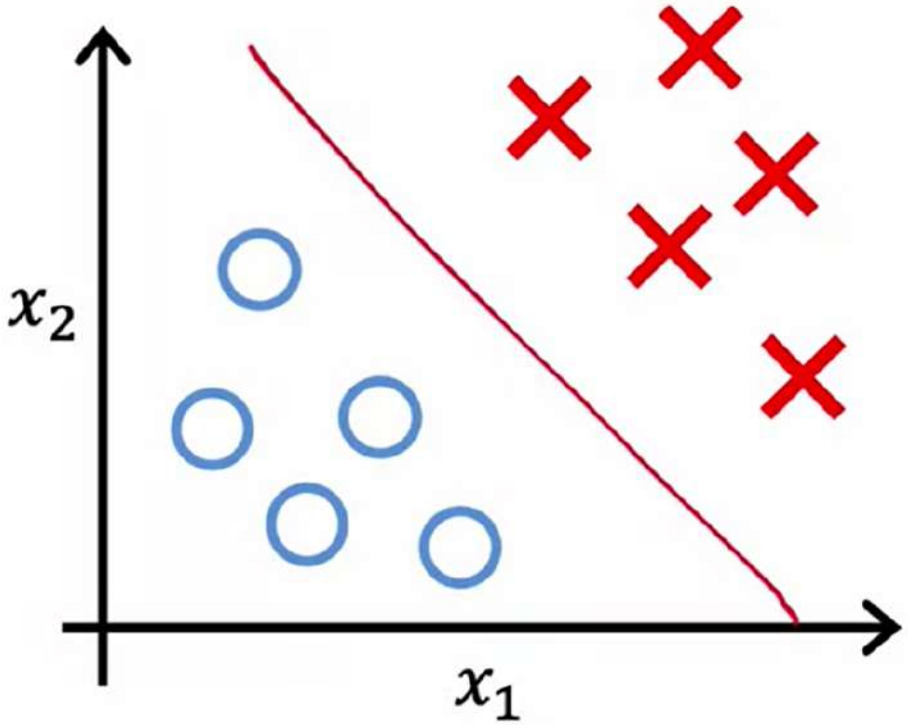
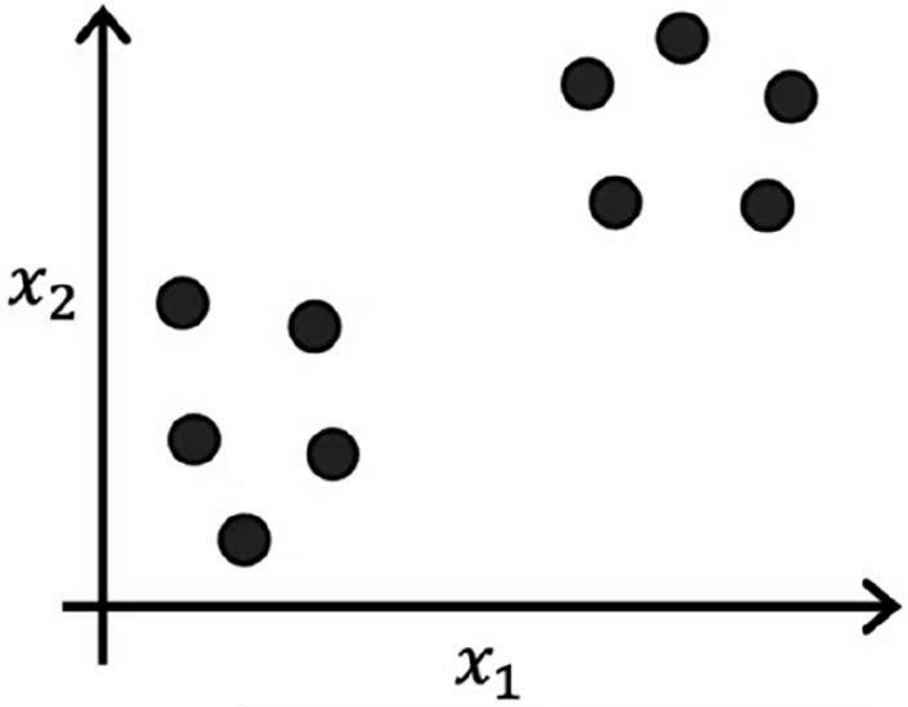
Supervised learning



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$?

In contrast, in unsupervised learning,

Unsupervised learning



Clustering

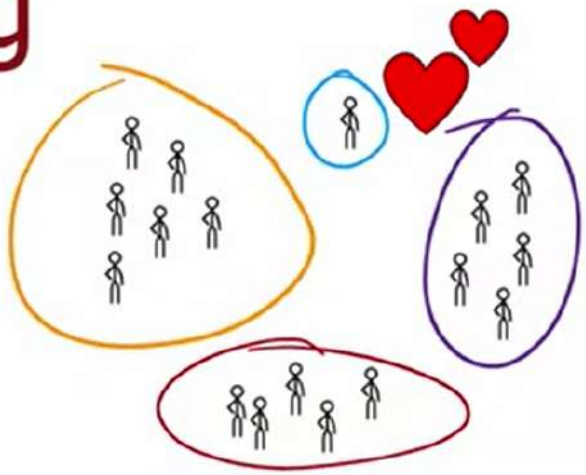
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

A clustering algorithm, in this case,

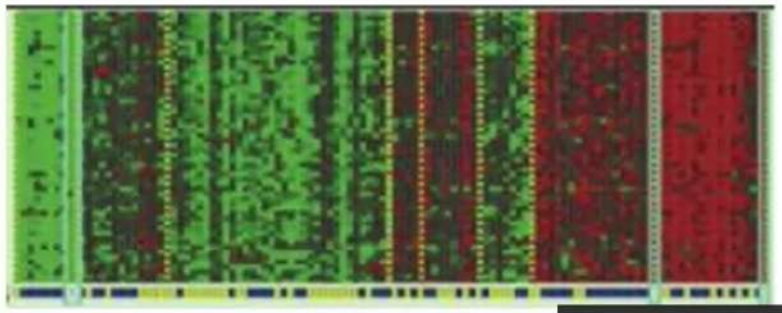
Applications of clustering

Grouping similar news

- Growing skills
- Develop career
- Stay updated with AI, understand how it affects your field of work



Market segmentation



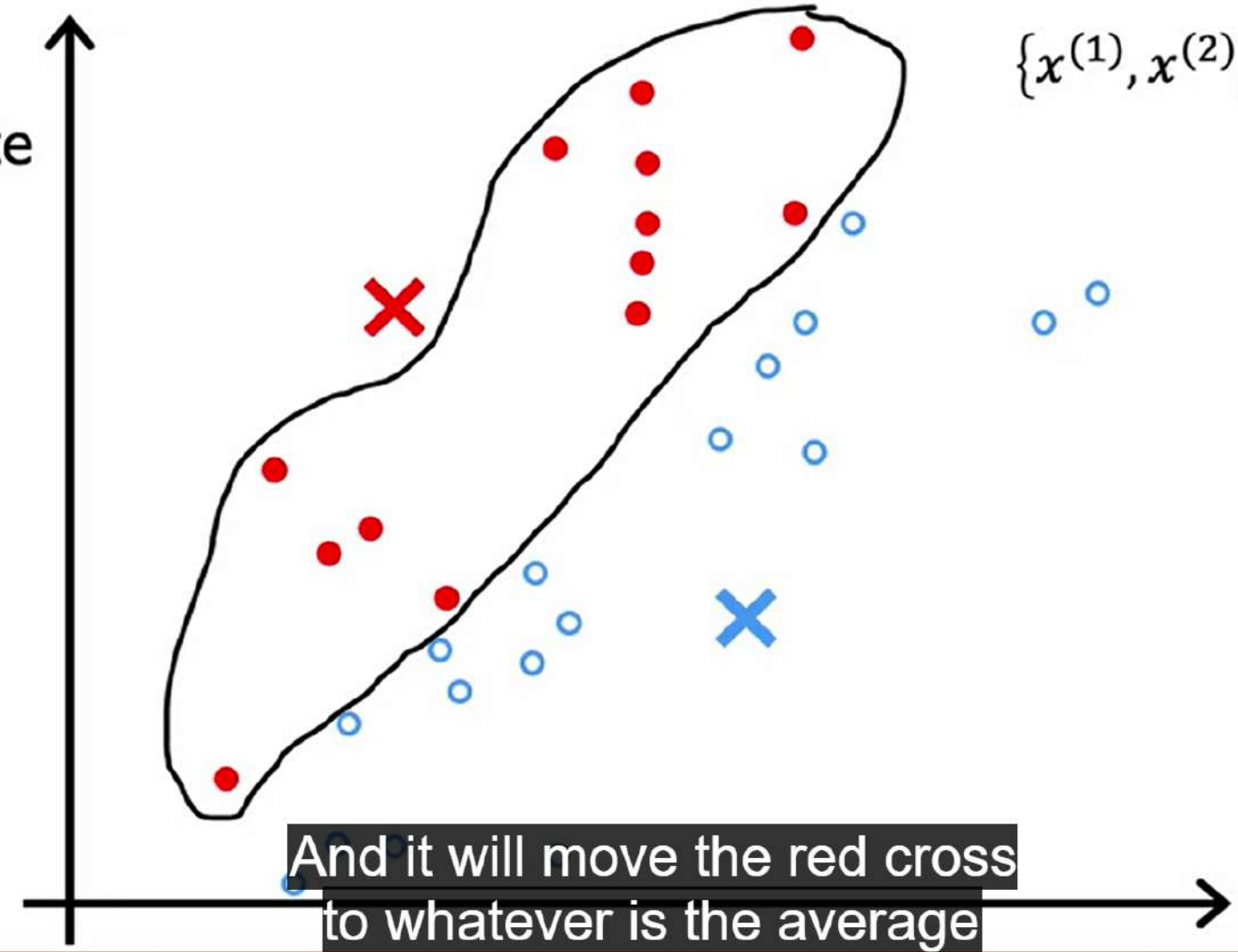
DNA analysis

Image credit: NASA/JPL-Calte
ch/E. Churchwell (Univ. of Wisconsin, Madison)

Astronomical data analysis

One of the applications I found

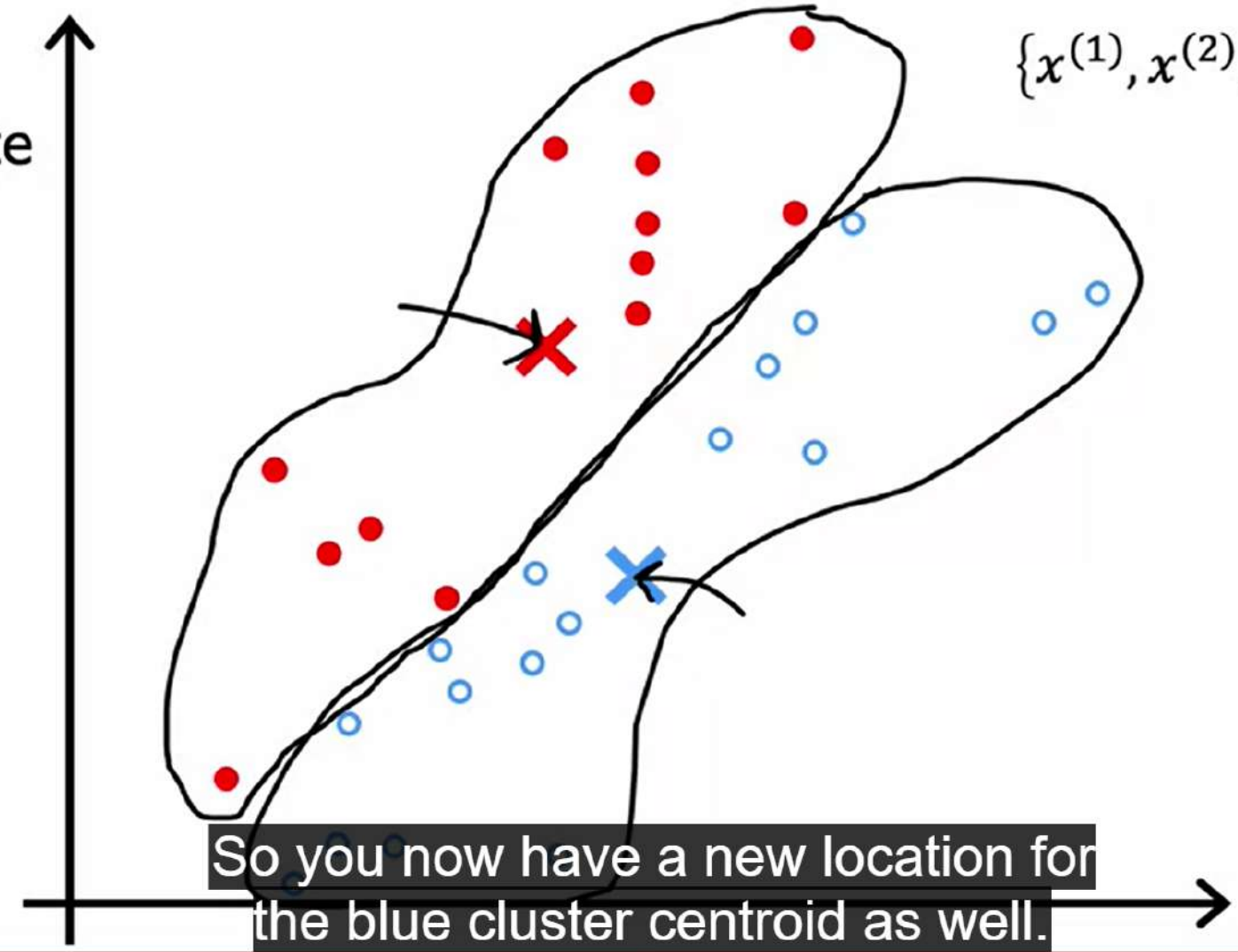
Step 2:
Recompute the centroids



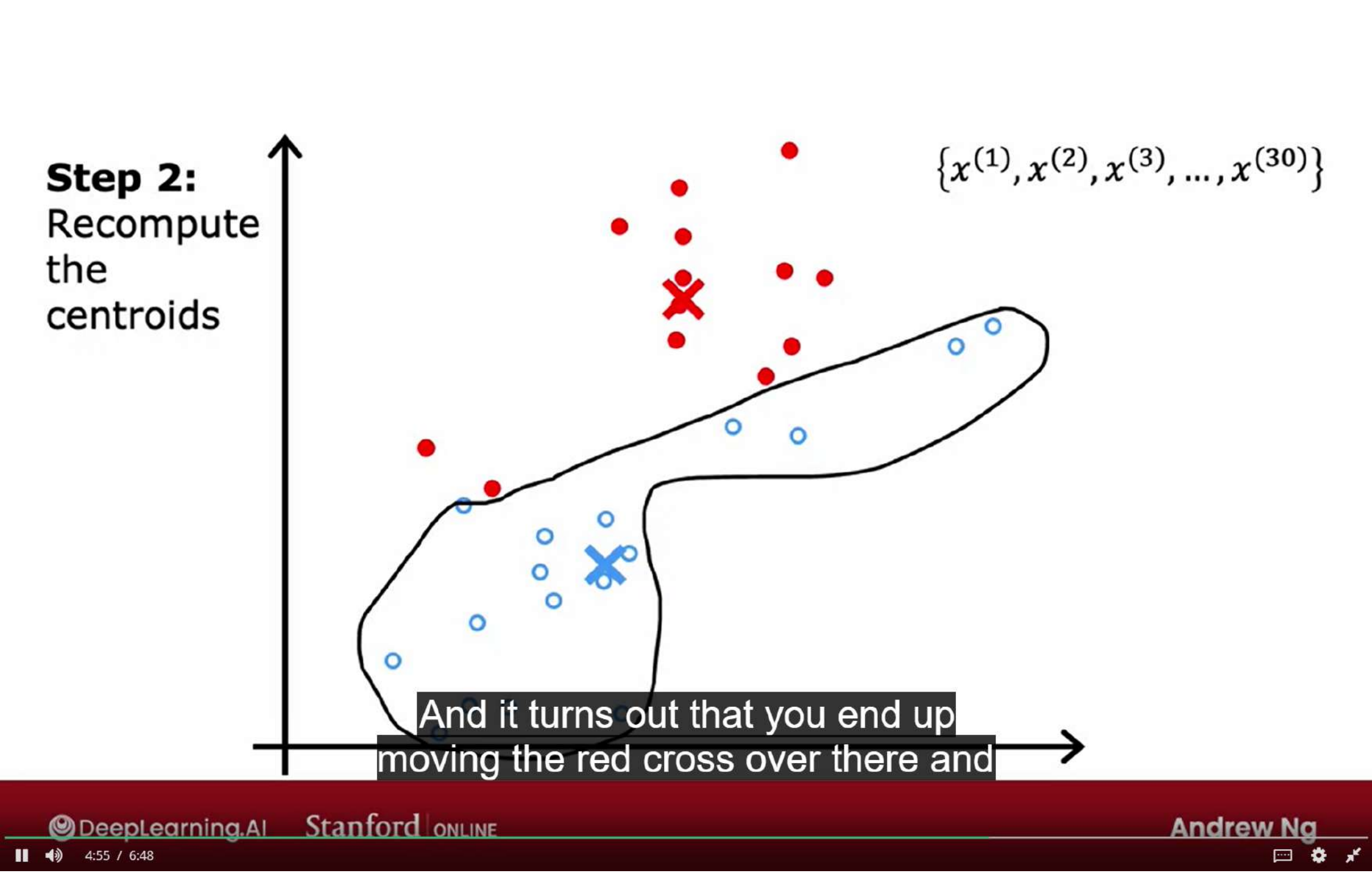
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$

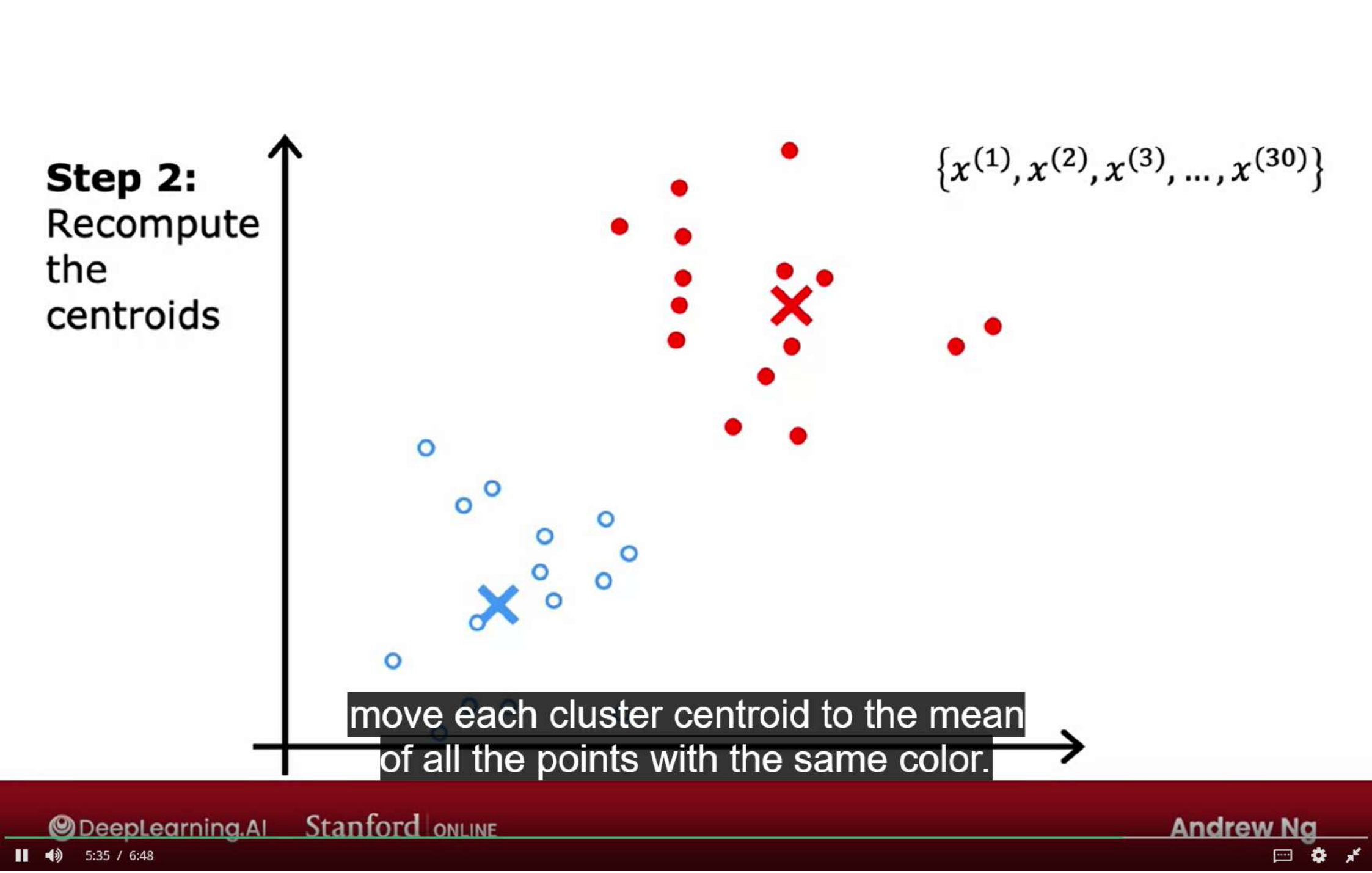
And it will move the red cross to whatever is the average

Step 2:
Recompute the centroids



So you now have a new location for the blue cluster centroid as well.


Step 2:
Recompute
the
centroids
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$
And it turns out that you end up
moving the red cross over there and
DeepLearning.AI
Stanford ONLINE
Andrew Ng
4:55 / 6:48


Step 2:
Recompute
the
centroids
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$
move each cluster centroid to the mean
of all the points with the same color.
DeepLearning.AI
Stanford ONLINE
Andrew Ng
5:35 / 6:48

K-means algorithm

μ_1, μ_2 $x^{(1)}, x^{(2)}, \ldots x^{(30)}$

$n = 2$ $K = 2$

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K$

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

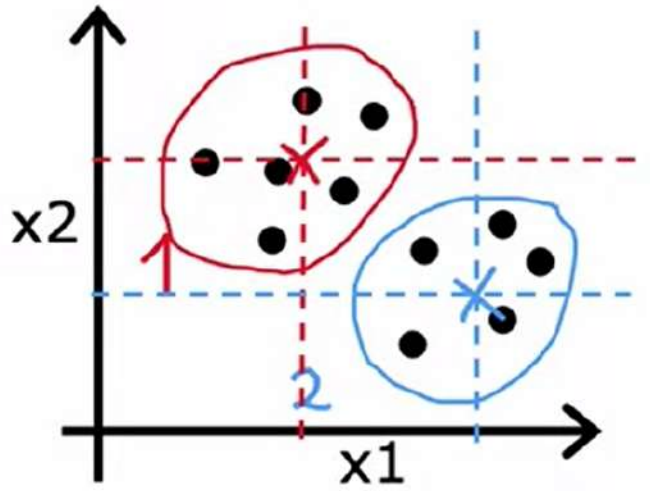
$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

for k = 1 to K

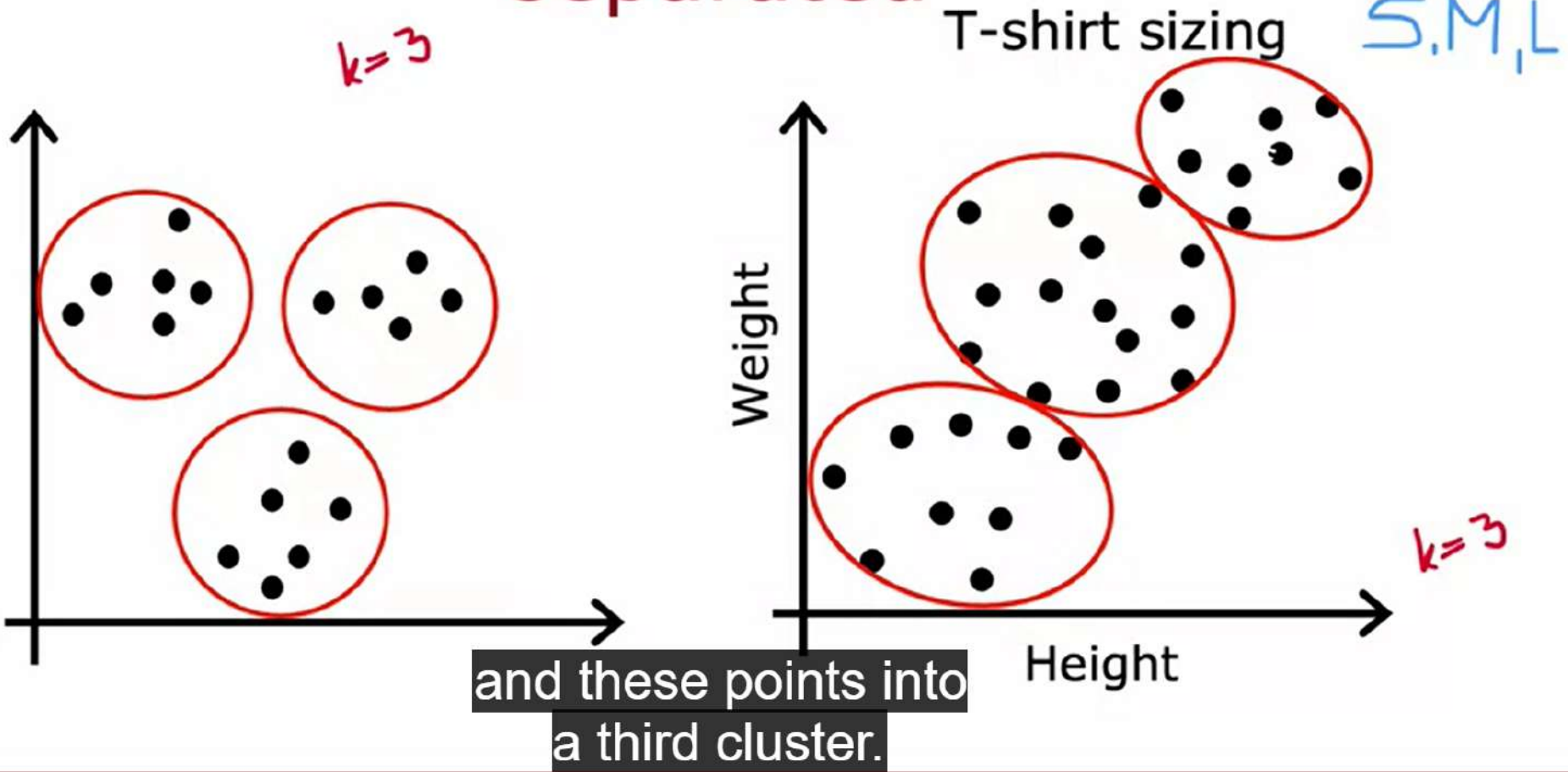
μ_k := average (mean) of points assigned to cluster k

}



$$\mu_1 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right]$$

K-means for clusters that are not well separated



and these points into a third cluster.

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

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

$x^{(10)}$ $c^{(10)}$ $\mu_{c^{(10)}}$

Cost function

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

$$\min_{\substack{c^{(1)}, \ldots, c^{(m)} \\ \mu_1, \ldots, \mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, .., \mu_K)$$

These are the locations of all the clusters centroid is defined as

Cost function for K-means

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

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

$c^{(i)}$:= index of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

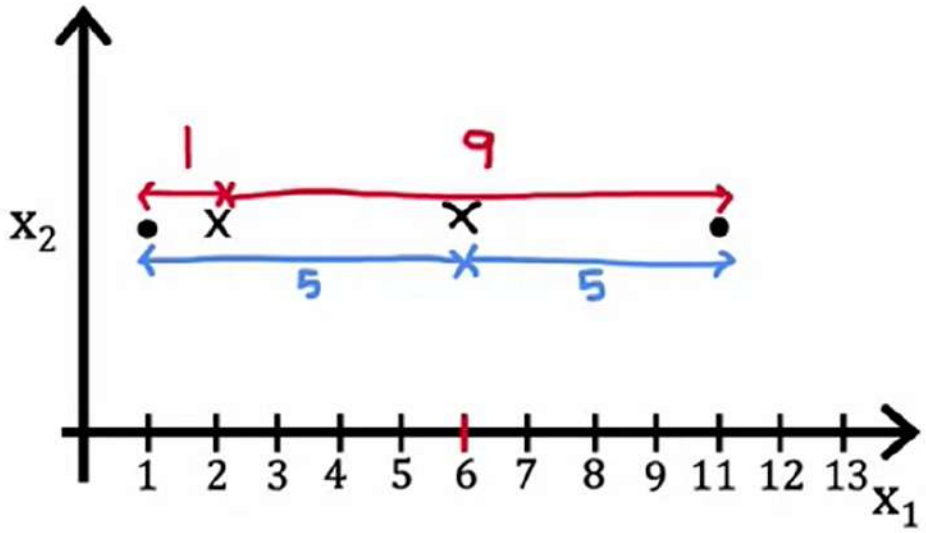
for k = 1 to K

μ_k := average of points in cluster k

}

the distortion as much as possible.

Moving the centroid

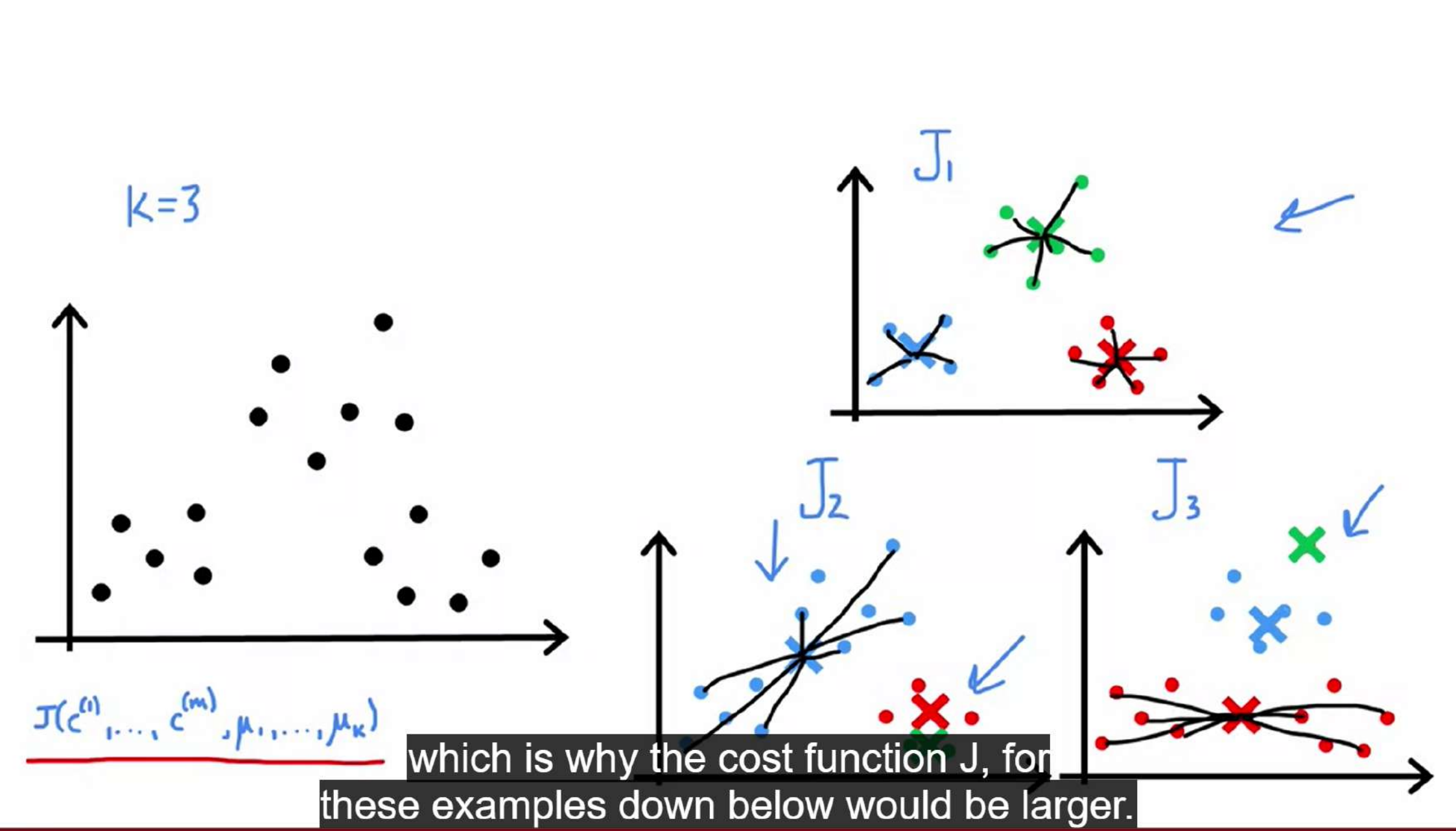


$\frac{1}{2}(1^2 + 9^2) = \frac{1}{2}(1+81) = 41$

$\frac{1}{2}(1 + 11) = 6$

$\frac{1}{2}(5^2 + 5^2) = 25$

This average location in the middle
of these two training examples,



which is why the cost function J, for these examples down below would be larger.

Random initialization

For $i = 1$ to 100 {

Randomly initialize K-means. ← k random examples

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

Computer cost function (distortion)

$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_1, \ldots, \mu_k)$ ←

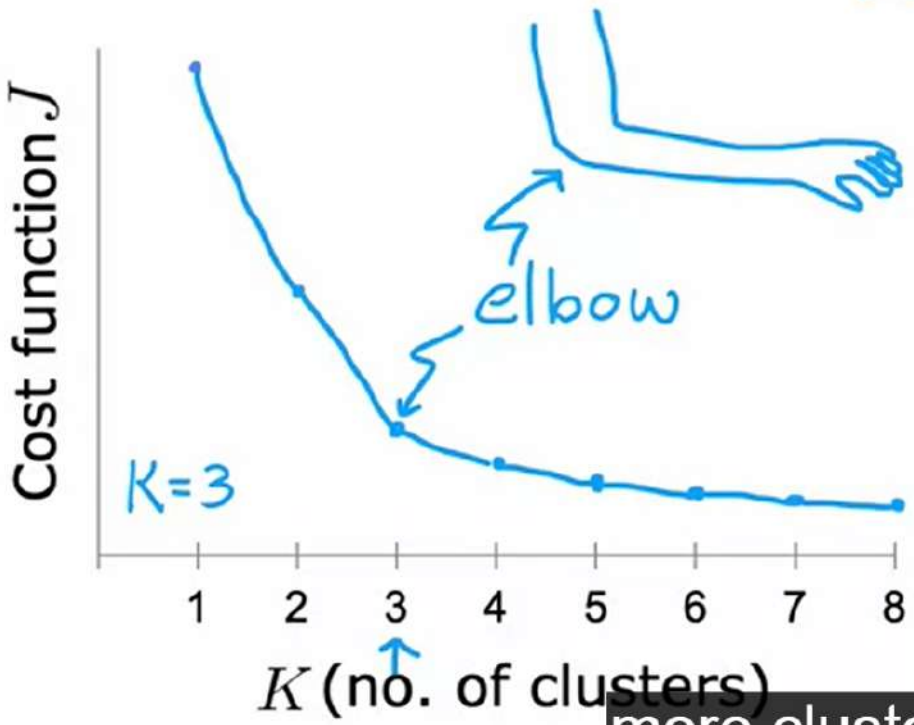
}

Pick set of clusters that gave lowest cost J

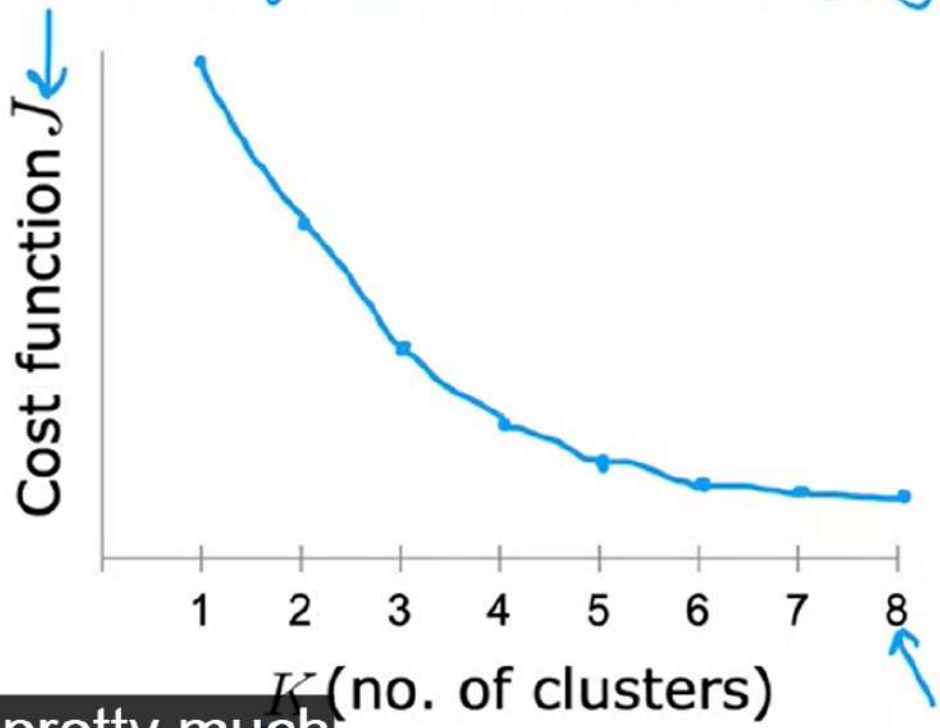
I plugged in the number up here as 100.

Choosing the value of K

Elbow method



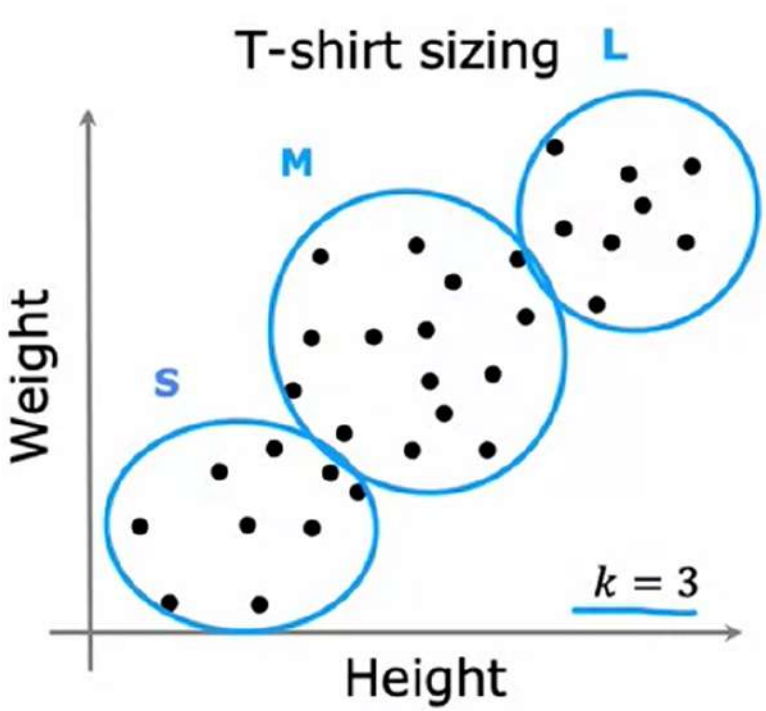
the right "K" is often ambiguous
Don't choose K just to minimize cost J

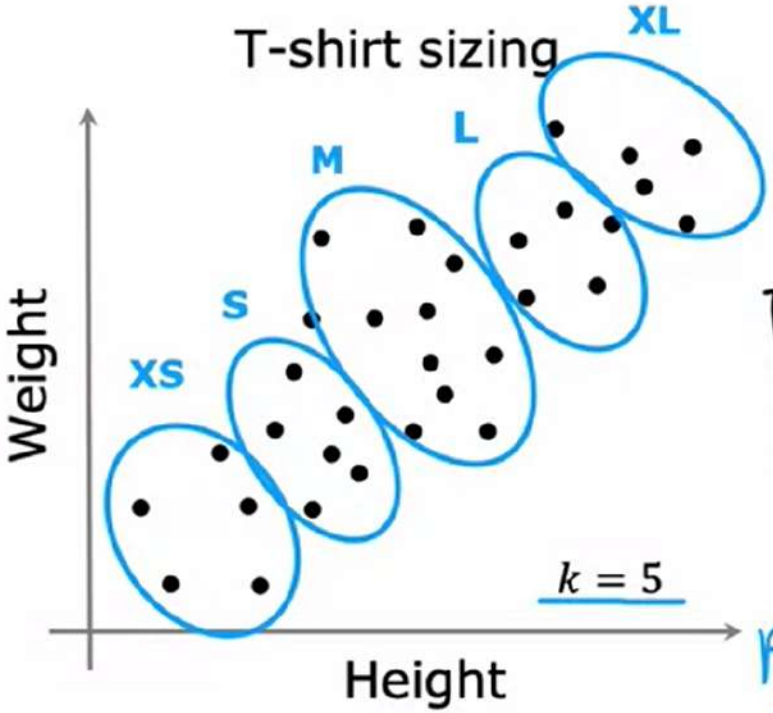


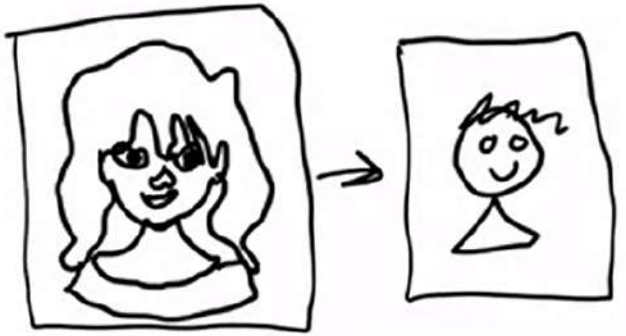
more clusters will pretty much

Choosing the value of K

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!

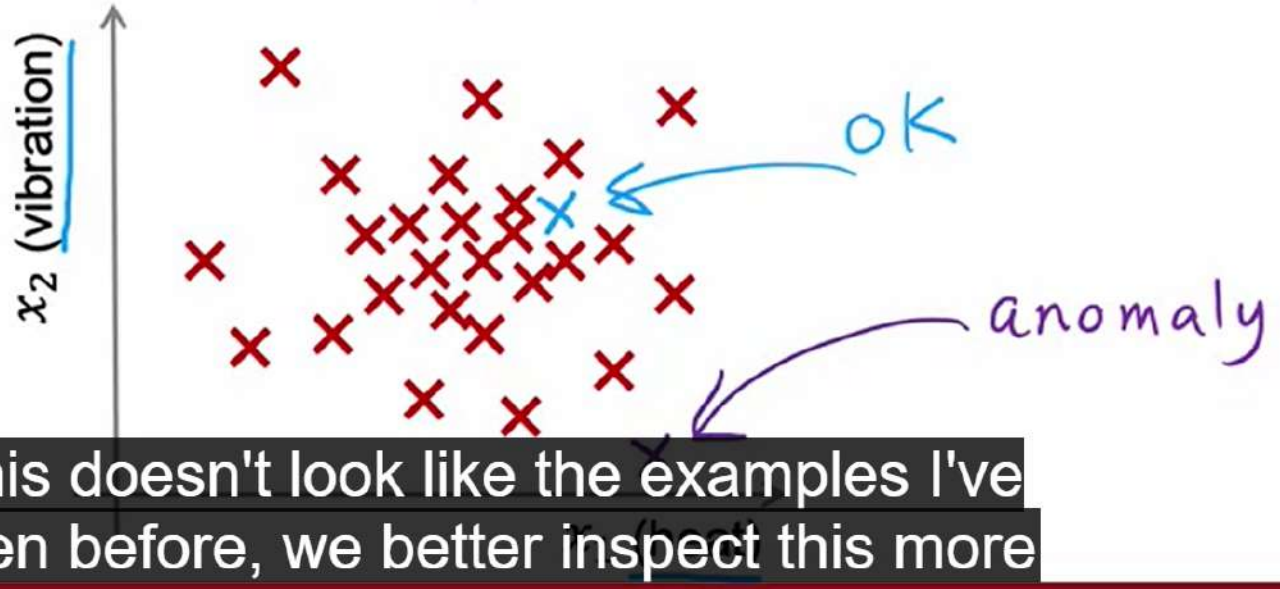
look versus how large

Anomaly detection example

Aircraft engine features:
x_1 = heat generated
x_2 = vibration intensity
...

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

New engine: x_{test}



This doesn't look like the examples I've seen before, we better inspect this more

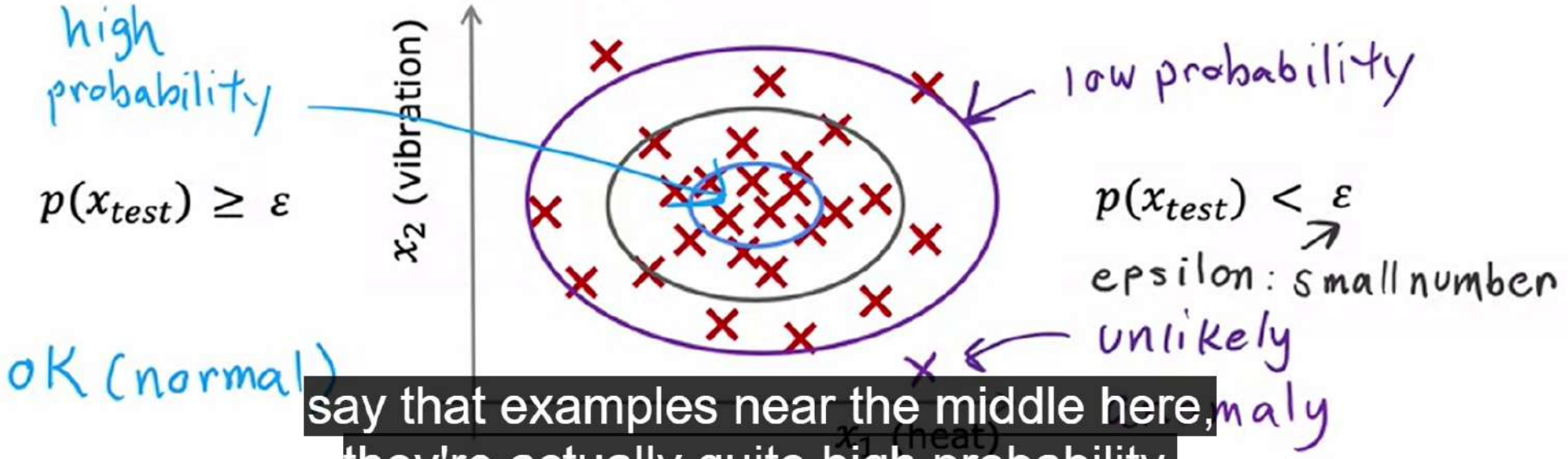
Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

probability of x being seen in dataset

Model $p(x)$

Is x_{test} anomalous?



say that examples near the middle here, they're actually quite high probability.

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

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.

it might be worth taking a look at that computer to see if something is wrong

Gaussian (Normal) distribution

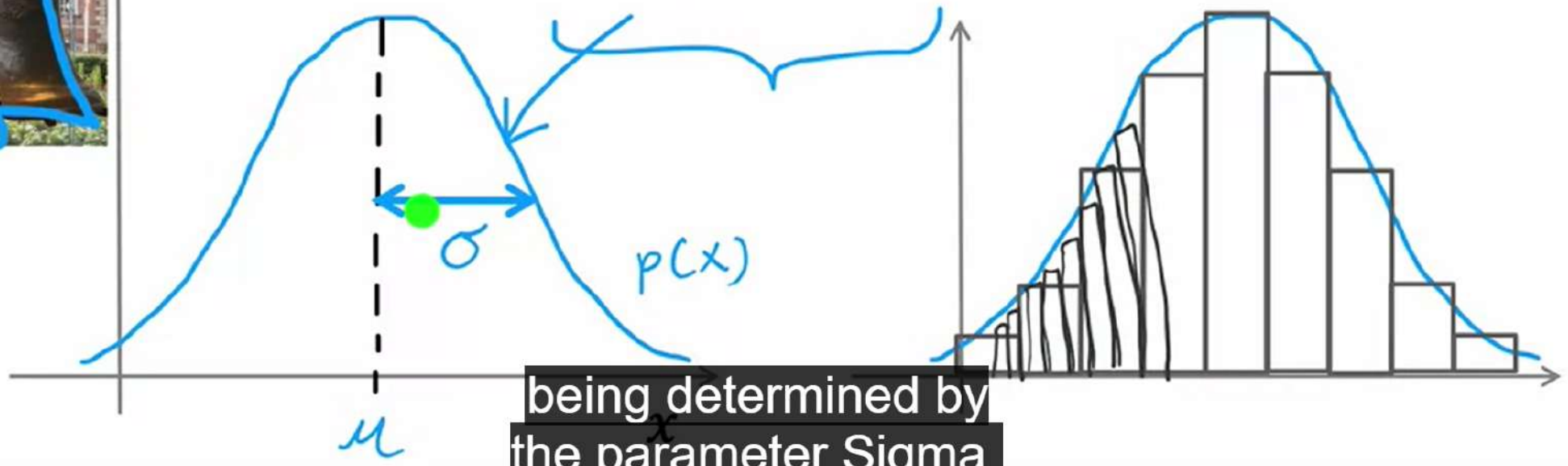
σ standard deviation

σ^2 variance

Say x is a number.
Probability of x is determined by a Gaussian with mean μ, variance σ^2.

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

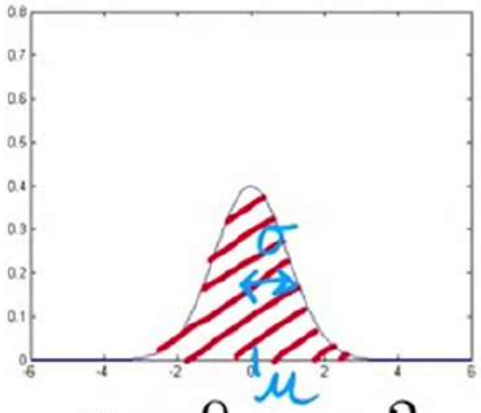
$\pi = 3.14$



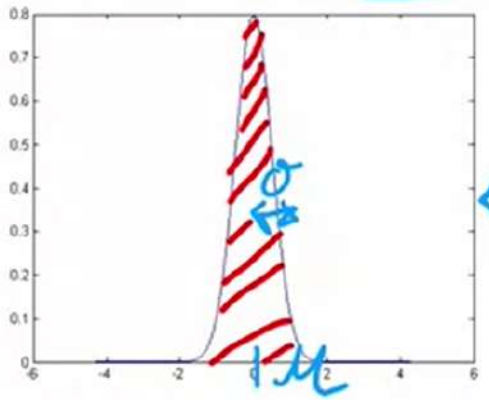
being determined by
the parameter Sigma.

Gaussian distribution example

$\mu = 0, \sigma = 1$

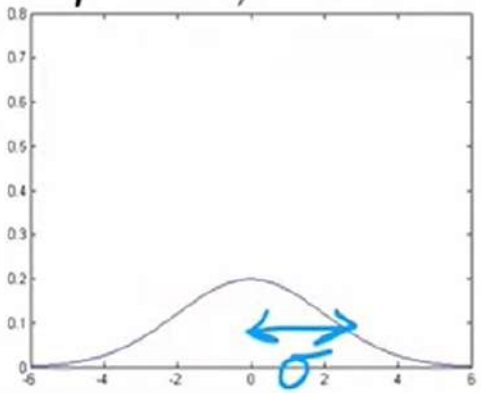


$\mu = 0, \sigma = 0.5$



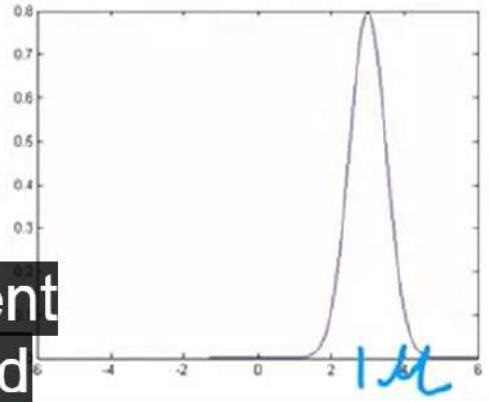
$\sigma^2 = 0.25$

$\mu = 0, \sigma = 2$



$\sigma^2 = 4$

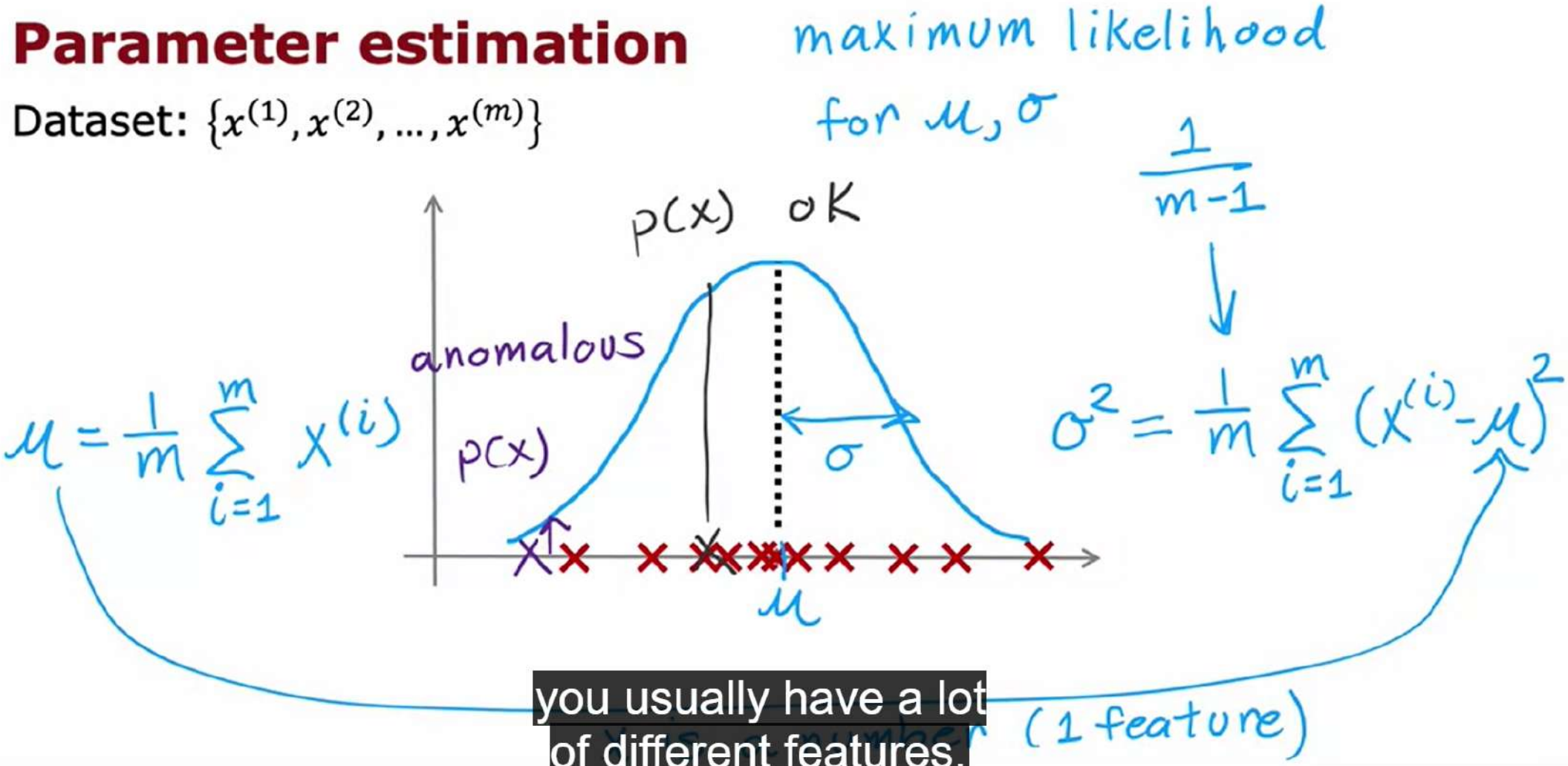
$\mu = 3, \sigma = 0.5$



This is how different choices of Mu and

Parameter estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

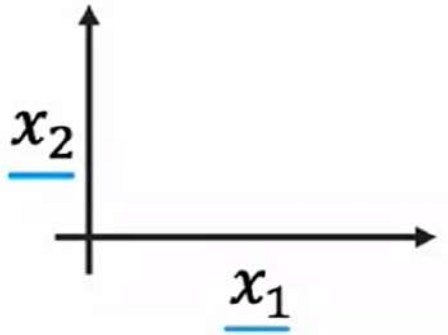


you usually have a lot of different features.

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)}\}$

Each example $\vec{x}^{(i)}$ has n features



$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

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

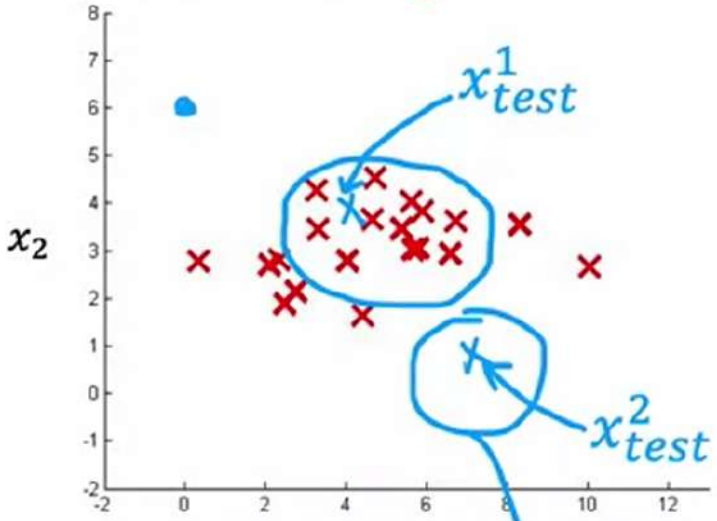
$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

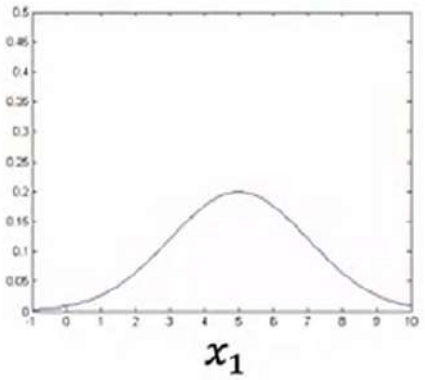
$\sum$ "add" $\prod$ "multiply"

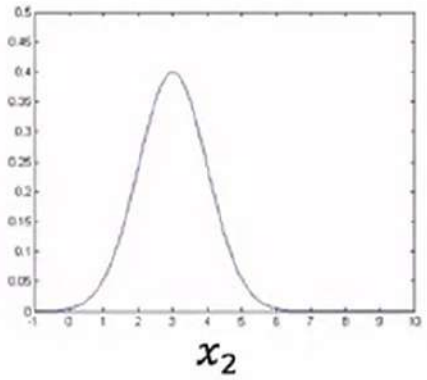
$$p(x_1 = \text{high temp}) = 1/10$$
$$p(x_2 = \text{high vibra}) = 1/20$$
$$p(x_1, x_2) = p(x_1) * p(x_2)$$
$$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$$

symbol here corresponds to multiplying these terms over here for j =1 through n.

Anomaly detection example





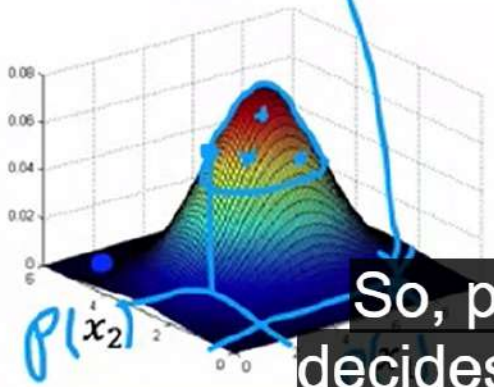


$\mu_1 = 5,\ \sigma_1 = 2$

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

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

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



$p(x)$

$\varepsilon = 0.02$

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

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

So, pretty much as you might hope it decides that x test 1 looks pretty normal.

Aircraft engines monitoring example

10000 good (normal) engines
20 flawed engines (anomalous)

2 to 50

2

$y=0$

$y=1$

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

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

Alternative: No test set

Training set: 6000 good engines 2

CV: 4000 good engines ($y = 0$) [illegible] $y = 1$)

tune ε tune x_j

put all of that in the cross validation set.

Anomaly detection vs. Supervised learning

Very small number of positive examples ($y = 1$). (0-20 is common).
Large number of negative ($y = 0$) examples.

$p(x)$

$y = 1$

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 we've seen so far.

Fraud

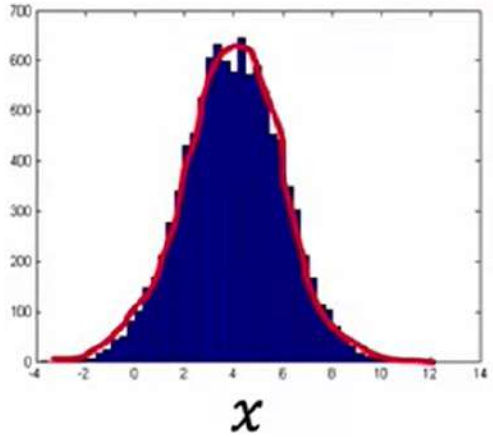
some individuals are trying to commit financial fraud.

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.

Non-gaussian features



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

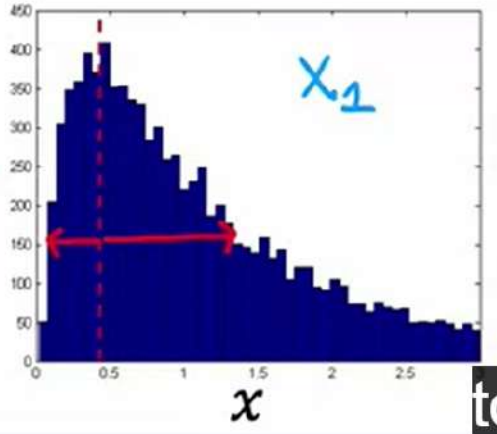
`plt.hist(x)`

$x_1 \leftarrow \log(x_1)$

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

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

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



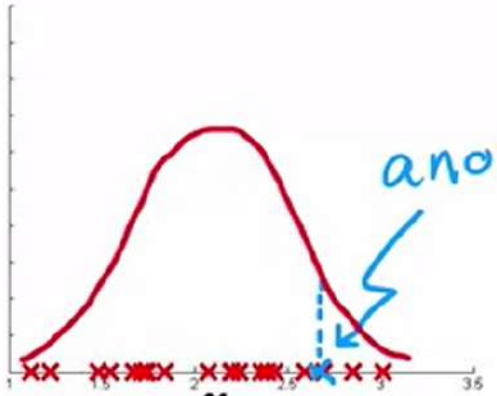
`np.log(x)`

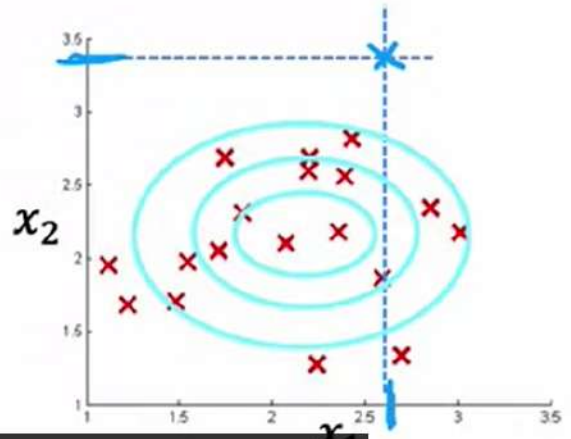
others, In order to try
to make it more Gaussian.

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)





x_1 num transactions x_2 typing speed

the creation of new features that would allow the algorithm to spot.

Predicting movie ratings

User rates movies using ~~one~~ zero to five stars

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

n_u = no. of users

n_m = no. of movies

$r(i,j)=1$ if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j)=1$)

$n_u = 4$

$n_m = 5$

$r(1,1) = 1$

$r(3,1) = 0$

$y^{(3,2)} = 4$

That's why we're going to define r(i,j)=1 if user j has rated movie i and

What if we have features of the movies?

$n_u = 4$
$n_m = 5$
$n = 2$

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$

$x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

For user 1: Predict rating for movie i as: $w^{(1)} \cdot x^{(i)} + b^{(1)}$ ← just linear regression

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$ $w^{(1)} \cdot x^{(3)} + b^{(1)} = 4.95$

For user j: Predict user j's rating for movie i as $w^{(j)} \cdot x^{(i)} + b^{(j)}$

movie i as w(j).X(i)+b(j).

Cost function

Notation:

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating given by user j on movie i (if defined)
- $w^{(j)}, b^{(j)}$ = parameters for user j
- $x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$

- $m^{(j)}$ = no. of movies rated by user j

To learn $w^{(j)}, b^{(j)}$

number of features

$$\min_{w^{(j)} b^{(j)}} J\left(w^{(j)}, b^{(j)}\right) = \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

you should end up with the same value of w and b.

Cost function

To learn parameters $w^{(j)}, b^{(j)}$ for user j :

$$J\left(w^{(j)}, b^{(j)}\right) = \frac{1}{2} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \ldots w^{(n_u)}, b^{(n_u)}$ for all users :

$$J\begin{pmatrix} w^{(1)}, \ldots, w^{(n_u)} \\ b^{(1)}, \ldots, b^{(n_u)} \end{pmatrix} = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

$f(x)$

where that plays a role similar to the output f(x) of linear regression.

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)	
Love at last	5	5	0	0	?	?	
Romance forever	5	?	?	0 ←	?	?	$x^{(2)}$
Cute puppies of love	?	4	0	? ←	?	?	$x^{(3)}$
Nonstop car chases	0	0	5	4 ←	?	?	
Swords vs. karate	0	0	5	? ←	?	?	

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$, $w^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$

$b^{(1)} = 0$, $b^{(2)} = 0$, $b^{(3)} = 0$, $b^{(4)} = 0$ ←

using $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$w^{(1)} \cdot x^{(1)} \approx 5$
$w^{(2)} \cdot x^{(1)} \approx 5$
$w^{(3)} \cdot x^{(1)} \approx 0$
$w^{(4)} \cdot x^{(1)} \approx 0$

→ $x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

That's what makes it possible to try to

Cost function

Given $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots, w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$:

$$J(x^{(i)}) = \frac{1}{2} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

→ To learn $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$:

$$J(x^{(1)}, x^{(2)}, \ldots, x^{(n_m)}) = \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

of x1 through x^n_m

Collaborative filtering

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)}, \cdots, w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)}, \cdots, x^{(n_m)}$:

$$\min_{x^{(1)}, \cdots, x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

where we do have a rating

Collaborative filtering

	$j=1$ Alice	$j=2$ Bob	$j=3$ Carol
$i=1$ Movie1	5	5	?
$i=2$ Movie2	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\cdots,w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)},\cdots,x^{(n_m)}$:

$$\min_{x^{(1)},\cdots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\min_{\substack{w^{(1)},\ldots,w^{(n_u)}\\ b^{(1)},\ldots,b^{(n_u)}\\ x^{(1)},\ldots,x^{(n_m)}}} J(w,b,x) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

How do you minimize this cost function

Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} J(w,b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$w_i^{(j)} = w_i^{(j)} - \alpha \frac{\partial}{\partial w_i^{(j)}} J(w,b,x)$$

$$b^{(j)} = b^{(j)} - \alpha \frac{\partial}{\partial b^{(j)}} J(w,b,x)$$

$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

}

parameters w, b, x x is also a parameter

w and b, as well as x.

Example applications

1. Did user j purchase an item after being shown? 1, 0, ?
2. Did user j fav/like an item? 1, 0, ?
3. Did user j spend at least 30sec with an item? 1, 0, ?
4. Did user j click on an item? 1, 0, ?

Meaning of ratings:
1 - engaged after being shown item
0 - did not engage after being shown item
? - item not yet shown

A labor of one means that the user engaged after being shown an item And

From regression to binary classification

- Previously:
  - Predict $y^{(i,j)}$ as $w^{(j)} \cdot x^{(i)} + b^{(j)}$
- For binary labels:
  - Predict that the probability of $y^{(i,j)} = 1$ is given by $g\left(w^{(j)} \cdot x^{(i)} + b^{(j)}\right)$

where $g(z) = \frac{1}{1+e^{-z}}$

turn it into something that would be a lot like a logistic regression

Cost function for binary application

Previous cost function:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\underbrace{w^{(j)}\cdot x^{(i)}+b^{(j)}}_{f(x)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Loss for binary labels $y^{(i,j)}$: $f_{(w,b,x)}(x)=g(w^{(j)}\cdot x^{(i)}+b^{(j)})$

$$L\left(f_{(w,b,x)}(x),y^{(i,j)}\right)=-y^{(i,j)}\log\left(f_{(w,b,x)}(x)\right)-\left(1-y^{(i,j)}\right)\log\left(1-f_{(w,b,x)}(x)\right)$$

Loss for single example

$$J(w,b,x)=\sum_{(i,j):r(i,j)=1}L\left(f_{(w,b,x)}(x),y^{(i,j)}\right)$$

we're going to use that loss function.

Users who have not rated any movies

Movie	Alice(1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

$$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

$$\min_{\substack{w^{(1)},\ldots w^{(n_u)} \\ b^{(1)},\ldots b^{(n_u)} \\ x^{(1)},\ldots x^{(n_m)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

Just to write out all the ratings including the question marks in a more

$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $b = 0$ $w^{(j)} \cdot x^{(i)} + b^{(j)}$

3:03 / 8:45

Custom Training Loop

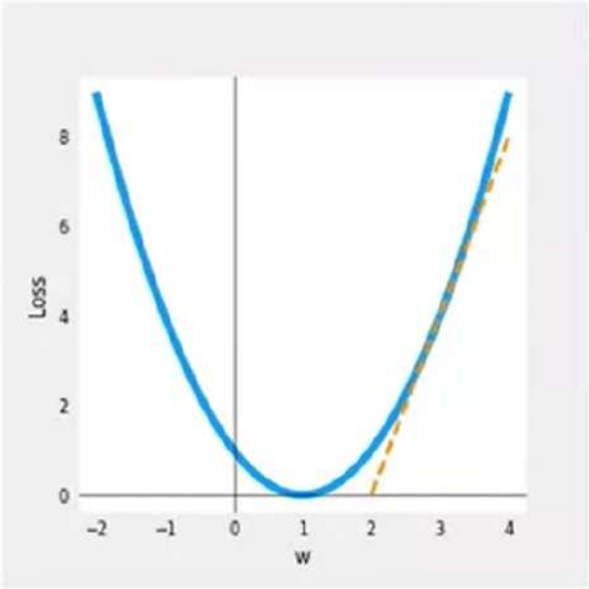
$$J = (\underbrace{wx}_{f(x)} - \underbrace{1}_{y})^2$$

Gradient descent algorithm

Repeat until convergence

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

Fix b = 0 for this example




```
w = tf.Variable(3.0)
x = 1.0
y = 1.0 # target value
alpha = 0.01

iterations = 30
for iter in range(iterations):
    # Use TensorFlow's Gradient tape to record the steps
    # used to compute the cost J, to enable auto differentiation.
    with tf.GradientTape() as tape:
        fwb = w*x
        costJ = (fwb - y)**2

    # Use the gradient tape to calculate the gradients
    # of the cost with respect to the parameter w.
    [dJdw] = tape.gradient( costJ, [w] )

    # Run one step of gradient descent by updating
    # the value of w to reduce the cost.
    w.assign_add(-alpha * dJdw)
```

Tf.variables are the parameters we want to optimize

f(x)

tf.variables require special function to

filtering algorithm correctly.

Implementation in TensorFlow

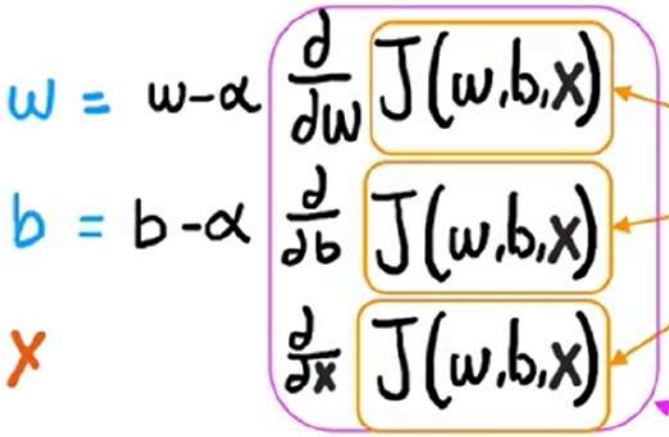
```
# Instantiate an optimizer.
optimizer = keras.optimizers.Adam(learning_rate=1e-1)

iterations = 200
for iter in range(iterations):
    # Use TensorFlow's GradientTape
    # to record the operations used to compute the cost
    with tf.GradientTape() as tape:

        # Compute the cost (forward pass is included in cost)
        cost_value = cofiCostFuncV(X, W, b, Ynorm, R,
            num_users, num_movies, lambda)
                 n_u        n_m
    # Use the gradient tape to automatically retrieve
    # the gradients of the trainable variables with respect to
        the loss
    grads = tape.gradient( cost_value, [X,W,b] )

    # Run one step of gradient descent by updating
    # the value of the variables to minimize the loss.
    optimizer.apply_gradients( zip(grads, [X,W,b]) )
```

And then by asking it to give you grads equals tape.gradient,

Dataset credit: Harper and Konstan. 2015. The MovieLens Datasets: History and Context

Implementation in TensorFlow

Gradient descent algorithm




```
# Instantiate an optimizer.
optimizer = keras.optimizers.Adam(learning_rate=1e-1)

iterations = 200
for iter in range(iterations):
    # Use TensorFlow's GradientTape
    # to record the operations used to compute the cost
    with tf.GradientTape() as tape:

        # Compute the cost (forward pass is included in cost)
        cost_value = cofiCostFuncV(X, W, b, Ynorm, R,
                                   num_users, num_movies, lambda)
                                   # n_u      # n_m

    # Use the gradient tape to automatically retrieve
    # the gradients of the trainable variables with respect to
    #  the loss
    grads = tape.gradient( cost_value, [X,W,b] )

    # Run one step of gradient descent by updating
    # the value of the variables to minimize the loss.
    optimizer.apply_gradients( zip(grads, [X,W,b]) )
```

Dataset credit: Harper and Konstan. 2015. The MovieLens Datasets: History and Context

And you repeat until convergence.

Finding related items

The features $x^{(i)}$ of item i are quite hard to interpret.

To find other items related to it,
find item k with $x^{(k)}$ similar to $x^{(i)}$

romance
action

x_1, x_2, x_3

n

i.e. with smallest distance

$$\sum_{l=1}^{n} \left(x_l^{(k)} - x_l^{(i)} \right)^2$$

$x^{(k)}$ $x^{(i)}$

$$\left\| x^{(k)} - x^{(i)} \right\|^2$$

an even more powerful recommended system as well.

Collaborative filtering vs Content-based filtering

- **Collaborative filtering:**
  Recommend items to you based on ratings of users who gave similar ratings as you

- **Content-based filtering:**
  Recommend items to you based on features of user and item to find good match

$r(i,j) = 1$ if user j has rated item i

$y^{(i,j)}$ rating given by user j on item i (if defined)

j to denote the rating

Examples of user and item features

User features:

- Age
- Gender (1 hot)
- Country (1 hot, 200)
- Movies watched (1000)
- Average rating per genre
- ...

$x_u^{(j)}$ for user j

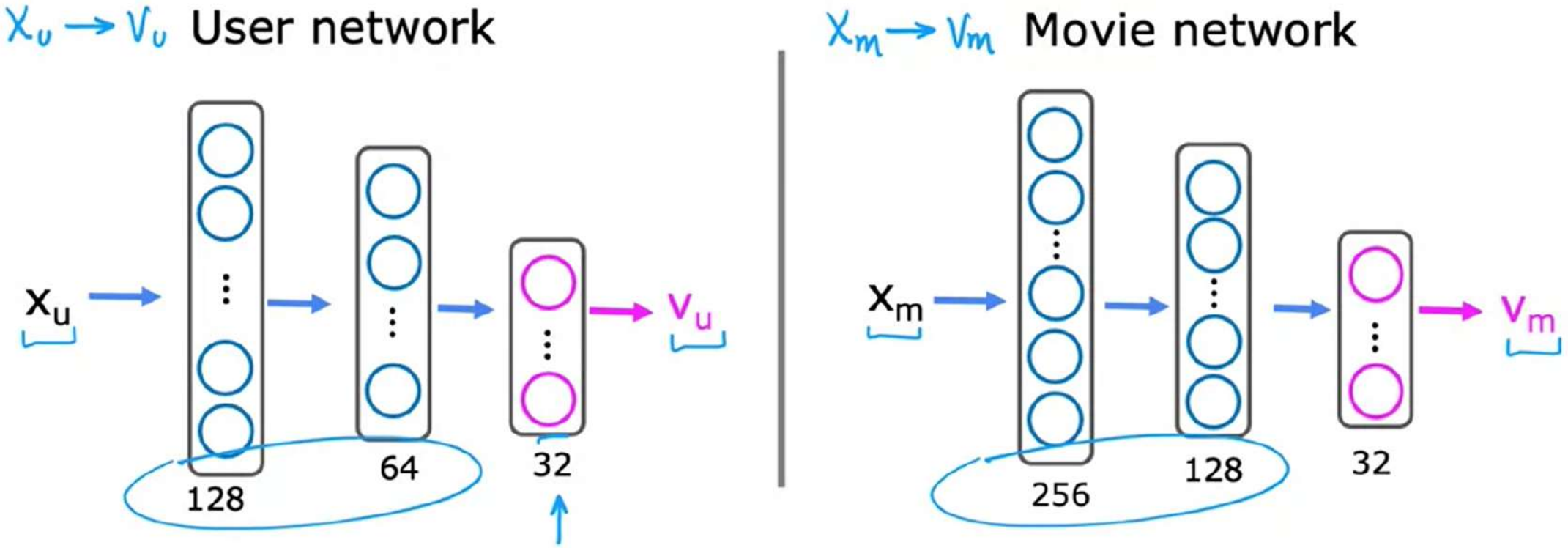
Movie features:

- Year
- Genre/Genres
- Reviews
- Average rating
- ...

$x_m^{(i)}$ for movie i

This feature again depends on the ratings

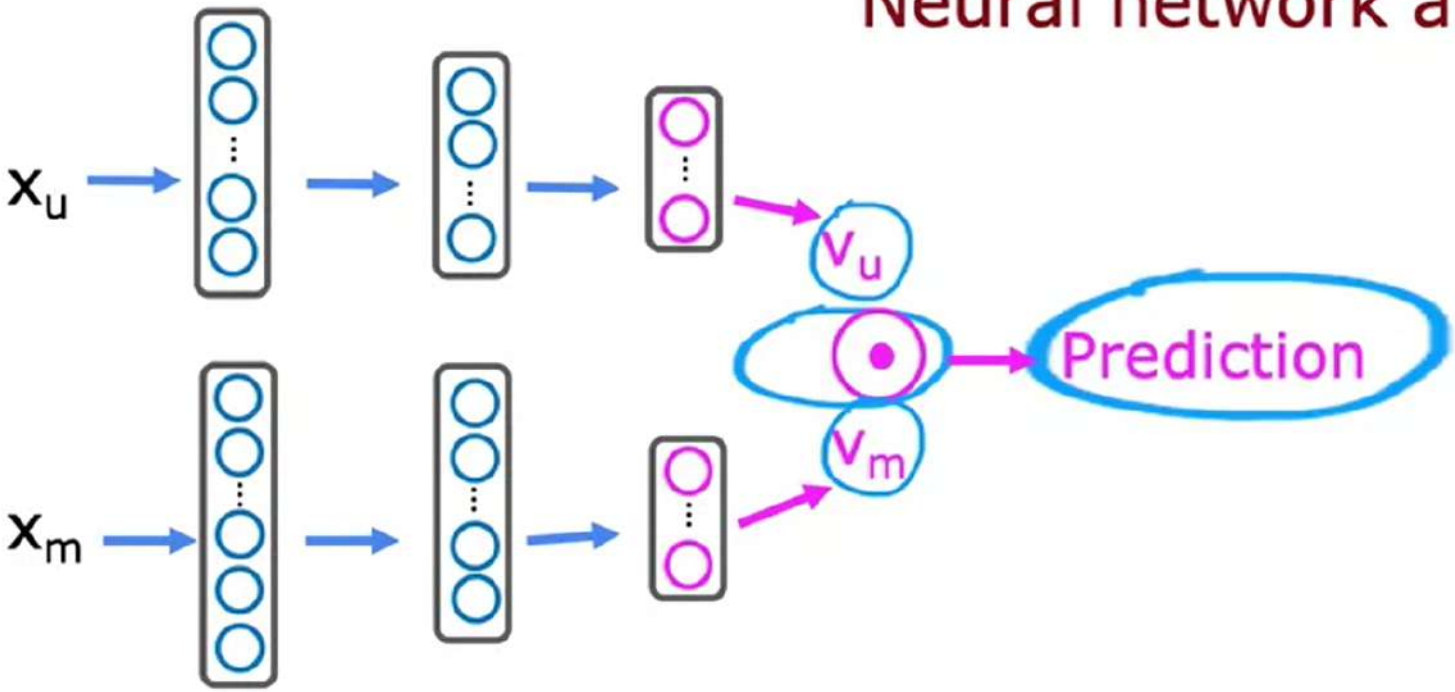
Neural network architecture



Prediction : $v_u \cdot v_m$

All the output layer needs to

Neural network architecture



Cost function $J = \sum_{(i,j):r(i,j)=1} \left(v_u^{(j)} \cdot v_m^{(i)} - y^{(i,j)}\right)^2$ + NN regularization term

v_m^i minus y^ij squared.

Learned user and item vectors:

$v_u^{(j)}$ is a vector of length 32 that describes user j with features $x_u^{(j)}$

$v_m^{(i)}$ is a vector of length 32 that describes movie i with features $x_m^{(i)}$

To find movies similar to movie i: $\| v_m^{(k)} - v_m^{(i)} \|^2$ small

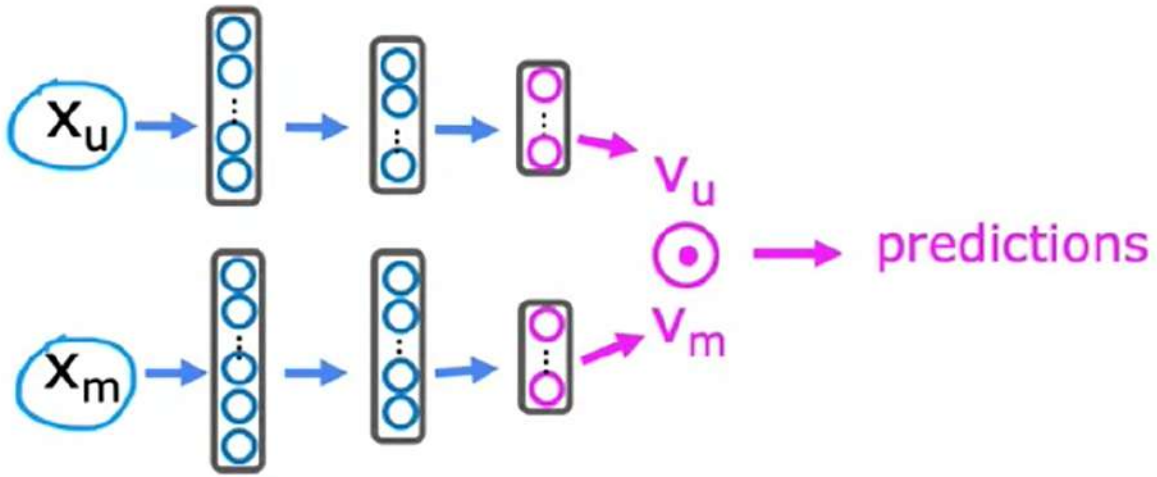
$\| x^{(k)} - x^{(i)} \|^2$

Note: This can be pre-computed ahead of time

later when we talk about scaling

How to efficiently find recommendation from a large set of items?

- Movies 1000+
- Ads 1m+
- Songs 10m+
- Products 10m+



becomes computational e infeasible.

Two steps: Retrieval & Ranking

Retrieval:

- Generate large list of plausible item candidates ~100s

  e.g.

  1) For each of the last 10 movies watched by the user, find 10 most similar movies

  $$\| v_m^{(k)} - v_m^{(i)} \|^2$$

  2) For most viewed 3 genres, find the top 10 movies

  3) Top 20 movies in the country

- Combine retrieved items into list, removing duplicates and items already watched/purchased

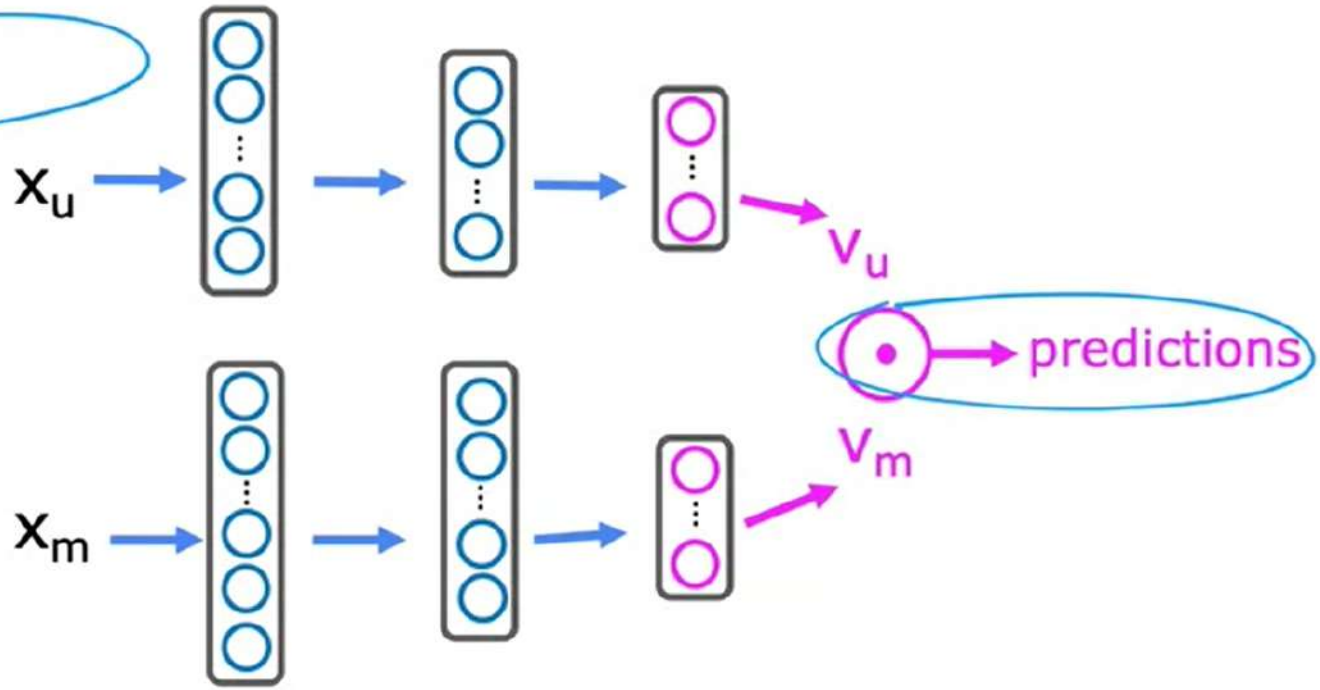
that the user has already purchased and

Two steps: Retrieval & ranking

Ranking:

~100s

- Take list retrieved and rank using learned model



- Display ranked items to user

what you think the user will give.

What is the goal of the recommender system?

Recommend:

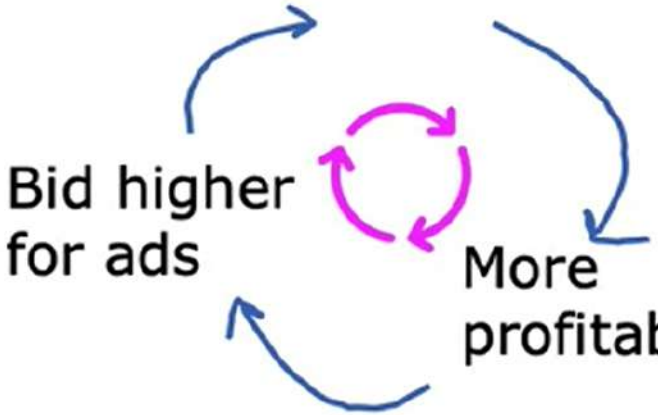
- Movies most likely to be rated 5 stars by user
- Products most likely to be purchased
- Ads most likely to be clicked on + high bid
- Products generating the largest profit
- Video leading to maximum watch time

Specifically, websites that are

Ethical considerations with recommender systems

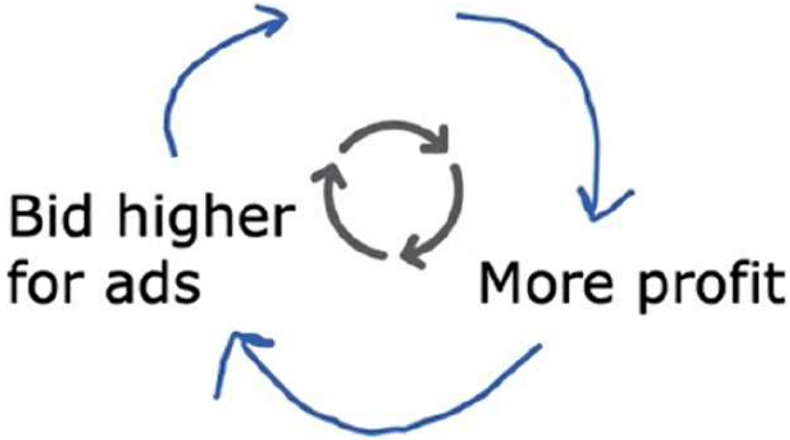
Travel industry

Good travel experience to more users



Payday loans

Squeeze customers more



to get sent more traffic.

Other problematic cases:

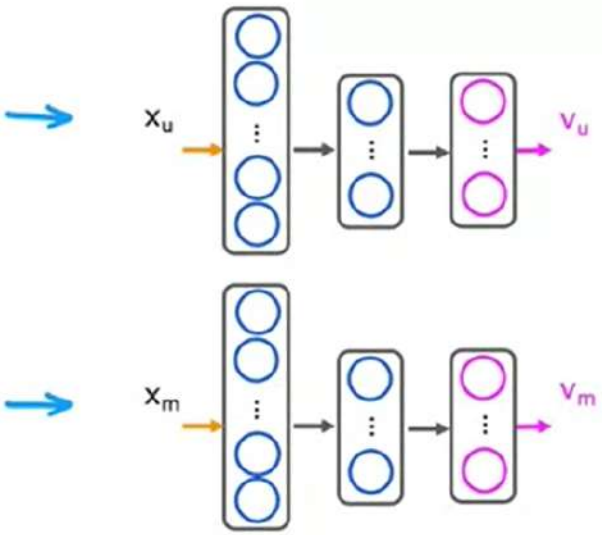
- Maximizing user engagement (e.g. watch time) has led to large social media/video sharing sites to amplify conspiracy theories and hate/toxicity

  → Amelioration : Filter out problematic content such as hate speech, fraud, scams and violent content

- Can a ranking system maximize your profit rather than users' welfare be presented in a transparent way?

  → Amelioration : Be transparent with users

transparent with
users about why we're



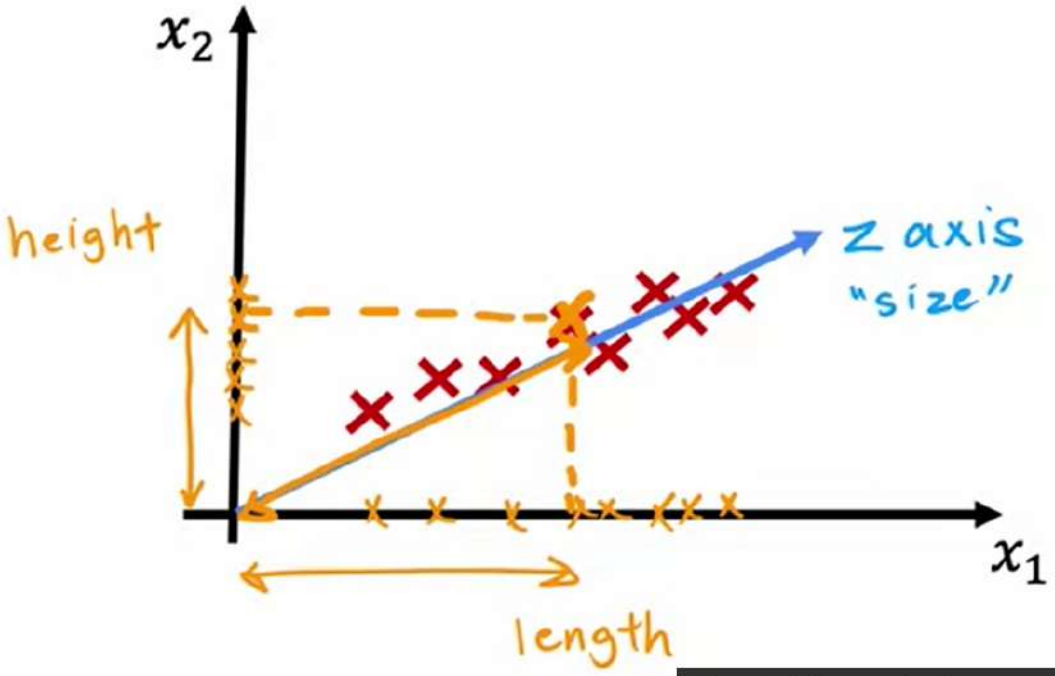

```
user_NN = tf.keras.models.Sequential([
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(32)
])
```

```
item_NN = tf.keras.models.Sequential([
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(32 )
])
```

```
# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vu = user_NN(input_user)
vu = tf.linalg.l2_normalize(vu, axis=1)

# create the item input and point to the base network
input_item = tf.keras.layers.Input(shape=(num_item_features))
vm = item_NN(input_item)
vm = tf.linalg.l2_normalize(vm, axis=1)

# measure the similarity of the two vector outputs
output = tf.keras.layers.Dot(axes=1)([vu, vm])

# specify the inputs and output of the model
model = Model([input_user, input_item], output)

# Specify the cost function
cost_fn = tf.keras.losses.MeanSquaredError()
```

Size

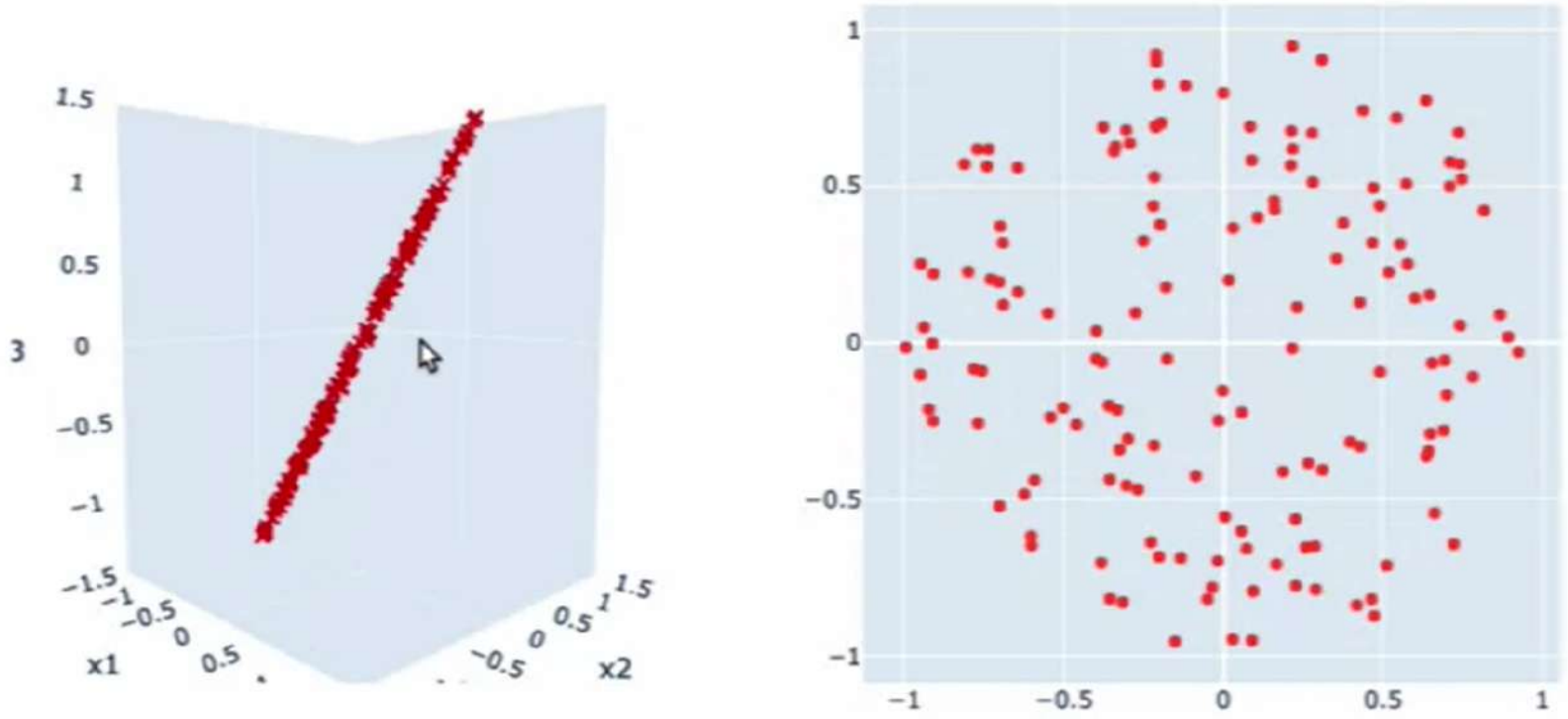
PCA: find new axis and coordinates



X_1	X_2
length	height

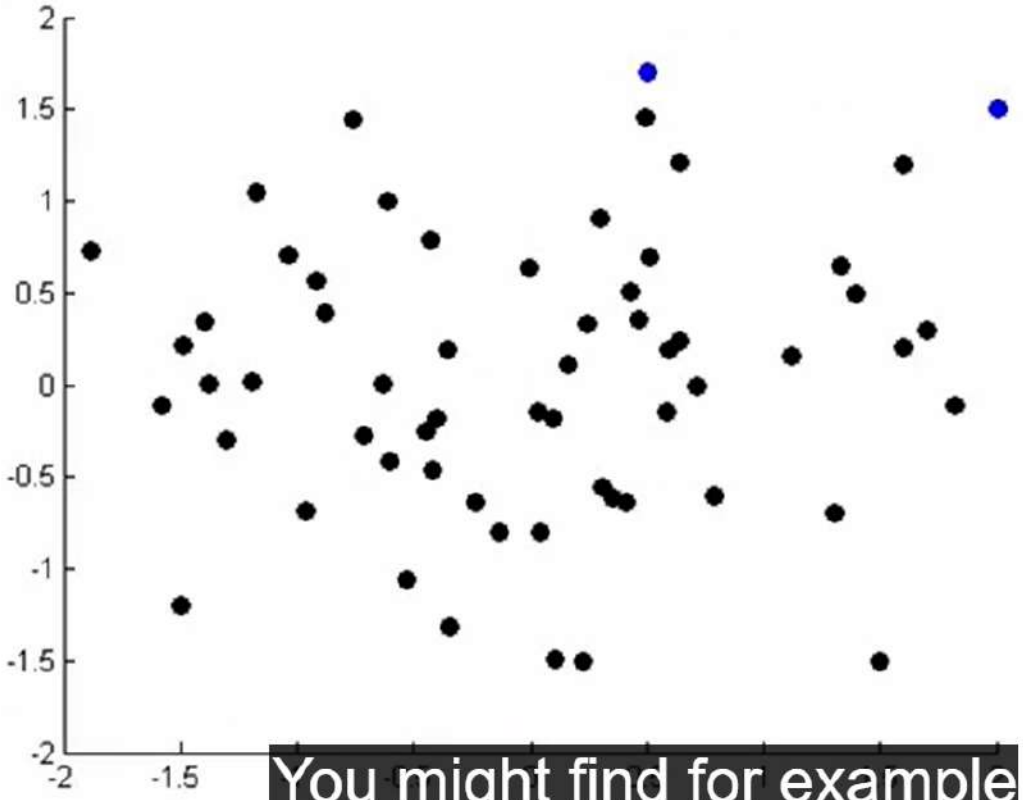
But the idea of PCA is to
find one or more new axes,

From 3D to 2D



on a two-dimensional pancake,

Data visualization



You might find for example
that Z_1 loosely corresponds

Anomaly detection algorithm

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

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

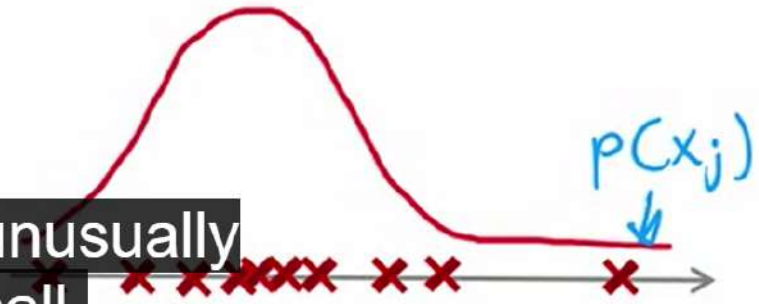
Vectorized formula

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

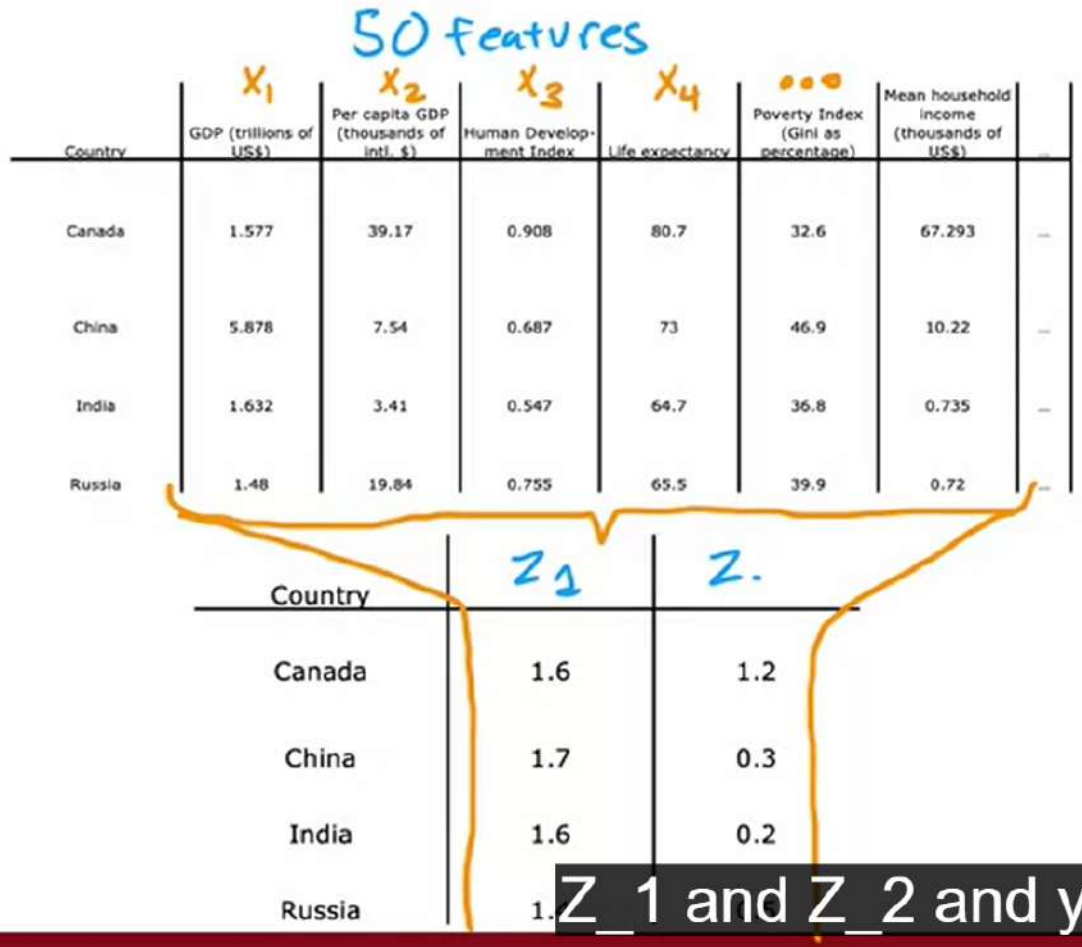
3. Given new example x, compute $p(x)$:

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

Anomaly if $p(x) < \varepsilon$



has any features that are unusually large or unusually small.

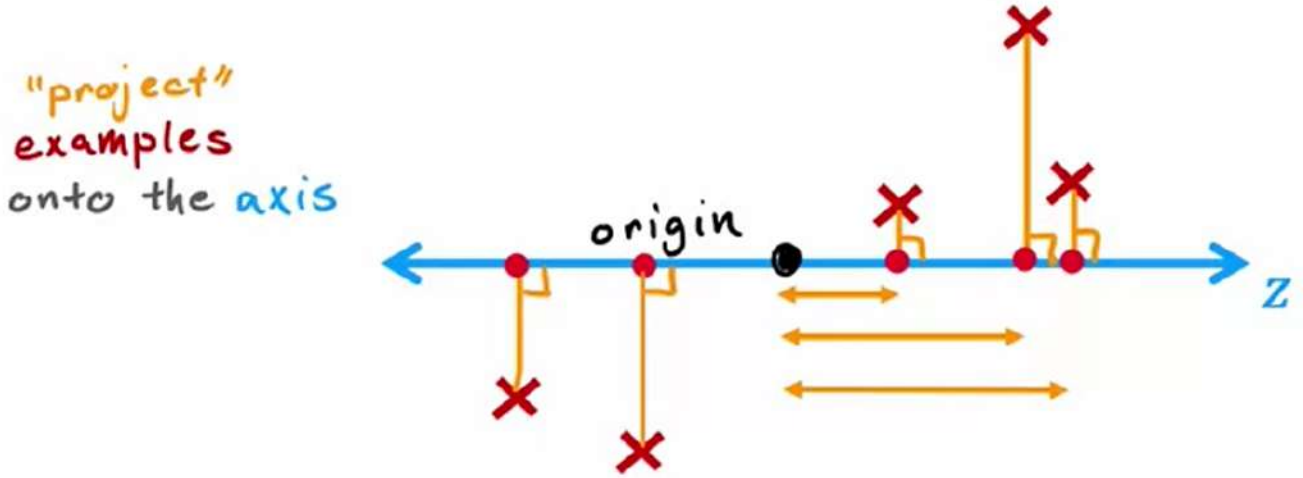


Z_1 and Z_2 and you can then

PCA algorithm

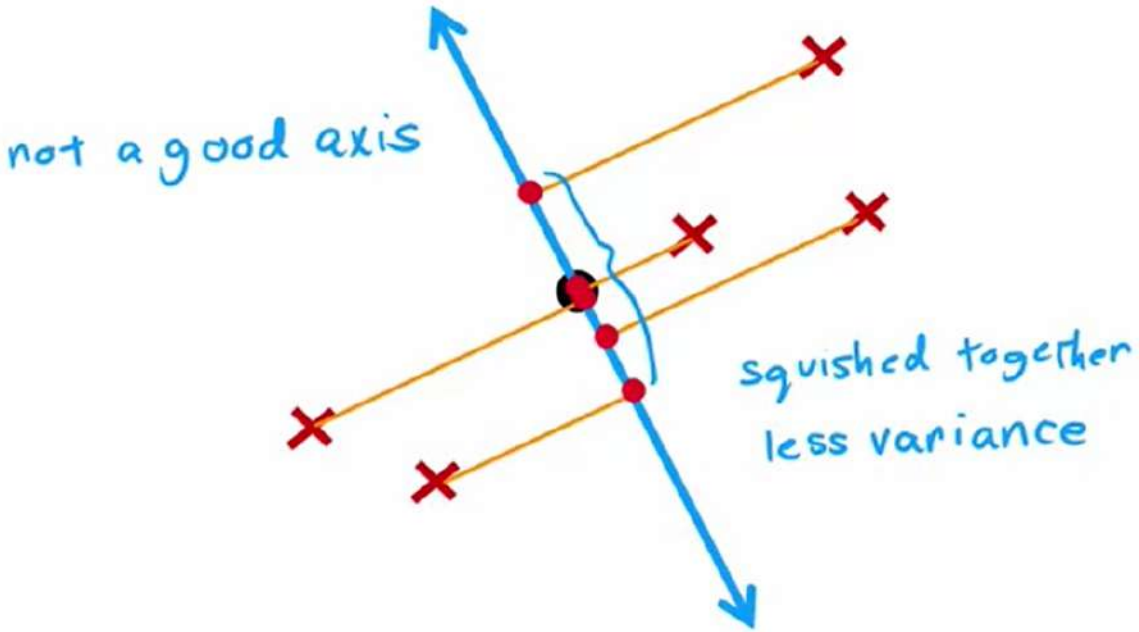
To examine what PCA does,

Choose an axis



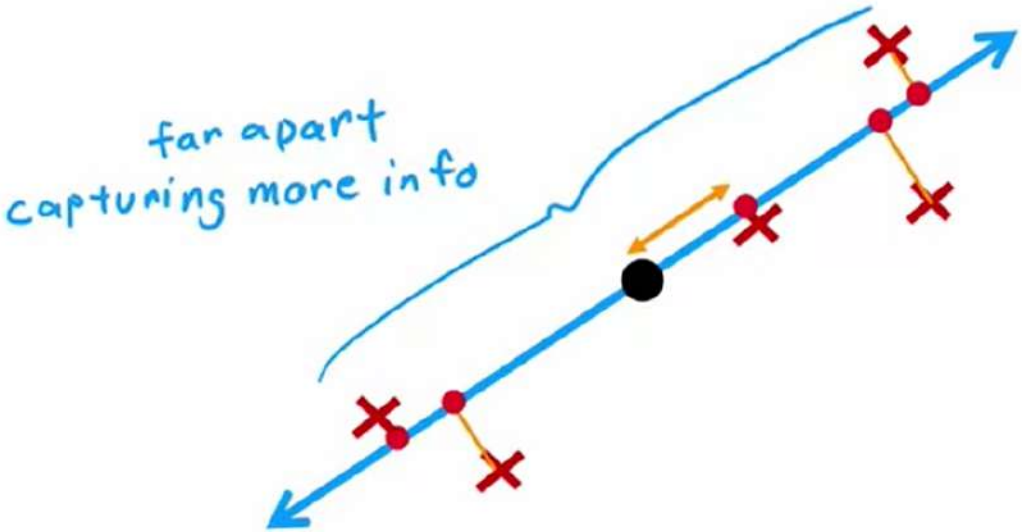
This choice isn't
too bad because

Choose an axis



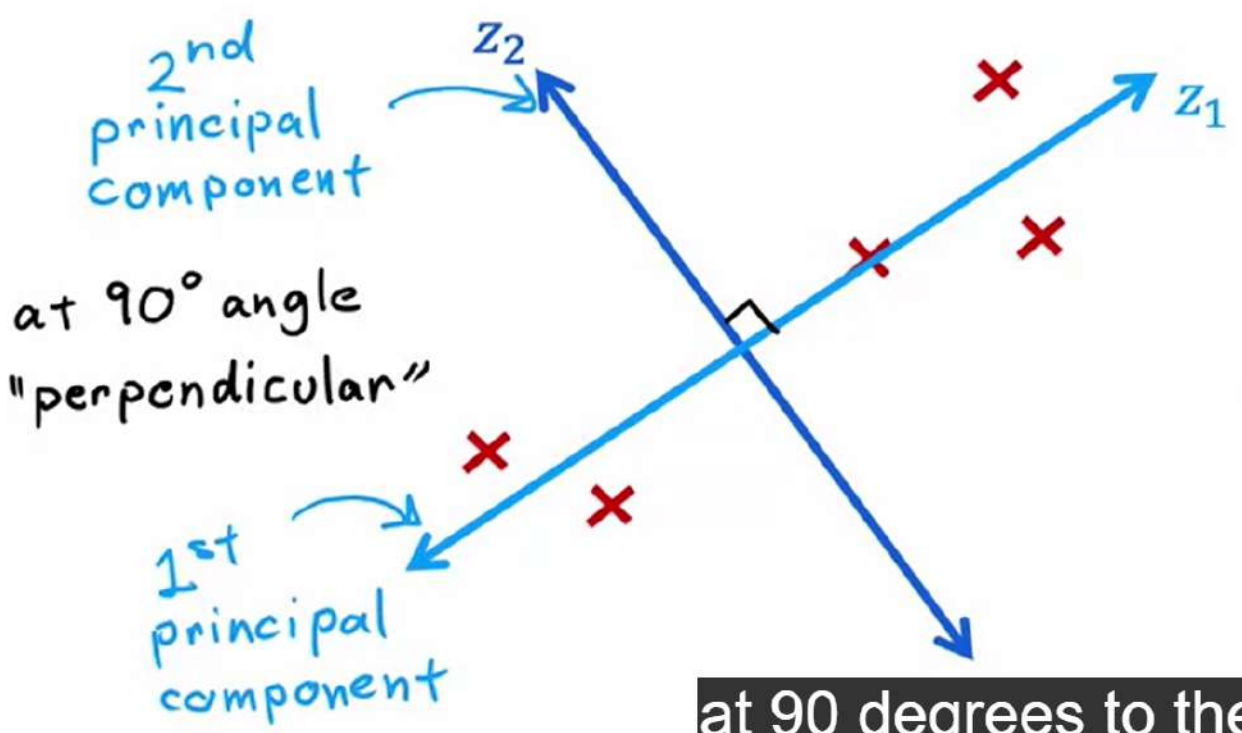
you're capturing much less
of the information in

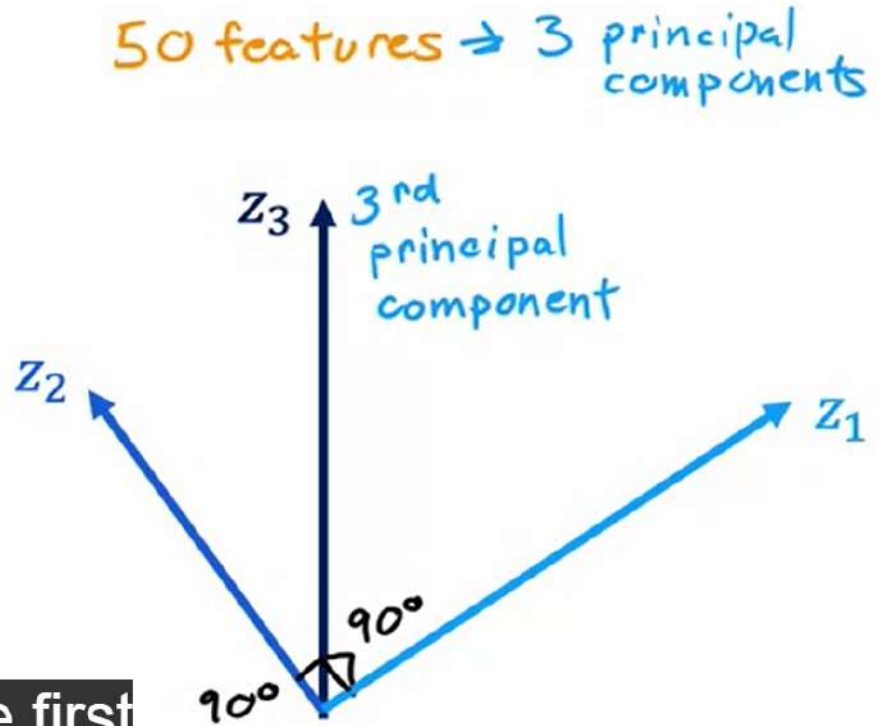
Choose an axis



this axis is called the principal component.

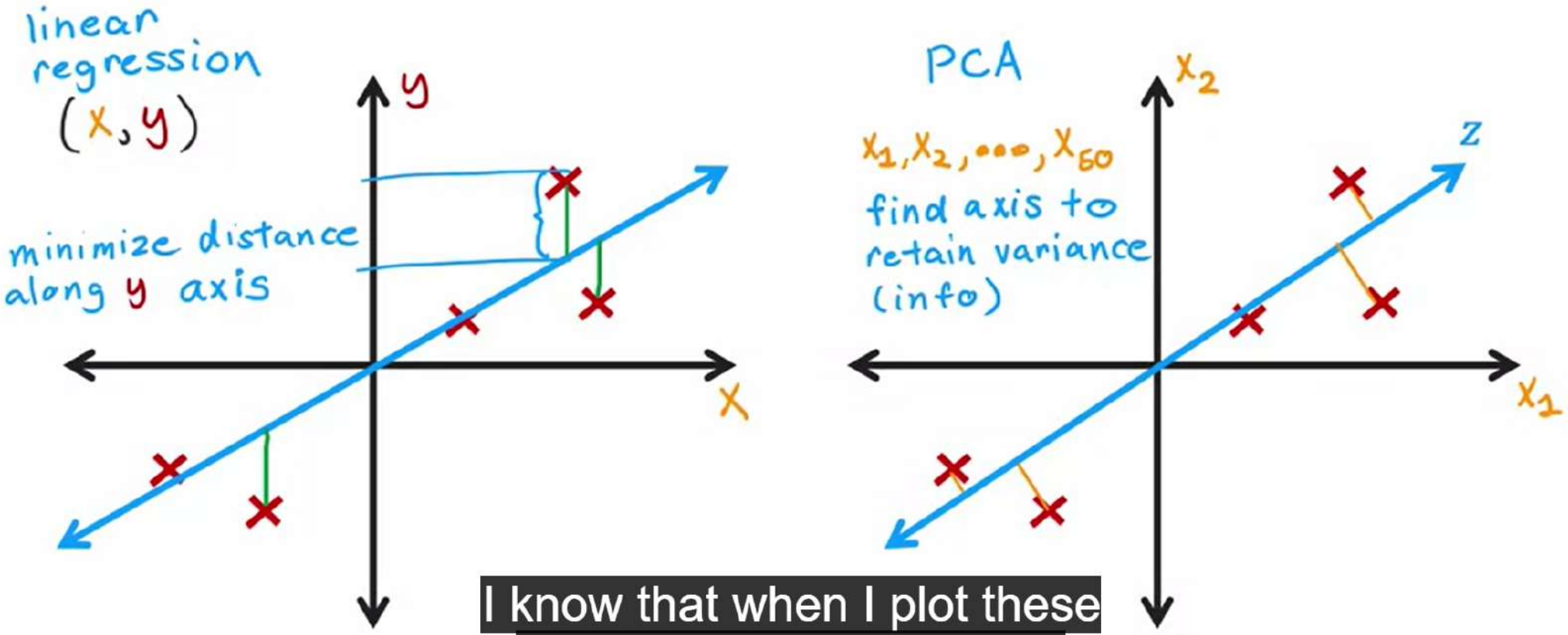
More principal components





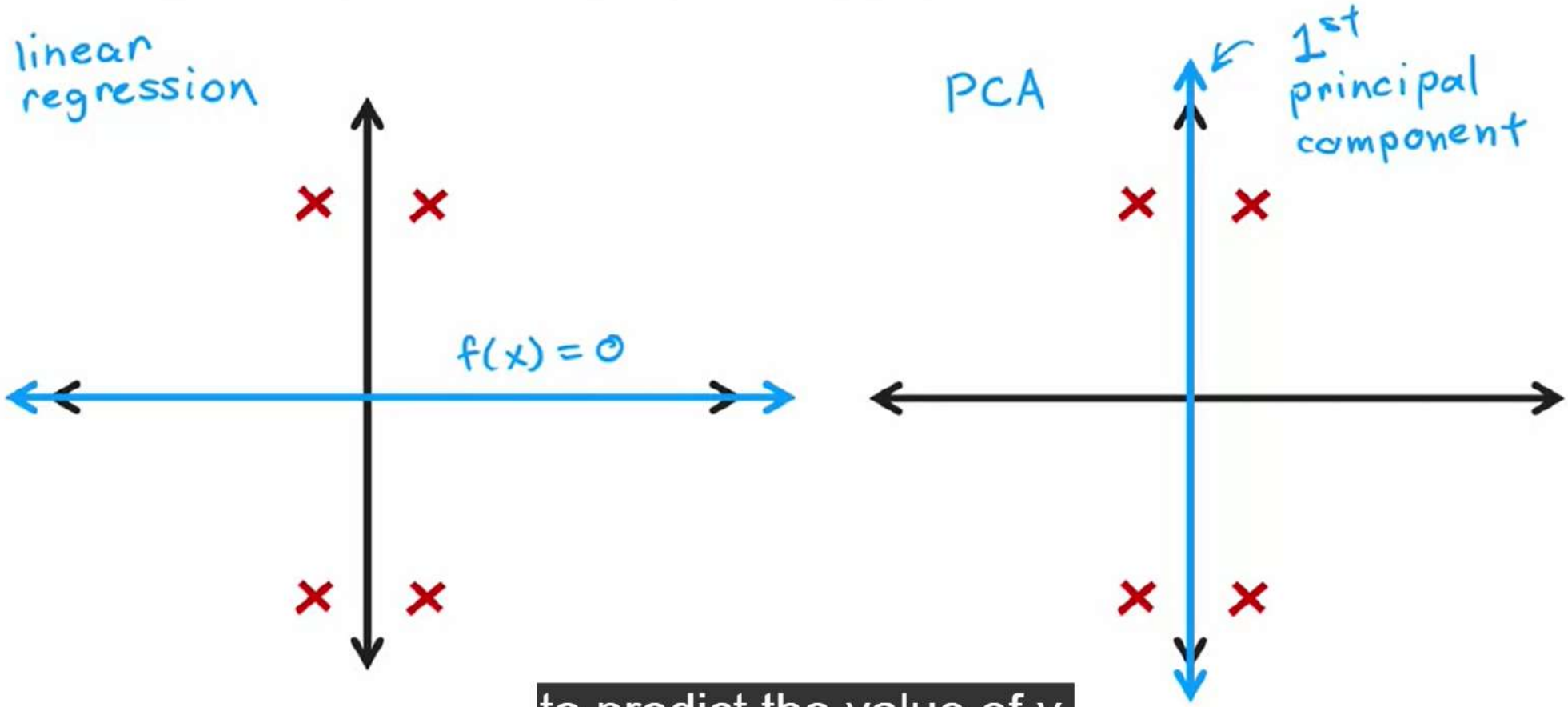
at 90 degrees to the first
and the second axis.

PCA is not linear regression



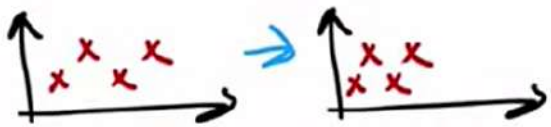
I know that when I plot these things in two-dimensions,

PCA is not linear regression



to predict the value of y,

PCA in scikit-learn

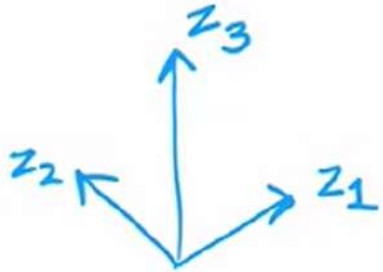


Optional pre-processing: Perform feature scaling

for visualization

1. "fit" the data to obtain 2 (or 3) new axes (principal components)

   `fit` includes mean normalization



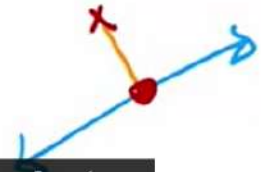
info

2. Optionally examine how much variance is explained by each principal component.

   `explained_variance_ratio`

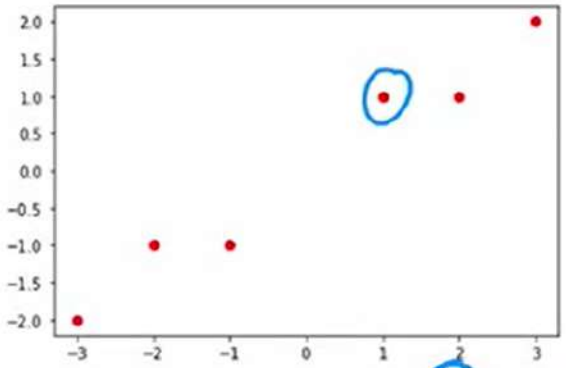
3. Transform (project) the data onto the new axes

   `transform`



to visualize your data.

Example



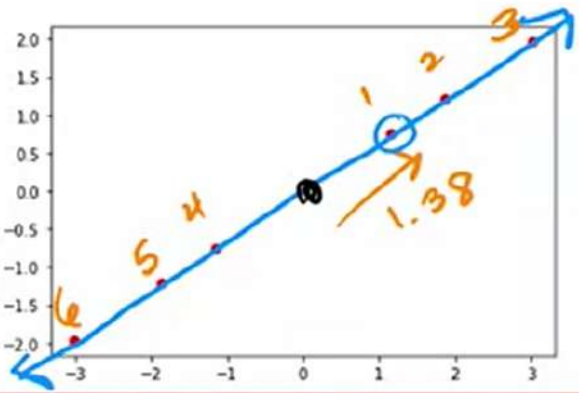
```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

2D

```
pca_1 = PCA(n_components=1)
pca_1.fit(X)
pca_1.explained_variance_ratio_
X_trans_1 = pca_1.transform(X)
X_reduced_1 = pca.inverse_transform(X_trans_1)
```

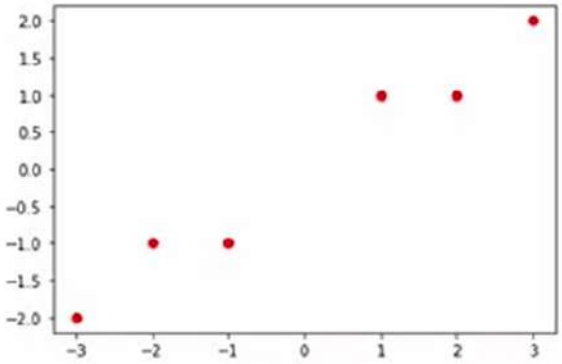
0.992

```
array([
 [ 1.38340578],
 [ 2.22189802],
 [ 3.6053038 ],
 [-1.38340578],
 [-2.22189802],
 [-3.6053038 ]])
```



This data is
two-dimensional data,

Example



```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

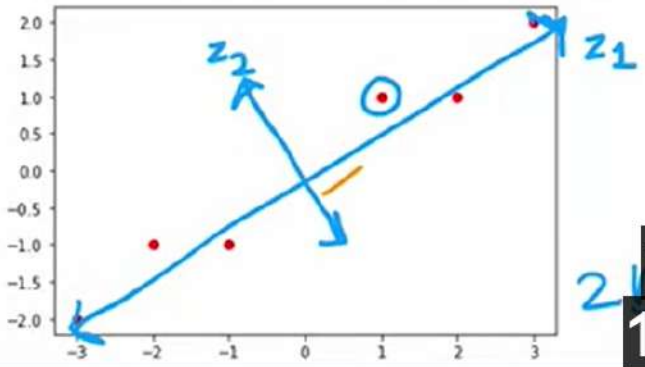
2D

```
pca_2 = PCA(n_components=2)
pca_2.fit(X)
pca_2.explained_variance_ratio_
X_trans_2 = pca.transform(X)
X_reduced_2 = pca.inverse_transform(X_trans_2)
```

z_1 z_2
0.992 0.008

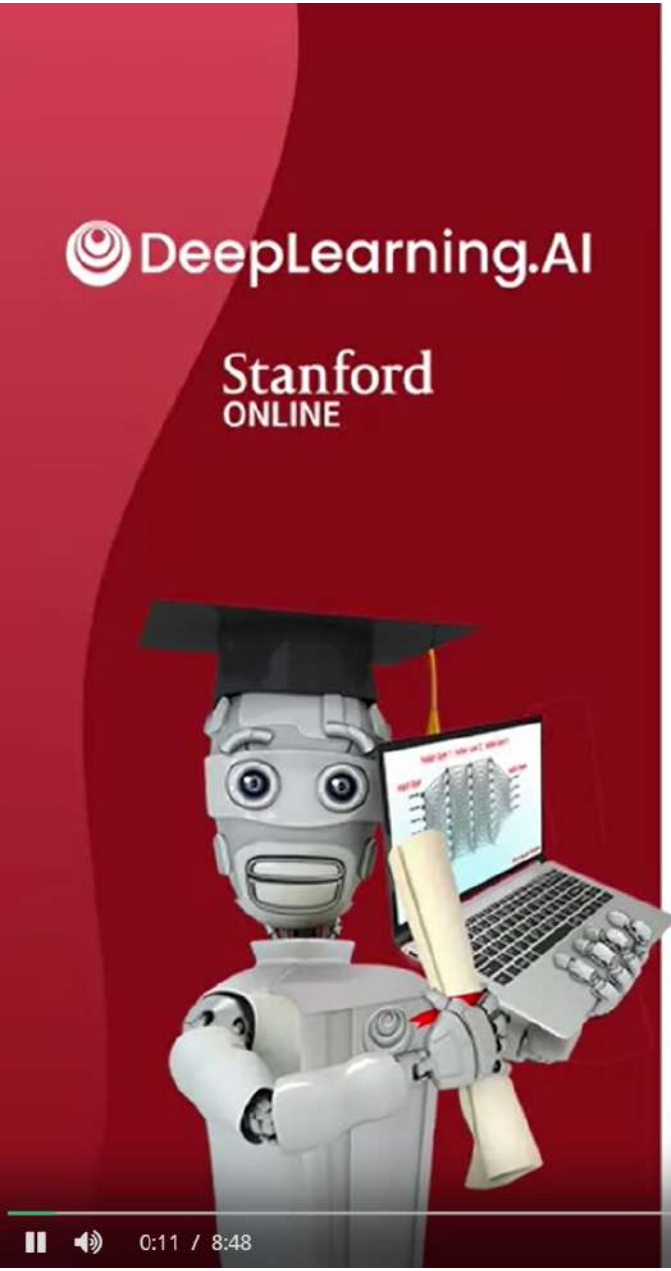
z_1 z_2

```
array([
[ 1.38340578,  0.2935787 ],
[ 2.22189802, -0.25133484],
[ 3.6053038 ,  0.04224385],
[-1.38340578, -0.2935787 ],
[-2.22189802,  0.25133484],
[-3.6053038 , -0.04224385]])
```



2D

1] has a distance of
1.38 on the z_1 axis,



Reinforcement Learning Introduction

What is Reinforcement Learning?

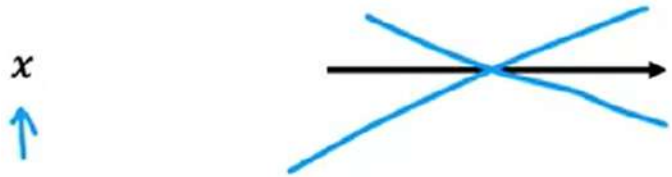
specialization, but
I'm looking forward to this week,

0:11 / 8:48

Reinforcement Learning

position of helicopter → how to move control sticks

state s → action a



reward function

positive reward : helicopter flying well +1

negative reward : helicopter flying poorly −1000

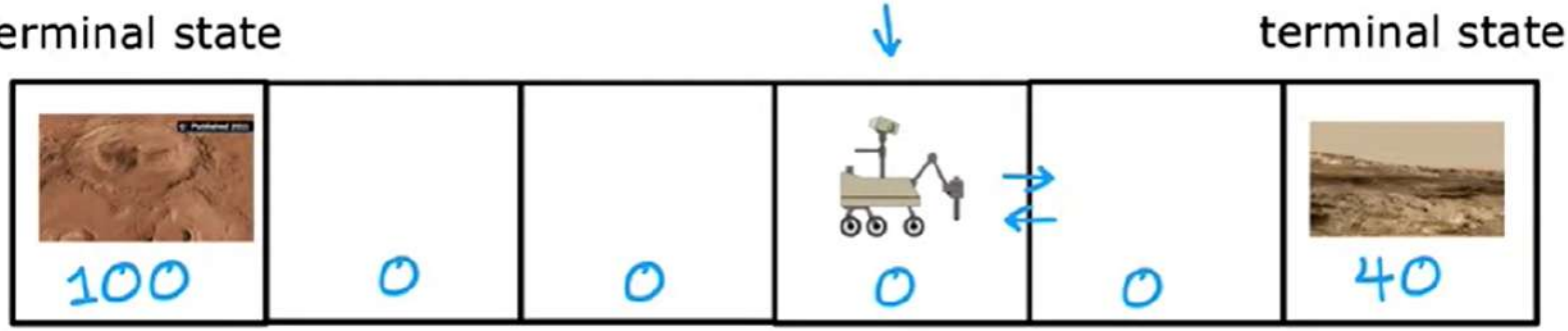
time flying well and
hopefully to never crash.

Applications

- Controlling robots
- Factory optimization
- Financial (stock) trading
- Playing games (including video games)

Finally, there have also been many applications of reinforcement

Mars Rover Example



left

right

$\rightarrow (s, a, R(s), s')$

state 4 3 2 1

0 0 0 100

0 0 40

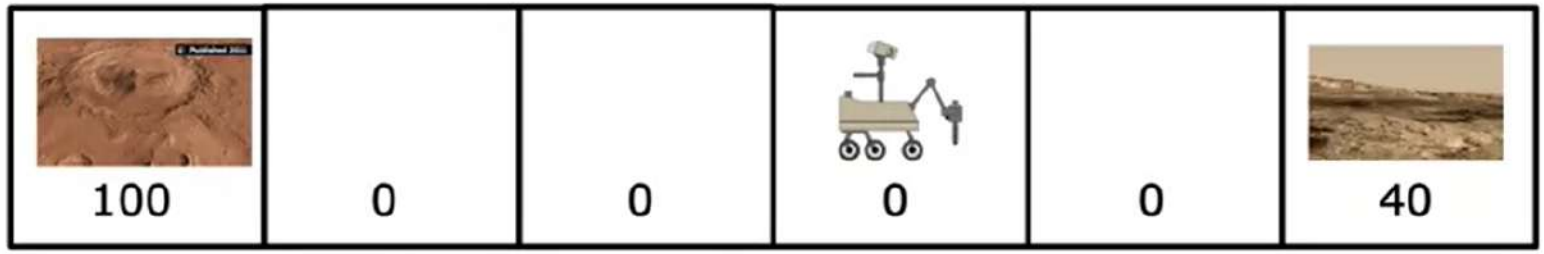
0 0 0 0 0

$(4, \leftarrow, 0, 3)$

with state 4 rather than with state 3.

[Credit: Jagriti Agrawal, Emma Brunskill]

Return



Return $= 0 + (0.9)0 + (0.9)^2 0 + (0.9)^3 100 = 0.729 \times 100 = 72.9$

Return $= R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$ (until terminal state)

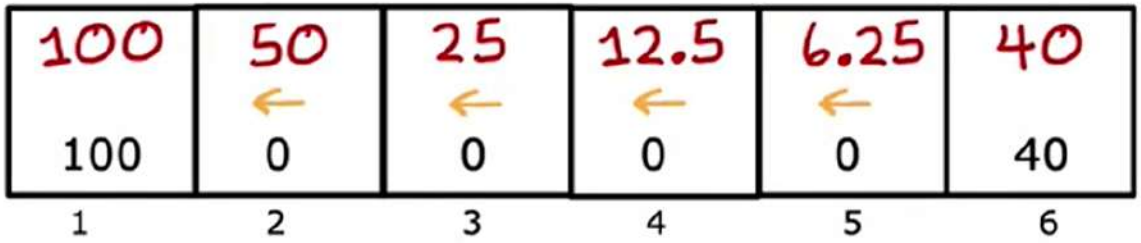
Discount Factor $\gamma = 0.9 \quad 0.99 \quad 0.999$

$\gamma = 0.5$

Return $= 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100 = 12$

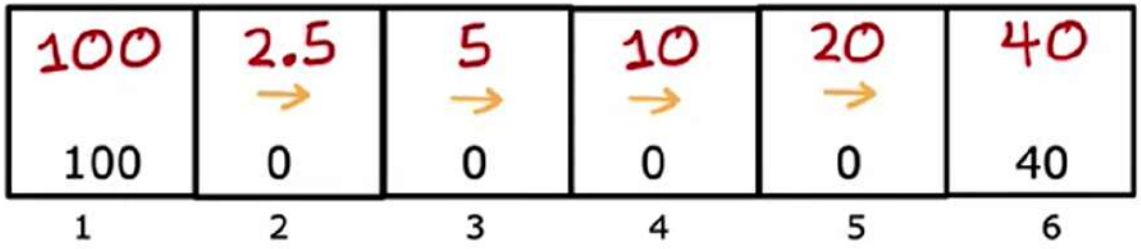
and this turns out to
be a return of 12.5.

Example of Return



$\gamma = 0.5$

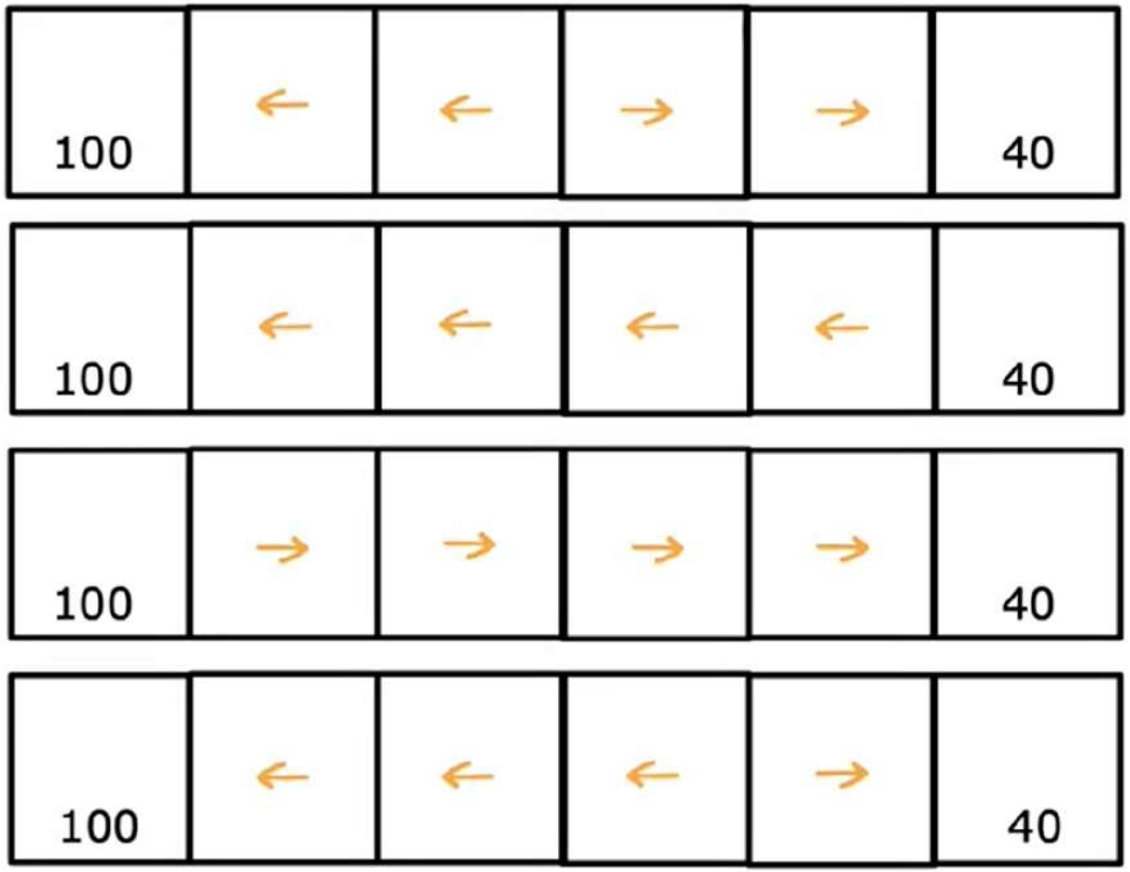
The return depends on the actions you take.



$0 + (0.5)0 + (0.5)^2 40 = 10$

and if you were to
always go to the right.

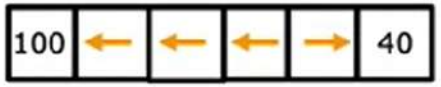
Policy



$\pi(s) = a$
$\pi(2) = \leftarrow$
$\pi(3) = \leftarrow$
$\pi(4) = \leftarrow$
$\pi(5) = \rightarrow$

A policy is a function $\pi(s) = a$ mapping from states to actions, that tells you what action a to take in a given state s.

tells us what action it wants us to take in that state.

	Mars rover 	Helicopter 	Chess 
states	6 states	position of helicopter	pieces on board
actions	← →	how to move control stick	possible move
rewards	100, 0, 40	+1, −1000	+1, 0, −1
discount factor γ	0.5	0.99	0.995
return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
policy π		Find $\pi(s) = a$	Find $\pi(s) = a$

actually has a name.