COL761: Data Mining

Homework 1 - Question 3

1 Introduction

Graph classification is a crucial task in machine learning, widely applied in fields like bio-informatics, chemo-informatics, and social network analysis. In bio-informatics, it helps predict protein functions and supports drug discovery by analyzing molecular interactions. In chemo-informatics, it plays a role in predicting compound activities and assessing molecular structures.

In this assignment, the goal is to classify molecular graphs by identifying key subgraphs that contribute significantly to classification. Given a set of molecular graphs, the task is to extract relevant substructures and use them to convert each graph into a binary presence/absence feature vector with at most 100 dimensions. A straightforward approach is to select the top-k most frequent subgraphs, but a more effective method would involve mining a larger set of frequent subgraphs and filtering out the most discriminative ones.

2 Understanding the Problem

To perform classification, it is necessary to identify discriminative subgraphs within the graphs that are most useful for distinguishing between different classes. These subgraphs will be used as features to represent each graph in a structured numerical format. The final representation should be a binary feature vector indicating the presence or absence of selected subgraphs.

A challenge in this task is choosing the right descriminative subgraphs. Simply selecting the most frequent subgraphs may not be optimal, as frequent subgraphs might not be discriminative. Instead, the focus should be on finding subgraphs that provide the highest distinction between the two classes. Another challenge is ensuring that the feature extraction process generalizes well across unseen data.

3 Methdology Used

3.1 Preprocessing

We perform two primary preprocessing steps on the input data:

- Text Modification: Replace all occurrences of the letter 'e' with 'u'. Modify lines containing # to add a numbered identifier (t # Graph counter).
- Graph Cleaning: Remove duplicate edges where an undirected edge (u, v, w) might appear as both (u, v, w) and (v, u, w), keeping only one occurrence.

The replacement of 'e' with 'u' is necessary to standardize text formatting and use the fsg algorithm further up ahead. Adding the numbered identifier (t # Graph counter) ensures that graph datasets are properly labeled and distinguishable.

Standardizing labels and removing duplicates simplifies subsequent operations such as feature extraction, graph analysis, or model training. By applying these pre-processing steps, we create a clean, structured dataset that is easier to work with, improving both efficiency and accuracy in further data processing tasks.

3.2 Using FSG to find frequent subgpraph

A *frequent subgraph* is a subgraph (a graph that can be part of a larger graph) that appears in many of the graphs in a given dataset. The key points in this definition are:

- **Graph Dataset** ($D = \{G_1, \ldots, G_n\}$): You have a dataset D consisting of n graphs. Each graph G_i is part of this dataset.
- **Frequency Threshold** (θ): The threshold θ defines the minimum percentage of graphs in the dataset in which a subgraph must appear to be considered "frequent." For example, if $\theta = 50$, the subgraph must appear in at least 50% of the graphs.
- **Observed Support** (μ_0): The support of a subgraph g refers to how many graphs in the dataset it appears in. μ_0 is the number of graphs that contain the subgraph g.
- **Condition for Frequent Subgraph**: A subgraph g is considered **frequent** if its observed support μ_0 satisfies:

$$\mu_0 \geq \frac{\theta \times |D|}{100}$$

where $|D|$ is the total number of graphs in the dataset. This means the subgraph must appear in at least a certain percentage ($\theta\%$) of the graphs.

In our approach using the **Frequent Subgraph Growth (FSG)** algorithm, we set a **low support threshold** to generate a large number of frequent subgraphs. This allows for a more comprehensive exploration of the subgraph space, ensuring that potentially discriminative patterns are not overlooked. By generating many frequent subgraphs, we can accurately identify the most relevant discriminative subgraphs for classification.

Additionally, we do not ignore subgraphs with size 1, because the presence of toxicophores such as aliphatic halides (e.g., -Cl, Br, I) in the Mutagenicity dataset may capture essential structural patterns associated with mutagenic behavior, making them valuable in the feature selection process.

> *Reference: The Mutagenicity dataset is a well-known benchmark dataset in cheminformatics, used to study the structural properties of molecules that contribute to mutagenic activity.*

In conclusion, we use the FSG algorithm with low support(25%) and also generate the .tid files in the algorithm for further processing.

3.3 Using .tid files generated to get a labelled .csv file

We utilise the .tid file which contains transactional data where each line consists of a range identifier (e.g., 6-2) followed by a list of transaction IDs (TIDs). Since this raw format is unstructured and difficult to analyze, we preprocess it by converting it into a structured binary matrix where each row represents a TID, and columns indicate the presence (1) or absence (0) of a specific range identifier. This transformation facilitates efficient querying, pattern recognition, and compatibility with machine learning models.

Once the structured CSV is generated, the next step is to associate each TID with its corresponding label from the labeles text file. The label file contains one label per line, corresponding to TIDs in order. We map each TID from the CSV to its respective label, adding a new "Label" column to the dataset. This step is essential for selecting discrminative subgraphs that represent a side of binary classifiation we are given.

3.4 Identification of Discriminative Subgraphs

We employ the following techniques for the identification of discriminative subgraphs. The inclusion of statistical analysis was inspired by the work presented in [2]. The remaining techniques were conceived through collaborative brainstorming efforts, with significant input from ChatGPT's suggestions. These methodologies were then carefully developed to enhance the model's performance and interpretability, incorporating both traditional statistical approaches and machine learning strategies to identify and retain the most relevant features for binary classification tasks.

Handling Class Imbalance

Class imbalance is a prevalent issue in classification tasks, where one class may dominate the dataset, leading to biased model predictions. To address this, we employ the **Synthetic Minority Over-sampling Technique (SMOTE)** [3], which generates synthetic samples for the minority class by interpolating between existing samples. This technique helps balance the class distribution, ensuring that the model is not biased toward the majority class and can effectively predict both classes. While class imbalance is not significant in the Mutagenicity dataset, it is present in the NCI-H23 dataset and may also affect the third dataset.

Removing Highly Correlated Features

To improve model performance, we identify and eliminate highly correlated features. Features with high correlation (above a predefined threshold) provide redundant information, increasing model complexity and introducing multicollinearity. Removing these features reduces the risk of overfitting and ensures that the model captures independent patterns in the data, enhancing its generalization capabilities.

Feature Selection

Following data preprocessing, multiple feature selection techniques are applied to identify the most informative features for model training:

- **Variance Thresholding**: Removes features with very low variance, as they do not provide significant discriminative power.
- **Mutual Information**: Measures the shared information between each feature and the target variable, selecting features with a stronger association.
- **Chi-Square Test**: Evaluates the independence of categorical features from the target variable, selecting those with significant relationships.
- **Recursive Feature Elimination (RFE)**: Iteratively removes less significant features using a machine learning model (Logistic Regression) until only the most relevant features remain.
- **Lasso (L1 Regularization)**: Applies L1 regularization to penalize less important features by shrinking their coefficients toward zero, thereby selecting only the most impactful features.

These combined methods ensure that only the most significant features are retained, improving both model efficiency and interpretability.

Feature Importance Calculation Using SHAP

After feature selection, we compute **SHAP (SHapley Additive exPlanations)** values to quantify the contribution of each feature to the model's predictions. SHAP values provide an interpretative framework that breaks down a model's output, illustrating the individual effect of each feature on the predicted outcome. In this study, a Logistic Regression model is trained on the selected features, and SHAP values are computed to measure each feature's significance in the model's decision-making process.

Ranking Features by SHAP Importance

Finally, features are ranked according to their SHAP values, with higher SHAP values indicating greater importance in the model's predictive behavior. This ranking provides valuable insights into the most influential features, aiding in model interpretation and guiding future refinements. By focusing on the most significant features, the model can be optimized for improved predictive accuracy and interpretability.

Saving the Identified Discriminative Subgraphs

The selected subgraph are saved by its unique identifier for further analysis. These discriminative subgraphs represent structural patterns in molecular graphs that strongly influence classification outcomes.

3.5 Extracting structure format of the discriminative subgraphs

The .fp file generated in fsg was used to fetch the structure format of the selected discriminative subgraphs. Each discriminative subgraph is located using its unique identifier (t #subgraph id), and all related structural details, such as nodes and edges are extracted until the next pattern appears. This .txt file containing the discriminative subgraphs' structures is later used to convert graphs into feature vectors by performing subgraph isomorphism which in turn is used in the classifier for prediction.

References

[1] OpenAI, "ChatGPT is used to generate codes for faster progress," *ChatGPT*, https://chatgpt.com/.

[2] Sayan Ranu and Ambuj K. Singh, "Mining Statistically Significant Molecular Substructures for Efficient Molecular Classification," *Journal of Chemical Information and Modeling*, https://pubs.acs.org/doi/abs/10.1021/ci900035z.

[3] J. Brownlee, "SMOTE: Synthetic Minority Over-sampling Technique," *Machine Learning Mastery*, https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.