**Anomaly Detection**

Anomaly Detection algorithm using Gaussian distribution is a common method to identify outliers in a dataset. Compared to supervised learning they are particularly good to identify anomalies in a dataset where there are very few data which are anomalous. This implies that there should be many normal examples in the dataset. It can be used in cases where existing anomalies are different and the future anomolous values which may occur cannot be modelled from the existing values. Supervised learning is much suited to these cases where the number of positive and negative examples are big and so the algorithm can learn what future anomalies will look like. The future anomalies will mostly be like the already existing values. So, for fraud detection, Anomaly detection algorithms are better.

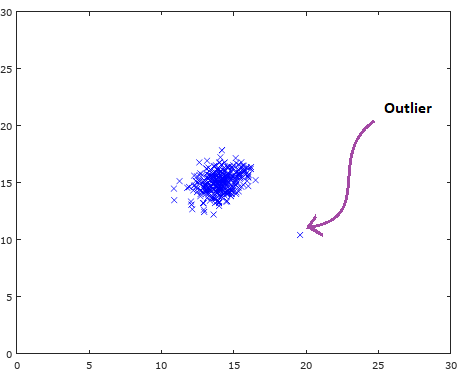


Fig. 1 Outlier in Dataset.

Fig 1 shows a dataset of features plotted in x-y axis with each feature in one axis. As we can see from the picture, an outlier in marked. To explain the algorithm, we can take the example of compressor and we are plotting two of its features in the two axis. If we introduce a new compressor, we will compare it with the features of the existing compressors. If the new data is far from the old data, we can say it is an outlier.

For the plot like the one shown in Fig 1. We will separate the data so that they come in different contours. The data points which are in the same contour are having the same probability. The points which are further away have less probability of occurrence. This is represented by Fig. 2.

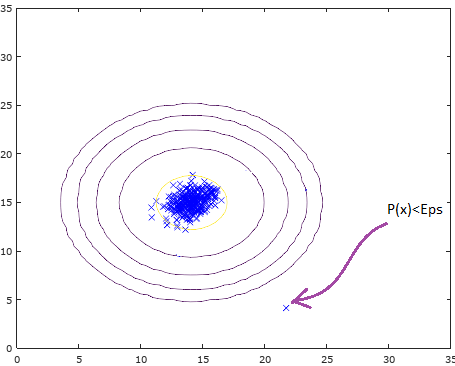


Fig. 2 Estimation of points with same Probability

For a new given point, we will check whether the value of the probability is less than a threshold value Epsilon to mark anomaly.

According to “Central Limit Theorem”, “when [independent random variables](https://en.wikipedia.org/wiki/Statistical_independence) are added, their properly normalized sum tends toward a [normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) (informally a "bell curve") even if the original variables themselves are not normally distributed” [1].

If X ϵ ℝn and probability distribution of X is Normally distributed, then

 P(x)=1/(sigmasqrt(2pi))e^(-(x-mu)^2/(2sigma^2))  where μ, σ2 are mean and variance respectively.

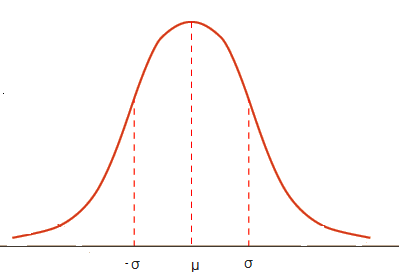


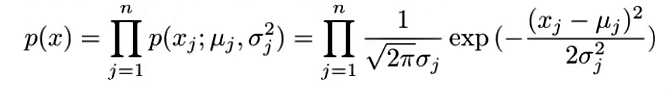
Fig. 3 Bell shaped cure of Normal Distribution

Fig 3 shows the histogram of a normal distribution with mean μ and standard deviation σ. To Estimate anomolous data we need to first estimate the features which can easily show us the anomalous examples. This means that if an intrusion occurs, these data should take high values or low values compared to the usual. If n features are selected in this way and considering data set x1, x2, ..... xm  which are unlabelled training set of m examples and where x ϵ ℝ. We then calculate their mean μj and variance σj for each feature as

μj = ( Σmi=1 xij ) / m

σj. = ( Σmi=1 ( xij - μj)2 / m

Then we will model p(x). This is the value we must compare with the threshold value to check whether the new data is okay. The formula for calculation of p(x) is,



The above equation is known as Probability density estimation. The P(X) calculated in this way helps to define the region where (X) has more probability. The thing to notice here is that the above equation assumes independence relationship among values of the features x1 till xn. However, it is observed through experiments that even though the features are not independent, the algorithm works fine.

Consider a data set with m examples. We will fit a model p(x) on the data. For the p(x) calculated if p(x)<ϵ we mark y=1 (anomalous). This algorithm best work with less number of data. However for the project we used Multivariate Gaussian Distribution which can be considered to be the next level of Density estimation. The advantage of Multivariate Gaussian distribution is no need to compute individual probabilities.

**Multivariate Gaussian Distribution** –

If number of samples are high and number of features are less, the relationship between features maynot be too apparent. We may need to modify features to get better result. Multivariate Gaussian distribution is advantageous in this scenario. If m is number of samples and n is number of features, it is better if m>10n. In multivariate Gaussian Distribution, we will calculate P(x, μ, Σ)



Where,

μ is Mean and Σ is the Co-Variance matrix.

The Anomaly Detection Algorithm explained above can be considered as a special case Multivariate Gaussian distribution where data points are axis aligned. This means that Σ is having only diagonal elements with all off-diagonal elements being zero.

For finding Threshold, we used cross validation dataset. In which few anomalies are marked with y=1 and all others are marked with y=0. We then calculate p(x(i)cv ) and compute ϵ and F1 score to be used for this dataset and check which ϵ will give best F1 score.

F1 =2\*prec\*rec/(prec + rec)

prec =tp/(tp + fp)

rec =tp/(tp + fn)

tp is the number of true positives. Algorithm classifies the data correctly when data is originally an anomaly

fp is the number of false positives, Algorithm classifies the data as anomaly when data is not an anomaly

fn is the number of false negatives, Algorithm classifies the data as non-anomaly when data is an anomaly

Then on test set we will implement the algorithm explained above and identify the anomalies.

**Code –**

**Loading data**

X=load('X.csv'); %dataset

Xval=load('xval.csv'); %cross validation dataset

yval=load('yval.csv');

**Estimating parameters**

[a, b] = size(X);

mu = zeros(b, 1);

sigma2 = zeros(b, 1);

mu=sum(X,1)/a;

for i=1:a

Xsigma(i,:)=(X(i,:)-mu).^2;

end

sigma2=sum(Xsigma,1)/a;

pval = multivariateGaussian(Xval, mu, sigma2);

**Multivariate Gaussian Distribution**

k = length(mu);

if (size(Sigma2, 2) == 1) || (size(Sigma2, 1) == 1)

Sigma2 = diag(Sigma2);

end

X = bsxfun(@minus, X, mu(:)');

p = (2 \* pi) ^ (- k / 2) \* det(Sigma2) ^ (-0.5) \* ...

exp(-0.5 \* sum(bsxfun(@times, X \* pinv(Sigma2), X), 2));

**Estimating Threshold**

increment = (max(pval) - min(pval)) / 1000;

for eps = min(pval):increment:max(pval)

truep= sum((pval<eps )&(yval == 1));

falsep= sum((pval<eps )&(yval == 0));

falsen= sum((pval>=eps )&(yval == 1));

precision= truep/(truep+falsep);

recall = truep/(truep+falsen);

F1 = 2\*prec\*rec/(prec+rec);

if F1 > finalF1

finalF1 = F1;

finalEps = eps;

**Finding Outliers**

outliers = find(p < finaleps);

**Plotting Output**

plot(X(:, 1), X(:, 2), 'bx');

ylabel('Real Power');

xlabel('Time of Day');

plot(X(outliers, 1), X(outliers, 2), 'ro', 'LineWidth', 2, 'MarkerSize', 10);

**Reference**

[1] https://en.wikipedia.org/wiki/Central\_limit\_theorem