Graph Convolutional Networks

CMU 11441/11741: Machine Learning with Graphs

Due date: 04/09/2024, 11:59 PM EST

<https://github.com/cmu-ml4graph/gcn_assignment_s2024> <https://www.overleaf.com/read/czfccfjtrczk#db25dc>

# Instructions

* **Allowed libraries:** This assignment involves implementing graph convolutional net- works. 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.
* **Getting feedback**: You can create a **private** fork of the repository on GitHub and add the TAs as collaborators (usernames: Edward-Sun). This might help you in asking questions without having to copy-paste your code on piazza (you can just reference Github code/copy permalink). You can use these instructions or just copy-paste the code into a new repository.
* **Posting your solutions online**: As with all the other assignments, please do not share your solutions publicly.

### Statement of Assurance

* 1. Did you receive any help whatsoever from anyone in solving this assignment?
     + Yes, I have a verbal discussion with Blessed.
  2. Did you give any help whatsoever to anyone in solving this assignment?
     + Yes, I have had a verbal discussion with Blessed.
  3. Did you find or come across code that implements any part of this assignment?
     + No, I did not.

# 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*.

### Solution

### Performing a dot product of two vectors of size n would take a time complexity of O(n). We have a matrix W of size k x k. For each neighbor of v with a hidden vector of k \* 1, we are performing a multiplication. This has a time complexity of k x k. If a vector v has E neighbors, we would need to perform |E| x k x k multiplications. If we have L layers then the overall time-complexity would be O(|V| ×|E| ×k2 ×L)

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.

### Solution

### In the above equation the weight matrix W has a dimension of K x K. The hidden matrix at current timestep and next timestep have a dimension of |V| x K. The Space complexity for the above computation is there for O(|V| \*K).

# Graph Exploration (20 points)

### Solution

|  |  |  |  |
| --- | --- | --- | --- |
| Graph | Karate | Cora | Citeseer |
| Max in-degree | 18 | 169 | 100 |
| Min in-degree | 2 | 2 | 1 |
| Average in-degree | 5.58 | 4.89 | 3.73 |
| # nodes | 34 | 2708 | 3312 |
| # edges | 190 | 13264 | 12384 |
| Node feature dim | 34 | 1433 | 3703 |

Table 1: Graph statistics

# Node classification

## Implementation (60 points)

### Solution

|  |  |  |
| --- | --- | --- |
| Graph | Accuracy % | Loss |
| KARATE | 100 | 0 |
| CORA | 82.66 | 0.88 |
| CITESEER | 66.67 | 1.12 |

Table 2: Node classification results

## 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.

**Solution**

**Graphs for Cora**

**A graph of a loss

Description automatically generated**

**A graph of a graph

Description automatically generated**

Increasing the number of layers from 2 to 7 does not help increase the performance for Cora. In fact, as the number of layers increased the loss stopped decreasing and the accuracy was low. This shows that the increasing the number of layers is not an effective strategy. The experiment with 2 GCN layers achieved the highest performance of 82.66 on the test data, while all the other experiments resulted in lower performance.

**Graphs for Citeseer**

**A graph of a line

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated**

Adding more layers, specifically from 2 to 7, did not improve performance for the Citeseer dataset. Again, as the layer count increased, there was no further reduction in loss and accuracy remained very low. This indicates that adding layers isn't an effective approach. The experiment with 2 GCN layers achieved the best performance, reaching 66.67 accuracy on the test set, while other experiments yielded lower results.

## Topological features vs. inbuilt features (20 points)

**Solution**

|  |  |  |
| --- | --- | --- |
| Graph | \_topo | plus\_topo |
| CORA | 60.33 | 82.73 |
| CITESEER | 33.33 | 66.90 |
|  |  |  |

# Link prediction

## Training data for link prediction (20 points)

### Solution

|  |  |  |
| --- | --- | --- |
| Graph | # Positive edges | # Negative edges |
| KARATE | 190 | 190 |
| CORA | 13264 | 13264 |
| CITESEER | 12384 | 12384 |

Table 3: Training data statistic for link prediction

* + 1. How is the training data for link prediction created? Please explain in 2-3 lines.

**Solution**

The training data for link prediction is generated by extracting the edge list from the adjacency matrix, which constitutes the positive edges. To complement this, negative edges are created by randomly permuting nodes. The dataset is divided into train, validation, and test.

## Implementation (80 points)

### Solution

|  |  |  |
| --- | --- | --- |
| Graph | Accuracy % | Loss |
| KARATE | 51.34 | 1.008 |
| CORA | 91.70 | 0.204 |
| CITESEER | 91.42 | 0.210 |

Table 4: Link Prediction Results

# Graph classification

## Graph Statistics (10 points)

### Solution

|  |  |  |
| --- | --- | --- |
| 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

## Implementation (90 points)

### Solution

Graph MUTAG ENZYMES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | P | R | F1 | P | R | F1 |
| Mean-pooling | 83.1 | 88 | 85.32 | 32.17 | 40.04 | 32.17 |
| Max-pooling | 84 | 83 | 83 | 38 | 38.06 | 37.27 |
| Last-node pooling | 88.1 | 90.8 | 89.3 | 35.66 | 28.69 | 31.45 |

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

**References**