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

1. Did you receive any help whatsoever from anyone in solving this assignment? YES
2. Did you give any help whatsoever to anyone in solving this assignment? YES
3. Did you find or come across code that implements any part of this ass YES

1 GCN Review (30 points)

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 $d < k$.

Q2·1

$$h_v^{(l+1)} = \sigma\left(\frac{1}{A(v)} \sum_{w \in A(v)} W^l h_w^l + W^l h_v^l\right).$$

for each layer:

There are $|A(v)|+1$ matrix multiplication. and plus. per node

for each multiplication the time complexity is $O(k \cdot k) = O(k^2)$

$|A(v)|$ on average equals to $\frac{2|E|}{|V|}$

$\therefore$ Time complexity is $O(|V| \frac{2|E|}{|V|} \cdot k^2 \cdot L) = O(|E| k^2 L)$.

for $d = k$ $\therefore$ This is a upper bound.

More precisely, the first layer we have $O(|E| d k)$.

For all the other layer we have $O(|E| k^2)$.

Thus, total time complexity is $O(|E| d k + |E| k^2 (L-1))$.

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.

Q 2-2.

For first layer

H^1 takes $|V| \times d$ space

W^1 takes $k \times d$ space

For other layer:

H^L takes $|V| \times k$

W^L takes k^2

All in all:

Space upper bound = $L \cdot (|V| \times k + k^2)$ · data bytes.

space precise = $\underbrace{(L-1)(|V| \times k + k^2)}_{L-1 \text{ layer}} + \underbrace{|V| \times d + k \times d}_{\text{first layer}}$.

2 Graph Exploration (20 points)

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	2708	3312
# edges	190	13264	12384
Node feature dim	34	1433	3703

Table 1: Graph statistics

3 Node classification

3.1 Implementation (60 points)

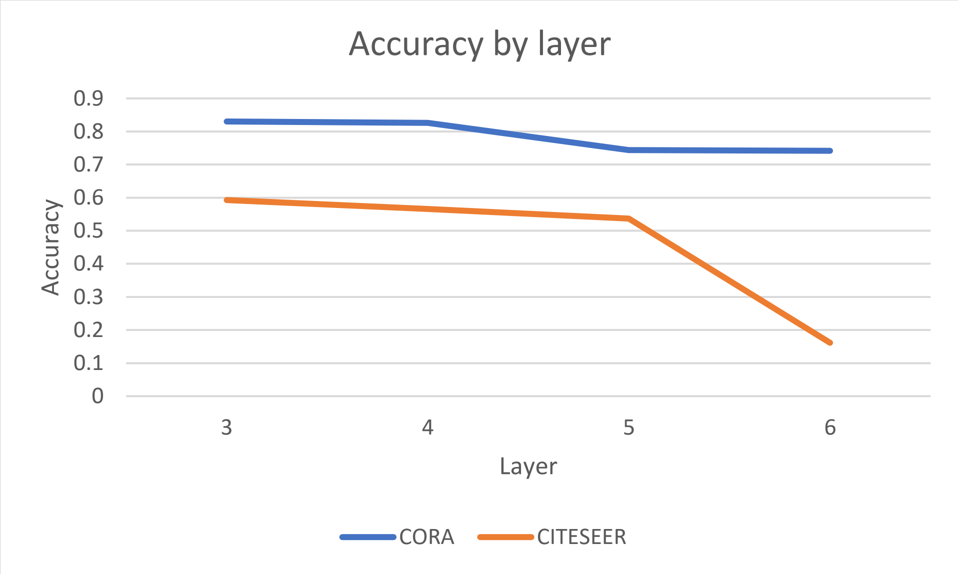
Graph	Accuracy %	Loss
KARATE	100	0
CORA	0.8579	0.5009
CITESEER	0.6697	0.9573

Table 2: Node classification results

3.2 Varying L (20 points).

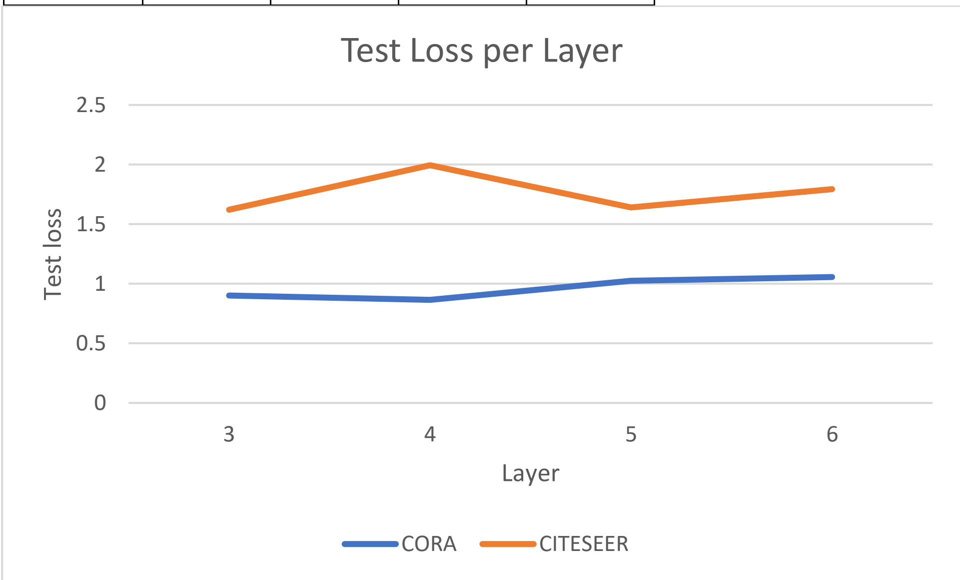
Accuracy:

acc	3	4	5	6
CORA	0.8303	0.8266	0.7435	0.7417
CITESEER	0.5928	0.5656	0.537	0.1614



Loss:

loss	3	4	5	6
CORA	0.901	0.8653	1.0238	1.0565
CITESEER	1.6213	1.9947	1.6411	1.7943



Observation:

Using the same training parameters, the deeper GCN generally has worse performance.

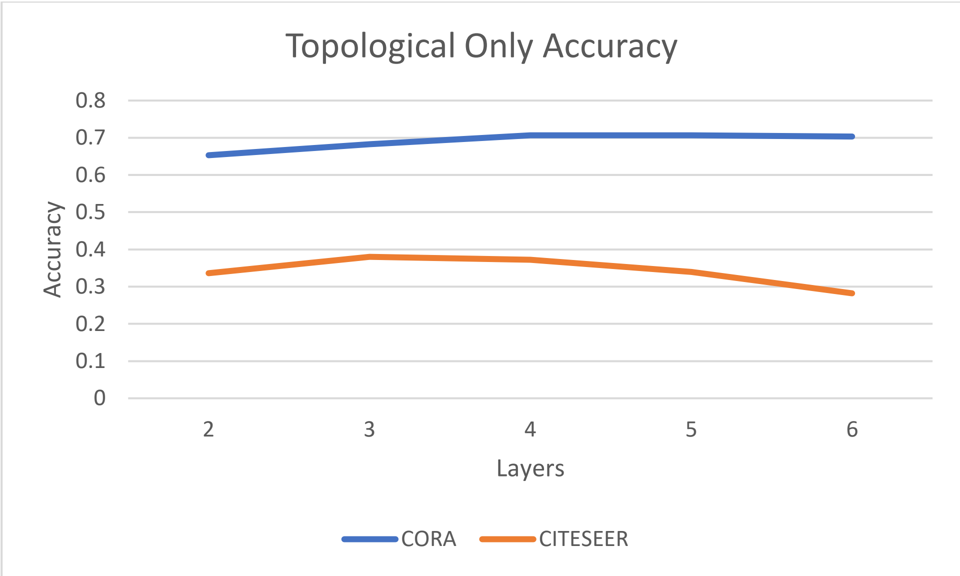
The loss generally increases along with layers and accuracy drops along with layers.

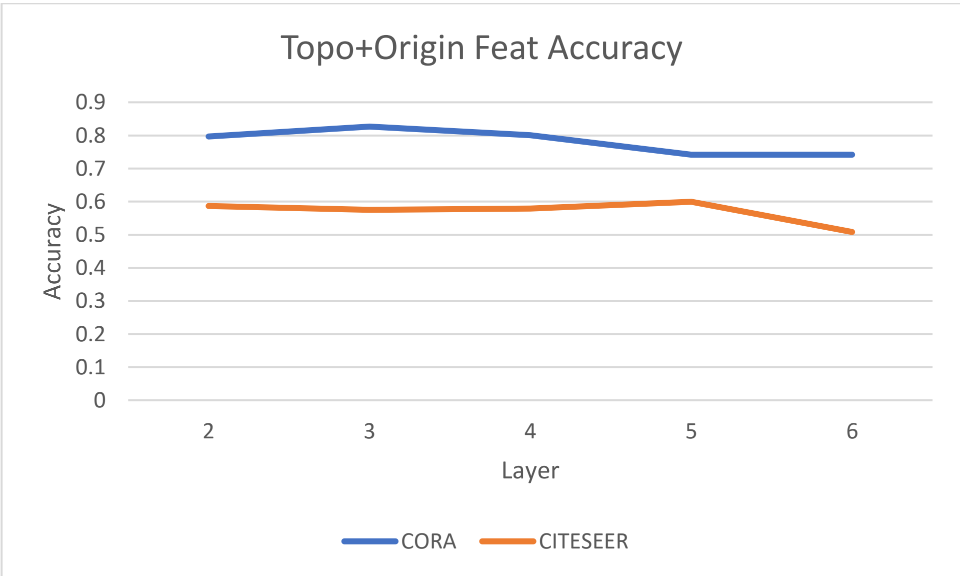
When layer=6, the original training algorithm is at a very bad point for CITESEER set.

Although deeper network has greater power, it is hard to train. In the latter section, I will not only try increasing layers, but decease lr and increase epochs as well.

3.3 Topological features vs. inbuilt features (20 points)

With only topological features, the result of test set accuracy and loss after preliminary hyperparameter tuning is:





The tables of results are:

TOPO Only					
acc	2	3	4	5	6
CORA	0.6531	0.6827	0.7066	0.7066	0.703
CITESEER	0.3363	0.3801	0.3725	0.3394	0.2821

TOPO +Original feature					
acc	2	3	4	5	6
CORA	0.797	0.8266	0.8007	0.7417	0.7417
CITESEER	0.5867	0.5747	0.5792	0.5996	0.5083

According to the results, I think that topological features should have some kind of discriminatory power. However, the topological features need extra mechanism to handle because during training, topological features require a higher learning rate. That may explain why they all under perform the original features.

4 Link prediction

4.1 Training data for link prediction (20 points)

A.

Graph	# Positive edges	# Negative edges
KARATE	190	190
CORA	7960	13264
CITESEER	12384	12384

Table 3: Training data statistic for link prediction

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

The training data is generated through proper negative sampling technique. First, it counts the existing edges and then manually establish some dummy edge as negative sample. This technique can be improved by smart sampling. For example, sample from each edge's one point.

4.2 Implementation (80 points)

Graph	Accuracy %	Loss
KARATE	51.34	1.008
CORA	0.9131	0.2129
CITESEER	0.9101	0.2161

Table 4: Link Prediction Results

For this question, I exhaust all the measurement of implementation and try countless time. I will detail my implementation attempt and result. In general, my attempt uses three class of feature aggregation for links.

General Implementation Structure

The forward pass of this task consists of two parts. The first part is the GCN. Original features are fed into GCN, and we obtain the raw logits output of gcn without any activation in the last layer.

The, we generate a representation of link embedding and then let it go through a classifier for link classification.

Concatenation

In this approach, after obtaining the features from GCN, I concatenate two features and feed them into a MLP classifier. The classifier can be one layer or multiple layer. However, my experiment all get accuracy of 50%. I tried using ReLU() layer after GCN and adding dropout layer but none of them work.

Difference

In this approach, I use the difference/absolute difference of the linear projection/Identity projection of the GCN features.

Product

I attempt using product of the projection of two features.

5 Graph classification

5.1 Graph Statistics (10 points)

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

Table 5: Graph statistics for the graph classification datasets

5.2 Implementation (90 points)

Graph	MUTAG			ENZYMES		
	P	R	F1	P	R	F1
Mean-pooling	67.0	64.7	65.3	41	40	40
Max-pooling	84	83	83	46	48	44
Last-node pooling	62	60	61	44	43	42

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

References

Thanks Ruohong so much for your valuable help!