

Copyright Notice

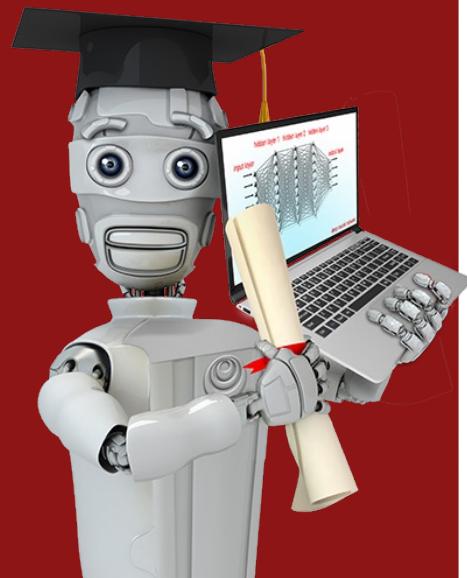
These slides are distributed under the Creative Commons License.

[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

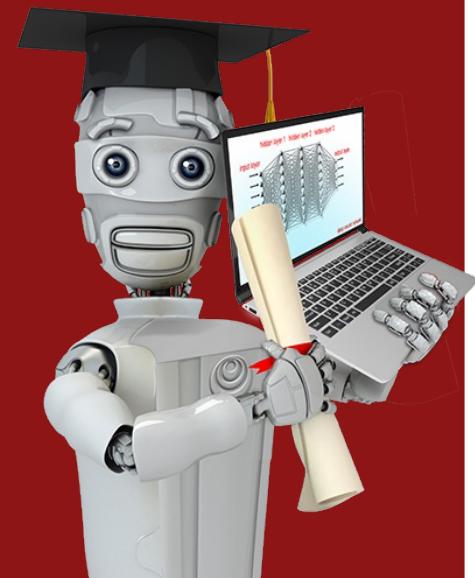


Unsupervised learning, recommender systems and reinforcement learning

Welcome!

Beyond Supervised Learning

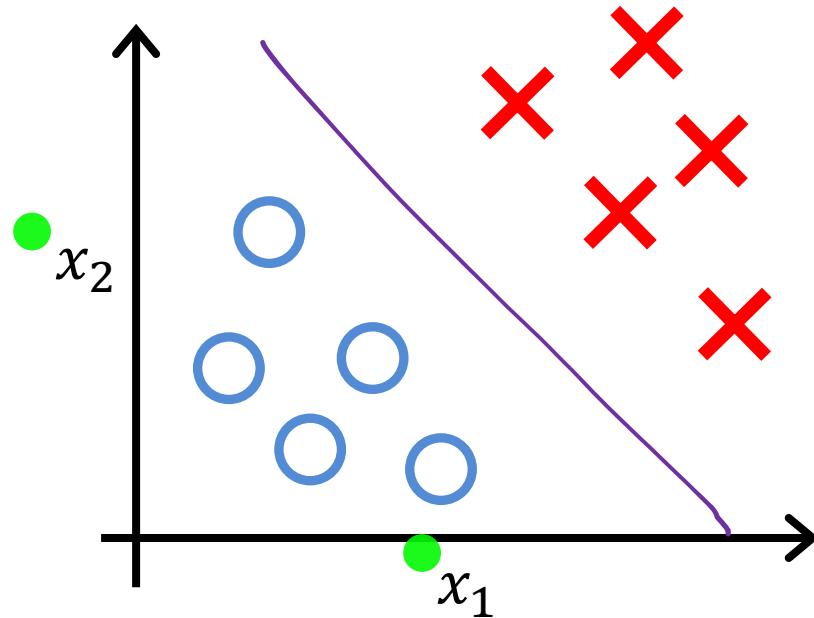
- Unsupervised Learning
 - Clustering
 - Anomaly detection
- Recommender Systems
- Reinforcement Learning



Clustering

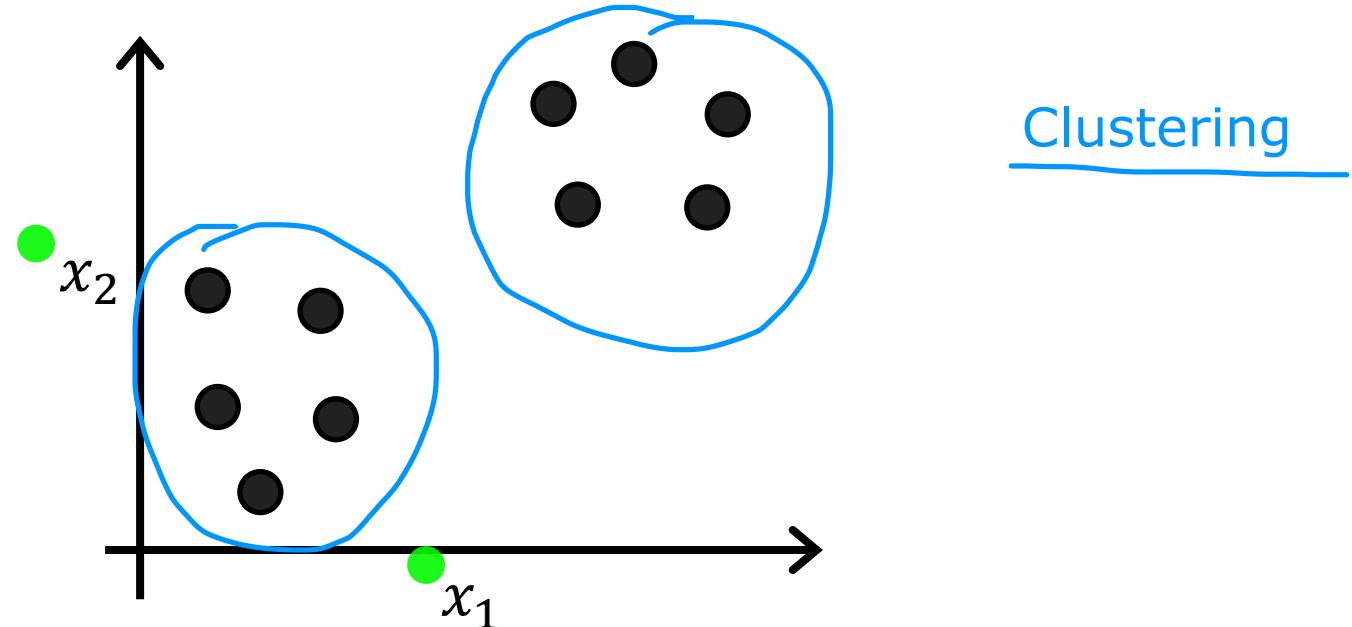
What is clustering?

Supervised learning



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

Unsupervised learning



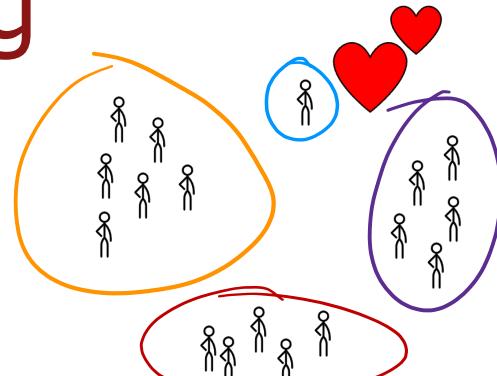
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$

Applications of clustering

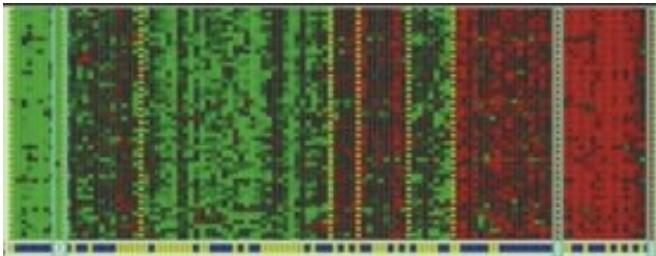


Grouping similar news

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



Market segmentation



DNA analysis

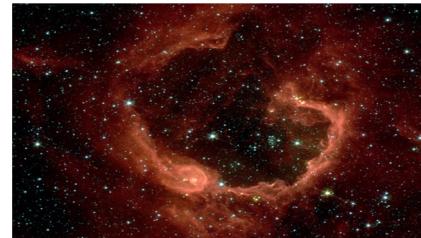
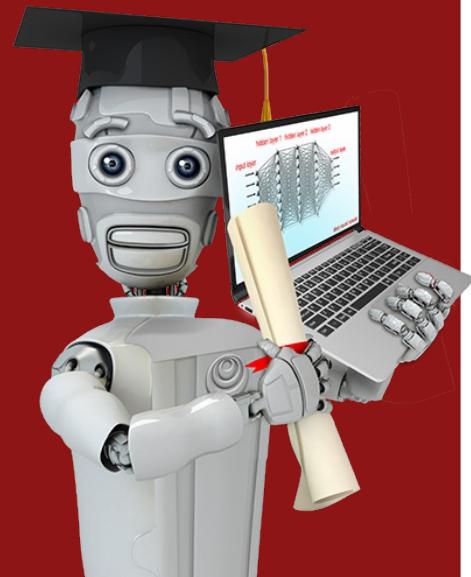


Image credit: NASA/JPL-Caltech/E. Churchwell (Univ. of Wisconsin, Madison)

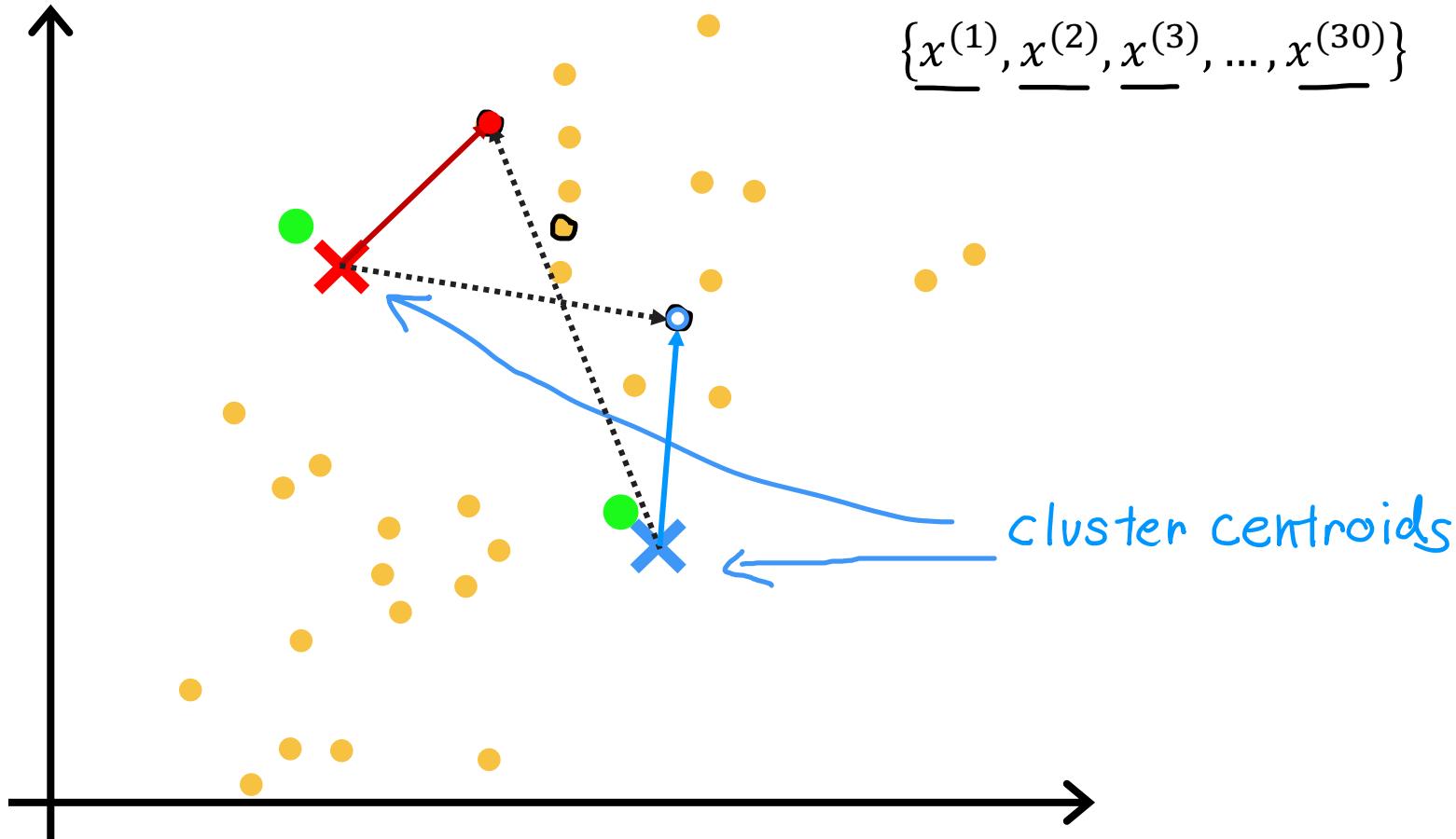
Astronomical data analysis



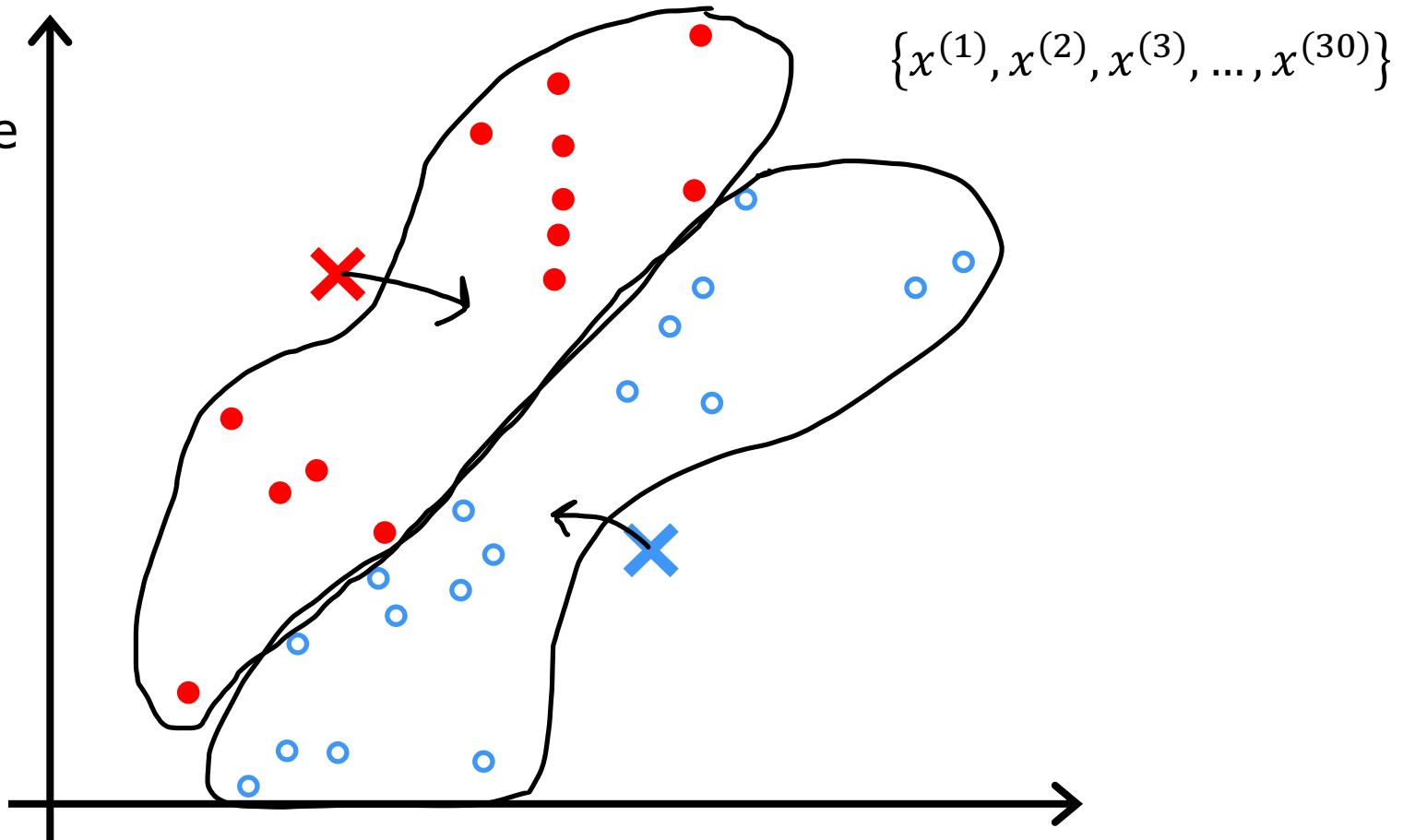
Clustering

K-means intuition

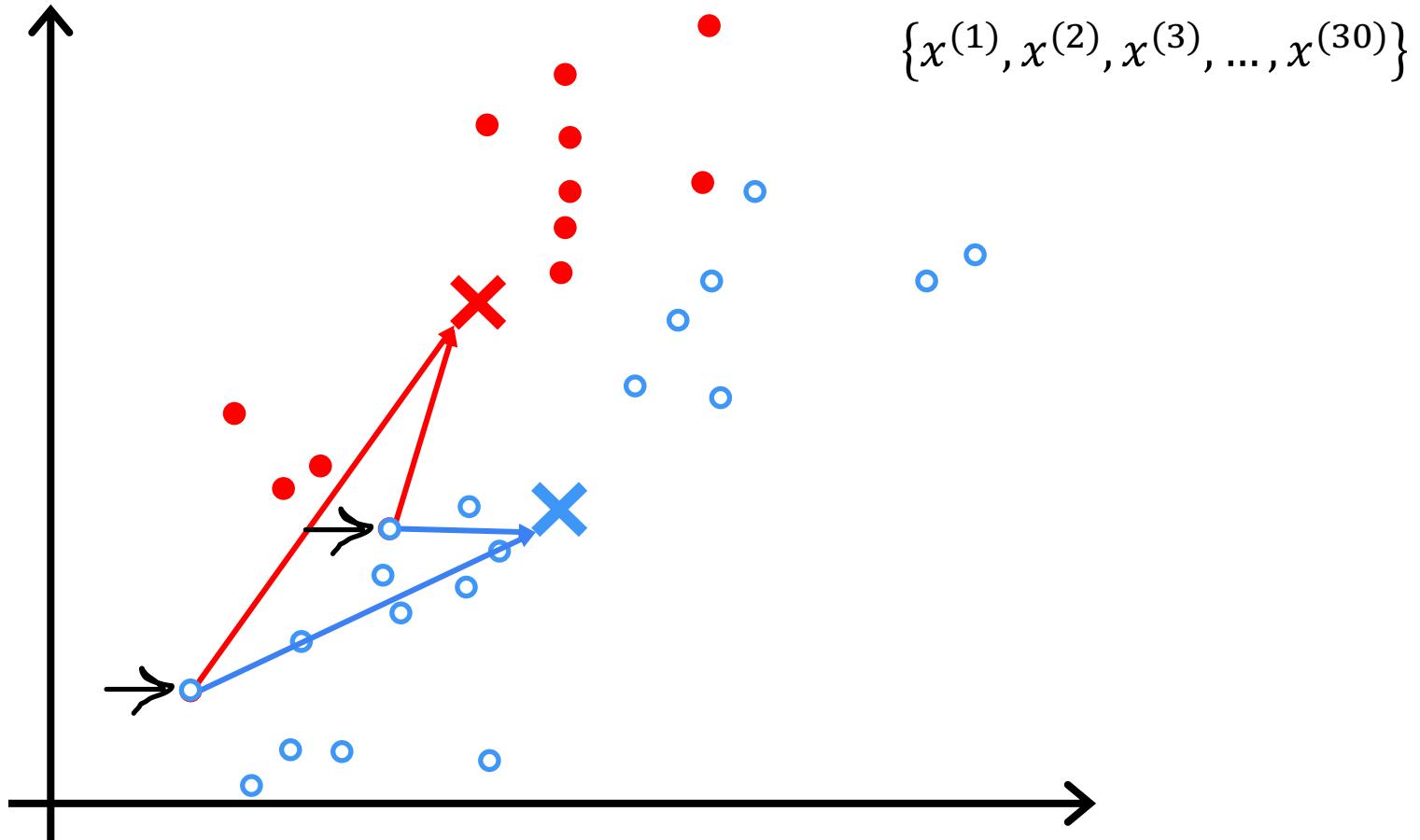
Step 1:
Assign
each point
to its
closest
centroid



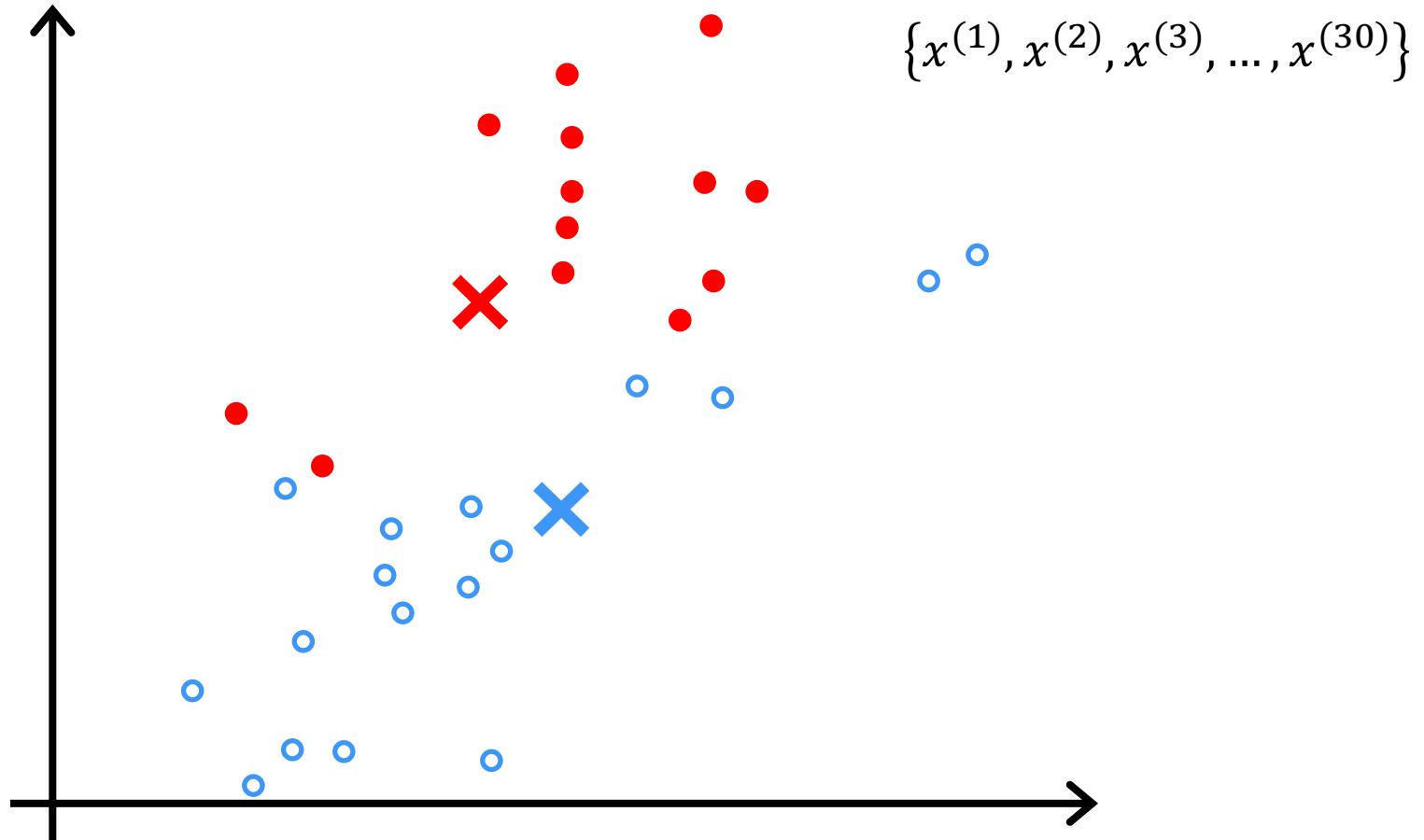
Step 2:
Recompute
the
centroids



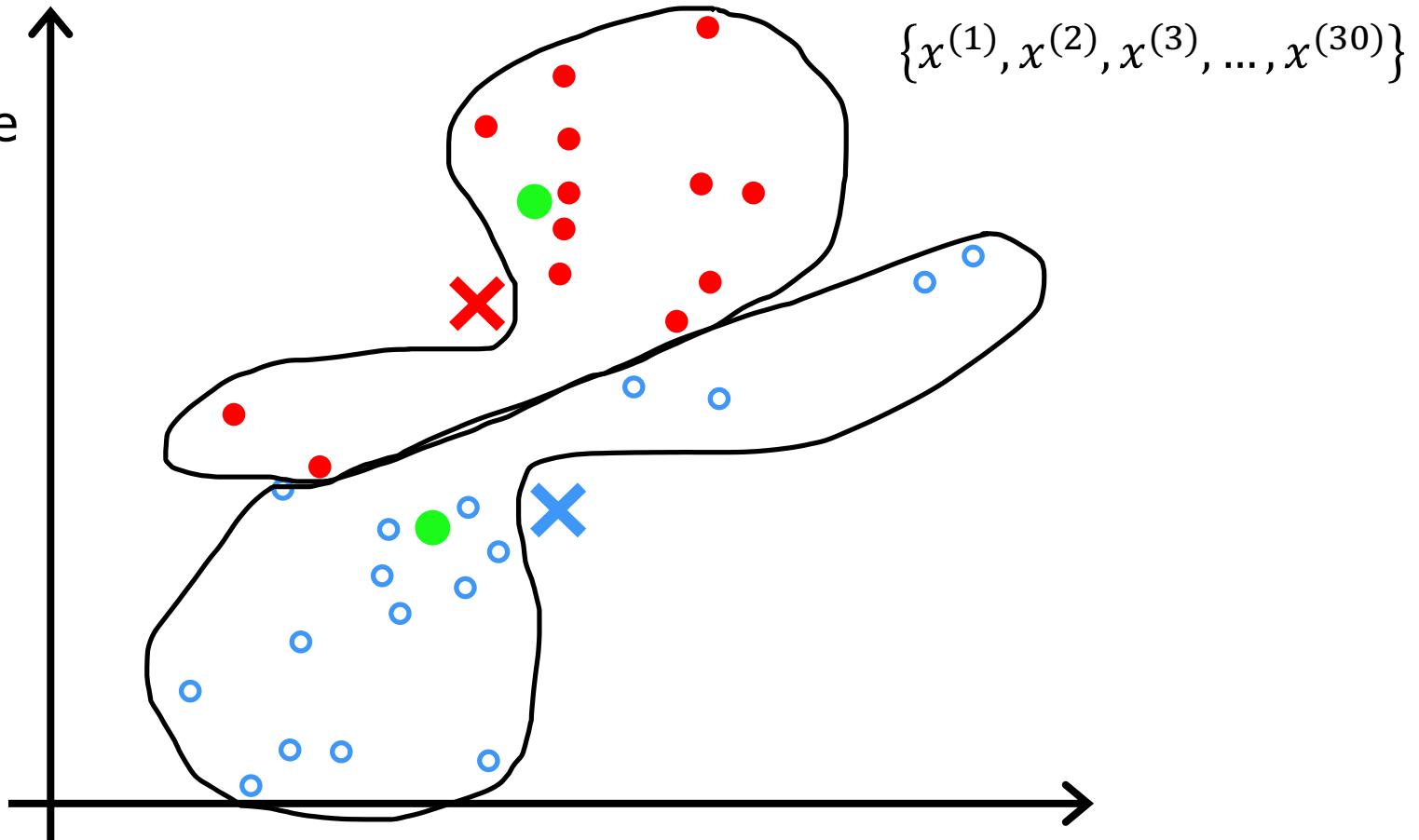
Step 1:
Assign
each point
to its
closest
centroid



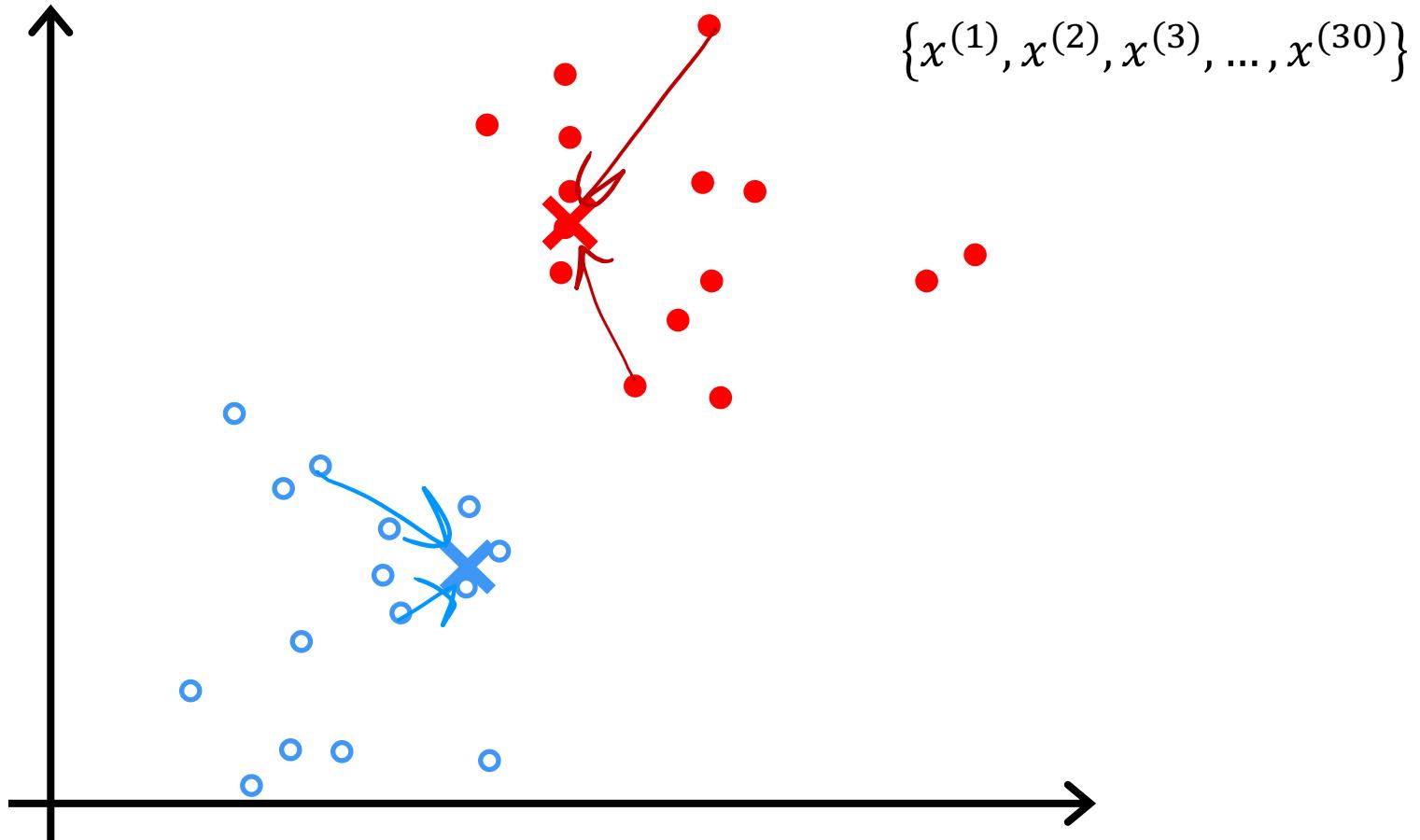
Step 1:
Assign
each point
to its
closest
centroid



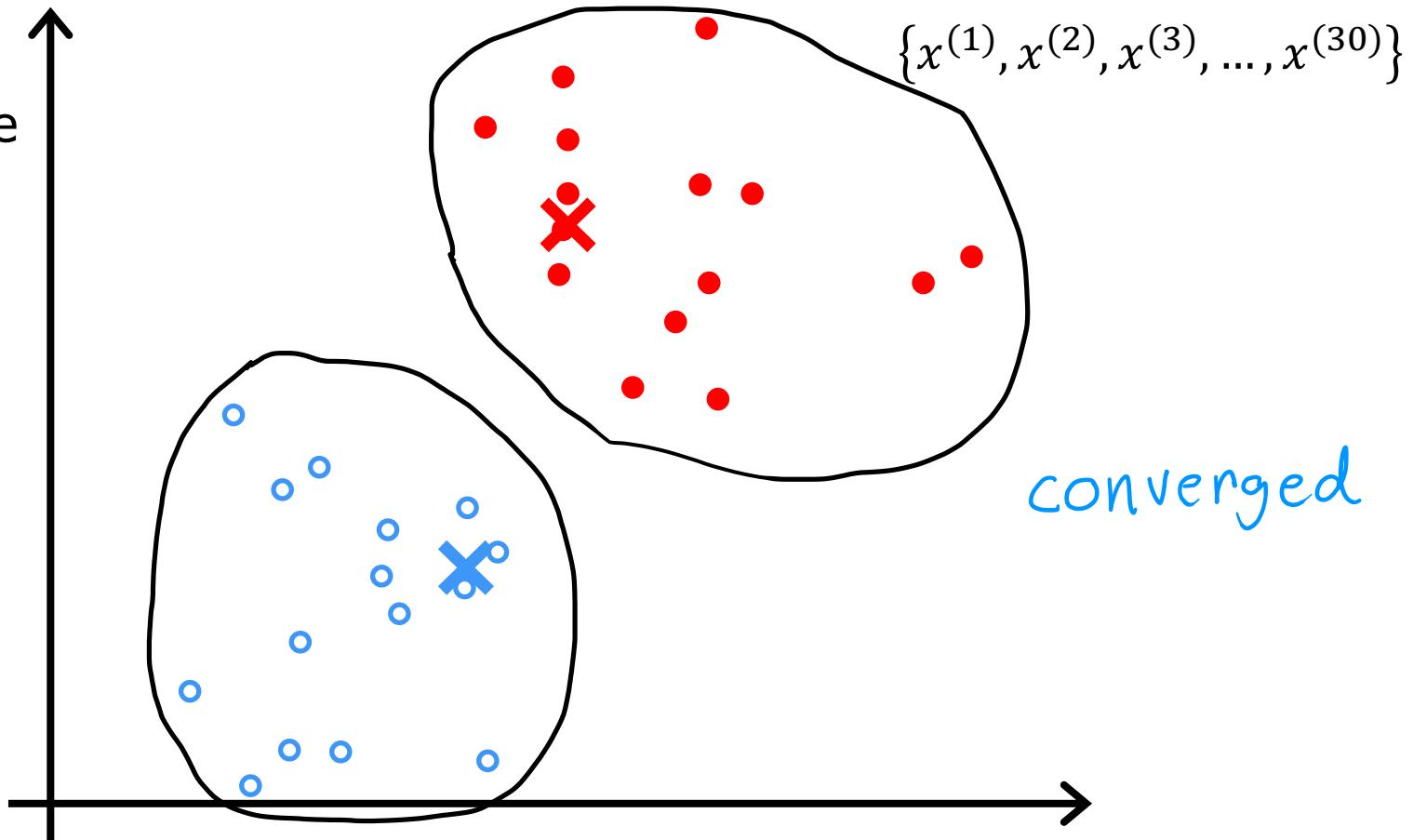
Step 2:
Recompute
the
centroids



Step 1:
Assign
each point
to its
closest
centroid

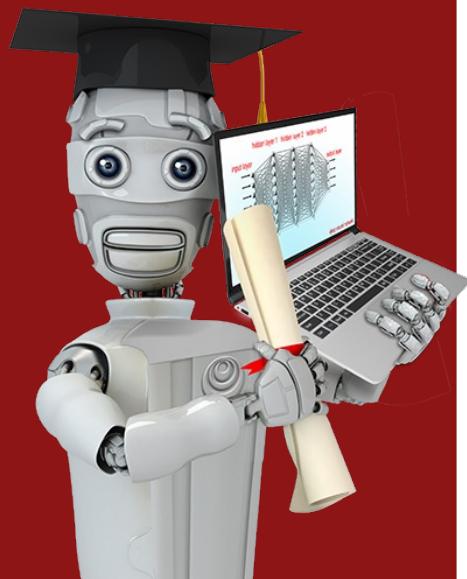


Step 2:
Recompute
the
centroids



Clustering

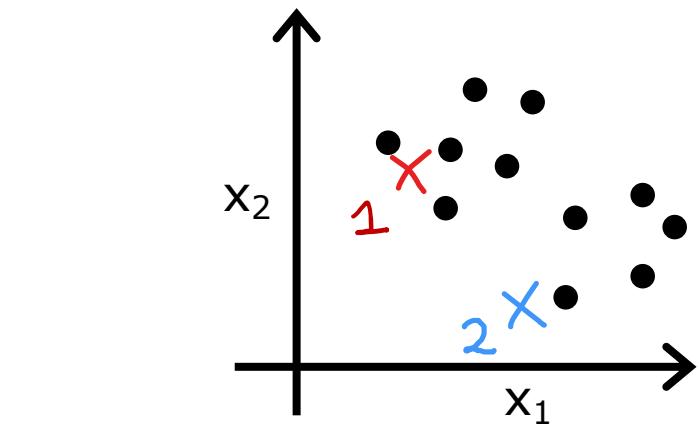
K-means algorithm



K-means algorithm

Randomly initialize K cluster centroids

μ_1, μ_2
 $n=2$ $x^{(1)}, x^{(2)}, \dots, x^{(30)}$
 $K=2$



K-means algorithm

Randomly initialize K cluster centroids $\underline{\mu_1}, \underline{\mu_2}, \dots, \underline{\mu_K}$

Repeat {

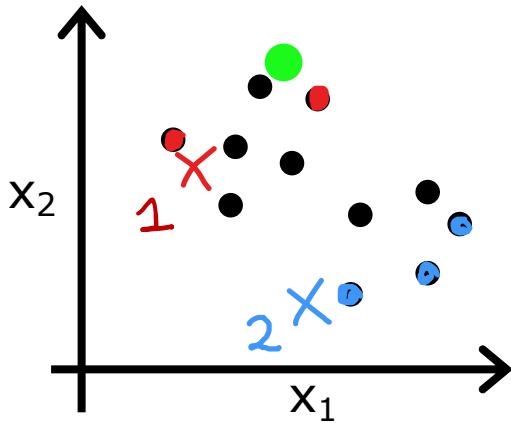
Assign points to cluster centroids

for $i = 1$ to m

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

$$\min_K \left\| \mathbf{x}^{(i)} - \boldsymbol{\mu}_K \right\|^2$$

$$\begin{array}{l} \color{blue}{\mu_1 \mu_2} \\ n=2 \end{array} \quad \begin{array}{l} \color{brown}{x^{(1)}, x^{(2)}, \dots, x^{(30)}} \\ k=2 \end{array}$$



K-means algorithm

Randomly initialize K cluster centroids

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

$$\min_K \sum_{k=1}^K \|x^{(i)} - \mu_k\|^2$$

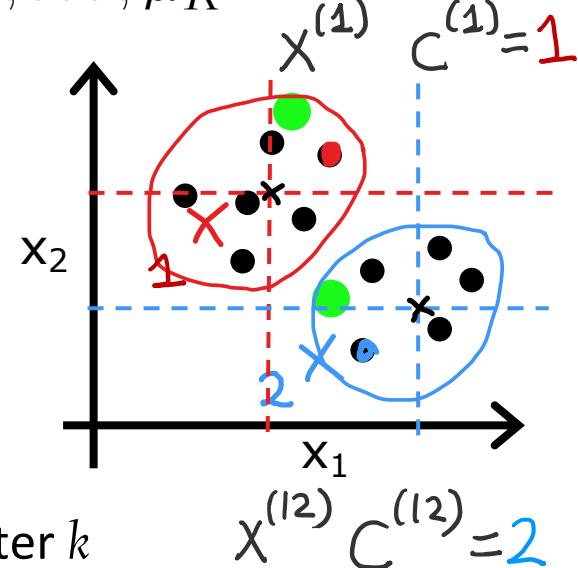
for $k = 1$ to K

$\mu_k :=$ average (mean) of points assigned to cluster k

}

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

$$\begin{aligned} & \mu_1 \mu_2 \\ & x^{(1)}, x^{(2)}, \dots, x^{(30)} \\ & n=2 \quad K=2 \\ & \mu_1, \mu_2, \dots, \mu_K \end{aligned}$$



K-means algorithm

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

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

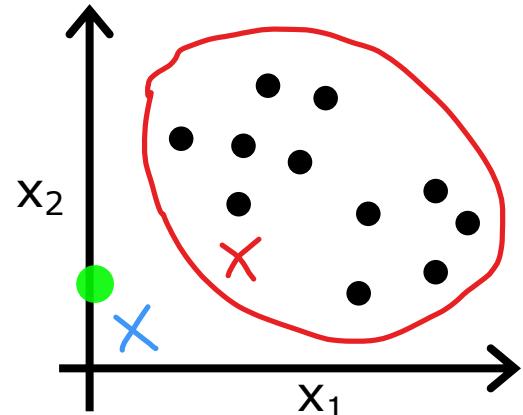
$$\min_K \sum_{i=1}^m \|x^{(i)} - \mu_k\|^2$$

for $k = 1$ to K

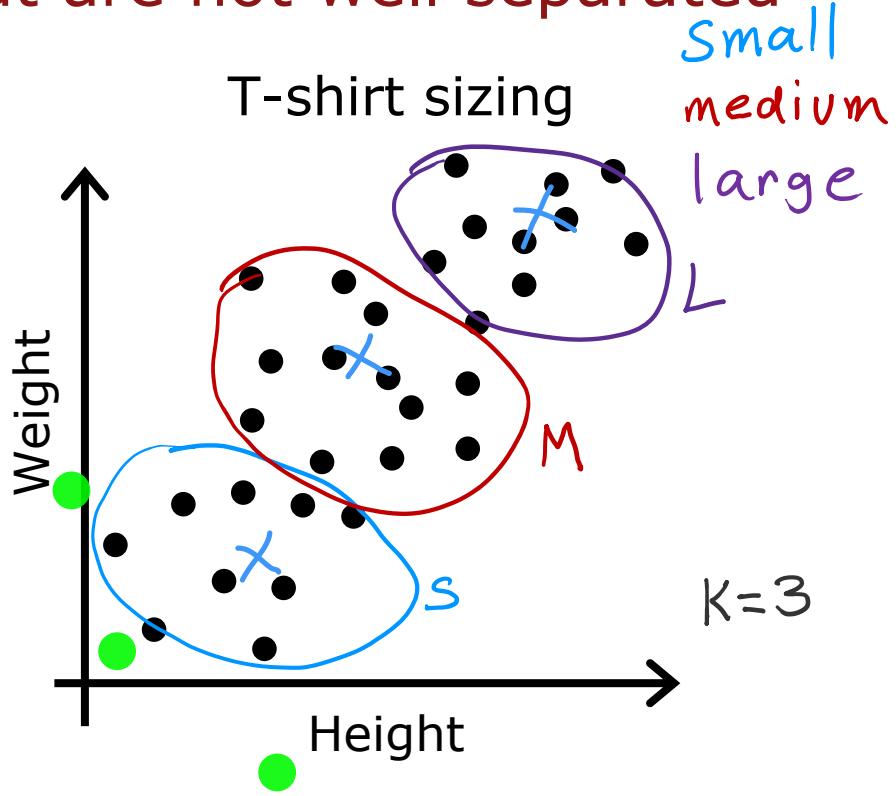
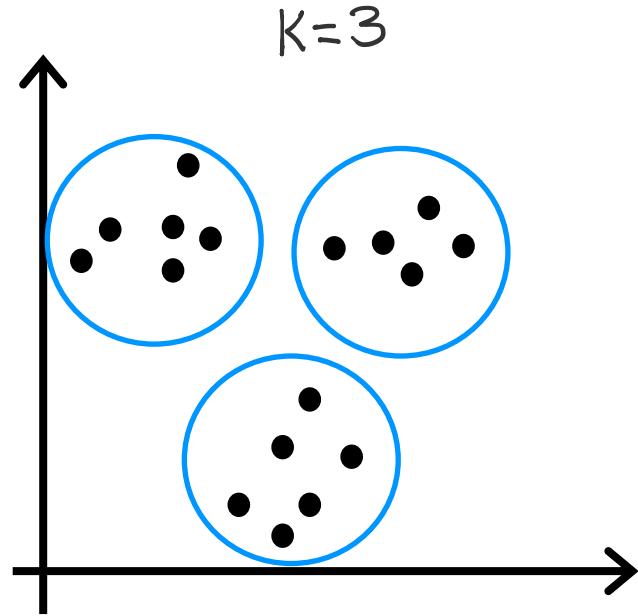
$\mu_k :=$ average (mean) of points assigned to cluster k

}

$\mu_1 \mu_2$
 $x^{(1)}, x^{(2)}, \dots, x^{(30)}$
 $n=2$ ~~K=2~~
K = K-1 ✓

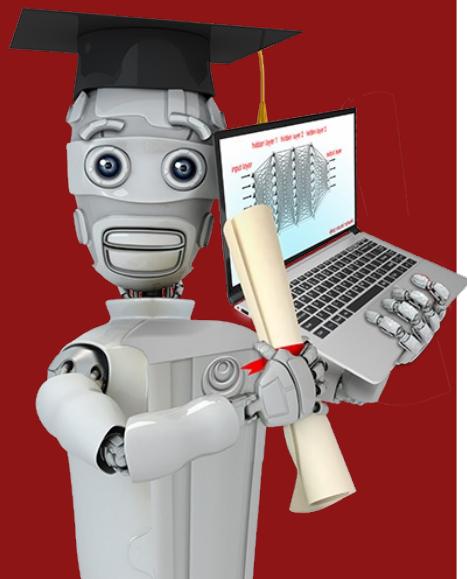


K-means for clusters that are not well separated



Clustering

Optimization objective



K-means optimization objective

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

μ_k = cluster centroid k

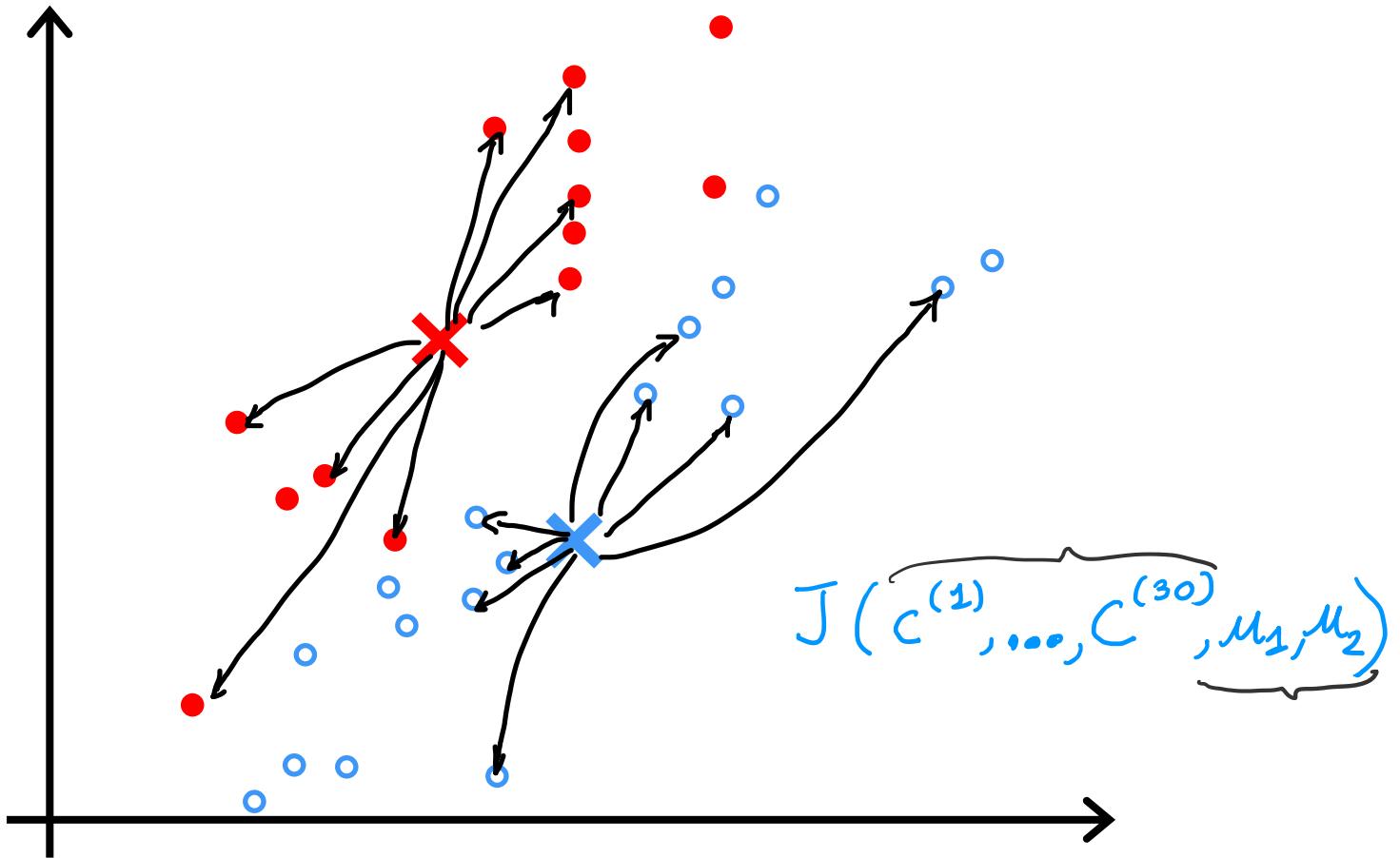
• $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Cost function

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

$\min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$ distortion

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

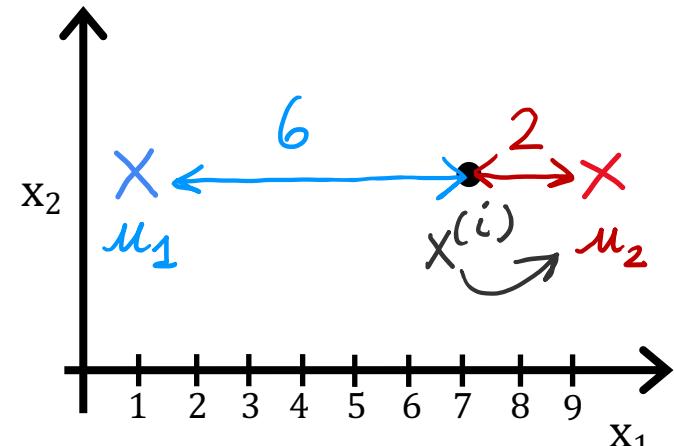


Cost function for K-means

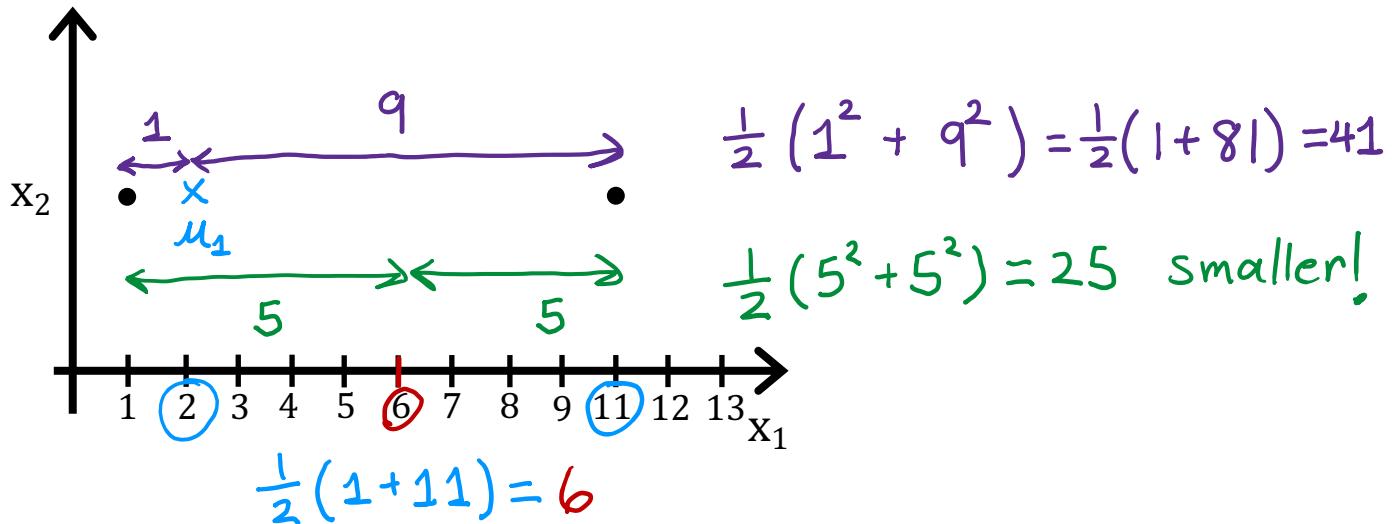
$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

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

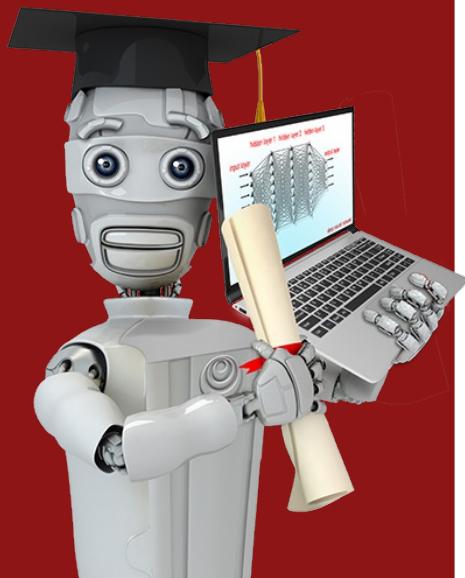


Moving the centroid



Clustering

Initializing K-means



K-means algorithm

- Step 0: Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_k$

Repeat {

Step 1: Assign points to cluster centroids

Step 2: Move cluster centroids

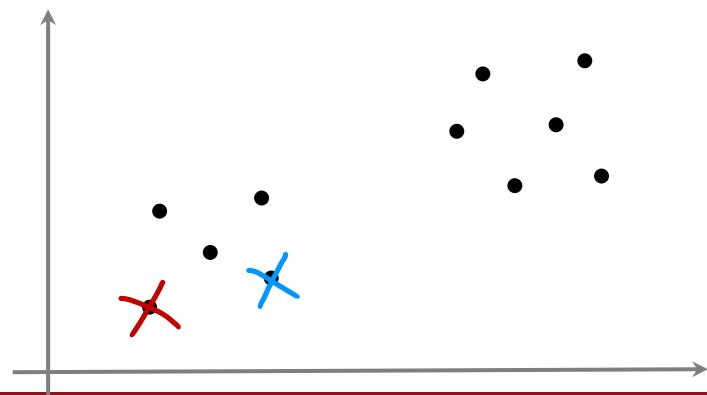
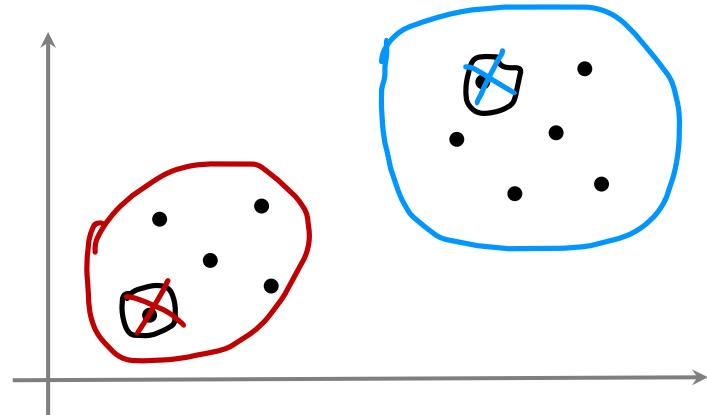
}

Random initialization

Choose $K < m$

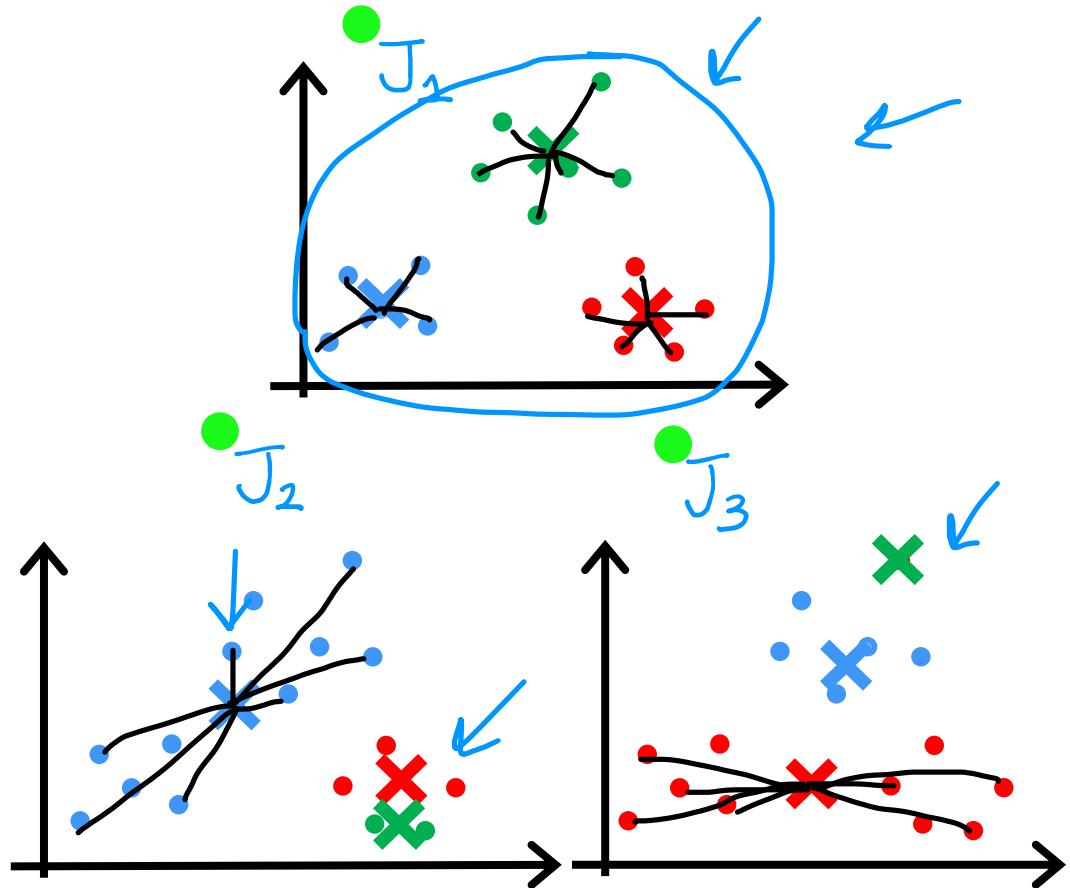
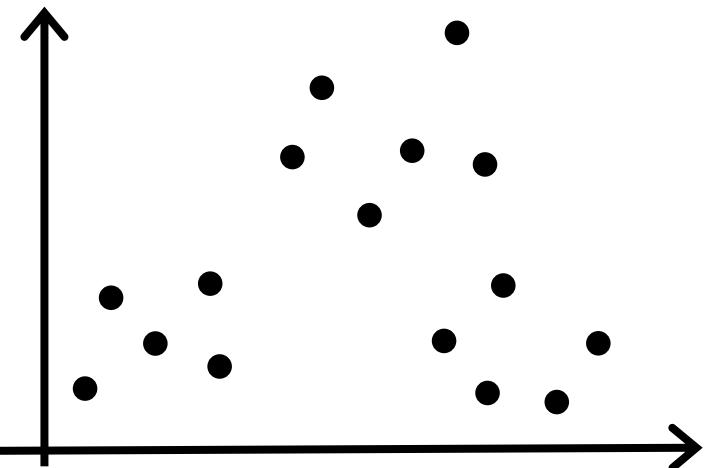
Randomly pick K training examples.

Set $\mu_1, \mu_2, \dots, \mu_K$ equal to these K examples.



$K=3$

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



Random initialization

better than running K-means once

For $i = 1$ to 100 {

50-1000

Randomly initialize K-means.

PICK K random examples
Set as cluster centroids

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

Compute $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \mu_1, \dots, \mu_k)$

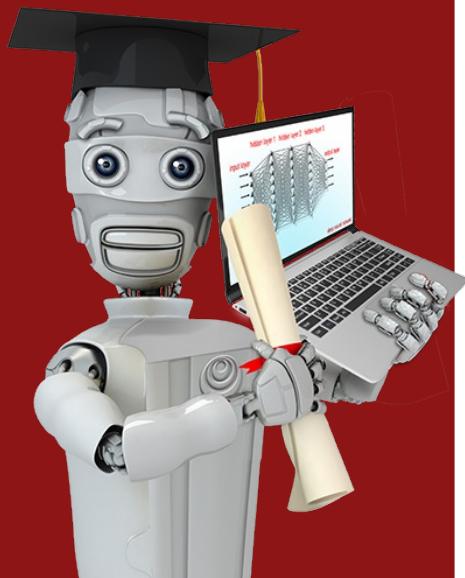
distortion
(cost function)

}

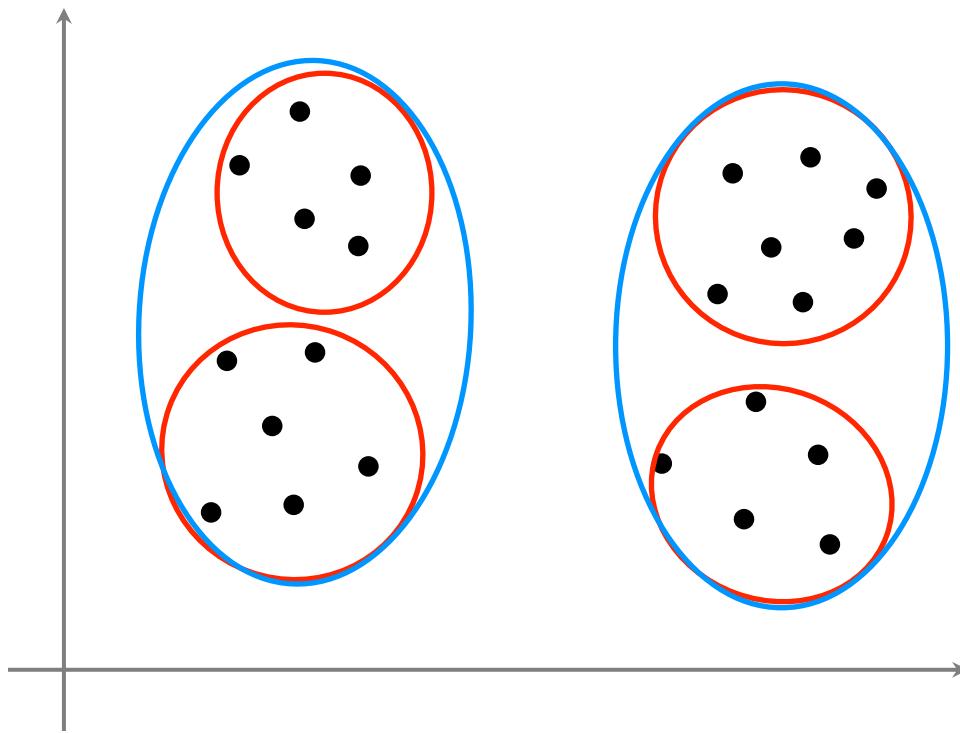
Pick set of clusters that gave lowest cost J

Clustering

Choosing the Number of Clusters

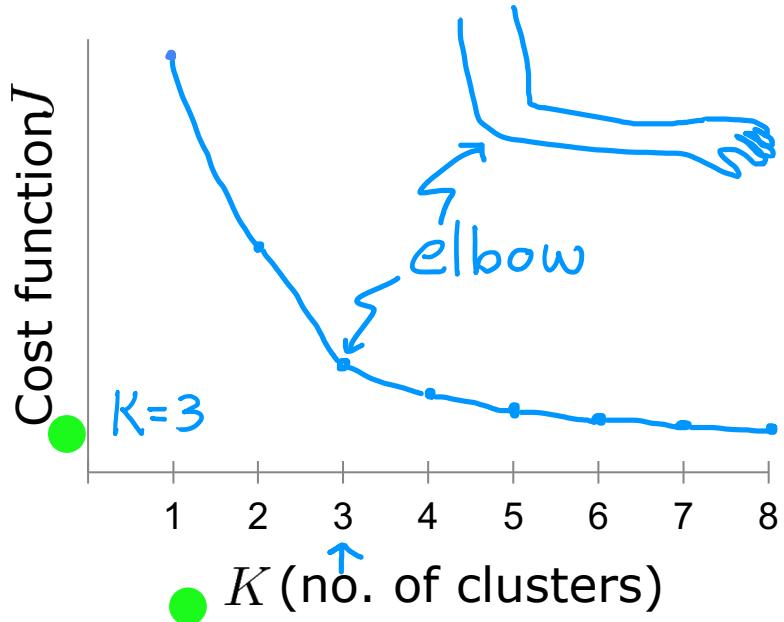


What is the right value of K?

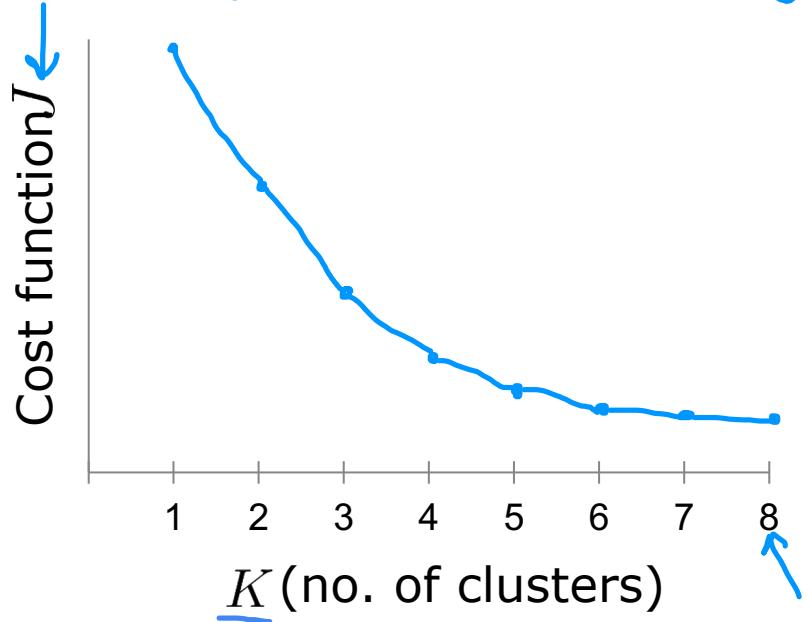


Choosing the value of K

Elbow method

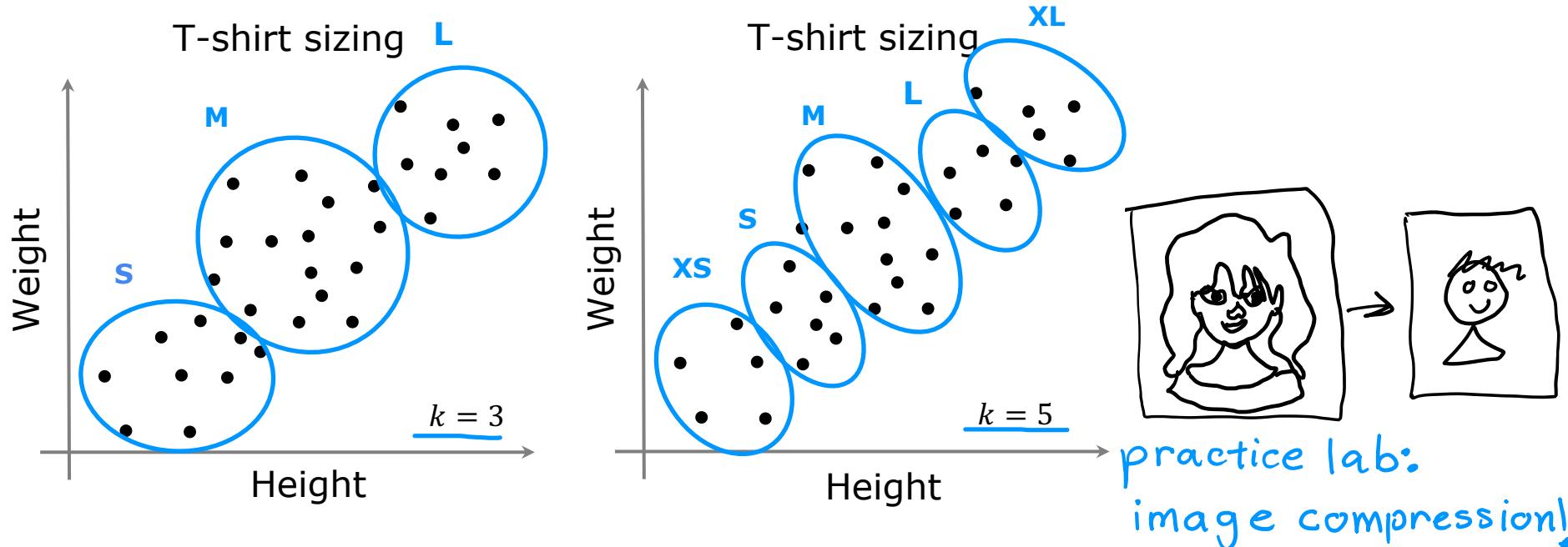


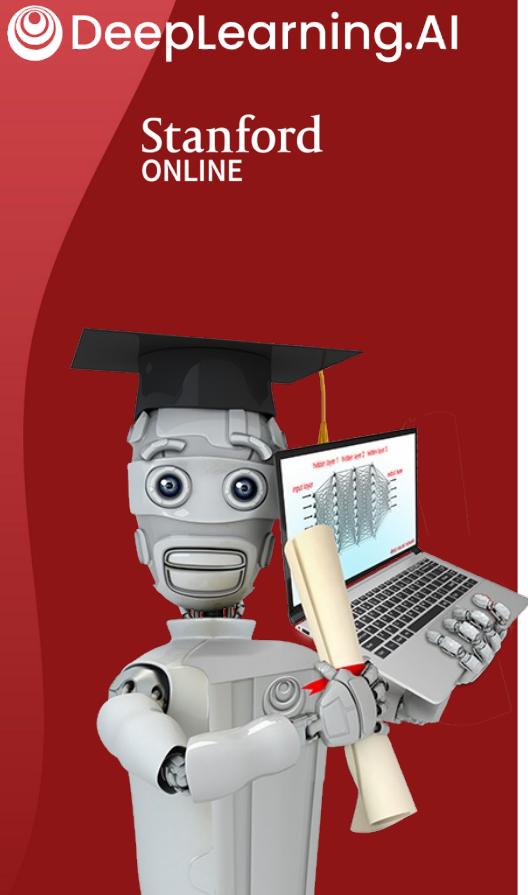
the right "k" is often ambiguous
Don't choose K just to minimize cost J



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.





 DeepLearning.AI

**Stanford
ONLINE**

Anomaly Detection

Finding unusual events

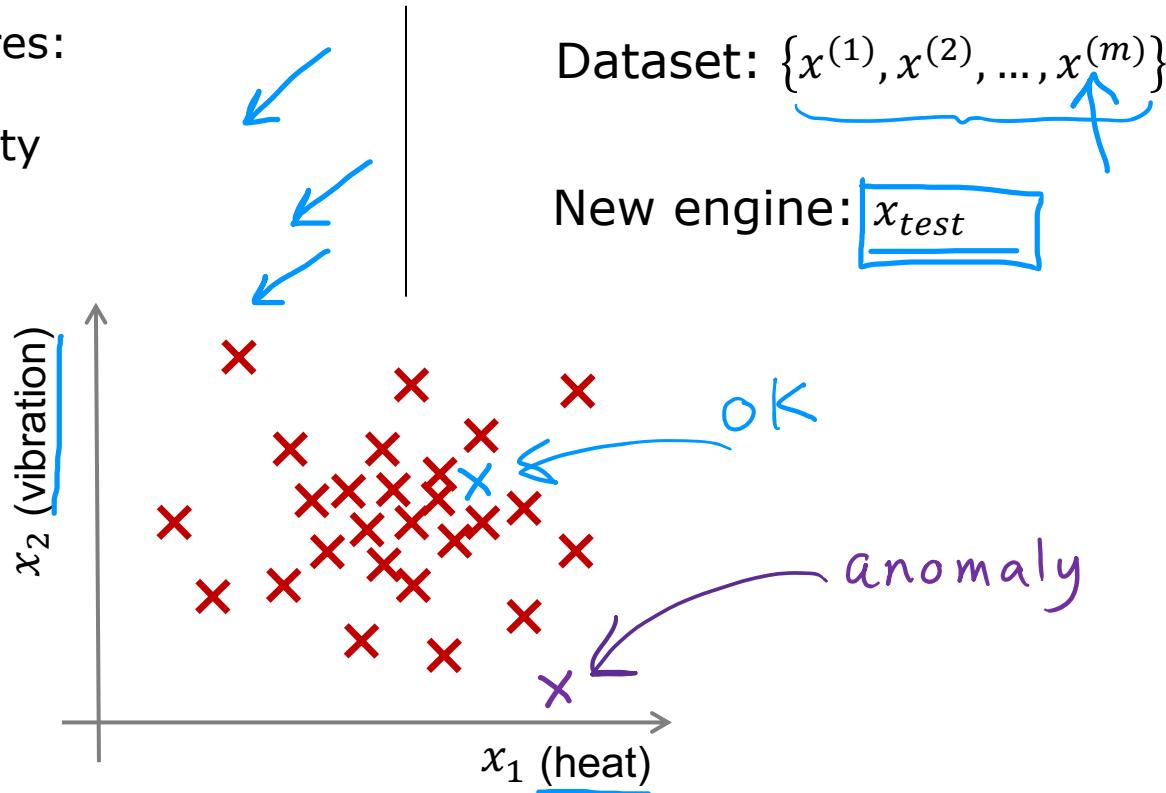
Anomaly detection example

Aircraft engine features:

x_1 = heat generated

x_2 = vibration intensity

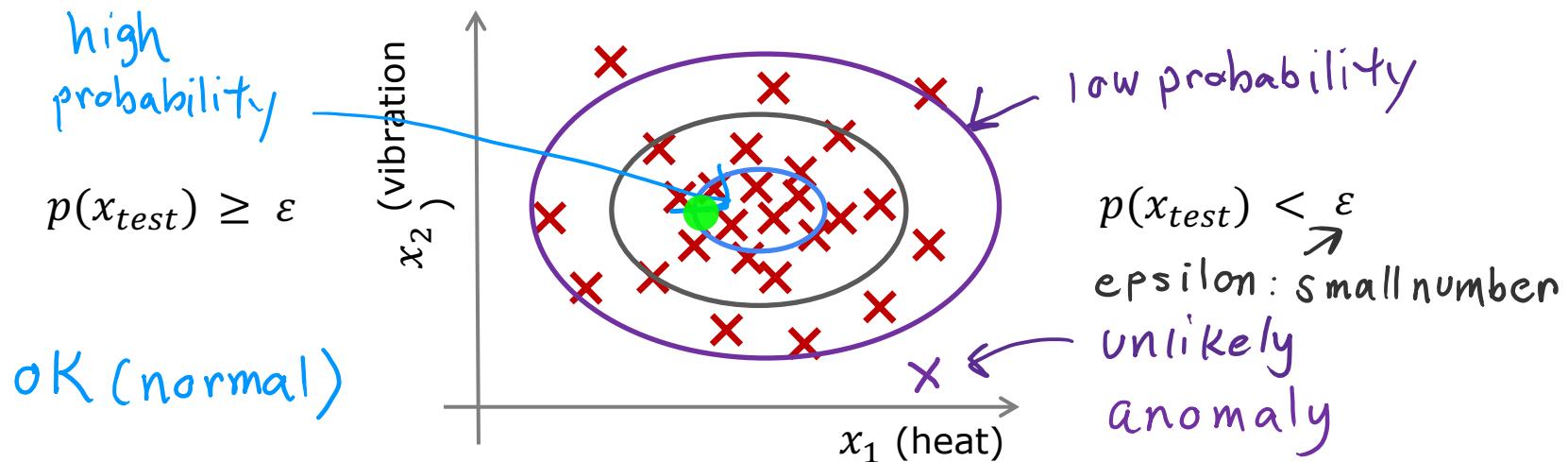
...



Density estimation

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

Is x_{test} anomalous?



Anomaly detection example

Fraud detection:

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

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

perform additional checks to identify real fraud vs. false alarms

Manufacturing:

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

airplane engine

circuit board

smartphone

rations

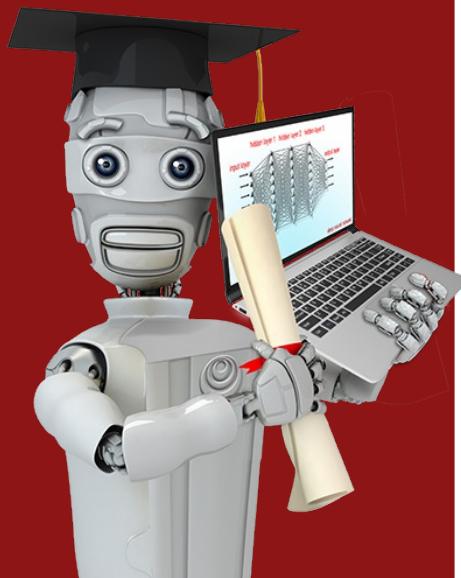
Monitoring computers in a data center:

$x^{(i)}$ = 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.

Anomaly Detection

Gaussian (Normal) Distribution

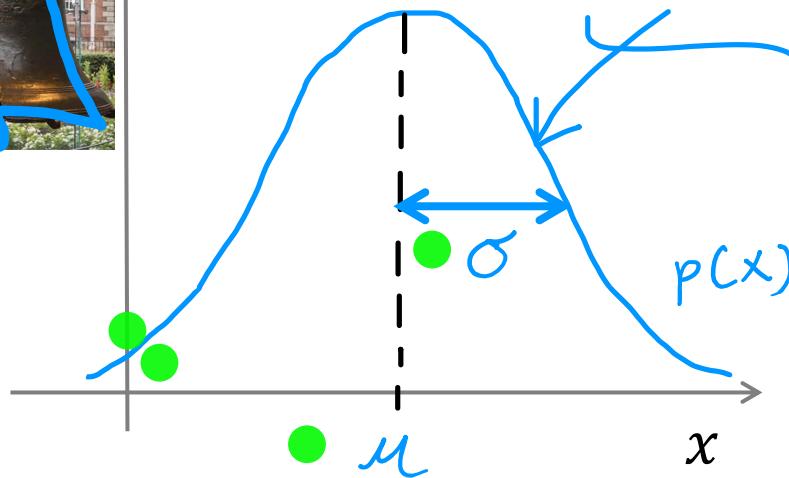


Gaussian (Normal) distribution

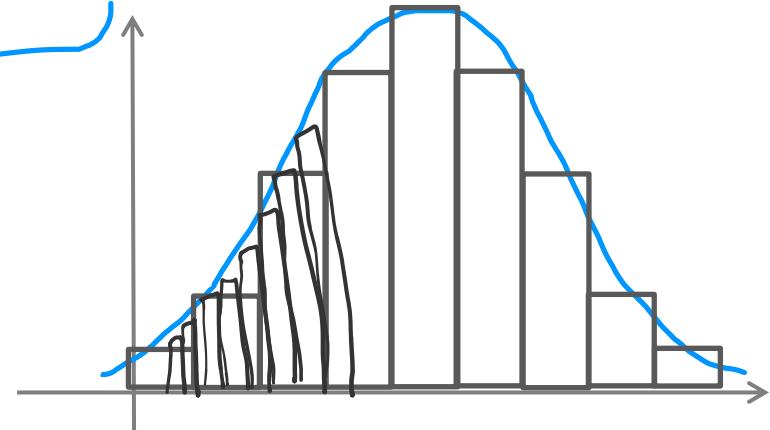
Say x is a number.

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

σ standard deviation
 σ^2 variance

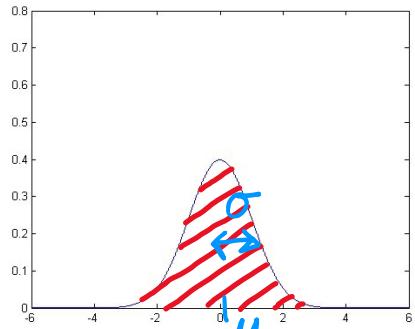


$$\pi = 3.14$$

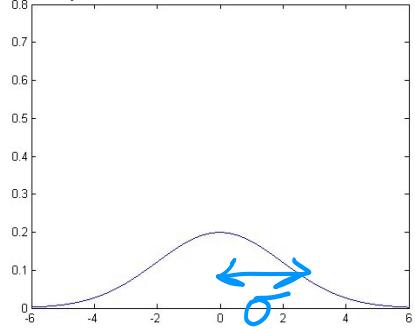


Gaussian distribution example

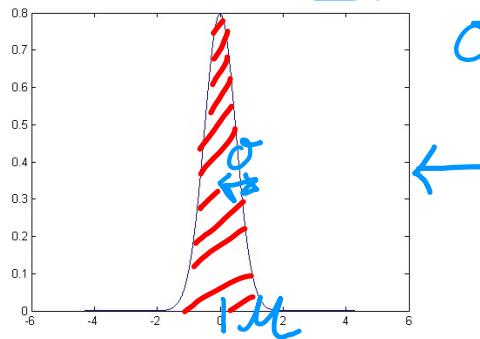
$$\mu = 0, \sigma = 1$$



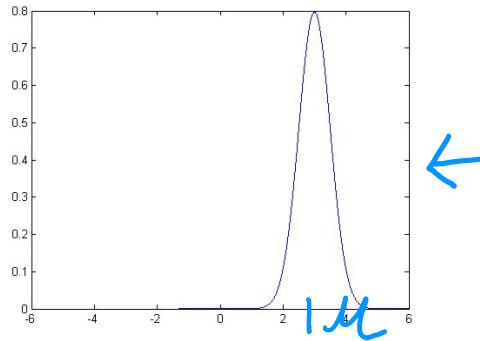
$$\mu = 0, \sigma = 2$$



$$\mu = 0, \underline{\sigma} = 0.5$$



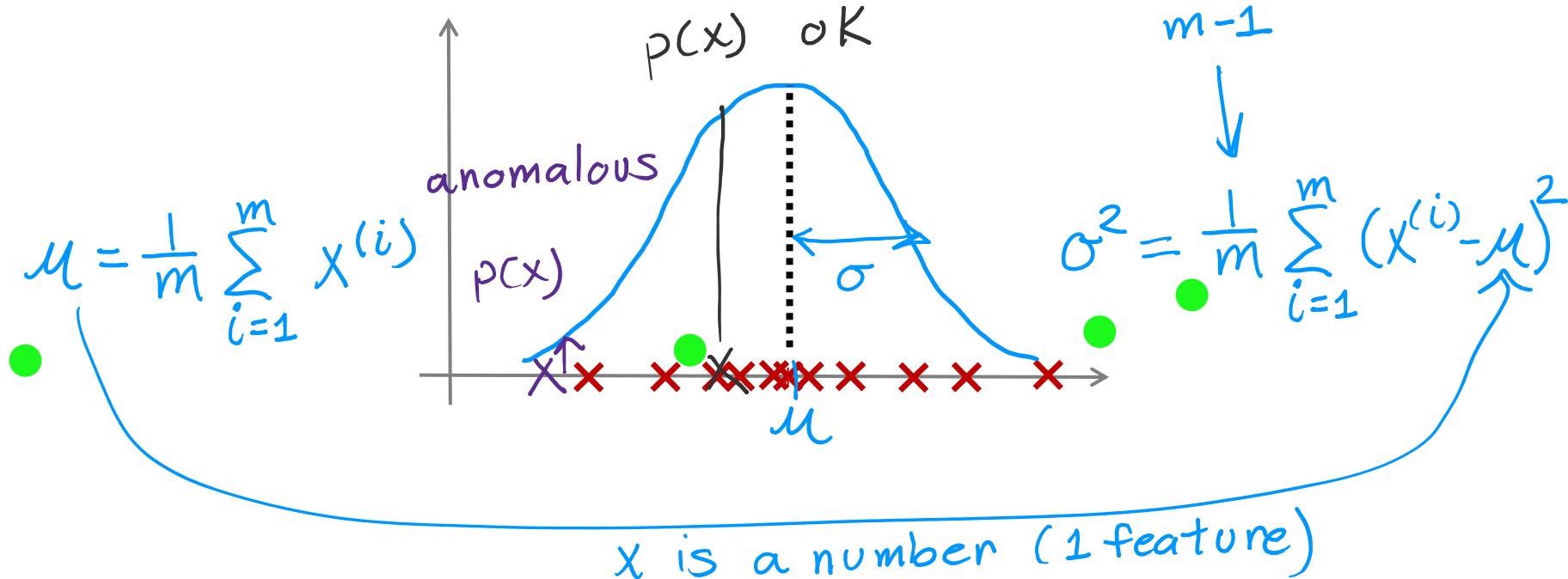
$$\mu = 3, \sigma = 0.5$$



Parameter estimation

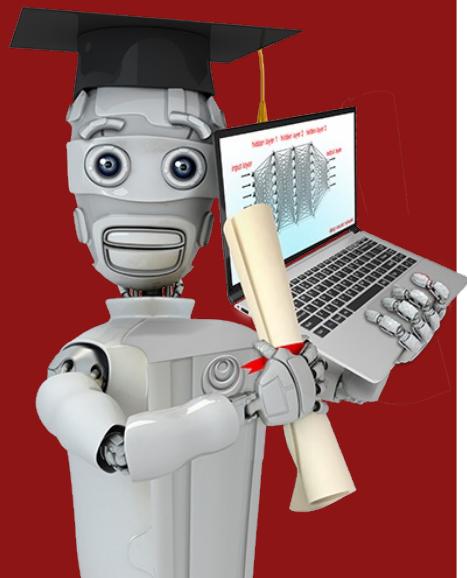
Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

maximum likelihood
for μ, σ



Anomaly Detection

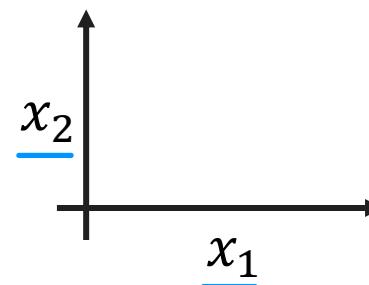
Algorithm



Density estimation

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

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


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

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

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \sum_{\text{"add"}} \prod_{\text{"multiply"}}$$

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

Anomaly detection algorithm

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

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

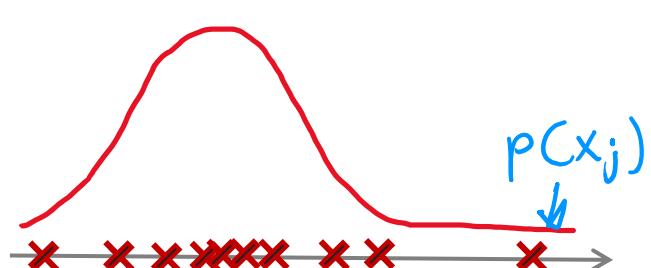
3. Given new example x , compute $p(x)$:

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

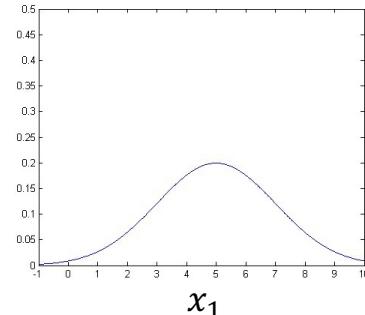
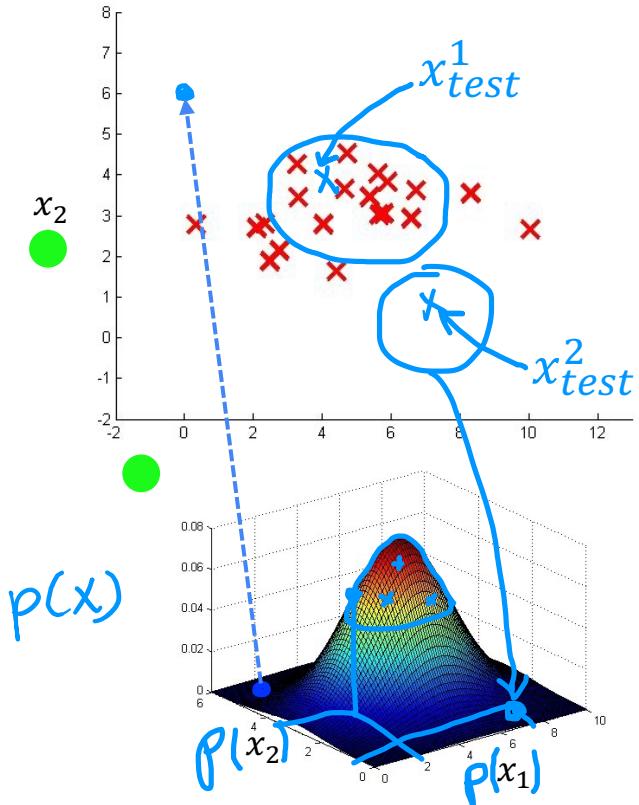
Anomaly if $p(x) < \varepsilon$

Vectorized formula

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



Anomaly detection example



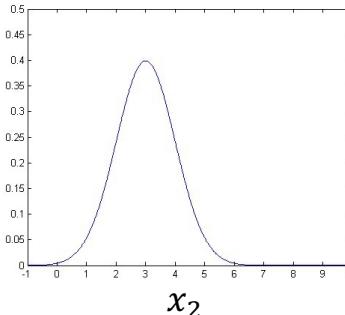
$$\mu_1 = 5, \sigma_1 = 2$$

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

$$\varepsilon = 0.02$$

$$p(x_{test}^{(1)}) = 0.0426$$

$$p(x_{test}^{(2)}) = 0.0021$$



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

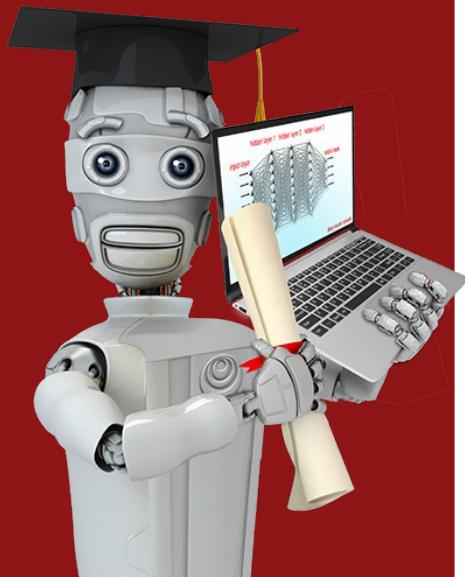
$$p(x_2; \mu_2, \sigma_2^2)$$

"OK"

anomaly

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.


$$y = 1 \quad y = 0$$

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

$y=0$ for all training examples

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

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

} 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 $y=0$

Training set: 6000 good engines

CV: 2000 good engines ($y = 0$)
use cross validation set

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

2 to 50
 $y=1$

train algorithm on training set

10 anomalous ($y = 1$)
tune ϵ tune x_j

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

Training set: 6000 good engines 2 higher risk of overfitting

CV: 4000 good engines ($y = 0$) 20 anomalous ($y = 1$)
tune ϵ tune x_j

Algorithm evaluation

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

On a cross validation/test example x , predict

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

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

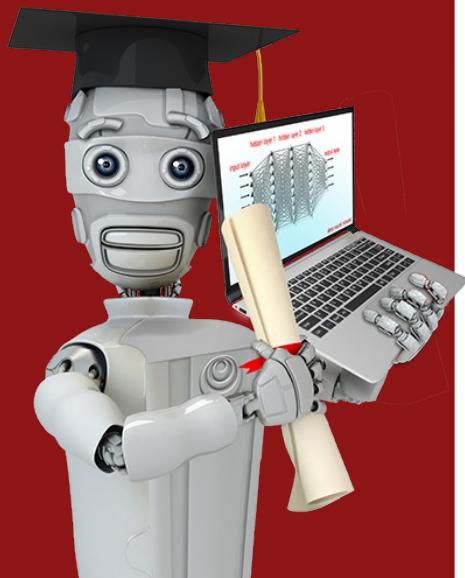
course 2 week 3
skewed datasets

10

2000

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.

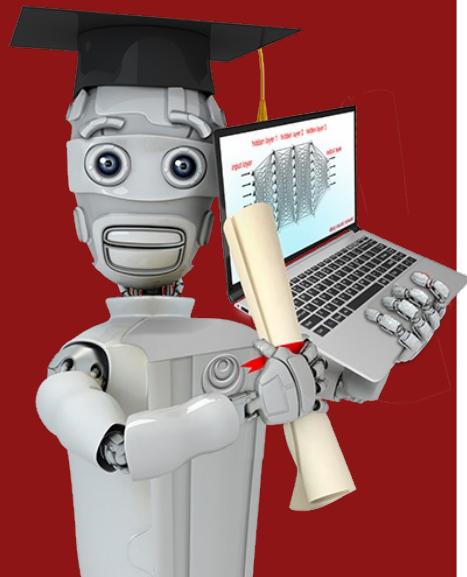
Supervised learning

- Fraud detection
- Manufacturing:
Finding new previously unseen defects.
(e.g. aircraft engines)
- Monitoring machines in a data center
- Security applications

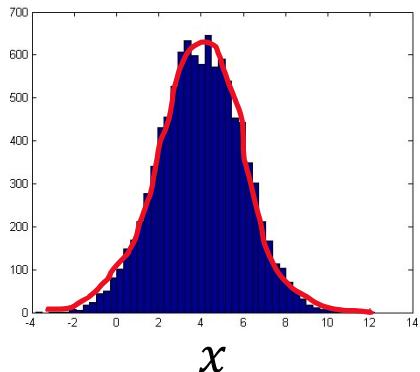
- Email spam classification
- Manufacturing:
Finding known, previously seen defects
(e.g. scratches on smartphones, $y = 1$)
- Weather prediction (sunny/rainy/etc.)
- Diseases classification

Anomaly Detection

Choosing what features to use



Non-gaussian features



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

`plt.hist(x)`

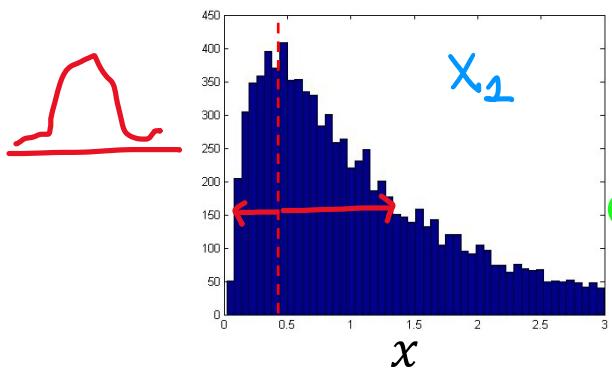


$$x_1 \leftarrow \log(x_1)$$

$$x_2 \leftarrow \log(x_2 + 1) \quad \text{log}(x_2 + c)$$

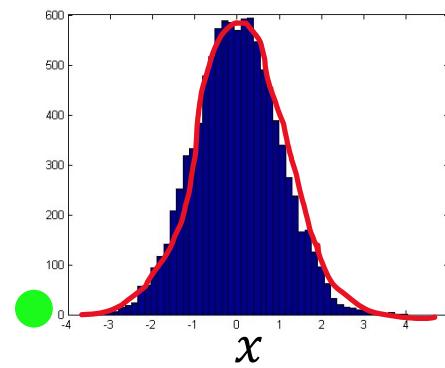
$$x_3 \leftarrow \sqrt{x_3} = x_3^{1/2}$$

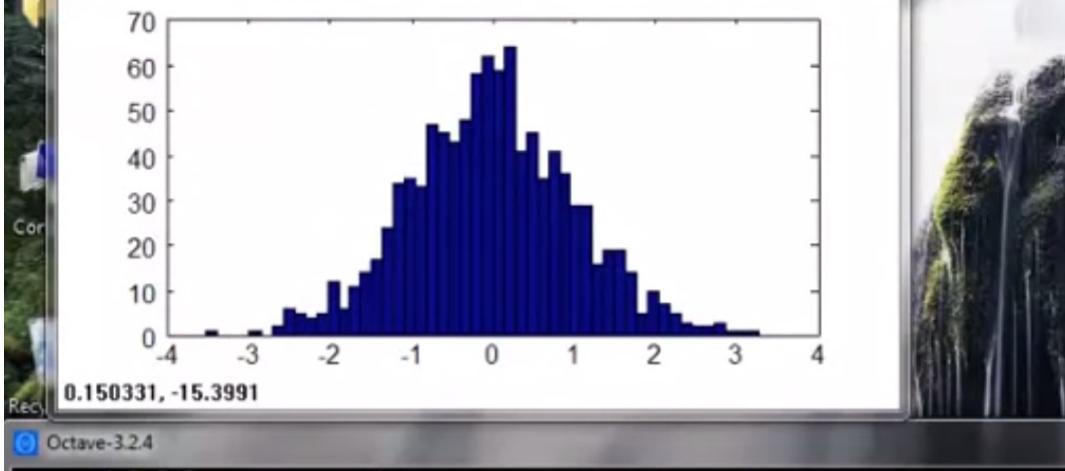
$$x_4 \leftarrow x_4^{1/3}$$



x_1

`np.log(x)`





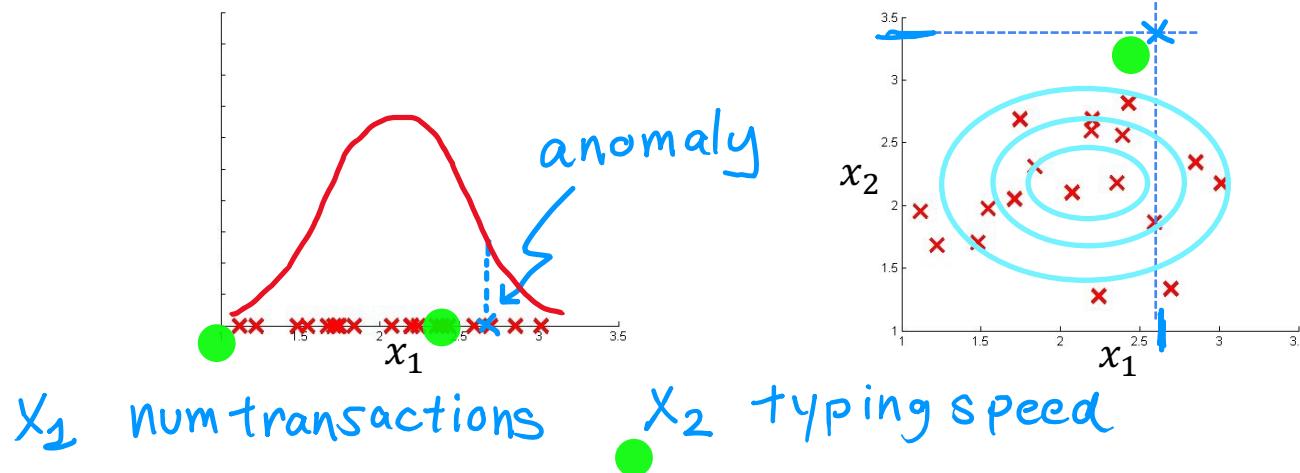
```
1000      1
octave-3.2.4.exe:6> hist(x)
octave-3.2.4.exe:7> hist(x,50)
octave-3.2.4.exe:8> hist(x.^0.5, 50)
octave-3.2.4.exe:9> hist(x.^0.2, 50)
octave-3.2.4.exe:10> hist(x.^0.1, 50)
octave-3.2.4.exe:11> hist(x.^0.05, 50)
octave-3.2.4.exe:12> xNew = x.^0.05;
octave-3.2.4.exe:13> hist(log(x),50)
octave-3.2.4.exe:14> xNew = log(x);
```

Error analysis for anomaly detection

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

Most common problem:

$p(x)$ is comparable for normal and anomalous examples.
($p(x)$ is large for both)



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

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

not unusual

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

Deciding feature choice based on $p(x)$

Large for normal examples;

Becomes small for anomaly in the cross validation set