Classifying Gene Modules Using GNNs

Deep Learning - 0460211

Yaniv Slor Futterman & Jonathan Israel

Introduction

Project Goal:

The goal of the project is to successfully classify gene modules associated only with a subset of the patients in a gene-gene correlation graph. We do this by using graph neural networks (GNNs).

Motivation

Cancer is not a single disease. In fact most cancers have a variety of subtypes, each exhibiting different biological pathways that determine their lethality and treatment response. This means that if we want to achieve the goal of personalized medicine, we must first be able to identify and differentiate between the different cancer subtypes.

Previous Work

Previous work used traditional graphical clustering algorithms to identify gene modules in gene-gene correlation graphs, but this was done on patient-specific graphs, meaning the module was associated only with a single patient. This could be useful in certain cases, but did not generalize well when dealing with multiple patients, which is often the case. We try here for the first time to expand the problem to multidimensional graphs containing data from all patients.

Dataset

The graphs

The dataset contains 4000 graphs, generated using the graph simulator built in Prof. Shai Shen-Orr's lab. Each graph contains 500 nodes (genes) and 100 patients, of which roughly 30% are "sick", meaning they exhibit the target gene module in the graph. The graph structures are different every time and are based follow a power-law edge distribution.

Each edge of the graph contains a feature vector with length equal to the number of patients plus one. This edge feature vector describes how much each patient contributed to the correlation of the two genes. The last value is the correlation measure of the edge. This means that most of the information is stored on the edges of the graphs as opposed to the nodes, which is uncommon.

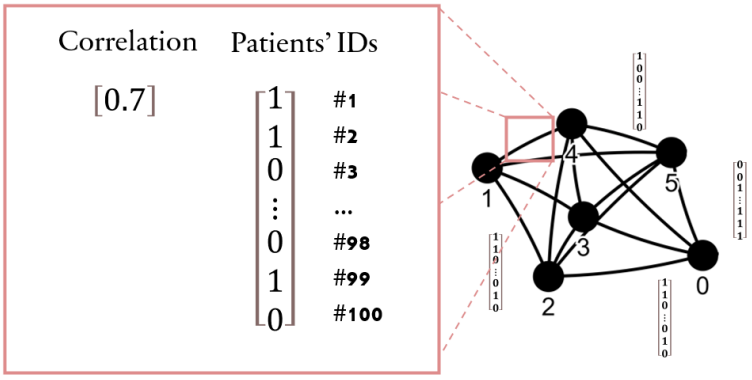

Image 1: Edge feature vectors based on individual patient contributions to the correlation

Difficulty Levels

The graphs vary in 4 levels of difficulty (1000 graphs for each level). This impacts the strength of the signal in the graph. The difficulty effects the probabilities for "sick" and "healthy" patients contributing to the edges of the target module vs those of the other modules in the graph, as well as the strength of the contribution from "sick" vs "healthy" patients.

Training and Testing

The training was done on 3600 out of the 4000 original graphs.
The testing was done on the remaining 400 of the original graphs, as well as graphs generated outside the dataset with varying #nodes (400, 600 nodes).

Method

Permutation Invariant Edge Embedding

The "sick" patients for every graph are different, meaning their indices on the edge features changes for each graph. This is problematic when working with GNNs, since the message passing function uses either a weighted matrix or a convolution kernel to pass the messages (both aren't invariant to permutation). This means the model assumes each value in the feature vector has a consistent meaning between all graphs, which is not the case.

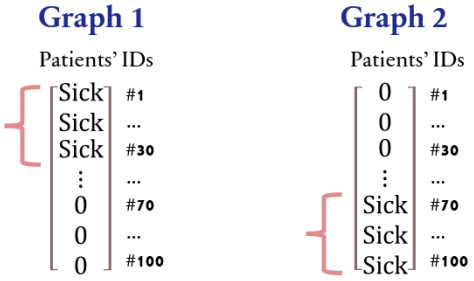

Image 2: "Sick" patient location varies between graphs

To address this, we used our own edge embeddings. We first define a patient vector as the set of contributions a patients makes to all the edges in the graph. We then rank the patients based on their pairwise dot-product with all other patients..

Reordering patients by rank ensures the edge feature vector has a meaningful and consistent order across different graphs.

The edge embedding was found to be crucial for model performance as we describe in the results section.

Initial Node Embedding:

We compared different methods for initial node embeddings. We compared using node embeddings based on the spectral eigenvectors of the graph Laplacian, a weighted average of the edge feature vectors of the node (weighted by the correlation of the edges), and an unweighted average of the edge feature vectors of the node. We compared the initializations using Optuna and found the weighted sum yielded improved results, so it was used moving forward.

GAT Layers

For the message passing layer we used the GATv2Conv layer. This layer was chosen because it uses attention and takes into account the edge features, which as mentioned is crucial for proper node classification.

In each message pass, the node x_i is updated to x_i' as a weighted sum of its neighboring nodes as follows:

$$\mathbf{x}_i' = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{i,j} \mathbf{\Theta}_t \mathbf{x}_j,$$

Image 3: Message passing of GATv2Conv layer

The matrix θ_t is the learned message passing matrix, $\alpha_{i,j}$ is the attention weight, and we only sum over the neighboring nodes x_j of the current node.

The attention weight is calculated as follows:

$$\alpha_{i,j} = \frac{\exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\mathbf{\Theta}_s \mathbf{x}_i + \mathbf{\Theta}_t \mathbf{x}_j + \mathbf{\Theta}_e \mathbf{e}_{i,j}\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\mathbf{\Theta}_s \mathbf{x}_i + \mathbf{\Theta}_t \mathbf{x}_k + \mathbf{\Theta}_e \mathbf{e}_{i,k}]\right)\right)}.$$

Image 4: Attention weight of GATv2Conv layer

As we can see, we use self-attention with softmax. $\theta_t, \theta_s, \theta_e$ are the learned transformation matrices for the query and the two keys (the neighboring node, and the edge between them).

The difference lies in the similarity score between the query and the keys. We take the sum of all three, pass it through $LeakyReLU$ and then take the dot product of this vector with a learned vector a. This implementation allows for the easy addition of the edge feature vectors into the attention weight.

Architecture

We found our model architecture using Optuna for hyperparameter optimization. We compared different hidden-dimensions, number of attention heads, number of hidden layers, dropout rates and more. The final architecture is given below:

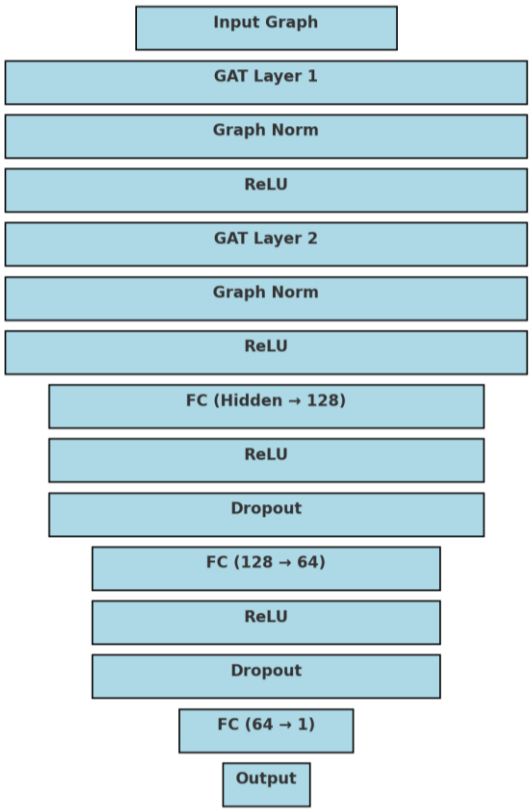

Image 5: Final model architecture

The final result is a scalar describing the confidence of the model that the current node is part of the target gene module.

Results

Hyperparameter Optimization

As mentioned above, we used Optuna and Wandb to find and visualize the optimal hyperparameters over the validation set.

Here is a visualization of the epoch loss over many model runs:

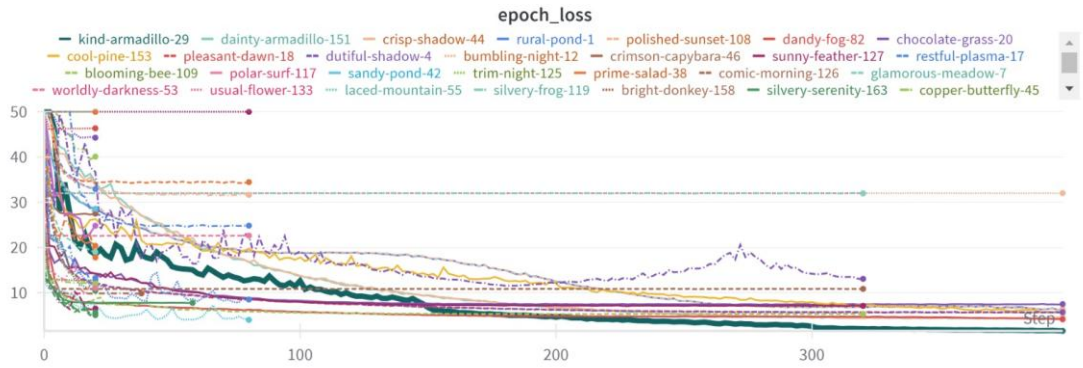

Image 6: Epoch loss of various models with Optuna for hyperparameter optimization

There were two final hyperparameter configurations found:

Hyperparameter	**Config 1**	**Config 2**
Batch Size	1	1
Dropout	0.265	0.368
Attention Heads	8	8
Hidden Dimension	64	64
Initial Node Embedding	Weighted Sum	Weighted Sum
Loss Function	BCE with Logits	BCE with Logits
Learning Rate	0.024	0.001
Optimizer	AdamW	Adam
Positive Weight Ratio	18	17
Weight Decay	0.0003	0.0005
Schedular	Exponential	Cosine Annealing
Schedular Learning Rate	0.0001	0.00001
Schedular Gamma (Exp)	0.93045	-
Schedular min Learning Rate	-	0.000001
Schedular Max Decay Period	-	85

Moving forward we used configuration 1.

Edge Embedding Comparison

We compare the result of the model using optimal hyperparameters on the validation set with our custom-made edge embedding vs the original unaltered edges:

Image 7: validation f1 score over-time of models using custom edge embedding (in green) vs standard edges (in yellow)

The final results were:

Edge Embedding	**Validation f1 Score**
Custom Edge Embedding	**0.77**
Standard Edges	0.19

As we can see, the edge embedding was crucial for proper node classification.

Graph Size Results

After training the model with the optimal hyperparameters on the original dataset of 3600 train graphs for 250 epochs, we tested it on graphs of different sizes, 400 graphs for every size (1200 graphs in total). The results for different sizes:

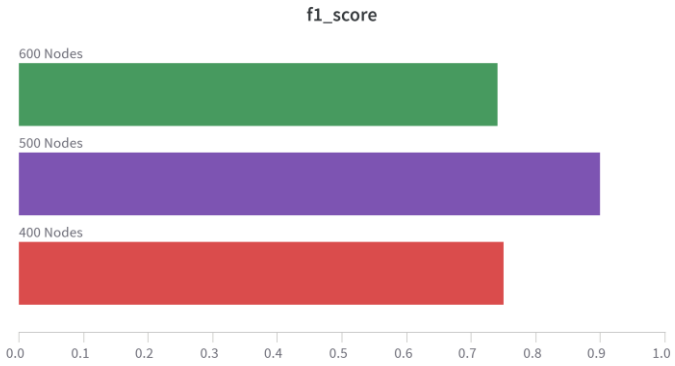

Image 8: f1 test scores over varying graph sizes

Final Results

Using the trained model as described above, the results for graphs of varying #nodes:

# Nodes in graph	400	500	600
Test F1 score	0.7493	0.8958	0.7401
Test Precision	0.8149	0.8955	0.7961
Test Recall	0.6935	0.8962	0.6916

Conclusion and Future Work

In conclusion, we have designed and trained a GNN model capable of correctly classifying gene modules associated with a subgroup of the patients ("sick") within a gene-gene correlation graph.

Yet, we must address the limitations of the model:

During the project we worked on relatively small graphs (a few hundred nodes per graph), containing a single target gene module.

In the future we would like to explore graphs containing multiple gene modules, as well as unsupervised or self-supervised methods for the module classification, or custom message passing functions that are invariant to the patient order.

Ethics Statement

Introduction

Student names: Yaniv Slor Futterman & Jonathan Israel

Project Title: Identifying Subgroup-Specific Genetic Modules in Gene-Gene Correlation Networks Using Graph Neural Networks

Describe your project: We use GNNs to find gene modules within gene-gene correlation networks that are associated only with a subset of the patients. Such gene modules act as biomarkers for different subtypes of cancer, and once found can improve diagnosis and accelerate drug discovery for personalized medicine designed specifically for each subtype.

Stakeholders

The three main stakeholders that will be affected by the project:

Medical Researchers & Clinicians – They rely on genetic biomarkers to improve cancer diagnosis, treatment planning, and personalized medicine.

Biotech & Pharmaceutical Companies – They use genetic insights to develop targeted therapies and accelerate drug discovery for specific cancer subtypes.

Patients & Patient Advocacy Groups – Patients benefit from more accurate diagnoses and tailored treatments, improving their chances of effective therapy.

Explanation for Each Stakeholder

Medical Researchers & Clinicians – "Our project uses Graph Neural Networks to identify gene modules that are linked to specific cancer subtypes. This means we can pinpoint genetic patterns that are relevant only to certain patient groups, enabling you to refine diagnostic criteria and personalize treatment strategies based on precise molecular signatures."

Biotech & Pharmaceutical Companies –"By uncovering genetic modules unique to different cancer subtypes, our approach helps you identify potential drug targets more efficiently. This accelerates the development of personalized therapies, reduces trial-and-error in drug discovery, and enhances precision medicine approaches."

Patients & Patient Advocacy Groups – "This research aims to improve cancer treatment by finding genetic patterns that help classify patients into subgroups based on how their cancer behaves. With this knowledge, doctors can prescribe treatments tailored to each patient's genetic profile, leading to better outcomes and fewer side effects."

Responsibility for giving the explanation to each stakeholder

Medical Researchers & Clinicians – The principal investigator (PI) or lead bioinformatician should present findings at medical conferences, publish in scientific journals, and collaborate with hospitals to integrate insights into clinical practice.

Biotech & Pharmaceutical Companies – A technology transfer officer, industry liaison, or research commercialization expert should communicate with biotech firms, presenting the potential applications and facilitating partnerships for drug development.

Patients & Patient Advocacy Groups – A science communicator, patient outreach specialist, or clinician involved in the study should explain the findings in accessible language through patient forums, advocacy groups, and public science communication platforms.

Reflection on the AI Answers

The responses to the different stakeholders were clear and concise, but they did not offer any ethical consideration, especially regarding the patients.

The response to the patients should have discussed the possible issues with patient privacy, discussing ways to ensure the genetic data of the patients remains anonymous and secure.

Furthermore, we must acknowledge that the use of such a treatment could put pressure on patients to have their genomes sequenced, which some might prefer to avoid for various reasons.

The response should have addressed this and offered possible solutions such as partial genome sequencing or single cell RNA sequencing which does not sequence the entire genome, rather the mRNA of a single cell at a specific time.