Anomaly Detection in Time Evolving Networks

Anomaly detection refers to the problem of finding patterns in data that fail to conform to the expected standard. Such anomalous patterns are useful to a number of business applications, such as identifying trending topics on social media and suspicious traffic on computer networks, as well as detecting credit card fraud, insurance fraud, and e-auction fraud.

There are a variety of anomaly detection algorithms available, and the correct one to use depends on the type of anomaly you are trying to detect. A few examples are point anomalies, contextual anomalies, and collective anomalies. This project will be focused on anomaly detection in time evolving graphs. Various methods have been explored, including time series models, similarity-based algorithms, and community-based algorithms. Evaluation of these approaches is often not straightforward because of the lack of ground truth data. However, we can compare the detected anomalies with normal time points in the dataset, or compare results among different methods in order to evaluate the performance of the methods indirectly.

Datasets

A collection of time evolving graph data has been provided to you. This data was primarily taken from the Stanford Large Network Dataset Collection. There are four time-evolving graphs from different domains. For example, the Enron graph represents an email network and the p2p-Gnutella graph represent a peer-to-peer network. Each time-evolving graph is represented as a series of text files that define the graph at a given time point. Each text file is given as an edge list but with the first line stating the number of nodes and edges, respectively.

Project Requirements

You are required to implement an algorithm from one of the four provided papers. You may choose whatever paper/algorithm you are most interested in. If you have already completed this project in the Graph Mining course, you must select a different paper.

Specifically, you must do the following:

1. Read and understand the scientific publication assigned to you. If your publication has other emphases besides anomalous time point detection, then your implementation should focus only on the anomalous time point detection part.
2. Implement the algorithm described in the publication using either R or Python. Your code must contain detailed comments and a `README` file that specifies any software that needs to be installed (using python's `pip` or R's `install.packages`). Your main file should be called `anomaly.py` or `anomaly.R` and should take one argument, the data directory, as input. **Make sure your code includes detailed comments.**
3. Have your code output a time series of your algorithms output to the file `time_series.txt`. If your paper uses a similarity score then output the similarity value

time series. If it uses a distance metric then output the distance value time series. Let each line represent a single time step (i.e., one number per line).
4. Finally, generate a plot of the above time series. This does not have to be performed in the program and can simply be created manually from the results file. On the plot, indicate the threshold value (for determining an anomaly) with a horizontal line.

Calculating the threshold value for determining anomalies

To obtain the threshold value you will need to calculate the moving range average, M, and the median value of the time series. The upper threshold will be given by median + 3M, and the lower threshold will be given by median − 3M. For a time series with n points, M will be calculated as follows:

$$M = \frac{\sum_{i=2}^{n} M_i}{n-1}, \text{ where } M_i = |x_i - x_{i-1}|$$

Submission Instructions

Submit a single zipped folder containing the following things:

- the anomaly detection code with `anomaly.py` or `anomaly.R` being the main file that is run
- a `README` file
- a `time_series .txt` file for each of the four graphs, append the graph name as a prefix (e.g., `enron_by_day_time_series.txt`)
- a plot of the time series for each graph (image or pdf filetype). Save using same prefix convention as above
- the pdf of the paper whose algorithm you implemented

Paper Clarifications

Clarifications for Paper 1: Implement only the algorithm described in Section 5.5 using Signature Similarity. Because of the way the similarity is calculated, anomalous graphs are identified by two consecutive anomalous time points in the output. For example, similarity score 1 is between graphs 1 and 2 and similarity score two is between graphs 2 and 3. If both similarity score 1 and similarity score 2 are found to be anomalous, the anomalous graph is then graph 2, since it is the one in common. Use the lower bound from the individual moving range threshold, and anything below it is an anomaly.

Clarifications for Paper 2: Implement only algorithm 2. You do not have to implement the graph clustering and classification uses, only the pairwise similarity. Test the algorithm using different values for g and report the results. Because of the way the similarity is calculated anomalous graphs are identified by two consecutive anomalous time points in the output. For example, similarity score 1 is between graphs 1 and 2 and similarity score two is between graphs 2 and 3. If both similarity score 1 and similarity score 2 are found to be anomalous, the anomalous graph is then graph 2, since it is the one in common. Use the lower bound from the individual moving range threshold, and anything below it is an anomaly.

Clarifications for Paper 3: Because this paper uses directed, weighted graphs, you will have to implementing the algorithm using a different set of features. The features you should use are: degree of the node, clustering coefficient of the node and number of edges in the egonet of the node. Note that the egonet of a node is the subgraph induced by the node and its neighbors. A plot should be generated for each of the features. You should use a window size W of 7, as in the paper. Compute the “typical eigen-behavior” using only the average; using SVD is not required. Calculate the upper threshold of each feature using the moving range average. Any time point above the threshold is an anomaly.

Clarifications for Paper 4: It is not necessary to implement the clustering portion of the paper. Implement only the similarity scoring. The similarity score is the Canberra distance between the two graphs signature vectors. To find the outliers do a pairwise comparison between consecutive time stamps. Compute the Canberra distance between graph Gt to Gt+1. Note, the egonet for a node is the subgraph induced by the vertex and it’s neighbors. Because of the way the similarity is calculated anomalous graphs are identified by two consecutive anomalous time points in the output. For example, similarity score 1 is between graphs 1 and 2 and similarity score two is between graphs 2 and 3. If both similarity score 1 and similarity score 2 are found to be anomalous, the anomalous graph is then graph 2, since it is the one in common. Use the upper bound from the individual moving range threshold, and any distance above it is an anomaly.