

**Network intrusion detection
using gaussian anomaly.**



Contents

1.	Introduction to anomaly detection	02
2.	Anomaly Detection Algorithm	03
	<ul style="list-style-type: none">• Gaussian Distribution• How the algorithm works• Algorithm• Multivariate gaussian distribution	
3.	Implementation	07
	<ul style="list-style-type: none">• Extracting features	



	<ul style="list-style-type: none">• Data Splitting• Training• Selecting the threshold• Algorithm evaluation	
4.	Results	11
	<ul style="list-style-type: none">• Accuracy parameters on cross validation dataset• Accuracy parameters on test dataset	
5.	Conclusion and future work	12
6.	References	12

Introduction to anomaly detection

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behavior, called outliers. It has many applications in business, from intrusion detection (identifying strange patterns in network traffic that could signal a hack) to system health monitoring (spotting a malignant tumor in an MRI scan), and from fraud detection in credit card transactions to fault detection in operating environments.

Before getting started, it is important to establish some boundaries on the definition of an anomaly. Anomalies can be broadly categorized as:

1. **Point anomalies:** A single instance of data is anomalous if it's too far off from the rest.

Business use case: Detecting credit card fraud based on "amount spent."

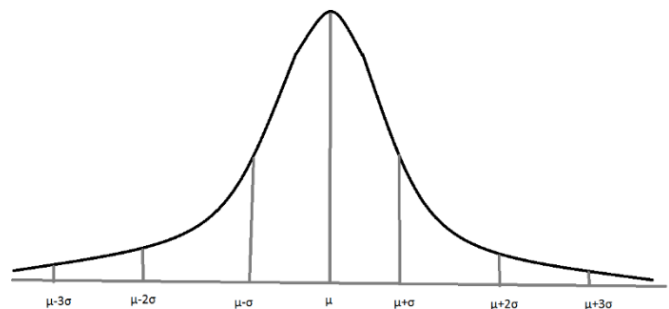
2. **Contextual anomalies:** The abnormality is context specific. This type of anomaly is common in time-series data. *Business use case:* Spending \$100 on food every day during the holiday season is normal, but may be odd otherwise.
3. **Collective anomalies:** A set of data instances collectively helps in detecting anomalies. *Business use case:* Someone is trying to copy data from a remote machine to a local host unexpectedly, an anomaly that would be flagged as a potential cyber attack.

Anomaly Detection Algorithm

Gaussian Distribution

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behaviour, called **outliers**. It has many applications in business, from intrusion detection (identifying strange patterns in network traffic that could signal a hack) to system health monitoring (spotting a malignant tumour in an MRI scan), and from fraud detection in credit card transactions to fault detection in operating environments.

Here we've chose the Gaussian approach for Anomaly Detection. In this approach, all the features are modeled on a Gaussian Distribution and given a new data-point, the probability of the data-point is given by Gaussian/Normal Distribution Function. The Gaussian Distribution is a bell-shaped curve



Gaussian/Normal Distribution with Mean = μ and Standard Deviation = σ

(as shown in the figure) that can be described by a function $N(\mu, \sigma^2)$.

Let $x \in \mathbb{R}$. If the probability distribution of x is Gaussian with **mean μ , variance σ^2** , then:

$x \sim N(\mu, \sigma^2)$

The little \sim or 'tilde' can be read as "distributed as." The Gaussian Distribution is parameterized by a mean and a variance. **μ , or μ** , describes the center of the curve, called the **mean**. The width of the curve is described by **sigma, or σ** , called the **standard deviation**. The full function is as follows:

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

We can estimate the parameter μ from a given dataset by simply taking the average of all the m examples

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

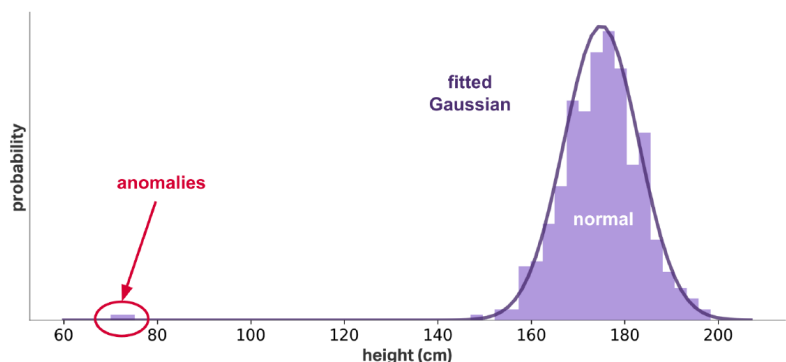
We can estimate the other parameter σ^2 , with our familiar squared error formula:

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2$$

How the algorithm works

Let's say we have a dataset where each observation is a person. Each person is described by only one variable : their height in centimeters. In the dataset we have some anomalies, such as two dogs inserted by mistake. Gaussian fitting starts with a strong assumption on the distribution of your data, that it follows the normal, or Gaussian, distribution. A Gaussian distribution appears as a bell curve and is completely described by two parameters, its center (or mean) and its scale (standard deviation or variance). Fitting a Gaussian is just finding the best values for these two parameters to better fit the distribution of our samples. If our dataset is big enough and representative, we will probably end up with something like 175 cm as mean and 8 cm as the standard deviation.

Once this is done, we can tell the probability of appearance for any height value (including ones not present in our training data). By setting a threshold on this probability, like 0.1%, we can consider everything below to be an anomaly. This will likely detect our



two dogs, who are well outside the core of the distribution.

Algorithm

Given a training set of examples, $\{x(1), \dots, x(m)\}$ where each example is a vector, $x \in \mathbb{R}^n$. We are then given a new example, x_{test} and we want to know whether this new example is abnormal or anomalous. We define a model $p(x)$ that tells us the probability that the example is not anomalous. We also use a threshold ϵ (epsilon) as the dividing line so we can say which examples are anomalous and which are not.

Here n is the number of features.

$$p(x) = p(x_1; \mu_1, (\sigma^2)_1) p(x_2; \mu_2, (\sigma^2)_2) \cdots p(x_n; \mu_n, (\sigma^2)_n)$$

In statistics, this is called an "independence assumption" on the values of the features inside training example x . More compactly, the above expression can be written as follows:

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

Choose features x_i that you think might be indicative of anomalous examples.

Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$.

Calculate

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

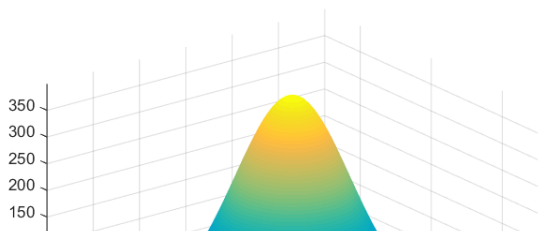
Given a new example x , compute $p(x)$:

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sigma_j \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x_j - \mu_j}{\sigma_j} \right)^2}$$

Anomaly if $p(x) < \epsilon$.

Multivariate Gaussian Distribution

In certain cases the normal Gaussian distribution is not enough to accurately flag anomalies. A



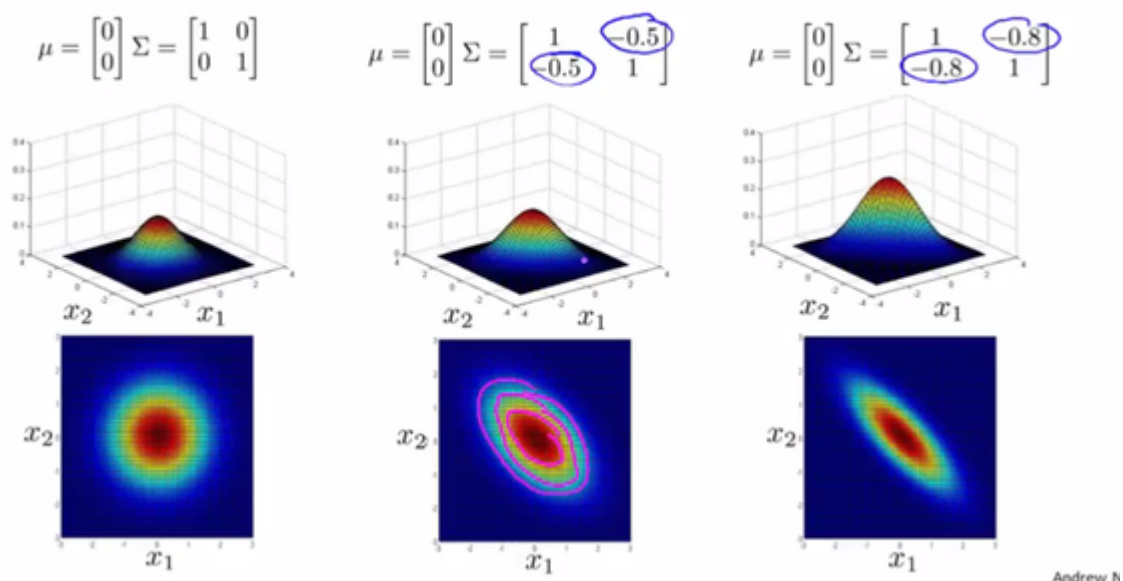
multivariate Gaussian distribution calculates the probability model of x at once, instead of modelling the probabilities for each feature alone. It uses a covariance matrix instead of Sigma squared. The multivariate gaussian distribution is an extension of anomaly detection and may (or may not) catch more anomalies.

Instead of modeling $p(x_1), p(x_2), \dots$ separately, we will model $p(x)$ all in one go. Our parameters will be: $\mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$

The formula looks like:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

The important effect is that we can model oblong gaussian contours, allowing us to better fit data that might not fit into the normal circular contours. Varying Σ changes the shape, width, and orientation of the contours. Changing μ will move the center of the distribution.



When doing anomaly detection with multivariate gaussian distribution, we compute μ and Σ normally. We then compute $p(x)$ using the new formula in the previous section and flag an anomaly if $p(x) < \epsilon$. The original model for $p(x)$ corresponds to a multivariate Gaussian where the contours of $p(x; \mu, \Sigma)$ are axis-aligned. The multivariate Gaussian model can automatically capture correlations between different features of x . However, the original model maintains some advantages: it is computationally cheaper (no matrix to invert, which is costly for large number of

features) and it performs well even with small training set size (in multivariate Gaussian model, it should be greater than the number of features for Σ to be invertible).

Implementation

We've implemented our above mentioned algorithm in Octave/MATLAB (All the files with .m file extension).


Extracting Features

Once the data is collected, it's time to assess the condition of it, including looking for trends, outliers, exceptions, incorrect, inconsistent, missing, or skewed information. The first step towards implementation of our model is loading our data into a suitable place and prepare it for use in our machine learning training. We've used basic MS Excel tools to prepare our dataset. To draw features from the given dataset it's fairly important that we derive meaning from each of the entries.

Raw dataset (Random 5 entries)

SourceIP	Source Port	Destination IP	Destination Port	Pkt Count	Pkt Size	Malicious or Not
202.54.250.39	36728	23.20.239.12	25	4	1920	0
202.54.250.39	51100	18.213.250.117	25	4	1920	0
202.54.250.39	51110	18.213.250.117	25	4	1920	0
202.54.250.39	52932	52.4.209.250	25	4	1920	0

After getting features (Random 5 entries)



Destination Port	Pkt Count	Pkt Size	Email Reputation	Web Reputation	IP Reputation	Pkt Size	Country Score	Malicious or Not
25	4	1920	0.5	0.5	0.5	0.1	1	0
25	4	1920	0.5	0.5	0.5	0.1	1	0
25	4	1920	0.5	0.5	0.5	0.1	1	0
443	1	512	1	0.1	0.1	0.1	1	1
443	1	480	1	0.1	0.1	0.1	1	1

The Columns in red are the ones which will be used for further training the model. We obtained these 5 features from the raw data.

Features chosen:

1. **Email reputation** : Email (or sender) reputation is the measurement of your email sending practices and how closely you follow the standards established by Internet service providers (ISPs).
2. **Web reputation** : By tracking a broad set of attributes for email and web, the Talos Reputation Center supports very accurate conclusions about a given host. Sophisticated security modeling leverages the breadth of this data to generate a granular reputation score ranging from -10 (for the worst) to +10 (for the very best). On this page the granular reputation score is grouped into **Good, Neutral and Poor** for simplicity reasons. We've obtained email rep for all the IPs using a data scraper.

Url used: https://www.talosintelligence.com/reputation_center

Scoring:

Good or 1 = Little or no threat activity has been observed from this IP address or domain. Email or Web traffic is not likely to be filtered or blocked.

Neutral or 0.5 = This IP address or domain is within acceptable parameters. However, email or Web traffic may still be filtered or blocked.

Bad or 0.1 = A problematic level of threat activity has been observed from this IP address or domain. Email or Web traffic is likely to be filtered or blocked.

3. **IP reputation** :

If Source Port or Destination Port is 25, IP Rep = Email Rep.

If Source Port or Destination Port is either 443 or 80, IP Rep = Web Rep.

Else, IP Rep = Average of Email Rep and Web Rep).

4. **Pkt Size**: If Pkt Count ≥ 20 , Pkt Size = 0.1
If Pkt Count < 20 , Pkt Size = 1.
5. **Country Score**: We can give scores to a country's reputation through the IP address location tracking sites. Ex: For a country like the United States, a score of 0.1 means that an IP address with that location can be a potential threat to our system. Lower the score, the higher the threat.

Url used: <https://www.iplocation.net/>

(Note: Data fields with Source port and Destination Port other than 80,25,443 are already malicious and removed from the training dataset.)

Data Splitting

In machine learning we have certain conventional rules regarding the preparation of the Training Set, Validation Set and Test Set:

1. The **Training Set** should have 60% (approx.) of the total number of instances present in the dataset. All the instances in the Training Set should be non-anomalous (as per the labels).
2. The **Validation Set** should have 20% (approx.) of the total number of instances containing both anomalous and non-anomalous examples (as per the labels).
3. The **Test Set**, containing the remaining instances should also have both anomalous and non-anomalous examples (as per labels).

```

10
11 raw=csvread('rawdata.csv');
12 X=raw(1:600,11:13);
13 y=raw(1:600,15);
14 Xval=raw(601:800,11:13);
15 yval=raw(601:800,15);
16 Xtest=raw(801:1000,11:13);
17 ytest=raw(801:1000,15);
18

```

Training

We first estimate the parameters of our assumed Gaussian distribution using the input data (X here). The function 'estimateGaussian.m' takes as input the data matrix X and should output an n-dimension vector mu that holds the mean of all the n features and another n-dimension vector sigma2 that holds the variances of all the features. Using the output of this function as an input to our next function 'multivariateGaussian.m', which calculates the probability density function of the examples X under the multivariate gaussian distribution with parameters mu and Sigma2.

Selecting the threshold, ϵ

After we have estimated the Gaussian parameters, we can investigate which examples have a very high probability given this distribution and which examples have a very low probability. The low probability examples are more likely to be anomalies in our dataset. One way to determine which examples are anomalies is to select a threshold based on a cross validation set. For this, we will use a cross validation set $\{(x(1)_{cv}, y(1)_{cv}), \dots, (x(mcv)_{cv}, y(mcv)_{cv})\}$, where the label $y = 1$ corresponds to an anomalous example, and $y = 0$ corresponds to a normal example. For each cross validation example, we will compute $p(x(i)_{cv})$. The vector of all of these probabilities $p(x(1)_{cv}), \dots, p(x(mcv)_{cv})$ is passed to a function named 'selectThreshold.m' in the vector pval. The corresponding labels $y(1)_{cv}, \dots, y(mcv)_{cv}$ is passed to the same function in the vector yval. The

function `selectThreshold.m` will return two values; the first is the selected threshold ϵ . If an example x has a low probability $p(x) < \epsilon$, then it is considered to be an anomaly. The function should also return the F1 score, `prec` and `rec` which tells you how well you're doing on finding the ground truth anomalies given a certain threshold. For many different values of ϵ , we computed the resulting F1 score by computing how many examples the current threshold classifies correctly and incorrectly.

Algorithm Evaluation

We use F1 score to find the best possible value of epsilon. This score also helps us in the performance analysis of our model.

The F1 score is computed using precision (`prec`) and recall (`rec`):

$$F1 = (2 \times (\text{prec}) \times (\text{rec})) / (\text{prec} + \text{rec})$$

We can compute precision and recall by:

$$\text{prec} = \text{tp} / (\text{tp} + \text{fp})$$

$$\text{rec} = \text{tp} / (\text{tp} + \text{fn})$$

where

tp is the number of true positives:

the ground truth label says it's an anomaly and our algorithm correctly classified it as an anomaly.

fp is the number of false positives: the ground truth label says it's not an anomaly, but our algorithm incorrectly classified it as an anomaly.

fn is the number of false negatives: the ground truth label says it's an anomaly, but our algorithm incorrectly classified it as not being anomalous.

Once training is complete, it's time to see if the model is any good, using **Evaluation**. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

		Observation	
		Present	Absent
Prediction	Present	True positive	False positive
	Absent	False negative	True negative

Results

Best epsilon found : 6.215632e-03

Best F1 score on CV Set: 0.829268

Anomalies found: 24

Accuracy parameters on cross validation dataset :

true_positive = 17

false_negative = 0

false_positive = 7

prec = **0.70833**

rec = **1**

F1_score = **0.82927**

Accuracy parameters on test dataset :

true_positive = 14

false_negative = 0

false_positive = 8

prec = **0.63636**

rec = **1**

F1_score = **0.77778**

Conclusion and future work

This project marks the beginning of intrusion detection for the given kind of data and it aims to roughly create a working and highly accurate model for the same. It won't be right to say that this project can detect intrusions of any kind as this project operates somewhere only around the layer 3 of the networking structure. The dataset used in this project is also not sufficient enough to train a machine learning model of this kind, hence improvements can be done in data acquisition. This is a very good start to intrusion detection and it needs further developments. Making a web/mobile application out of this project is what we aim to do next where the user needs to enter data manually or in the form of a datasheet and gets to check his data for any anomalies.

Github link for review

<https://github.com/rohanchoudhary/Network-intrusion-detection>

References

Marcio Andrey Teixeira, Tara Salman, Maede Zolanvari, Raj Jain, Nader Meskin and Mohammed Samaka : SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach

Machine learning by Stanford University , Andrew Ng : coursera.org

<https://www.iplocation.net/>

https://www.talosintelligence.com/reputation_center



<https://www.stackoverflow.com>

<https://www.cs.utexas.edu/users/mooney/cs391L/paper-template.html>

<https://www.towardsdatascience.com>

<http://cs229.stanford.edu/>

<https://www.datascience.com/blog/python-anomaly-detection>

Machine learning by Stanford University , Andrew Ng : coursera.org