Disease Prediction via Graph Neural Networks for CS598 DL4H in Spring 2023

Chris Park
{hpark102}@illinois.edu

Group ID: 209
Paper ID: 153
Presentation link: https://youtu.be/53CgNJQeKdc
Code link: https://github.com/parkchrishj/GCN-Disease-Diagnosis

1 Introduction

The authors (Sun et al., 2020) highlight the challenge of predicting rare diseases due to performance drop with long sequences, requirement of large training data, and limited applicability to rare diseases, and propose a solution by incorporating external knowledge bases to augment the electronic medical records (EMR) data. The proposed model uses a graph convolutional network (GCN)-based approach to learn embedding for patients, diseases, and symptoms from both the medical concept graph and the patient record graph. The proposed model has several advantages, including the ability to capture complex relationships among diseases, symptoms, and patients in a graph structure, the efficient use of shared information across the graph, and the ability to generate embedding for previously unseen nodes during training.

The authors emphasize that their model is able to handle new patients without historical EMRs, making it more practical for real-life scenarios. The authors conduct extensive experiments on a real-world EMR dataset to demonstrate the effectiveness of their proposed model in disease prediction, including rare disease prediction. The results show that the proposed model outperforms existing methods and has the potential to significantly improve disease prediction accuracy. The proposed model has the potential to aid in the early diagnosis of rare diseases, leading to better patient care and outcomes.

2 Scope of reproducibility

Aggregating on the features of directly connected neighbors in a graph convolutional network (GCN)-based approach to learn embedding for patients, diseases, and symptoms significantly improve the prediction accuracy in disease and rare disease than GATs with attention mechanisms and GINs with neighborhood-based embedding.

This novel approach makes it an interesting and innovative study to reproduce and validate the claimed improvements in disease prediction accuracy. Reproducing and validating the results would provide valuable insights into the effectiveness of the proposed approach in comparison to existing methods. It could potentially contribute to the development of more accurate and effective prediction models for rare diseases, which are often challenging to diagnose and manage.

2.1 Addressed claims from the original paper

Clearly itemize the claims you are testing:

- The incorporation of both the medical knowledge graph and the patient graph in the GNN architecture improves the performance of disease prediction.
- The use of attention mechanisms in the GNN model helps to highlight the most relevant information in the medical knowledge graph and patient graph for predicting the disease.
- A graph encoder, which uses an aggregation scheme on the features of directly connected neighbors, is better than other GNN aggregators as it efficiently uses shared information across the graph and allows for embeddings to be generated for previously unseen nodes during training.

3 Methodology

With the authors' provided model on GitHub, I aimed to re-implement the approach from the paper description. Additional documentation to the code can be found in my GitHub page, https://github.com/parkchrishj/GCN-Disease-Diagnosis. The experiment did require a significant amount of computation resources and involved in multiple GPUs. Performing the training required 2199 seconds.

3.1 Model descriptions

The authors propose a new GNN-based framework for disease prediction that leverages both node and graph-level features. The framework consists of two types of graphs: the medical concept graph and the patient record graph. The medical concept graph represents the relationship between diseases and symptoms, while the patient record graph captures the relationship between patients and symptoms. Specifically, they use a multi-layer graph convolutional network (GCN) to learn node embeddings that encode the information in both graphs and learn representations of each node. The graph encoder model uses an aggregation scheme to calculate embeddings of each node in the graph. The aggregation function takes into account the first-order neighbors of a node and applies the same transformation across all nodes in the graph.

This approach makes efficient use of information shared across regions in the graph and allows embeddings to be generated for previously unseen nodes during training. Two distinct neighborhood information aggregators, namely the Graph Attention Networks (GATs) and Graph Isomorphic Networks (GINs) are explored through comparative studies. The learned embeddings of a patient's symptoms are used to predict the likelihood of them being diagnosed with a particular disease. They also introduce a graph attention mechanism to weight the importance of different features in the graph-level embedding. Finally, they use a fully connected layer to predict the disease label based on the learned embeddings. The framework is designed to perform rare disease prediction by evaluating the prediction performance exclusively on patients diagnosed with rare diseases.

3.2 Data descriptions

The paper has provided the necessary data in the .pkl data file format. The data is a real-world EMR patient clinical dataset from local hospitals. It contains 806 patients, while 451 among them were diagnosed with at least one rare disease. There are 71 rare diseases and 131 types of symptoms. Each patient has an average of 1.49 diagnosed diseases and have a maximum of 5 diseases at a time. Further explanation on data types of each data files are available in the following GitHub repository. https://github.com/parkchrishj/GCN-Disease-Diagnosis/blob/main/README.md

3.3 Hyperparameters

The paper studies the impact of three hyperparameters, namely the initialization embedding dimension, the output embedding dimension, and the aggregator layer size, on the overall disease prediction performance. The source of their values was from a grid search approach, where they varied each hyperparameter over a range of values and recorded the corresponding performance of the model.

For the initialization embedding dimension, the paper varied it in 500, 1,000, 2,000, 4,000, 6,000, 8,000, 10,000 and observed that the higher the dimension, the better the performance achieved. The most significant performance gain was observed at K = 2 and K = 3. When the dimension exceeded 2,000, the model's performance became stable.

For the output embedding dimension, the paper adjusted it in 50, 100, 200, 400, 600, 800, 1,000 and found a dramatic performance boost as the embedding dimension increased from 50 to 100. The performance increase became negligible when the dimension was over 100, indicating that the proposed model can preserve the node and structural information well with a relatively compact output dimension.

For the aggregator layer size, the paper varied it in 1, 2, 3, 4 and found that adding extra deep layers did not incur further performance gain. Setting the layer size to 1 was sufficient for the model to learn representative node embedding for disease prediction, while propagating more information with additional layers may infuse noise into node embedding and lead to inferior performance. The time cost increased when L increased from 1 to 4 because it took more computational steps to generate the embedding for each target node, but the proposed graph neural network was highly efficient with the layer size of 1 as the average running time per batch was less than 0.1 second.

3.4 Implementation

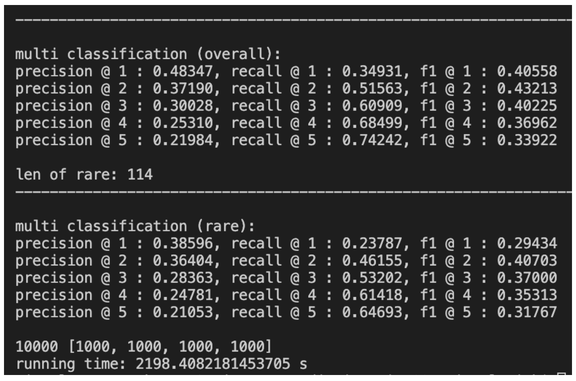
The paper has provided an existing code of the model including the encoders and aggregators. I used the available GitHub repository as reference and integrate it as part of this paper https://github.com/zhchs/Disease-Prediction-via-GCN. In my GitHub repository, a clear documentation of the code and the model is provided in comments of each file and in README.md file.

3.5 Computational requirements

I initially estimated in the proposal that the experiments likely require a significant amount of computational resources, potentially involving multiple GPUs and many hours or days of training time. The plan was to perform the training on a windows computer with 8-core processor, 3.90 GHz, and 16.0 GB of installed RAM. Instead, I used macOS with 2.2 GHz Quad-Core Intel Core i7 and 16GB of installed RAM. The proposed model's training required 2199 seconds of average runtime per epoch. For each run, the model undergoes 8,000 training epoch. Performing training for other models and ablation took around similar time. Altogether, the training did take many hours.

4 Results

The work support the claims listed in section 2.1. As shown in (Figure 1), Aggregating on the features of directly connected neighbors in a graph convolutional network (GCN)-based approach to learn embedding for patients, diseases, and symptoms significantly improve the prediction accuracy in disease and rare disease than GATs with attention mechanisms and GINs with neighborhood-based embedding.



```
----------------------------------------------------------------

multi classification (overall):
precision @ 1 : 0.48347, recall @ 1 : 0.34931, f1 @ 1 : 0.40558
precision @ 2 : 0.37190, recall @ 2 : 0.51563, f1 @ 2 : 0.43213
precision @ 3 : 0.30028, recall @ 3 : 0.60909, f1 @ 3 : 0.40225
precision @ 4 : 0.25310, recall @ 4 : 0.68499, f1 @ 4 : 0.36962
precision @ 5 : 0.21984, recall @ 5 : 0.74242, f1 @ 5 : 0.33922

len of rare: 114
----------------------------------------------------------------

multi classification (rare):
precision @ 1 : 0.38596, recall @ 1 : 0.23787, f1 @ 1 : 0.29434
precision @ 2 : 0.36404, recall @ 2 : 0.46155, f1 @ 2 : 0.40703
precision @ 3 : 0.28363, recall @ 3 : 0.53202, f1 @ 3 : 0.37000
precision @ 4 : 0.24781, recall @ 4 : 0.61418, f1 @ 4 : 0.35313
precision @ 5 : 0.21053, recall @ 5 : 0.64693, f1 @ 5 : 0.31767

10000 [1000, 1000, 1000, 1000]
running time: 2198.4082181453705 s
```

Figure 1: Multi classification of overall and rare disease prediction accuracy scored in precision, recall, and F1.

4.1 Analyses

We compare our model with three well-established classifiers, namely Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF), as well as four state-of-the-art graph embedding-based models, which are DeepWalk, LINE, SDNE, and Struc2Vec. We evaluate our model in terms of performance on both general disease prediction and rare disease prediction tasks. For our model, we set the hyperparameters including learning rate, batch size, aggregator layer size, and dimensions of initialization and output embeddings. In classic baselines, we take symptoms in each patient's EMRs as the input features for classification. In all graph embedding-based peer methods, we transform the EMRs into a patient record graph as their input. We randomly split the patients in the proprietary EMR dataset with a ratio of 7:3 for training and evaluation, respectively. We utilize recall, precision, and F1 score of the top-K diseases in each patient's prediction result as evaluation metrics, where K is chosen from 1, 2, 3, 4, 5 based on the average and maximum number of diagnosed diseases in the dataset.

Overall, our model outperforms all state-of-the-art baselines by a significant margin in most cases (K = 2, 3, 4, 5) in terms of recall, precision, and F1 score. This verifies the effectiveness of our model in inductively representing a disease by aggregating the learned representations of its neighbor nodes (i.e., diseases). The general trend shows an increasing recall and a decreasing precision with an increasing K, and all models achieved the highest F1 score when K = 2 or K = 3. However, for the case of K = 1, SVM performs the best, but our model still performs competitively. Further details of the experimental results can be found in Table II. The source code of our model has been released for easy implementation at the provided GitHub repository.

4.2 Plans

Any additional ablations I performed are as follows:

- Varying the aggregation function: In the current paper, the mean aggregator is used for information aggregation at each layer. However, other aggregation functions like max pooling, RNNs, etc. can also be used. Comparing the performance of the model with different aggregation functions could give insights into which aggregation function is more suitable for the given task.

- Using different projection weights: In the current paper, a learnable projection weight W is used to align the contexts of different types of nodes. Using different projection weights, or even not using any projection weight at all, could provide insights into the importance of context alignment in the proposed framework.

- Varying the size of the embedding space: In the current paper, the embedding space size is set to d. It would be interesting to investigate the effect of varying the size of the embedding space on the performance of the model. A larger embedding space may allow for more complex representations, but may also require more training data.

- Varying the number of layers: In the current paper, the authors use L layers for information propagation. However, it would be interesting to investigate how the performance of the model changes with varying number of layers. Too many layers may lead to over-fitting, while too few may not capture enough information for effective disease prediction

4.3 Result 1

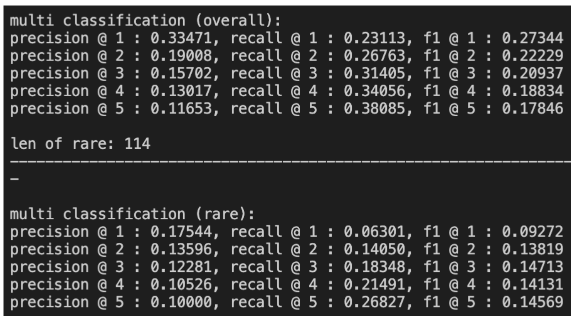
Through many reproduction of the GCN, the precision, recall, and F1 scores are within 15% of each other and to the result the authors have shared. There were instances of GCN results having a lower precision, recall, or F1 score than other models where K = 1,4,5 as the authors have implied while at K = 2,3, GCN prediction performance was significantly the best. (Figure 1) is the resulting disease prediction performance of the GCN architecture incorporating both the medical knowledge graph and the patient graph.

4.4 Result 2

The attention mechanism weight is evenly distributed. To demonstrate the improvement in disease prediction, The weight of attention mechanism was modified to 0.2 and 0.8, resulting in lower precision, recall, and F1 score. (Figure 2)

4.5 Result 3

The author provides GCN, GAT, and GIN encoders for reproducibility. By changing the parameters in the model to utilize the GAT encoder, (Figure 3) displays the result of GAT. GCN is significantly more efficient in training, taking only 0.26 average batch time. Meanwhile, GAT takes 16.57 average batch time. Epoch time had to be reduced to 80 for the GAT in order to complete it in 30 minutes.

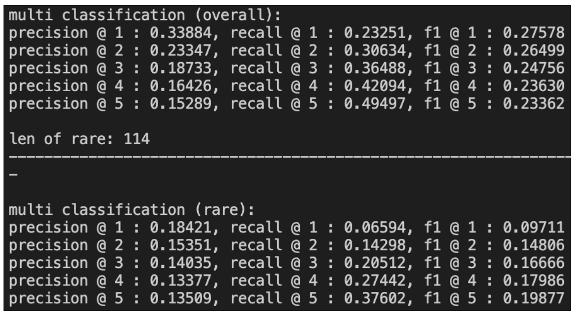


```
multi classification (overall):
precision @ 1 : 0.33471, recall @ 1 : 0.23113, f1 @ 1 : 0.27344
precision @ 2 : 0.19008, recall @ 2 : 0.26763, f1 @ 2 : 0.22229
precision @ 3 : 0.15702, recall @ 3 : 0.31405, f1 @ 3 : 0.20937
precision @ 4 : 0.13017, recall @ 4 : 0.34056, f1 @ 4 : 0.18834
precision @ 5 : 0.11653, recall @ 5 : 0.38085, f1 @ 5 : 0.17846

len of rare: 114
----------------------------------------------------------------
-

multi classification (rare):
precision @ 1 : 0.17544, recall @ 1 : 0.06301, f1 @ 1 : 0.09272
precision @ 2 : 0.13596, recall @ 2 : 0.14050, f1 @ 2 : 0.13819
precision @ 3 : 0.12281, recall @ 3 : 0.18348, f1 @ 3 : 0.14713
precision @ 4 : 0.10526, recall @ 4 : 0.21491, f1 @ 4 : 0.14131
precision @ 5 : 0.10000, recall @ 5 : 0.26827, f1 @ 5 : 0.14569
```

Figure 2: After the weight of attention mechanism was modified to 0.2 and 0.8, multi classification of overall and rare disease prediction accuracy scored in precision, recall, and F1.



```
multi classification (overall):
precision @ 1 : 0.33884, recall @ 1 : 0.23251, f1 @ 1 : 0.27578
precision @ 2 : 0.23347, recall @ 2 : 0.30634, f1 @ 2 : 0.26499
precision @ 3 : 0.18733, recall @ 3 : 0.36488, f1 @ 3 : 0.24756
precision @ 4 : 0.16426, recall @ 4 : 0.42094, f1 @ 4 : 0.23630
precision @ 5 : 0.15289, recall @ 5 : 0.49497, f1 @ 5 : 0.23362

len of rare: 114
----------------------------------------------------------------
-

multi classification (rare):
precision @ 1 : 0.18421, recall @ 1 : 0.06594, f1 @ 1 : 0.09711
precision @ 2 : 0.15351, recall @ 2 : 0.14298, f1 @ 2 : 0.14806
precision @ 3 : 0.14035, recall @ 3 : 0.20512, f1 @ 3 : 0.16666
precision @ 4 : 0.13377, recall @ 4 : 0.27442, f1 @ 4 : 0.17986
precision @ 5 : 0.13509, recall @ 5 : 0.37602, f1 @ 5 : 0.19877
```

Figure 3: After the weight of attention mechanism was modified to 0.2 and 0.8, multi classification of overall and rare disease prediction accuracy scored in precision, recall, and F1.

4.6 Additional results not present in the original paper

Some of the ablations I performed had some effect to the result. For instance, changing the layer size to 0.05 from 1.0 have decreased the scores. Eliminating the projection weight by changing the weight to false did not significantly affect the GCN model. Varying the size of the embedding space to a lower size or the aggregation function from mean aggregator to a regular aggregator result in low disease prediction performance.

5 Discussion

The larger implications of the experimental results describes the authors' proposed model advantages in the use of GCN based approach to learn embedding for patients, diseases, and symptoms from both the medical concept graph and the patient record graph. The original paper was reproducible as the authors have provided the code. Furthermore, I could perform the authors' claims and my planned ablations. Gathering all these results, I am confident in my judgement that I feel the evidence

I got from running the code supports the claims of the paper. The strengths in my approach is the well-put documentation of the reproducibility. It facilitated the modification, ablations, and claim testing. Each run on the modification took significant amount of time, so the required epoch of 8,000 had to be reduced to 80 in certain cases like the GAT model. With only 1% of the normal training epoch, the weakness in this approach is that lower training epoch would have an effect on lower score.

5.1 What was easy

It was easy to run the author's code, re-implement their method based on the description in the paper. It did take couple of hours overall. The code had some models other than the GCN to compare the result against each other. Certain models take 100 folds longer to train than the original model, indicating the efficiency in the proposed model. The work around longer training is to reduce the epoch time from 8,000 to a lower number deemed necessary.

5.2 What was difficult

The difficulty in the reproduction study is the access to the data. The data are of the type file .pkl, so it was difficult to see the raw data and modify it to support the authors' claim. The training was difficult to modify as the labels, medical concept graph, and patient record graphs were required. It was also difficult to prove the claim that the GCN can perform better with those inputs as it was difficult to reproduce the model without those inputs.

5.3 Recommendations for reproducibility

Each 8,000 training epoch takes at least 30 minutes, so I would suggest reducing the training epoch. In a worst case scenario, running a GAT model with the same training epoch would take 480 minutes to complete. If there was a reproducibility method with smaller training epoch, this would reduce the time to complete it.

6 Communication with original authors

In the original paper, the authors' emails were listed along with their background. For instance, Zhenchao Sun and Lizhen Cui are with the School of Software, Shandong University, Jinan 250100, China (e-mail: zhenchao.sun@mail.sdu.edu.cn; clz@sdu.edu.cn). I emailed the first four authors in the list with couple of questions, such as Utils.RARE_INFO dependency, along with the full report for their feedback. I am awaiting for a response.

References

Zhenchao Sun, Hongzhi Yin, Hongxu Chen, Tong Chen, Lizhen Cui, and Fan Yang. 2020. Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3):818–826.