Graph Anomaly Detection

NAMES : HEMANI CHAND, JATIN RATTAN and SUPERVISOR: SIMON KLUETTERMANN

Fig. 1. Graph Anomaly Detection is a Social Network

The field of graph anomaly detection has seen a drastic increase in interest in the recent years. While the definition of anomaly is not restricted, anomalies are the deviated data points that do not conform to the expected behaviour and anomaly detection is finding patterns in our data to identify and detect anomalies. Anomalies in graph-structured data are relevant in a variety of sectors, including economics, social networking, and network security. Anomalies, in many cases, may also have real and adverse impacts, for instance, fake news can create panic and chaos with misleading beliefs. Yet, in reality, many objects have rich relationships with each other, which can provide valuable complementary information for analysis. Graph anomaly detection aims to identify anomalous graph objects (i.e., nodes, edges or sub-graphs) in a single graph or multiple graphs. Graph anomaly detection with deep learning is expected to generate more fruitful results. Furthermore, due to the complexity of graph data (e.g., irregular topologies, relational dependencies,

Authors' address: Names : Hemani Chand, Jatin Rattan; Supervisor: Simon Kluettermann.

Manuscript submitted to ACM

node/edge types/attributes/directions/multiplicities/weights, enormous size, etc.), traditional anomaly detection approaches are unable to effectively address this problem. However, the introduction of deep learning has helped us to overcome these constraints.

ACM Reference Format:
Names : Hemani Chand, Jatin Rattan and Supervisor: Simon Kluettermann. 2022. Graph Anomaly Detection. 1, 1 (February 2022), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Anomaly detection in graph-structured data have gotten a lot of attention in recent years since it has a lot of applications in many industries. The network communication behaviour can be represented as a directed graph in the network intrusion detection issue, for example, where the hosts are nodes/vertices and the network connections are edges. Users are referred to as nodes/vertices in the social network scenario, while user interactions are referred to as edges. To identify fraud or spam activity in the graph, graph anomaly detection methods can be applied. Because of the structural information included in graphs, identifying anomalies in the graph becomes a more difficult challenge in non-Euclidean space graph anomaly detection, which tries to discover anomalous graph objects (i.e. nodes, edges, or sub-graphs) in a single graph. We define words that are common to all representation learning and structured data embedding approaches. Graph embedding is an approach that is used to transform nodes, edges, and their features into a vector space (a lower dimension) whilst maximally preserving properties like graph structure and information. Graphs are tricky because they can vary in terms of their scale, specificity, and subject.

Malicious nodes, edges, sub-graphs, and graphs can all be detected via anomaly detection. Anomalous node detection attempts to locate and detect malevolent objects in the real world. Anomalous edge detection is used since it is not just the items themselves that can be malicious, but also the relationships between them. Sometimes nodes or edges appear normal when examined individually, but become abnormal when viewed collectively, sub graph anomaly detection becomes critical in such scenarios. When an entire graph in a collection or database of graphs is an anomaly, anomalous graph detection is performed.

To detect anomalous nodes in plain graphs and attributed graph, the classic non-deep learning procedures are summarized first, followed by a more modern, sophisticated detection strategy based on representation learning. Firstly, we concentrate on OddBall, a non-traditional approach. We focus on neighbourhoods in this part, which is a sphere or a ball (hence the name OddBall) surrounding each node (the ego), that is, we consider the induced sub-graph of its nearby nodes, which is referred to as the egonet, for each node. Secondly, we will talk about the Deep Neural Network technique named as DONE (Deep Outlier aware attributed Network Embedding). In this section, we will discuss an unsupervised algorithm which reduces the effect of outliers in network embedding. Finally, we will compare the two strategies to see how DONE outperforms OddBall in a few different scenarios and will conclude from our observations.

2 ANOMALY DETECTION IN GRAPHS

Anomaly detection in graph data will be discussed in two categories: plain graphs and attributed graphs. A graph with attributed nodes and edges is one in which nodes and edges have characteristics associated with them. For example, users in a social network, they may have a variety of hobbies, work/live in different areas, have varying levels of education, and so on, whereas relational connections may have varying strengths, kinds, and frequencies. A plain graph, on the other hand, is made up of nodes and edges that is the connections between them.

The nodes or edges in the graphs may also be labelled numerically or categorically to denote their classes (e.g., normal or abnormal). As a result, we look at three different kinds of anomalous nodes: structural anomalies, community

anomalies, and global anomalies. Firstly, global anomalies, where only node attributes are considered. The node attributes that differ considerably from those of the other nodes in the network. Secondly, structural anomalies, in this we only consider graph structural information, that they have abnormal nodes with distinctive connectivity patterns. Lastly, community anomaly, in this we consider both node attribute and graph structural information.

2.1 Plain graphs

The structural information in real-world networks is represented by plain graphs. The graph structure has been widely used from many perspectives to discover anomalous nodes in simple graphs. We will be discussing traditional non-deep learning approach named as Oddball under plain graphs.

2.2 Attributed graphs

It is possible to have a richer graph representation for some types of data, in which nodes and edges have (non-unique) attributes. Social networks with user interests as properties, transaction networks with time, location, and amount as attributes, automobile shipments with visited regions, personal information, and other graphs are examples of this type. This class of attributed graph anomaly detection algorithms makes use of the structure and coherence of the graph's attributes to uncover patterns and abnormalities. Under this category, we will be discussing a deep neural network technique named as DONE.

3 TRADITIONAL NON DEEP LEARNING TECHNIQUE

Traditional non-deep learning techniques have been widely used in many real-world networks to identify anomalous nodes prior to recent developments in deep learning and other state-of-the-art data mining technologies. Many a times, one of the steps of deep learning techniques generally involves converting a graph anomaly detection problem to a traditional anomaly detection problem and then follow a series of steps that perform deep learning to detect the anomalies. This could also imply that even if we think that the deep learning techniques will overcome the traditional non deep learning techniques, it can never really happen because we will carry forward the basic idea.

3.1 OddBall

The core principle behind the entire OddBall anomaly detection technique is the OddBall paradigm, which is used in psychology research to display sequences of repeated stimuli that are infrequently disrupted by a deviant stimulus and the participant's reaction to this "oddball" stimulus is recorded. Similar to the deviant stimulus, anomaly detection adopts OddBall as a feature-based classical anomaly detection technique that works on plain graphs to capture unusual nodes. The name OddBall originates from the idea of focussing on the neighborhood of a node, that is, a sphere, or a ball around each node. It takes a graph as an input and outputs a list of outliers. OddBall's non-deep learning feature makes it very simple and outright. Whenever there is a large plain graph with a lot of weight, it is often difficult to discriminate. The primary concern being if such an effortless technique could detect anomalies in a vast graph for real. Oddball indeed discovers irregular nodes even in large graphs based on statistical features and several pattern discoveries that lead to power laws. The method finds structural anomalous nodes by employing statistical features which could be the number of neighbours of a particular node, the number of edges of a particular node, the total weight of the edges. OddBall method is not just used to detect anomalies in a variety of applications, but is also a significant part of the data cleaning process which is a very vital step of data pre-processing.

3.1.1 *Ego and Egonet.*

The node around which a neighborhood, a sphere or a ball is built is called an ego, and the induced sub-graph of its nearby neighboring nodes, is the egonet. While we choose statistical features to detect anomalies in this method, we could extract the ego and the egonet of a single node and study it individually. By doing this each node comes out to be a low dimensional feature space. The OddBall technique considers a node to be an anomaly if it i) forms local structures in the shape of near-clique or near stars, ii) has heavy vicinities in the sense that the total weight of all the links in the neighborhood of a node is very high (heavy vicinities), or iii) has a single very heavy link with one of the neighbors (dominant heavy link). In simpler terms, if the neighborhood of an ego in a network is significantly different from those of others, it is considered as an anomalous node.

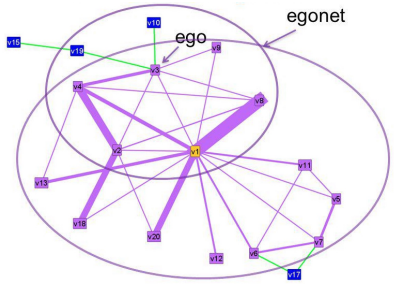
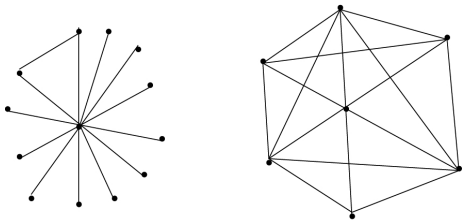


Fig. 2. Glimpse of an ego node and egonet

- Near cliques or near stars: Nodes whose neighbors are either very strongly connected (near-cliques) or not connected (near-stars) appear unusual and are classified as anomalous. These structures are illustrated in figure 3. For example if a malicious user on Facebook tries to befriend me and the user's profile says that the user is studying in TU Dortmund. Now having no mutual friends or all mutual friends should be considered suspicious and lead to the malicious user being an anomaly. Cliques and stars generate the extreme values of a power-law relationship(discussed later) between the number of nodes in an ego-net and the number of edges. Therefore, anomalies can be detected by fitting a power-law curve to the network and analyzing the residuals for significant deviation from the expected relationship.



(a) Near star (b) Near clique

Fig. 3. Near-clique and near-star structures

- Heavy vicinities: This can be understood with the help of an example where a malicious user is trying to connect to multiple users in order to extract information to commit some kind of fraud like credit card fraud. Now calling

multiple people where the number of calls is the weight will lead to a multiple weights with all links around which will result in a high total weight.

- Dominant heavy link: Similar to the above scenario if the malicious user already has a target in mind after doing the background check, we can expect multiple calls to a single user which will cause a single dominant heavy link.

3.1.2 K-step neighborhood and K-value.

The "k step neighborhood" is defined as the collection of node i, all of its k-step-away nodes, and all of the connections across these nodes in the induced sub graph, which is also known as the "egonet." While we aim to detect abnormalities in social networks and do social network research, an egonet is typically a node's 1-step neighborhood (where we choose k = 1 in social network analysis). This does not mean that the value of k cannot change in other scenarios. The value of k in k step neighborhood can be assigned any value. However as observed from various previous research studies, taking a value of k > 1 has not resulted in any additional relevant information. As a result, selecting a basic k value of 1 is mostly viewed to be wise.

3.1.3 Feature Extraction.

The most critical aspect of OddBall lies under feature extraction. If the statistical features are properly chosen, OddBall works effectively. However the hard truth is that it is often hard to select appropriate features in a real world that has uncountable features and since the domain specialists are constantly generating new statistics. Statistical features that are quickly computable and produce readable patterns should ideally be selected. A set of four features have been highly successful in spotting patterns. These are as follows:

- The number of neighbors of node i which is also known as the degree N_i,
- The number of edges in an egonet E_i,
- The total weight of the egonet W_i ,
- The principal eigenvalue of the weighted adjacency matrix of egonet $\lambda_{(w,i)}$.

Different kinds of grouping can be applied to the above features in order to see what yields us with the best resulting patterns to flag anomalies. This could be grouping number of nodes and edges as features to get near cliques or stars or any other kind of grouping that could lead us to fruitful results.

3.1.4 Laws.

Pattern recognition is inevitable and is governed by a few laws. OddBall identifies and employs multiple power law patterns that drive the ego-nets of all nodes, i.e. subgraphs of nodes and their neighbors, for anomaly identification. OddBall calculates a Least Squares fitting line for a power law first, then calculates the anomalousness of each node based on its distance from the fitting line. There are a total of four laws namely Egonet Power Density Law, Egonet Weight Power Law, Egonet λ_w Power Law and Egonet Rank Power Law. These laws detect patterns among graphs that finally lead us to recognising anomalies. Egonet Density Power Law detects near-cliques and stars. Egonet Weight Power Law detects recurrences of interactions (heavy vicinities). Egonet Rank Power Law detects single dominating heavy edge or the one strongly connected pair (dominant heavy link). These laws are given below:

Egonet Density Power Law

EDPL is the abbreviation of this law which takes into account the number of nodes N_i and the number of edges E_i of a given graph G_i and the law is given by:

$$E_i \propto (N_i^{\alpha}), 1 \leq \alpha \leq 2.$$

Egonet Weight Power Law

EWPL is the abbreviation of this law which takes into account the total weight $W_i$and the number of edges E_i of a given graph G_i and the law is given by:

$$W_i \propto E_i^{\beta} \ , \beta \geq 1.$$

Egonet λ_w Power Law

ELWPL is the abbreviation of this law which takes into account the principal eigenvalue $\lambda_{w,i}$ of the weighted adjacency matrix and the total weight W_i of a given graph G_i and the law is given by:

$$\lambda_{w,i} \propto W_i^{\gamma}, 0.5 \leq \gamma \leq 1.$$

Egonet Rank Power Law

ERPL is the abbreviation of this law which takes into account the rank R_i, $jandtheweightW_{i,j}$ of edge j of a given graph G_i and the law is given by:

$$W_{i,j} \propto R_{i,j}^{\theta}, \theta \leq 0.$$

3.1.5 Anomaly score.

Let y_i be the y-value of node i and x_i be the x value of node i when we have a particular feature pair f(x, y). The anomaly score of node i given the power law equation y = Cx_i^{θ} is given below.

$$\text{AS(i)} = \frac{\max(y_i, Cx_i^{\theta})}{\min(y_i, Cx_i^{\theta})} * \log(|y_i - Cx_i^{\theta}| + 1)$$

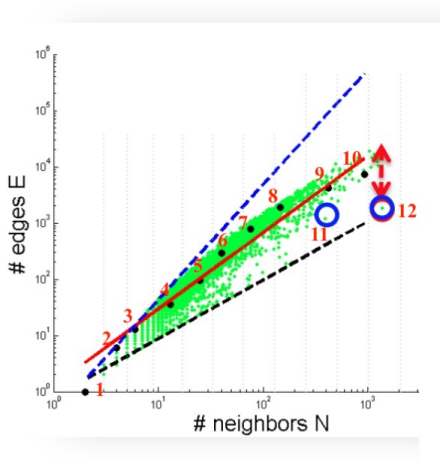


Fig. 4. Graph depicting outlier score calculations

The score of the outlier could also be measured as the distance to the fitting line where we see how much the does the calculated value deviate from the actual value(as it can be seen in figure 4, distance between point 12 and the fitting line gives us the anomaly score). Doing this calculation, anomaly score (AS) will be 0 when actual value y_i is equal to the expected value Cx_i^{θ}(points 2,3,4,5 will have as anomaly score close to 0 as all these points lie almost on the red fitting line in figure 4). This simple strategy not only aids in the detection of outliers, but also allows the nodes to be sorted according to their anomaly scores. This approach, however, is prone to missing some outliers and so may produce false negatives. If it is the case that there are some points that are far from the rest of the points, but are still close to the fitting line (if hypothetically point 1 in figure 2 was really far away from all other points, it would still be

near to the fitting line and would not have a high anomaly score and hence would not be anomalous). In such a case when x and y values are severe, this will clearly result in false negatives(should have been an anomaly but is actually a normal data point).

Overall OddBall builds its solution based on the analysis of the egonetworks by extracting all these egonetworks, adopting feature extraction and selection, and then using the principle laws to identify anomalies. One of the drawbacks of OddBall is the process of choosing appropriate features, which are also quite critical in anomaly detection. If the features are wrongly chosen it would just lead to a waste of time and we will land up with results that are worthless(not to forget with efforts gone in vain). The other drawback of this technique is that it relies heavily on near clique or near star structure which cannot be used to detect anomalies in bipartite or heterogenous graphs. [1]

4 DEEP NEURAL NETWORK TECHNIQUE

Deep neural networks, which can be used to learn multiple levels of feature representations, has achieved successful results in different fields. Autoencoder(AE) and deep neural networks are two deep learning methods that provide a robust foundation for learning data representations. The AE are the types of unsupervised learning technique used primarily for getting a representation of a given data. It has three major parts namely, encoder, latent space and decoder. Firstly, the encoder is responsible for reducing the dimension of the input data to minimize the computation required to learn the pattern and feature of the data. Secondly, the latent space is called the lower dimension code or the low dimensional representation that acts as the intermediate step. Finally, a decoder is responsible to transform the encoding back to the input dimension and produce the most similar or closest generation. The task of mapping the nodes of a network (or graph) to a lower dimensional vector space is known as network embedding. These embedding techniques often take use of the fact that node attribute values are significantly associated with the network's link structure, and therefore give additional information for network representation. Real-world networks, on the other hand, have nodes that infringe on the community's property. Such a node may have edges with nodes from other communities at random, or its properties may be comparable to those of nodes from other communities. Community outliers are the names given to these nodes. Therefore, to detect the outliers while generating node embedding, we propose an AE based deep architecture (DONE) for unsupervised network embedding that minimizes the influence of outliers. [3]

4.1 DONE

It is AE based deep architecture (DONE) technique to minimize the effect of outliers for network embedding, in an unsupervised way. The goal of DONE is to find global, structural, and communal anomalies in attributed graphs. This study calculates three anomaly scores for each node, indicating the likelihood of one of three scenarios:

- It shares similar attributes with nodes in other communities.
- It connects with other communities.
- It structurally belongs to one community but has attributes that follow the pattern of another community.

DONE uses two parallel autoencoders: a structural AE and an attribute AE. Both AEs are trained by reducing reconstruction errors and maintaining homophily, which assumes that all linked nodes in the network have the same representation. Because their structure or attribute patterns do not conform to the conventional behavior, nodes showing the established characteristics are difficult to reconstruct while training the AEs, resulting in increased reconstruction mistakes. Each encoder and decoder have L layers, and both autoencoders employed a Leaky ReLU nonlinearity function with a negative input slope, where ReLU is an activation function defined as the argument's positive component.

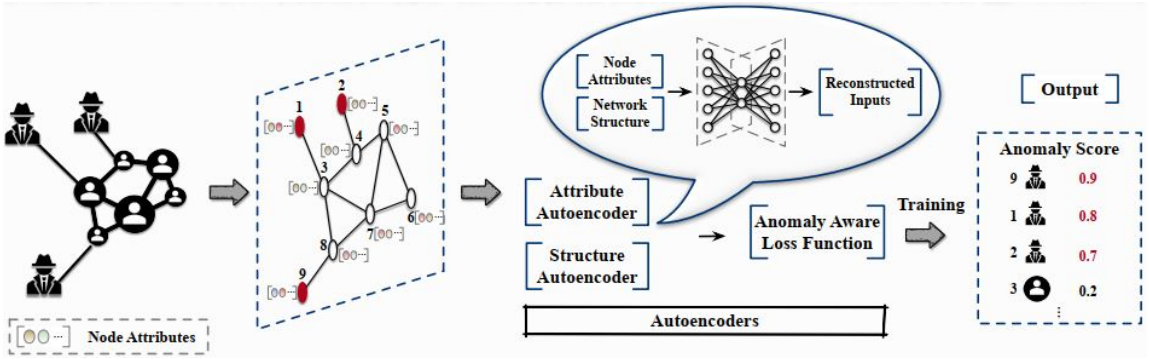


Fig. 5. Deep neural network based approach (DONE).

The structural outlier, attribute outlier, and combination outlier scores for each node correspond to the three categories of outliers, denoted as o_s^i , o_a^i and o_{com}^i corresponding to structural, attribute and combined outliers respectively for node i, i = 1,.....,N. The set of all the outlier scores is denoted by O. We assume that the overall outlier score for each kind of outlier in the network is constant. In a perfect network with no outliers, the outlier scores of all nodes are identical. Accordingly, we formulate the loss function for this approach. DONE creates a loss function that is anomaly-aware and has five distinct loss, namely proximity loss, homophily loss both for structural AE and attribute AE and the combining of structural and attributes. Firstly, we must compute proximity loss since we want to retain the network's distinct orders of proximities. Because the input to the structural autoencoder captures a node's local neighborhood, minimizing this reconstruction loss preserves the network's higher order proximity.

$$L_{str}^{Prox} = \sum_{i=1}^{N} \log(\frac{1}{O_i^s}) \|x_i - \hat{x_i}\|_2^2$$

Secondly, next component of the loss function is used to preserve homophily in networks. Edge-connected nodes have a tendency to behave similarly, and they should be near in the embedding space as well. Again, structural outliers should contribute less to homophily loss since they have links to nodes from various communities at random.

$$L_{str}^{Homo} = \sum_{i=1}^{N} \log(\frac{1}{O_i^s}) \frac{1}{|(N(i)|} \sum_{j \in N(i)} \|h_i^s - \hat{h_i^s}\|_2^2$$

For the attribute autoencoder, the following two losses may be stated with a similar reasoning.

$$L_{atr}^{Prox} = \sum_{i=1}^{N} \log(\frac{1}{O_i^a}) \|c_i - \hat{c_i}\|_2^2$$

$$L_{atr}^{Homo} = \sum_{i=1}^{N} \log(\frac{1}{O_i^a}) \frac{1}{|(N(i)|} \sum_{j \in N(i)} \|h_i^a - \hat{h_i^a}\|_2^2$$

Lastly, the connection structure and node properties of a node in a network are highly associated, according to the homophily property. As a result, it's critical to use one to compliment the other. Although we are obtaining embedding that match to the network's structure and properties, it is necessary to regularize them for each node. As a result, we create the final loss function component.

$$L^{Com} = \sum_{i=1}^{N} \log(\frac{1}{O_i^{com}}) \|h_i^s - h_i^a\|_2^2$$

$$L^{DONE} = \alpha_1 L_{str}^{Prox} + \alpha_2 L_{str}^{Homo} + \alpha_3 L_{atr}^{Prox} + \alpha_4 L_{atr}^{Homo} + \alpha_5 L^{Com}$$

$$\alpha = coefficient.$$

The anomaly scores of each node are defined by minimizing the sum of these loss functions, and the top-k nodes with greater scores are recognized as anomalies. [2]

5 COMPARISON

While traditional techniques like OddBall do not take into account community outliers, DONE which deep neural network based technique stands out in this situation. Most network embedding algorithms also do not consider collective anomalies when they generate the node embeddings. To counter this, many algorithms have been proposed until now. However, these algorithms are complex and expensive. Also, these counter algorithms do not scale for large real world networks. Most importantly, these algorithms do not capture non linear behaviour which most real networks exhibit. The false negatives problem in case of OddBall can also be reduced using deep learning technique which handles high dimensional intricate data very effortlessly. DONE which is an unsupervised learning algorithm has been able to solve all the problems by giving an efficient performance. Overall in cases where both techniques (OddBall and DONE) could be used to solve a single anomaly detection use case, DONE is expected to give a better accuracy.

6 CONCLUSION

In this work, we propose two techniques, detecting and minimizing the effect of outliers. In the first technique we proposed the idea of OddBall that focuses on a node's immediate surrounding nodes in order to detect anomalies by identifying deviations while choosing some features on the basis of pattern recognition which is based on a few laws. In the second technique we propose an outlier resistant network embedding approach (DONE) based on adversarial learning. Both our methods are fast but deep learning techniques are generally more scalable in terms of computing, and they can handle larger data sets.

Some of our personal future expectations involve having prior knowledge about each graph before tackling with it in order to detect anomalies and proper management of imbalanced data sets. Having prior domain knowledge can prove to be extremely useful for example in case of OddBall (which relies heavily on statistical features). If we have a prior idea of what features could be beneficial, it can be of utmost importance as it will lead to drastically improved results. Also imbalanced data poses a problem in the sense that anomalies are often rare and their proportion is far smaller compared to the normal objects. As a result these anomalies are ignored which in the longer run leads to a sub-optimal and depleted performance. So in order to tackle this data imbalanced problem we could either undersample our majority class(which could lead to huge data loss) or oversample our minority class(which is still a better option since it does not cause much data loss). We hope to expand both these methods in the future of dynamic and multiplex networks, which are common in many real-world applications.

REFERENCES

[1] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. OddBall: Spotting Anomalies in Weighted Graphs. 410–421. https://doi.org/10.1007/978-3-642-13672-6_40

[2] Sambaran Bandyopadhyay, Lokesh N, Saley Vivek, and M. Murty. 2020. Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding. 25–33. https://doi.org/10.1145/3336191.3371788

[3] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Quan Sheng, and Hui Xiong. 2021. A Comprehensive Survey on Graph Anomaly Detection with Deep Learning.