Copyright Notice

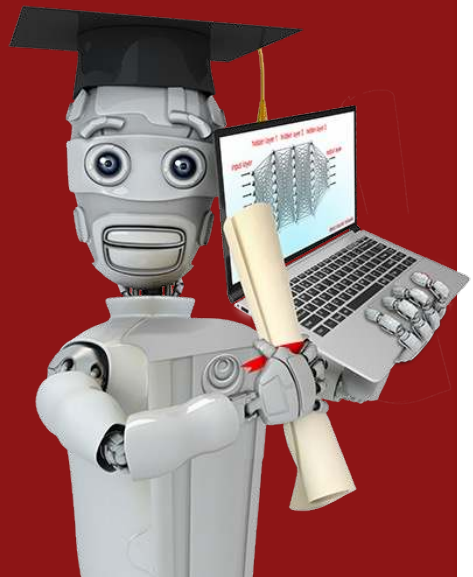
These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

DeepLearning.AI

Stanford
ONLINE



Unsupervised learning, recommender systems and reinforcement learning

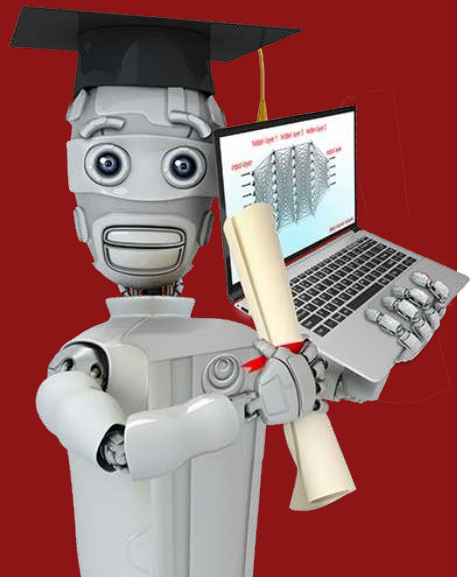
Welcome!

Beyond Supervised Learning

- Unsupervised Learning
  - Clustering
  - Anomaly detection
- Recommender Systems
- Reinforcement Learning

DeepLearning.AI

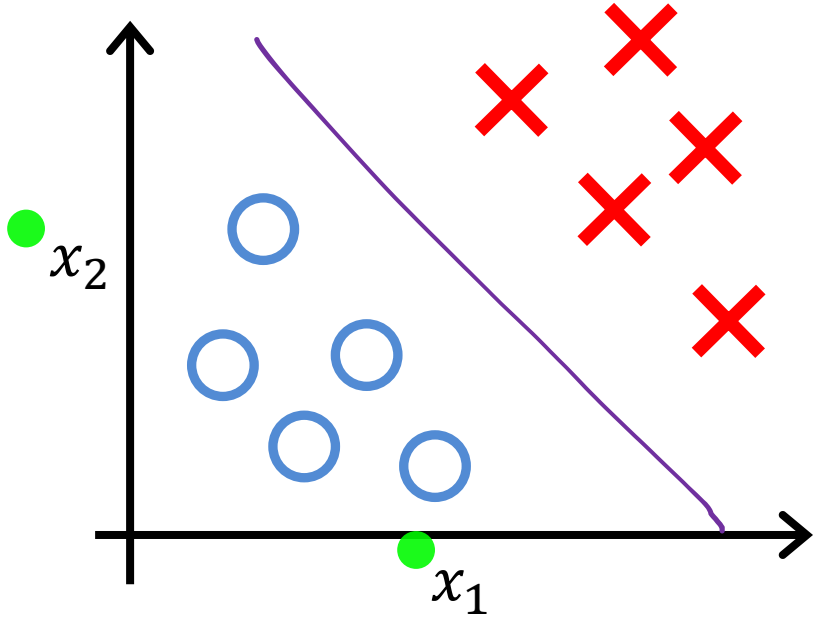
Stanford
ONLINE



Clustering

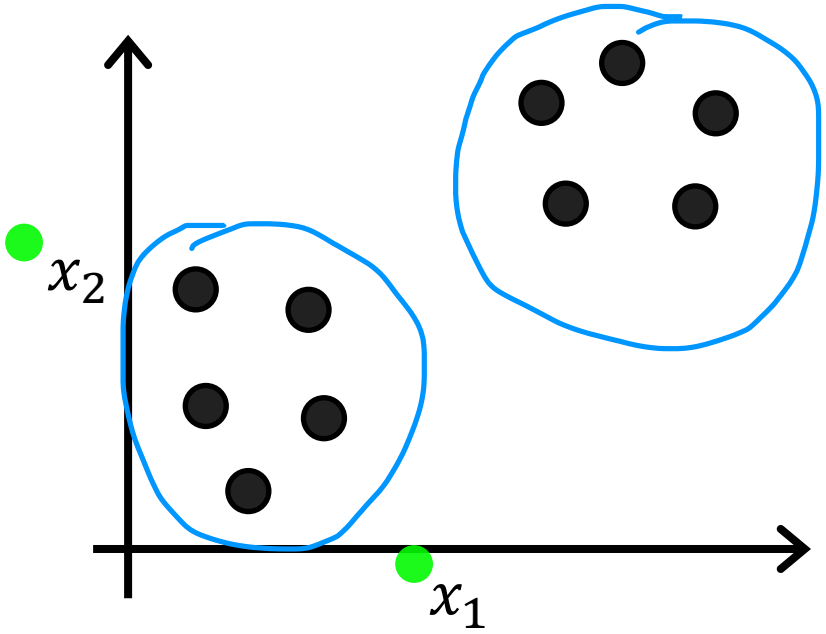
What is clustering?

Supervised learning



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

Unsupervised learning



Clustering

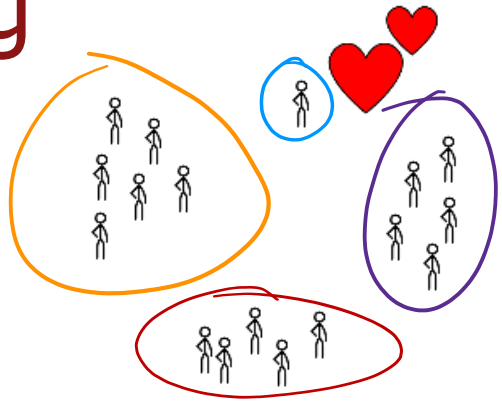
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Applications of clustering

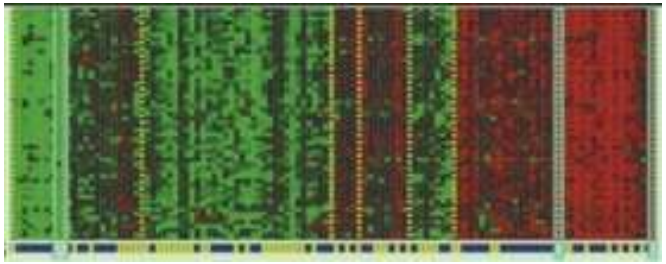


Grouping similar news

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



Market segmentation



DNA analysis

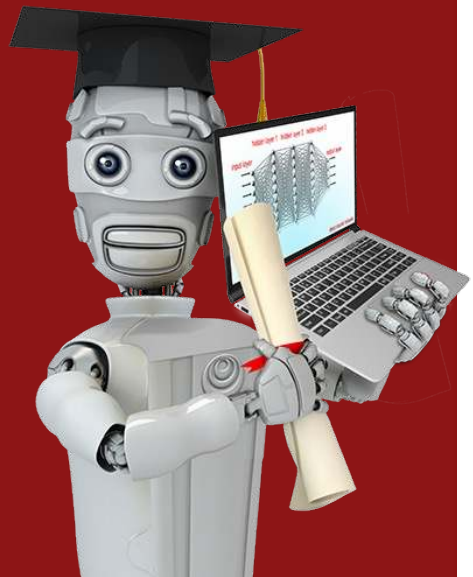


Image credit: NASA/JPL-Calte
ch/E. Churchwell (Univ. of Wisconsin, Madison)

Astronomical data analysis

DeepLearning.AI

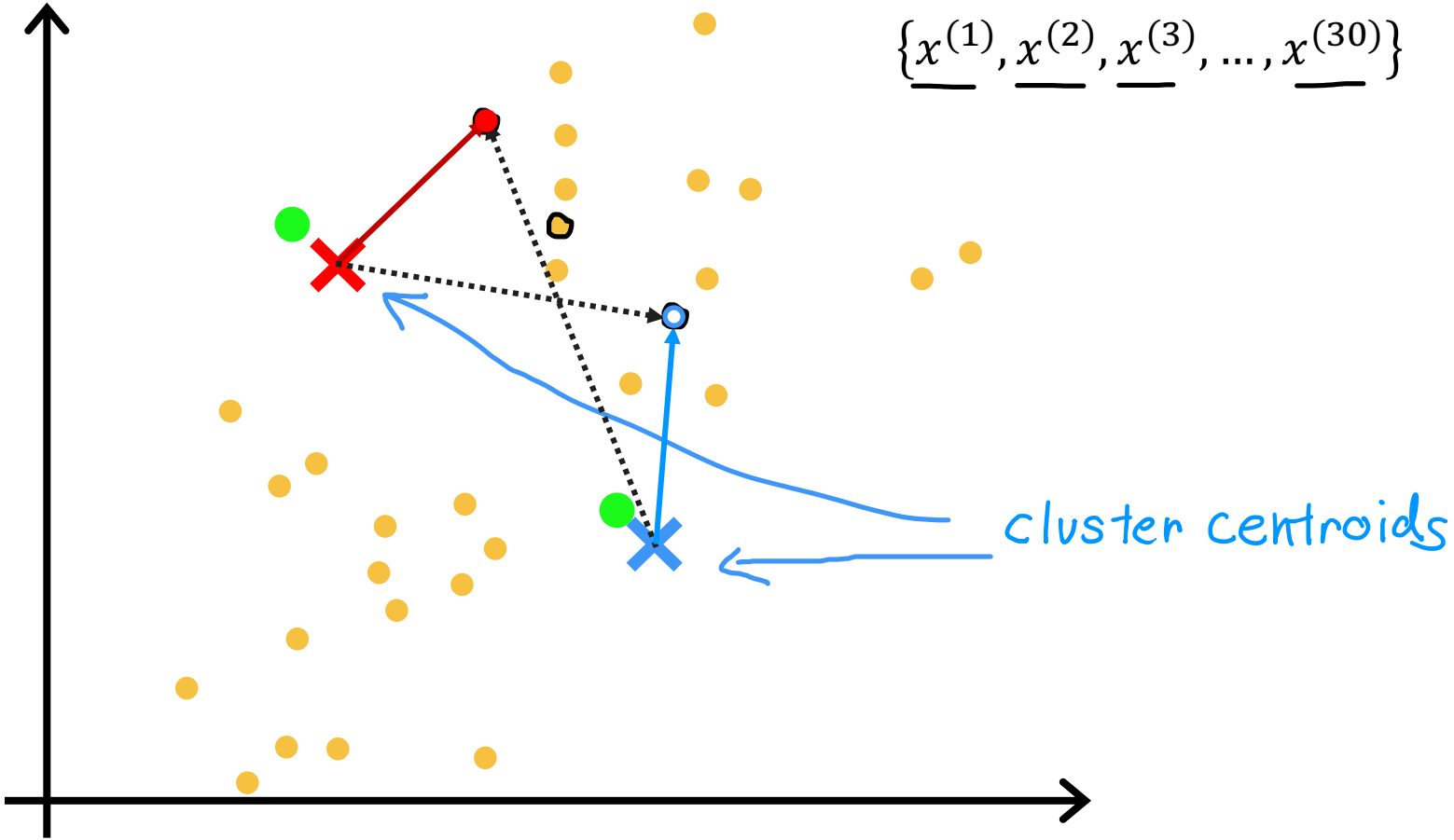
Stanford
ONLINE



Clustering

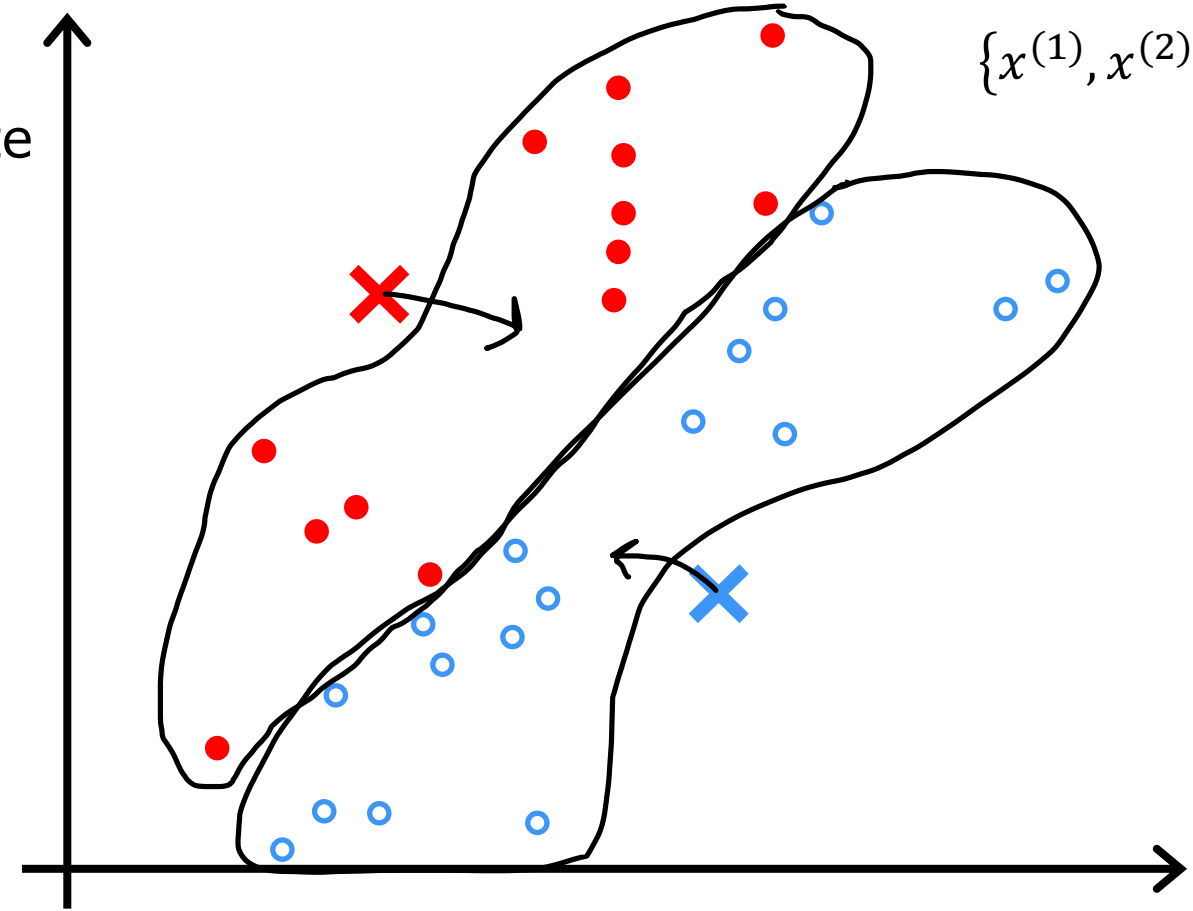
K-means intuition

Step 1:
Assign each point to its closest centroid

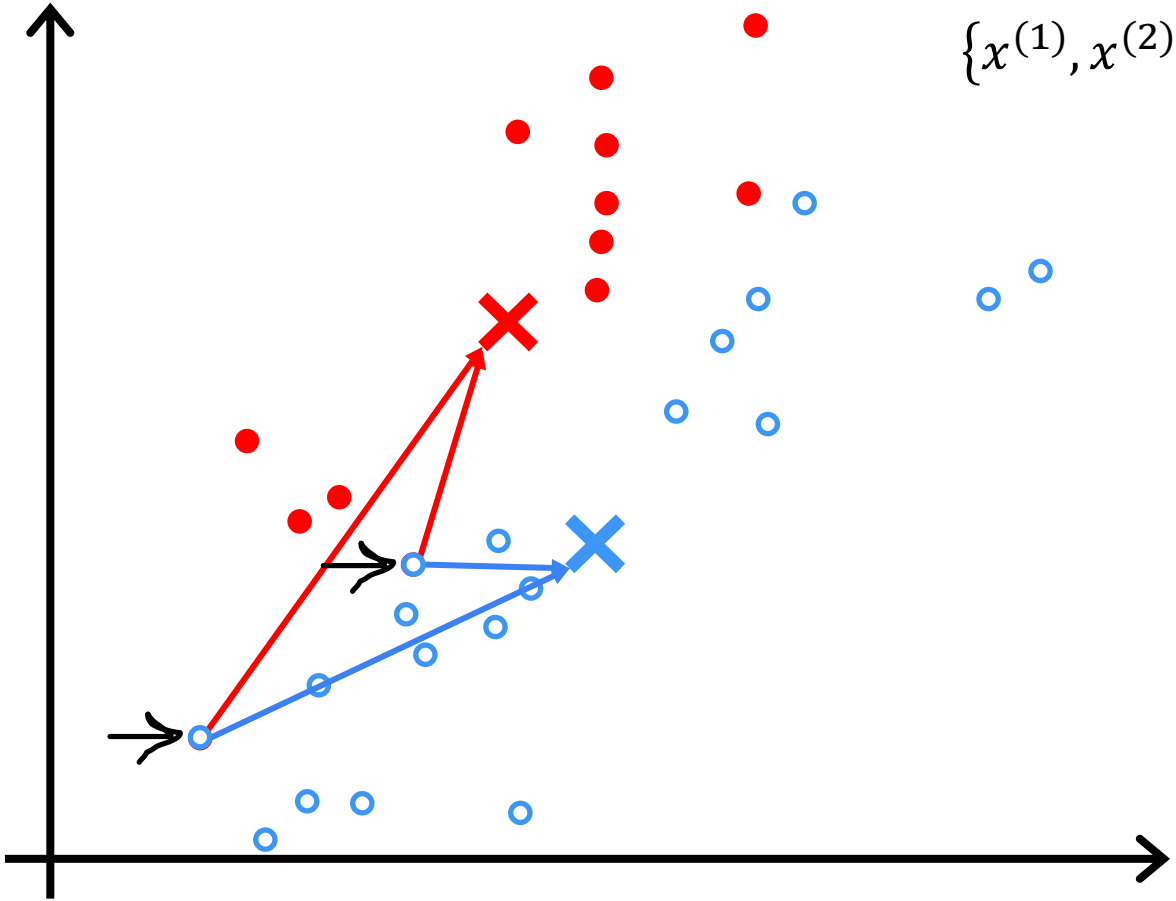


Step 2:
Recompute the centroids

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

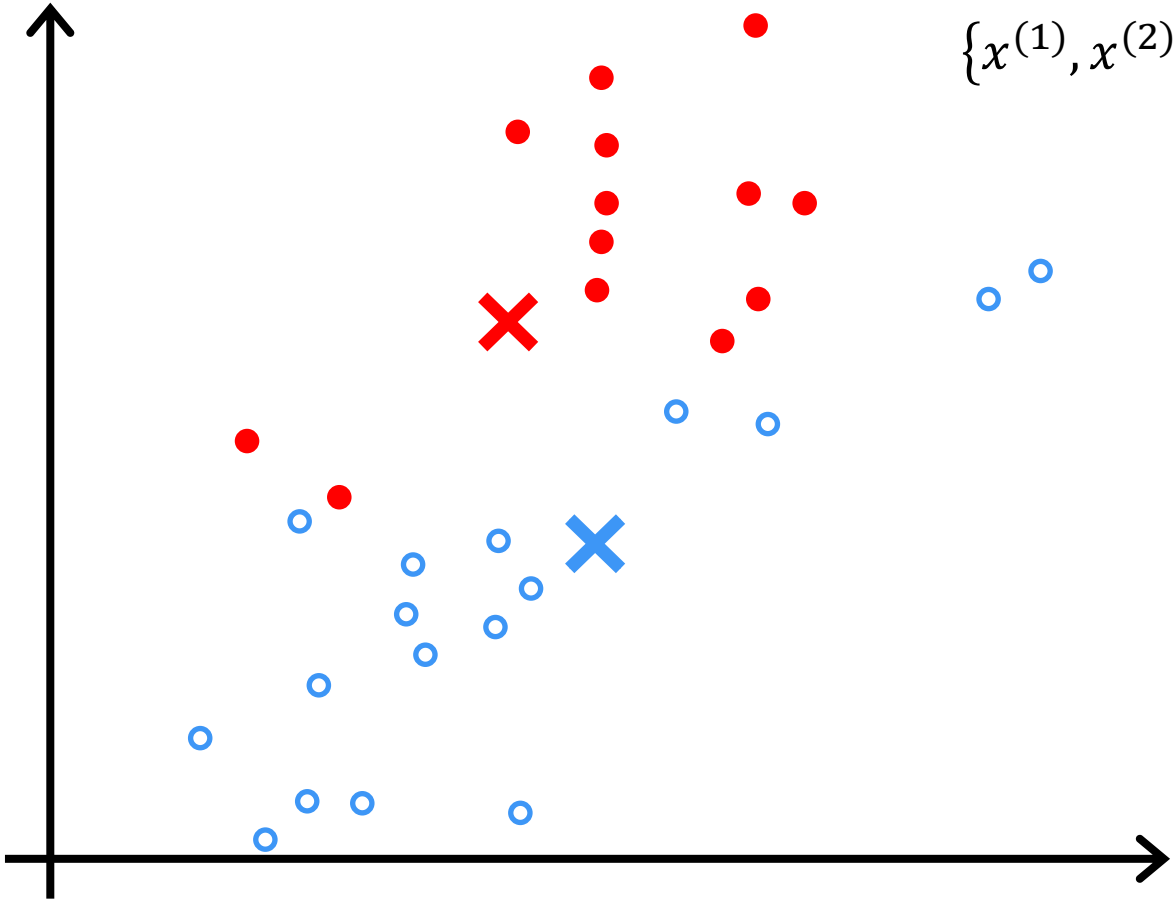


Step 1: Assign each point to its closest centroid



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

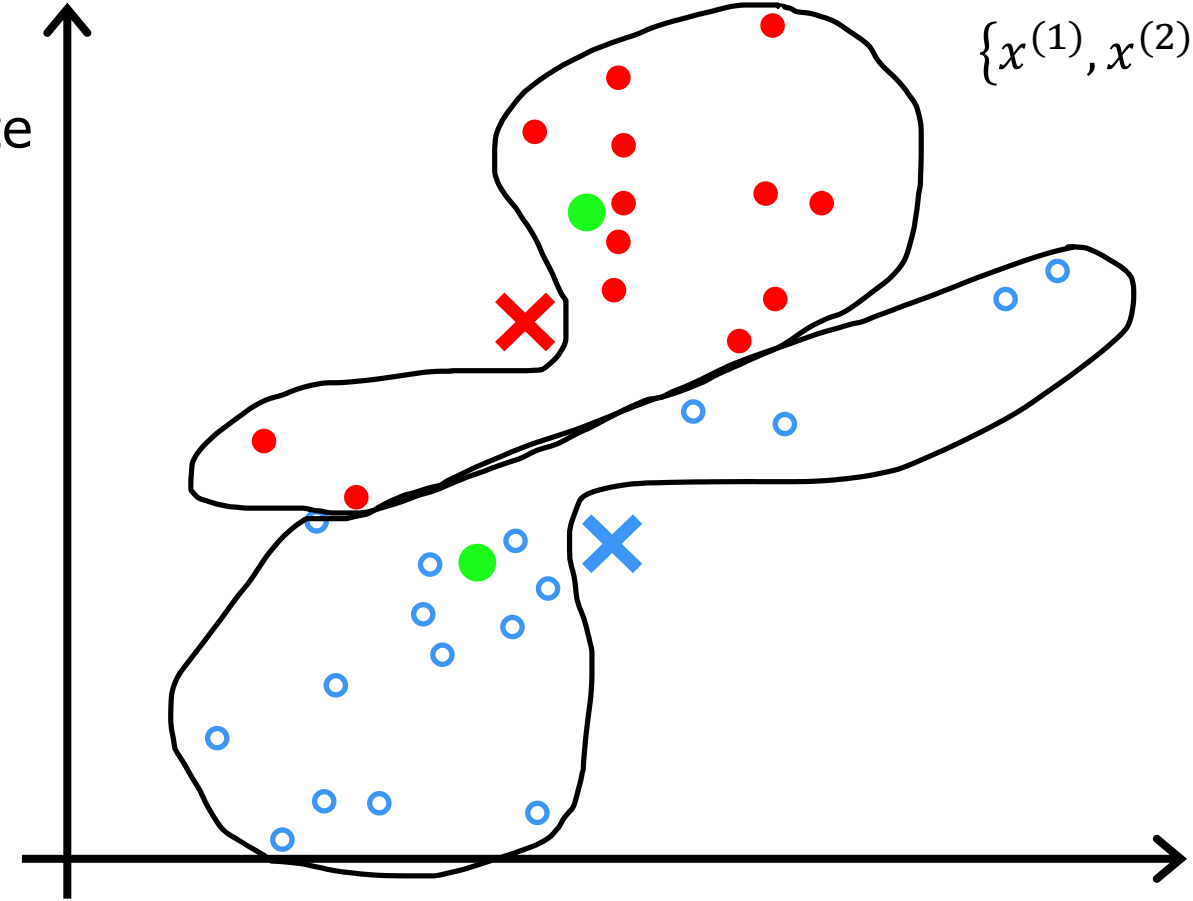
Step 1: Assign each point to its closest centroid



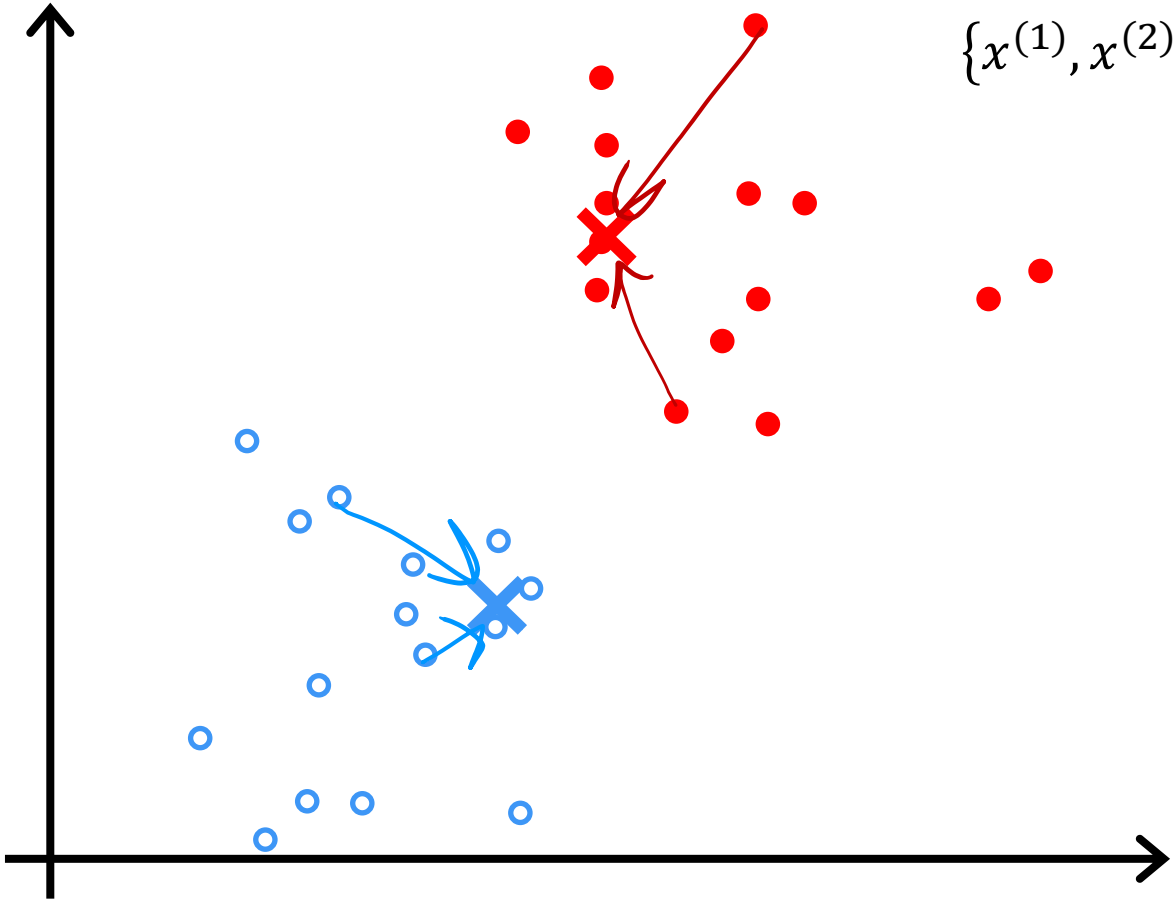
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$

Step 2:
Recompute the centroids

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

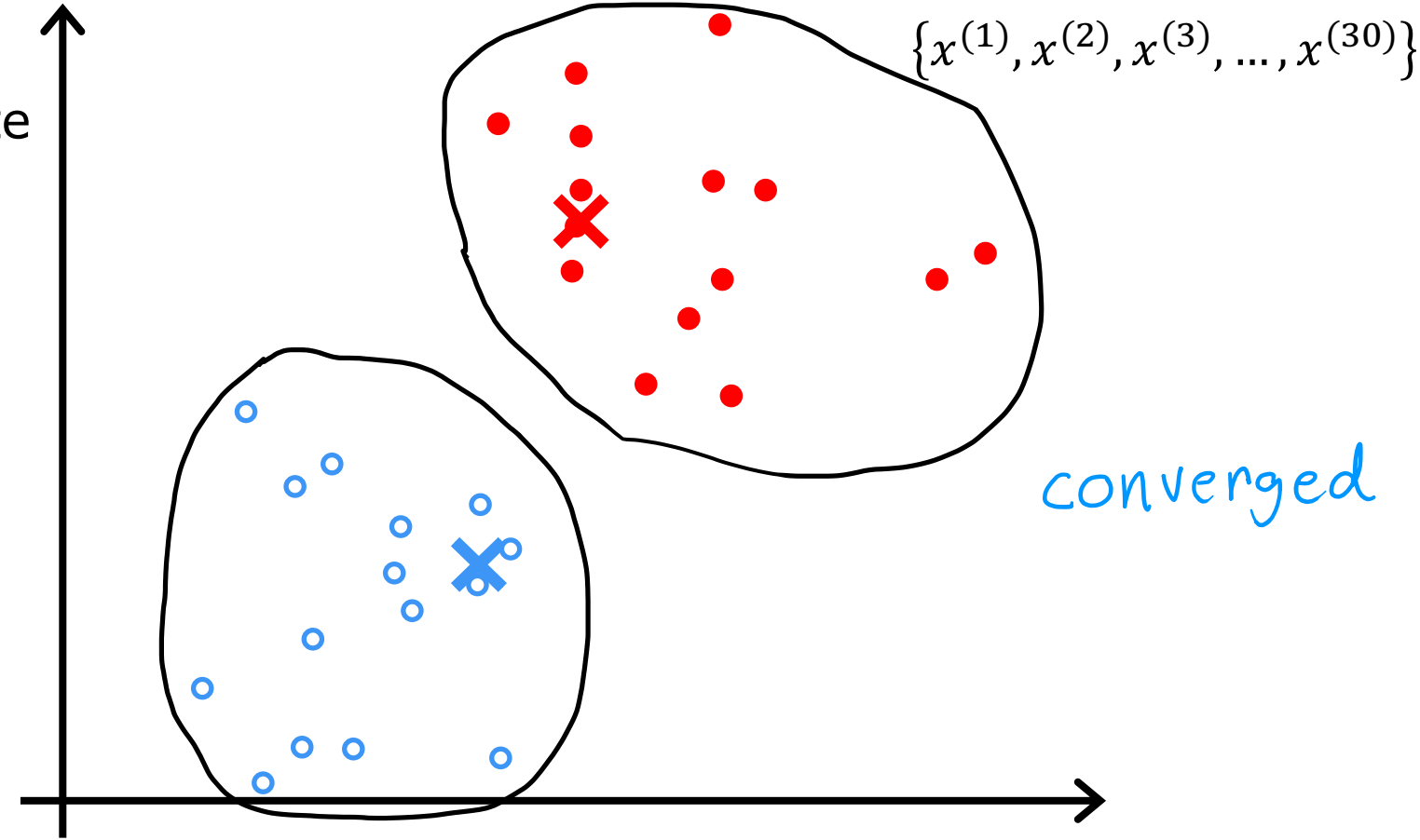


Step 1: Assign each point to its closest centroid



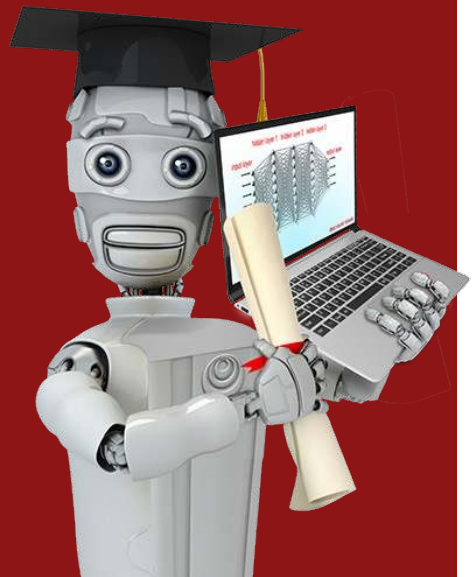
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(30)}\}$

Step 2:
Recompute the centroids



DeepLearning.AI

Stanford
ONLINE



Clustering

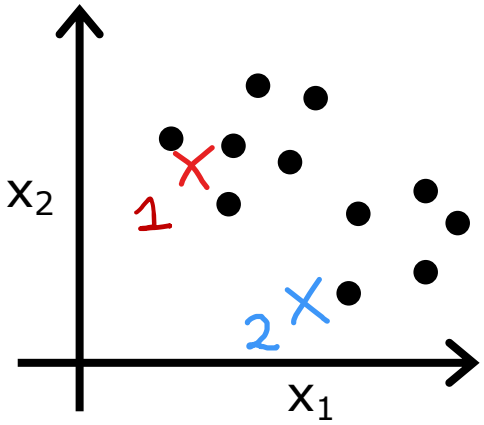
K-means algorithm

K-means algorithm

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

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

$n=2$ $K=2$



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 {

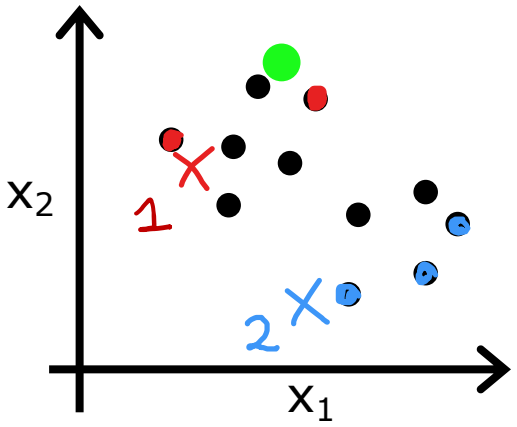
 # Assign points to cluster centroids

for i = 1 to m

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

L2 norm

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



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 $x^{(i)}$

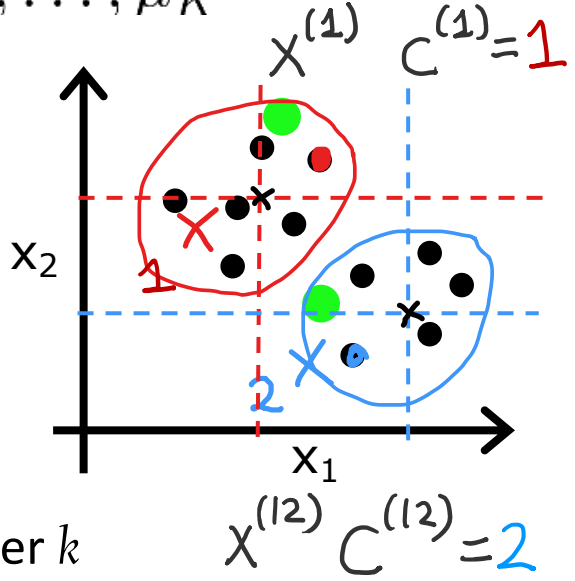
$\min_k \|x^{(i)} - \mu_k\|^2$

 # 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 algorithm

$\mu_1 \mu_2$ $x^{(1)}, x^{(2)}, \ldots, x^{(30)}$

$n=2$ ~~$k=2$~~

$K = K-1$ ✓

Randomly initialize k cluster centroids, $\mu_1, \mu_1, \ldots \mu_k$

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

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

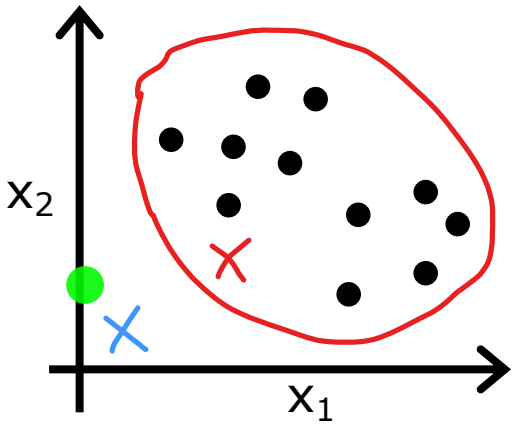
$\min_K \|x^{(i)} - \mu_K\|^2$

 # Move cluster centroids

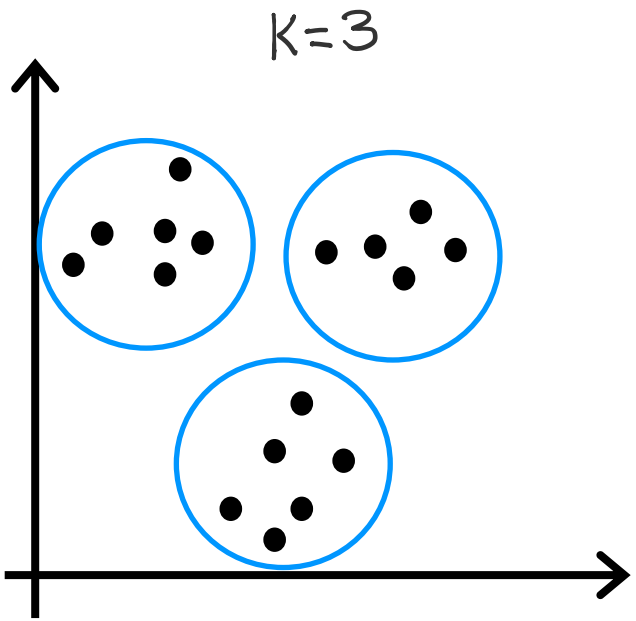
for k = 1 to K

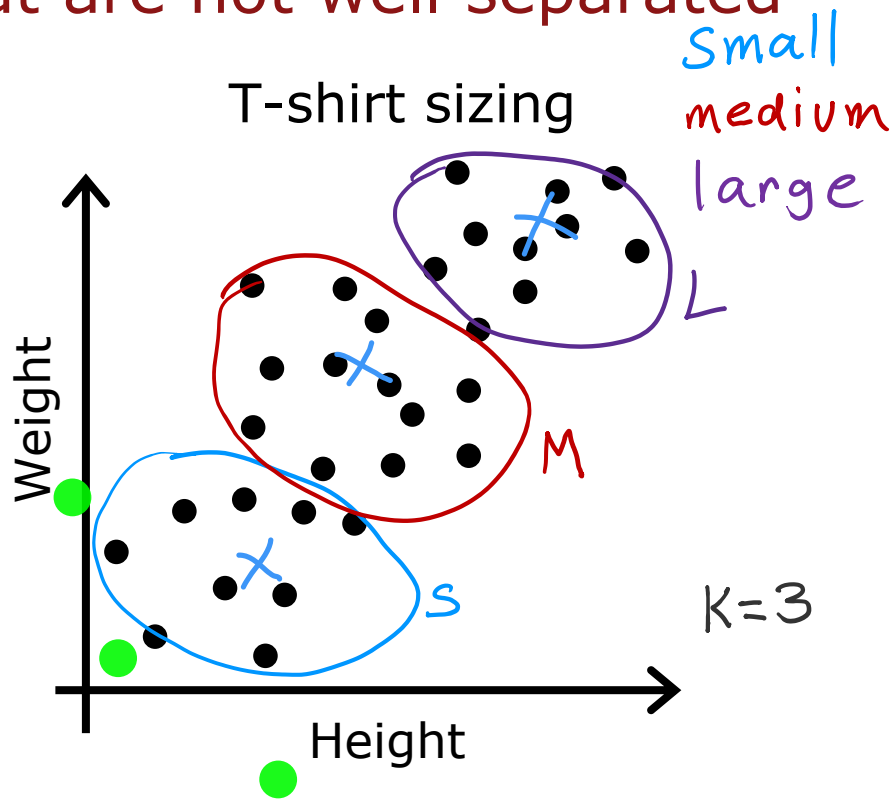
μ_k := average (mean) of points assigned to cluster k

}



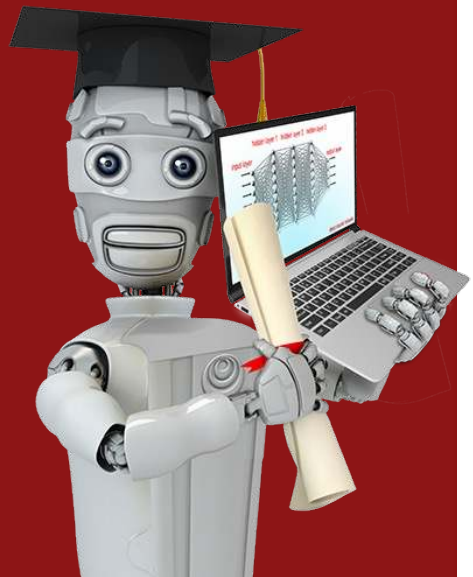
K-means for clusters that are not well separated





DeepLearning.AI

Stanford
ONLINE



Clustering

Optimization objective

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

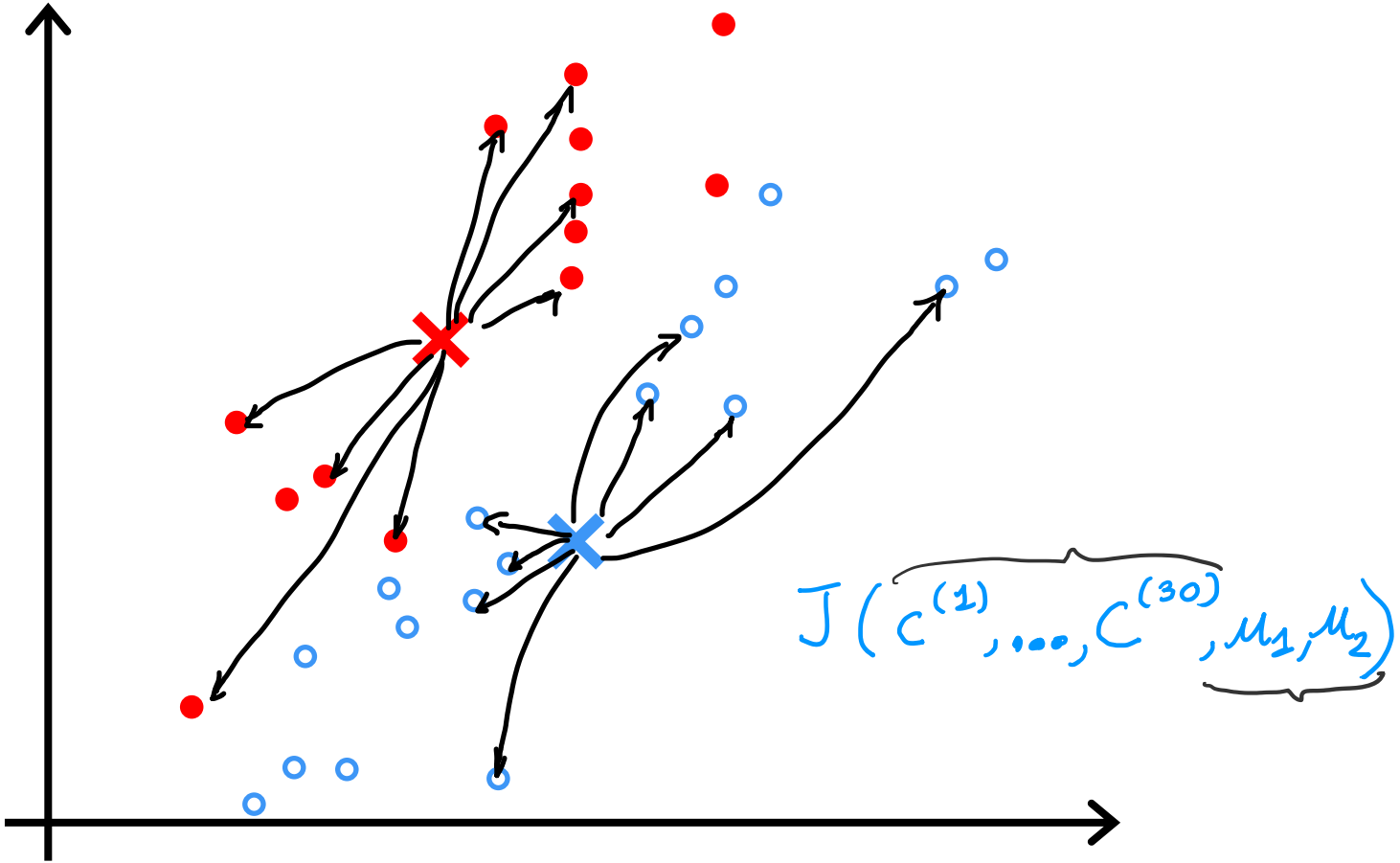
$x^{(10)}$ $c^{(10)}$ $\mu_c^{(10)}$

Cost function

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

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

distortion


$J(c^{(1)}, \ldots, c^{(30)}, \mu_1, \mu_2)$

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:

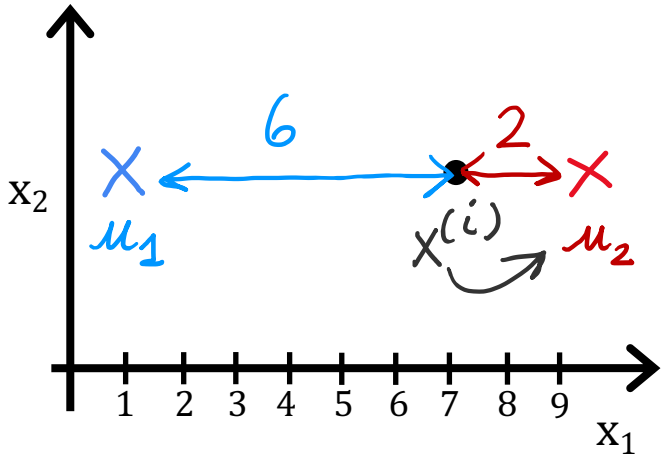
$c^{(i)}$ = index of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

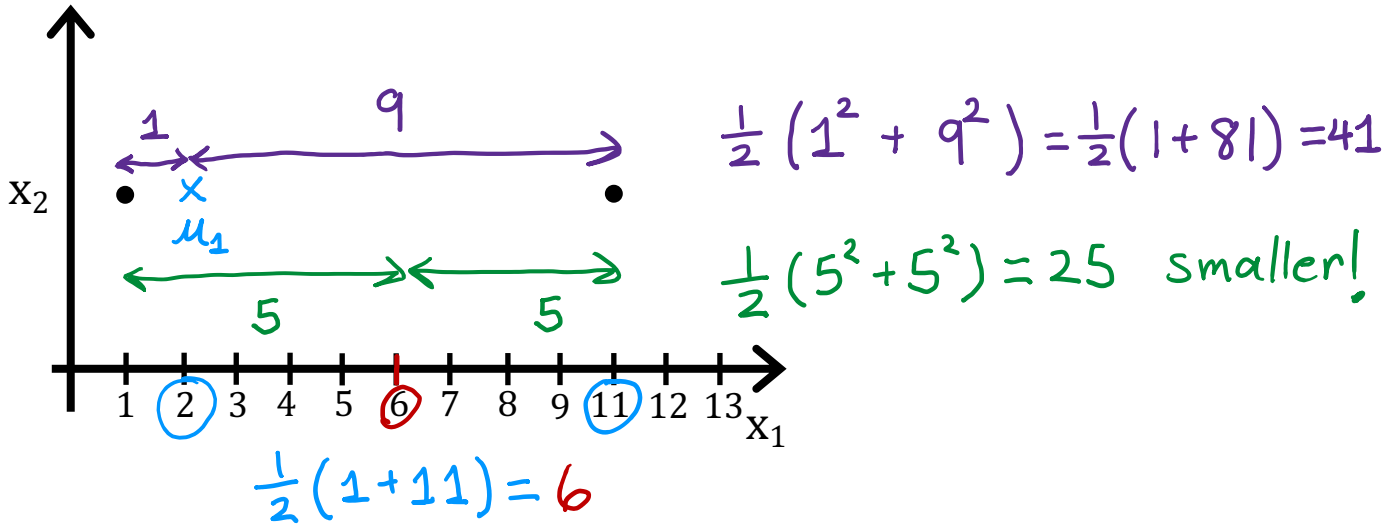
for i = 1 to K:

μ_k = average of points in cluster

}

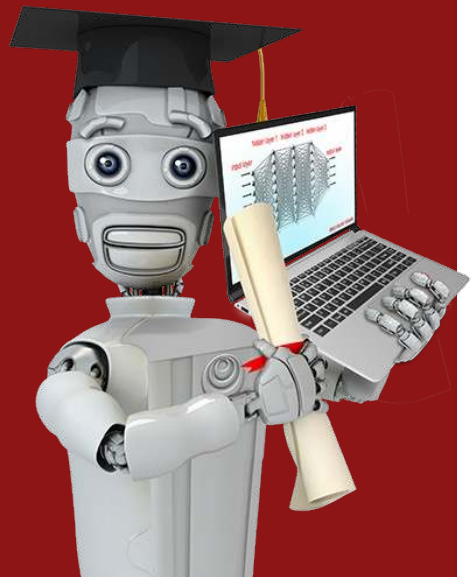


Moving the centroid



DeepLearning.AI

Stanford
ONLINE



Clustering

Initializing K-means

K-means algorithm

Step 0: Randomly initialize K cluster centroids $\mu_1, \mu_1, \ldots, \mu_k$

Repeat {

Step 1: Assign points to cluster centroids

Step 2: Move cluster centroids

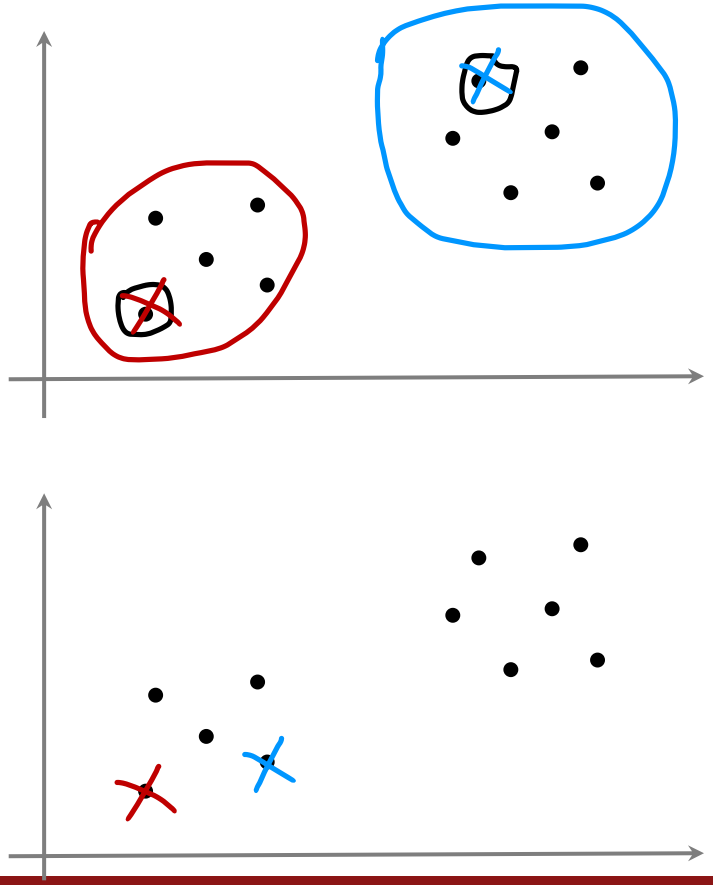
}

Random initialization

Choose $K < m$

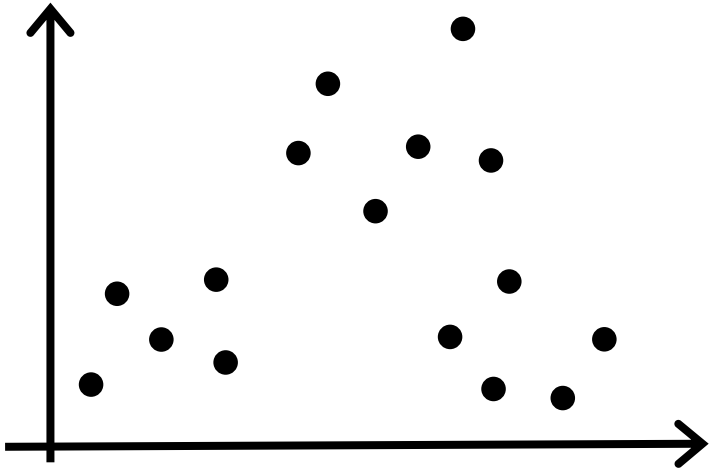
Randomly pick K training examples.

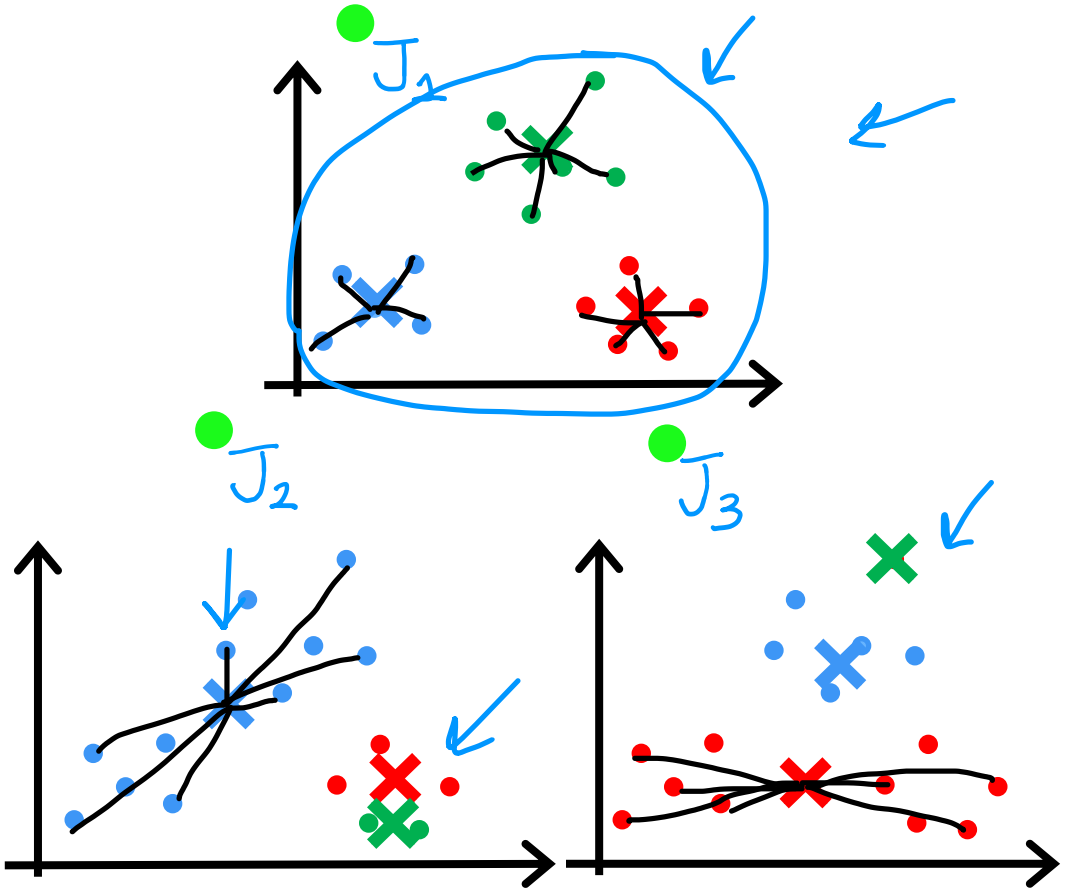
Set μ_1, μ_1,…, μ_k equal to these K examples.



K=3

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





Random initialization

better than running K-means once

For $i = 1$ to 100 {

50-1000

pick K random examples
set as cluster centroids

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

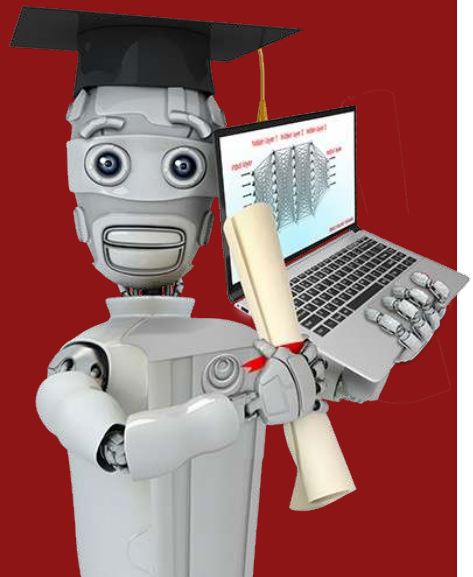
distortion
(cost function)

}

Pick set of clusters that gave lowest cost J

DeepLearning.AI

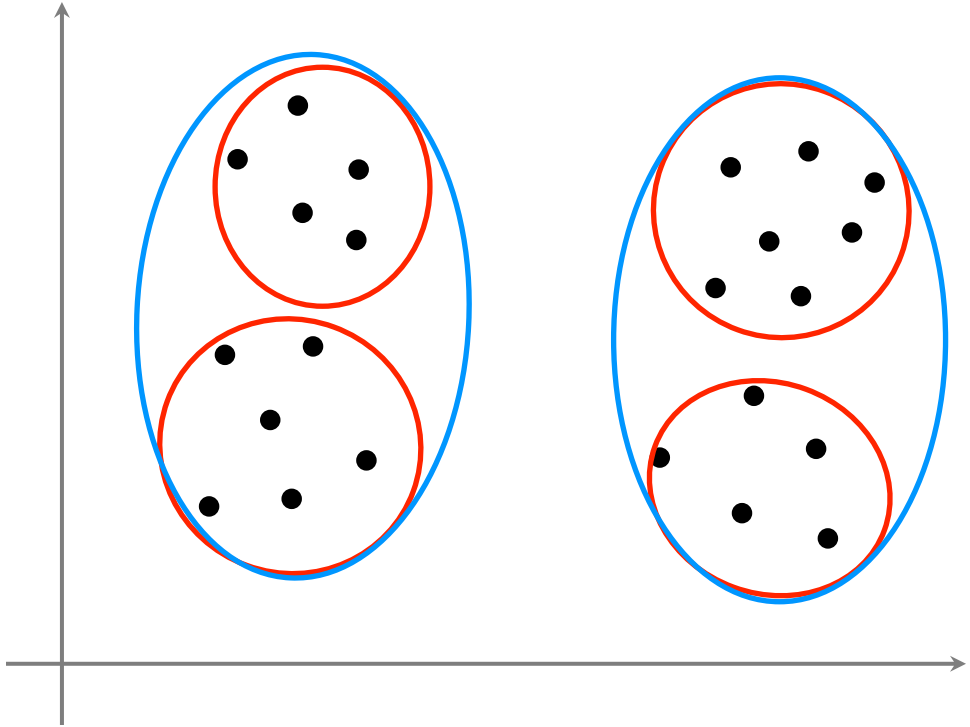
Stanford
ONLINE



Clustering

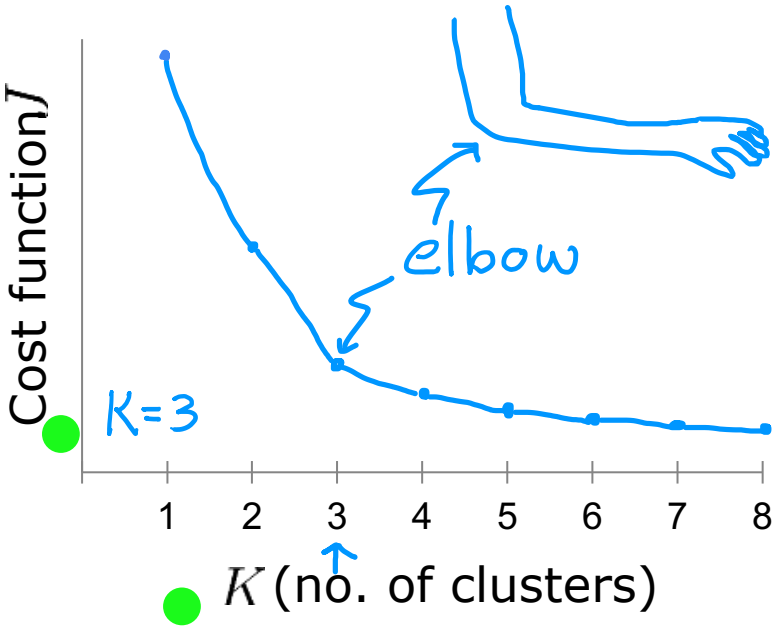
Choosing the Number of Clusters

What is the right value of K?



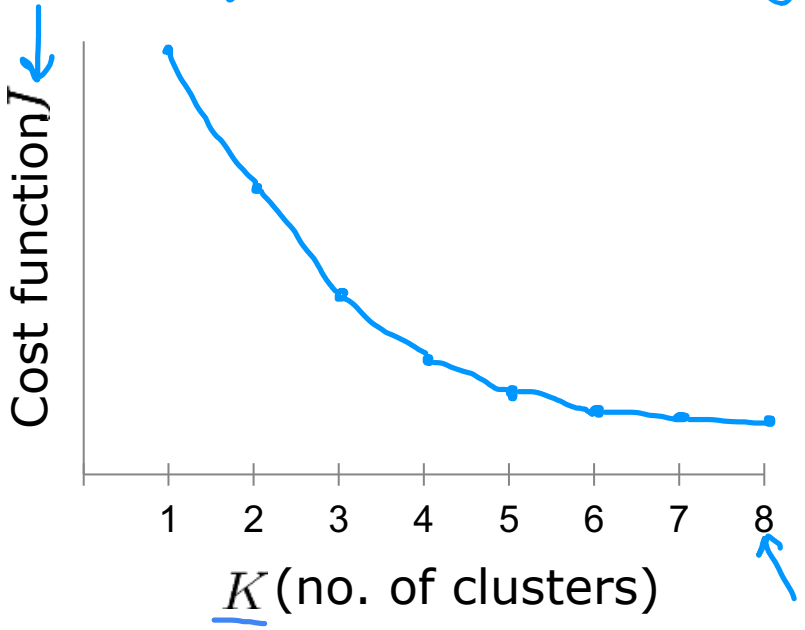
Choosing the value of K

Elbow method



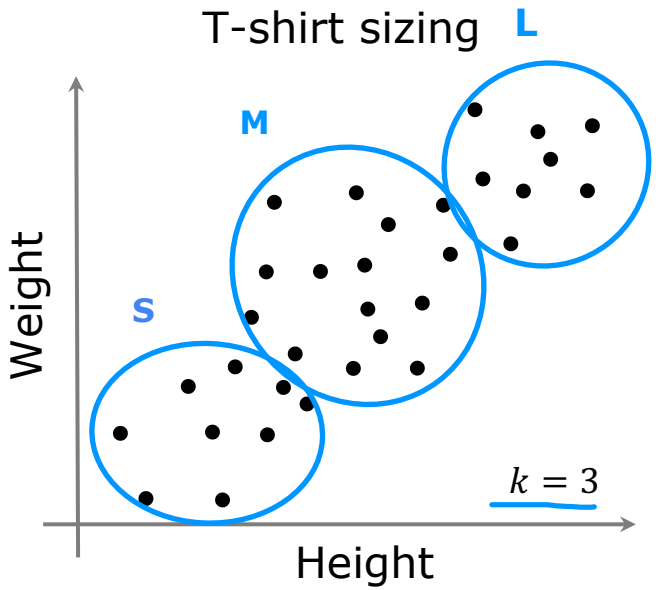
the right "K" is often ambiguous

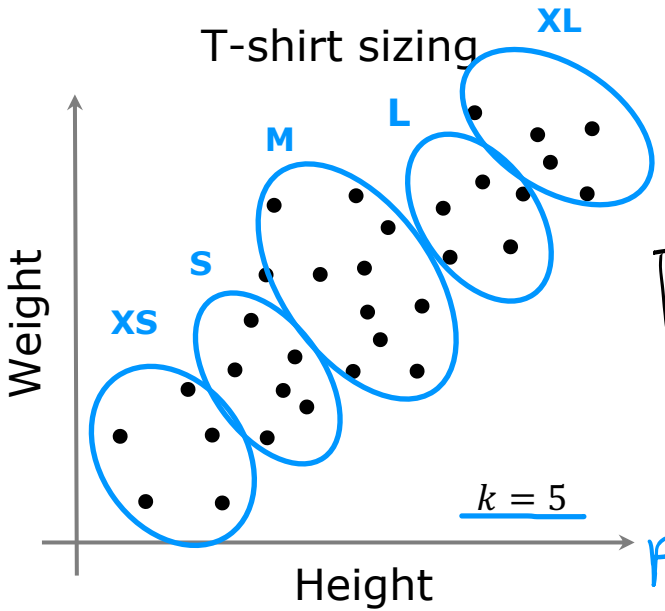
Don't choose K just to minimize cost J

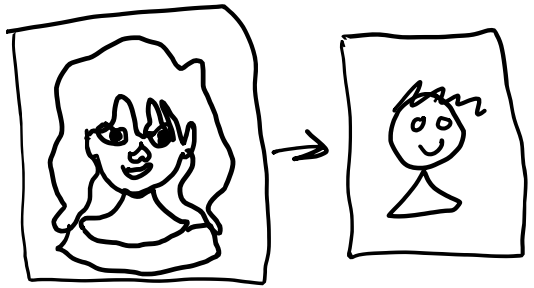


Choosing the value of K

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



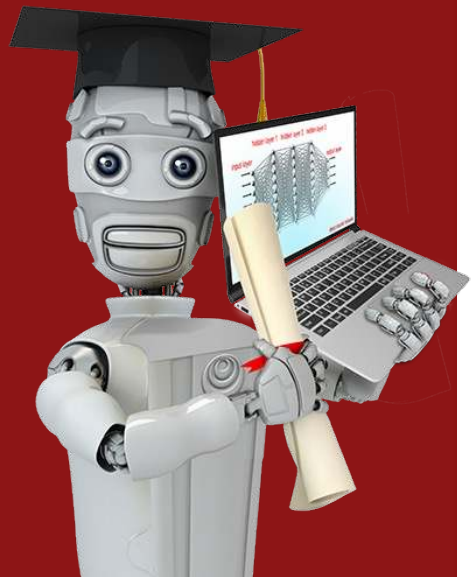




practice lab:
image compression!

DeepLearning.AI

Stanford
ONLINE



Anomaly Detection

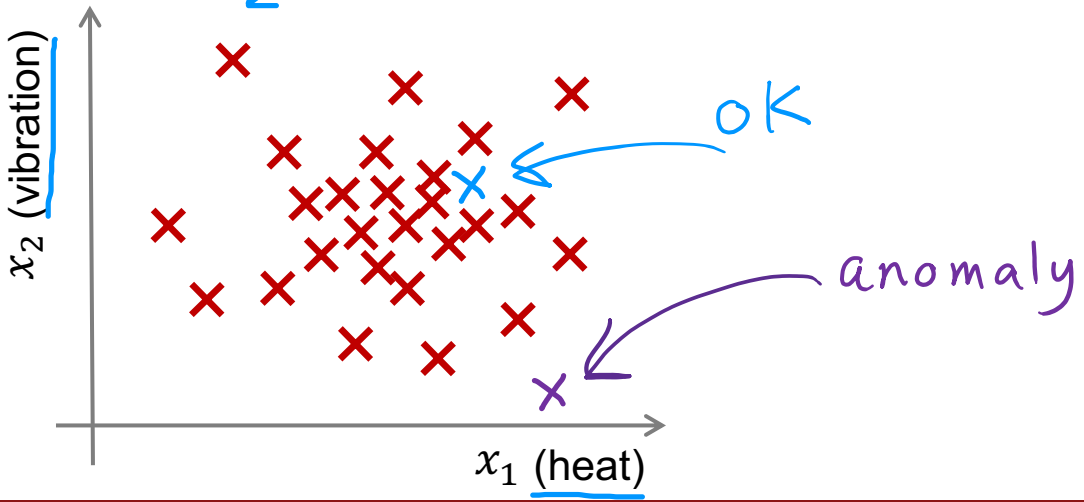
Finding unusual events

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}



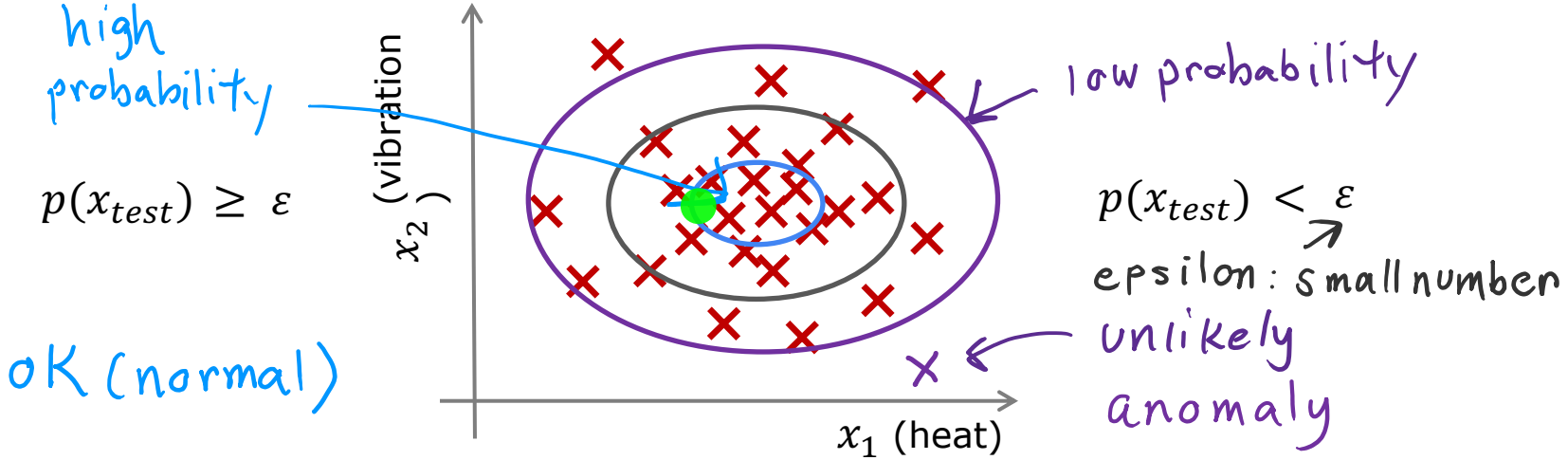
Density estimation

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

probability of x being seen in dataset

Model $p(x)$

Is x_{test} anomalous?



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

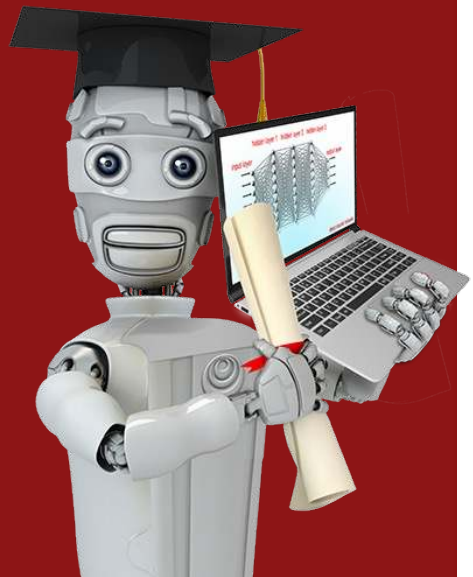
Monitoring computers in a data center:

$x^{(i)}$= 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.

DeepLearning.AI

Stanford
ONLINE



Anomaly Detection

Gaussian (Normal) Distribution

Gaussian (Normal) distribution

σ standard deviation

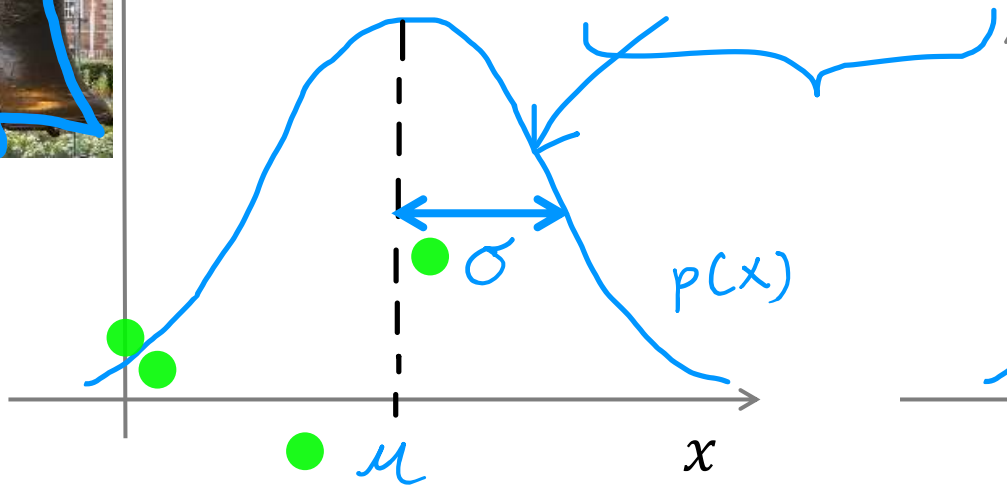
σ^2 variance

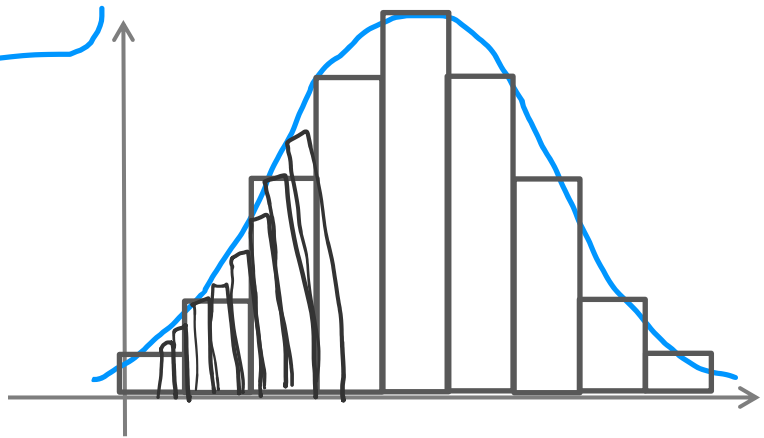
Say x is a number.
Probability of x is determined by a Gaussian with mean μ, variance σ^2.

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

$\pi = 3.14$

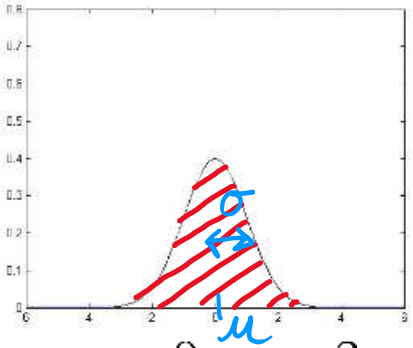




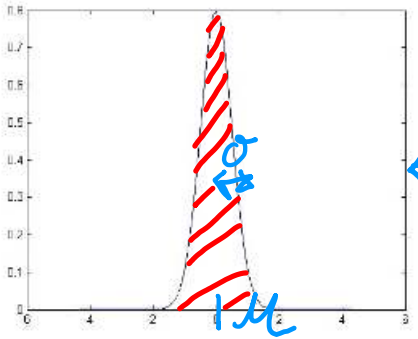


Gaussian distribution example

$\mu = 0, \sigma = 1$

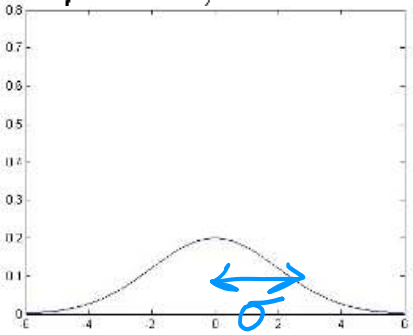


$\mu = 0, \sigma = 0.5$



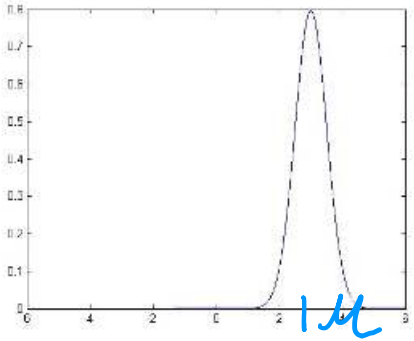
$\sigma^2 = 0.25$

$\mu = 0, \sigma = 2$



$\sigma^2 = 4$

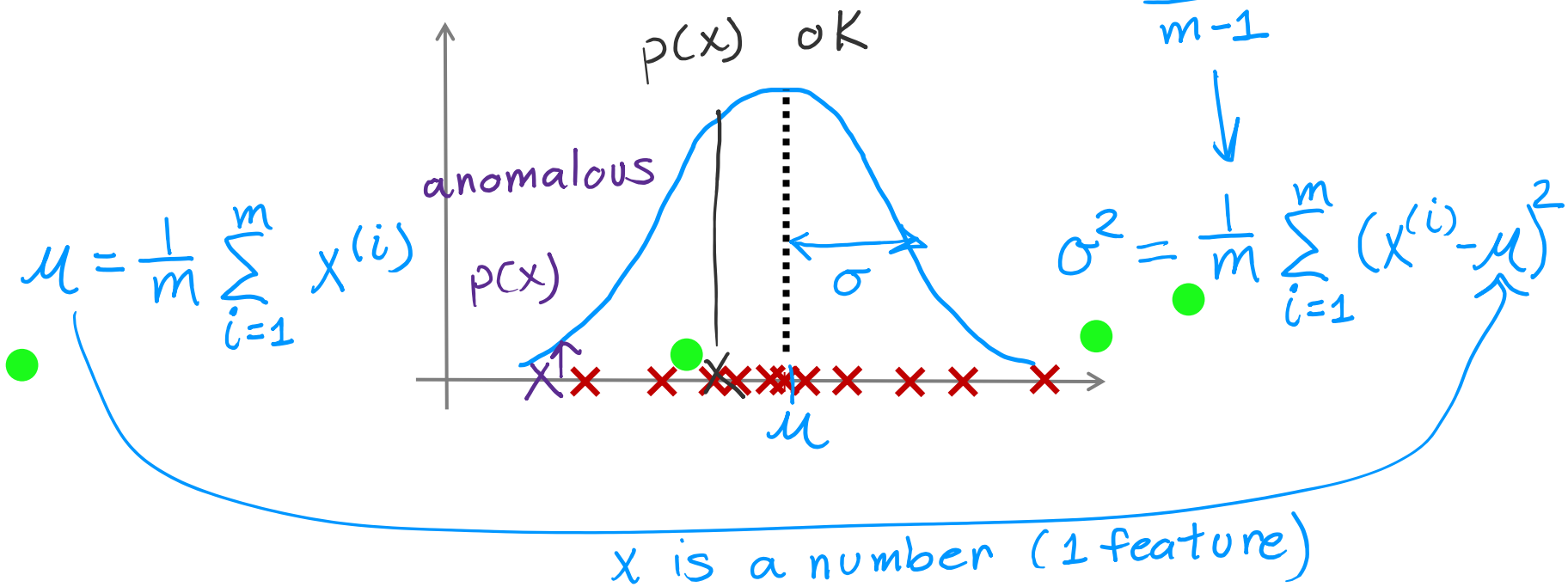
$\mu = 3, \sigma = 0.5$



Parameter estimation

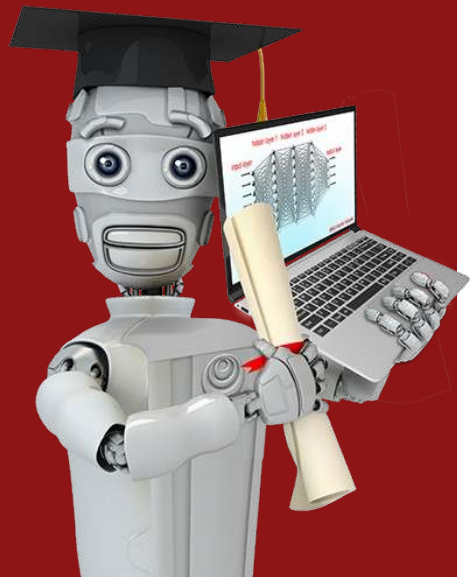
Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

maximum likelihood for μ, σ



DeepLearning.AI

Stanford
ONLINE



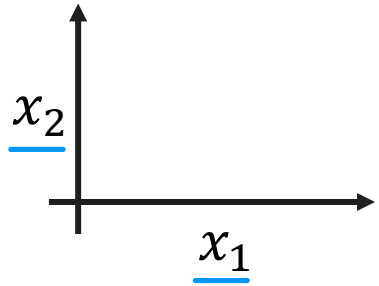
Anomaly Detection

Algorithm

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"

$p(x_1 = \text{high temp}) = 1/10$

$p(x_2 = \text{high vibra}) = 1/20$

$p(x_1, x_1) = p(x_1) * p(x_2)$

$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$

Anomaly detection algorithm

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

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

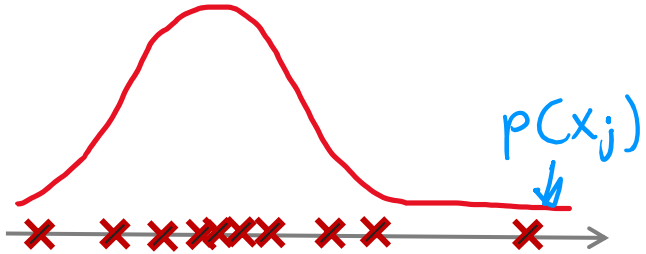
Vectorized formula

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

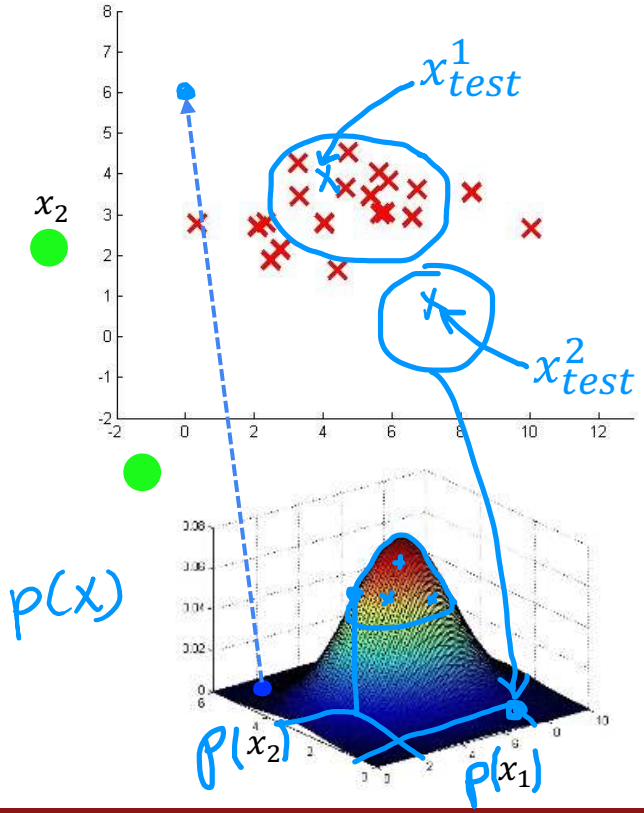
3. Given new example x, compute $p(x)$:

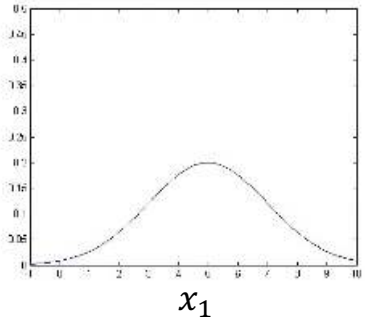
$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$$

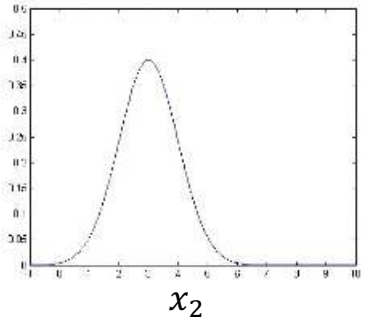
Anomaly if $p(x) < \varepsilon$



Anomaly detection example







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

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

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

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

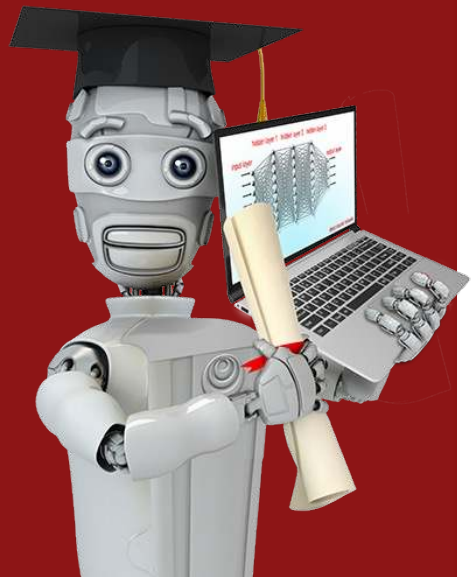
$\varepsilon = 0.02$

$p\left(x_{test}^{(1)}\right) = 0.0426$ → "ok"

$p\left(x_{test}^{(2)}\right) = 0.0021$ → 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

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 $y = 0$

Aircraft engines monitoring example

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

2 to 50

y=1

2

y=0

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

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

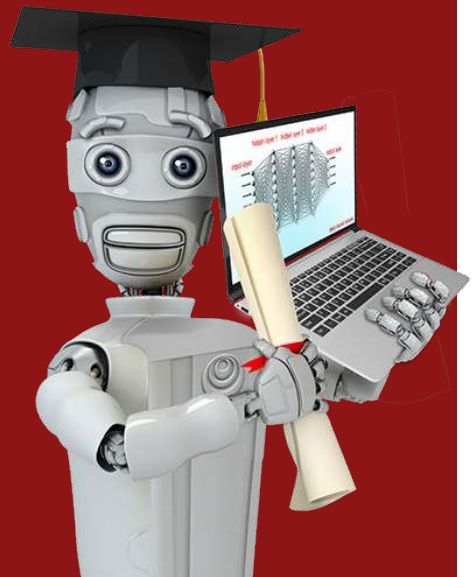
Possible evaluation metrics:

- 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



Anomaly Detection

Anomaly detection vs. supervised learning

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

- Email spam classification

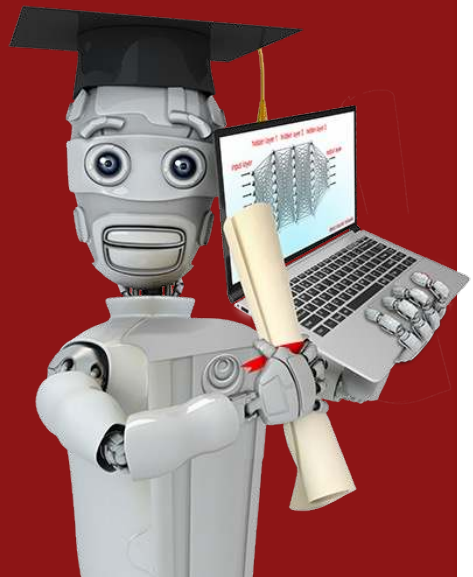
- Manufacturing:
Finding known, previously seen defects
(e.g. scratches on smartphones, $y = 1$)

- Weather prediction (sunny/rainy/etc.)

- Diseases classification

DeepLearning.AI

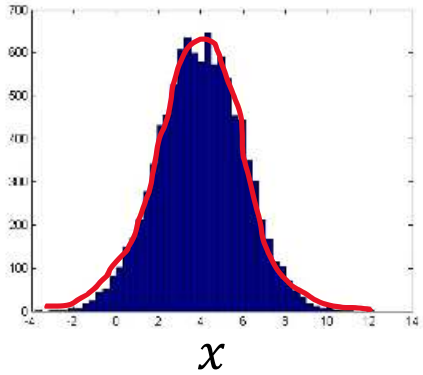
Stanford
ONLINE



Anomaly Detection

Choosing what features to use

Non-gaussian features



x

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

`plt.hist(x)`

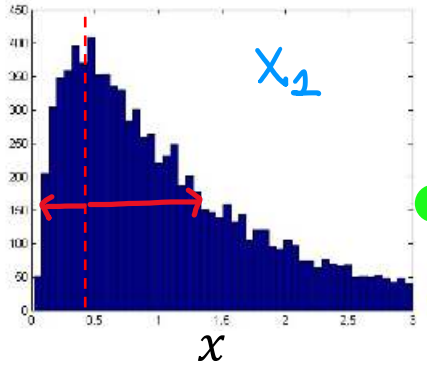
$x_1 \leftarrow \log(x_1)$

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

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

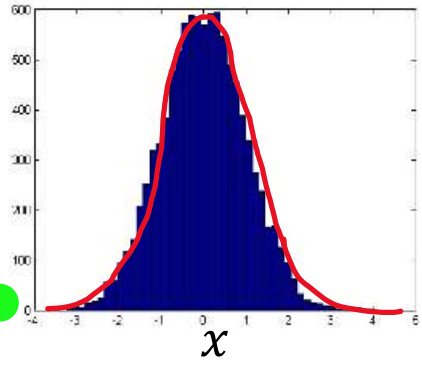
$x_4 \leftarrow x_4^{1/3}$



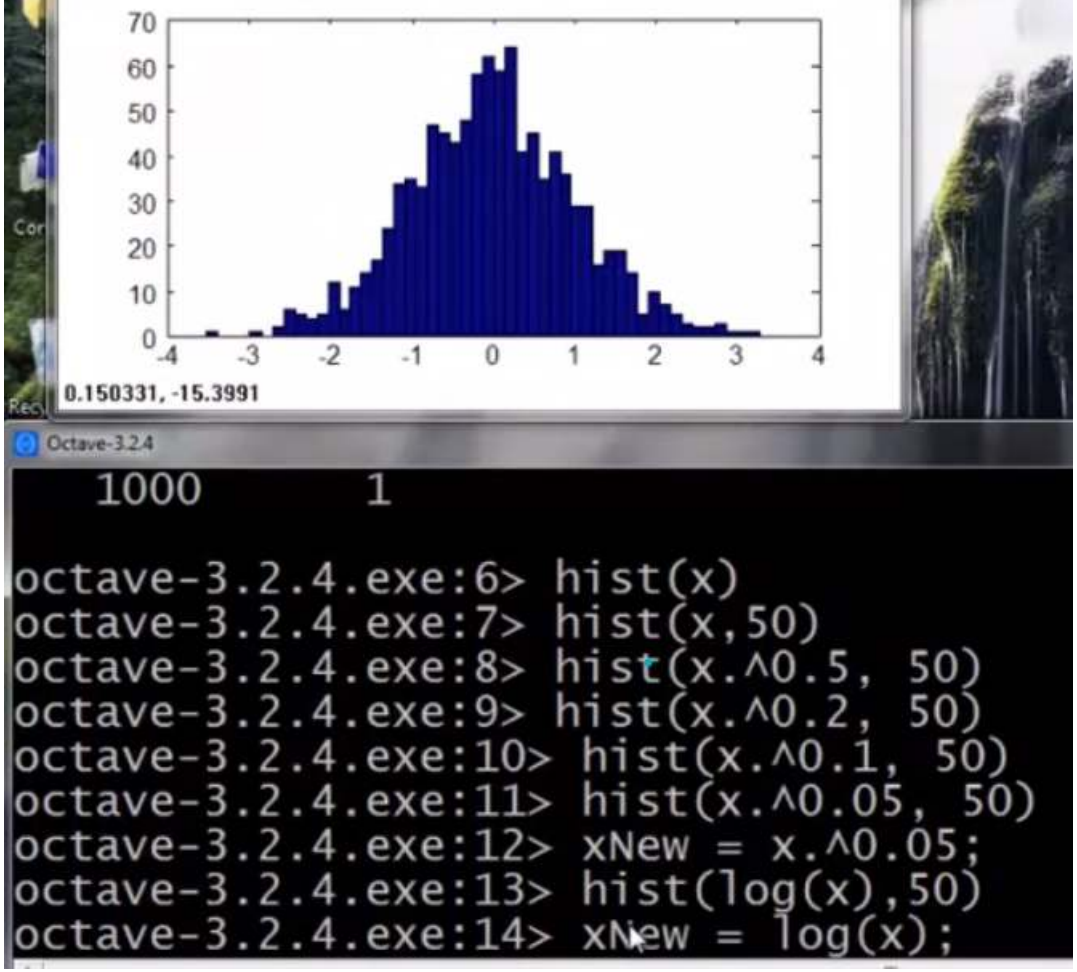


x

`np.log(x)`



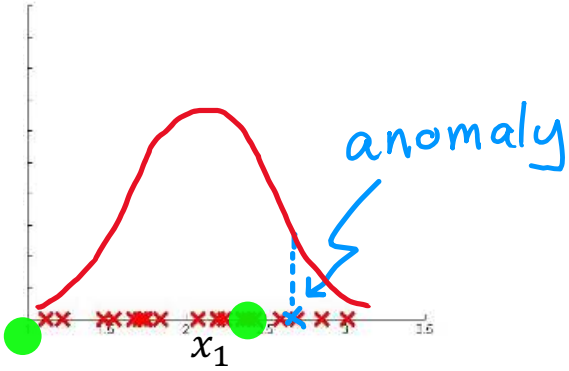
x

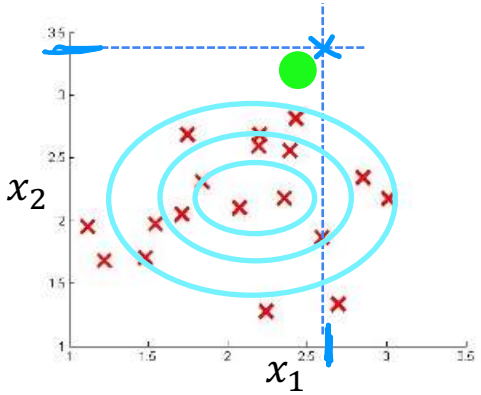

70
60
50
40
30
20
10
0
-4
-3
-2
-1
0
1
2
3
4
0.150331, -15.3991
Octave-3.2.4
1000 1
octave-3.2.4.exe:6> hist(x)
octave-3.2.4.exe:7> hist(x,50)
octave-3.2.4.exe:8> hist(x.^0.5, 50)
octave-3.2.4.exe:9> hist(x.^0.2, 50)
octave-3.2.4.exe:10> hist(x.^0.1, 50)
octave-3.2.4.exe:11> hist(x.^0.05, 50)
octave-3.2.4.exe:12> 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)





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

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

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

Deciding feature choice based on $p(x)$

Large for normal examples;

Becomes small for anomaly in the cross validation set