CSE 881 PROJECT REPORT

Node Classification Using GNN's

MAY 6, 2023

ADITYA JAIN
SAUMYA SHAH

Exploring Graph Neural Networks for Node Classification with GCN, GAT, and AGNN's

Abstract:

In this project, we investigated the performance of Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Attention-based Graph Neural Networks (AGNN) for node classification on a graph dataset with 2480 nodes and 1390 features. We extensively tuned hyperparameters and created an ensemble model using GCN and AGNN layers. Our results showed that the tuned 2-layer GCN architecture and the ensemble model achieved an average accuracy of 0.862 and 0.869 on the validation set using 5-fold cross-validation respectively.

Role of each team member:

Aditya's role was focused on exploring GCN and AGNN networks, performing hyperparameter tuning using grid search, random search, and Bayesian optimization. He also performed model ensemble with GCN and AGNN and contributed to the project report.

Saumya's role involved loading data, setting up the GAT network, and performing different hyperparameter tuning approaches. He helped visualize model performance while training and explored various performance metrics. He also contributed to the project report.

The solution:

We implemented multiple GNN architectures, including GCN, GAT, and AGNN, each with different numbers of layers. We observed that fewer layers performed better and had better accuracy, possibly due to insufficient training data. We started by testing the above three different architectures to compare their performance with a simple two-layer design. After testing these architectures, we decided to extensively tune the best-performing model (GCN) and applied tuning on a smaller search space. We used methods like grid search, random search, and Bayesian optimization to find hyperparameters. We also tried model ensemble by adding GCN and AGNN layers sequentially in the architecture, with more hyperparameters to tune. The hyperparameters used in our project were learning rate, weight decay, number of hidden nodes, optimizer type, activation function, and dropout rate.

Dataset and Preprocessing:

The dataset consists of 2480 nodes, 10100 edges, and 1390 features per node. There are 7 classes for node classification. We used the provided adjacency matrix, feature matrix, and list of labels for our models. We split the training data into train and validation sets, with 20% of the data (100 samples) used for validation.

Model Architectures:

- GCN:
  - 2-layer architecture with GCNConv layers
  - ReLU activation function
  - Dropout applied after the first layer.
- GAT and AGNN:
  - 2-layer architecture with GATConv layers
  - ReLU activation function
  - Dropout applied after the first layer.
- AGNN:
  - Linear layer followed by two AGNNConv layers and a linear layer.
  - ReLU activation function
  - Dropout applied after the first layer.
- Ensemble model:
  - A GCNConv layer followed by AGNNConv layer and a linear layer.
  - ReLU and leaky_relu activation function
  - Dropout applied after the first layer.

Results and Discussion:

- Before hyperparameter tuning, we achieved an accuracy of around 83%.
- After tuning and ensemble models, we obtained similar classification results with tuned 2-layer GCN architecture and a tuned ensemble model of GCN and AGNN convolution layers with an average accuracy of 86.9 % on the validation set using 5-fold cross-validation.
- Our ensembled GCN and AGNN architecture model had the best performance on the test set with 86.08 % accuracy.

The learned lessons and further improvements:

Throughout this project, we learned the importance of implementing different approaches, exploring attention networks, and understanding how weighted edges are used in prediction with attention networks. We explored how to perform tuning along with cross-validation and implemented different architectures and ensembled them. For further improvement, we can investigate tuning more parameters, exploring parallel ensemble rather than sequential, and tuning the hops parameter in the graph that accounts for n-hop neighbors for better embeddings. Additionally, we can investigate other optimization techniques or newer GNN architectures to further enhance our model's performance.