Beyond outliers and on to micro-clusters: Vision-guided Anomaly Detection

October-2018

Wenjie Feng (fengwenjie@software.ict.ac.cn / wenchiehfeng.us@gmail.com)

1. General Information

- Python 2.7
- Version: 2.0
- Date: Oct.-2018
- Author: Wenjie Feng (wenchiehfeng.us@gmail.com)

2. Introduction

EagleMine is a novel tree-based mining approach to recognize and summarize the micro-clusters in the heatmap.

Inspired by the mechanism of human vision and cognitive system, EagleMine detects and summarizes micro-clusters (dense blocks) in the heatmap with a hierarchical tree structure (WaterLevelTree), and reports the suspiciousness score of each micro-cluster (based on the deviation from the normal).

For the large graph, the heatmap can be constructed with correlated features of graph nodes, and the micro-clusters correspond to node groups, some of them deviating from the majority and contain anomaly / suspicious objects with high probability.

EagleMine has the following properties:

- **automatic summarization**: automatically summarizes the heatmap derived from correlated graph features, and recognizes node groups forming disjointed dense areas as human vision does.
- **effectiveness**: detects interpretable groups, and outperforms the baselines, achieving better performance both in quantitative (i.e., code length for compact model description) and qualitative (i.e., consistent with vision-based judgment) comparisons;
- **anomaly detection**: detects spot, and even explain anomalies on

real data by identifying suspicious micro-clusters, and achieves higher accuracy compared with the state-of-the-art methods

- **scalable**: EagleMine is scalable, with nearly linear time complexity in the number of graph nodes, and can deal with more correlated features in multi-dimensional space.

Detailed information about the methods is explained in the following paper.

- Wenjie Feng, Shenghua Liu, Christos Faloutsos, Bryan Hooi, Huawei Shen and Xueqi Cheng. ” *Beyond Outliers and on to Micro-clusters: Vision-Guided Anomaly Detection*”, The 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 541-554, Spinger, 2019.

3. Installation

- This package requires python 2.7, type *install_libs.sh* to install some dependency library packages.
- For packaging (optional), type *make tar*.
- For demo (optional) for scratch graph features, type *make*.
- For extract graph node correlated feature, type *make graph*.

4. Input File Format

4.1 General heatmap analysis

heatmap (histogram) file:

The input file lists the non-zero data of the heatmap with the format "***x,y,val***", where ***x, y*** are the coordinate of the cell of the histogram for each dimension and ***val*** is the value of the cell.

histogram.out gives an example. In the example file, the first three lines started with '#' are comments for some basic information, that is, 1st line shows the shape of 2-dimensional histogram, and the 2nd line gives the corresponding real coordinate of each cell coordinate ***x***, and the 3rd line is the corresponding real coordinate of each cell coordinate ***y***, these two lines are used for visualizing the histogram. Then the followed lines are non-zero data records of the histogram.

Note: EagleMine can also handle general histogram-like data with the format: "***x,y,z,⋯,val***", where (***x,y,z,⋯***) is the coordinate of each histogram cell and ***val*** can be the weight or value for the cell.

node2hcel file (a node for a graph can also be a point):

The input file maps the node id to the corresponding histogram cell based on the correlated features of the node.

node2hcel.outs gives an example, where the 1st line is the comment for basic information, i.e., *#point*, and *shape*. The following lines are the records with the format "***node_id, x, y***", where ***x, y*** are the corresponding coordinates of the histogram cell for the for each dimension.

hcel2avgfeat file:

The input file gives the value of average features in a discrete histogram cell of each dimension.

hce2avgfeatl.outs gives an example with the format "***x, y, avgfeat_x, avgfeat_y, #npts***", where ***avgfeat_x*** denotes the average feature value of ***x***-dimension of all points falling into the cell (***x, y***), ***avgfeat_y*** has similar means for ***y***-dimension, #npts is the number of point in the cell (***x, y***).

4.2 Graph analysis

Correlated features file of graph nodes:

The input file lists all tuples in the node features of a graph. Each line corresponds to a tuple for feature values, which are separated by a comma or other delimiters. These node features can be following: *(out- / in-) degree, hubness, authority, pagerank, # triangles and the average neighbor degree etc.*

outd2hub_feature, ind2aut_feature are examples of the input file.

Edgelist file for a graph:

The input format of the graph is an edgelist format, which lists all tuples denoting the relationship between nodes. Each line in the edgelist file corresponds to an edge *e_{u,v}* between nodes *u* and *v* (direct edge for the bipartite graph, or undirect edge for the homogenous graph), which is separated by a comma or other delimiters.

We provide script *run_graph_feature.py* as an example tool to extract the above common features for the graphs.

example.graph is an example of the graph input file. To extract features (*out-degree vs. hubness; in-degree vs. authority*) for this example graph, type *make graph*.

Notice:

The graph feature extraction may have some tiny difference since the machine precision or other random factors for computing (like for SVD). And these differences could influence the histogram construction from features and later result of EagleMine.

5. Running

For any *run_*.py* script,

See help for usage and parameter explanation by typing:

```
python run_*.py -h    or    python run_*.py –help
```

See **Example** usage in *Makefile* or the doc string in each *run_*.py*.