Graph Classification

Name: Francesco Palermo StudentId: 45539669

Macquarie University, Sydney, Australia
COMP8230 Mining Unstructured Data
Assessment Task 2

Keywords: Graph Classification, Graph Mining, Frequent Subgraph

1 Introduction

Graphs are general and powerful data structures that can be used to represent diverse kinds of objects [1]. Thanks to their easy generalization, they have been progressively crucial in modelling complex structures in a broad variety of applications. In fact, the majority of the data in the world at the moment can be classified as unstructured and graph have become a very practical tool to represent this kind of data. Examples can be found in semi-structured data such as XML and HTML, chemical compounds, biological sequences, computer vision, Web analysis and Social Networks.

Over the last few decades, the topic of graph mining has developed various methods for processing graph data. Unsupervised, semi-supervised and supervised machine learning techniques were used in order to discover hidden patterns within data graphs.

The primary focus of this research paper is to provide an overview of existing graph classification techniques. It is important to note that the term “graph classification” can denote three different tasks.

The first task can refer to building a model that predicts the class label of the entire graph. For instance, chemical compounds can be defined as graphs and according to the shape of the graph, the functionality of the compound varies.
In particular, the training data comprises of several chemical compounds (data points) and their known functionality, such as being toxic on nontoxic (labels).

The binary classification problem here is to predict the functionality of an unseen compound structures.

The second task is to build a model that predicts the class labels of nodes in a single large graph. The evident distinction between the first task and the second task is that we are now dealing with a single vast graph, where some nodes are not labelled. This method is also known as *"label propagation"* and it is classified as semi-supervised machine learning technique.
This method is applied in various social networking applications where users are associated with nodes. For example, let's assume we want to find people who have interest in hockey. The labels of a small number of users needs to available.
We are training a model on the available labels to predict those nodes for which the label is unknown and therefore enable forecasting on whether users are hockey fans or not.

The third task is to build a model that predict the class labels of edges in a single large graph. The goal here is to predict the interaction between nodes.
This is mainly a binary classification problem where we are interested to see whether or not there is a relation between the nodes. The kind of relation (edges) depends on the particular application considered. For instance, a particularly interesting use is learning protein-protein interactions. Protein-protein interactions are central to all the biological processes and structural scaffolds in living organisms [2].
In this situation, each node in the graph is a particular protein, while the edges display the protein interaction. Thus, the classification problem consists in predicting the presence or absence of edges among proteins, given some known edges.

The rest of the paper mostly deal with the graph classification related to the described first task. In the **Related Work** section, we introduce three existing approaches in the literature based on distance methods. Then, in the **Identified Methodologies** section, two frequent substructure-based methods will be described and compared to the distance based approaches. In the **Conclusion**, the summary of the topic and some possible future improvements will be described.

2 Related Work

Distance-based methods are most appropriate when the sizes of the underlying graphs are small, and the distances can be computed efficiently [3].
The following methods will be discussed below: **K-Nearest neighbors**, **collective classification** and **kernel methods**.

K-Nearest neighbors (*knn*) is one of the simplest non-parametric (not making an assumption on the data) algorithm used for both classification and regression.
It is often described as the *lazy-algorithm* because it does not attempt to build any models during training time, which leads to no training steps.

The following pseudo-code will help the reader to better understand the logic of the algorithm.

1. *Load the training and test set*
2. *Choose the value of k.*
3. *For each graph in the test set (unknown label)*
    - *Find the distance to all graphs in the training set*
    - *Store those distances in a list and sort it*
    - *Choose the first k points in this list*
    - *Assign a class to the graph in the test set based on the majority of the classes present in the chosen points*
4. *End*

The hyper-parameter *k* needs to be set in advanced. A large value of *k* makes *Knn* less sensitive to noise, while a small value allows capturing finer structure. The precise value of *k* depends on the data and hyper-parameter tuning techniques such as Grid Search or Randomized search helps finding it.
The distance metric between graphs can vary, but usually GED [4] (graph edit distance) is the preferred measure.
Finally, the advantages of *Knn* are the simplicity of the algorithm, the highly performance and the little tuning required. On the other hand, the drawbacks are that both speed and accuracy can degrade for large dataset.

Collective classification is a semi-supervised method. This means that both training set and test set need to be accessible at the same time for the classification process.
Next, an explanation of this method is provided. An additional higher level graph called a neighborhood graph is built on top of the training and test graph objects. This new graph does not need to be mixed up with the starting graph objects.
The training and test graphs comprises the nodes in the neighborhood graph.
This ends up in a final graph which contains both labeled and unlabeled nodes (note that here nodes are actual graphs).
Then, every node is linked to its *k* nearest neighbor objects based on a distance metric (see GED above). This is a collective classification problem, which through some algorithms such as the *Iterative Classification Algorithm (ICA)*, derive labels of the unlabeled nodes in the neighborhood graphs. (See [5] in the Reference Section for the algorithm detail)

Kernel methods, such as support vector machines, construct a prediction rule based on a similarity function between two objects [6]. These similarity functions are also called *kernel functions*.
Let's define K (g_i, g_j) to be the similarity measure between two graphs g_i, g_j.

The prediction model (classification function) is a linear combination of the kernel functions for all training data points.
There are numerous *kernel functions* available in literature and they differ to each other according to how they gather features from the graphs. The main disadvantage of graph kernels is that it is vague which features have strong contribution to the classification.

In conclusion, the discussed distance-based methods are more efficient when the size of the graphs in the training set are small. This is because each described algorithm needs to calculate several distances between graphs and this can become too computationally costly when the dataset size is too large.

3 Identified Methodologies

Frequent substructure-based methods are the proposed alternative approach in comparison to the distance-based methodologies studied before. During the discussion, we are examining the selected techniques and the corresponding benefits of this system. This section will be split up in two parts for clarity.

3.1 The important concept of frequent subgraphs

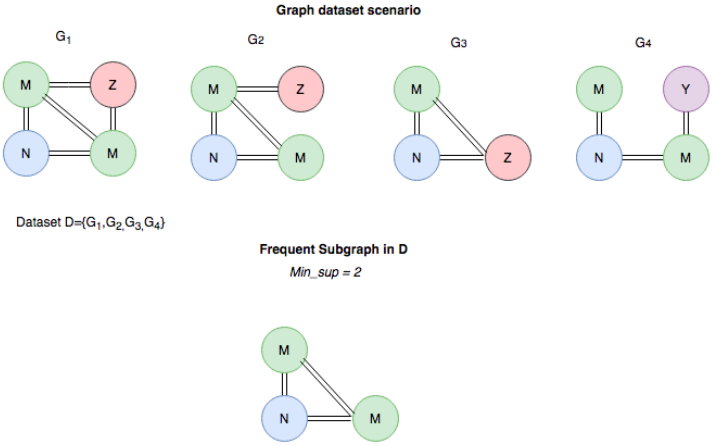
Let $D = \{G_1, ..., G_{nt}, ..., G_n\}$ be the given dataset of graphs. Among these, we suppose that the first *nt* graphs have been observed and therefore labeled. These graphs comprise the training set together with the matching labels $\{l_1, ..., l_k\}$. The remaining $(n - nt)$ graphs are unlabeled and make up the testing set. The desired goal is to infer the labels in the testing set (unknown) by training a classification model in the test set.

The core idea of these methods is the extraction of **frequent subgraphs** from the data. They are important for characterizing graph sets by facilitating discrimination between different groups of graphs. Thus, both unsupervised and supervised machine learning methods uses frequent subgraphs as building block tool.
These are turned to features which enable example classification models to be built in the traditional way. It comes naturally to ask: *"How can we derive frequent subgraphs from graphs?"*.

The original idea comes from a classic problem in data mining called *itemset mining*, where frequent subsets (that satisfy a certain condition) are counted from a particular series of sets. Market Basket Analysis, which is a famous technique largely used by large retailers, uses itemset mining by finding combinations of objects that appear together often in transactions.

Researchers were capable of applying this frequent pattern mining in graph data. Let's first define the concept of **subgraph**. According to Jiawei Han [7], *a graph g is a subgraph of another graph g', if there exists a subgraph isomorphism from g to g'.*

A graph isomorphism occurs when there is an exact match between the involved graphs, or in other words a one to one relationship exists between the two objects.
Then, we can define the **support(g)** as the percentage of graphs in *D* where *g* is a subgraph. Finally, those sets of subgraphs whose support is higher than a pre-set minimum support threshold (*min sup*) can be called **frequent subgraphs.**
The following chart aims to show the above concepts graphically by simulating a very simplified scenario with only four graphs (*D*). Then a frequent subgraph, which support is higher than 2, have been identified and displayed.



The identification of frequent substructures can be summarized with the following two stages. In the first stage, frequent substructure candidates are generated.
In the last stage, the support of each candidate is obtained and only those who support are higher than a threshold is kept.

Next, we briefly look at a method for frequent substructure mining. Among different methodologies usually an Apriori-based approach is preferred. This algorithm has been one the greatest successes of business data mining.

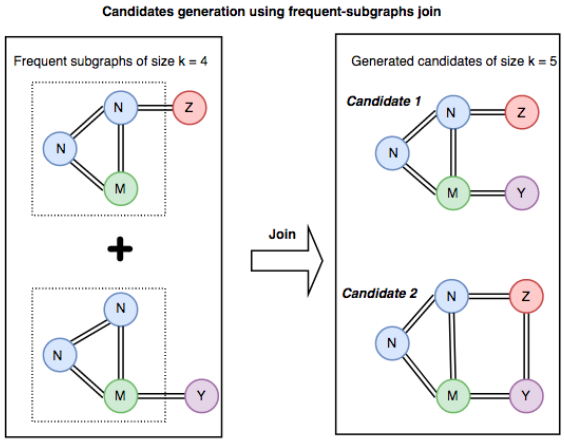
The Apriori is an iterative algorithm which generates candidate patterns of size *(k + 1)* by joining frequent patterns of size *k*. The discovery of frequent graphs begins with graphs of size *k=1* and continues in an iterative bottom-up manner by generating new candidates at each step. The *graph size* varies on whether we count the number of nodes or edges, however the following description is more general and does not consider which approach is used (*node extension* or *edge extension*).

The algorithm takes two input parameters which are the graph database *D* and the threshold *minsup*. A level-wise method is adopted, in which candidate subgraphs C_{k+1} of size *(k + 1)* are created by joining on graph pairs from the set of frequent subgraphs

F_k of size *k*. The key point of Apriori-based subgraph mining algorithms is the candidate generation step.

According to Charu C. Aggarwal [8], "*the two frequent graphs need to be matching in a subgraph of size (k − 1) for a join to be successfully performed. The resulting candidate subgraph will be of size (k +1). Therefore, one of the important steps of join processing, is determining whether two graphs share a subgraph of size (k−1) in common*".

The next graph displays two possible resulting candidates from a join between frequent subgraphs. Note how in the left side of the graph, the rectangles with the dotted lines highlight the shared subgraph of size *k-1*which is the necessary condition for the join.



Finally, after some candidates C_{k+1} are pruned, the remaining ones are used for computing the support. Only those candidates whose support is equal or greater than *minsup* are retained in a list F_{k+1}. The iterative process is repeated until a stopping criteria is met (empty set F_{k+1} is generated). The algorithm ends and the set of frequent subgraphs $\bigcup_{i=1}^{k} F_i$ is returned.

3.2 Frequent substructure-based methodologies

Frequent subgraph-based methods transform frequent subgraphs from the data into features in order to build classification models. **Generic Transformation** and **XRules** approaches will be covered in the following subsections.

3.2.1 Generic Transformational Approach

This approach can be explained by the following three steps:

1. Execute the Apriori-algorithm to find frequent subgraph patterns from the initial graph dataset *D*. Apply *feature selection* algorithm, such as the filter method, to reduce redundancy and maximize the importance of the features. Let *d* be the total number of frequent subgraphs (features).

2. For every training graph G_i in *D*, a vector-space is built. This represents the *d* features found. The value of each feature is binary which indicates presence or absence of subgraphs.

3. Select any classification model such as Decision Tree, Random Forest, etc, and train a model which is able to classify unseen graph label in the testing set.

3.2.2 Xrules: A Rule-based approach

This is a rule-based strategy that connects frequent substructures to different classes. As before, the training phase consists of three steps:

1. First, frequent substructures with sufficient support and confidence are determined. Among various *rules strength* available in literature, rule-confidence is typically the preferred one. It measures how often the rule has been found to be true. Each rule is of the form:
   $$F_g \rightarrow l_k$$
   Where F_g is a frequent subgraph and l_k represents a class label.

2. Next, the rules are ordered by decreasing confidence. Some statistic thresholds can be used for eliminating rules with low strength. This create a more compact set of rules for classification purposes.

3. Once the training model have been built, it is ready for classification tasks. For each test graph G_i, the rules that are fired by G_i are determined. A default class (usually the training class label mode) is used if no rules are fired.

4 Conclusion and Future Work

The main advantage of frequent-based methodologies over the distance-based approaches is that these algorithms can deal with large graph datasets.
In addition, H. Saigo and others, have stated how these have better performance than distance-based in terms of classification accuracy in chemical compound datasets [9].

This research paper has covered two different methods for graph classification. Distance-based uses similarity measures between two graphs, while Frequent substructure-based methods derive representative subgraphs that are used for any machine learning algorithms.

We also introduced some mining graph techniques such as deriving frequent subgraphs from a graph dataset. We described how the frequent subgraphs are considered a significant building block for building classification model and implement an Apriori-Algorithm to derive them.
The generic transformational approach converts frequent subgraph into valuable new features, while the Xrules uses ruled-based techniques.

In this paper, I tried to unify some of the previous works, but in general the topic as a whole still needs a reliable theoretical background. In a world where especially unstructured data (graphs included) are generated every second, we need to shift the attention to this type of data so we do not lose any important insights.

5 References

1) Tsuda K., Saigo H. (2010) Graph Classification. In: Aggarwal C., Wang H. (eds) Managing and Mining Graph Data. Advances in Database Systems, vol 40. Springer, Boston, MA, Chapter 11, pag337

2) Mohamed TP, Carbonell JG, Ganapathiraju MK. Active learning for human protein-protein interaction prediction. *BMC Bioinformatics*. 2010;11 Suppl 1(Suppl 1): S57. Published 2010 Jan 18. doi:10.1186/1471-2105-11-S1-S57

3) Aggarwal, Charu.C., Data Mining the Textbook 2015, Chapter 17, pag 583

4) https://en.wikipedia.org/wiki/Graph_edit_distance

5) Aggarwal, Charu.C., Data Mining the Textbook 2015, Chapter 19

6) Tsuda K., Saigo H. (2010) Graph Classification. In: Aggarwal C., Wang H. (eds) Managing and Mining Graph Data. Advances in Database Systems, vol 40. Springer, Boston, MA, Chapter 11, pag339

7) Han, Jiawei, et al. Data Mining, Southeast Asia Edition, Elsevier Science & Technology, 2006, Chapter 9.

8) Aggarwal, Charu.C., Data Mining the Textbook 2015, Chapter 17, pag 576

9) H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. GBoost:
A mathematical programming approach to graph classification and regres-
sion. Machine Learning, 2008.