Layer Stacking VS Attention

June 30, 2021

Dimitrios K. Kelesis
Introduction

Graph Convolution Network

Jumping Knowledge Graphs

Graph Attention Model

Oversmoothing & Discussion
Graph Preliminaries

- Graph defined as $G = (V, E, X, L)$
- $V$ stands for the vertex set
- $E$ stands for the edge set
- $X$ is the feature set (C features per node)
- $L$ is the set of labeled nodes
Intuition of GNNs

- Use current state and neighbors’ state to find future state.
- Every layer creates a new hidden state per node.
- \( h_i^{(l+1)} = \text{COMBINE}(\text{AGGREGATE}([h_j^l, j \in N(i) \cup [i]])) \)
- COMBINE and AGGREGATE functions define the model.
Graph Convolution Network

- First classical approach of GNNs.
- Each node aggregates information from its neighbors.
- Attempt to bridge the gap between CNNs and GNNs.
Graph Convolution Network

Every layer in a GCN network can be explained as:

1. $H^{(l+1)} = f(H^{(l)}, A)$
2. $H^{(0)} = X$
3. $H^{(L)} = Z$, $L$ is the total number of layers
Graph Convolution Network

- A is not the proper form to go.
- Normalize to $D^{-1/2}AD^{-1/2}$ to keep scale in vectors.

The final propose of a simple GCN is:

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

where $\hat{A} = A + I$
How it works?

▶ Central node aggregates information from its neighbors.
▶ Combines neighbor information with previous state.
▶ Formulate its new state.
Access of distant nodes results in **stacking** more layers.

- Stacking results in more information.
- Central node can ’see’ more hops away neighbors.
- Number of stacked layers (L) is limited.
- The smaller the diameter the smaller L.
JK-Nets

- Use the standard GNNs method.
- Aggregate Neighbors’ information.
- Combine aggregated information with node’s information.
- Stacking multiple layers is equivalent of aggregating information from distant nodes
JK-Nets

- Stacking layers assumes that final information properly incorporates all subsequent.
- JK-Nets leverage previous information from lower layers.
- Jumping Knowledge stands for aggregating all layers representation before extracting the final node representation.
JK-Nets

- Incorporate all intermediate information.
- Search about the best way to incorporate it.
- Concatenation / Max-Pool / LSTM-attention
JK-Nets show improvements in comparison with GAT and GCN

Indicate the importance of using intermediate states.

LSTM-attention allows each node to determine the importance of its neighborhood’s features on different layers and evaluate them.
Layer aggregation
Concat/Max-pooling/LSTM-attn

$h_v^{(4)} \in \mathbb{R}^{d_h}$
$h_v^{(3)} \in \mathbb{R}^{d_h}$
$h_v^{(2)} \in \mathbb{R}^{d_h}$
$h_v^{(1)} \in \mathbb{R}^{d_h}$

N. A.
N. A.
N. A.
N. A.

Input feature of node $v$: $X_v \in \mathbb{R}^{d_i}$
First attempt to use attention in GNNs
Not all neighbors have the same importance
Weight them!
In order to compute hidden state self-attend to neighbors.

Incorporated attention restricts only to 1-hop neighbors.

Guides learning within neighborhood.
The hidden state update is given by:

\[ h_i^{l+1} = \rho \left( \sum_{j \in \mathbb{N}(i)} \alpha_{i,j} Wh_j^l \right) \]

\[ \alpha_{i,j} = \text{softmax}(e_{i,j}), \]

\[ e_{i,j} = \text{LeakyReLU}(a^T[Wh_i \Vert Wh_j]) \]

Where \( a \) is the weight vector of a single layer MLP.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>GCN</th>
<th>JK - Nets</th>
<th>GAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use 1-hop neighbors</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Use previous state</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mix self and Neighbor states</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Use intermediate representations</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Use attention of neighbors</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Use distant nodes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Use paths between nodes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Point out the last two rows. Can we do something better? To be continued...
Oversmoothing

- Multiple graph convolutions makes node embeddings indistinguishable.
- Main problem that limits stacking in GNNs
- Limits the ability to construct deep GNNs
Can we use attention to overcome oversmoothing?

Using multi-layer architectures like JK-Nets with attention to avoid oversmoothing?

Is there a direct transformation from stacked layers with attention (over their hidden states) and GNNs with larger neighborhood definition?
Thank you!