HELLO WORLD

Applying Data Science for Anomaly and Change Point Detection

BY OANA NICULAESCU

What do we mean when we say we are trying to find anomalies in a data set? What are anomalies? How can we find the point at which the data is becoming anomalous just by looking at previous data behavior? Those are the questions we are going to try to answer in this introductory article about anomaly detection. By the end of this, and with the help of a running example using network data, you will be able to devise a few simple algorithms for anomaly detection.

WHAT ARE ANOMALIES?

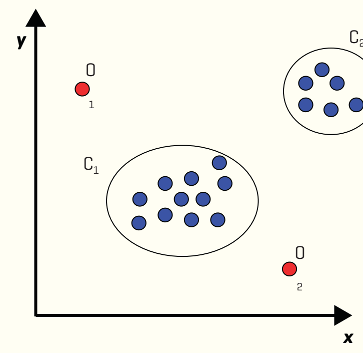
Anomaly detection refers to the problem of finding patterns in data that are not aligned with the expected behavior. We are looking for outliers, exceptions, or discordant observations that look out of place when viewing the entire set of data.

Anomalies are patterns in the data that do not conform to a well-defined notion of normal behavior. In Figure 1 we can observe a series of two-dimensional points that are being categorized in a couple of regions that form two clusters, C1 and C2 . The points that are far away from those regions, like O1 and O2, are considered outliers since they do not align to the normal region.

Anomalies can be broadly categorized as point anomalies when a single instance of data is anomalous if it's too far off from the rest, like O1 and O2 from Figure 1. Point anomalies can be used to detect credit card fraud based on the amount spent. Contextual anomalies, where the abnormality is context specific, are very common in time-series data, for example, the anomalies in network traffic patterns based on the throughput a network has at some point. It might be normal to experience high traffic during the early morning hour and a slow down during lunch hours, but not vice-versa. Collective anomalies are the type of anomalies being detected with the use of a set of data instances. For example, on a Linux machine the following events might appear: http-web, buffer-overflow, http-web, http-web, smtp-mail, ftp, http-web, ssh, smtp-mail, http-web, **ssh**, **buffer-overflow**, **ftp**, http-web, ftp, smtp-mail,http-web, etc. The events in bold are not an anomaly taken individually, but taken collectively with the ftp_transfer event might signal a potential cyber-attack, i.e. someone exploits a buffer overflow vulnerability and copies data from a remote machine to the local host.

The process of anomaly detection has a couple of important challenges. Since we want to discover a pattern that does not conform to normal expected behavior in the data, the first step in the anomaly detection process is to define a normal baseline behavior and then declare any observations that are not aligned with this normal behavior as anomalies. But defining a region that covers all normal behavior is very difficult, especially when the boundary between normal and abnormal behavior is hard to define. On top of that, in many domains what is classified as normal behavior keeps evolving, so the model needs to evolve as well. Speaking of models, usually the labeled data for training/validating the model is not readily available and in most cases contains noise that might be similar to the anomalies we want to detect.

Figure 1. Representation of normal and anomalous region in a data set.



Considering those challenges, the problem of anomaly detection in a general form is usually not easy to solve, that's why there are specific variations of the problem being solved keeping in mind the nature of the data, the availability of labeled and unlabeled data, and the types of anomalies one wants to detect.

AN OVERVIEW OF ANOMALY DETECTION TECHNIQUES

When it comes to classifying anomaly detection techniques there are a few attributes over which they can be classified. Based on the extent of which labels are available in the data we can apply supervised anomaly detection techniques, which assume the availability of a data set that has labeled instances for normal as well as abnormal classes. Semi-supervised anomaly detection techniques operate in a semi-supervised mode, in which the data has labels only for the normal classes. And finally, unsupervised anomaly detection techniques do not require training data and are most widely used.

Another way to classify anomaly detection techniques is based on the underlying algorithm they are using. We have classification-based techniques where classification is used to learn a model from a set of labeled data instances (training set) and then classify a test instance into one of the instances (testing). In this category are neural-network based, Bayes-based, rule-based, and support vector machine-based algorithms. Clustering based anomaly detection techniques are used to group similar data into clusters, with the assumption that normal data belongs to a cluster while anomalous data doesn't belong to any cluster. Statistical anomaly detection techniques are techniques that assume an anomaly is "an observation that is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed" [1]. The underlying principle is normal data points occur in high probability regions of a stochastic model, while anomalies occur in low probability regions of the said model.

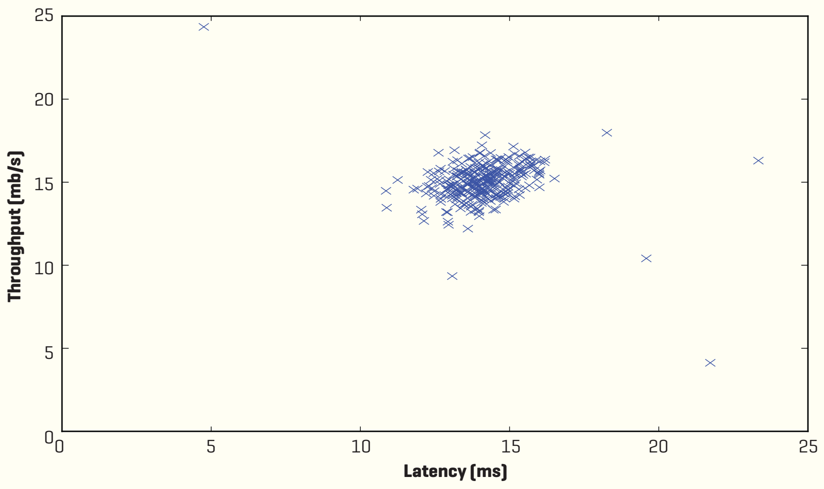
Next we are going to take a detailed look at statistical anomaly detection techniques, specifically the parametric and non-parametric ones, and later we are going to observe the two methods in action by running an example on some network data.

STATISTICAL ANOMALY DETECTION TECHNIQUES

As previously mentioned, when we are using statistical anomaly detection techniques we are looking for normal data instances that occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the said stochastic model. How does a statistical technique work? It fits a statistical model for normal behavior to the given data and then it applies a statistical inference test to determine if an unseen instance belongs to this model or not. If an unseen instance has a low probability of being generated from the learned model, based on the applied statistic, then that instance is an anomaly.

There are two types of statistical techniques: parametric and non-parametric ones. The main difference is parametric techniques assume the knowledge of the underlying distribution and estimate the parameters from the given data

Figure 2. Visualizing the data points and identifying outliers.



Listing 1. Selecting a threshold for determining the anomalies.

```
def selectThreshold(probs,gt):
    best_epsilon = 0
    best_f1 = 0
    f = 0
    stepsize = (max(probs) - min(probs)) / 1000;
    epsilons = np.arange(min(probs),max(probs),stepsize)
    for epsilon in np.nditer(epsilons):
        predictions = (probs < epsilon)
        f = f1_score(gt, predictions,average='binary')
        if f > best_f1:
            best_f1 = f
            best_epsilon = epsilon

    return best_f1, best_epsilon
```

[2], while the non-parametric ones don't assume any knowledge about the distribution of the underlying data. We will compare the two models by running a parametric technique, the Gaussian Model algorithm, and a non-parametric one, the CUSUM algorithm, on a data set and then try to detect the anomalies and points of change in the trends underlying the data. We are also going to identify outliers in computer server network data.

ANOMALY DETECTION USING NETWORK DATA

The data we are using to illustrate this example is quite simple, it only has two features: 1) throughput in mb/s and 2) latency in milliseconds of response for each server. The data will be broken down into two sets, the training set and the test set, each of them with around 300 samples (which is a very small data set).[1] Since we are dealing with a limited number of features here, two to be exact (throughput and latency), plotting helps to visualize the outliers. In most cases for higher dimensional data this is not possible. We can see the plotted data in Figure 2.

The Gaussian model algorithm assumes the data is generated from a Gaussian distribution. The parameters are estimated using maximum likelihood estimate.[2] The distance between a data instance and the estimated mean is the

1 https://github.com/elf11/XRDS/blob/master/tr_server_data.csv
2 https://en.wikipedia.org/wiki/Maximum_likelihood_estimation

Figure 3. Outliers' visualization.

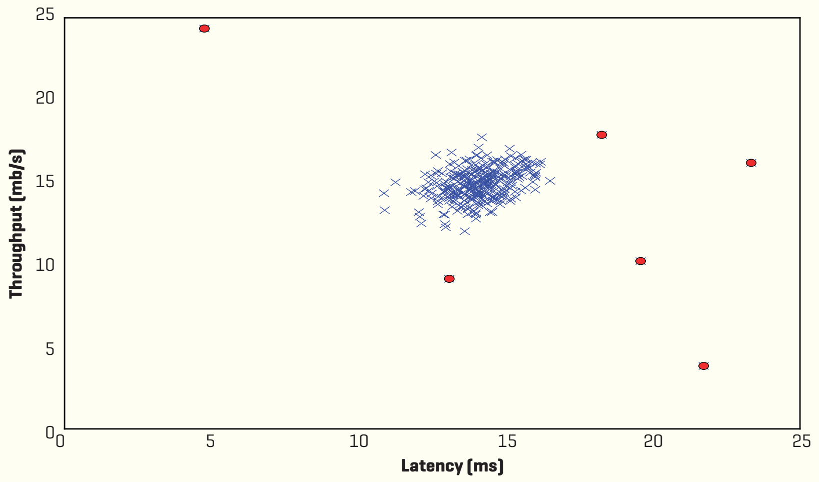
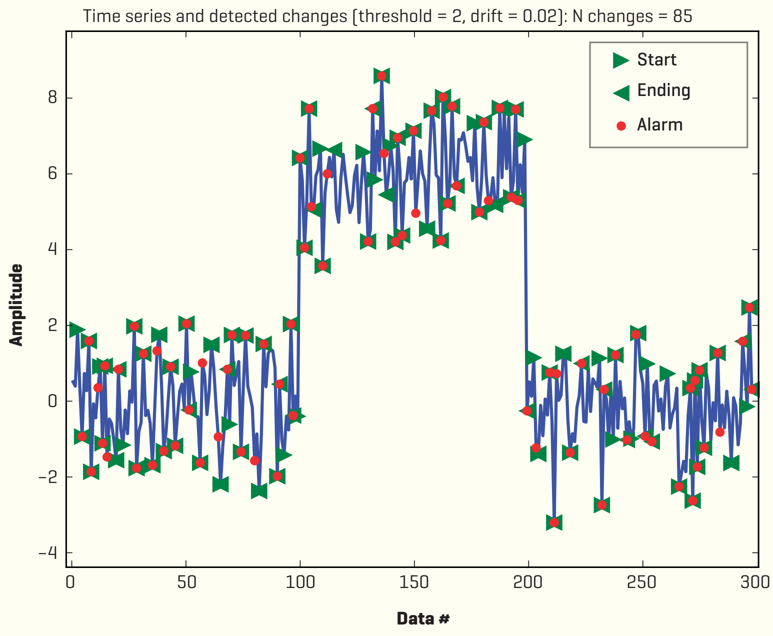


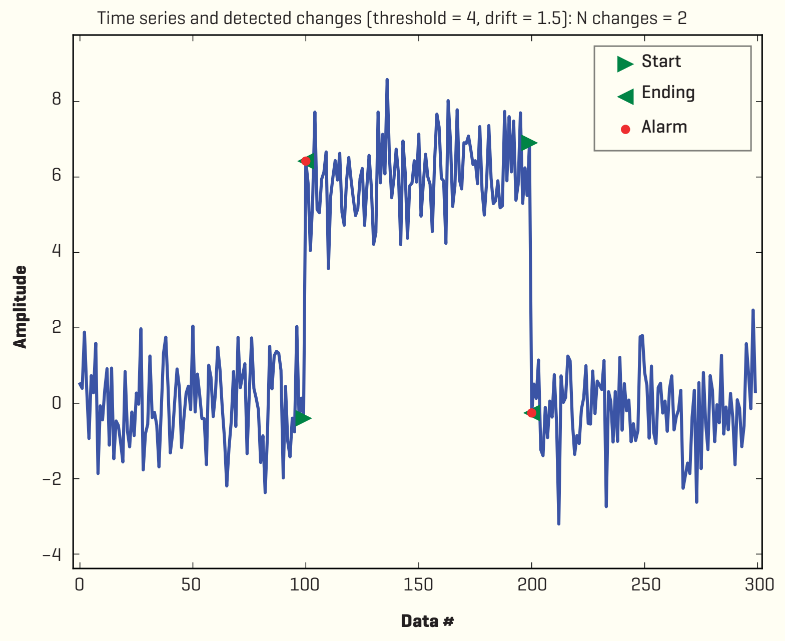
Figure 4. Point change and anomaly detection using a threshold of 2 and drift set to 0.02.



anomaly score for that instance. To be able to determine the anomalies we apply a threshold to the anomaly scores.

This model is used to learn an underlying pattern of the dataset with the hope that our features follow the Gaussian distribution. We find data points with very low probabilities of being normal and thus with a high probability of being outliers. For the training set, we learn the Gaussian distribution of each feature for which mean and variance of features are required. To find the optimal value of the threshold we try different values in a range of learned probabilities on a cross-validation set. The f-score, or the measure of the test's accuracy,

Figure 5. Point change and anomaly detection using a threshold of 4 and drift set to 1.5.



is being calculated for predicted anomalies based on the ground truth data available. In Listing 1 we can see the threshold function that we are going to utilize for selecting the anomalies. The highlighted outliers are visualized in Figure 3; we can see our assumption that the initial data was having an underlying Gaussian distribution was correct. But the method is not very precise, since many of the close points to the main cluster were not detected as outliers, though in some situations they might be.

The CUSUM algorithm is one of the classic techniques used for detecting point changes in data and more recently for detecting anomalies. The implementation for the CUSUM algorithm varies, one form of it involves the calculation of the cumulative sum of positive and negative changes (g_t^+, g_t^-) in the data and comparing to a threshold. If the threshold is exceeded then a change must have occurred and the cumulative sum is restarted from zero. Of course, the data might present a slow drift either upward or downward and that might be detected as a change. To avoid that the algorithm can be tuned with a drift parameter. The proper working of the algorithm depends on those two parameters: threshold and drift. So, properly tuning those is desirable. The literature recommends to start with a very high threshold, choose drift as half of the expected change in the data, then set the threshold in such a way that the number of false alarms (false changes in the data) is obtained.

We are going to try to detect the points of change and anomalies on the same data set from before, we are going to use the latency column as the time series on top of which we are going to run the CUSUM algorithm. And to illustrate the point of tuning the threshold and drift parameters properly, we are going to have two runs of the algorithm. The first run is very bad, almost every point in the series is considered an anomaly and a change point. This can be observed in Figure 4 where we have a threshold of 2 and a very low drift parameter equal to .02 . Next observe Figure 5, we can see that by adjusting the threshold, this time it has a value of 4, and the drift parameter, we are allowing the trend to move either way by 1.5, we are obtaining a much better outcome.

CONCLUSION

There is a huge number of applications for anomaly detection techniques. Each one of those applications comes with its own set of challenges; there is no one-size-fits-all model that would work for all of those problems. Some of the applications and their particularities are intrusion detection systems where the biggest challenge is data size, so we need methods that are computationally efficient for fraud detection and detection of criminal activities occurring in banks, credit card companies, insurance systems, and so on. But this comes with time constraints, the sooner the fraudulent transactions are being detected the better. And as we have seen, there is an application of anomaly detection in computer networks as well, in detecting changes in throughput and quality of service offered by the network.

Each of the techniques discussed in this article have their own strengths and weaknesses, and of course there are many more techniques than those addressed here. The main takeaway is that it's important to know which anomaly detection technique is best suited to a given problem. Given the complexity of the problem space, even though it is not feasible to provide such an understanding for every anomaly detection problem we can at least get a general intuition about them.

References

[1] Anscobme, F. J. Rejection of outliers. *Technometrics* 2, 2 (2012), 123-146; http://amstat.tandfonline.com/doi/pdf/10.1080/00401706.1960.10489888

[2] Eskin, E. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, 2000, 255-262. https://academiccommons.columbia.edu/download/fedora_content/download/ac:125814/content/anomaly-icml00.pdf

DOI: 10.1145/3220677