CS-502 Homework 2: Graph Neural Networks

October 6, 2023

Important Note: PyTorch Geometric is prohibited for use in this assignment (except for data loading). You are required to exclusively utilize PyTorch.

1 Introduction

In this homework assignment, you will explore various aspects of graph-based machine learning using the MUTAG dataset. The MUTAG dataset is a widely used graph classification dataset in the field of cheminformatics. It is designed for predicting the mutagenicity of chemical compounds, making it a crucial resource for toxicology and drug discovery research.

1.1 About the MUTAG Dataset

The MUTAG dataset consists of a collection of chemical compounds, where each compound is represented as a graph. In these graphs:

- Nodes represent atoms within the chemical compound, their features indicate the atom type.
- Edges represent chemical bonds between atoms, their features indicate the chemical bond type.
- The dataset is labeled, with each graph labeled as either mutagenic (positive class) or non-mutagenic (negative class).

The goal of this assignment is to develop and evaluate graph convolutional network models for the graph classification task. By predicting whether a chemical compound is mutagenic or non-mutagenic, you'll gain valuable insights into the applications of graph-based machine learning in cheminformatics.

1.2 Dataset Download Instructions

You may access the MUTAG dataset from the Hugging Face Datasets library via the following link:

https://huggingface.co/datasets/graphs-datasets/MUTAG

The dataset comprises a total of 188 small graphs, each representing a molecule.

Please note that the website provides only the training partition, which we will treat as the entire dataset. Additionally, it's important to highlight that the dataloader provided on the website is non-functional. Consequently, you will need to create your own custom dataloader for this dataset.

1.3 Assignment Structure

This assignment is divided into three parts, each focusing on different aspects of graph convolutional networks and network design. You will explore various graph convolution layers, customize your network architecture, tune hyperparameters, and investigate the impact of incorporating both node and edge features.

Let's begin by implementing different graph convolution layers in Part 1 of the assignment.

Part 1: Implementing Different Graph Convolution and Pooling Layers

In this part, you will implement different graph convolutional and Pooling layers.

1. Normal Convolution (Graph Convolution):

   Implement a Normal Convolution (Graph Convolution) layer for your GCN model. The Normal Convolution aggregates neighbor data without any specific attention mechanism. The equation for a Normal Convolution at layer l can be defined as follows:

   $$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_l \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l)}}{|N(v)|} + \mathbf{B}_l \mathbf{h}_v^{(l)}\right),$$

   where $\mathbf{h}_v^{(l)}$ is the embedding of node v at layer l, $N(v)$ is the set of neighbors of node v, σ is an activation function, and $\mathbf{W}_l$ and $\mathbf{B}_l$ are the weight matrices for layer l.

2. GraphSAGE (Customized Aggregation):

   Implement a GraphSAGE layer with a customized aggregation function for your GCN model, which could for example be a mean or LSTM aggregator. The equation for GraphSAGE at layer l with an arbitrary aggregator $\mathrm{AGG}(\cdot)$ can be defined as follows:

   $$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_l \cdot \mathrm{CONCAT}\left(\mathbf{h}_v^{(l)}, \mathrm{AGG}\left(\left\{\mathbf{h}_u^{(l)}, \forall u \in N(v)\right\}\right)\right)\right),$$

   where we use the same notation as above, with the added aggregation function $\mathrm{AGG}(\cdot)$ (e.g., mean aggregation as in the Normal Convolution), and the concatenation function $\mathrm{CONCAT}(\cdot)$.

Note: as you have seen in the lectures, we can use the adjacency matrix to write the above layer more easily. Therefore, feel free to transform the edge indices into an adjacency matrix if you need to.

3. Attention-based Convolution:

   Implement an Attention-based Convolution layer for your GCN model. Attention-based Convolution applies attention mechanisms to neighbor aggregation. The equation for Attention-based Convolution at layer l can be defined as follows:

$$\mathbf{h'}_v^{(l)} = \mathbf{W}_l \mathbf{h}_v^{(l)}$$

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u \in N(v) \cup \{v\}} a_{vu}^{(l)} \cdot \mathbf{h'}_u^{(l)}\right),$$

   where $a_{vu}^{(l)}$ is the normalized attention weights between node v and its neighbors node u as well as itself, see below how they are computed. $N(v) \cup \{v\}$ is the union of the neighbors of node v and itself,

The attention score e_{vu} between node v and its neighbor u is calculated as the dot product between a learnable weight vector and the concatenation of their features and passed through a LeakyReLU activation:

$$e_{vu} = \text{LeakyReLU}\left(\mathbf{S}^T \cdot \text{CONCAT}\left(\mathbf{h'}_v^{(l)}, \mathbf{h'}_u^{(l)}\right)\right),$$

where $\mathbf{S}$ is a learnable weight vector, shared amongst all nodes. Note that this attention score is applied to the transformed node embedding $\mathbf{h'}_v^{(l)}$.

After calculating the attention scores, we apply the softmax function to obtain normalized attention weights. These weights represent the importance of each neighbor node in relation to node v:

$$a_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v) \cup \{v\}} \exp(e_{vk})}$$

For more on attention-based graph convolutions, you can refer to [1].
As for Pooling, we will only consider global pooling strategies such as :

4. Mean Pooling :

   Given a graph with node features $\mathbf{X} \in \mathbb{R}^{N \times D}$, where N is the number of nodes and D is the feature dimension, mean pooling computes the graph-level representation $\mathbf{h}_{\text{global}}$ as the mean (average) of all node features:

$$\mathbf{h}_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}_i$$

   Here, $\mathbf{X}_i$ represents the feature vector of the ith node.

5. Max Pooling :

   Max pooling selects the maximum value from each feature dimension across all nodes to create the graph-level representation:

$$\mathbf{h}_{\text{global}}[d] = \max_{i=1}^{N} \mathbf{X}_i[d]$$

   for each feature dimension d.

As a side note, it's worth mentioning that there are more advanced graph pooling strategies. DiffPool [2] and SAGPool [3] are notable examples of hierarchical pooling techniques that offer advanced methods for graph coarsening and pooling.

2 Part 2: Custom Network Design with Node Features

In this part, you will design a custom neural network architecture that incorporates different types of graph convolutional layers (from Part 1). The steps include:

1. Custom Network Architecture: Implement a neural network with user-defined choices for the types and number of graph convolutional layers. For simplicity, the network's prediction head should only have one pooling layer (Mean or Max) and one Fully Connected Layer.

2. Data Partitioning: Split the dataset into training (70%), validation (15%), and test sets (15%) for consistent evaluation.

3. Hyperparameter Tuning: Experiment with different hyperparameters such as the number of layers, step size, and the type of convolutional layers (e.g., Normal, GraphSAGE, Attention-based). Train and evaluate multiple network configurations on the training and validation sets.

4. Performance Evaluation: Report the results of your experiments, comparing the performance of different network configurations on the validation set. Analyze which combination of hyperparameters works best and provide reasons for your choice. Report your final performances on the test set.

3 Part 3: Incorporating Edge Features

In this part, you will extend your custom network to incorporate both node and edge features and compare the results with Part 2.

5. Explain your strategy for incorporating edge features.

6. Modify your GNN models to utilize edge features in addition to node features for the node task.

7. Evaluate the performance of your models and compare the results with those from Part 2 to determine if using edge features improves performance.

Submission Guidelines

- Submit your code in a jupyter notebook format and make sure that all cells run smoothly.
- Submit a two-page report detailing your experiments and findings.
- Include any necessary comments and explanations in your code.
- Make sure your code is well-documented and organized.

References

[1] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. arXiv preprint arXiv:1710.10903.

[2] Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In Advances in neural information processing systems.

[3] Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In International Conference on Learning Representations (ICLR).

Grading Rubric

In this section, you will find the grading criteria and point allocation for this homework assignment. Each criterion is assigned a certain number of points, and your final grade will be based on the total points earned out of the maximum possible points.

We emphasize that code clarity holds significant importance, and as a result, it will account for 25% of the grading weight for coding questions.

Part 1: Implementing Different Graph Convolutions and Pooling (2.5 points)

(0.5 points) Implementation of Normal Graph Convolution.

(0.5 point) Implementation of GraphSAGE Convolution.

(1 point) Implementation of Attention-Based Graph Convolution.

(0.5 point) Implementation of Global Pooling (Mean or Max Pooling).

Part 2: Custom Network Design with Node Features (3 points)

(0.5 point) Custom network architecture design.

(0.5 point) Code for hyperparameter tuning (number of layers, step size, etc.).

(2 points) Experimental results analysis and insights.

Part 3: Incorporating Edge Features (5 points)

(2 points) Successful integration of edge features into the network (explain your strategy).

(1 point) Hyperparameter tuning considering edge features.

(2 points) Comparative analysis with Part 2 (node features only).

Analysis and Reporting (9.5 points)

(3 points) Clarity and completeness of the analysis section.

(4 points) Effective visualization (use your creativity).

(2.5 points) Insights and conclusions drawn from the experiments.