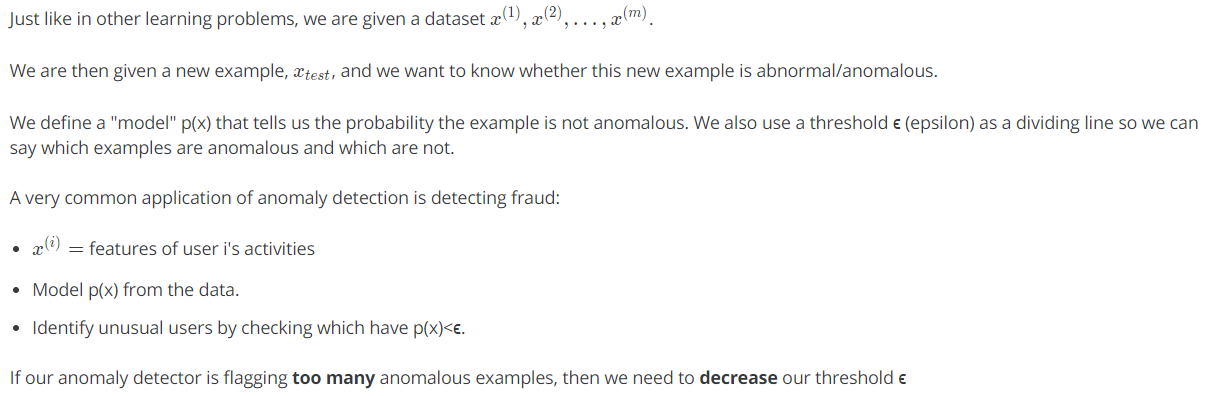
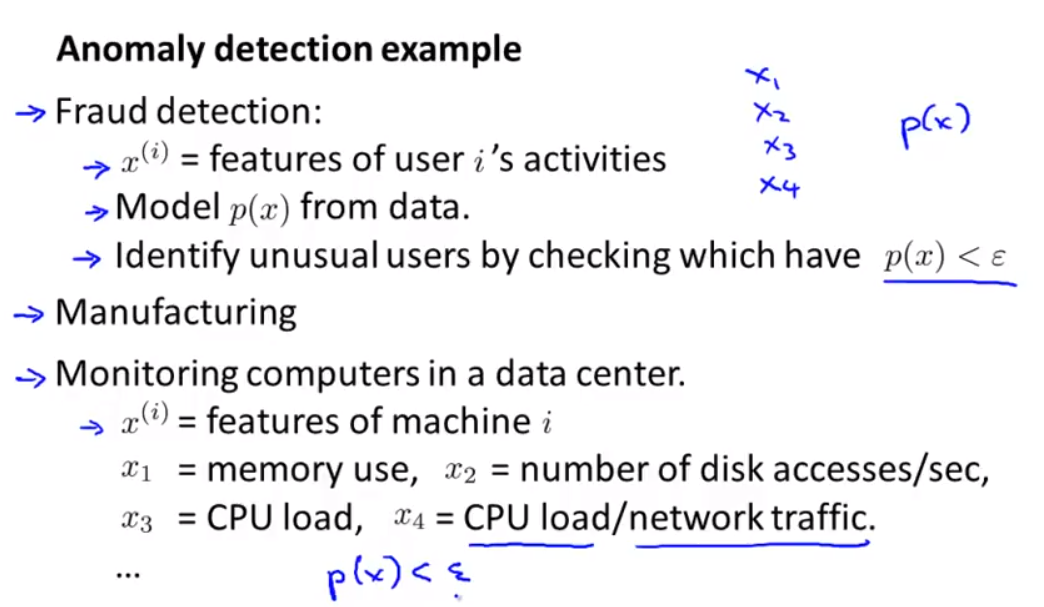
Anomaly Detection System

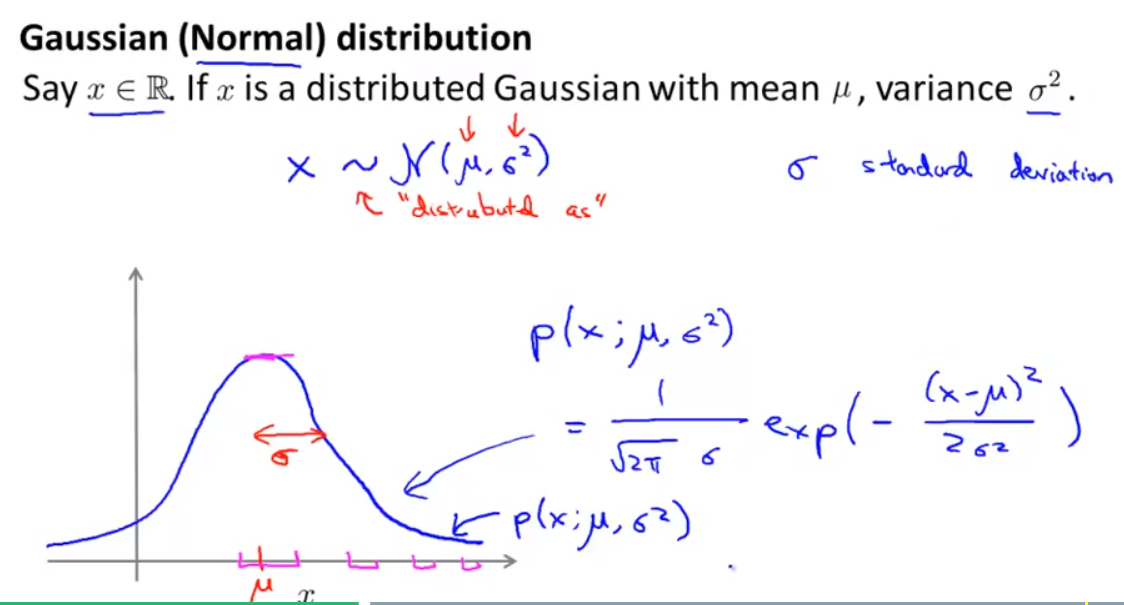
# Density Estimation

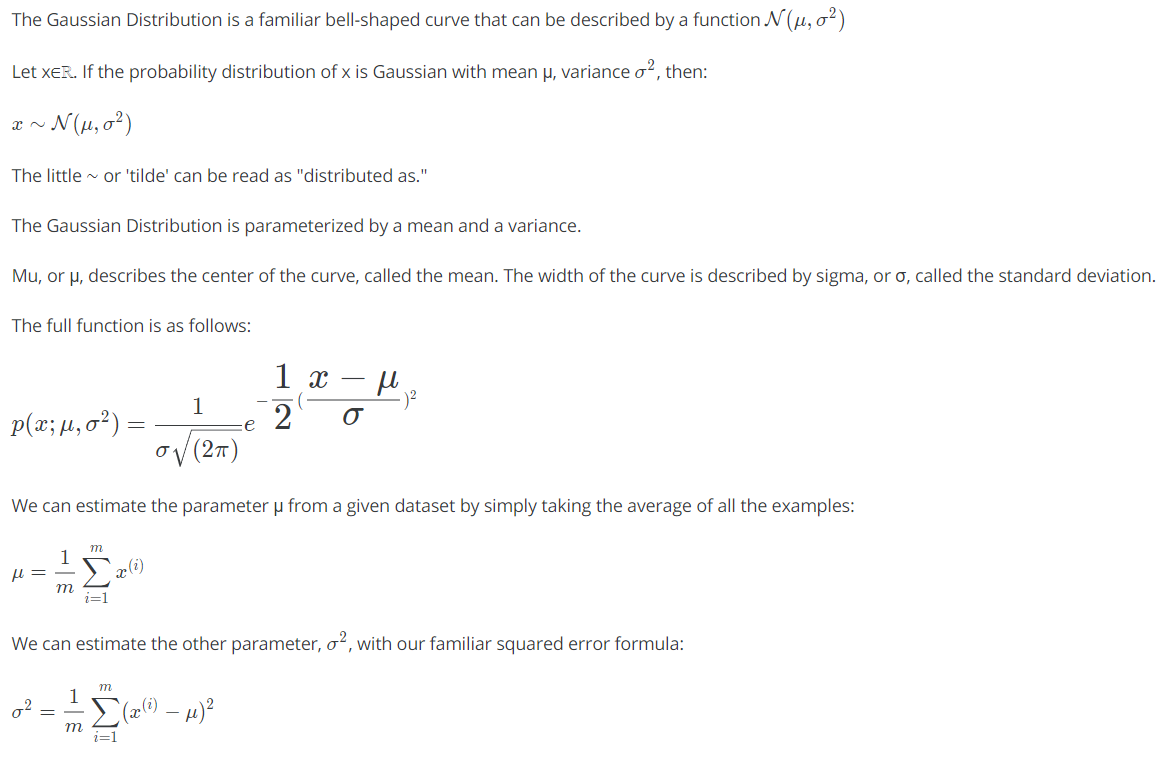
## Problem motivation



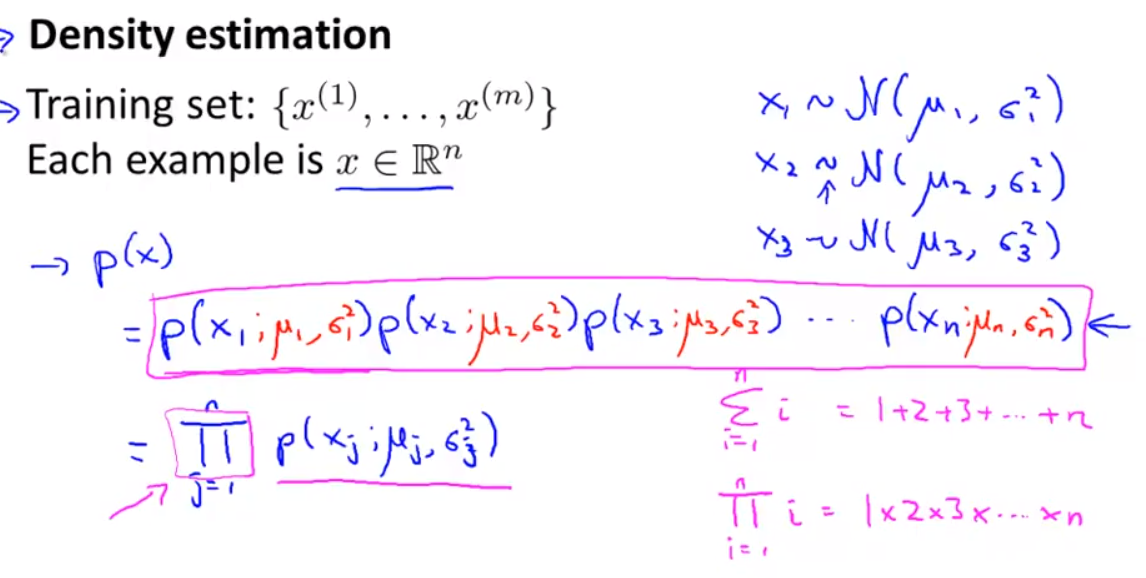


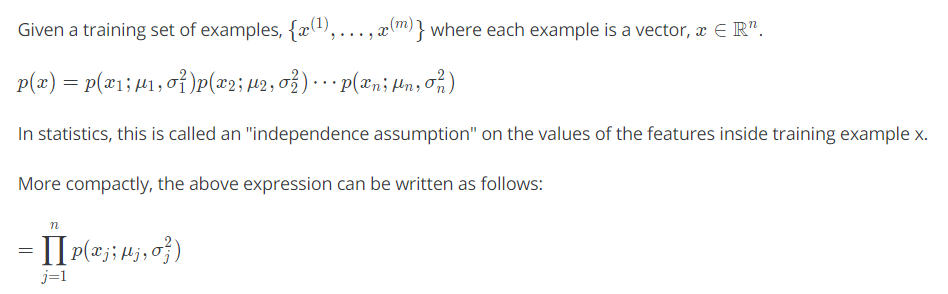
## Gaussian or Normal Distribution



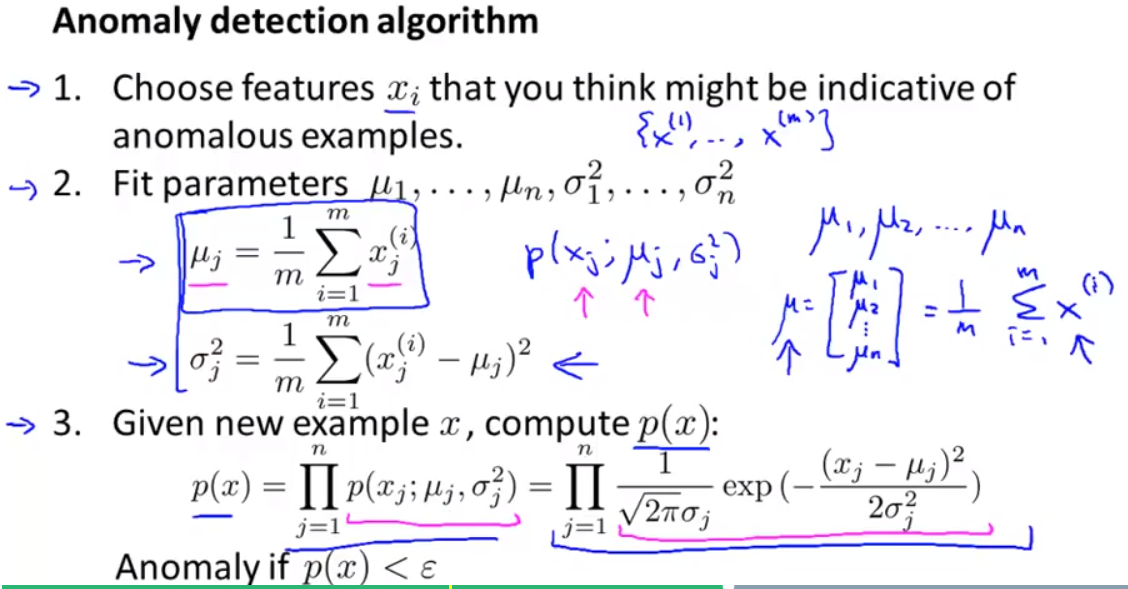


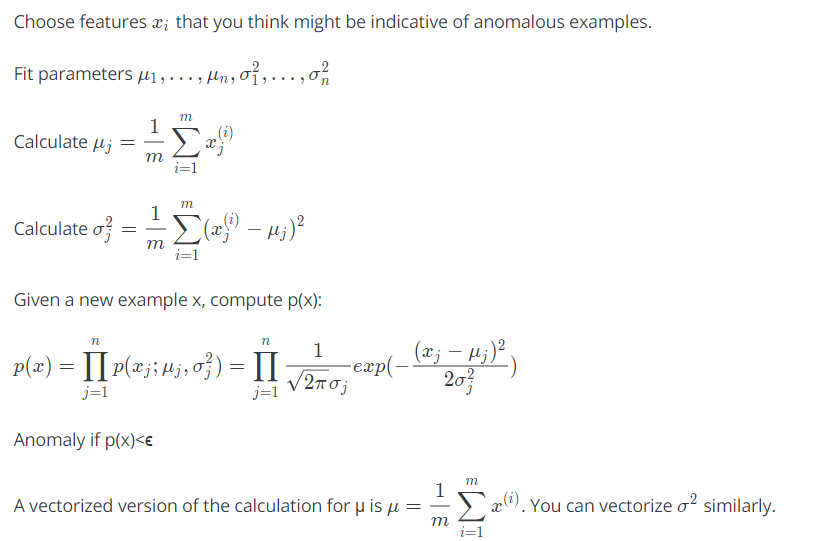
## Algorithm





The algorithm can then be written as: -





# Building an Anomaly Detection System

## Developing and evaluating an Anomaly Detection System

To evaluate our learning algorithm, we take some labelled data, categorized into anomalous and non-anomalous examples (y = 0 if normal, y = 1 if anomalous).

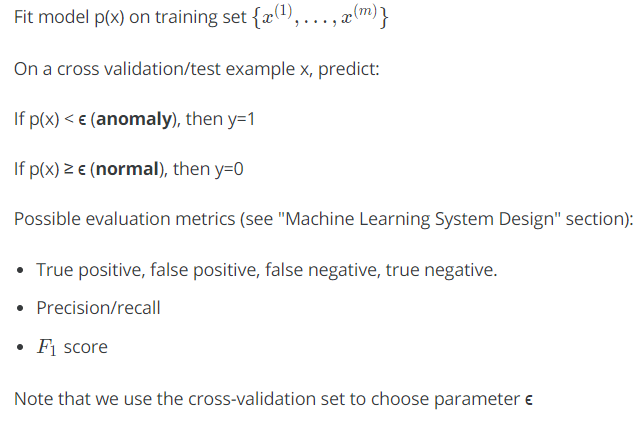
Among that data, take a large proportion of **good**, non-anomalous data for the training set on which to train p(x).

Then, take a smaller proportion of mixed anomalous and non-anomalous examples (you will usually have many more non-anomalous examples) for your cross-validation and test sets.

For example, we may have a set where 0.2% of the data is anomalous. We take 60% of those examples, all of which are good (y=0) for the training set. We then take 20% of the examples for the cross-validation set (with 0.1% of the anomalous examples) and another 20% from the test set (with another 0.1% of the anomalous).

In other words, we split the data 60/20/20 training/CV/test and then split the anomalous examples 50/50 between the CV and test sets.

**Algorithm evaluation**: -



## Anomaly Detection vs. Supervised Learning

When do we use anomaly detection and when do we use supervised learning?

Use anomaly detection when...

* We have a very small number of positive examples (y=1 ... 0-20 examples is common) and a large number of negative (y=0) examples.
* We have many different "types" of anomalies and it is 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 we've seen so far.

Use supervised learning when...

* We have a large number of both positive and negative examples. In other words, the training set is more evenly divided into classes.
* We have enough positive examples for the algorithm to get a sense of what new positives examples look like. The future positive examples are likely similar to the ones in the training set.

## Choosing what features to use

The features will greatly affect how well your anomaly detection algorithm works.

We can check that our features are **gaussian** by plotting a histogram of our data and checking for the bell-shaped curve.

Some **transforms** we can try on an example feature x that does not have the bell-shaped curve are:

* log(x)
* log(x+1)
* log(x+c) for some constant
* sqrt{x}​
* x^(1/3)

We can play with each of these to try and achieve the gaussian shape in our data.

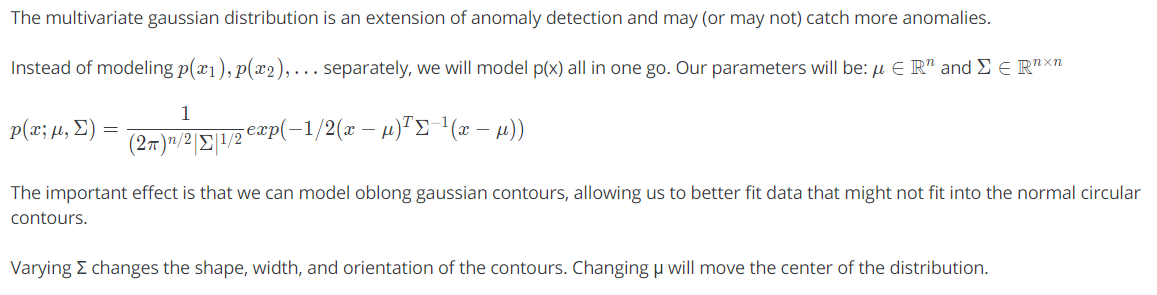
There is an **error analysis procedure** for anomaly detection that is very similar to the one in supervised learning.

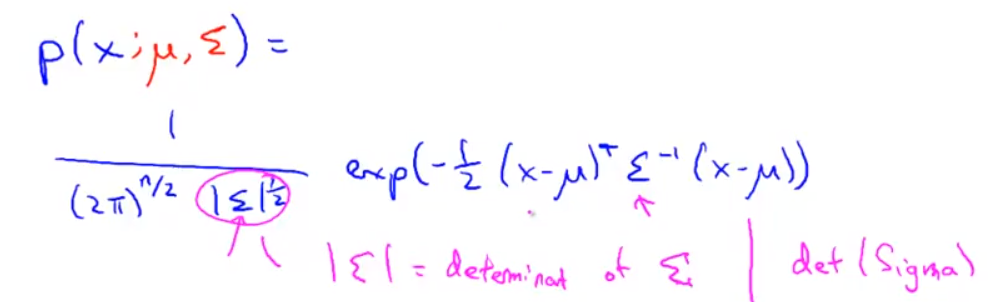
* Our goal is for p(x) to be large for normal examples and small for anomalous examples.
* One common problem is when p(x) is similar for both types of examples. In this case, you need to examine the anomalous examples that are giving high probability in detail and try to figure out new features that will better distinguish the data.

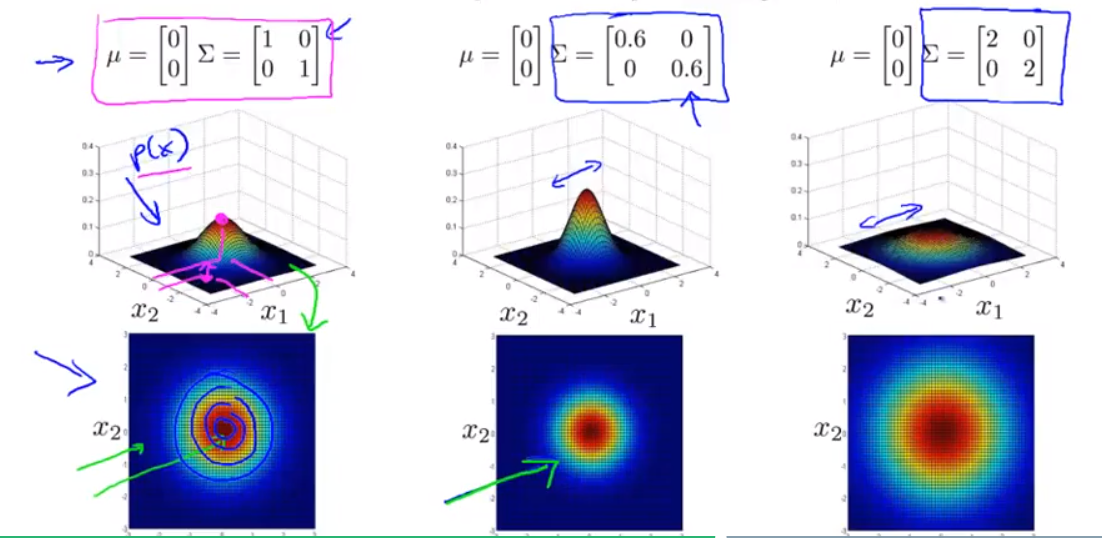
In general, choose features that might take on unusually large or small values in the event of an anomaly.

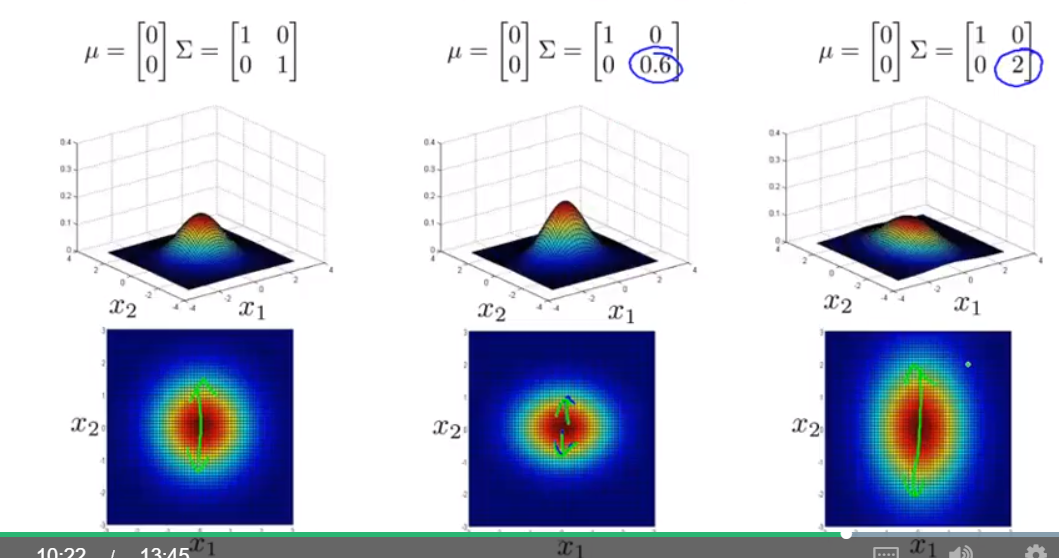
# Multivariate Gaussian Distribution

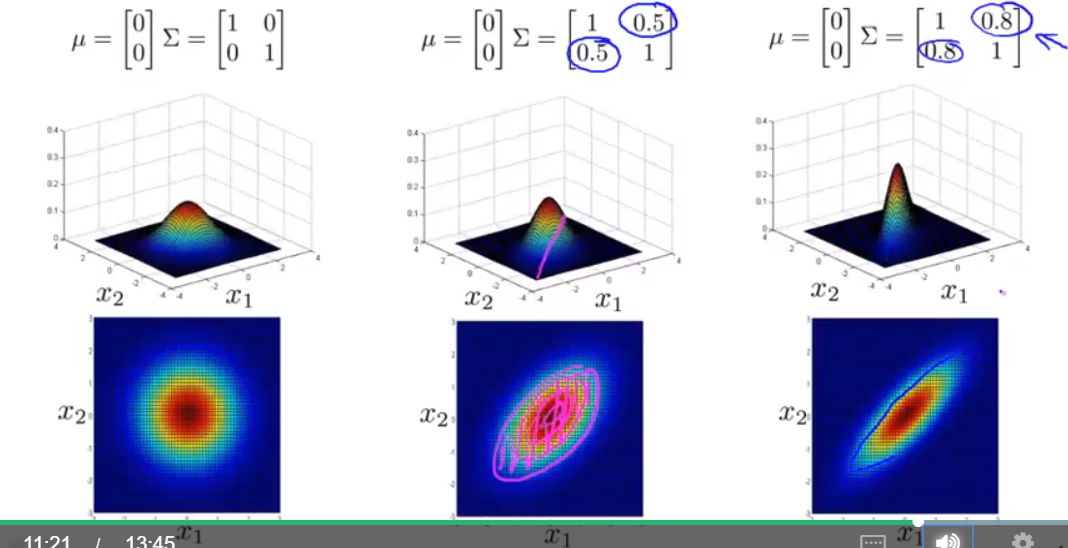
## Multivariate gaussian distribution







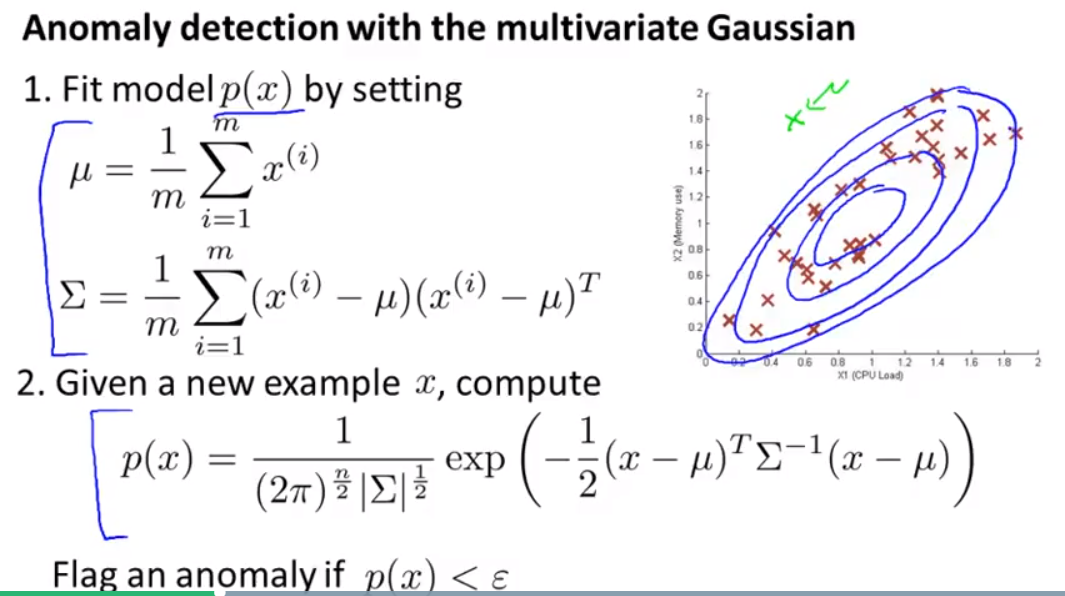




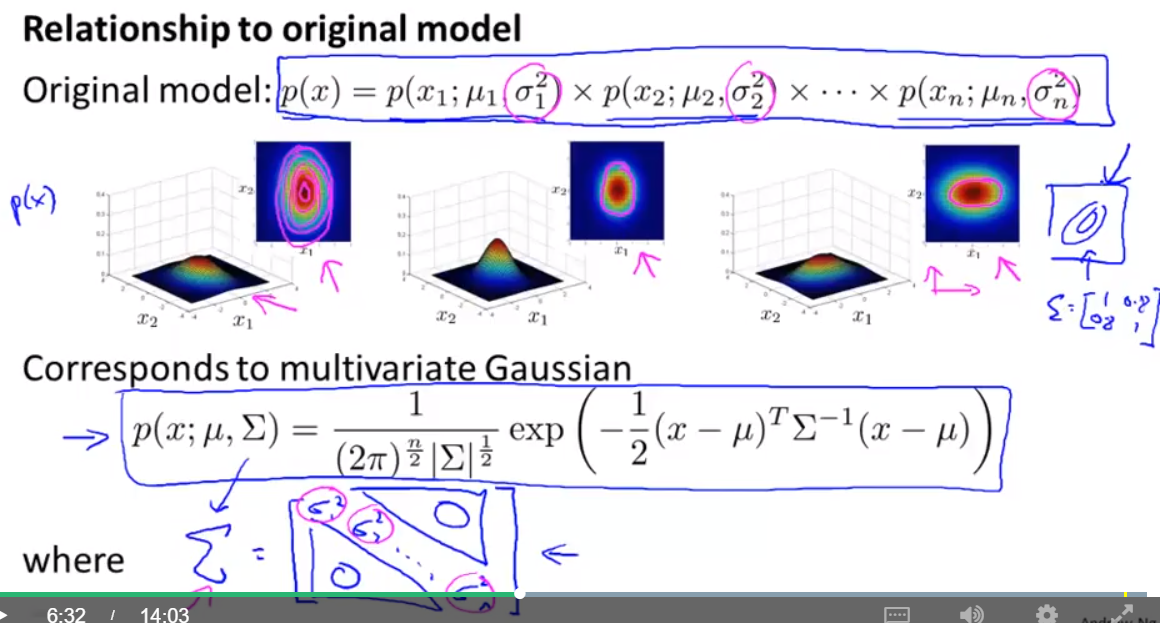


## Anomaly detection using multivariate gaussian distribution

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) < ϵ.



* The original model for p(x) corresponds to a multivariate Gaussian where the contours of *p*(*x*;*μ*,Σ) 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).

