, ..., K$$

- h_v^{l}, hidden of node v at layer l, \sigma ReLU
- W^{l-1}, weight parameter of layer l-1
- N(v), neighborhood of node v; z_v, final output of node v
- Next up: Matrix Form GCN

Simple Convolution on Graphs

- What does convolution do?
  - Take neighbor’s input, apply weight and non-linearity

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right) ,$$

- *A* as adjacency matrix, *H^{(l)}* as hidden for layer-l
- *W^{(l)}* as weight for layer-l, *sigma* as ReLU
- What is H^{(0)}? What is learnable? Is it authentic convolution?

Solved: Convolution that Includes Itself

$$f(H^{(l)}, A) = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right) \ ,$$

- Where $\hat{A} = A + I$,
- What is A^{\^}H^{(l)} now?

- Imagine one node has 2 neighbors, one node has 2000
- Does it make a difference?

Solved: Averaging over Neighbor

- A

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-1} \hat{A} \; H^{(l)} W^{(l)} \right)$$

- Where $\hat{A} = A + I,$ and $\hat{D}$ is degree mat with self-connect

- A renormalization trick (symmetric operation):

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

Summary: GCN

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

- How is it a convolution?
- How does GCN leverage node features?
- Why GCN is deep encoder? How many trainable params?
- Inductive vs. transductive?

Table of Contents

GraphSAGE Motivation

- Ego issues
  - My features are important when evaluating myself
- Aggregation Methods
  - In GCN, average is taken on neighbors
  - Other aggregators? (addressed in GraphSAGE)
  - Different importance for different nodes? (addressed in GAT)

GraphSAGE

$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k \cdot \boxed{\mathrm{AGG}\left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)}, \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

- Generalized aggregator
  - Opens up to AGG() options
- Concatenation rather than addition
  - ‘ego’ issue

AGG(.) Choices

- AGG(.) must be
  - Order invariant
  - LSTM?

Mean: Take a weighted average of neighbors

$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

Pool: Transform neighbor vectors and apply symmetric vector function

Element-wise mean/max

$$\text{AGG} = \boxed{\gamma}\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

LSTM: Apply LSTM to reshuffled of neighbors

$$\text{AGG} = \text{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

Summary: GraphSAGE

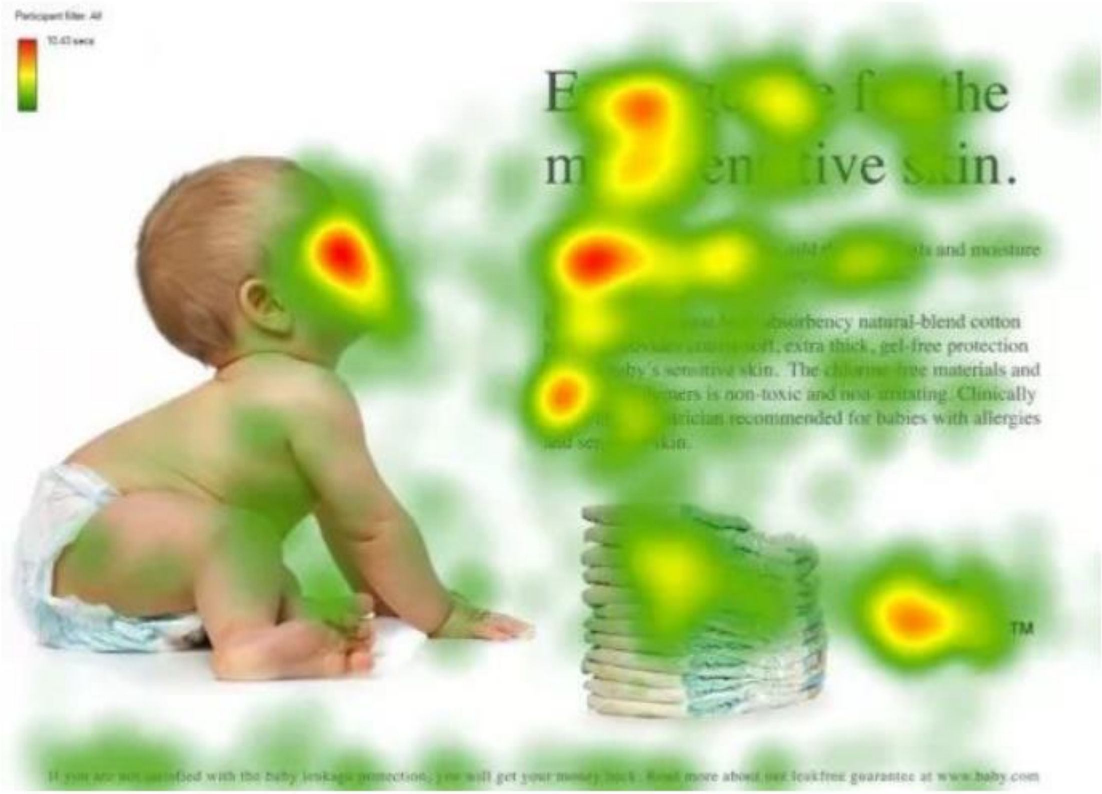
$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k \cdot \boxed{\text{AGG}\left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)}, \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

- AGG(.)
  - Mean, pool, LSTM

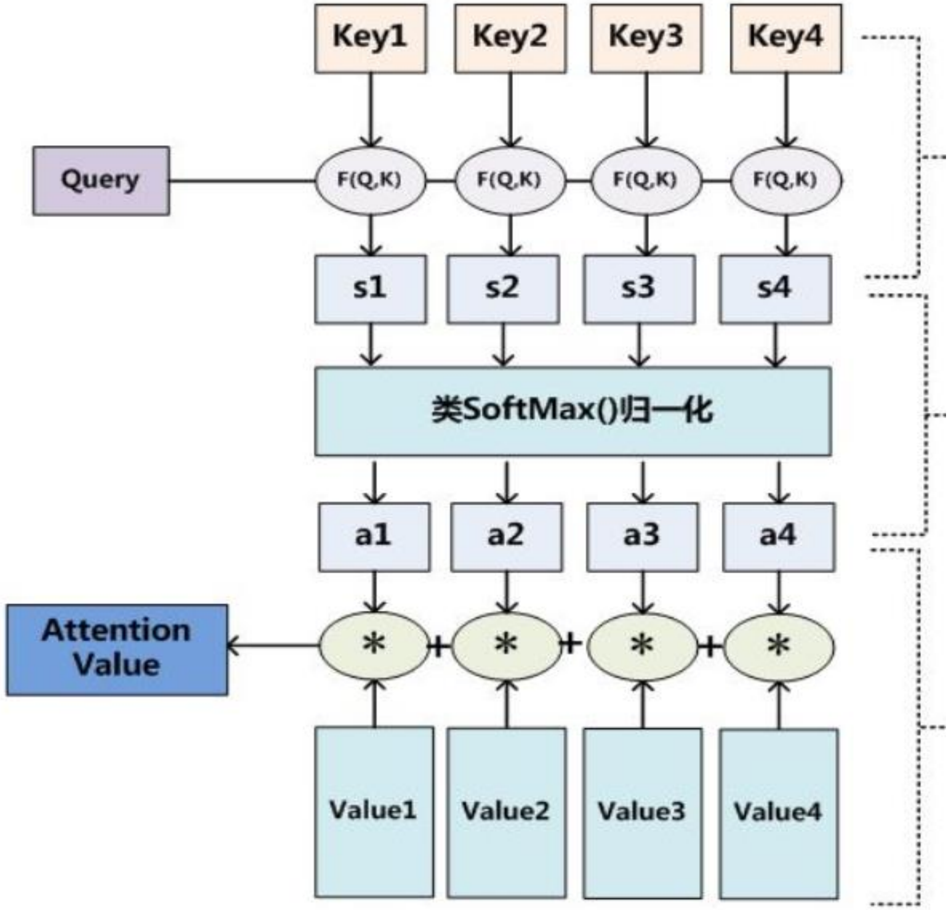
- When is GraphSAGE similar to GCN?
- Where is GraphSAGE different to GCN?
- Hidden Representation Size Increase?

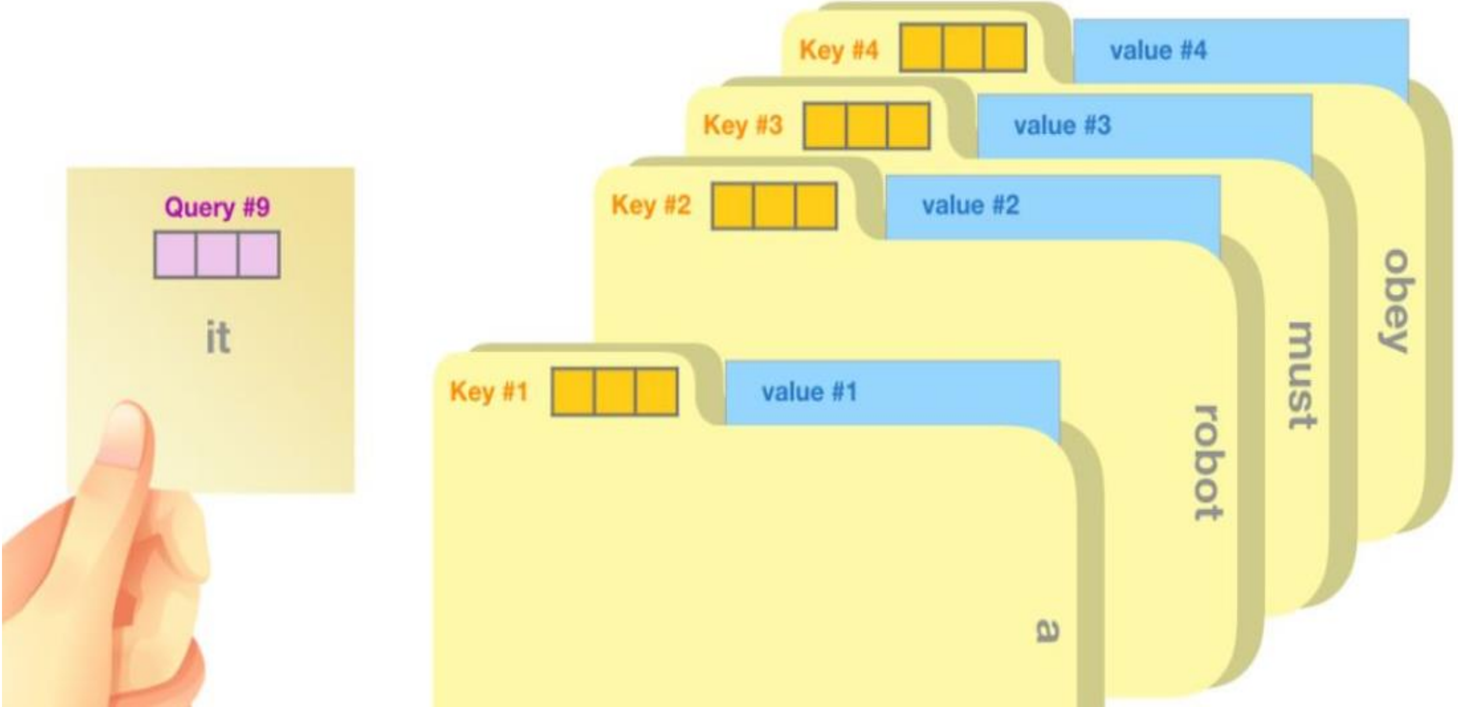
Table of Contents

Attention



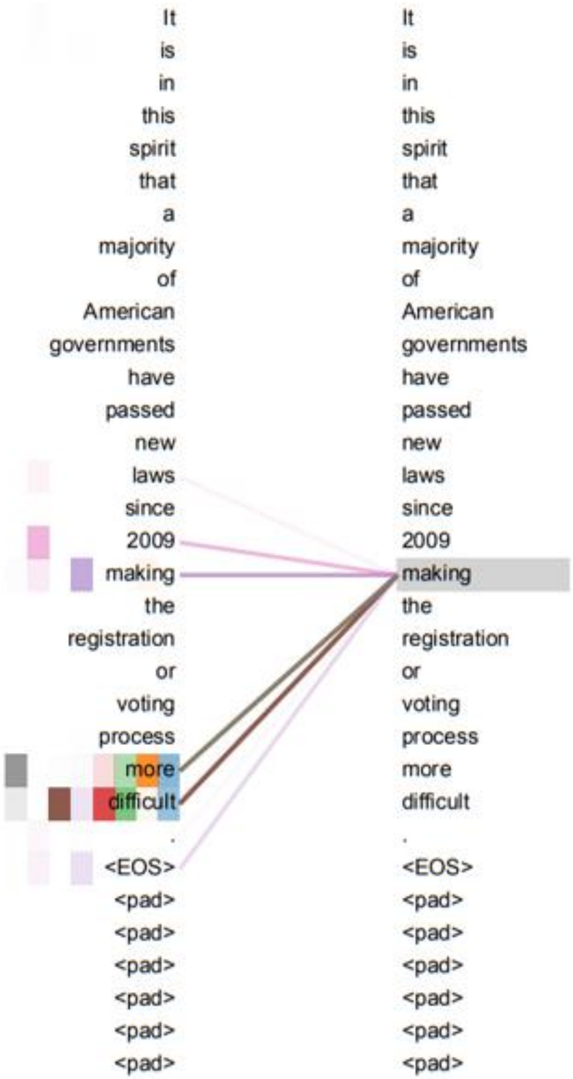
Attention

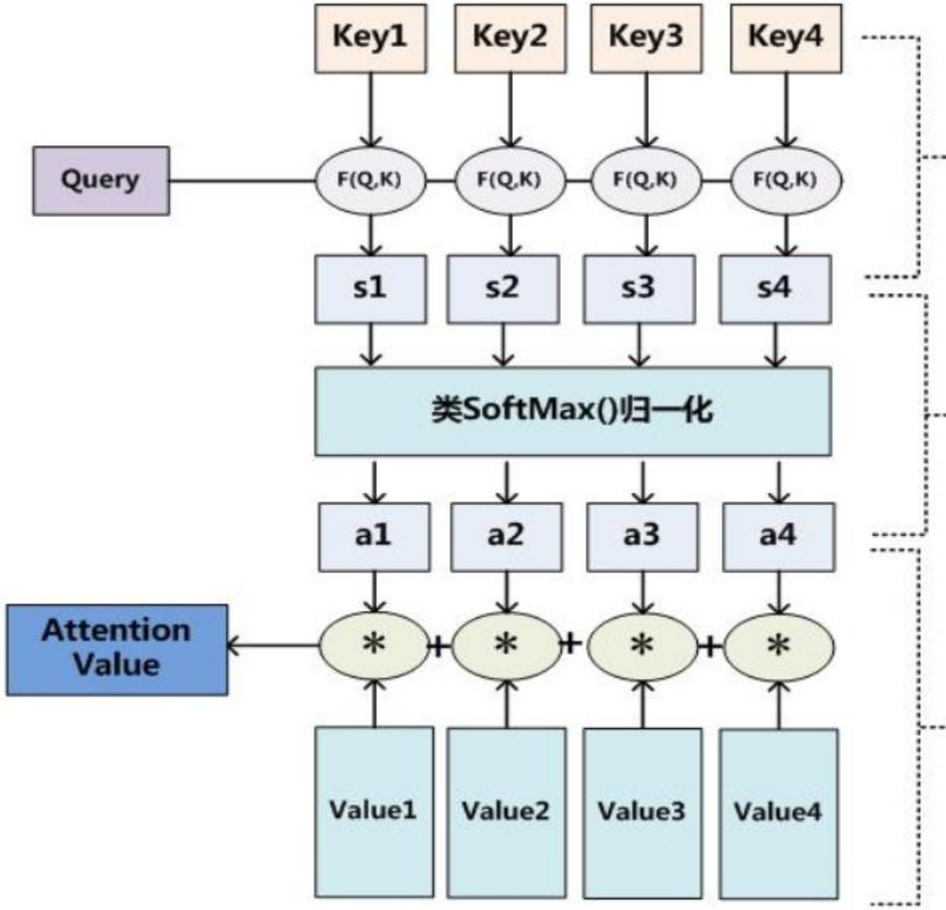

Key1
Key2
Key3
Key4
Query
F(Q,K)
F(Q,K)
F(Q,K)
F(Q,K)
s1
s2
s3
s4
类SoftMax()归一化
a1
a2
a3
a4
Attention Value
*
+
*
+
*
+
*
Value1
Value2
Value3
Value4


Query #9
it
Key #4
value #4
Key #3
value #3
Key #2
value #2
Key #1
value #1
obey
must
robot
a

Self-attention

- Q=K=V





Motivation

- In NLP
  - Not all tokens are equally important!
- In GNN
  - Not all nodes are equally important!
- Naturally, the incorporation of attention comes to mind

Q: How is it self-attention? (QKV?)

- Generate Attention Weight

$$e_{vu} = a(\boldsymbol{W}_k \boldsymbol{h}_u^{k-1}, \boldsymbol{W}_k \boldsymbol{h}_v^{k-1})$$

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

- Average with Weights

$$\boldsymbol{h}_v^k = \sigma(\textstyle\sum_{u \in N(v)} \alpha_{vu} \boldsymbol{W}_k \boldsymbol{h}_u^{k-1})$$

Summary: GAT

$$\boldsymbol{h}_v^k = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \boldsymbol{W}_k \boldsymbol{h}_u^{k-1}\right)$$

α_{vu} is **self-attention** weight on W_k h^k-1

Table of Contents

Summary

- GNN Framework
  - Node Encoder: Shallow vs Deep
- GCN
  - Imitates authentic convolution, matrix form
- GraphSAGE
  - Generalized aggregator, mean/pool/LSTM, ‘ego’ solution
- GAT
  - Self-attention over transformed hidden

Table of Contents

How to node classification with GNNs?

- a

torch_geometric

- Pytorch implementation of graph neural networks

https://pytorch-geometric.readthedocs.io/en/latest/index.html

CLASS GCNConv (**in_channels**: int, **out_channels**: int, **improved**: bool = `False`, **cached**: bool = `False`, **add_self_loops**: bool = `True`, **normalize**: bool = `True`, **bias**: bool = `True`, ****kwargs**) [source]

The graph convolut

Convolutional Netv

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

its diagonal degree

edge weights via th

Its node-wise formu

with $\hat{d}_i = 1 + \sum_{j}$

node `i` (default: `1`

CLASS GATConv (**in_channels**: Union[int, Tuple[int, int]], **out_channels**: int, **heads**: int = `1`, **concat**: bool = `True`, **negative_slope**: float = `0.2`, **dropout**: float = `0.0`, **add_self_loops**: bool = `True`, **edge_dim**: Optional[int] = `None`, **fill_value**: Union[float, torch.Tensor, str] = `'mean'`, **bias**: bool = `True`, ****kwargs**) [source]

The graph attentic

where the attentic

If the graph has m

computed as

$\alpha_{i,j} =$

CLASS SAGEConv (**in_channels**: Union[int, Tuple[int, int]], **out_channels**: int, **normalize**: bool = `False`, **root_weight**: bool = `True`, **bias**: bool = `True`, ****kwargs**) [source]

The GraphSAGE operator from the "Inductive Representation Learning on Large Graphs" paper

$$\mathbf{x}'_i = \mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \cdot \mathrm{mean}_{j \in \mathcal{N}(i)} \mathbf{x}_j$$

PARAMETERS:

- **in_channels** (*int or tuple*) – Size of each input sample, or `-1` to derive the size from the first input(s) to the forward method. A tuple corresponds to the sizes of source and target dimensionalities.
- **out_channels** (*int*) – Size of each output sample.
- **normalize** (*bool, optional*) – If set to `True`, output features will be ℓ_2-normalized, *i.e.*, $\frac{\mathbf{x}'_i}{\|\mathbf{x}'_i\|_2}$. (default: `False`)
- **root_weight** (*bool, optional*) – If set to `False`, the layer will not add transformed root node features to the output. (default: `True`)
- **bias** (*bool, optional*) – If set to `False`, the layer will not learn an additive bias. (default: `True`)
- ****kwargs** (*optional*) – Additional arguments of `torch_geometric.nn.conv.MessagePassing`.

CORA dataset

- The Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words.

Thx for Attention

Qingyue Zhang, Xi’an Jiaotong University

zhangqingyue2019@stu.xjtu.edu.cn

February 7th, 2022