Anomaly Detection for Univariate Time-Series Data

Krati Nayyar KRATI14@VT.EDU
Virginia Tech

Saket Vishwasrao SAKET02@VT.EDU.EDU
Virginia Tech

Saurabh Chakravarty SAURABC@VT.EDU
Virginia Tech

Sina Dabiri SINA@VT.EDU
Virginia Tech

Abstract

Some of the biggest challenges in anomaly based network intrusion detection systems have to do with being able to handle anomaly detection at huge scale, in real time. The incoming data stream is homogeneous, containing different anomalous patterns along with a large amount of normal data. We pose the problem as that of detecting the anomaly in the data stream in real-time. We define an approach to classify the pattern in the sliding window based data stream and attempt to identify the class of anomaly a pattern belongs to. These classes contain the normal class along with the anomalous ones. If the data detected in the sliding window is classified as belonging to one of the anomalous patterns, we dispatch it to the respective algorithmic handler for predicting which data points in the sliding window are anomalous, based on the model that we have trained in the handlers. Experiments show that we are able to get good reasonable accuracy based on our methods. There is always that chance of generating false positives and considering the problem domain, we conclude that it is fine to generate a few false alarms rather than being silent on an actual anomaly. We use the F1 score attributes like precision and recall to measure our experimental results quantitatively.

Virginia Tech CSx824/ECEx424 final project report.

1. Introduction

Anomaly detection refers to the problem of finding patterns in data that do not conform to the normal behaviour. In other words, anomaly detection is a kind of technique that seeks to specify a region representing normal behaviour and declare any observation in the data which does not belong to this normal region as an anomaly. This problem is also referred to novelty detection, outlier detection, one-class classification, exceptions, aberrations and surprises. However, anomalies and outliers are the most common terms in the literature of anomaly-based intrusion detection in networks(K. & K., 2013). Anomaly detection is applied to a broad spectrum of domains including IT, security, finance, vehicle tracking, health care, energy grid monitoring as well as e-commerce. In the real world, several studies investigated the role of anomaly detection algorithms to explore an anomalous traffic pattern in a computer network, an anomalous MRI image, anomalies in credit card transaction data as well as anomalous readings from a space craft sensor(Chandola V. & V., 2009).

Anomaly detection can also be applied to find unexpected patterns in time series data. A time series is a sequence of data points, typically consisting of successive measurements made over a time interval. In this study, we strove for developing a framework for a univariate time series data set. We used publicly available dataset released by Yahoo. A value and a label (0 for normal points and 1 for anomaly points) are assigned to each data points. For specific formulation of the problem, we need to evaluate various aspects of our data including types of anomalies, data labels and output of anomaly detection.

In the literature, types of anomalies in time series are categorized into following groups: (1) Point Anomalies. If an

individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly, (2) Contextual Anomalies. If a data instance is anomalous in a specific context (but not otherwise), then it is termed as a contextual anomaly, (3) Collective Anomalies. If a collection of related data instances is anomalous with respect to the entire data set, it is termed as a collective anomaly. This classification indicates that extracting different anomaly patterns in our data set is vital step before developing any algorithms. As our training data set is labeled as anomaly versus normal, we are going to focus on supervised anomaly detection. The output of our anomaly detection are labels, which assigns anomaly or normal to each test instance.

In this project, we envisioned to develop a framework for detecting anomaly points in our data set. As the values of data points in different files are not compatible with each other, we scaled them in the similar format. Scaled data provided an opportunity to combine all data sets. In the next step, we plotted data set to visualize different anomaly patterns. We classified each time-window data set to four categories, three anomaly groups and one normal group. For example, if there were no anomaly points in a portion of data set, we assigned a normal label to that portion. Thus, we developed a multi-classification algorithm, which defined the category of each time-window. Then, for each of anomaly time-window data, we applied a specific algorithm to detect the anomaly points in that time-window data. In the last step, we reported the accuracy of the algorithms by two indexes: Prediction accuracy and F-score. The later index is more valuable as it takes into account the effects of false positive points and false negative points.

2. Our Approach

Analysis of the dataset suggests that each window in which an anomaly occurs can be classified into 3 distinct patterns. Figure 1 shows a threshold based pattern. Such patterns have values of the anomalous points significantly larger than the global average. Section 4.1 discussed later suggests our approach in detecting anomalies in such windows. Figure 2 shows the anomaly detected due to change in the past pattern to a completely new pattern. Section 4.2 describes this. Figure 3 shows anomalous patterns where anomaly occurs due to change in frequencies. Such patterns are detected using a sliding window based algorithm that correlates the spectrum of successive windows as described in Section 4.3. From our observations, we find that a single anomaly detection algorithm in general does not give good results as different algorithms have different strengths. So we do a preprocessing on the data to identify which pattern it belongs to. Due to the limitations of time we have not created a machine learning algorithm for classifying a window to a given class, but we label each window manually. The rest of the paper is organized as follows. We discuss our strategies for scaling the data in Section 3.1 and then we propose various algorithms as suggested earlier. Section 5 provides the quantitative analysis of the results of the algorithms.

3. Dataset

We used a real world data set which is created by Yahoo! consisting of Web requests time series statistics. This data set is a combination of real world and synthetic data sets. Real world data set was considered for this project. A value and a label (0 for normal points and 1 for anomaly points) are assigned to each data point. For specific formulation of the problem, we evaluated various aspects of our data including types of anomalies, data labels and output of anomaly detection.

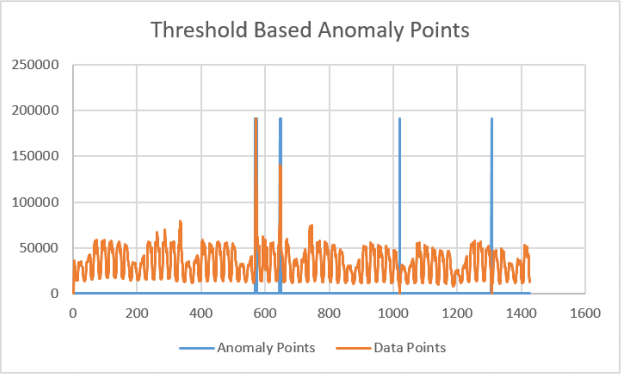

Figure 1. Threshold Based Anomaly points

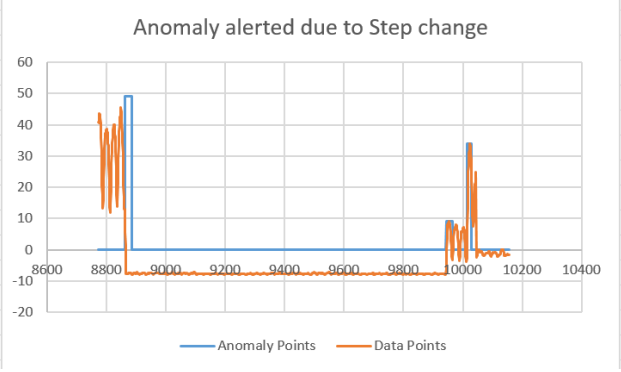

Figure 2. Anomaly alerted due to step change

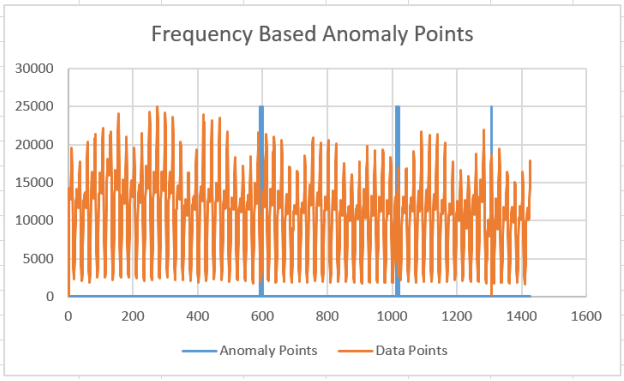

Figure 3. Frequency based anomaly points

3.1. Scaling the data

The dataset consisted of 67 files and each file had a time series data. As these files were not scaled on the same magnitude, in order to process them together , they had to be scaled to a fix magnitude. The following steps were followed for scaling the Yahoo dataset:

- Time series in each file was centered about the mean
- Average value of all the maximas was calculated
- Average value of all the minimas was calculated
- Scaling factor of each file was set at $(maxima + minima)/2$
- Concatenation of all files into single file for training.

4. Algorithms Proposed

The classification algorithm classifies the data pattern into three classes. Depending upon the classification, the algorithm corresponding to the class would be run on the time window dataset to detect the anomaly points. The algorithms for the classes are:

1. Point anomaly algorithm for Threshold based anomaly.
2. Step Change anomaly detection for anomaly alerted due to sudden transition in the mean.
3. Frequency based algorithm for anomaly points which are caused due to change in periodicity in the dataset.

4.1. Point anomaly algorithm

In this section, we only focus on the portions of the data set which contain the threshold based anomaly points. Many approaches in the literature were based on fitting statistical model to the given data set, then apply a statistical inference test to determine if an unseen instance belongs to this model. The main assumption for these approaches is that normal data instances occur in high probability regions of the fitted statistical model whereas anomalies occur in the low probability regions of the model(Chandola V. & V., 2009). Thus, we applied the Kolmogorov-Smirnov test to compare our normal data set with a variety of well-known probability distribution. The null hypothesis (H_0) and the alternative hypothesis (H_a) are as follows:

H_0 = Our normal data comes from an assumed probability distribution, H_a = It does not come from the assumed probability distribution.

Although many statistical anomaly detection techniques assumed that the data is generated from a Gaussian distribution (Grubbs, 1969; Stefansky, 1972), we investigated other famous probability distributions such as Weibull and Exponential. However, as the P-value for all assumed distributions was approximately zero, we could not reject the null hypothesis. Consequently, the algorithm for detecting threshold based anomaly points needed to be independent of any assumed probability distribution for data set.

The main concept in the proposed algorithm is that to extract the behavior of anomaly points in comparison to normal points. Like many machine-learning based methods, the algorithm is divided in two parts: training model and test model. In the training model, first of all, the normal data is separated from anomaly data. Then, the absolute distance of all anomaly points from the average of normal data are calculated and assigned to a vector. This vector is employed for computing a score function for unseen test instances in the test data model. For labeling unseen test data instances to a normal or anomaly , a threshold value needs to be optimized. In this algorithm, the threshold value shows the percentage of all anomaly points in the training data which are either less than average of local maxima or greater than average of local minima. The rationale behind such a selection for threshold value is to find the portion of anomaly points which behaves more similar to normal data in comparison to the anomaly data. The training model can be summarized in the following three steps:

1. Separate normal data from anomaly data
2. Find the average of normal data, and plug it into model.mean
3. Find the absolute distance of each anomaly point to model.mean, and plug it into the vector of model.anomalyDistance
4. Find a threshold value α with following computation: X_1= Number of positive anomaly points less than av-

erage of normal local maxima X_2= Number of negative anomaly points greater than average of normal local minima

$$\alpha = \frac{(X_1 + X_2)}{Numberof Anomalypoints} \tag{1}$$

In the test model, we consider a score function associated to each unseen test instance (η). We declare η to be anomalous if its score is greater than α. For finding the score value, the absolute distance of the unseen test instance to model.mean needs to be calculated. We called this value TestDistance. If this distance is greater than each value in model.anomalyDistance, the test instance receives a penalty. The following score function maps the score value to [0,1]:

$$F(\eta) = \frac{1}{N}\sum_{i=1}^{N}(II)_{TestDistance(\eta) \leq Model.AnomalyDistance(i)} \tag{2}$$

Where N is the number of anomaly points in the training data, and II is the indicator function. The larger value of F(η) increases the possibility that η is anomaly. The three following steps illustrate the test model:

1. Find the absolute distance of each unseen test point (η) to model.mean, and plug it into TestDistance
2. Find score function for the test instance, F(η).
3. If F(η)>α, then η is anomaly. Otherwise, η is normal.

4.2. Identifying anomalies through statistical methods

First Attempt at the algorithm This algorithm is designed for data which had sudden transition in the mean between two consecutive data windows. This transition is considered to be anomalous.

This is a type of supervised algorithm which uses the technique of k-nearest neighbors as it takes the mean of k points in the time-window.

The algorithm is divided into training the model and testing the model obtained. The training algorithm is as follows:

- Window size for the running window time frame is initialized
- Set $Model.threshold = max(dataPoints)$
- For every window, calculate the average of the data points in the window
- If the new data point included in the window frame is an anomaly, take the difference between the present and previous $average = \epsilon$
- If(ϵ<threshold), update threshold to the new value
- $Model.windowSize = windowSize$ for maximum accuracy

Algorithm for the predictions:

- Window frame of Model.windowSize is considered
- If the difference between two consecutive average is less than Model.threshold, mark the point as anomaly
- For increasing the accuracy of the algorithm:
- Take the weighted average of the data points in the window frame e.g. Squared index weights, Exponential weights
- Multiplier was used for taking a better range of the threshold
- Multiplier acts as a function of the variance and we continue to tune this function

Based on the pattern of the data that we believed could be used for detecting anomalies, we started off with a pure threshold based pattern. In our training approach, we were using a k size window of data. We continue to have this sliding window of size k and compute the mean of the values in this window. For each anomalous point, we deduct the new mean from the previous mean and set that as a threshold. We vary the window size and select the lowest threshold.

This approach was not giving us a good result since there were a lot of false positives. As we were selecting the least threshold during our training, it was not working well for the test set since the data variances in the test dataset were beyond the threshold in a lot of cases and were predicted as anomalies, but were not anomalies. We concluded that using a hard threshold won't work.

Second attempt

After researching some more on this, we had to make this threshold a variable value that is a function of the variance of the data, if we consider the data in the sliding window to be distributed normally.

The distribution of the data points does not need to actually be normal, but it is better if it is at least normal-like. For our approach, since the pattern that we are targeting will have the non-anomalous data in the normal form, we can use the statistical approach of fitting the data in a normal distribution and detect outliers this way. A typical formulation for this technique is to find the mean and variance for the feature of the data in the window, then define the probability p(x) of a combination of a new data point, that just

follows the sliding window. An anomaly occurs whenever $p(x) < \epsilon$.

Determining ϵ

The algorithm is fit to negative examples (non-anomalies). But ϵ is determined from the cross-validation set, and is typically selected as the value that provides the best F1-score.

To compute the F1-score, we needed to know what is anomalous and what is not; that is true positives are when the system predicts an anomaly and it actually is an anomaly, false positives are predicted anomalies that actually aren't and so on. We determined the best ϵ by analyzing the anomalous points and find a tolerance range as a function of the variance that determines whether a point is anomalous or not.

The following diagram illustrates the point in a better way. Figure 4 shows the distribution of normal and anomalous points from the perspective of probability densities.

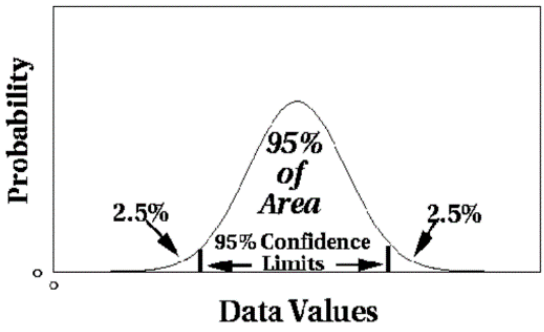

Figure 4. Relation between Variance and Confidence between data points(Gao)

As per the diagram above, we just need to find a variation based threshold that we can train that will give us the model parameter that we can use to predict anomalies. For our case, if the probability density of the new point falls outside the confidence limits, we mark that point as anomalous.

4.3. Frequency Based Anomaly Detection

This class of anomaly detection techniques deals with the patterns in time series where the anomaly occurs due change in periodicity or sudden change in periodic data trend. One of the methods of detecting such a trend is to find the correlation coefficient between the frequency spectrums of successive windows. Windows with similar frequency components will have high correlation(et al Giorgio Glacinto, 2008). Figure 5 shows the actual data and Figure 6 plots the correlation between successive windows.

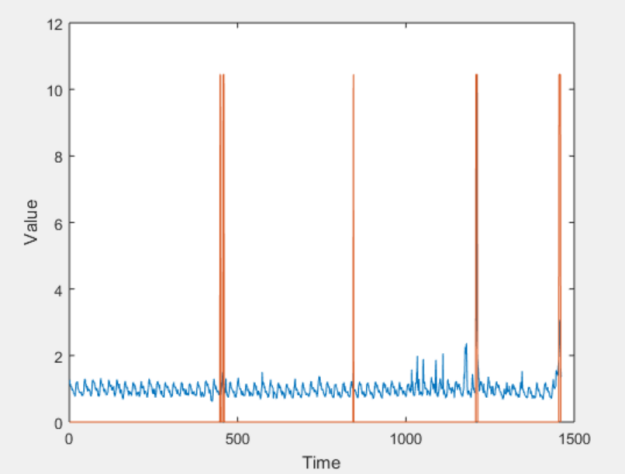

Figure 5. Anomalies corresponding to periodicity change in data points where Blue represents the data points and Red are the Anomalous points

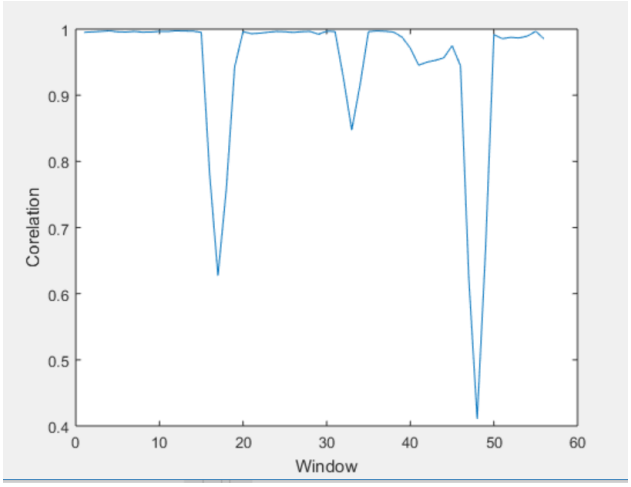

Figure 6. Three out of four Anomaly window detected

5. Experimental Results

Accuracy measurement

We started with the method of calculating accuracy based on whether our predictions matched the labeled anomalous points. We were getting pretty high accuracy using this methodology, but it was flawed. This methodology doesn't take into account the facts that there were a lot of misclassifications since we predicted a lot of normal data points as anomalous. These false positives are bound to reduce the effectiveness of the system in production because a network operations administrator will start to ignore the alerts

Table 1. Truth Table Matrix for F1 Score

	POSITIVE	NEGATIVE
POSITIVE	TRUE POSITIVE	FALSE POSITIVE
NEGATIVE	FALSE NEGATIVE	TRUE NEGATIVE

for anomalous data because of the huge number of false positives. We needed to use a metric that could quantitatively take the misclassifications into account and penalize the accuracy score accordingly. We came across such an exact methodology named the F1 score using the truth table matrix.

This method identifies the results in comparison with the ground truth and slots them in each of the 4 different buckets and calculates the following metrics.

Precision can be defined as the classifiers exactness. A low precision can also indicate a large number of False Positives

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (3)$$

Recall can be defined as classifiers completeness. A low recall indicates a large number of false negatives.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (4)$$

The F1 score is defined as

$$F1 = \frac{2 * precision * recall}{precision + recall} \quad (5)$$

The truth table matrix, Table 1, shows the comparison between the predictions and the ground truth.

5.1. Point anomaly algorithm

This algorithm worked well on threshold based anomalies as it was assumed to be. This would only provide the right anomalous points for the time window classified in the threshold based class.

The accuracy for the algorithm was 0.98 and Fscore was 0.68 for the dataset.

5.2. Anomalies through statistical methods

The first part of the algorithm which included step based anomaly detection using weighted average of the data points in the time window gave an accuracy of 0.92 for the dataset with sudden transitions in mean and due to the some points having false positives F1 Score turned out to be 0.68.

5.3. Frequency Based Anomaly Detection

The evaluation is slightly different from the rest of the methods in the sense that it does not give us a particular point of anomaly, but a small window within which the anomaly lies. The window size used in evaluation is 50 and successive windows are 25 units apart (sliding window).

There are no false positives but, this method detects anomalous windows with 51.7% accuracy which is equal to

$$\left(\frac{number\ of\ anomalous\ windows\ detected}{total\ anomalous\ windows}\right)$$

This method performs well for highly periodic data, with major changes in frequency but cannot detect anomaly points where the data is not perfectly periodic. Also the selection of window size matters because the closer it is to the period of the series, the better the performance of the algorithm.

6. Conclusion

The paper discusses the algorithms for different types of anomalies present in the Yahoo dataset along with the quantitative analysis of the algorithms proposed. The multi classification algorithm along with separate algorithms for each type of anomaly detected in the time window worked well for the real world dataset.

7. Future Work

In this project, we explored three anomaly patterns although other types of anomalies may exist. In pursuit of more investigation on available data, we lead to detect more anomalous patterns. On the other hand, applying a variety of algorithms to our data set results in having a more complex model.One way is to fuse various algorithms and propose a unique algorithm, which is able to detect more than one anomalous type. However, integrating algorithms may increase the chance of missing a portion of anomalies. Thus, future research in this area could focus on finding the optimized trade-off between the complexity of models and the number of applied algorithms which brings about higher accuracy. A better, but a slightly complicated algorithm as continuation of Frequency based anomaly detection would be to use the spectrum to extract the top n harmonics from the window and use this as a n-dimensional vector for performing clustering. Windows that fall in cluster of larger size can be considered as normal while the windows in other group will be anomalous.We assume that the data is classified into one of the 3 classes.An algorithm could be built that given a window, it classifies the window in one of the classes.

Acknowledgments

This project would not have been possible without the constant guidance of Dr. Bert Huang and we are very grateful for all the support provided by him during the duration of this project.

References

Chandola V., Banerjee, A. and V., Kumar. Anomaly detection: A survey. pp. 41, 2009.

E., Grubbs F. Procedures for detecting outlying observations in samples. In *Technometrics*, pp. 11, 1969.

et al Giorgio Glacinto. Anomaly detection with score functions based on nearest neighbor graphs. In *Information Fusion 9.1*, pp. 69–82, 2008.

Fen Fu, Mooi Choo Chuah. Ecg anomaly detection via time series analysis. http://link.springer.com/chapter/10.1007

Gao, Jing. Anomaly detection. In *Advances in Neural Information Processing Systems*, SUNY, Buffalo.

K., Bhattacharyya D. and K., Kalita J. (eds.). *Network anomaly detection: A machine learning perspective*. CRC Press, 2013.

L. Portnoy, E. Eskin, S. Stolfo. Intrusion detection with unlabelled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001.

labs Yahoo. A benchmark dataset for time series anomaly detection, 2015.

Mangi Zhao, Venkatesh Saligrama. Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in Neural Information Processing Systems*, 2009.

Michail Vlachos, Philip Yu, Vittorio Castelli. On periodicity detection and structural periodic similarity. http://epubs.siam.org/doi/pdf/10.1137/1.9781611972757.40.

W., Stefansky. Rejecting outliers in factorial designs. In *Technometrics*, pp. 14, 1972.