Machine Learning for Molecular Engineering
3/7/10/20.C01 (U) 3/7/10/20.C51 (G)
Spring 2025

Problem Set #3

Date: March 27, 2025
Due: April 23, 2025 @ 3pm

Instructions

- **Important:** This problem set requires a GPU. Before you start in Google Colab, find `Notebook Settings` under the `Edit` menu. In the `Hardware accelerator` drop-down, select a GPU as your hardware accelerator. Make sure you are using Colab Pro (talk to TAs about this, if needed) and train the model using a deep learning GPU (A100 preferably).
- Please submit your solution on Canvas as an IPython (Jupyter) Notebook file, `*.ipynb`. Your submission should contain *all* of the code that is needed to fully answer the questions in this problem set when executed from top to bottom and should run without errors (unless intentional). Please ensure that your plots are visible in the output file and that you are printing any additional comments or variables as needed. Commenting your code in-line and adding any additional comments in markdown (as "Text cells") will help the grader understand your approach and award partial credit.
- Note that collaboration between students is encouraged, however every student must be responsible for all the work in their individual submission. I.e., discussing solutions and debugging together should be considered fruitful teamwork, however physically taking the keyboard of another student and directly entering code for their submission would be intellectually dishonest and as such is prohibited. You should first read through and attempt every problem before discussing with others. Please list the names of all collaborators at the bottom of your .ipynb submission file.
- To get started, open your Google Colab or Jupyter notebook starting from the problem set template file here. You will need to use the data files here.
- Questions that require coding are highlighted throughout the PSET. Others just require you to run the relevant cells and answer the questions in the markdown cells provided at the end of the relevant code cells. Although there is extensive code in this exercise, you are not expected to understand every single line of it. The questions we ask in each section are meant to guide your understanding of the code. Our emphasis is on your understanding of why and how GNNs are useful to predict Synthetic Lethality.

Background on Synthetic Lethality

- Two genes are **synthetic lethal (SL)** if mutations in both of them reduce cell viability and/or lead to cell death, but mutations in either of them individually have minimal effects.
- SL is particularly relevant in the context of human cancers, and identifying SL gene pairs is essential for developing **anti-cancer therapeutics and drugs.**

- There are many **biological factors (a.k.a. SL-related factors)** that can explain why a gene pair is synthetic lethal, including whether or not the two genes interact or share a molecular function, among many others. There are databases with annotated and curated information about the nature of the interaction between approximately 10,000 human genes, namely SL relationships, such as the SynLethDB Database.
- In this exercise, you will leverage published work [1] to **train a Graph Neural Network (GNN) for prediction of SL interactions.** In the GNN field, this is known as "link prediction." This is a **supervised classification problem** where either a gene pair is synthetic lethal (SL) or not, and you will use a GNN to make predictions on human genes.

Part 1: Constructing Synthetic Lethality and Knowledge Graphs (10 points)

When predicting SL using a GNN, there are two complementary ways of representing data (i.e., two types of graphs - **SL and Knowledge graphs**), and both graphs are input to the model.

1. Modeling SL interactions between gene pairs using a **SL graph** $S = (V, E)$, where nodes V correspond to a set of n genes and edges E correspond to a SL relationship. If there is an edge between two genes, the value of the adjacency matrix $A_{1,2}$ ($\mathbb{R}^{n \times n}$) between two genes v_1 and v_2 is 1; otherwise, it is 0.

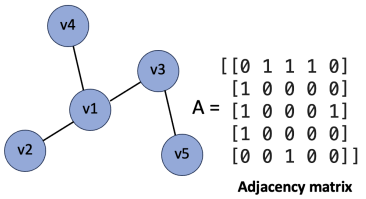

Figure 1: Synthetic Lethality graph (image source)

2. Modeling the relationships between genes in a **Knowledge Graph (KG)** $G = (V_e, E_r)$, where nodes V_e correspond to a set of gene entities e and edges E_r correspond to a set of SL-related factors/relationships r. Understanding what biological factors cause SL is a key part of predicting SL (Table 2 in [1]). Each KG can be represented as $G = \{(h, t, r)|h, t \in V_e, r \in G_r\}$. The (h, t, r) triple refers to head (equivalent to nodes in the SL graph) and tail (equivalent to edges in the SL graph) entities, and relationships between heads and tails, respectively.

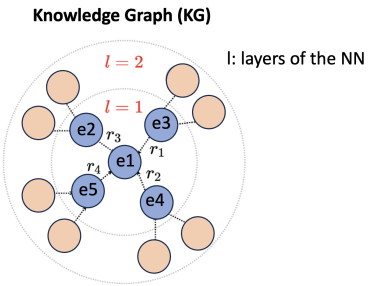

Figure 2: Knowledge Graph (image source)

Background on GNNs to predict SL

- Each layer in a GNN operates on graph-structured data in two steps: a **message** step and an **update** step. In this example, the message step takes information from connected nodes in the KG, and the update function transforms these parameterized messages to update the features (gene and entity embeddings) learned for each node in both the KG and SL graphs. Information from the most relevant neighboring nodes needs to be **aggregated** during message passing. Aggregated information comes from neighboring nodes and is performed over multiple iterations of message passing (a.k.a hops).

- After multiple convolutional layers, each node receives a parameterized embedding. The number of layers in the neural network (NN) is chosen based on the dimensions of the graph (Figure 2 - Knowledge Graph, above). A larger number of layers allows getting information about entities that are farther away and potentially improve model performance (as long as they do not exceed the size of the graph). Complex operations are required to map all the node embeddings onto a predicted scalar value. The most important function is the **softmax function** that takes this predicted scalar value and calculates the **probability of gene pair being SL.** Information from the KG is hence used for predicting edges in the SL graph.

  Key aspects of the model:

  1. **Factor modeling**: Modeling different SL-related factors to reflect how they can result in an SL interaction. Factor embeddings are based on the biological relationships defined in KG (Figure 2).
  2. Genes represented in the SL graph are input to the **Knowledge Graph Message Aggregation Graph Convolutional Network (GCN)**, consisting of multiple layers that learn embeddings for a gene at the entity level, leveraging information from the KG.
  3. The **Factor-based Message Aggregation Graph Attention Network (GAN)** updates these embeddings to generate new ones using information about preferences for different SL-related factors encoded in the KG.

- In both cases, embeddings are created based on features learned from neighboring nodes (hence the relevance of a graph structure!), with **each neighbor contributing differently to a relationship**. These embeddings are then used to make link predictions on SL graphs (i.e., is a gene pair SL?) (Figure 3).

- Given how in the KG, **two entities can be related through different relationships**, this model leverages attention mechanisms to understand which factors are the most relevant for a given SL interaction. Attention will be covered in detail in future lectures. All the information/context required to complete this PSET is provided here and in the questions below.

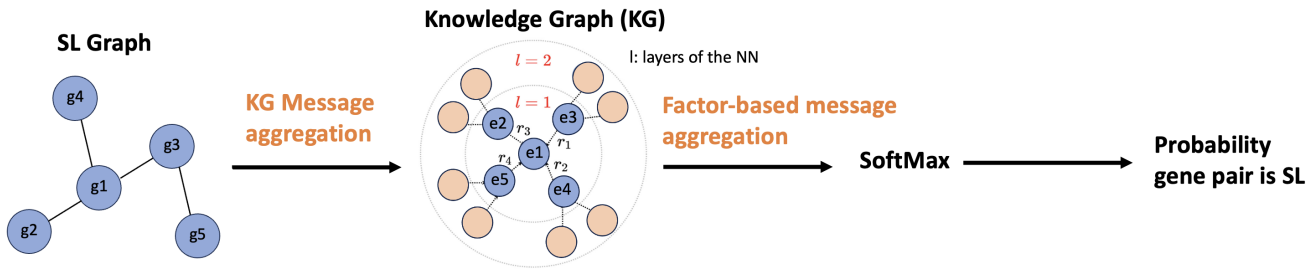

Figure 3: Overview of GNN to predict SL (image source)

Part 2: Training a GNN to predict SL gene pairs (25 points)

Part 2.1 Build Datasets and DataLoaders in PyTorch

You will start by constructing your dataset and loader, with `torch.utils.data.Dataset` and `torch.utils.data.DataLoader`. Construct train and validation data loader with using the train and validation datasets. Note the relevant hyperparameters have been defined in the **class ArgsConfig**.

Part 2.2 Define SL Model

Despite the extensive code shown below, you are not expected to understand every single line of it. The questions in each section are meant to guide your understanding of the code. Our emphasis is on your understanding of why and how GNNs are useful in the context of predicting SL. This model includes 3 classes: **Graph Attention Network (GAT) Class**, **Aggregator Class**, and **GraphConv Class**. The final model that calls all classes is named **SLModel**.

Part 2.3 Instantiate SL Model and implement functions for training

Here you will instantiate the model, move the model to the relevant device (cuda) as well as define the appropriate loss function (remember this is a classification task) and optimizer parameters. As in previous PSETs, you will be using the Adam optimizer to train the model. Once again, note the relevant hyperparameters have been defined in the **class ArgsConfig**.

Part 2.4 Training and evaluate the model

Adapt the code shown in the Colab notebook to loop over the training and validation sets. Make sure to save the training and validation losses in the relevant lists and compute the relevant metrics during validation. Note that the loss being minimized during model training is a summed loss, including the Binary Cross-Entropy (BCE) loss, a regularization (L2) loss, and a correlation loss based on a distance metric (the latter encourages the model to learn relationships between nodes that accurately reflect the graph structure).

During validation, we do not call the optimizer because the validation dataset is not used for training. You can make sure this is the case by calling `model.eval()` and turn on the `torch.no_grad()` context manager at the beginning of your validation function (we have implemented that for you). However, we need to call `model.train()` to turn the gradients and optimizer updates back during

training.

Record the average train and validation loss for each epoch and plot these on a single graph. During validation, you will also compute the **Area Under the Curve (AUC), Precision, Recall, and F1 scores** using functions from sklearn.metrics. Finally, comment on model performance

Part 3: Investigating one SL gene pair of interest (5 points)

As hinted at in the background, the SynLethDB Database contains information about human SL gene pairs, most of them relevant in human cancers. One of the gene pairs in this study includes HOXC11, a transcription factor involved in lung and colorectal cancer, and KRAS, a well-known oncogene (i.e., a gene that underwent mutations promoting unregulated cell growth and division). Look up the entry for HOXC11 in the SynLethDB Database - what SL interactions does it establish with other genes? Would you expect KRAS inhibitors to be particularly effective in cancer cells bearing loss of function mutations in HOXC11? Explain your answer.

References

[1] Li, Y. *et al.* Predicting synthetic lethality in human genes using a graph neural network. *Bioinformatics* **39** (2023).