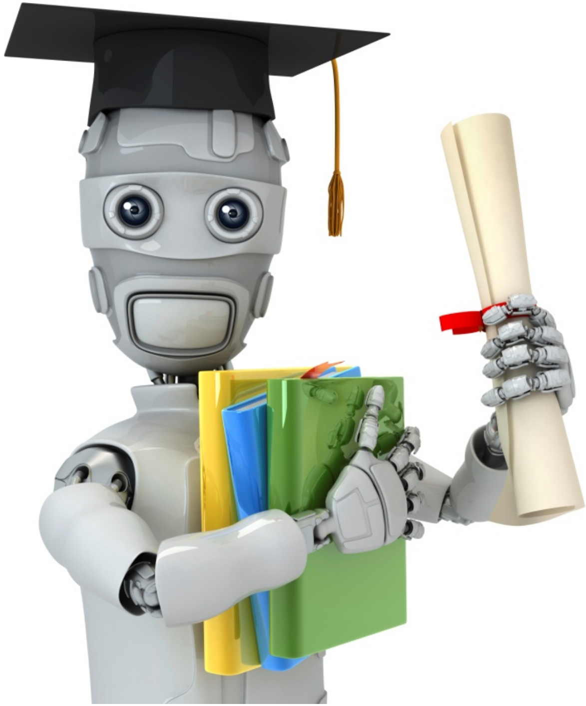


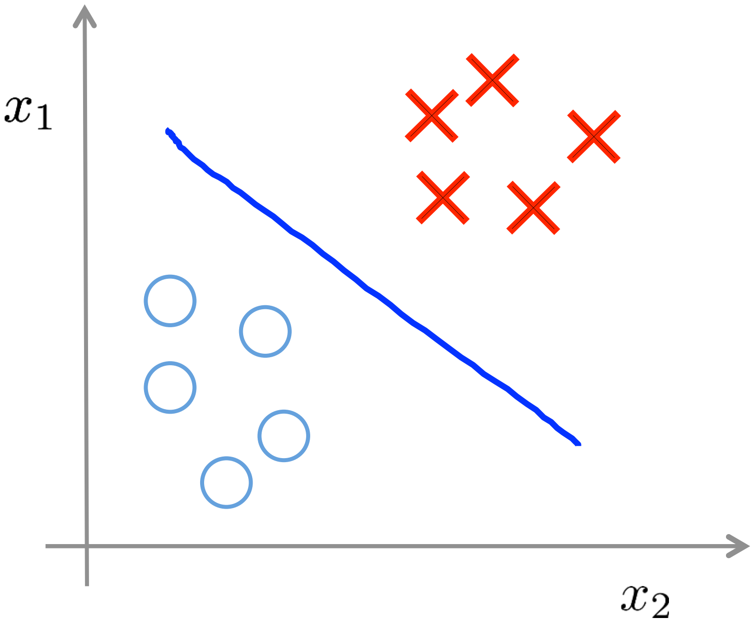
Machine Learning

# Clustering

## Unsupervised learning introduction

# Supervised learning

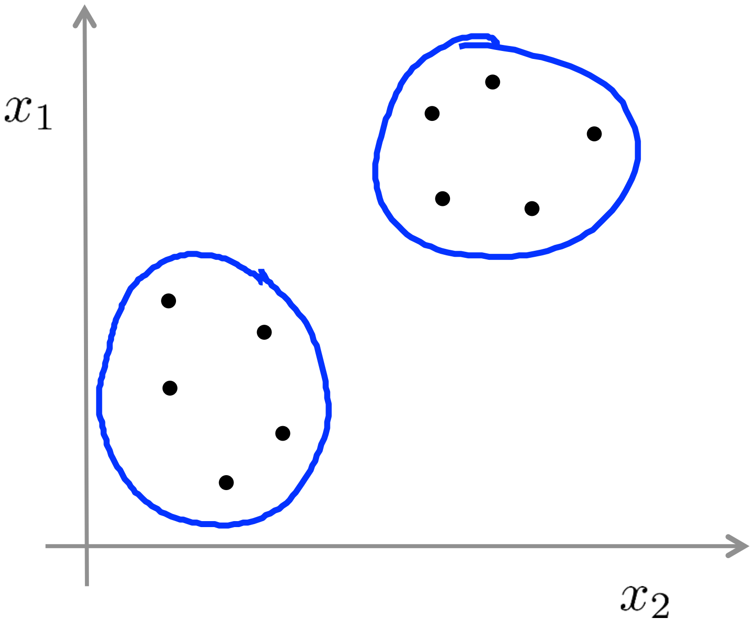
a typical supervised learning problem where we're given a labeled training set and the goal is to find the decision boundary that separates the positive label examples and the negative label examples.




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

# Unsupervised learning

in the unsupervised learning problem we're given data that does not have any labels associated with it. So, we're given data that looks like this. Here's a set of points add in no labels, and so, our training set is written just x1, x2, and so on up to xm and we don't get any labels y. And that's why the points plotted up on the figure don't have any labels with them. So, in unsupervised learning what we do is we give this sort of unlabeled training set to an algorithm and we just ask the algorithm find some structure in the data for us.

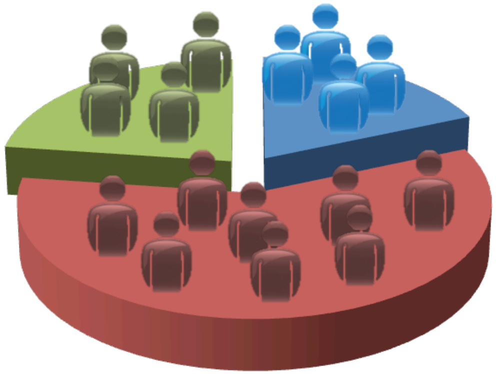



Clustering algorithm

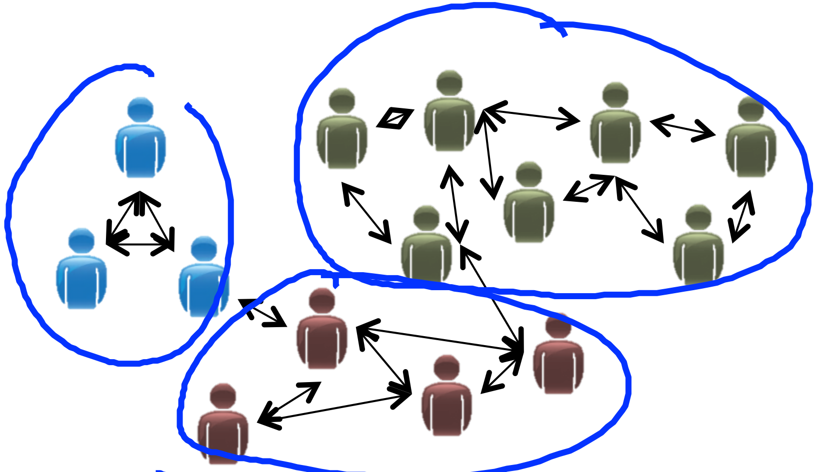
clustering is not the only unsupervised learning !

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

# Applications of clustering



Market segmentation



Social network analysis

Organize computing clusters

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

Astronomical data analysis

Which of the following statements are true? Check all that apply.

- [x] In unsupervised learning, the training set is of the form $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ without labels $y^{(i)}$.

**Correct**

- [x] Clustering is an example of unsupervised learning.

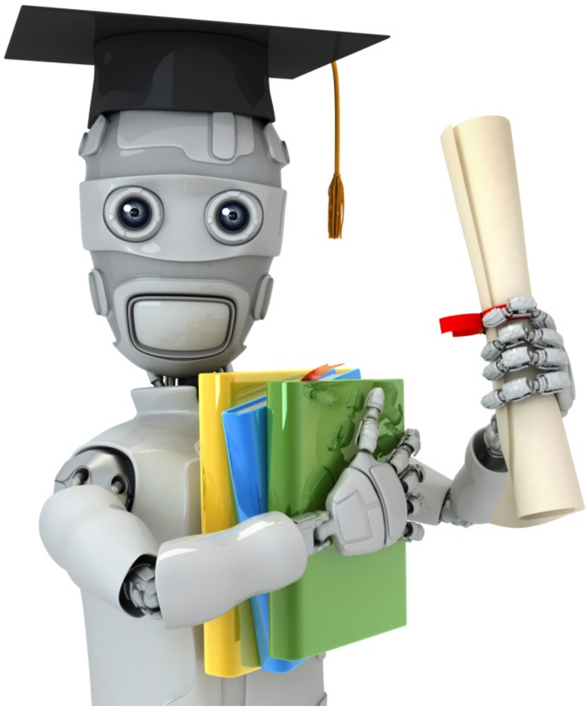
**Correct**

- [x] In unsupervised learning, you are given an unlabeled dataset and are asked to find "structure" in the data.

**Correct**

- [ ] Clustering is the only unsupervised learning algorithm.

**Un-selected is correct**

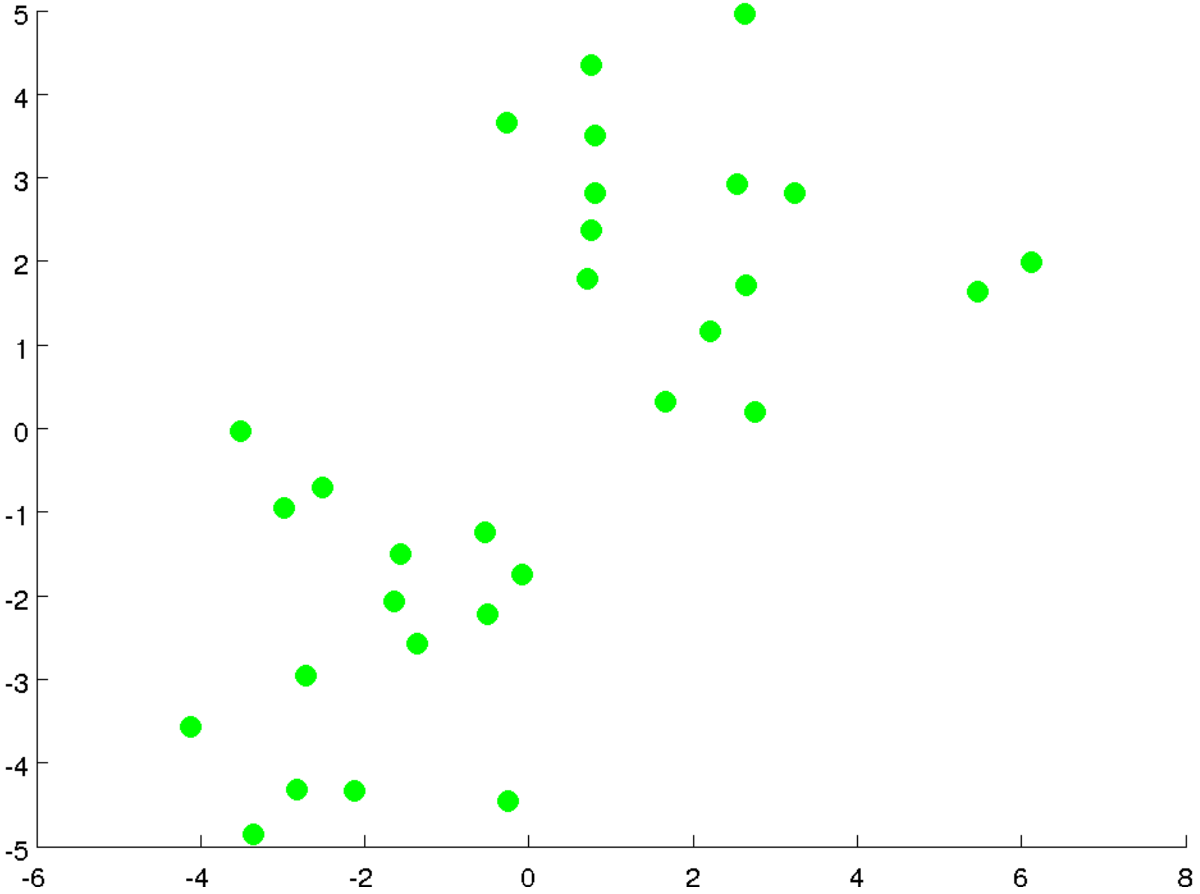


Machine Learning

# Clustering

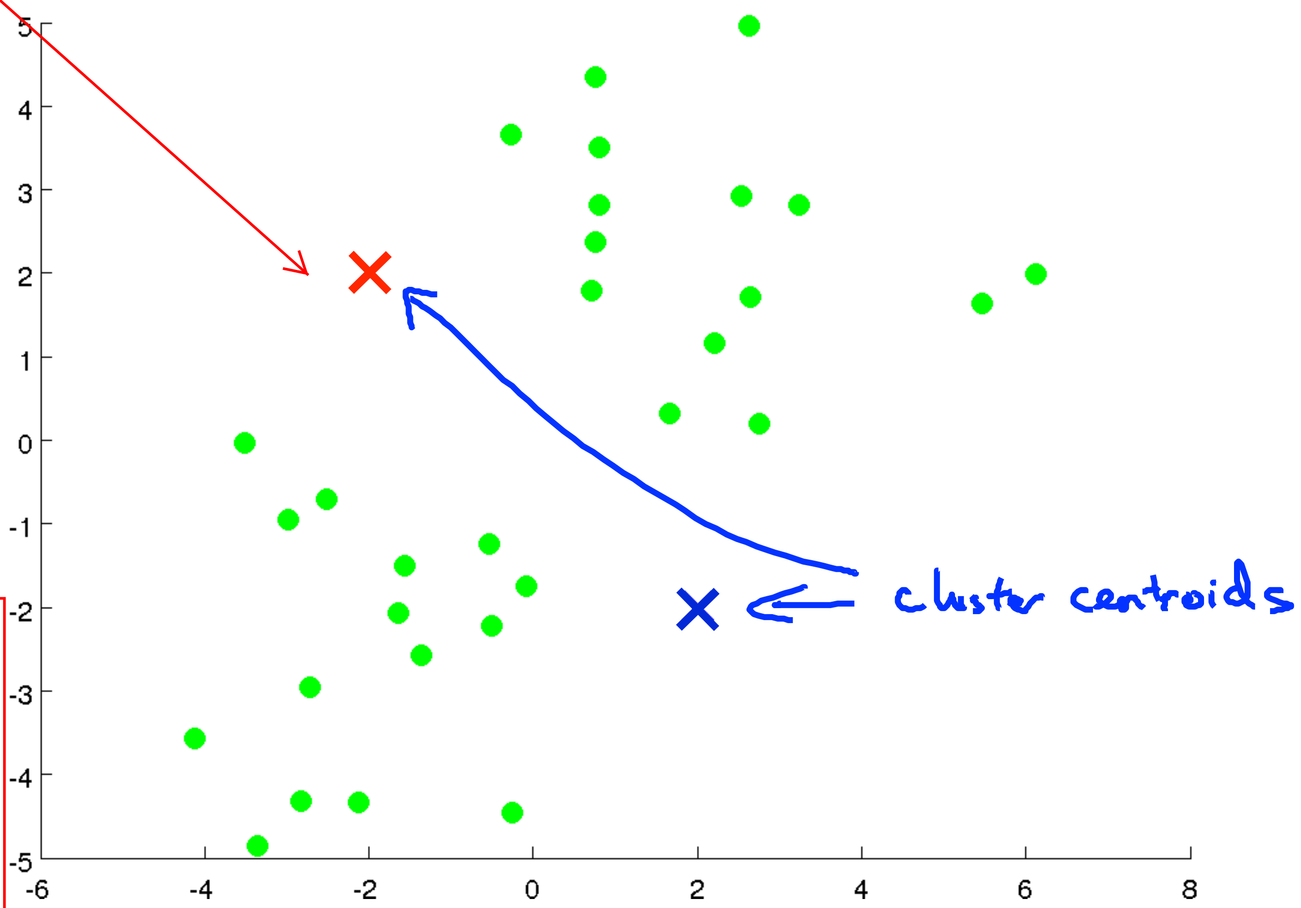
## K-means algorithm

In the clustering problem we are given an unlabeled data set and we would like to have an algorithm automatically group the data into coherent subsets or into coherent clusters for us. The K Means algorithm is by far the most popular, by far the most widely used clustering algorithm.



5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

The first step is to randomly initialize two points, called the cluster centroids. So, these two crosses here, these are called the Cluster Centroids,and I have two of them because I want to group my data into two clusters.



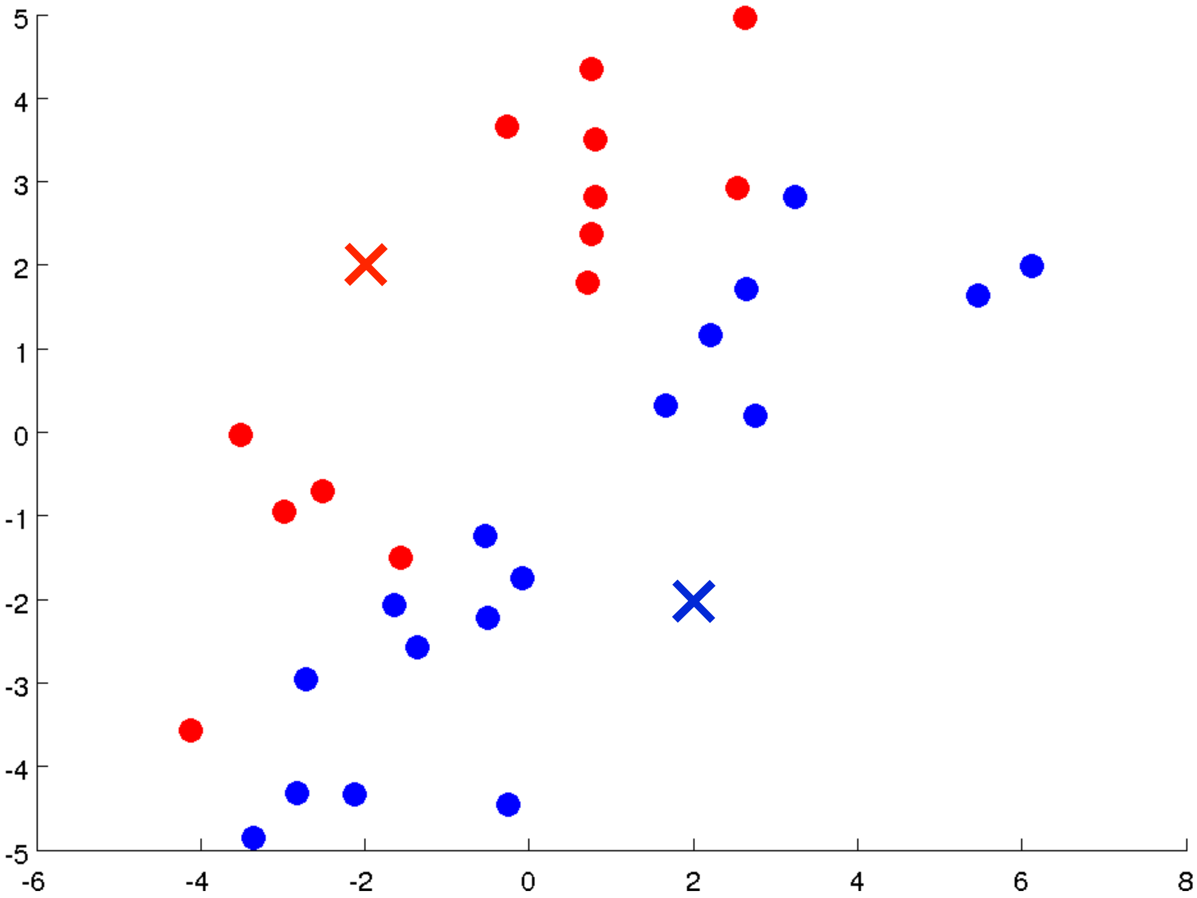

K Means is an iterative algorithm and it does two things:

(1)
First is a cluster assignment step

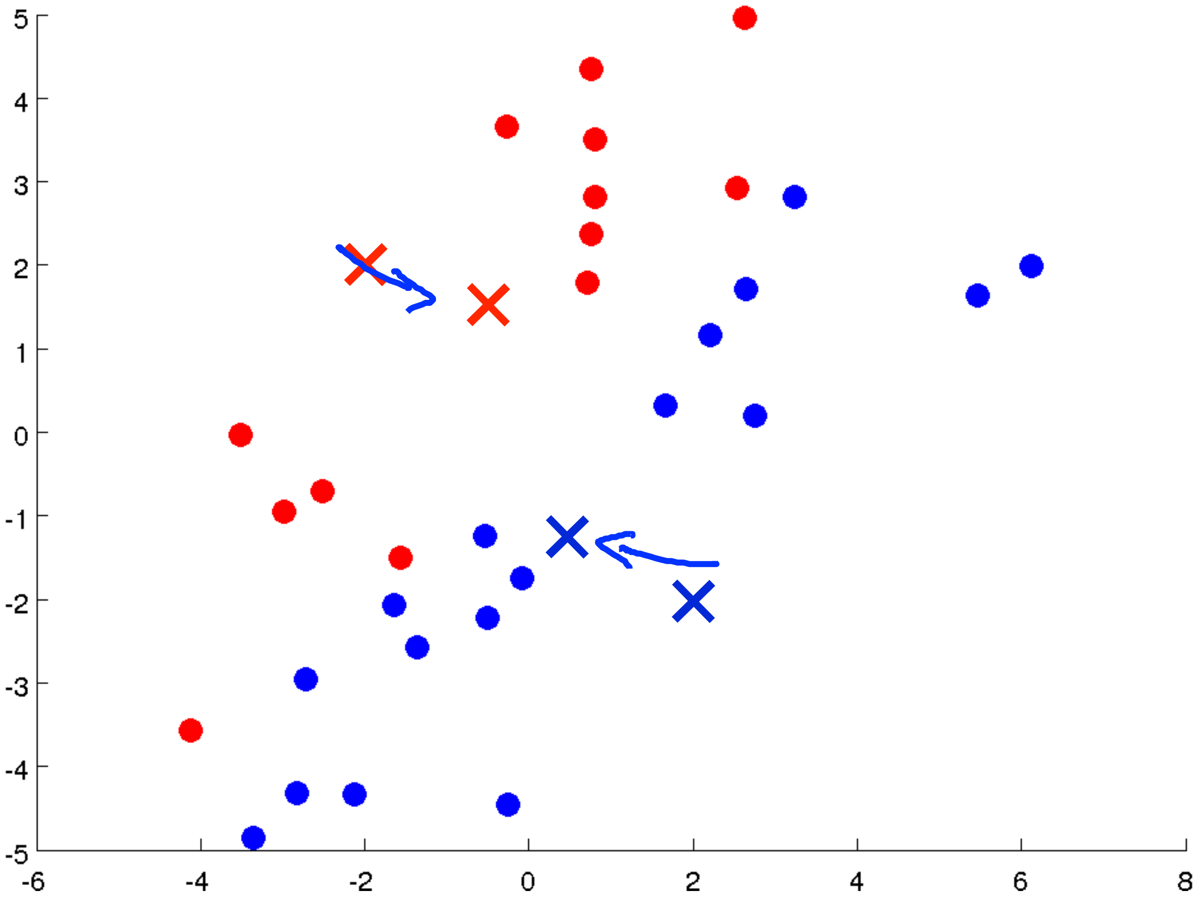
(2)
second is a move centroid step.

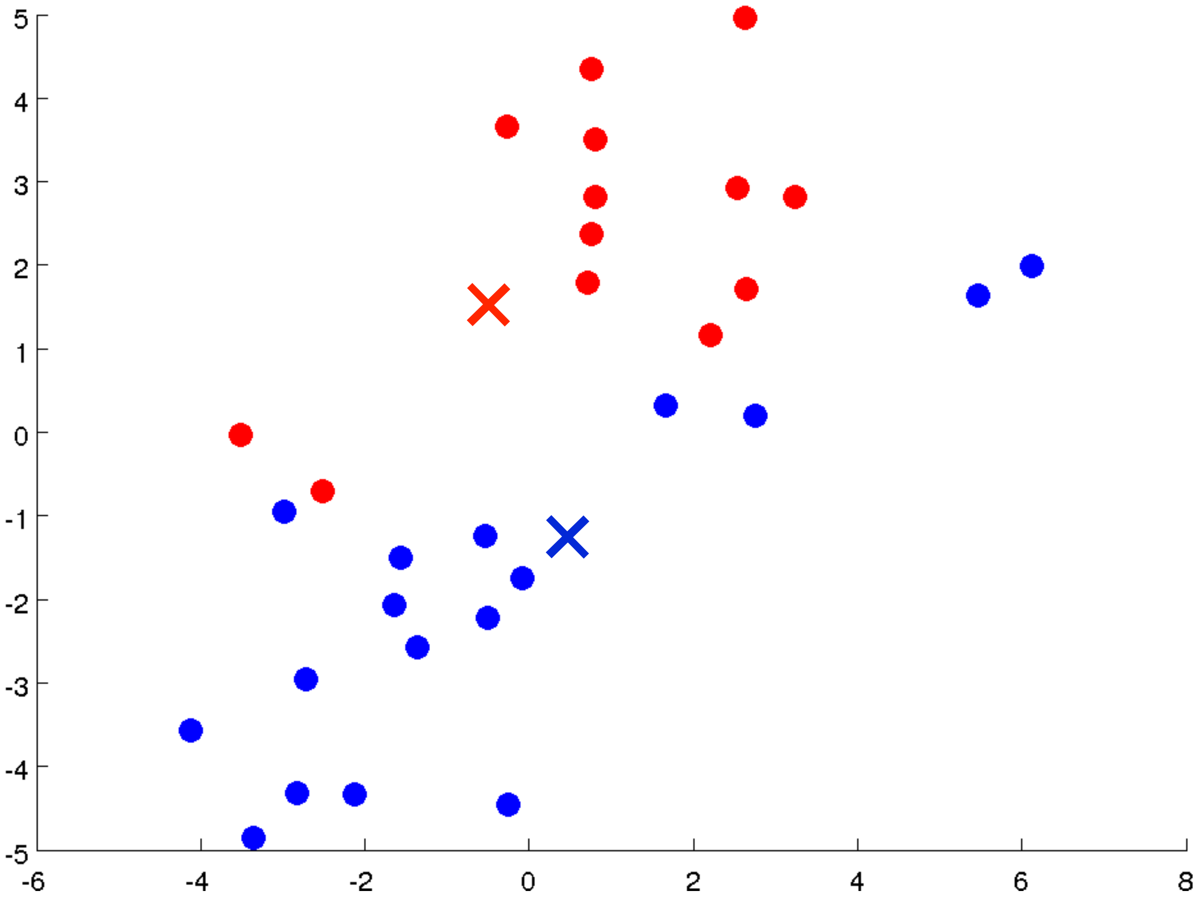
Andrew Ng

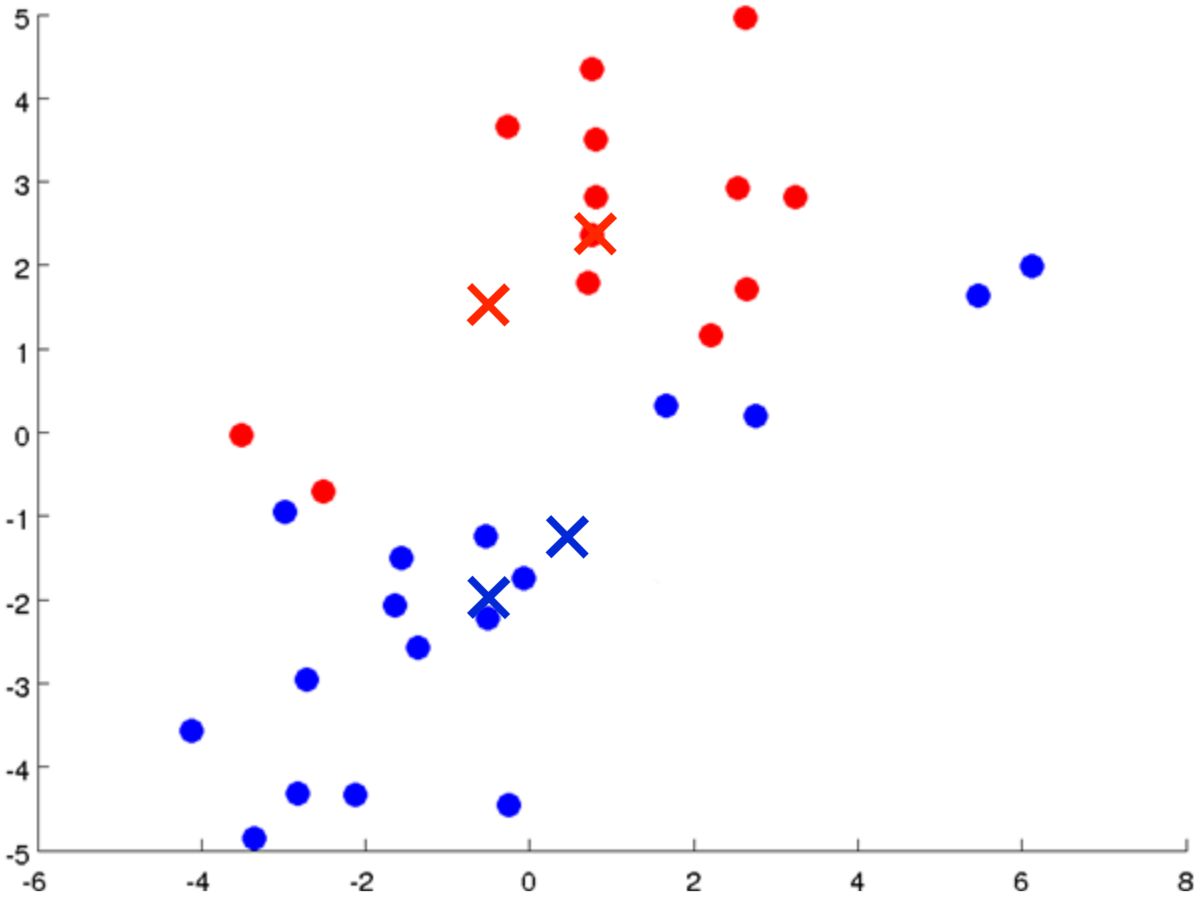
(1)cluster assignment step
*********************
it's going through each of the examples, each of these green dots shown here and depending on whether it's closer to the red cluster centroid or the blue cluster centroid, it is going to assign each of the data points to one of the two cluster centroids.

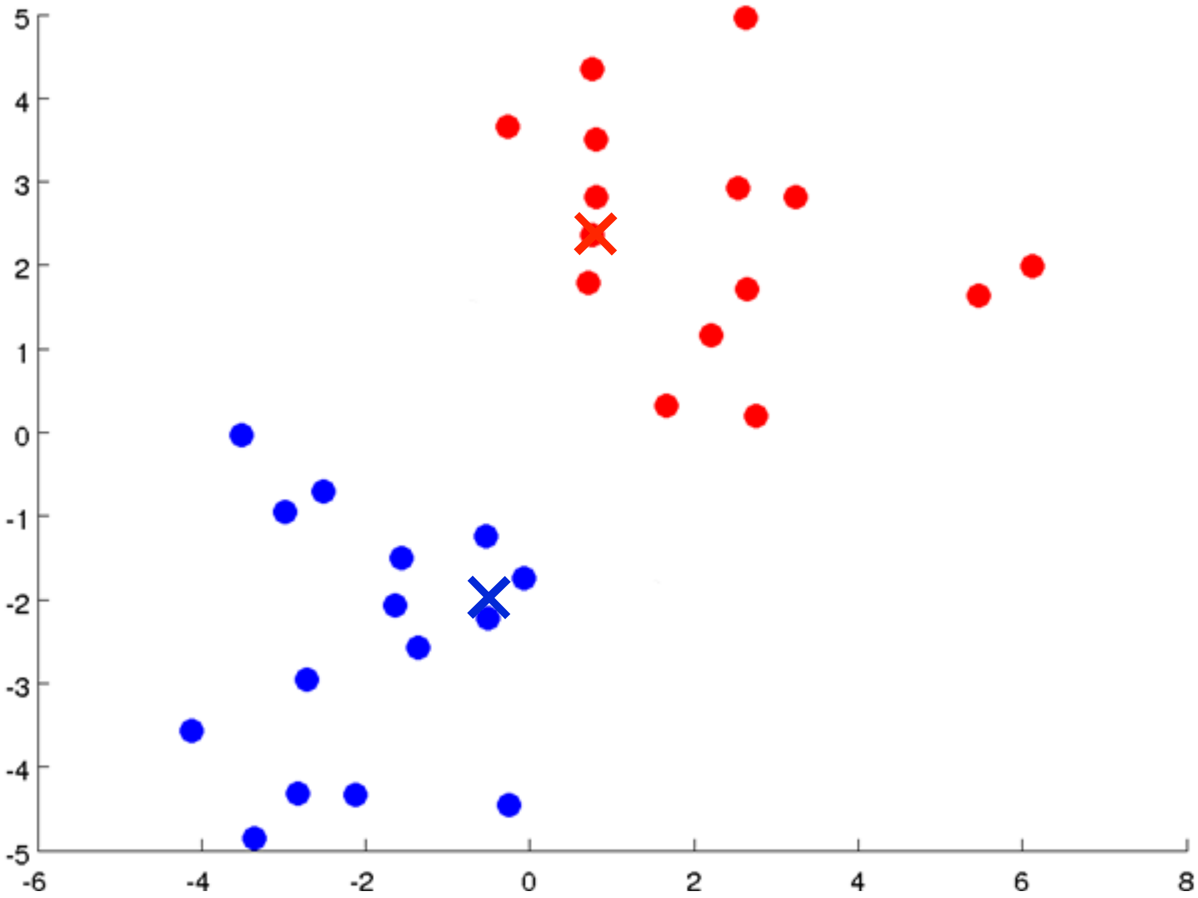


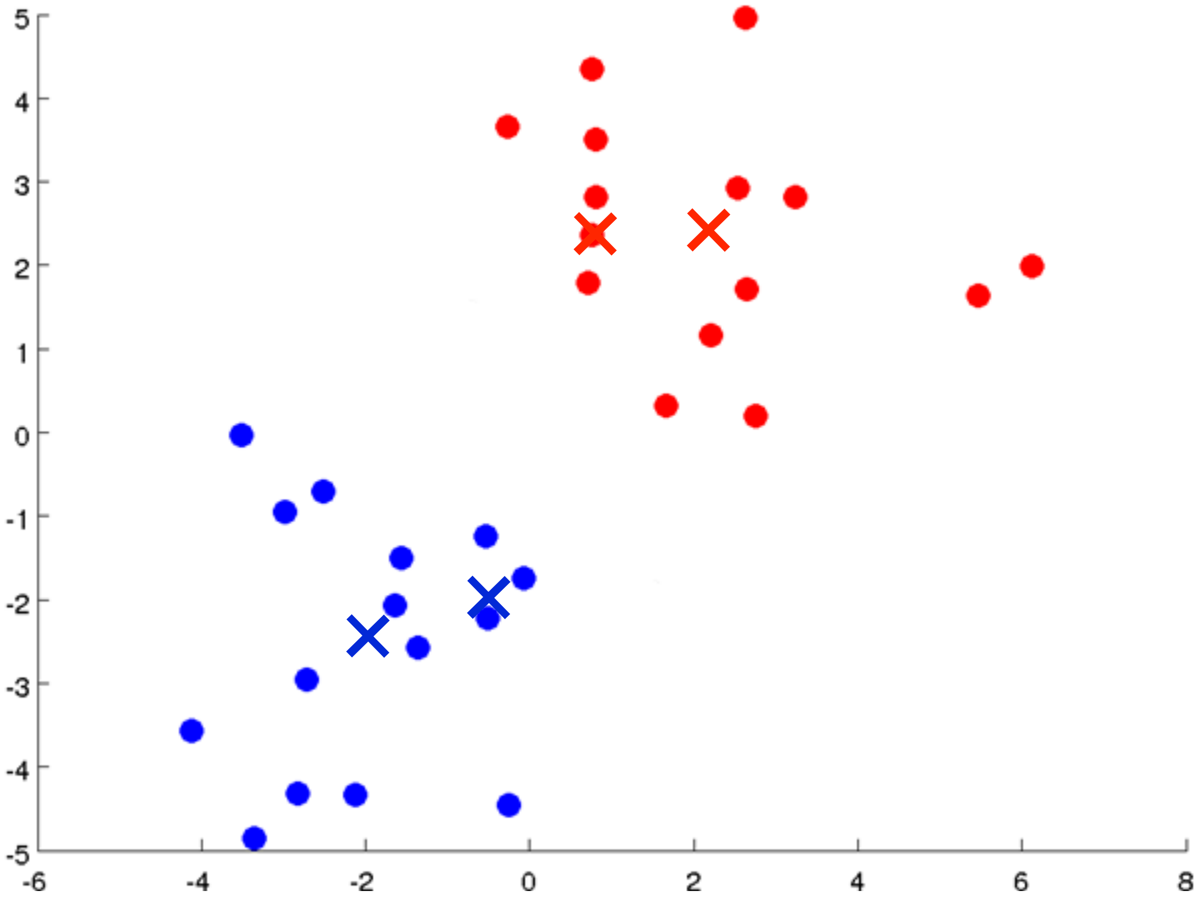
(2) move centroid step:
*********************
we are going to take the two cluster centroids, that is, the red cross and the blue cross, and we are going to move them to the average of the points colored the same colour.

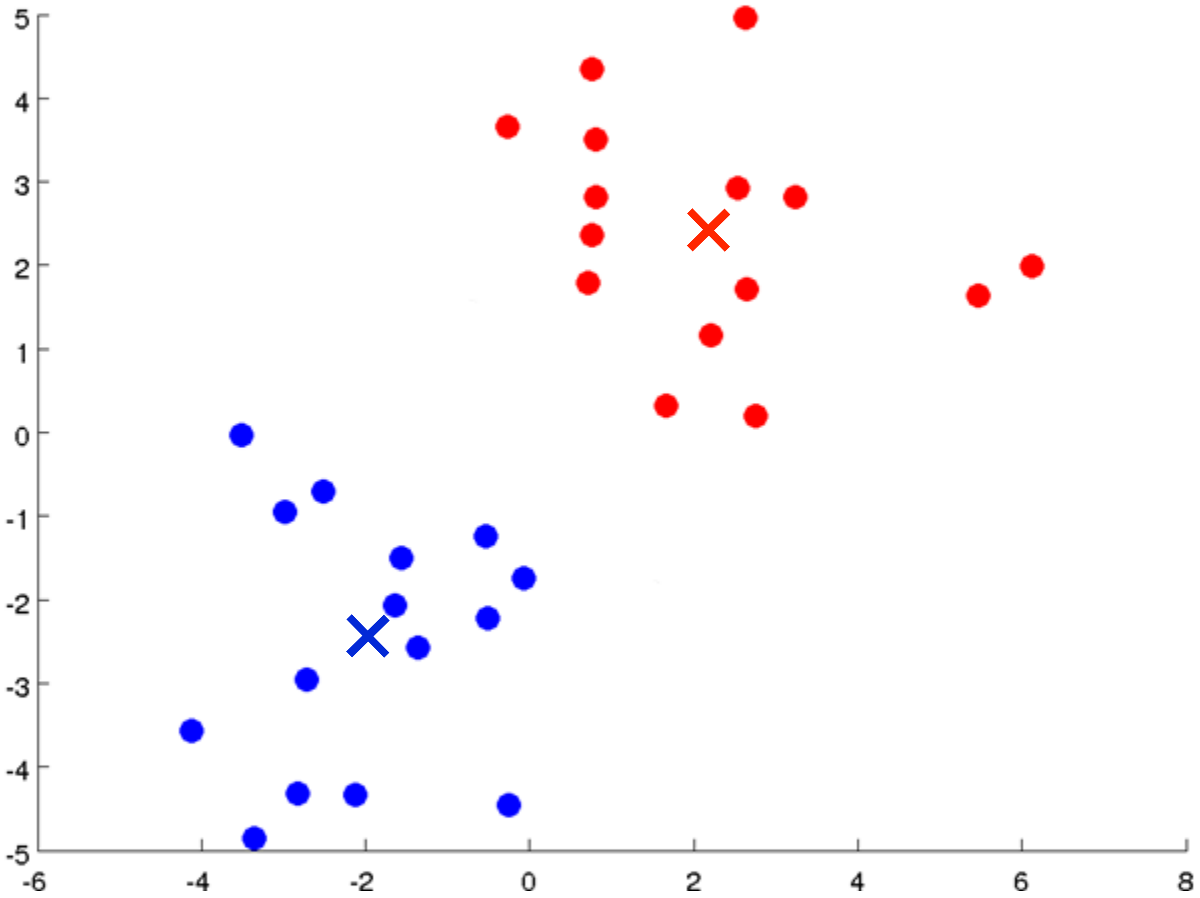



5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

# K-means algorithm

Input:

- $K$ (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

# K-means algorithm

$\mu_1$ × $\mu_2$ ×

Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment step

for $i$ = 1 to $m$

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

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

k=1 to K

Move centroid

for $k$ = 1 to $K$

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

$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)} \rightarrow c^{(1)} = 2, c^{(5)} = 2, c^{(6)} = 2, c^{(10)} = 2$

$\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right] \in \mathbb{R}^n$

}

Suppose you run k-means and after the algorithm converges, you have: $c^{(1)} = 3, c^{(2)} = 3, c^{(3)} = 5, \ldots$

Which of the following statements are true? Check all that apply.

- [x] The third example $x^{(3)}$ has been assigned to cluster 5.

Correct

- [x] The first and second training examples $x^{(1)}$ and $x^{(2)}$ have been assigned to the same cluster.

Correct

- [ ] The second and third training examples have been assigned to the same cluster.

Un-selected is correct

- [x] Out of all the possible values of $k \in \{1, 2, \ldots, K\}$ the value $k = 3$ minimizes $\|x^{(2)} - \mu_k\|^2$.

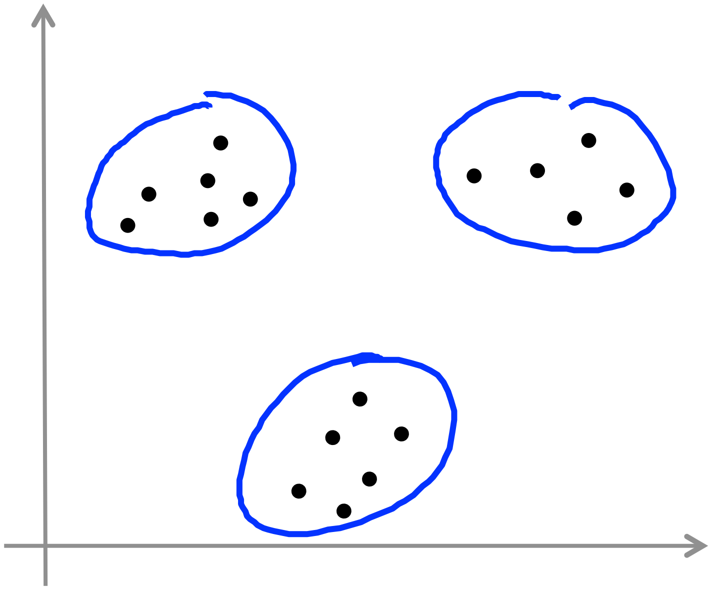
Correct

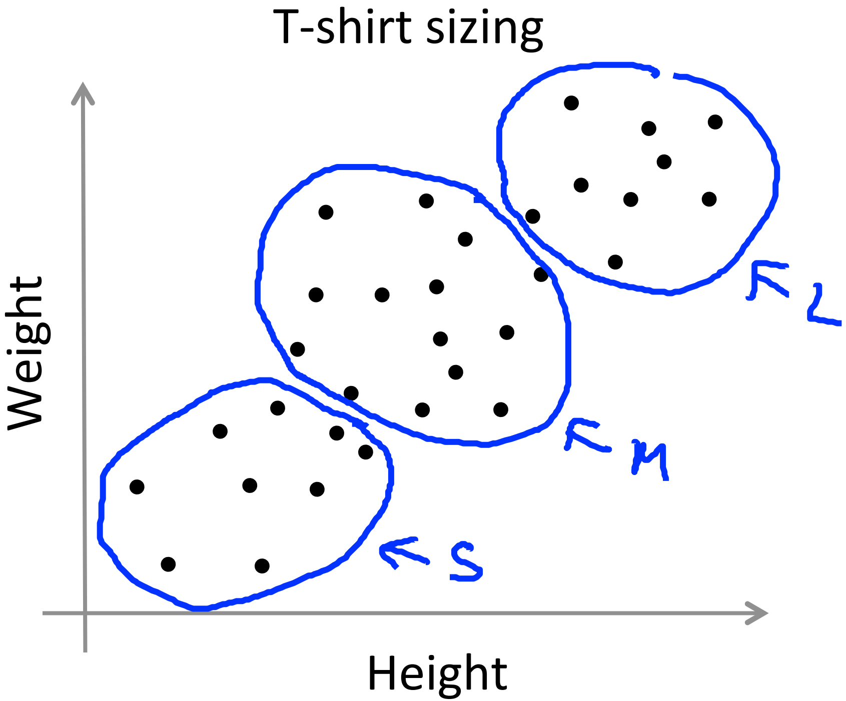
what if there is a cluster centroid no points with zero points assigned to it. In that case the more common thing to do is to just eliminate that cluster centroid. And if you do that, you end up with K minus one clusters instead of k clusters.
Sometimes if you really need k clusters, then the other thing you can do if you have a cluster centroid with no points assigned to it is you can just randomly reinitialize that cluster centroid, but it's more common to just eliminate a cluster if somewhere during K means it with no points assigned to that cluster centroid, and that can happen, although in practice it happens not that often.

# K-means for non-separated clusters

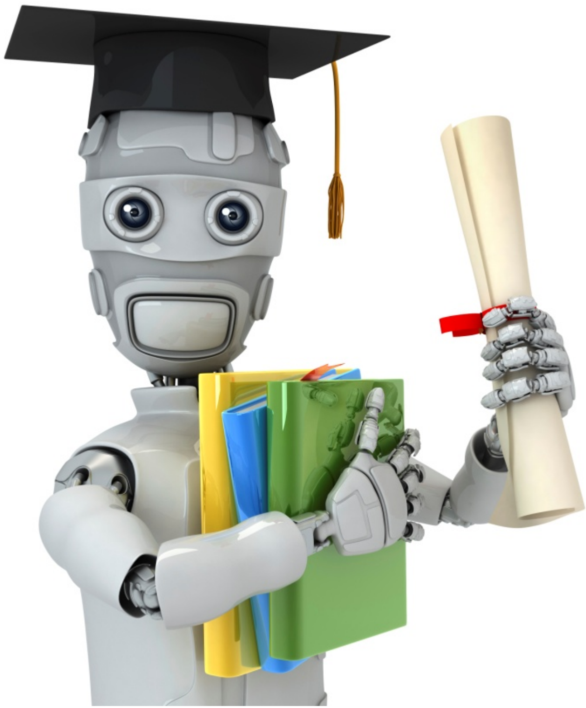
if you wanna design 3 sizes, how big should i make ?

S, M, L






where you're using K Means to separate your market into 3 different segments. So you can design a product separately that is a small, medium, and large t-shirts,



# Clustering

## Optimization objective

Most of the supervised learning algorithms we've seen, things like linear regression, logistic regression, and so on, all of those algorithms have an optimization objective or some cost function that the algorithm was trying to minimize. It turns out that k-means also has an optimization objective or a cost function that it's trying to minimize.

# K-means optimization objective

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

$\mu_k$ = cluster centroid $\underline{k}$ ($\mu_k \in \mathbb{R}^n$)   $K$   $k \in \{1, 2, \ldots, K\}$

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

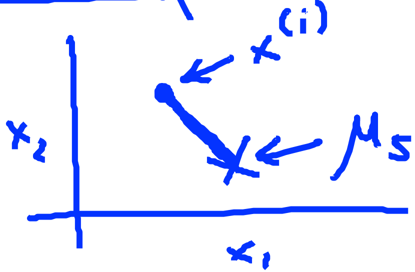
$\boxed{x^{(i)} \rightarrow \underline{5} \quad \underline{c^{(i)} = 5} \quad \mu_{\underline{c^{(i)}}} = \mu_5}$

Optimization objective:

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

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

Distortion




Andrew Ng

## K-means algorithm

Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Cluster assignment step

Minimize $J(\ldots)$ wrt $\boxed{c^{(1)}, c^{(2)}, \ldots, c^{(m)}}$ ←

(holding $\mu_1, \ldots, \mu_K$ fixed)

Repeat {

for $i$ = 1 to $m$

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

move centroid

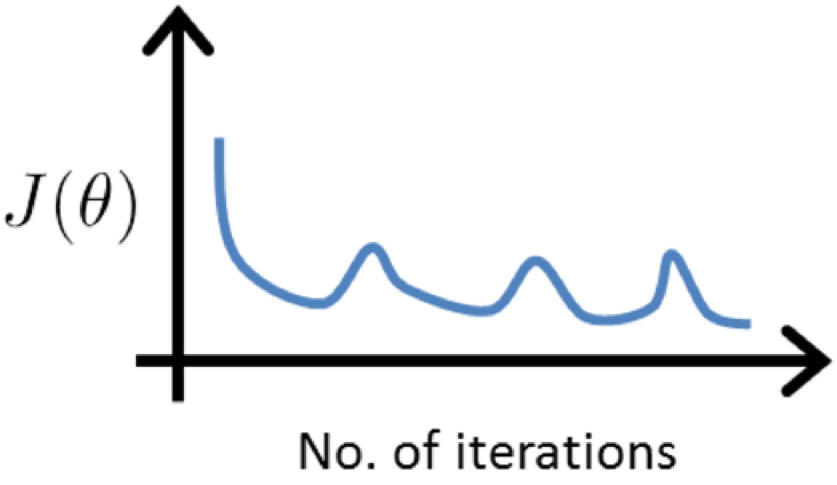
for $k$ = 1 to $K$

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

}

minimize $J(\ldots)$ wrt $\boxed{\mu_1, \ldots, \mu_K}$

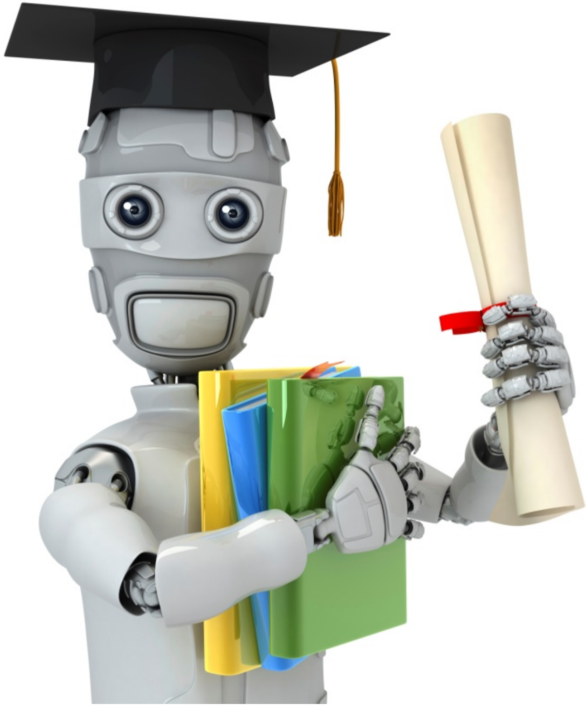
Suppose you have implemented k-means and to check that it is running correctly, you plot the cost function $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_k)$ as a function of the number of iterations. Your plot looks like this:




What does this mean?

- ◯ The learning rate is too large.
- ◯ The algorithm is working correctly.
- ◯ The algorithm is working, but $k$ is too large.
- ◉ It is not possible for the cost function to sometimes increase. There must be a bug in the code.

**Correct**



# Clustering

## Random initialization

how to initialize K-means and this will lead into a discussion of how to make K-means avoid local optima as well.

## K-means algorithm

$\boxed{\text{Randomly initialize } K}$ cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for $i$ = 1 to $m$

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

for $k$ = 1 to $K$

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

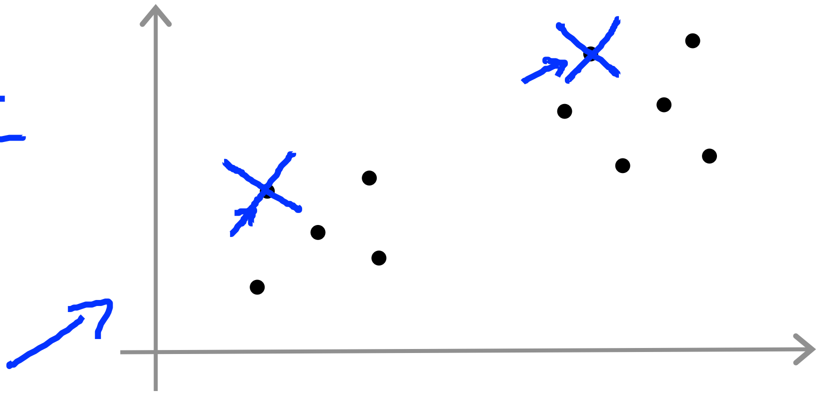
}

# Random initialization

Should have $K < m$

$K = 2$

Randomly pick $K$ training examples.

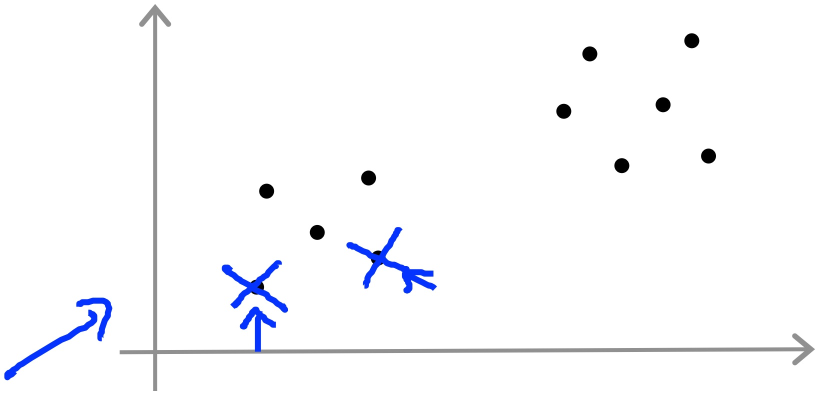


Set $\mu_1, \ldots, \mu_K$ equal to these $K$ examples.

$\mu_1 = x^{(i)}$

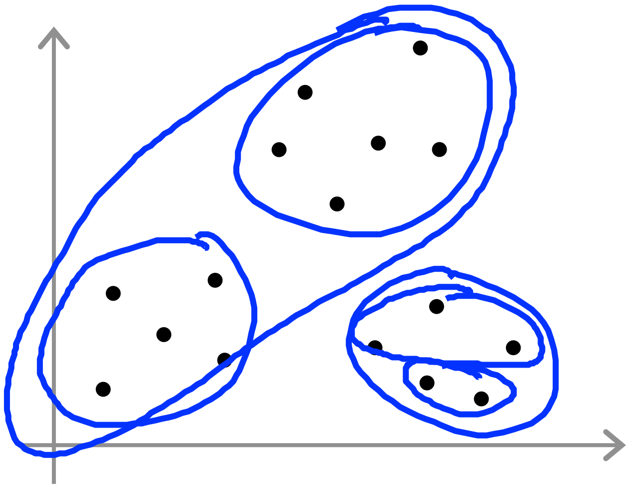
$\mu_2 = x^{(j)}$

$\vdots$

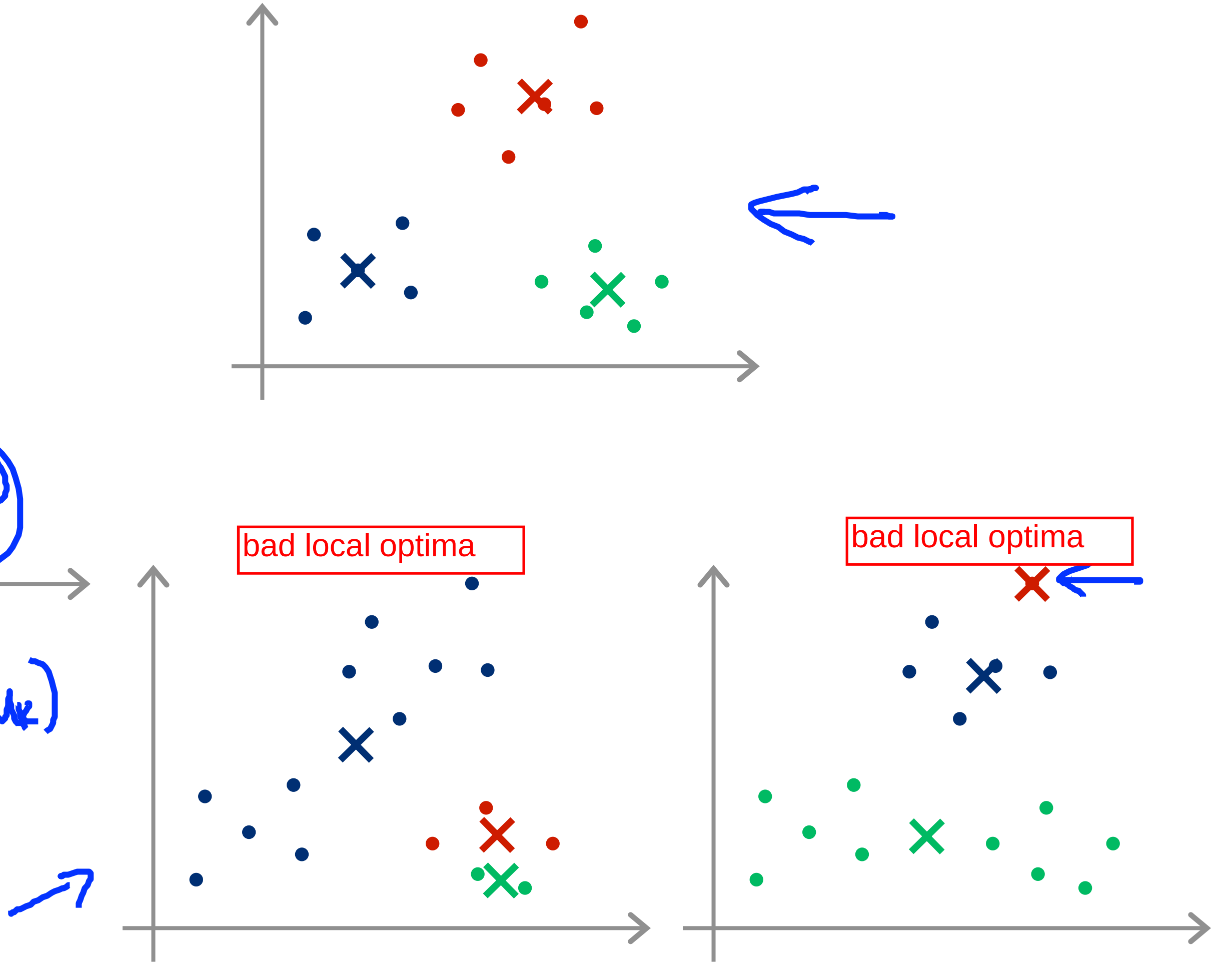


K-means can end up converging to different solutions depending on exactly how the clusters were random initialized, K-means can actually end up at local optima.

# Local optima



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



## Random initialization

if you want to increase the odds of K-means finding the best possible clustering, what we can do, is try multiple, random initializations. So, instead of just initializing K-means once and hopping that that works, what we can do is, initialize K-means lots of times and run K-means lots of times, and use that to try to make sure we get as good a solution, as good a local or global optima as possible.

For i = 1 to 100 {

Randomly initialize K-means.

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

Compute cost function (distortion)

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

}

Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

if you are running K-means with a small number of clusters (2~10), then doing multiple random initializations can often make sure that you find a better local optima. But if K is very large (> 10) then having multiple random initializations is less likely to make a huge difference and there is a much higher chance that your first random initialization will give you a pretty decent solution already and doing multiple random initializations will probably give you a slightly better solution but, but maybe not that much.

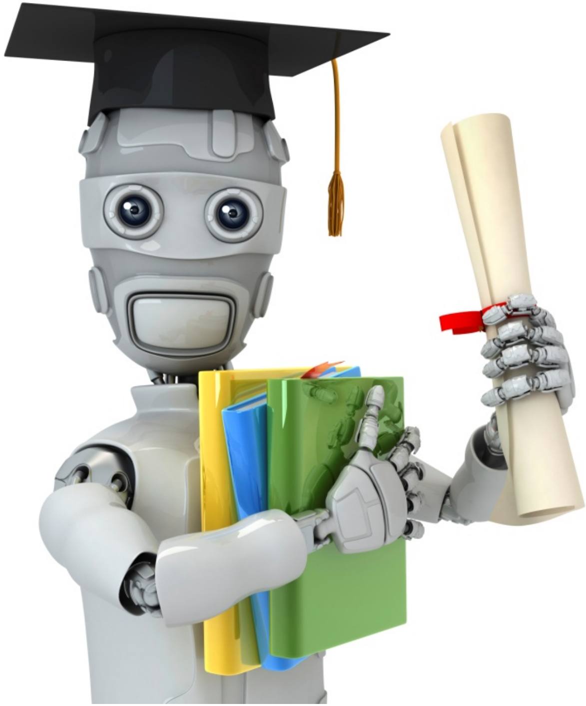
Which of the following is the recommended way to initialize k-means?

- Pick a random integer $i$ from $\{1, \ldots, k\}$. Set $\mu_1 = \mu_2 = \cdots = \mu_k = x^{(i)}$.
- Pick $k$ distinct random integers $i_1, \ldots, i_k$ from $\{1, \ldots, k\}$.

  Set $\mu_1 = x^{(i_1)}, \mu_2 = x^{(i_2)}, \ldots, \mu_k = x^{(i_k)}$.
- Pick $k$ distinct random integers $i_1, \ldots, i_k$ from $\{1, \ldots, m\}$.

  Set $\mu_1 = x^{(i_1)}, \mu_2 = x^{(i_2)}, \ldots, \mu_k = x^{(i_k)}$.

  **Correct**
- Set every element of $\mu_i \in \mathbb{R}^n$ to a random value between $-\epsilon$ and $\epsilon$, for some small $\epsilon$.



# Clustering

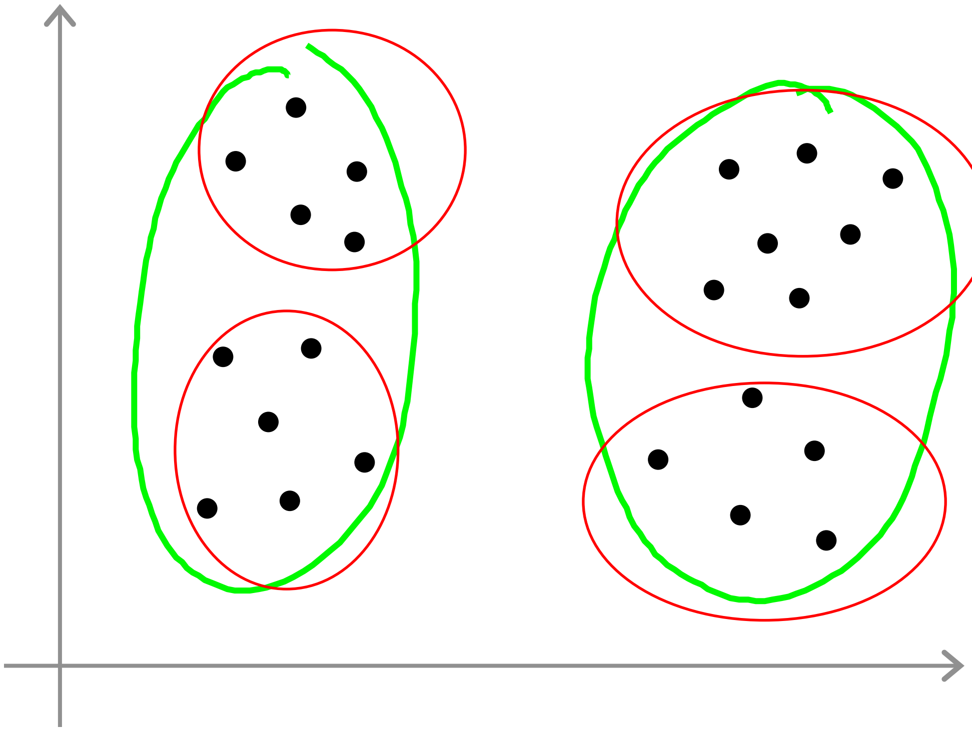
## Choosing the number of clusters

Machine Learning

there actually isn't a great way of answering this or doing this automatically and by far the most common way of choosing the number of clusters, is still choosing it manually by looking at visualizations or by looking at the output of the clustering algorithm or something else.

# What is the right value of K?

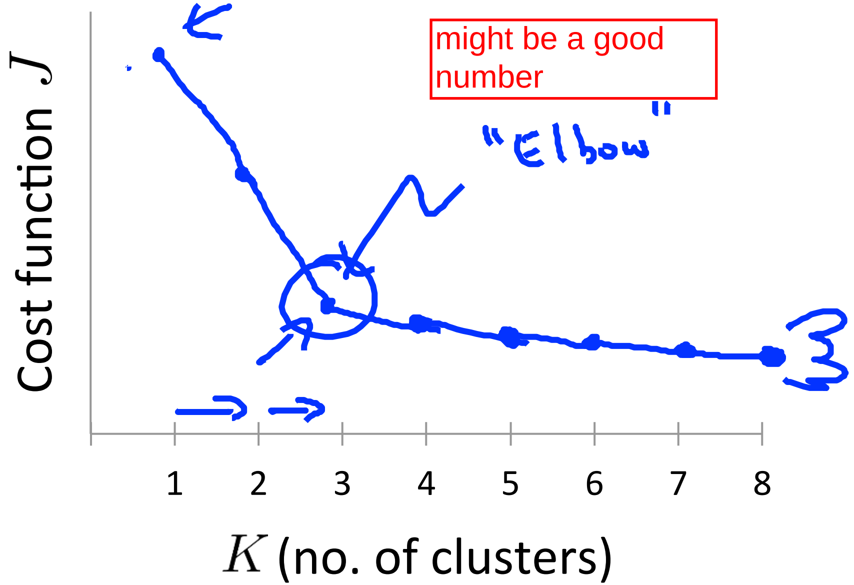
2, 3 or 4 clusters

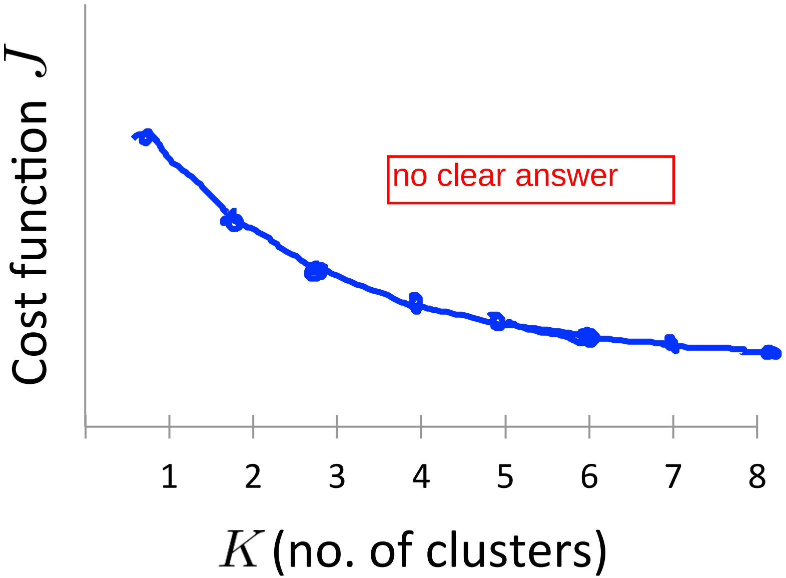


We are aren't given labels, and so there isn't always a clear cut answer. And this is one of the things that makes it more difficult to have an automatic algorithm for choosing how many clusters to have.

# Choosing the value of K

Elbow method:




the quick summary of the Elbow Method is that is worth the shot but I wouldn't necessarily, have a very high expectation of it working for any particular problem.

Suppose you run k-means using k = 3 and k = 5. You find that the cost function J is much higher for k = 5 than for k = 3. What can you conclude?

- This is mathematically impossible. There must be a bug in the code.
- The correct number of clusters is k = 3.
- In the run with k = 5, k-means got stuck in a bad local minimum. You should try re-running k-means with multiple random initializations.

  **Correct**

- In the run with k = 3, k-means got lucky. You should try re-running k-means with k = 3 and different random initializations until it performs no better than with k = 5.

## Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

K=3 S, M, L

K=5 XS, S, M, L, XL

E.g.