COL 761 : Data Mining Spring 2025

Assignment 1

Total Points : 100 Due Date: 11:59 pm, Monday, Feb 10 2025

Daksh Dhaker (2022CS51264)
Parth Verma (2022CS11936)
Umang Tripathi (2022CS51134)

3) Part 3: Graph classification (70 marks, competitive)

3.1) Introduction

Graph classification is a fundamental problem in machine learning, widely applied in bioinformatics, chemoinformatics, and social networks. In this assignment, we focus on classifying molecular graphs by identifying discriminative subgraphs and representing graphs as binary feature vectors.

3.2) Problem Statement

The goal is to classify molecular graphs by learning a function F(G) that maps a given graph G to a label L. This involves:

- Identifying discriminative subgraphs
- Constructing binary feature vectors
- Using machine learning models to classify graphs based on extracted features

The dataset consists of two molecular datasets for binary classification, with evaluation done on a third hidden dataset.

3.3) Dataset Overview

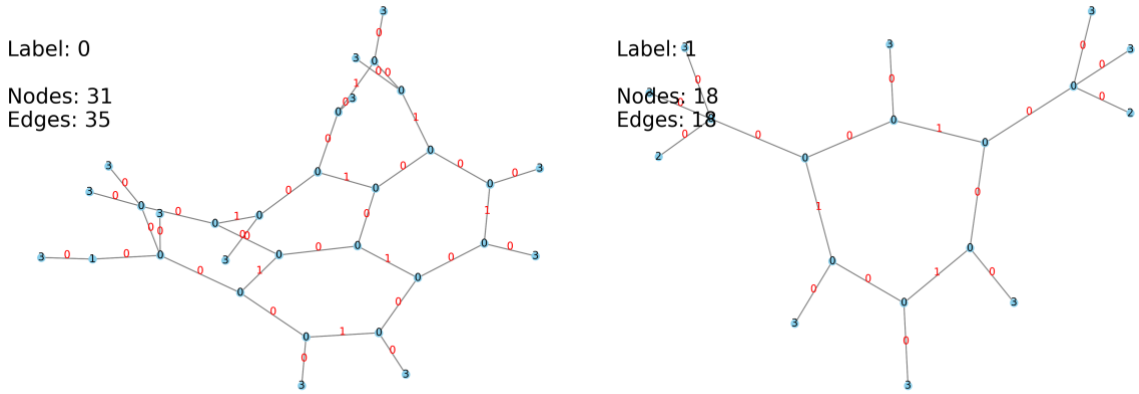
The dataset provided for this graph classification task consists of **molecular graphs**, which are structured representations of molecules where atoms are represented as **nodes** and chemical bonds as **edges**. These graphs are stored in a text-based format (.txt) and contain information on the graph structure, node labels (representing different atom types), and edge labels (representing different bond types). See *(Figure 1)* for data visualization, Other graphs visuals for Mutagenicity can be found on this link.

This dataset is used for **binary classification**, meaning that each graph is assigned a label of either 0 or 1. The objective is to identify key subgraph structures that are most indicative of the classification labels.

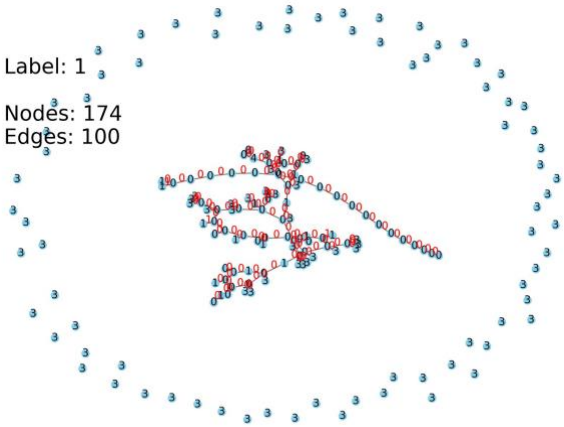
Dataset Size and Hidden Test Set

- The dataset consists of **two known training datasets** and **one hidden test dataset**.

- The number of graphs in the dataset varies, but each dataset contains **several hundred molecular graphs.**
- The training set is used to extract **discriminative subgraphs**, which help in classifying unseen test data.



(Figure 3.1. Mutagenicity graph No. 16 and 17)



(Figure 3.2. This is graph No. 76 from Mutagenicity, This has a bunch of node (3) which do not have any connected to them)

3.4) Approach Overview

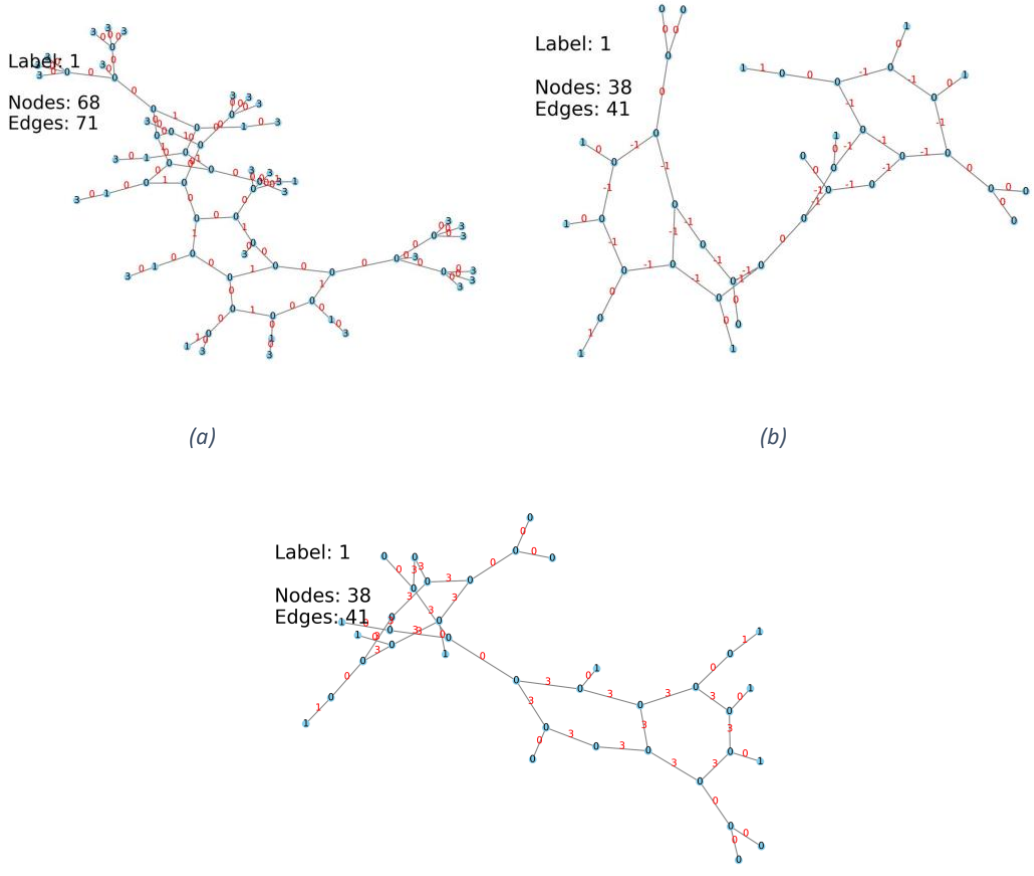
The primary objective of this approach is to classify molecular graphs based on their structural properties. The classification process is broken down into four key stages:

1. **Graph Pre-processing** - Normalize graphs by removing redundancies and ensuring consistency. Ensure graphs **follow a standardized representation** for efficient processing.

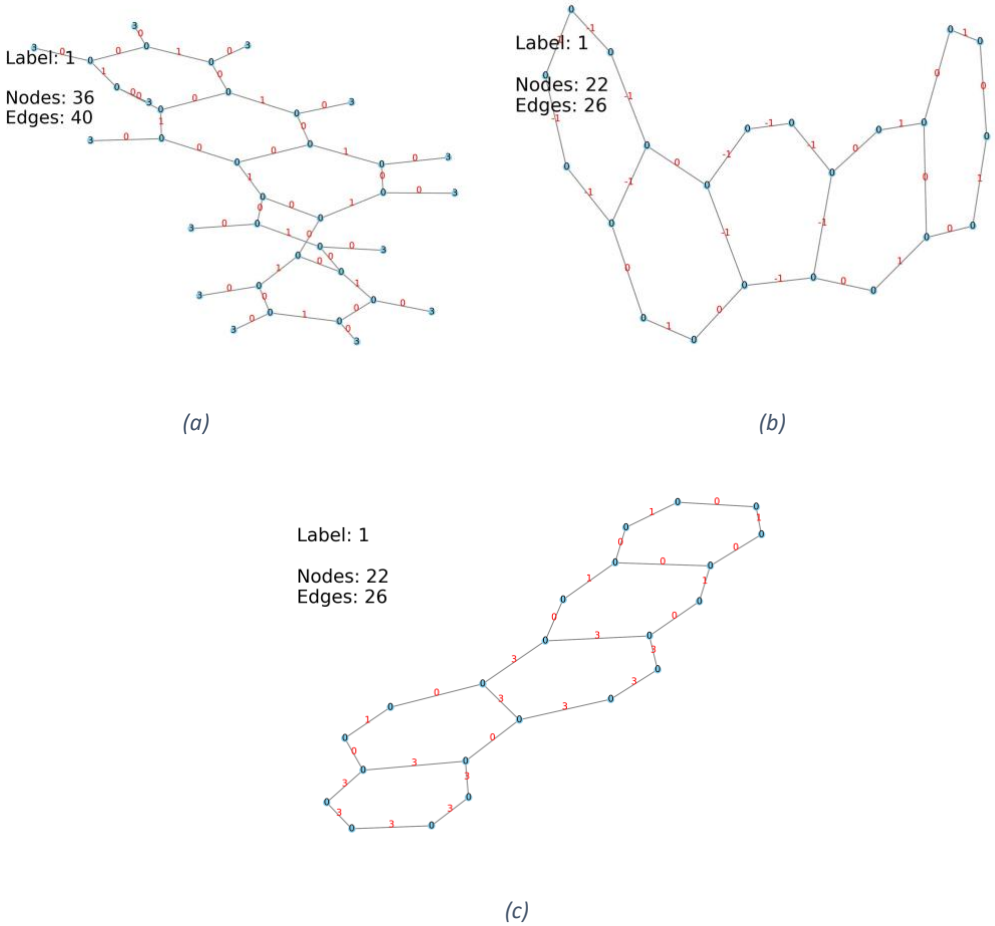
   Since we had a general idea about the structure of molecular graphs, we could for instance what we perceived to be Hydrogen, to do this we look at the total node label frequency in

entire dataset as well as the average number of edges connected to it, If the node label frequency was the highest and the average degree for that label was representing Hydrogen. So we **pruned** these labels from the original graph structure.

Also one noticeable thing in these molecular graphs was representation of a benzene ring, so we **identified such rings in the original graph and marked each edge of this ring** by '-1', later we realised that passing '-1' as label to edge in Gaston was resulting in some weird behaviour, so we replaced it with the edge label as the 'mex' of the original edge labels given in the dataset.The procedure is shown in *(Figure 3.3)*



(Figure 3.3 (a) is the original graph from Mutagenicity, (b) is graph after pruning label (3) and updating the ring edge to '-1' , (c) is after fixing the issue of edge label)



(Figure 3.4 (a) is the original graph from Mutagenicity, (b) is graph after pruning label (3) and updating the ring edge to '-1', (c) is after fixing the issue of edge label)

2. **Subgraph Mining** - Extract the most frequent and discriminative substructures. Ensure the selected subgraphs are **not just frequent**, but also **highly discriminative** between the two classes.

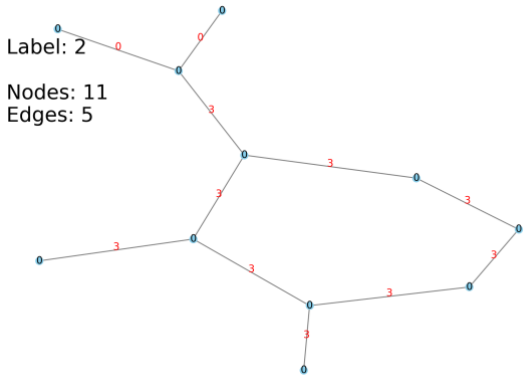
   To achieve this we ran Gaston on a minimum support (value of which is some function on the number of graphs which we found experimenting with different values) on graph labelled 1 (**graph_1**) and graph labelled 0 (**graph_0**) separately to get **subgraph_0** and **subgraph_1** respectively. Gaston along with the respective subgraph also gave us the frequency of its occurrence.

   For a subgraph from graph_0 we gave it utility value given by (and similarly for subgraph from graph_1)

$$U(subgraph) = \begin{cases} edge\_count \times (freq\ in\ graph\_0 - freq\ in\ graph\_1), & subgraph \in subgraph\_1 \\ edge\_count \times (freq\ in\ graph\_0), & subgraph \notin subgraph\_1 \end{cases}$$

   This function was found from experimenting and found what worked best for both Mutagenicity and NCI-H23 dataset. This is in a way telling to give higher preference to the subgraphs with more number of edges, this would help in classification since in the actual data these larger subgraphs might be the reason for the. To check whether the subgraph was in both Gaston output for graph_0 and

graph_1 we used igraph library to get their corresponding canonical label and matching canonical labelling across subgraph_1 and subgraph_0.



(Figure 3.5 Example subgraph found and reported)

After this we sorted the subgraph_1 and subgraph_0 in descending order (only considering positive utility values). If we just directly used them in equal ratio, we were getting bad score in NCI-H23 dataset , so in the final subgraph we reported them in a manner that represented the actual ratio of their presence in the dataset by considering top subgraphs from subgraph_1 and subgraph_0.

(a) *(b)*

(Figure3.6: (a) represent feature vector before fair division of subgraphs and (b) represent feature vector after taking subgraph in proportion, In each image the vertical axis is the top 200 graphs and horizontal axis is 100 subgraphs used to create feature vector, black denotes presence and grey denote absence of the subgraph)

3. **Feature Vector Creation** - Convert each **molecular graph** into a **fixed-length binary feature vector**. The presence (1) or absence (0) **of specific subgraphs** is used to represent each graph.

   We had initial idea of clustering the subgraphs so that on a single feature we could represent the fact that a bundle of subgraphs are present in the graph, but we realised that taking more than 200 subgraphs made the runtime exceed 1 hour on NCI-H23 dataset. We still did try clustering but it did not give a highly good result for a test-train split as we expected so we dropped this idea and just construct feature on the 100 subgraph that we got.

4. **Graph Classification** - Train a **machine learning classifier** using extracted **subgraph features.** Predict the **class label (0 or 1)** of an unseen molecular graph. If a subgraph is **too specific**, it may cause **overfitting.** (The ML classifier was given and not modified by us)

3.5) Implementation Details

Identifying Discriminative Subgraphs (final_subgraph.py)

- **Purpose**: Extract subgraphs that contribute the most to classification.
- **Process**:
    1. Parse frequent subgraphs from graph_0 and graph_1.
    2. Compute a score for each subgraph based on its occurrence.
    3. Select the top 100 discriminative subgraphs.
    4. Save them for feature extraction.

Graph Preprocessing (convert.py, convert_igraph.py)

- **Purpose:** Convert raw graphs into a structured format.
- **Steps:**
    1. Parse graphs into adjacency representations.
    2. Normalize edge labels.
    3. Remove unnecessary nodes and self-loops.
    4. Convert processed graphs to feature vectors.

Feature Vector Construction (convert_igraph.py)

- **Purpose:** Convert graphs into feature vectors based on subgraph occurrences.
- **Steps:**
    1. Parse graphs and subgraphs.
    2. Perform subgraph isomorphism checks.
    3. Construct a binary feature matrix

Edge Removal and Normalization

- **Files:** remove2.py, super_rem2.py, super_rem_dup.py
- **Purpose:**
    - Reindex graphs.
    - Remove duplicate edges.
    - Ensure canonical ordering of edges.

3.6) Analysis and Observations

- **Efficiency:** The approach efficiently handles large datasets.
- **Feature Selection**: Using discriminative subgraphs improves generalization.
- **Graph Pruning:** Removing redundant nodes enhances performance.

3.7) Challenges and Future Improvements

- **Computational Cost:** Subgraph isomorphism is expensive.
- **Feature Selection:** Advanced methods like Graph Neural Networks (GNNs) can be explored.

- **Scalability:** The approach works well for small datasets but may need optimization for larger graphs.

3.8) Conclusion

Graph classification is a fundamental problem in cheminformatics and bioinformatics, playing a crucial role in understanding molecular structures In this project, we successfully implemented a pipeline for classifying molecular graphs by identifying discriminative substructures, transforming them into feature representations, and applying machine learning techniques to classify them.

The core challenge was identifying relevant subgraphs that effectively differentiate between two classes while minimizing computational complexity. By leveraging frequent subgraph mining, we extracted 100 key substructures that best represent the molecular differences. This feature selection ensured that only meaningful substructures were used, reducing noise and improving classification performance.

Graph preprocessing was also a crucial step, involving redundancy removal, edge normalization, and node re-indexing to ensure consistency across the dataset. The feature vector construction phase then mapped molecular graphs to a binary feature representation, making it compatible with traditional machine learning models.

Despite the strengths of this approach, challenges remain. Subgraph isomorphism detection is computationally expensive, and more advanced techniques like Graph Neural Networks (GNNs) could be explored for further optimization. Additionally, handling larger datasets with complex molecular structures may require scalable feature extraction methods.

This project highlights the importance of graph-based learning techniques and their potential in real-world applications such as drug discovery, protein function prediction.