Detecting Network Intrusion Anomalies through Egonet-based Data Mining with Apache Spark

Nathan Paik, Se Ho Kwak; Dr. Enyue Lu
Department of Computer Science, Pomona College; Department of Computer Science, Salisbury University

Abstract

Network intrusions often contain dangerous breaches to network security systems and their data. We design an anomaly detection system to identify network intrusions. Our proposed detection method is inspired by the use of egonets in the Oddball Algorithm but differs by the extracted features and the anomaly classification procedure. The detection process follows the generalized design: create a k-nearest-neighbors graph from a network dataset; extract each node's egonet's edge weights, number of edges, and total eigenvector centrality sum; compare each node's egonet's features through pairwise comparisons; and define a median "truth" line from the comparison and label nodes as anomalous based on their distance from the line. We have achieved an anomaly detection accuracy score of up to 92.9% with the eigenvector centrality score vs. edge weight feature comparison. We parallelize our algorithm by implementing Resilient Distributed Datasets in Apache Spark.

Network Data

We utilized different sizes of the NSL-KDD dataset network dataset (1,000 samples, 5,000 samples, 10,000 samples) to work with. We also limit each connection record to contain 18 attributes that include five basic attributes and thirteen secondary attributes.

Graph Creation

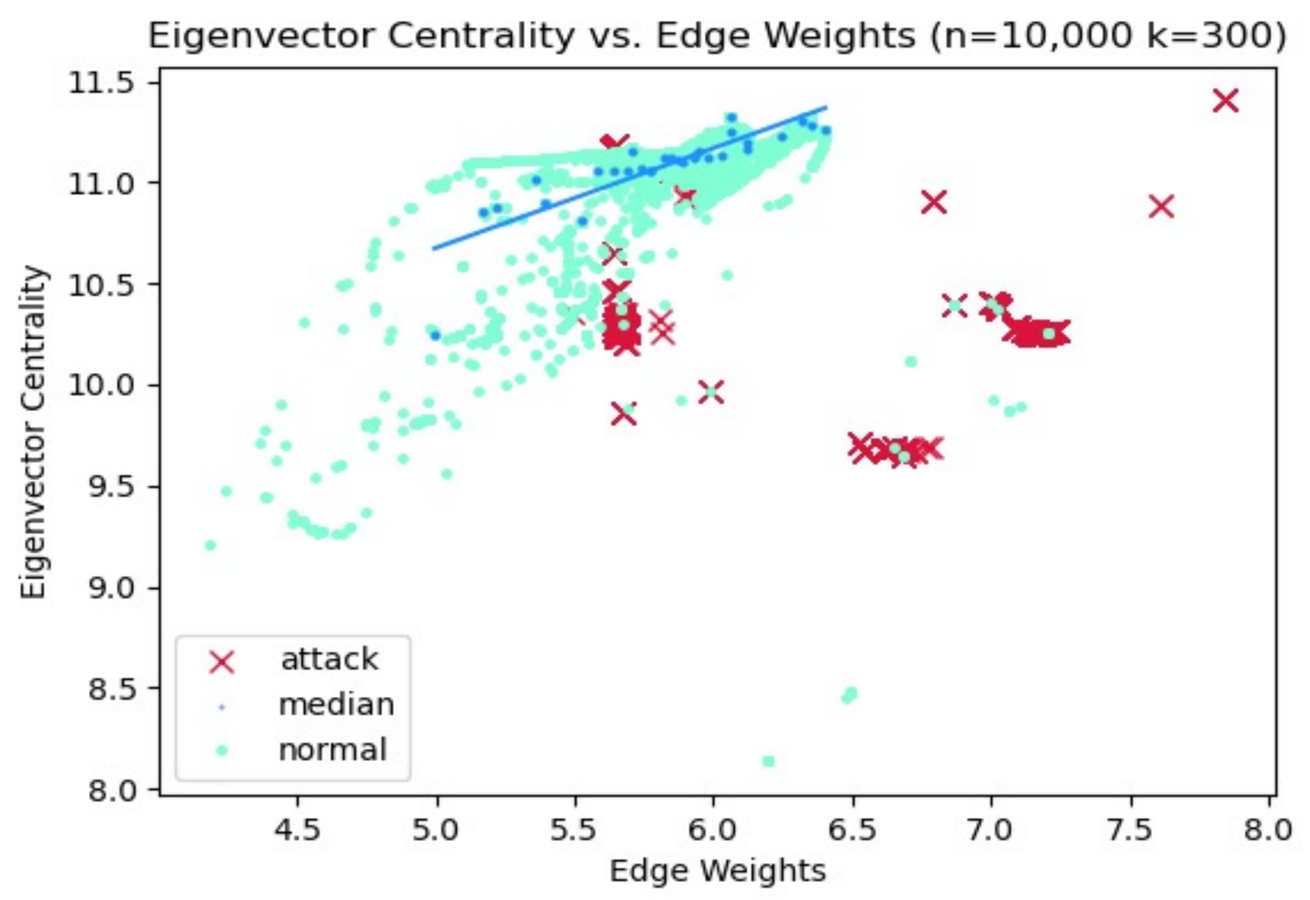
With the NSL-KDD dataset, each edge connecting two vertices represents the similarity measure between the two vertices. This similarity measure is based on two functions. First, the Euclidean distance is calculated between the two vertices, and then, the radial basis function is applied using the Euclidean distance as the vector distance to give the complete similarity measure.

$$w_{i,j} = \frac{1}{exp(\left\| \vec{v_i} - \vec{v_j} \right\|^2)}$$

This creates a complete graph of our dataset with each node connected to all other nodes with edges based on similarity. Therefore, to reduce the size of the graph to work with, we implemented a k-nearest-neighbors graph (kNN graph). Each sample takes a different k-value according to how many total nodes are in the graph.

Egonet Feature Extraction

From the kNN graph, we calculated each vertex's eigenvector centrality measure. We organized each node into their respective egonets, which are the collection of the node's immediate neighboring connections and all the edge connections amongst the neighboring nodes. We extracted different features to analyze for each egonet, which included each egonet's total number of edges, total sum of edge weights, and total eigenvector centrality measure.



Anomaly Score and Accuracy

Given each feature of the egonets, we paired features and analyzed them with each other. We performed three paired analyses per sample: total edge weights vs. total number of edges, total eigenvector centrality measure vs. total weight, and total eigenvector centrality measure vs. total number of edges.

We partitioned all the vertices into equal parts based off one of the measures from our pairwise comparison and used the central point of each partition as our "median" values for our ground truth line. We took the vertices that were farthest from the median truth line and labeled those as anomalies. Our accuracy is determined by the following equation where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives. The accuracy is represented by the following equation:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Comparison Type	**Eigenvector Centrality vs. Edge Weight**	**Eigenvector Centrality vs. Edge Count**	**Edge Weight vs. Edge Count**
1,000 nodes k=40	92.2%	90.6%	86.2%
5,000 nodes k=150	92.24%	87.44%	84.88%
10,000 nodes k=300	92.9%	85.76%	87.62%

Conclusions and Future Work

From the results the table above, we achieve up to 92.9% accuracy with our anomaly detection algorithm. Our best approach for anomaly detection accuracy is to apply the eigenvector centrality vs. total edge weight comparison. The graph to the left shows how attack types are separated from normal types after running our pairwise comparison. A possible future work includes improving our algorithm through supervised machine learning by enhancing our anomaly classifying technique to be more sophisticated rather than just taking the farthest points from the median "ground truth" line. Another possible work would be finding an algorithm to optimize the k-value for our kNN graphs. We successfully parallelized the detection process on a local cluster and the cluster in the high-performance computing laboratory at Salisbury University. Our runtimes for our detection algorithm on the 5,000-node sample and 10,000-node sample are 2.2 minutes and 20 minutes respectively using a 6-node cluster. We are in the process of optimizing the runtime and improving the efficiency on our GPU enabled clusters.

Acknowledgements

The work is funded by NSF CCF-1757017 under Research Experiences for Undergraduates (REU) Program. Nathan Paik and Se Ho Kwak did their research as REU students at Salisbury University in the summer of 2021.