Anomaly Detection in Time Series

-By PIYUSH GOEL

[piyushgoel997@gmail.com](mailto:piyushgoel997@gmail.com)

## Abstract

A lot of the data in today’s world appears in the form of time series. Human experts can detect anomalies (unexpected scenarios) up to some extent. Some examples of anomalies would be sudden electricity outage, breakdown of some machine, natural anomalies like earthquake, flood, etc. It would be highly beneficial for companies and humans as a whole if these could be predicted in advance, so as to prepare for these or even stop them from happening.

# Introduction

In this project the data received through the alarms fitted on a telecom tower site was analyzed. Then various Statistical and Machine learning techniques were applied to predict the **Site Down** alarm, which basically means that, that particular site is not currently operational. Whatever maybe the reason for that doesn’t matter. It can be due to battery out of charge, diesel generator out of diesel, power outage, etc. It is required to make the prediction in all of the cases. If the prediction can be made even in a few cases well in advance then preventive measures can be taken and it would be highly profitable.

# About the Data

All the alarms have an **Alarm ID** (positive integer)associated with them, and we’ll only deal with the IDs only. The data will be in the form of series of alarms. The given data is for 3 months for more than a 100 sites. Each site will have its own series of alarms. There are more than 100 unique alarms.

The only required Information is that the alarm id 441 represents Site Down. So, the goal of this project is to predict the occurrence of alarm whose id is 441.

Before the Anomaly Detection the following methods were used to analyze the data.

# Methods Used

## Checking previous alarms

Firstly, the series of alarms was taken for all sites and sorted by the Site ID and then from oldest to newest. Then analysis was done considering a single alarm at a time, to check if there is any particular alarm which is the cause of the site going down. For an Alarm, say, **a**.

Let **X** be the event that the site goes down

(i.e. the event of occurrence of 441)

And **Y(a)** be the event that the alarm **a** occurs.

Calculating the following probabilities using Bayes Theorem**:**

1. **P(X/Y(a))(a)** (i.e. X given Y(a))
2. **P(Y(a)/X)(a)** (i.e. Y(a) given X)

For information about conditional probability see [[1]](#_References:) and [[2]](#_References:).

The following results were obtained (Only the Top 4 rows are shown out of 100+ rows):

1. In Descending order of P(Y(a)/X)(a)

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **ALARM ID** | **P(Y(a)/X)(a)** | **P(X/Y(a))(a)** |
| **1** | 148 | 0.431 | 0.094 |
| **2** | 22 | 0.369 | 0.006 |
| **3** | 38 | 0.293 | 0.009 |
| **4** | 133 | 0.225 | 0.093 |

Note: The values of probabilities are approximated to only 3 digits after decimal.

1. In Descending order of P(X/Y(a))(a)

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **ALARM ID** | **P(X/Y(a))(a)** | **P(Y(a)/X)(a)** |
| **1** | 444 | 0.002 | 0.25 |
| **2** | 163 | 0.002 | 0.154 |
| **3** | 441 | 0.089 | 0.126 |
| **4** | 67/361/401 | 0.001 | 0.111 |

Note 1: The values of probabilities are approximated to only 3 digits after decimal.

Note 2: There was a Tie between 67, 361 and 401 so they are shown together.

### Observations

This analysis gave some useful insight about the data. From the First Table:

* It is moderately probable for some alarms (for ex- 148, 22) to have occurred before the site went down.
* But those alarms have very low P(Y(a)/X)(a). So it can’t be said that occurrence of those alarms increases the probability of the site going down.

It can be seen from the Second Table that no alarm has a high value of P(X/Y(a))(a) i.e. No alarm ensures or increases the probability of the site going down. So we might need to try something else, for better results.

## Checking Series of Alarms

In the previous method I concentrated only on one alarm at a time. Next, I’ll tried analyzing series of alarms at once. I did this with the help of the tree data structure. For information about Trees see [[3]](#_References:).

A Tree Node had two fields – **alarm\_id** and **count**. The count for a Node was the sum of the counts of all of its children. The count for the leaves was the number of times the Site went down after following the series of alarms in the path from the root of the tree to that leaf node. The site down alarm was always the leaf node due to the way the series was inserted in the tree, since the reason for site going down needs to be analyzed.

Using this method I had to keep one restriction in mind. One of the parameters in building the tree was the height of the tree, Let it be **H**. So the restriction was that **H** can’t be very large. This is because data insertion in a tree takes place recursively and if the height of the tree is increased beyond a certain limit, it would cause Stack Overflow. This upper limit for **H** would depend on the machine on which the program is running.

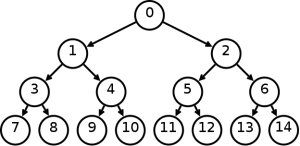
I kept the height constraint on the trees and kept the height of each tree constant instead of using the time of occurrence as the constraint. This can be thought of as a preference or to prevent even the slightest chance of stack overflow.

### Observations:

* The **count** was higher for nodes closer to the root and lower for those towards the leaves, as expected.
* Some Nodes had even more than 20 children, while some had just one.

The count of none of the leaf was high enough to lead to the conclusion that a particular path was occurring more than the other one.

The tree was so large that it can’t be shown here. But here’s an example of what a tree looks like –



🡨 These are the leaf nodes

Source - Google

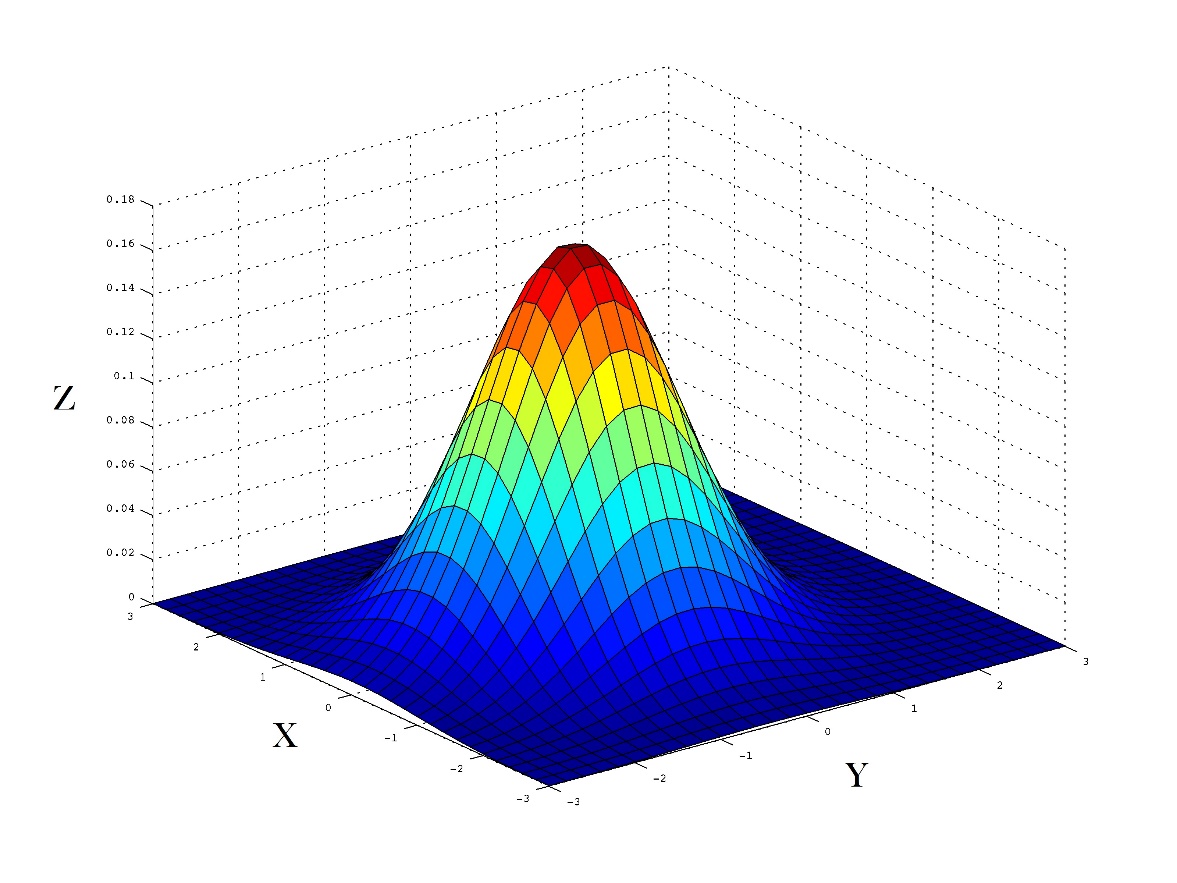
# Anomaly Detection

Anomaly Detection (also Outlier Detection) is the identification of items, events or observations which do not follow the general or expected trend. It can broadly be classified in three categories namely – **Unsupervised anomaly detection, supervised anomaly detection** and **Semi-supervised anomaly detection**. For further information on anomaly detection see [[4]](#_References:).

Here, Semi-supervised anomaly detection was done. The event of the site going down was considered as anomaly (labeled as positive example or belonging to the class **1**) and the rest of the data as non-anomalous (labeled as negative examples or belonging to the class **0**). The data was divided into three sets ­– **Training Set, Cross Validation Set and Testing Set** with their respective size in the ratio 0.6 : 0.2 : 0.2 approximately.

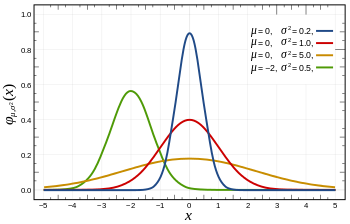
The training set consisted of unlabeled negative examples only. The cross validation set and the testing set consisted of labeled examples from both the classes but the number of examples from class 0 far less than the number of examples from class 1.

The training model used was **Multivariate Gaussian Distribution.** It is the same as Gaussian distribution (also Normal Distribution) but with multiple variables or dimensions. It forms a bell shaped curve for a single variable or a hill kind of structure with elliptical cross sections for two variables. An example for the distribution with two dimensions as plotted in MATLAB is shown on the next page.



Here the X and Y axes represent the two dimensions with respect to which the distribution is done and the Z axis represents the probability.

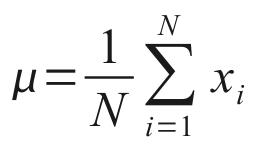
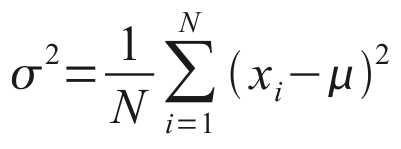
The plot shown below shows the normal distribution with respect to the different value of mean (μ) and variance (σ2).



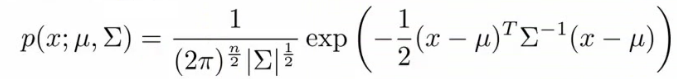
X ~ N (μ, σ2)

A Normal distribution is represented by –

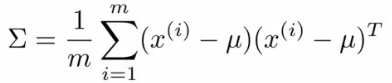
It can be interpreted as – x is distributed normally on μ and σ2 where (the symbols have their usual meanings)

For Multivariate Gaussian Distribution, with the set of examples as {x1, x2, ..., xm}. The probability for an example *x* as a function of μ and Σ (covariance matrix) is



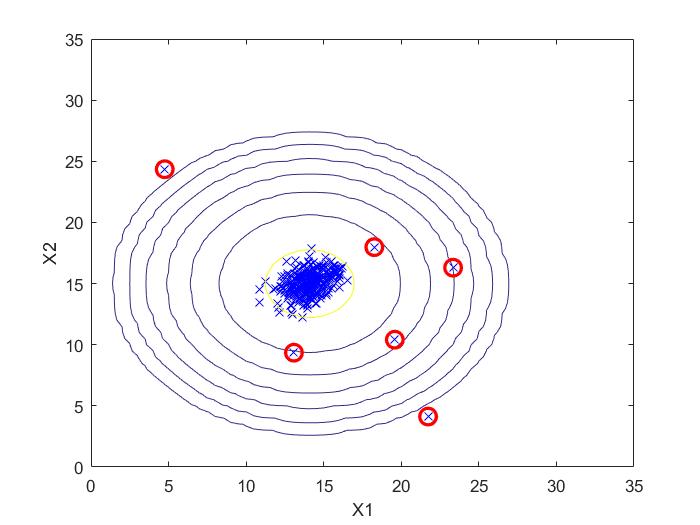
Where the symbols have their usual meanings and the covariance matrix is



For more information on Multivariate Gaussian Distribution see [[5]](#_References:).

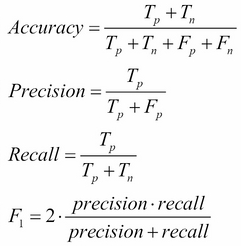
For anomaly detection using Multivariate Gaussian Distribution a separating value of probability ε is chosen. The parameters of the distribution are calculated using the training data set and the value of ε is calculated using the cross validation data set. All the examples having the probability lower than ε are classified as anomalies and the rest are classified as non-anomalous.

In the following example for two dimensions X1 and X2 as plotted in MATLAB. The x’s represent a data points plotted. The x’s in red circles represent those examples which were classified as anomalies. The blue circles show the curves for some random values of ε.



The appropriate correctness measure in the case of skewed classes as in anomaly detection is **F1 Score** rather than accuracy. It is calculated using the following table and formulae –

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Actual Class** | | |
| **Predicted Class** |  | 1 | 0 |
| 1 | True Positive (**Tp**) | False Positive (**Fp**) |
| 0 | False Negative (**Fn**) | True negative (**Tn**) |



The only parameter for Anomaly Detection was the number of hours, **H** for which the data is to be considered to form one example.

### Observations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| H | Cross Validation Set | | Test Set | |
| F1 | Accuracy | F1 | Accuracy |
| 1 | 0.426 | 0.472 | 0.440 | 0.532 |
| 6 | 0.421 | 0.472 | 0.434 | 0.500 |
| 12 | 0.470 | 0.568 | 0.469 | 0.548 |

The values are approximated to only 3 digits after decimal.

One important observation was that increasing the number of training examples reduced the F1 Score and accuracy.

# Conclusion and Future Work:

The results were better than random human guess for some cases. Two conclusions can be drawn -The data had too much of noise or the given information is not enough to make the prediction. And/or some other algorithm should be used.

In the first case, which is most likely the case, the data needs to be filtered of the noise or some more information should be used. In the second case, given enough resources and data the following might be tried to improve the results –

* The Anomaly Detection can be tested with more values of H.
* Different implementations of Neural Networks (such as RNN) may be used to classify the data given enough resources.

# References:

[1] <https://en.wikipedia.org/wiki/Conditional_probability>

[2] <https://en.wikipedia.org/wiki/Bayes%27_theorem>

[3] <https://en.wikipedia.org/wiki/Tree_(data_structure)>

[4] <https://en.wikipedia.org/wiki/Anomaly_detection>

[5] <https://en.wikipedia.org/wiki/Multivariate_normal_distribution>

# Languages and Packages used

* Java
* Python, Libraries – Numpy, Pandas, MatplotLib
* MATLAB

## Mentored By:

**Tulika Pradhan, Infozech**