2019sp-cs-8501-Deep2Read Scribe Notes: Large Scale Graph Representation Learning

Scribe: Ryan McCampbell

June 1, 2019

1 Introduction

Graphs have many diverse applications from social networks to biochemistry. A common task for graphs is node classification. To do this we need to create or learn node-level features that incorporate each node's and its neighbor's attributes.

2 Naïve Approach: Adjacency matrix

Use adjacency matrix as feature: feed rows from adjacency matrix plus other features directly into fully-connected network.

Problems:

- $O(n)$ parameters: doesn't scale
- Can't generalize between different sized graphs
- Not node order invariant

3 Convolutions

Intuition: CNNs on images apply a transformation to the neighbors of a particular node (pixel) and aggregate them. We can do the same thing with nodes in a graph.

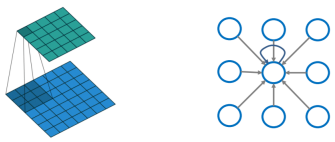

Figure 1: Convolutions on images vs. graphs

But how do we handle variable numbers of neighbors? And how do we determine canonical ordering?

4 GraphSAGE

What we want:

- Invariant to node ordering
- Locality: node operations depend on neighbors
- Model parameters independent of graph size
- Model independent of graph structure: able to transfer to other graphs

Idea: each node defines computational graph; each edge is transformation/aggregation

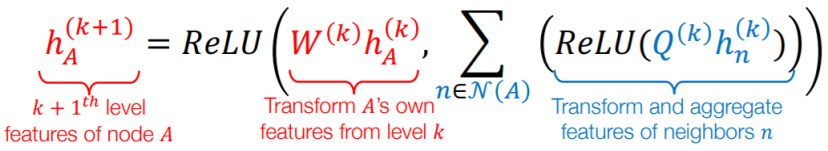

Figure 2: GraphSAGE update function

To compute node embeddings we iteratively update the feature vector for each node by combining it with an aggregate transformation of its neighbor's feature vectors. We use the same weights for each neighbor, so there is no order dependency. We however use distinct weights at each iteration so nodes at different distances have different amounts of influence on a given node.

Aggregation: The aggregation function (shown as a sum in the picture) can be any permutation-invariant function. The functions discussed in the paper are:

- Mean - similar to ordinary convolution
- LSTM - since RNNs are normally not order-invariant the LSTM is trained with random permutations of neighbors.
- Max pooling

Training: The embeddings are trained in an unsupervised manner with a loss function that encourages nearby nodes to have similar embeddings, and farther away nodes more distinct embeddings. It can also be trained end-to-end in a classifier with cross-entropy loss.

GraphSAGE is equivalent to the Weisfeiler-Lehman graph isomorphism test if the trainable aggregate function is replaced with a hash function.

5 Experiments

GraphSAGE was tested on three tasks: citation graphs, reddit posts and protein interaction networks. It outperformed the state of the art in all three tasks.

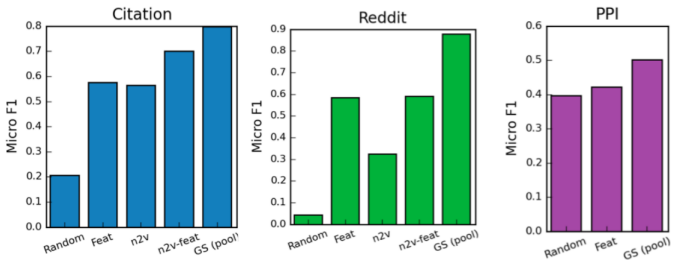

Figure 3: Results

6 Conclusion

Graph convolution-based methods can combine node information with network structure to learn useful node representations.