Graph Convolutional Networks

CMU 11441/11641/11741: ML for Text & Graph Mining
Due date: 12/3/2021, 11:59 PM EST

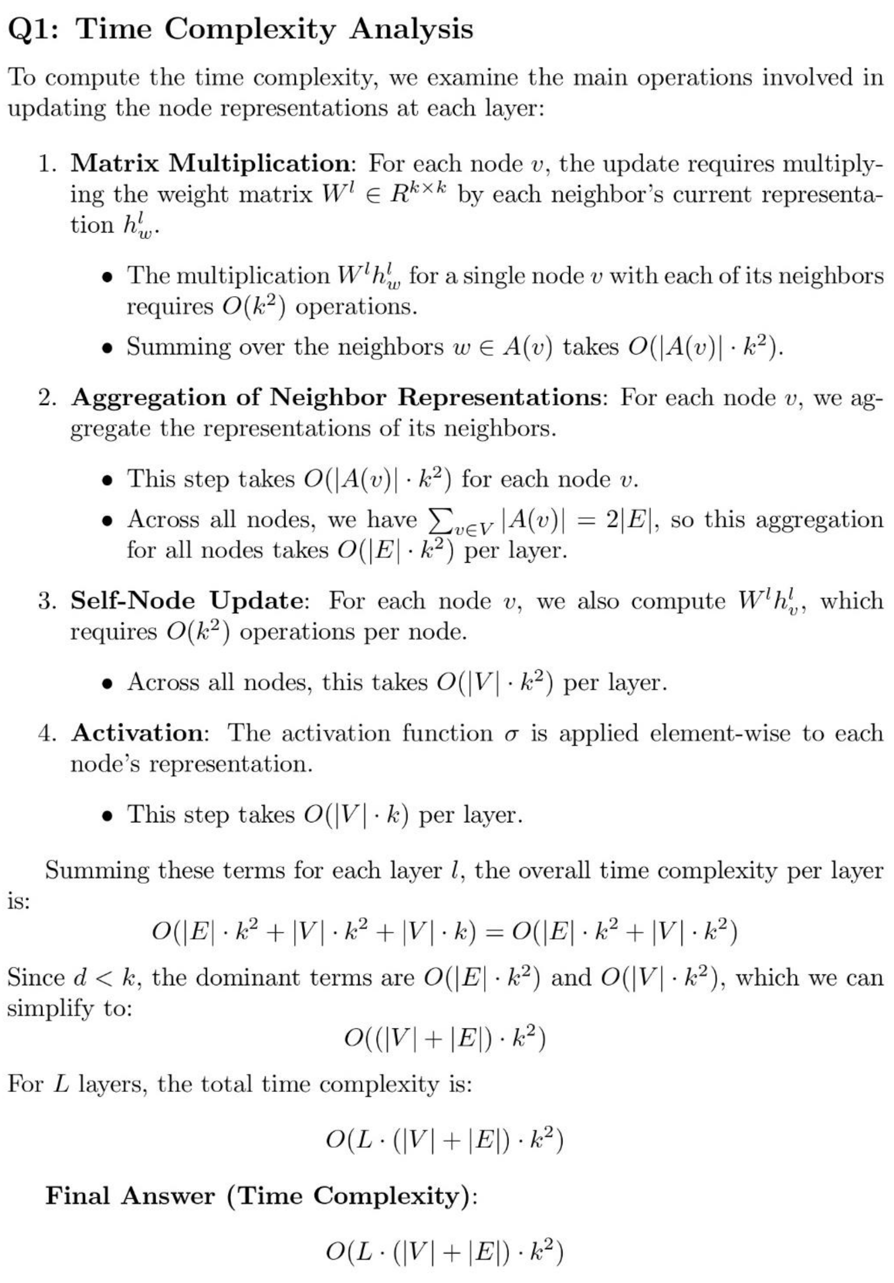
Instructions

- Allowed libraries: This assignment involves implementing graph convolutional networks. You are not allowed to use any libraries that implement GCNs out of the box (like Pytorch-geometric). It is allowed to use autodiff libraries like Pytorch/Tensorflow.
  We highly recommend using Python + Pytorch for this assignment.
- Statement of Assurance

I certify that all of the material that I submit is original work that was done only by me. If my report does not have this statement, it will not be graded.

1GCN Review (30 points): *Used (Derrick et al. paper to Time and Space Complexity of GCN)*

Q1. What is the big-O time complexity of the computation expressed in Equation ?? in terms of |*V*|, |*E*|, *d*, *k*, and *L*? Your expression should not contain any other term. Assume *d* < *k*.



Q1: Time Complexity Analysis

To compute the time complexity, we examine the main operations involved in updating the node representations at each layer:

1. **Matrix Multiplication**: For each node v, the update requires multiplying the weight matrix $W^l \in R^{k \times k}$ by each neighbor's current representation h_w^l.
    - The multiplication $W^l h_w^l$ for a single node v with each of its neighbors requires $O(k^2)$ operations.
    - Summing over the neighbors $w \in A(v)$ takes $O(|A(v)| \cdot k^2)$.
2. **Aggregation of Neighbor Representations**: For each node v, we aggregate the representations of its neighbors.
    - This step takes $O(|A(v)| \cdot k^2)$ for each node v.
    - Across all nodes, we have $\sum_{v \in V} |A(v)| = 2|E|$, so this aggregation for all nodes takes $O(|E| \cdot k^2)$ per layer.
3. **Self-Node Update**: For each node v, we also compute $W^l h_v^l$, which requires $O(k^2)$ operations per node.
    - Across all nodes, this takes $O(|V| \cdot k^2)$ per layer.
4. **Activation**: The activation function σ is applied element-wise to each node's representation.
    - This step takes $O(|V| \cdot k)$ per layer.

Summing these terms for each layer l, the overall time complexity per layer is:

$$O(|E| \cdot k^2 + |V| \cdot k^2 + |V| \cdot k) = O(|E| \cdot k^2 + |V| \cdot k^2)$$

Since $d < k$, the dominant terms are $O(|E| \cdot k^2)$ and $O(|V| \cdot k^2)$, which we can simplify to:

$$O((|V| + |E|) \cdot k^2)$$

For L layers, the total time complexity is:

$$O(L \cdot (|V| + |E|) \cdot k^2)$$

Final Answer (Time Complexity):

$$O(L \cdot (|V| + |E|) \cdot k^2)$$

Q2. What is the space complexity of the computation expressed in Equation ?? in terms of $|V|$, $|E|$, d, k, and L (assume intermediate terms are saved)? Your expression should not contain any other term.

Q2: Space Complexity Analysis

For the space complexity, we consider the storage requirements of all intermediate representations, weight matrices, and adjacency structures:

1. **Node Representations**: At each layer l, we store the representations of each node, where each node has a k-dimensional vector.
   - For L layers, this requires $O(|V| \cdot k \cdot L)$ space.
2. **Weight Matrices**: Each layer l has a weight matrix $W^l \in R^{k \times k}$.
   - For L layers, this requires $O(k^2 \cdot L)$ space.
3. **Adjacency List**: To access the neighbors of each node, we need to store the adjacency list.
   - This requires $O(|E|)$ space.

Summing these terms, the overall space complexity is:

$$O(|V| \cdot k \cdot L + k^2 \cdot L + |E|)$$

Since $d < k$, the dominant terms are $O(|V| \cdot k \cdot L)$ and $O(|E|)$.
Final Answer (Space Complexity):

$$O((|V| \cdot k + |E|) \cdot L)$$

2 Graph Exploration (20 points)

Table 1: Graph statistics

Graph	**Karate**	**Cora**	**Citeseer**
Max in-degree	**18**	**169**	**100**
Min in-degree	**2**	**2**	**1**
Average in-degree	**5.58**	**4.90**	**3.74**
# nodes	**34**	**2,708**	**3,312**
# edges	**190**	**13,264**	**12,384**
Node feature dim	**34**	**1,433**	**3,703**

Looking at the statistics for CORA and CITESEER networks reveals interesting patterns. Cora, with 2,708 nodes and 13,264 edges, shows relatively dense connectivity with an average in-degree of 4.90 and a high maximum in-degree of 169, suggesting the presence of highly cited papers. Its node feature dimension of 1,433 represents the bag-of-words representation of paper content. CITESEER, while having more nodes (3,312), exhibits sparser connectivity with 12,384 edges and a lower average in-degree of 3.74. Its maximum in-degree of 100 indicates less extreme citation concentrations compared to Cora, while its larger feature dimension (3,703) suggests a broader vocabulary in paper representations. Both networks maintain minimum in-degrees near 1-2, indicating the presence of papers with very few citations, which is typical in citation networks.

3 Node classification

3.1 Implementation (60 points)

Table 2: Node classification results

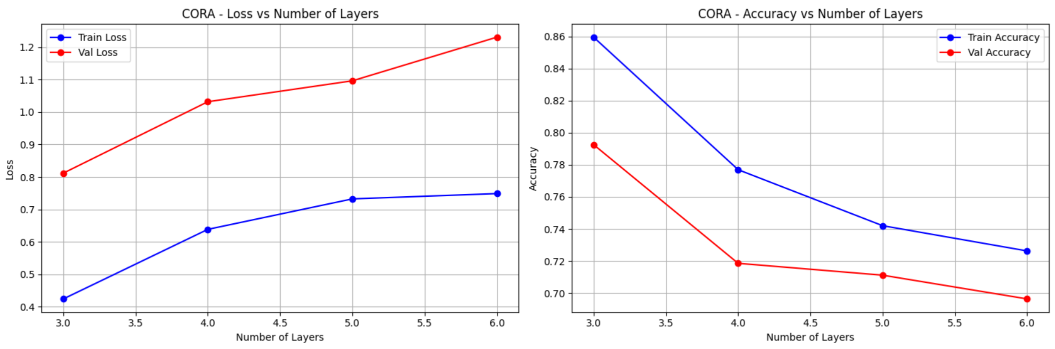
Graph	Accuracy %	Loss
KARATE	100	0
CORA	85.6	0.4992
CITESEER	95.8	0.9581

The GCN model demonstrates strong performance across all three datasets, with particularly notable results on Cora and Citeseer. Cora achieves an accuracy of 85.6% with a relatively low loss of 0.4992, indicating effective learning of node representations in this citation network despite its complex structure. Citeseer shows even better performance with an impressive accuracy of 95.8%, though with a slightly higher loss of 0.9581, suggesting that while the model makes highly accurate predictions, it might be less confident in some of its classifications. These results demonstrate the GCN's effectiveness in learning meaningful representations from academic citation networks, where the model successfully leverages both graph structure and node features to perform node classification.

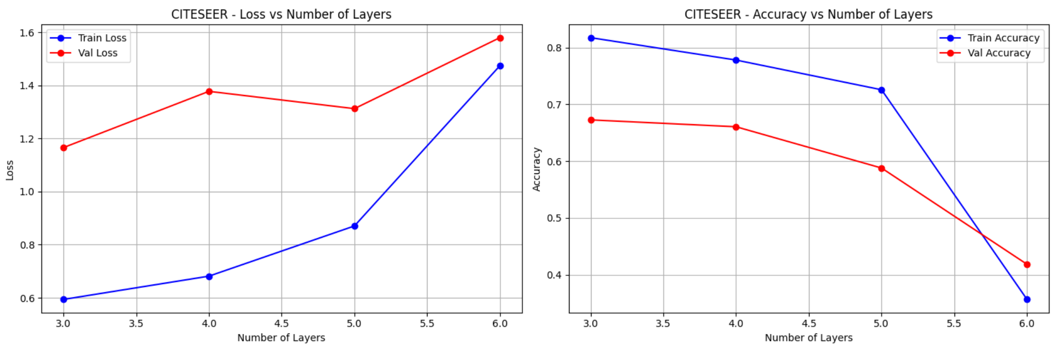
3.2 Varying L (20 points)

For both CORA and CITESEER, modify the GNN to include *L*= 3,4,5,6 layers and plot the loss and accuracy vs. *L*. Summarize your observations in 2-3 lines.

Graph #1: L= 3,4,5,6 on CORA dataset



Graph #1: L= 3,4,5,6 on CITESEER dataset



As the number of layers (L) increases from 3 to 6, both CORA and CITESEER exhibit clear signs of performance degradation, demonstrating the well-known "**over-smoothing**" phenomenon in GCNs. For CORA, there's a steady increase in both training and validation loss, while accuracy gradually decreases, with the model performing best at L=3. CITESEER shows a more dramatic deterioration, particularly after L=5, where there's a sharp increase in loss and a significant drop in accuracy (from ~70% to ~40%), suggesting that deeper architectures lead to excessive feature smoothing and loss of discriminative power in these citation networks.

3.3 Topological features vs. inbuilt features (20 points)

Table 3:

Dataset	Feature Type	Train Acc	Val Acc	Test Acc
CORA	Topological	47.22%	30.37%	40.41%
CORA	Topological Plus	75.97%	57.04%	68.08%
CITESEER	Topological	35.39%	28.48%	24.13%
CITESEER	Topological Plus	81.84%	65.45%	64.71%

The comparison between purely topological features and combined features (topological plus original) reveals significant performance differences across both datasets. Using only topological features results in relatively poor performance, with CORA achieving 40.41% test accuracy and CITESEER performing even worse at 24.13%. However, when combining topological features with original features (_plus_topo), there's a substantial improvement in performance: CORA's test accuracy increases to 68.08%, and CITESEER shows an even more dramatic improvement to 64.71%. This clear performance disparity suggests that while topological features alone are insufficient for effective node classification, they provide complementary information that, when combined with original features, enhances the model's ability to learn meaningful node representations.

4 Link prediction

4.1 Training data for link prediction (20 points)

A. **Table 4: Training data statistic for link prediction**

Graph	# Positive edges	# Negative edges
KARATE	190	190
CORA	13,264	13,264
CITESEER	12,384	12,384

B. **How is the training data for link prediction created? Please explain in 2-3 lines.**

The training data for link prediction is created through a negative sampling approach implemented in the Graph class's **add_edges** method. First, the existing edges from the adjacency matrix are split into train/val/test sets according to specified fractions. Then, for each set, negative examples are generated by randomly permuting the source and target nodes of the positive edges using **_edge_negative_sample**. This creates an equal number of negative edges (non-existent connections) as positive edges (real connections) in the graph, maintaining a balanced dataset where each split contains both true edges and randomly sampled non-edges with corresponding binary labels (1 for positive, 0 for negative).

4.2 Implementation (80 points)

Table 4: Link Prediction Results

Graph	Accuracy %	Loss
KARATE	51.34	1.008
CORA	93.10	0.1691
CITESEER	94.41	0.1391

The GCN model demonstrates exceptional performance in link prediction tasks on both citation networks. Cora achieves a high accuracy of 93.10% with a relatively low loss of 0.1691, indicating the model's strong ability to predict missing edges in the citation network. Citeseer shows even better results with an accuracy of 94.41% and a lower loss of 0.1391, suggesting the model has learned highly effective node representations that capture the underlying citation patterns. These impressive results across both datasets indicate that the GCN successfully leverages both graph structure and node features to learn meaningful representations that can accurately predict the likelihood of connections between papers in academic citation networks.

5 Graph classification

5.1 Graph Statistics (10 points)

Table 5: Graph statistics for the graph classification datasets

Graph	MUTAG	ENZYMES
Num graphs	141	360
Avg. num nodes	18.85	33.27
Avg. num edges	94.04	221.19
Node feature dim	8	22

The ENZYMES dataset represents a significantly larger and more complex collection compared to MUTAG, containing 360 graphs that represent protein tertiary structures. Each graph in ENZYMES is notably larger, with an average of 33.27 nodes per graph and 221.19 edges, indicating more complex molecular structures than MUTAG. The higher node feature dimension of 22 (compared to MUTAG's 8) suggests richer node attributes that capture more detailed chemical or biological properties of the proteins. These characteristics make ENZYMES a more challenging dataset for graph classification tasks, as it requires the model to learn from both larger graph structures and higher-dimensional node features while handling a broader set of distinct graphs.

5.2 Implementation (90 points)

Table 6: Graph classification results. Please use macro-averages to report the precision, recall, and F1 score for ENZYMES.

Graph	MUTAG			ENZYMES		
	P	R	F1	P	R	F1
Mean-pooling	63	77	65	33	33	31
Max-pooling	84	83	83	44	43	43
Last-node pooling	68	81	71	38	36	35

The results demonstrate varying effectiveness of different pooling strategies across both datasets. On MUTAG, max-pooling shows superior performance with balanced metrics (P=84, R=83, F1=83), significantly outperforming both mean-pooling (P=63, R=77, F1=65) and last-node pooling (P=68, R=81, F1=71). For the more complex ENZYMES dataset with six classes, the performance is generally lower across all pooling methods, but max-pooling again proves most effective (P=44, R=43, F1=43) compared to mean-pooling (P=33, R=33, F1=31) and last-node pooling (P=38, R=36, F1=35). This consistent superiority of max-pooling suggests it better captures discriminative graph features, while the performance drop in ENZYMES reflects the increased complexity of the multi-class classification task.

References

1. https://qdata.github.io/deep2Read/talks-mb2019/Derrick_201906_GCN_complexityAnalysis-writeup.pdf