ieeta instituto de engenharia electrónica e telemática de aveiro

universidade de aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Machine Learning

LECTURE : Anomaly detection

Petia Georgieva
(petia@ua.pt)

Popular applications of anomaly detection

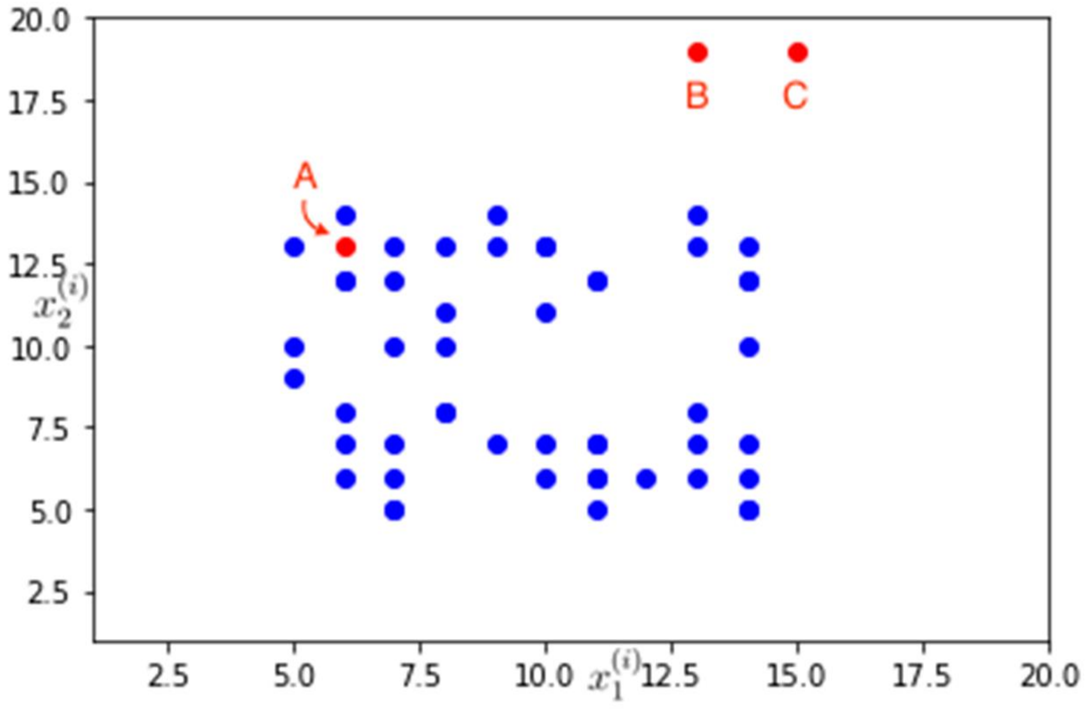
Fraud detection - $x^{(i)}$ – vector of features of user *i*
x_1 - how often does the user login
x_2 # of web pages visited / # of transactions
x_3 - the typing speed of the user
How to identify suspicious users ?

Manifacturing (e.g. aircaft engine features)
x_1= heat generated
x_2=vibration intensity,
x_3, $x_{4..}$ … other features
How to identify anomalous production (engines) for quality control ?

Monitoring computers in data center: $x^{(i)}$ = features of machine *i*
x1=memory use of computer
x2=number of disc accesses /sec
x3=CPU load
x4=network traffic &….other features…
How to identify if a computer is doing something strange and further inspect it?

Anomaly detection

We have a dataset of examples (blue points): $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
Each example has two features x_1 and x_2.
We get new examples (red points): A, B, C
How to decide that A is not an outlier, B and C are anomalous (outliers) ?



Anomaly detection – How ?

From given regular (not anomalous) data get a model of what is considered as normal. For example – a probability model *p(x)* .

Identify anomalous case by checking if

$$p(x_{test}) < \epsilon, \text{ flag as outlier or anomaly}$$
$$p(x_{text}) \geq \epsilon, \text{ flag as normal or non-anomalous}$$

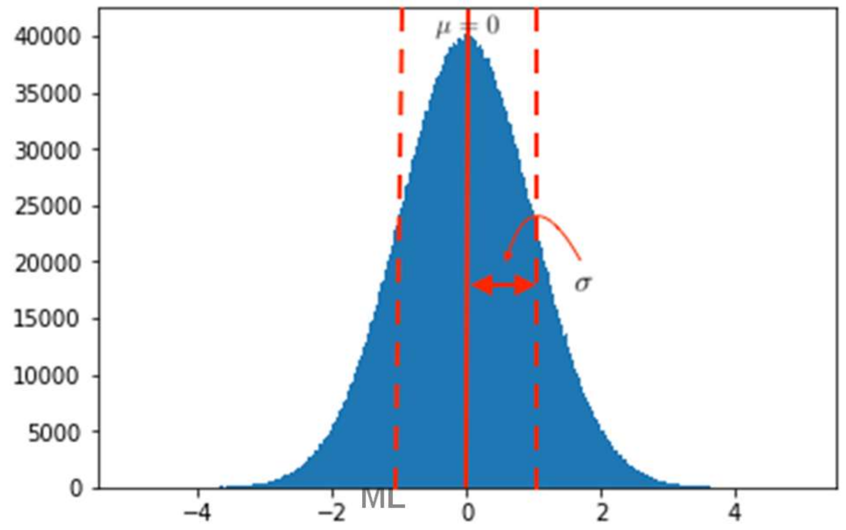
Gaussian (Normal) distribution

If $x \in \mathbb{R}$, and x follows Gaussian distribution with mean, μ and variance σ^2, denoted as,

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

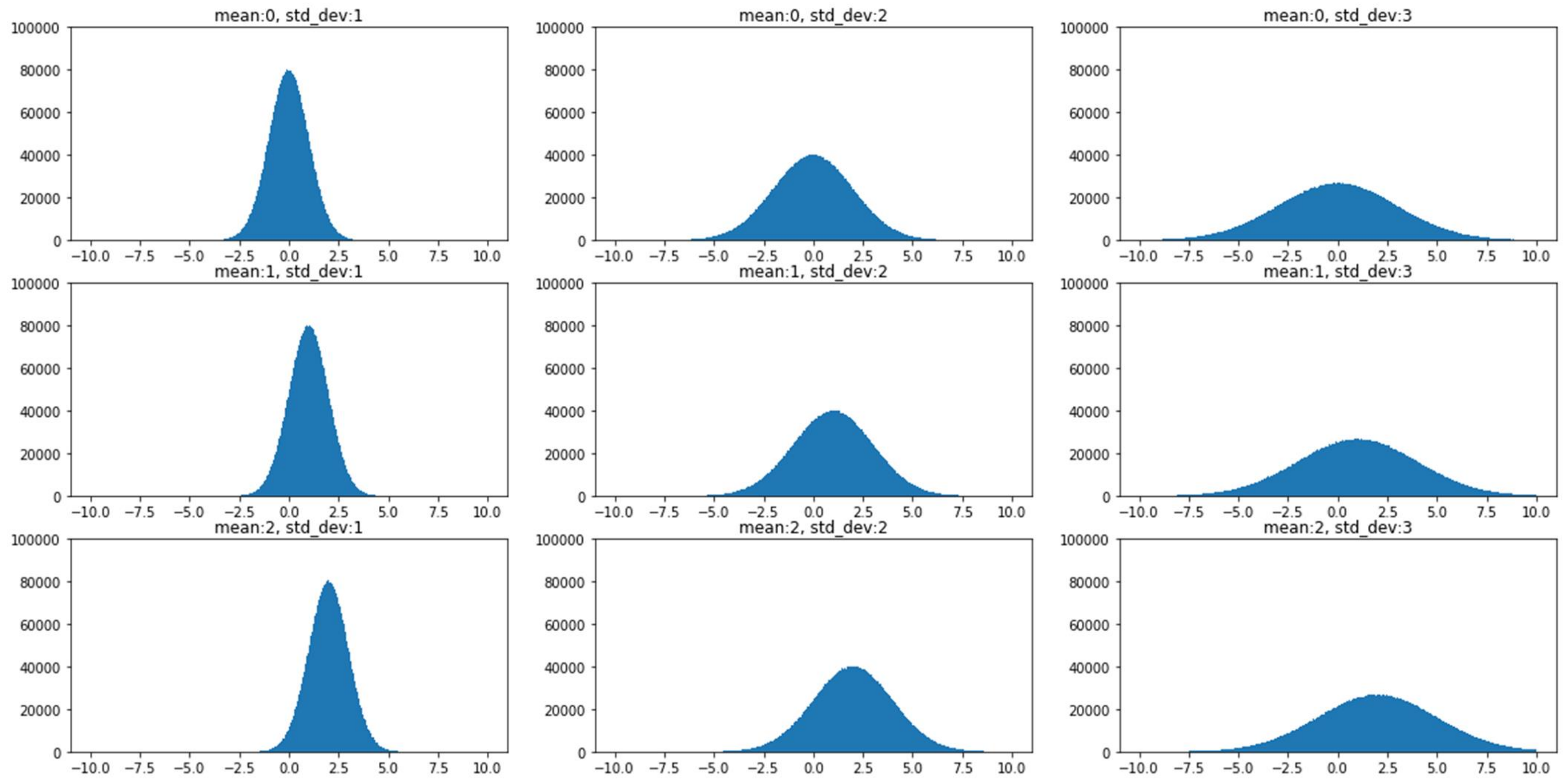
Standard normal Gaussian distribution (μ=0, standard deviation σ=1).
Density is higher around μ and reduces as distance from mean increases.
If we know parameters μ and σ, the probability of x in Gaussian distribution is:

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



Effect of μ and σ on Gaussian curve

Mean (μ) defines the centering of the distribution.
Standard deviation (σ) defines the spread of the Gaussian distribution.
As the spread increases the height of the plot decreases, because the total area under a probability distribution should be = 1.



Parameter estimation

Parameters (µ and σ) of the Gaussian distribution are estimated based on given data $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

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

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

Remark: In statistics instead of 1/m is used 1/(m−1) , but in ML the above formulas are used. This change brings slightly different math properties but for high values of m both give similar result.

Density Estimation Algorithm

Given m training examples => $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

Each example *i* has *n* features => $\{x_2^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)}\}$

Major assumptions:
The features have Gaussian distribution and are independent.

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$
$$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$
$$\vdots$$
$$x_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$$
$$\vdots$$
$$x_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$$

Compute μ and σ of each feature.

Compute *p(x_j)* of each feature.

Compute probability (density estimation) of all features *p(x):*

$$p(x) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

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

Anomaly Detection Algorithm

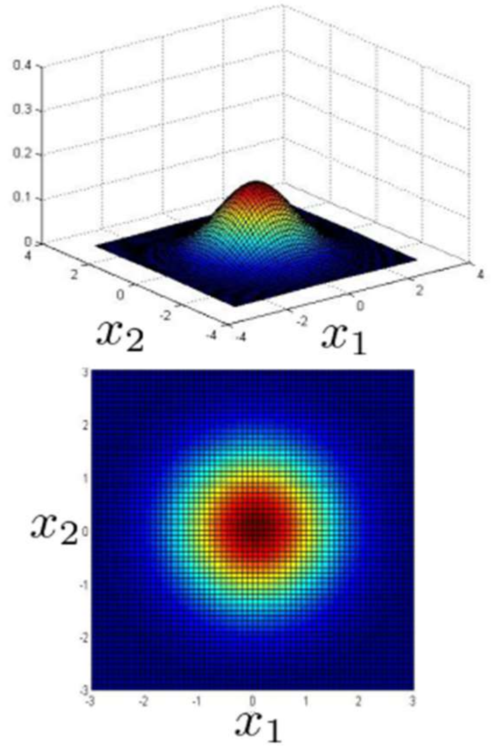
1. Choose features that you think might be indicative of anomalous examples.

2. Fit Gaussian distribution parameters (μ and σ) for each feature.

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

3. Given new example (x_{new}), compute $p(x_{new})$

4. Anomaly if $p(x_{new}) < \epsilon$ (some threshold)



Evaluation of Anomaly Detection System

When developing a learning algorithm making decisions is easier if we have a **single real-valued metric.**

Assume we have some <u>*labelled data*</u>, of anomalous and non-anomalous examples (y=0 if normal, y=1 if anomalous) $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

Training set (assume only normal examples/not anomalous): $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

Cross validation (CV) set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Ex. Manifacturing (e.g. aircaft engine features)
1000 good (normal) engines & 20 anomalous engines

Training set (only good examples): 6000 good engines (y=0)
CV set: 2000 good engines (y=0), 10 anomalous (y=1)
Test set: 2000 good engines (y=0), 10 anomalous (y=1)

Evaluation of Anomaly Detection System

1) Fit *p(x)* on training set (only normal examples) $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

2) Check the performance on cross-validation (CV) set.
From a range of possible values of ϵ, choose the best threshold (ϵ) to optimize some performance metric.

p(x_CV)< ϵ (anomaly)
p(x_CV)> ϵ (normal)

Possible performance metrics

- True positive, true negative, false positive, false negative
- Precision/Recall
- F1-score

Accuracy is not a good metric because in most cases the classes are unbalanced (much more normal examples than anomalous).

3) Test the final model on test set.

p(x_test)< ϵ (anomaly)
p(x_test)> ϵ (normal)

Anomaly Detection vs. Supervised Learning

- Very small number of positive (anomalous) examples.
- Large number of negative (normal) examples.
- Many different “types” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like.
- Future anomalies may look nothing like any of the anomalous examples seen before.

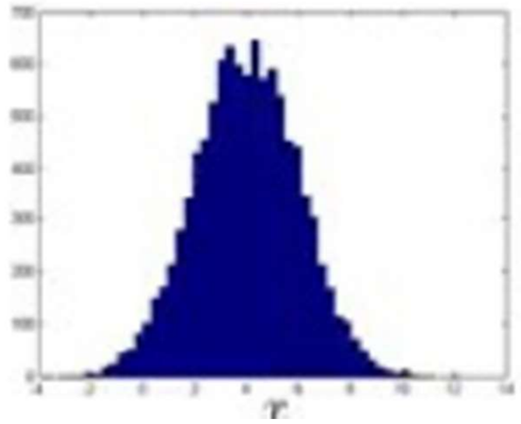
Exs.: fraud detection, manifacturing; monitoring computers in data centers

- Large number of positive and negative examples
- Enough positive examples for the algorithm to get a sense of what positive examples are like.
- Future positive examples likely to be similar to ones in training set.

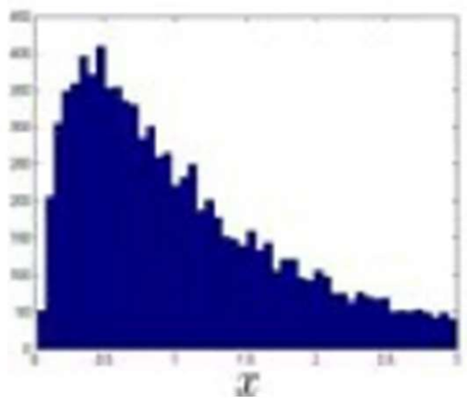
Exs.: Spam/fishing email classification; Cancer classification, weather prediction

Choosing Features

If features have Gaussian distribution =>
apply the presented anomaly detection algorithm .



If features do not have Gaussian distribution =>
play with different transformations of data to get more Gaussian curves.

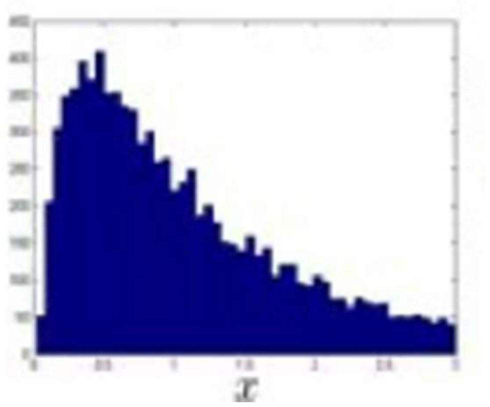


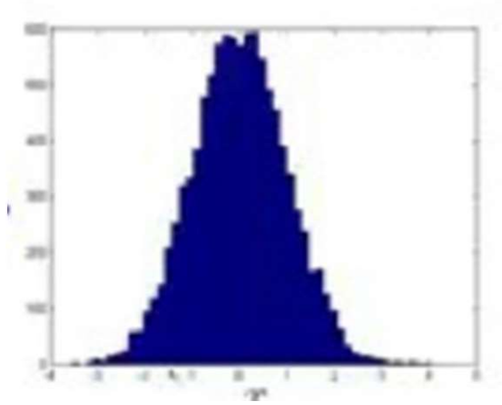
Choosing Features

Popular feature transformsations =>

- $log(x)$
- $log(x + c)$
- $\sqrt{x}$

for example: x => log(x) => much more Gaussian curve





Error Analysis for Anomaly Detection

Want: p(x) large for normal examples x;
p(x) small for anomalous examples x.

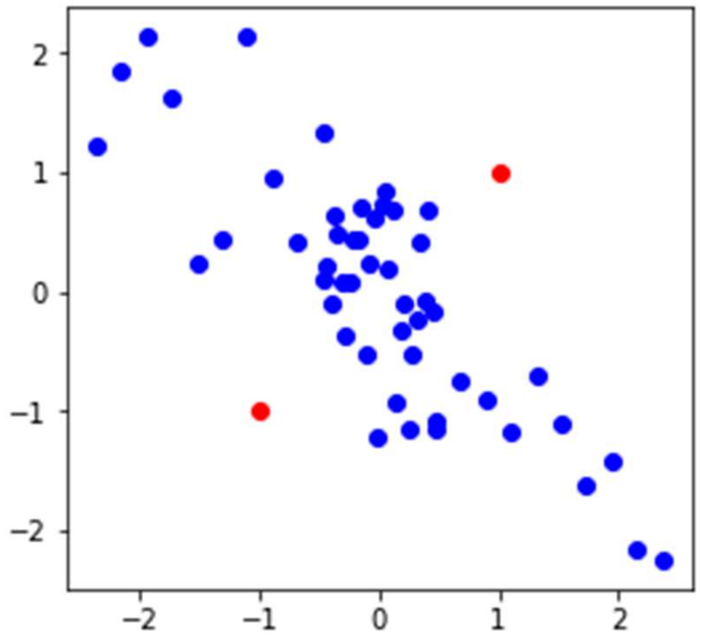
Most common problem:
p(x) is comparable (say both large) for normal and anomalous examples.

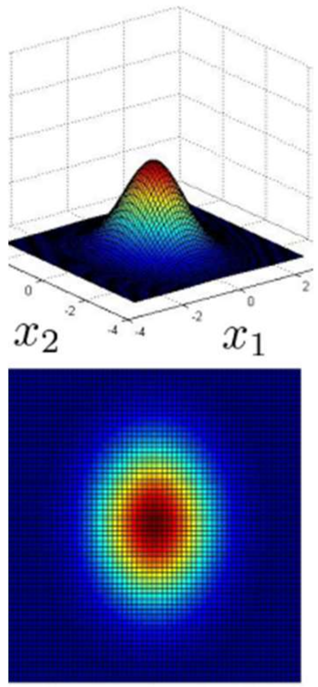
Solution: Make error analysis to create new features
Look at the anomalies the algorithm did not flag correctly (the mistakes) and try to create some new feature that may take unusually different (large or small) values in the event of anomaly and thus distinguish the abnormal ex.

Ex. Monitoring computers in data center
x1=memory use of computer
x2=number of disc accesses /sec
x3=CPU load
x4=network traffic

Feature engineering (new features)
x5=CPU load /network traffic
x6= (CPU load)^2 /network traffic

Multivariate Gaussian Distribution

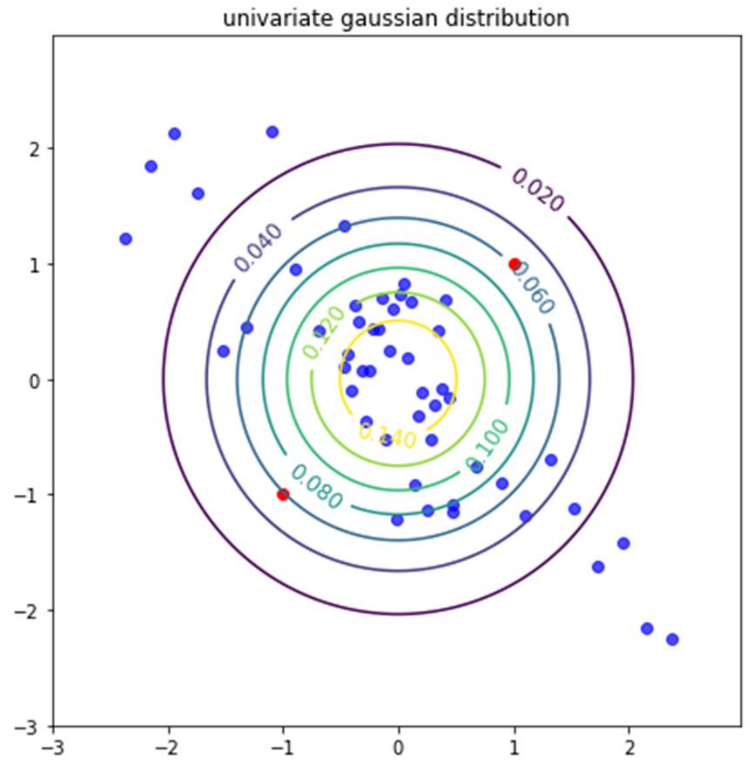


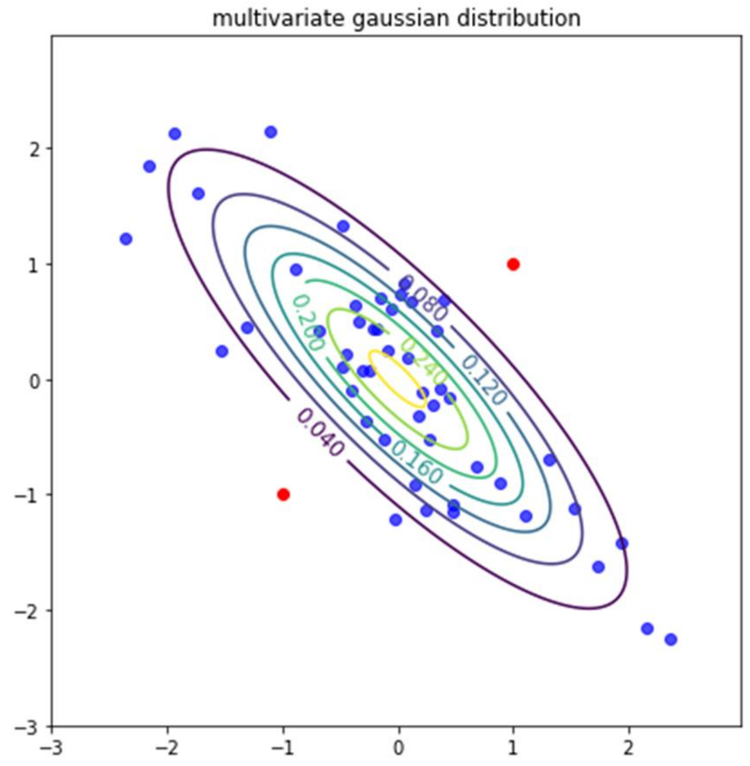


If we use univariate Gaussian distribution for this data, the contour plots (fixed value of the probability) will be circles (if both variances are the same) or axes alined elipces (if the variances are different) . The red points will have relatively high probability => not flaged as outliers.

But features x_1 and x_2 are negatively correlated (one increases, the other decreases). The assumption of independance is violated.
Red points are outliers.

Univariate vs.Multivariate Gaussian Distribution





Univariate Gaussian distribution considers separately probability models for each $p(x_1)$, $p(x_2)$ => it will not flag the red points as outliers.
Better use Multivariate Gaussian distribution.

Multivariate density estimation

Given training data, estimate μ (nx1 vector) and **Σ (nxn covariance matrix):**

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

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

For a new example x, compute:

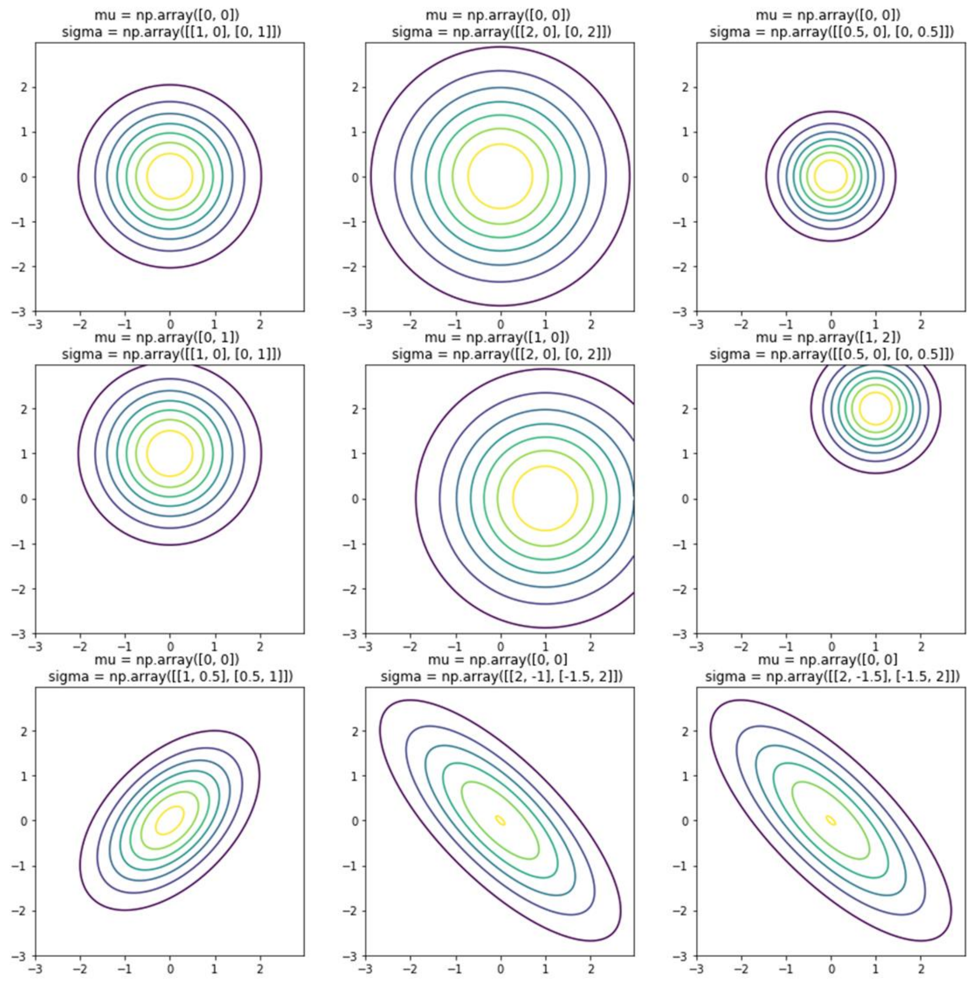
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}\,|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Flag as anomaly if $p(x)<\epsilon$.

Σ – nxn covariance matrix (the same as in Principal Component Analysis)
|Σ| - determinant of matrix Σ.
Σ - symmetric about the main diagonal.

Σ - major difference between univariate and multivariate Gaussian !!!

Effect of Mean and Covariance Matrix Shifting



μ shifts the center of the distribution.

Diagonal elements of Σ vary the spread of the distribution along the corresponding features

Off-diagonal elements of Σ (sigma) show the correlation among the features:

Positive off-diagonal values of Σ => positive correlation

Negative off-diagonal values of Σ => negative correlation

Univariate Gaussian distribution is a special case when off-diagonal values of Σ are 0.

Original vs. Multivariate Gaussian models for Anomaly detection

Original model

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

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

$$p(x) = p(x_1;\mu_1,\sigma_1^2)p(x_2;\mu_2,\sigma_2^2)\cdots p(x_n;\mu_n,\sigma_n^2)$$

Manually create features to capture anomalies
(e.g. x_new=CPU load /network traffic)

Computationally cheaper, scales better to large number of features (n)

OK even if the training set size (m) is small

Multivariate Gaussian

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

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

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

Automatically captures correlations between features

Computationally more expensive, takes time to compute inverse of Σ if number of features (n) is large

Must have training set size (m) >> number of features (n) , otherwise Σ is singular and not invertible.
In practice m>10*n