Finding unusual events

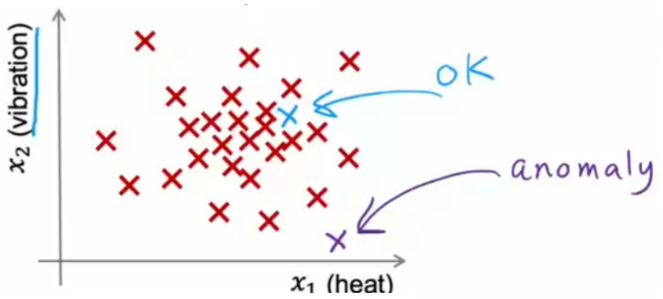
The Second unsupervised learning algorithm. Anomaly detection algorithms look at an unlabeled dataset of normal events and thereby learns to detect or to raise a red flag for if there is an unusual or an anomalous event.

Anomaly detection to detect for possible problems with aircraft engines
x_1 = heat generated
x_2 = vibration intensity

If m aircraft engines were manufactured, collect the existing data
Dataset: { $x^{(1)}, x^{(2)}, \ldots x^{m}$ }

After the learning algorithm has seen these m examples of how aircraft engines typically behave in terms of how much heat is generated and how much they vibrate. If a brand new aircraft engine were to roll off the assembly line and it had a new feature vector given by Xtest,
New engine: x_{test}

plot the examples x1 through xm over here via these crosses where each cross each data point in this plot corresponds to a specific engine with a specific amount of heat and specific amount of vibrations.



New aircraft (Blue) data point seems to be similar to other engines
In case, if the data point (purple) at the bottom, then it looks different

The most common way to carry out anomaly detection is through a technique called **density estimation**. And what that means is, when you're given your training sets of these m examples, the first thing you do is build a model for the probability of x. In other words, the learning algorithm will try to figure out what are the values of the features x1 and x2 that have high probability and what are the values that are less likely or have a lower chance or lower probability of being seen in the data set.

In this example, the ellipse in the middle, would have high probability, and the second ellipse around it may have a little bit lower probability and the outer ring will have much lower probability

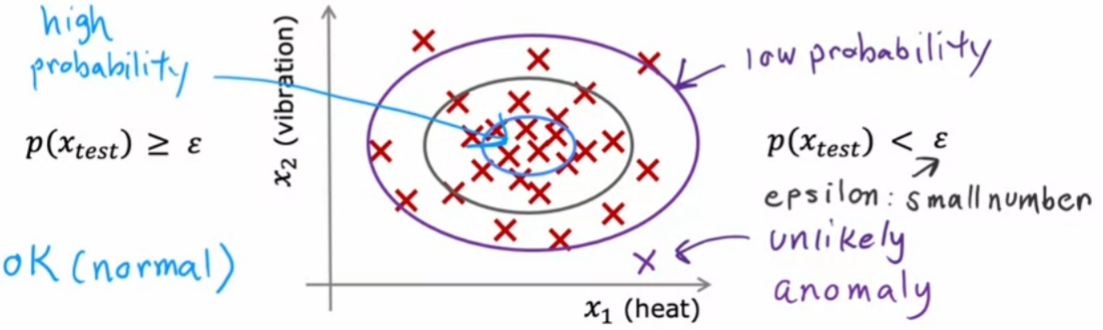
x_{test} is a low probability and would flag this as an anomaly.
When you are given the new test example Xtest. What you will do is then compute the probability of Xtest. And if it is small number which is indicated by epsilon

Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ probability of x being seen in dataset

Model $p(x)$

Is x_{test} anomalous?



If the new engine data points are very close to the data points in the inner middle ellipses then they are quite high probability and they are not anomalies.

Anomaly detection example

how often log in?
how many web pages visited?
transactions?
posts? typing speed?

Fraud detection:

- $x^{(i)}$ = features of user i's activities
- Model $p(x)$ from data.
- Identify unusual users by checking which have $p(x) < \varepsilon$

perform additional checks to identify real fraud vs. false alarms

Manufacturing:
$x^{(i)}$= features of product i

airplane engine
circuit board
smartphone

ratios

Monitoring computers in a data center:
$x^{(i)}$= features of machine i

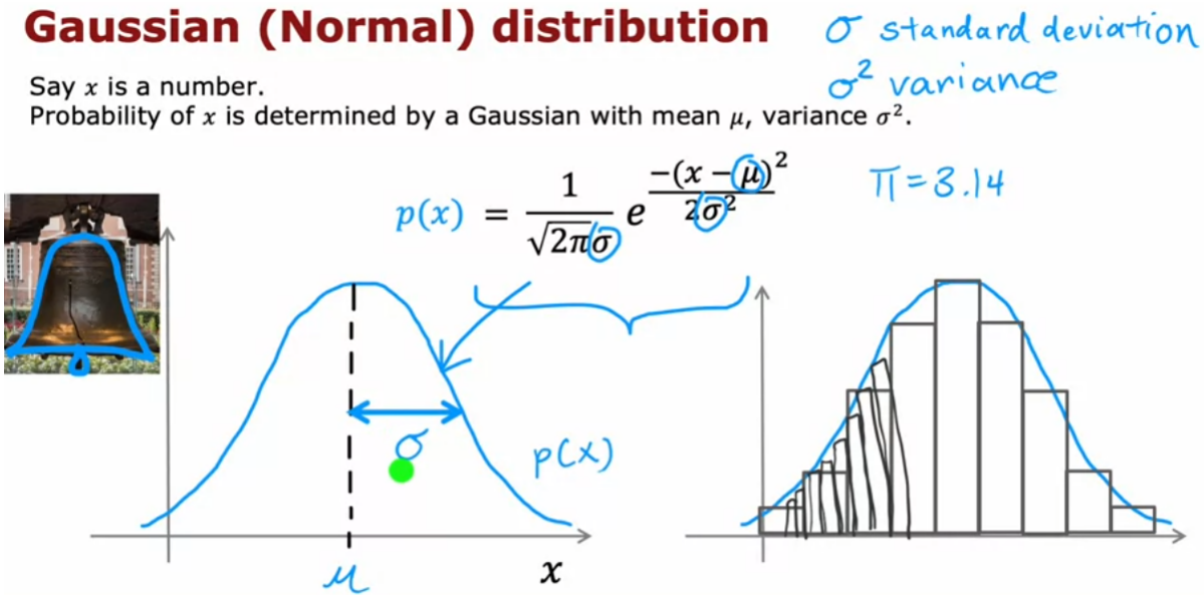
- x_1 = memory use,
- x_2= number of disk accesses/sec,
- x_3 = CPU load,
- x_4= CPU load/network traffic.

Find a pattern for a user (how many webpages which pages do they normally visit etc) and once identified escalate/investigate further.
Anomaly detection is used in many areas, fraud detection, manufacturing etc
Monitor computers in a network to detect any abnormal activities.

Gaussian (normal) distribution

In order to apply anomaly detection, we're going to need to use the Gaussian distribution, which is also called the normal distribution. (bell-shaped distribution)

p(x) probability of x



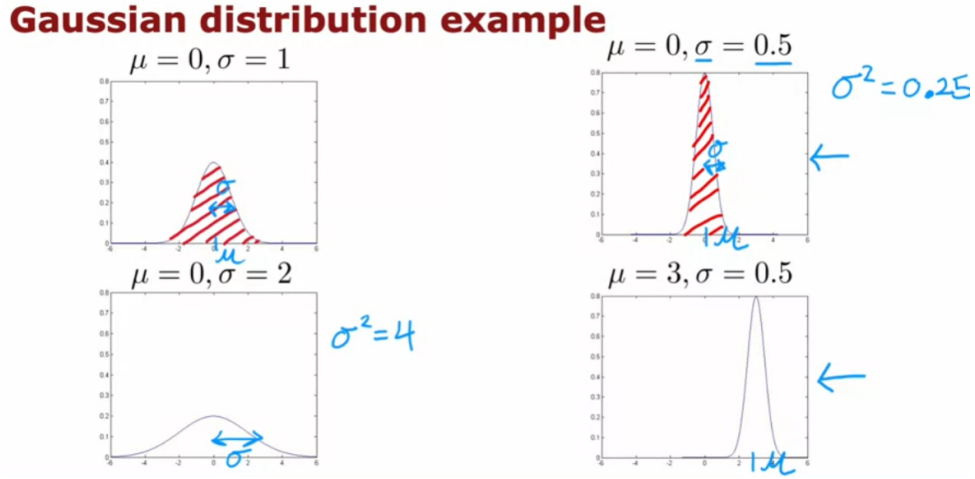
100 numbers drawn from this probability distribution, and you were to plot a histogram of these 100 numbers drawn from this distribution, you might get a histogram that looks like this. It looks vaguely bell-shaped.

The formula for p of x is given by this expression

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad \pi = 3.14$$

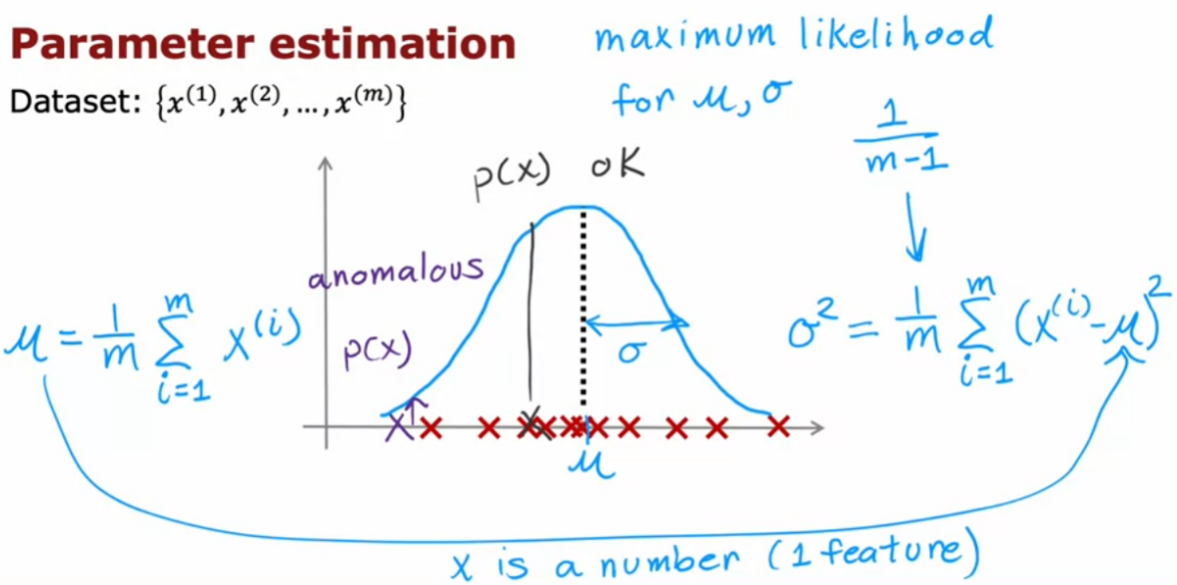
For any given value of Mu and Sigma, if you were to plot this function as a function of x, you get this type of bell-shaped curve that is centred at Mu, and with the width of this bell-shaped curve being determined by the parameter Sigma

centered at mu =0, but sigma is 1 and 0.5 (variance is sigma squared)



Probabilities should add up to 1 which is why when the Gaussian distribution becomes skinnier, it has to become taller

This is how different choices of Mu and Sigma affect the Gaussian distribution. When you're applying this to anomaly detection, You are given a dataset of m examples, and here x is just a number. Here, are plots of the training sets with 11 examples. What we have to do is try to estimate what a good choice is for the mean parameter Mu, as well as for the variance parameter Sigma squared.



compute Mu and Sigma squared mathematically
First mu, and then variance, replace the mu with result of the first calculation

It turns out that if you implement these two formulas encodes with this value for Mu and this value for Sigma squared, then you pretty much get the Gaussian distribution

In statistics, these formulas are called “maximum likelihood” estimates for mu and sigma.
some use 1/m-1 instead of 1/m

p(x) is low at the fringes and they are anomalous

Note that x is a number and single feature in the above example.

For practical anomaly detection applications, you usually have a lot of different features. You've now seen how the Gaussian distribution works. If x is a single number, this corresponds to if, say you had just one feature for your anomaly detection problem. But for practical anomaly detection applications, you will have many features, two or three or some even larger number n of features. Let's take what you saw for a single Gaussian and use it to build a more sophisticated anomaly detection algorithm.

Anomaly detection algorithm

A training set x1 through xm, where here each example x has n features.
In the case of aircraft engines, there were 2 features, heat and vibrations (n = 2) but this will be 10s or 100s for real world applications.

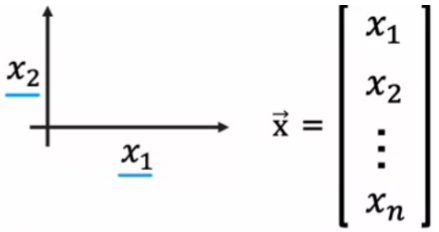
Carry out a density estimation, which is build a model or estimation of probability of p(x), probability of feature vector.

x is a feature vector x1, x2, … xn and p(x) is the probability of x1 times probability of x2 time upto probability of xn
(In statistics, this equation corresponds to assuming that the features x1, x2 and so on up to xn are statistically independent. But it turns out this algorithm often works fine even that the features are not actually statistically independent.)

Mean and variance of each feature.

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)}\}$
Each example $\vec{x}^{(i)}$ has n features



$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \cdots * p(x_n; \mu_n, \sigma_n^2)$$

If an aircraft engine there's a 1/10 chance that it is really hot, and maybe there is a 1 in 20 chance that it vibrates really hard. Then, what is the chance that it runs really hot and vibrates really hard. (1 in 200)

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

"add" "multiply"

$$p(x_1 = \text{high temp}) = 1/10$$
$$p(x_2 = \text{high vibra}) = 1/20$$
$$p(x_1, x_2) = p(x_1) * p(x_2)$$
$$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$$

Equation is written as above where Sigma is for addition, (pi symbo?) is for multiplication

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

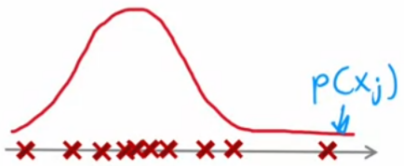
Vectorized formula

$$\vec{\mu} = \frac{1}{m} \sum_{i=1}^{m} \vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_n \end{bmatrix}$$

3. Given new example x, compute $p(x)$:

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

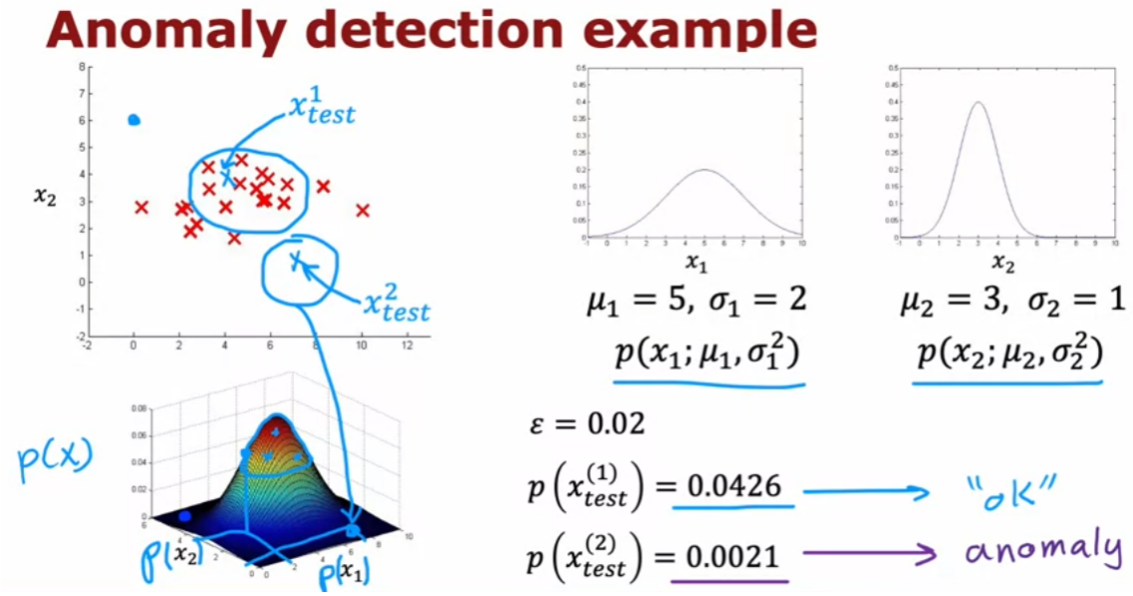
Anomaly if $p(x) < \varepsilon$



Steps
Choose features
Calculate the mean and variance

Given a new example x, calculate the p(x)

What anomaly detection is doing into the algorithm is a systematic way of quantifying whether or not this new example x has any features that are unusually large or unusually small.

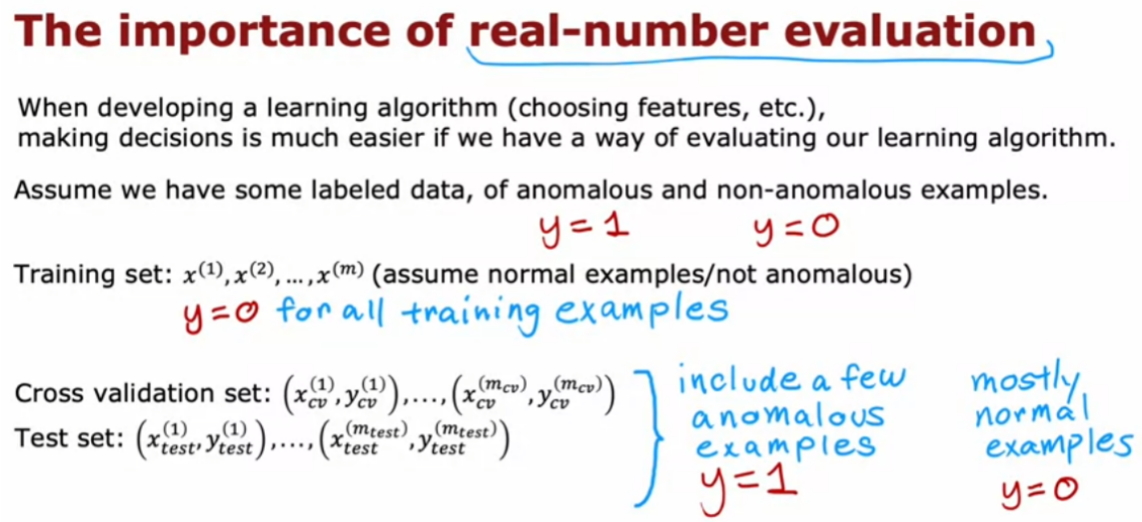


two example features, one is mu=5 and sigma=2 and other is mu=3 and sigma=1
if epsilon = 0.02, x1 is in the centre and x2 is at the bottom. How to select the epsilon value.

Developing and evaluating an anomaly detection system

some practical tips for developing an anomaly detection system. One of the key ideas will be that if you can have a way to evaluate a system, even as it's being developed, you'll be able to make decisions and change the system and improve it much more quickly.

change feature or parameter



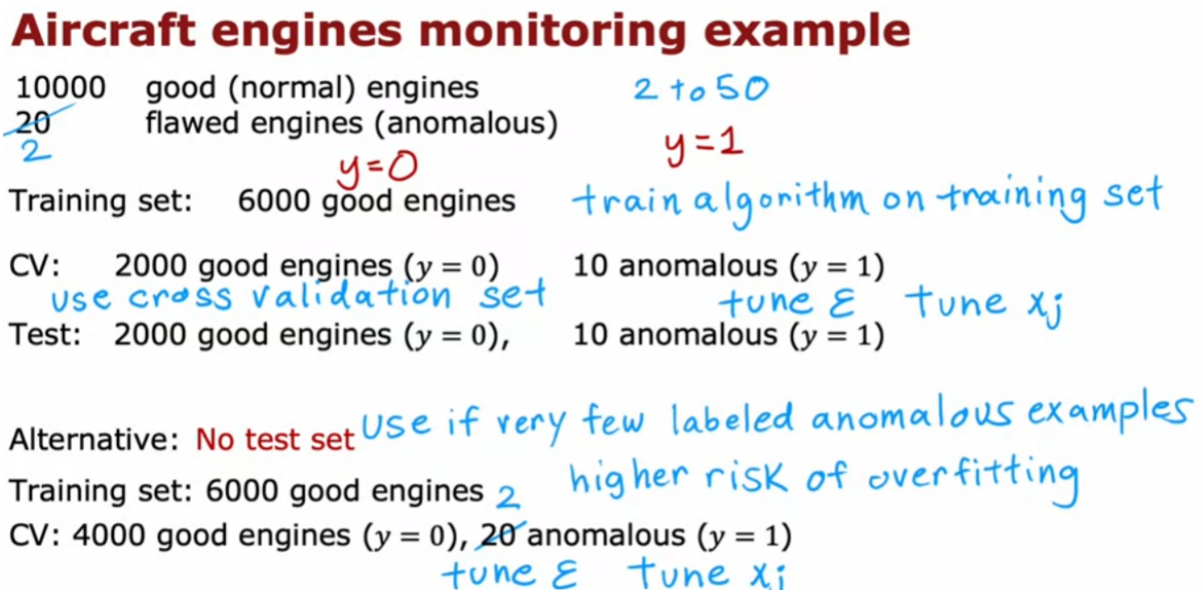
When you are developing a learning algorithm, say choosing different features or trying different values of the parameters like epsilon, making decisions about whether or not to change a feature in a certain way or to increase or decrease epsilon or other parameters, making those decisions is much easier if you have a way of evaluating the learning algorithm. This is sometimes called real number evaluation, meaning that if you can quickly change the algorithm in some way, such as change a feature or change a parameter and have a way of computing a number that tells you if the algorithm got better or worse, then it makes it much easier to decide whether or not to stick with that change to the algorithm. This is how it's often done in anomaly detection.
Which is, even though we've mainly been talking about unlabeled data, I'm going to change that assumption a bit and assume that we have some labelled data, including just a small number usually of previously observed anomalies. Maybe after making airplane engines for a few years, you've just seen a few airplane engines that were anomalous, and for examples that you know are anomalous,

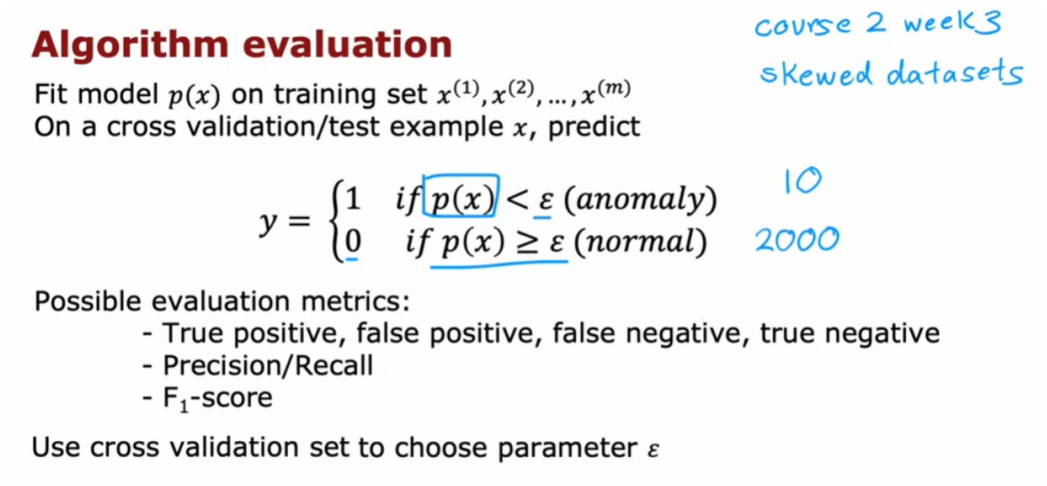
I'm going to associate a label y = 1 to indicate this anomalous, and for examples that we think are normal, I'm going to associate the label y = 0.

The training set that the anomaly detection algorithm will learn from is still this unlabeled training set of x1 through xm, and I'm going to think of all of these examples as ones that we'll just assume are normal and not anomalous, so y is equal to 0.

In practice, you have a few anomalous examples where to slip into this training set, your algorithm will still usually do okay. To evaluate your algorithm, come up with a way for you to have a real number evaluation,

it turns out to be very useful if you have a small number of anomalous examples so that you can create a cross validation set, which I'm going to denote x_cv^1, y_cv^1 through x_cv^mcv, y_cv^mcv. This is similar notation as you had seen in the second course of this specialization. As similarly, have a test set of some number of examples where both the cross validation and test sets hopefully includes a few anomalous examples. In other words, the cross validation and test sets will have a few examples of y equals 1, but also a lot of examples where y is equal to 0.





The intuition I hope you get is to use the cross-validation set to just look at how many anomalies is finding and also how many normal engines is incorrectly flagging as an anomaly. Then to just use that to try to choose a good choice for the parameter Epsilon. You find that the practical process of building an anomaly detection system is much easier if you actually have just a small number of labeled examples of known anomalies.

Now, this does raise the question, if you have a few labeled examples, since you'll still be using an unsupervised learning algorithm, why not take those labeled examples and use a supervised learning algorithm instead? In the next video, let's take a look at a comparison between anomaly detection and supervised learning and when you might prefer one over the other.

Anomaly detection vs. supervised learning

When you have a few positive examples with Michael's 1 and a large number of negative examples say y = 0? When should you use anomaly detection and when should you use supervised learning?

Anomaly Detection	Supervised Learning
Very small number of positive examples (y=1) (0-20 is common) Large number of negative (y=0)	Large number of positive and negative 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 so far.	Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.
Fraud Detection	Email Spam classification
Manufacturing - Finding new unseen defects (Aircraft engines)	Manufacturing - Finding known, previously seen defects
Monitoring Machines in a data center	Weather prediction (sunny/rainy) Diseases classification

Anomaly Detection
Small number of positive examples, (0-20) and large number of negative examples with which to try to build a model for p(x)
The parameters for p(x) are learned only from the negative examples and this is much smaller. So the positive examples is only used in your cross validation set and test set for parameter tuning and for evaluation.

What anomaly detection does is it looks at the normal examples that is the y = 0 negative examples and just tries to model what they look like. And anything that deviates a lot from normal It flags as an anomaly. Including if there's a brand new way for an aircraft engine to fail that had never been seen before in your data set.

i.e
Financial Fraud (new ways are invented)

Supervised Learning
When there are a large number of positive and negative examples.

Large number of examples available to get a sense of positive examples and future positive examples likely to be similar

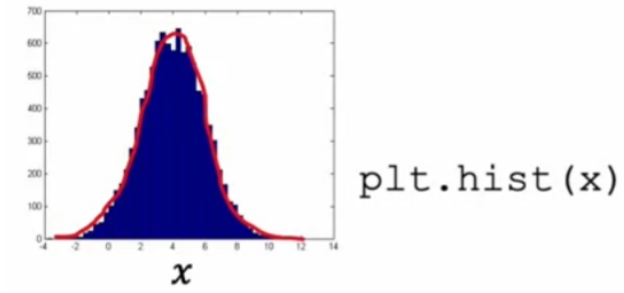
i.e
Email spam detection (Even if there are wide variety, they are similar in some ways)

Choosing what features to use

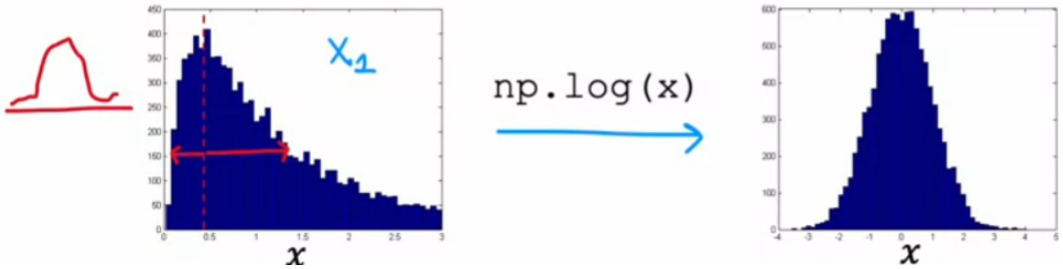
choosing a good choice of features turns out to be really important. In supervised learning, if you don't have the features quite right, or if you have a few extra features that are not relevant to the problem, that often turns out to be okay. Because the algorithm has to supervised signal that is enough labels why for the algorithm to figure out what features ignore, or how to re scale feature and to take the best advantage of the features you do give it. But for anomaly detection which runs, or learns just from unlabeled data, is harder for the anomaly to figure out what features to ignore. So I found that carefully choosing the features, is even more important for anomaly detection, than for supervised learning approaches.

anomaly detection algorithm, is to try to make sure the features you give it are more or less Gaussian
If features are not Gaussian, sometimes you can change it to make it a little bit more Gaussian

plt.hist(x) - this is how it should be for anomaly detection



Suppose if it is look like the one on left, need to transformed it to the one on right



in addition to log(x)

log(x) + C	log(x) + 1
sqrt (x)	$\sqrt{x}$
⅓ power	$x^{\frac{1}{3}}$

larger value of C, will end up transforming this distribution less. But in practice, try a bunch of different values of C,

With jupyter notebook "Choosing Features"

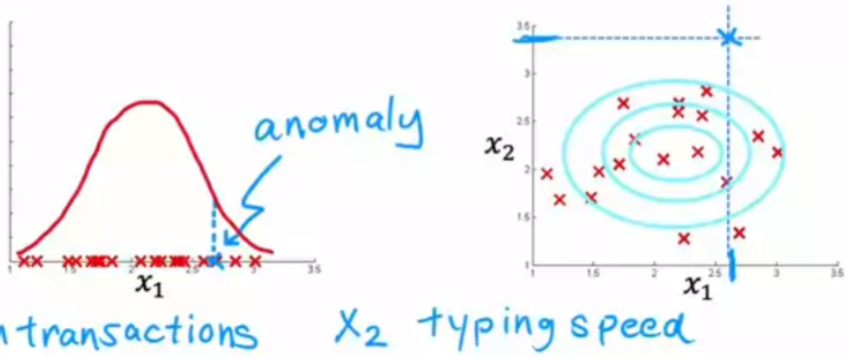
```
from scipy.stats import skewnorm
import matplotlib.pyplot as plt

numValues = 10000
maxValue = 100
skewness = 20 # Negative values are left skewed, positive values are right skewed.

random = skewnorm.rvs(a = skewness, loc=maxValue, size=numValues) # Skewnorm function
random = random - min(random) # Shift the set so the minimum value is equal to zero.
random = random / max(random) # Standardize all the values between 0 and 1
random = random * maxValue    # Multiply the standardized values by the maximum value.

x =  random

plt.hist(x, bins=50);
plt.hist(x, bins=50, color='r');
plt.hist(x**0.4, bins=50);

import numpy as np
#plt.hist(np.log(x), bins=50)   # This will result in an error as the x contains 0
# np.min(x)
plt.hist(np.log(x+0.001), bins=50); # trick is to add a tiny number

plt.hist(np.log(x+1), bins=50);

plt.hist(np.log(x+7.5), bins=50);
```

So, by trying things out in Jupiter notebook, you can try to pick a transformation that makes your data more Gaussian. And just as a reminder, whatever transformation you apply to the training set, please remember to apply the same transformation to your cross validation and test set data as well.

Error analysis for anomaly detection

Want $p(x) \geq \varepsilon$ large for normal examples x.
$p(x) < \varepsilon$ small for anomalous examples x.

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



Other than making sure that your data is approximately Gaussian, after you've trained your anomaly detection algorithm, if it doesn't work that well on your trust validation set, you can also carry out an error analysis process for anomaly detection.

What we want is for P(X) to be large. For normal examples X, so greater than equal to epsilon, and p(X) to be small or less than epsilon, for the anomalous examples X.

The most common problem that you may run into is that, P(X) is comparable in value say is, large for both normal and for anomalous examples. In the above fig on the right dataset, you might fit gaussian into it and the example (blue) in your cross validation set or test set, that is anomalous then this has high probability.
Even though this is an anomaly, P(X) is actually pretty large. And so the algorithm will fail to flag this particular example as an anomaly.
In that case, look at the example and find out why it looked an anomaly, even if feature x1 took on values similar to other training examples, identify another new feature x2 which can distinguish this example from the normal examples.
Adding that feature, can help improve the performance of the algorithm.

if x1 is the number of transactions made, adding new feature x2 typing speed in a fraud detection example and plot the data (right) then it becomes easier for the anomaly detection to recognize x2 is anomalous user.

the learning anomaly may fit a Gaussian distribution that assigns high probability to points in the centre and little bit lower in the surrounding areas and becomes easier to detect the (x1, x2) easily.

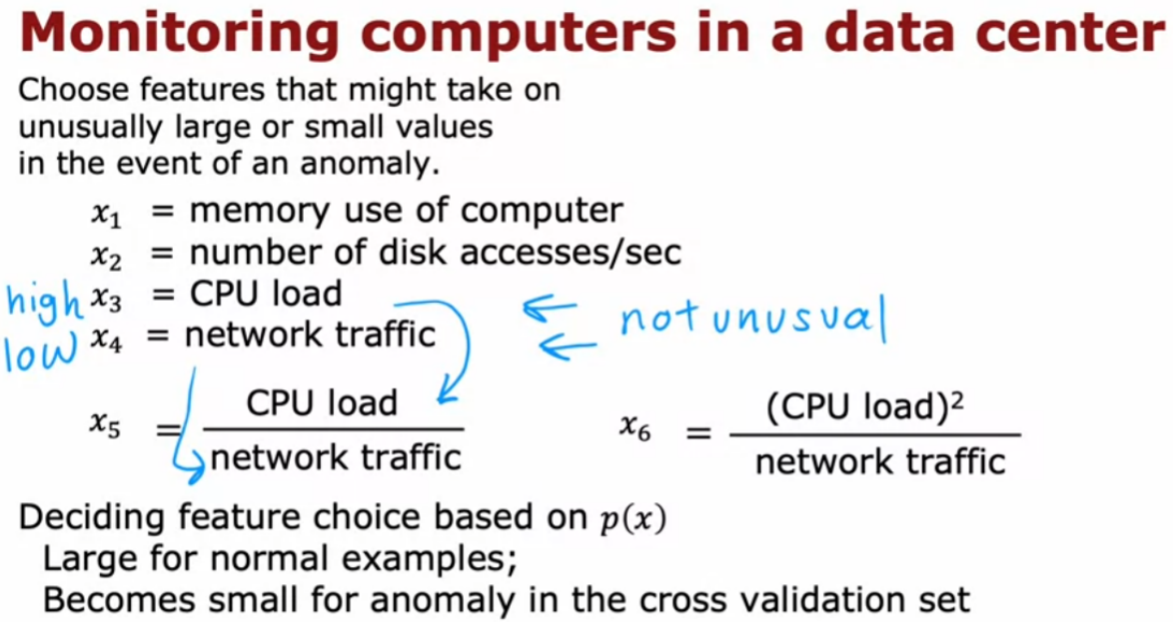
Another example
Monitoring computers in a data centre
Choose features that might take on unusually large or small values in the event of an anomaly.

x1 = memory use of computer
x2 = number of disk accesses/sec
x3 = CPU load
x4 = network traffic

if there is a computer with high CPU load and low network traffic (Usually data center computer will have high CPU load and high network traffic or vice versa)

In that case a new feature ratio of CPU load to network traffic and this feature may flagg future examples like the specific machine you may be seeing as anomalous.

x5 = CPU load / network traffic
or
x6 = $(\text{CPU load})^2$ / network traffic



Practice Quiz: Anomaly detection

1. Question 1
You are building a system to detect if computers in a data center are malfunctioning. You have 10,000 data points of computers functioning well, and no data from computers malfunctioning. What type of algorithm should you use?

- Anomaly detection ANSWER
- Supervised learning

Correct
Creating an anomaly detection model does not require labeled data.

2. Question 2

You are building a system to detect if computers in a data center are malfunctioning. You have 10,000 data points of computers functioning well, and 10,000 data points of computers malfunctioning. What type of algorithm should you use?

- Anomaly detection
- Supervised learning ANSWER

Correct

You have a sufficient number of anomalous examples to build a supervised learning model.

3. Question 3

Say you have 5,000 examples of normal airplane engines, and 15 examples of anomalous engines. How would you use the 15 examples of anomalous engines to evaluate your anomaly detection algorithm?

- You cannot evaluate an anomaly detection algorithm because it is an unsupervised learning algorithm.
- Use it during training by fitting one Gaussian model to the normal engines, and a different Gaussian model to the anomalous engines
- Put the data of anomalous engines (together with some normal engines) in the cross-validation and/or test sets to measure if the learned model can correctly detect anomalous engines. ANSWER
- Because you have data of both normal and anomalous engines, don't use anomaly detection. Use supervised learning instead.

Correct

Anomalous examples are used to evaluate rather than train the model.

4. Question 4

Anomaly detection flags a new input x as an anomaly if $p(x)<\epsilon$. If we reduce the value of ϵ, what happens?

- The algorithm is more likely to classify new examples as an anomaly.
- The algorithm is less likely to classify new examples as an anomaly. ANSWE
- The algorithm is more likely to classify some examples as an anomaly, and less likely to classify some examples as an anomaly. It depends on the example x. Not the ANSWER
- The algorithm will automatically choose parameters μ and σ to decrease $p(x)$ and compensate.

Incorrect

Actually, when ϵ is reduced, the probability of an event being classified as an anomaly is reduced.

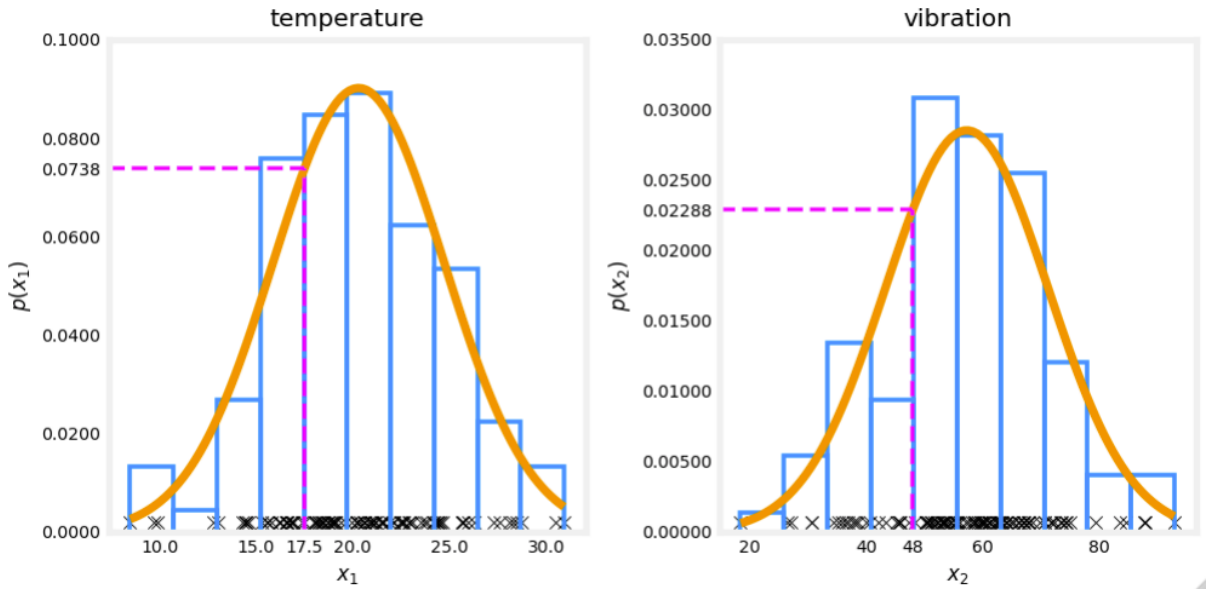
Correct

When ϵ is reduced, the probability of an event being classified as an anomaly is reduced.

5. Question 5

You are monitoring the temperature and vibration intensity on newly manufactured aircraft engines. You have measured 100 engines and fit the Gaussian model described in the video lectures to the data. The 100 examples and the resulting distributions are shown in the figure below.

The measurements on the latest engine you are testing have a temperature of 17.5 and a vibration intensity of 48. These are shown in magenta on the figure below. What is the probability of an engine having these two measurements?



- 17.5 * 48 = 840
- 17.5 + 48 = 65.5
- 0.0738 * 0.02288 = 0.00169 ANSWER
- 0.0738 + 0.02288 = 0.0966

Correct

According to the model described in lecture, p(A, B) = p(A) * p(B).