COL 761 HW1 Report Question 3 Analysis

February, 2025

Team members	Entry Number
Hasit Nanda	2024VST9015
Ghosal Subhojit	2022MT61976
Arnab Goyal	2022MT61963

1 Introduction

The goal in this question was to classify molecular graphs by identifying key subgraphs that serve as discriminative features. We developed a systematic pipeline for selecting frequent and unique subgraphs by incorporating a variety of statistical tests, and then transforming graphs into feature vectors, and utilizing them in classification.

2 Methodology

2.1 Approach 1 of generating a pool of frequent subgraphs

2.1.1 Step 1: Splitting Graphs Based on Labels

To ensure a structured approach to identifying discriminative subgraphs, we first split the dataset into two subsets based on labels (0 and 1). This step was crucial to mitigate potential label imbalance, ensuring that the frequent subgraph selection process remained unbiased.

2.1.2 Step 2: Frequent Subgraph Mining

We applied the Gaston algorithm to extract frequent subgraphs within each category. The threshold for Gaston was set at 40%, ensuring that the selected subgraphs are frequent enough to be representative while avoiding noise from overly rare patterns.

2.1.3 Step 3: Filtering Common Subgraphs Across Categories

We generated canonical labels for each subgraph and retained only the unique canonical labels to eliminate redundancy. The canonical labeling process used a partial ordering based on vertex labels and degrees.

2.1.4 Step 4: Selecting a Set of Frequent Subgraphs

After filtering, we retained around 300-500 unique subgraphs, ensuring a manageable feature set.

2.2 Approach 2 of generating a pool of frequent subgraphs

2.2.1 Step 1: Mining Frequent Subgraphs

We apply the Gaston algorithm with an initial threshold of 30. If the total number of mined subgraphs multiplied by the total number of input graphs exceeds 3,800,000, we increment the threshold by 5 and repeat the mining process until the number of mined subgraphs multiplied by the total graphs falls below or equal to 3,800,000. This ensures that the subsequent subgraph isomorphism computations remain computationally feasible.

2.2.2 Step 2: Computing Subgraph Occurrences

After obtaining the final set of frequent subgraphs, we perform subgraph isomorphism checks to generate occurrence lists. Since this step is computationally expensive, our thresholding mechanism from Step 1 helps manage the total computational load effectively.

2.2.3 Step 3: Feature Selection and Classification

The extracted subgraph occurrence data is then used as features for classification. This refined feature set results in more informative graph representations, leading to improved classification performance.

2.2.4 Performance Comparison of approach 1 and 2

Table 1 compares the classification accuracy of the two approaches.

Method	Average Accuracy
Approach 1	74.8%
Approach 2	>81.2%

Table 1: Performance comparison of the two approaches.

2.3 Step 4: Selecting a Set of Frequent Subgraphs

After filtering, we retained the most frequent subgraphs from both categories. This resulted in a pool of around 300-500 unique subgraphs, providing a sufficiently large yet manageable feature set.

2.4 Step 5: Ranking Subgraphs

To rank subgraphs, we made a feature selection pipeline that combines multiple statistical methods, ensuring both relevance and non-redundancy.
First, we filter redundant features using the Jaccard similarity coefficient.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A and B are sets of samples containing a given feature. If similarity exceeds a threshold, the feature is removed.
For ranking, we use four statistical methods:
1. **Mutual Information (MI)** Which is used to determine the dependency between a feature and the target using:

$$I(X,Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

2. **Chi-Square Test** determines the independence between features and the target using:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

where O and E are the observed and expected counts.

Finally, we find a weighted combination of these scores. Sort them and select the top 100 subgraphs, ensuring robust and discriminative feature selection.

2.5 Step 6: Transforming Graphs into Feature Vectors

Each molecular graph was converted into a binary feature vector, where each dimension represented the presence or absence of one of the selected 100 discriminative subgraphs. Subgraph isomorphism was performed using the `retworkx` library.

2.6 Step 7: Classification

The resulting feature vectors were fed into the provided classification algorithm to evaluate performance.

3 Results

We evaluated our algorithm on two datasets:

Dataset	Train Accuracy	Test Accuracy
NCI	0.935	0.832
Mutagenicity	0.885	0.802

Table 2: Performance of the classification algorithm on different datasets.

These results demonstrate that our feature selection approach was effective, achieving a reasonably high test accuracy while avoiding overfitting.

4 Justification of Our Approach

The key justifications for our improved approach are:

- **Dynamic Support Threshold**: The threshold is adjusted iteratively to balance the trade-off between computational feasibility and extracting a sufficient number of subgraphs, ensuring efficient mining.

- **Scalability**: By limiting the total number of mined subgraphs through threshold adjustments, we ensure that the computational burden of subgraph isomorphism remains manageable.
- **Using Gaston for Efficient Mining**: Gaston is well-suited for mining frequent subgraphs efficiently in large datasets.
- **Robust Feature Ranking:** Our ranking method combines multiple statistical approaches to ensure strong, non-redundant feature selection, leveraging mutual information and chi-square independence.

5 Conclusion

We successfully implemented a graph classification pipeline that identifies and utilizes discriminative subgraphs as features. Our method achieved strong performance on the provided datasets, demonstrating its viability for molecular classification tasks. Future improvements could explore optimizing subgraph selection criteria further to enhance generalization performance.