Graph Classification

Assignment of Graph Theory and Algorithms

Alessandro Bregoli

1 Introduction

The study of graphs is an extremely vast discipline and of great interest to multiple fields. Potential applications range from the study of social networks to molecular analysis.

In this assignment I will deepen the graph classification and, in particular, I will present the graph classification method developed by (Lara and Pineau (2018)). Graph classification is the process of predicting the class of a given graph. Many machine learning approaches has beed applied to this task. The main idea behind the approach of (Lara and Pineau (2018)) and many others is to extract a set of features from each graph and then, use one of the many Machine Learning classifiers such as, Random Forest, Decision Tree, SVM, MLP...

The rest of the document is organized as follows. In Section 2 I present the model from (Lara and Pineau (2018)). In Section 3 I present the Rust implementation of the model. Finally, in Sections 4 I briefly show the result achieved by the rust implementation of the model and in Section 5 I draw conclusion about the model and the implementation.

2 Model

The model developed by (Lara and Pineau (2018)) can be divided in two parts:

- Feature extraction
- Classification

Let $G = (V, E)$ be an undirected and unweighted graph and A its adjacency matrix. Let D be the diagonal matrix of node degrees, the normalized Laplacian of G is defined as:

$$\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{1}$$

The model uses the k smallest positive eigenvalues of $\mathcal{L}$ in ascending order as input of the classifier. If the graph has less than k nodes, the authors use right

zero padding to get a vector of appropriate dimensions.

The authors then, suggest applying a Random Forest classifier on the extracted features. However, since the goal of the authors was to develop a simple model, I decided to use an even simpler classifier: the decision tree.

3 Implementation

The program language I decided to use is Rust. The implemented algorithm is available on github[1] and is capable of:

- Read a dataset composed by a set of labelled graphs
- Train the Decision Tree and evaluate the method computing the accuracy in cross validation.

The implementation I developed provides a cli (Listing 1)with the following parameters:

- *-a adj_list.txt*: required. Path to the adjacency list file in csv format
- *-n node_to_graph.txt*: required. Path to graph identifiers file in csv format for all nodes of all graphs
- *-g graph_labels.txt*: required. Path to the class labels for all graphs in the dataset in csv format
- *cross_validate*: required. Execute cross validation with a decision tree classifier
- *-k 10*: required. Number of folds to apply
- *-f 18*: required. Number of features to extract from each graph

Listing 1: rgclass - Execute a cross validation on the MUTAG dataset

```
$ rgclass -a datasets/MUTAG/MUTAG_A.txt\
  -n datasets/MUTAG/MUTAG_graph_indicator.txt\
  -g datasets/MUTAG/MUTAG_graph_labels.txt\
  cross_validate\
    -k 10\
    -f 18

Accuracy: 0.8666666746139526
```

4 Results

In order to evaluate *rgclass* I selected a subset of the datasets used by the authors of the model:

- **MT**: The MUTAG dataset consists of 188 chemical compounds divided into two classes according to their mutagenic effect on a bacterium.

[1] https://github.com/AlessandroBregoli/rgclass

- **EZ**: ENZYMES is a dataset of protein tertiary structures consisting of 600 enzymes from the BRENDA enzyme database. In this case the task is to correctly assign each enzyme to one of the 6 EC top-level classes.

- **PF**: Proteins full
- **NCI1**: NCI1 represents a balanced subset of datasets of chemical compounds screened for activity against non-small cell lung cancer.

After the selection of the datasets I computed the 10-fold cross-validation accuracy with embedding dimension (k) set to the average number of nodes for each dataset. The result are reported in the following table.

	MT	EZ	PF	NCI1
RandomForest	88%	43%	74%	75%
DecisionTree	85%	12%	59%	56%

5 Conclusions

The model presented by (Lara and Pineau (2018)) is really simple. However, this simplicity come with a low computational complexity.

In order to further reduce the execution time I decided to apply the Decition Tree classifier. The results presented in Section 4 shows that the performace drastically decrease using a Decision Tree. The only exception is the experiment on MT dataset. This is probably due to the fact that MT dataset is the smallest one both in the number of networks and in the size of the networks.

Bibliography

Lara, Nathan de, and Edouard Pineau. 2018. "A Simple Baseline Algorithm for Graph Classification." *Relational Representation Learning Workshop, NIPS 2018*.