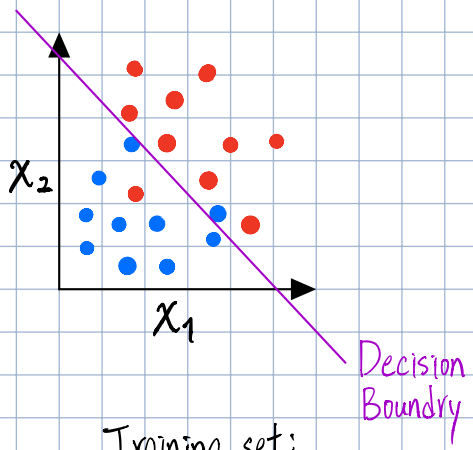
Supervised Learning



Training set:

$x^{(1)}, y^{(1)}$

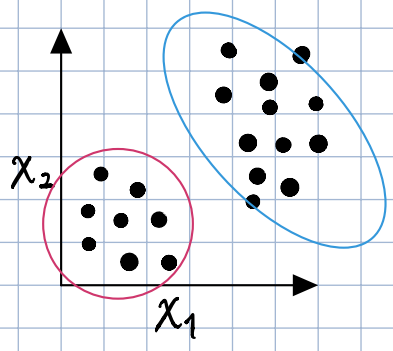
$x^{(2)}, y^{(2)}$

$\vdots$

$x^{(m)}, y^{(m)}$

input feature — target

Unsupervised Learning



Training set:

$x^{(1)}$

$x^{(2)}$

$\vdots$

$x^{(m)}$

There's no target y.

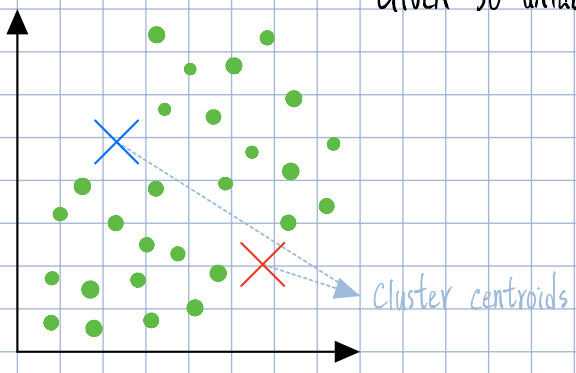
One algorithm for unsupervised learning is Clustering.

Here are some applications of clustering.

- Grouping similar news on Google News.
- Market Segmentation.
- DNA Analysis.
- Astronomical Data Analysis.

K-means Intuition

Given 30 unlabelled points $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots, x^{(30)}\}$



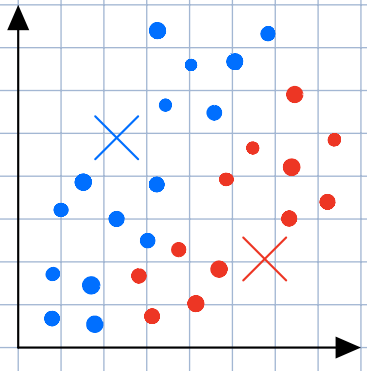
Initialization:

Determine the number of clusters. Assume we know it's 2 clusters.

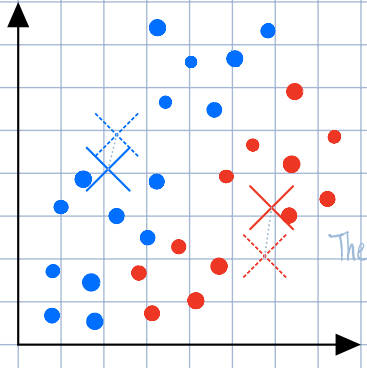
Randomly pick 2 points as the centers of 2 clusters.

Repeat:

Step 1: Assign each point $x^{(i)}$, $i = 1, \ldots, 30$ to its closet centroid.



Step 2: Recompute the centroids.



K-means Algorithm

Input:

K: No. of clusters

training set $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$

Initialization:

Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Assign each points to cluster centroids.

for $i = 1$ to m

$c^{(i)}$ = index (from 1 to K) of cluster centroid closest to $x^{(i)}$ $\underset{K}{\operatorname{argmin}} \| x^{(i)} - \mu_K \|_2^2$

Move cluster centroids.

for $k = 1$ to K

μ_k = average (mean) of points assigned to cluster k.

}

If a cluster has 0 training examples assigned to it, it should be eliminate or reinitialize K cluster centroids.

Now, the big question to ask is that will K-means ever converge. It may keep running forever.

K-means Optimization Objective:

$c^{(i)}$ = index of cluster $(1, 2, \ldots, K)$ to which example $x^{(i)}$ is currently assigned.

μ_k = cluster centroid $k \in \mathbb{R}^n$

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned.

E.g., if we want to know the cluster centroid to which $x^{(10)}$ has been assigned, we need to first find $c^{(10)}$ which tells us the index of the cluster to which $x^{(10)}$ is belonged to, and then go to $\mu_{c^{(10)}}$ to find the location (vector $\in \mathbb{R}^n$) of the cluster centroid to which $x^{(10)}$ has been assigned.

Cost function: $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$

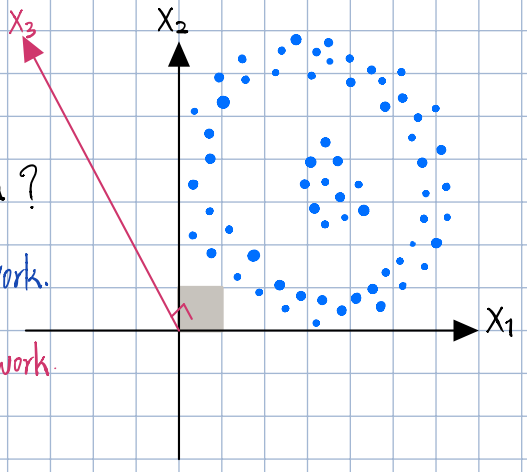
This function is called "Distortion" function in literature.

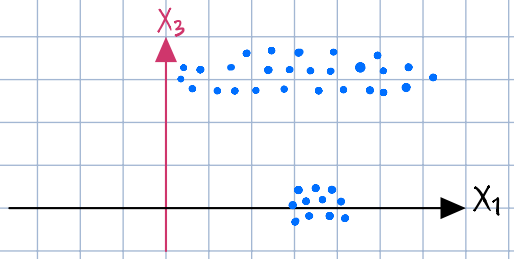
The goal: $\underset{\substack{c^{(1)}, \ldots, c^{(m)}, \\ \mu_1, \ldots, \mu_K}}{\operatorname{argmin}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

What if there's a pattern in the unlabelled data?

In the right example, conventional K-means doesn't work.

However, if adding one more dimension, K-means could work.





Initializing K-means :

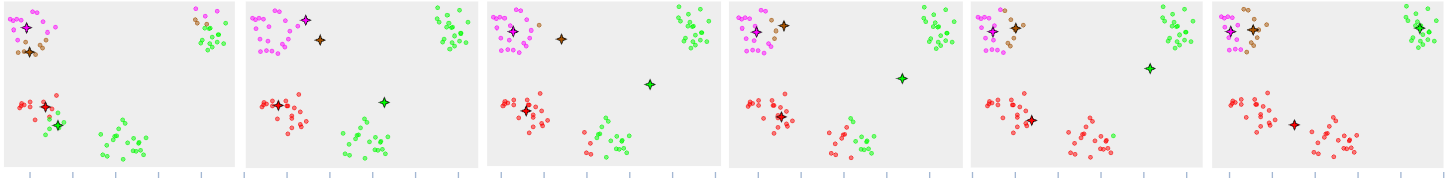
Random Initialization :

Choose $K < m$.

Randomly pick K training examples.

Set $\mu_1, \ldots, \mu_K$ equal to these K training examples.

One big problem of K-means is that the cost function J may reach local minima, which is shown in the image below. Thus, we should run random initialization multiple times.



for i = 1 to 100 { (50 ~ 100)

Randomly initialize K random examples.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

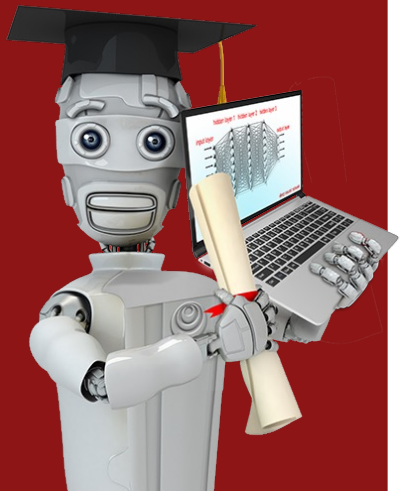
Compute cost function $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

}

Pick the set of clusters that gave lowest cost J.

DeepLearning.AI

Stanford
ONLINE

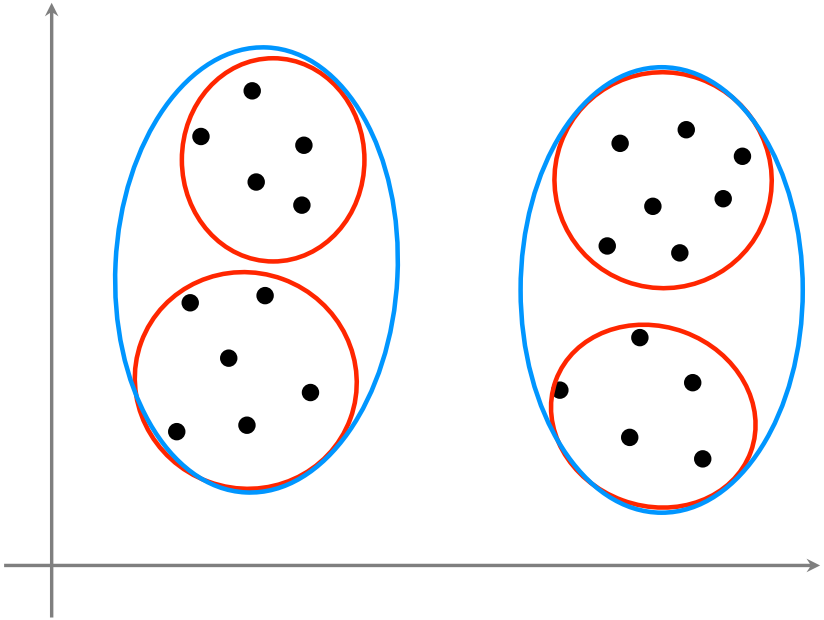


Clustering

Choosing the Number of Clusters

What is the right value of K?

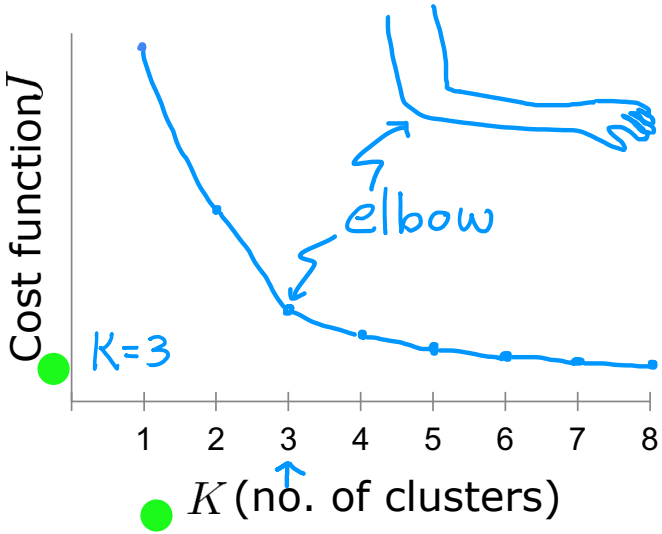
Could be 4 or 2.



Choosing the value of K

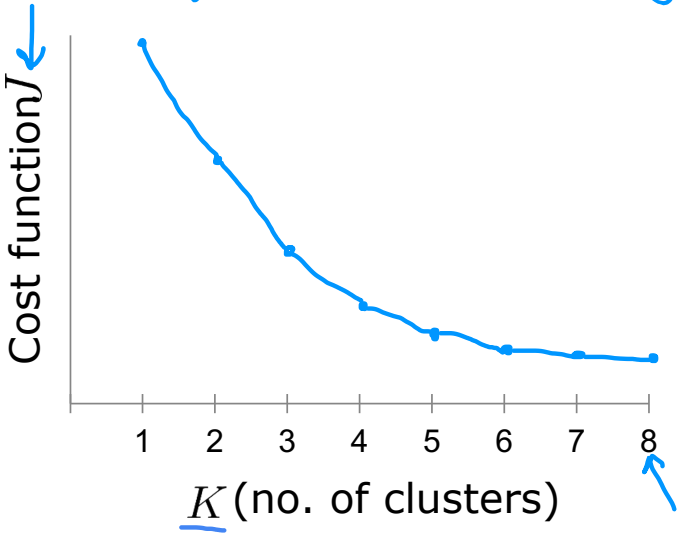
Andrew said he never uses the Elbow method.

Elbow method



the right "K" is often ambiguous

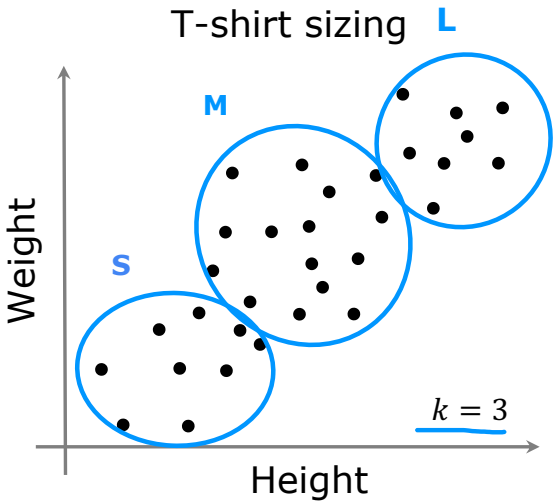
Don't choose K just to minimize cost J

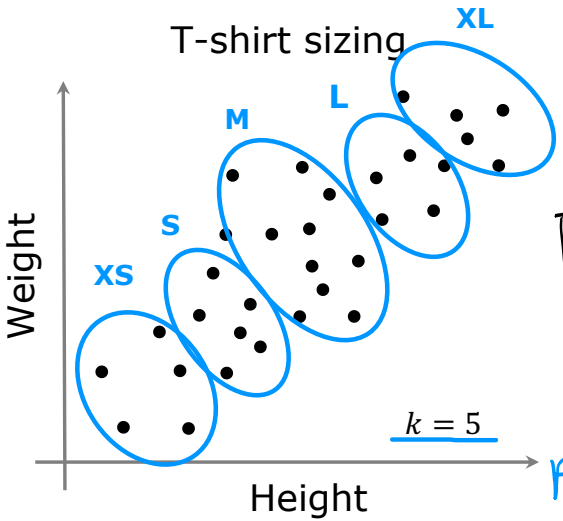


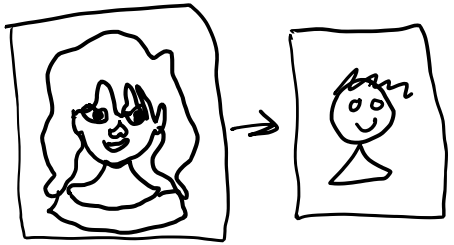
Choosing the value of K

Andrew suggests using this method.

Often, you want to get clusters for some later (downstream) purpose.
Evaluate K-means based on how well it performs on that later purpose.







practice lab:
image compression!

(abnormal) Anomaly Detection

Motivation: When manufacturing aircraft engines, we want to know if a newly manufatured engine is normal or abnormal?

Aircraft engine features:

x_1: heat generated

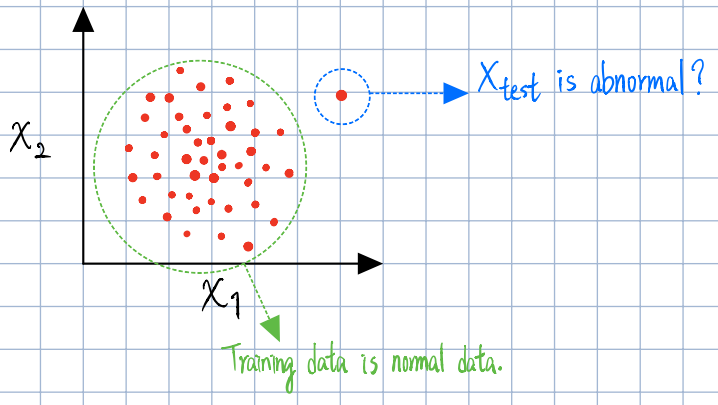
x_2: vibration intensity

$\vdots$

x_n

Training: Training Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ we know these are good engines.

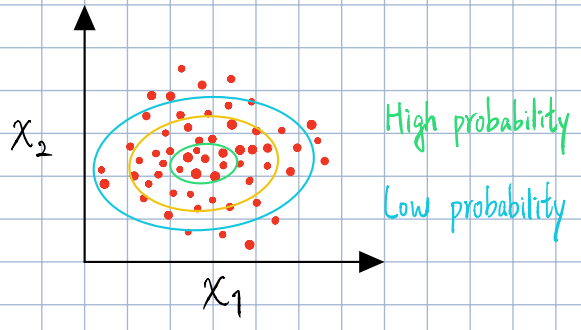
Inference: new engine x_{test}



Density Estimation

Given a Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

The first thing to do is build a model $p(x)$, the probability of x being seen in the dataset



Given x_{test}, calculate $p(x_{test})$.

If $p(x_{test}) < \varepsilon$, x_{test} is an anomaly.

If $p(x_{test}) \geq \varepsilon$, x_{test} is normal.

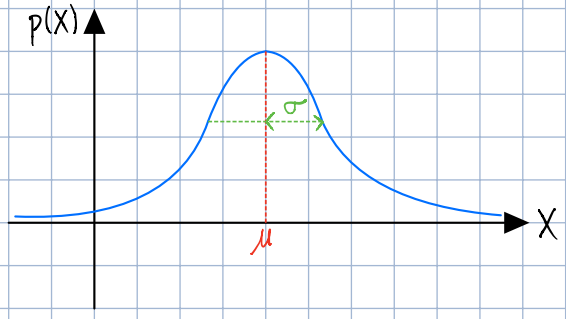
However, if $p(x_{test}) < \varepsilon$ happens frequently, which is not actually so, trying decreasing ε.

Gaussian (Normal) Distribution

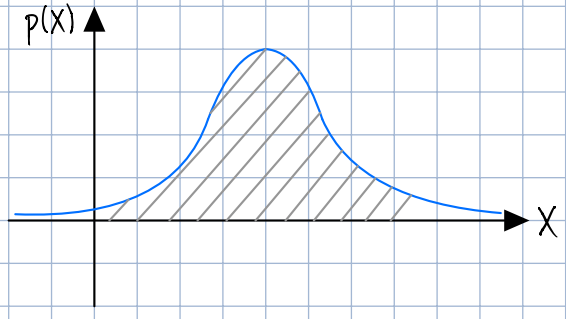
Say X is a number (random variable).

Probability of X is determined by a Gaussian with mean μ and variance σ^2.

standard deviation σ



The probability $p(X; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(X-\mu)^2}{2\sigma^2}}$



The sum of $p(X)$ must be 1. Thus, the area underneath the curve is always 1.

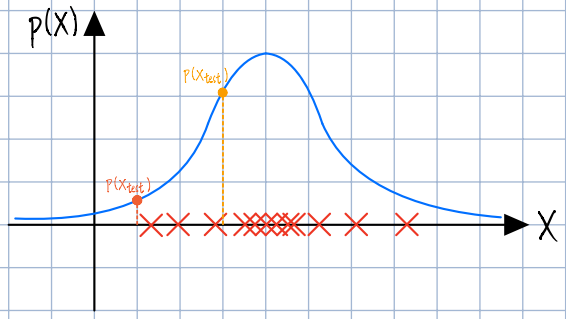
Parameter Estimation:

Given a Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$. Calculate mean μ, variance σ^2

$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$, $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2$ maximum likelihood for μ, σ^2

↳ or $m-1$. Andrew always uses m though.

Given x_{test}, calculate $p(x_{test})$



In this naive example, X is just a number (1 feature).

In practice, there are multiple features.

Anomaly Detection Algorithm

Step 1:

Given a training dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$.

Each example $x^{(i)}$ has n features. $x^{(i)} \in \mathbb{R}^n$ $\quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

Step 2:

Fit parameters $\mu_1, \dots, \mu_n, \sigma_1, \dots, \sigma_n$

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

Use vectorization.

Step 3:

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

$$p(x) = p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

If $p(x) < \varepsilon$, x is an anomaly.

Multivariate Gaussian Distribution

Given training set $x^{(1)}, x^{(2)}, \dots, x^{(m)}$, $x^{(i)} \in \mathbb{R}^n$

Don't model $p(x_1; \mu_1, \sigma_1^2)$, $p(x_2; \mu_2, \sigma_2^2)$, ... separately.

Model $p(x)$ all in one go.

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

Compute "covariance matrix". $\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$. $\Sigma \in \mathbb{R}^{n \times n}$

Notice $\Sigma \neq \sigma^2$, where $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - \mu|^2$

E.g.: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$

$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

It decides correlation of x_1 and x_2.

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

If $p(x; \mu, \Sigma) < \varepsilon$, x is an anomaly.

Notice that when $\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \ddots \\ & & & \sigma_n^2 \end{bmatrix}$, it's a special case

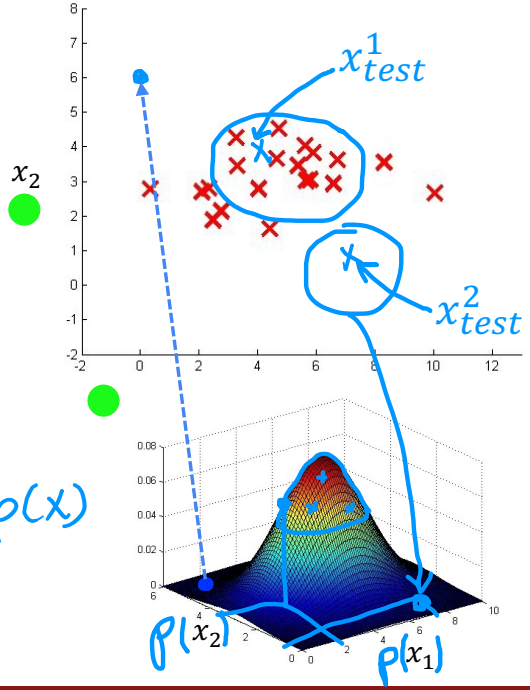
such that $p(x) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$

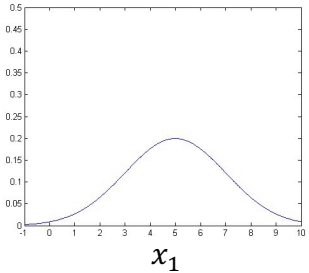
When Σ^{-1} does not exist (non-invertible), there are two possible reasons.

Possibility 1: number of training examples is less than number of features. $m < n$

Possibility 2: features are correlated. E.g., $x_1 = x_2 + x_3$

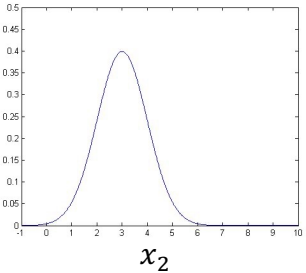
Anomaly detection example





$\mu_1 = 5,\ \sigma_1 = 2$

$p(x_1; \mu_1, \sigma_1^2)$



$\mu_2 = 3,\ \sigma_2 = 1$

$p(x_2; \mu_2, \sigma_2^2)$

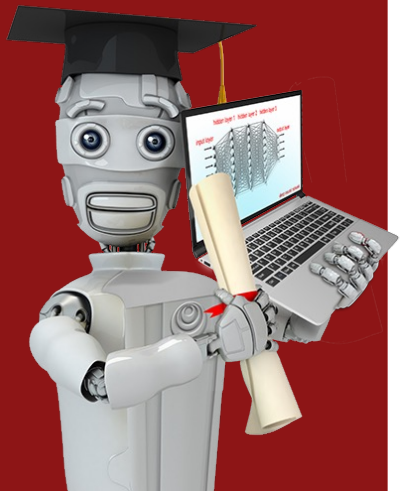
$\varepsilon = 0.02$

$p\left(x^{(1)}_{test}\right) = 0.0426 \longrightarrow$ "ok"

$p\left(x^{(2)}_{test}\right) = 0.0021 \longrightarrow$ anomaly

DeepLearning.AI

Stanford
ONLINE



Anomaly Detection

Developing and evaluating an anomaly detection system

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.),
making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples.

$y=1$ $y=0$

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

$y=0$ for all training examples (In practice, it should be fine if a few anomalous examples slip into the training set.)

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

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

include a few anomalous examples $y=1$

mostly normal examples $y=0$

Due to the fact that labelled data is used during development process (cross-validation), anomaly detection is not supervised learning. 我說的,不服來辯

Aircraft engines monitoring example

10000 good (normal) engines $y=0$

~~20~~ 2 flawed engines (anomalous) $y=1$ 2 to 50

Training set: 6000 good engines — train algorithm on training set

CV: 2000 good engines ($y = 0$) 10 anomalous ($y = 1$)

use cross validation set — tune ε tune x_j (which features to use)

Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Alternative: No test set — Use if very few labeled anomalous examples

higher risk of overfitting

Training set: 6000 good engines 2

CV: 4000 good engines ($y = 0$) ~~20~~ anomalous ($y = 1$)

tune ε tune x_j

Algorithm evaluation

course 2 week 3

skewed datasets

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

On a cross validation/test example x, predict

$$y = \begin{cases} 1 & if\ p(x) < \varepsilon\ (anomaly) \\ 0 & if\ p(x) \geq \varepsilon\ (normal) \end{cases}$$

10

2000

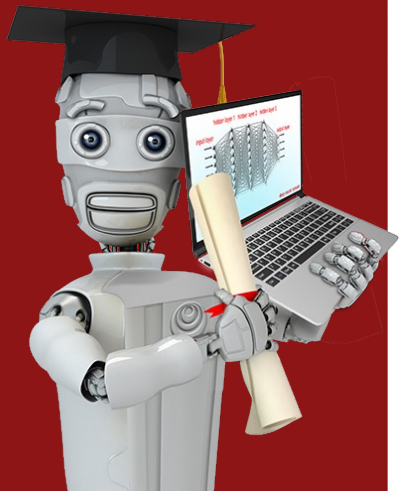
Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1-score

Can also use cross validation set to choose parameter ε

DeepLearning.AI

Stanford
ONLINE



說這麼多，就用 supervised learning (neural network) 就好拉。何必用 Anomaly Detection。

Anomaly Detection

Anomaly detection vs. supervised learning

我看起來，anomaly detection 是在建立一個 Generative model。

然而，supervised learning 是在建立一個 Discriminative model。

(logistic regression
neural networks)

Anomaly detection vs. Supervised learning

Very small number (0 to 20) of positive examples $y = 1$,
large number of negative examples ($y = 0$);
Model $p(x)$ with just negative examples.
Use positive examples for cv and test sets

Many different "types" of anomalies.
Hard for any algorithm to learn (from just positive examples) what the anomalies look like.
Future anomalies may look nothing like any of the anomalous examples seen so far.

financial fraud

Large number of positive and negative examples.

≥ 20 positive examples

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.

email spam

Anomaly detection vs. Supervised learning

- Fraud detection

- Manufacturing:
Finding new previously unseen defects.
(e.g. aircraft engines)

- Monitoring machines in a data center
- Security applications
Hackers always use New ways to crack systems.

- Email spam classification

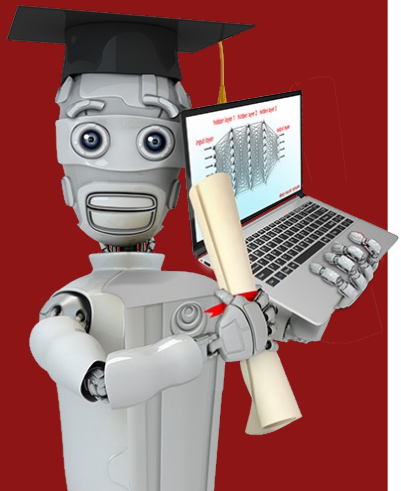
- Manufacturing:
Finding known, previously seen defects
(e.g. scratches on smartphones, $y = 1$)

- Weather prediction (sunny/rainy/etc.)

- Diseases classification

DeepLearning.AI

Stanford
ONLINE



In supervised learning, if we don't have the features quite right, or if we have a few extra features that are not relevant to the problem, that often turns out to be OK. Because the algorithm has the supervised signal that has enough labels y for the algorithm to figure out what features to ignore or how to rescale features and to take the best advantage of the features.

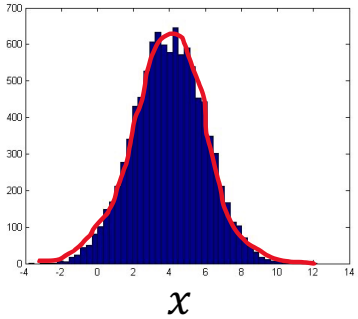
Anomaly Detection

Choosing what features to use

Non-gaussian features

It's crucial that features need to Gaussian more or less. Why? Because the model is Gaussian.

Gaussian



$p(x_1; \mu_1, \sigma_1^2)$

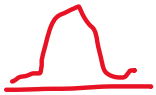
`plt.hist(x)`

$x_1 \leftarrow \log(x_1)$

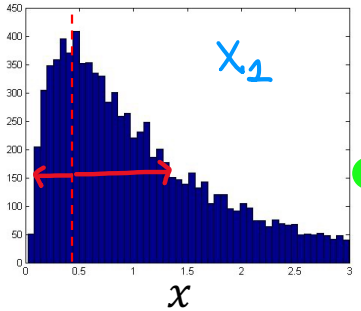
$x_2 \leftarrow \log(x_2 + 1)$ $\log(x_2 + c)$

$x_3 \leftarrow \sqrt{x_3} = x_3^{1/2}$

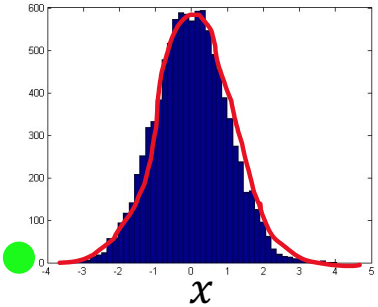
$x_4 \leftarrow x_4^{1/3}$



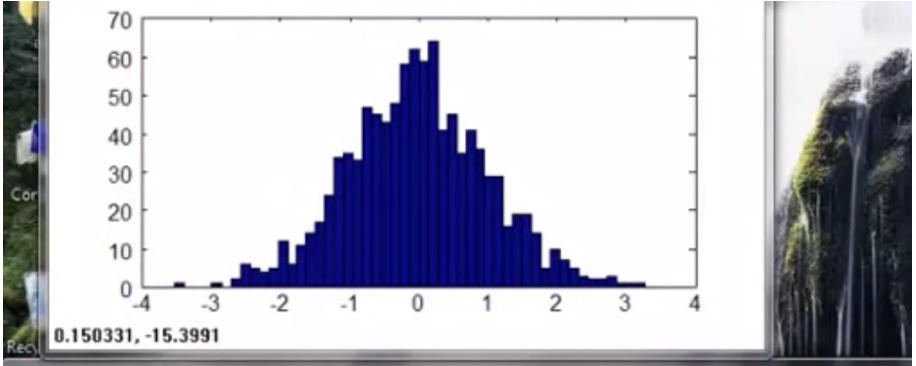
non-Gaussian



`np.log(x)`



more Gaussian



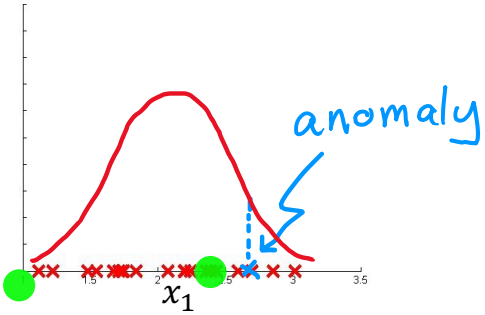
Octave-3.2.4

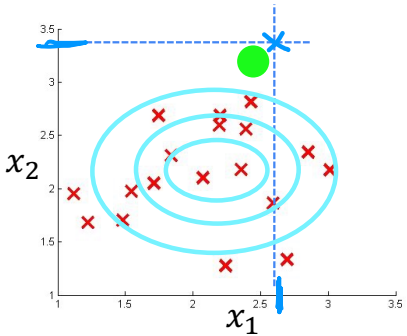
```
   1000      1

octave-3.2.4.exe:6> hist(x)
octave-3.2.4.exe:7> hist(x,50)
octave-3.2.4.exe:8> hist(x.^0.5, 50)
octave-3.2.4.exe:9> hist(x.^0.2, 50)
octave-3.2.4.exe:10> hist(x.^0.1, 50)
octave-3.2.4.exe:11> hist(x.^0.05, 50)
octave-3.2.4.exe:12> xNew = x.^0.05;
octave-3.2.4.exe:13> hist(log(x),50)
octave-3.2.4.exe:14> xNew = log(x);
```

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) Solution: adding new features.





x_1 num transactions x_2 typing speed

Monitoring computers in a data center

Choose features that might take on unusually large or small values in the event of an anomaly.

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

high
low

not unusual

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

E.g., in a data center that streams videos, then it's usual for computers to have high CPU load and high network traffic, or low CPU load and low network traffic. But, it's unusual to have high CPU load and low network traffic. In this case, I should create a new feature.

Deciding feature choice based on $p(x)$
Large for normal examples;
Becomes small for anomaly in the cross validation set