

Machine Learning

Week 8

Section 1

Unsupervised Learning

This week, you will be learning about unsupervised learning. While supervised learning algorithms need labeled examples (x,y) , unsupervised learning algorithms need only the input (x) . You will learn about clustering—which is used for market segmentation, text summarization, among many other applications.

We will also be introducing Principal Components Analysis, which is used to speed up learning algorithms, and is sometimes incredibly useful for visualizing and helping you to understand your data.

Clustering

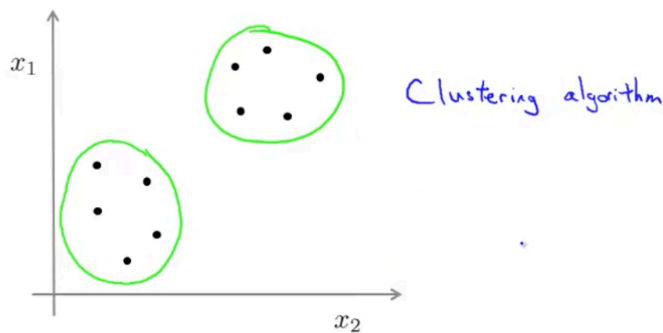
Unsupervised Learning: Introduction

Cluster is our first unsupervised learning algorithm where we learn from unlabeled data.

We give the unsupervised learning algorithm this unlabeled data and it will find some structure in the data for us.

In this example we can see 'clustering' patterns in the data:

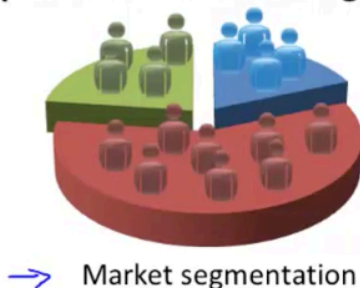
Unsupervised learning



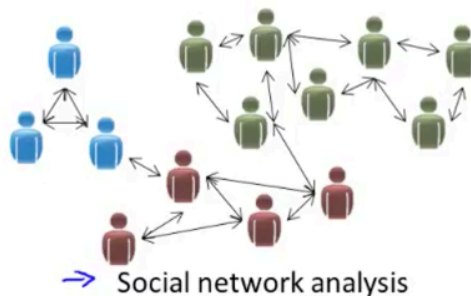
Training set: $\{\underline{x^{(1)}}, \underline{x^{(2)}}, x^{(3)}, \dots, \underline{x^{(m)}}\}$ ←

Some example applications of clustering:

Applications of clustering



→ Market segmentation



→ Social network analysis



Organize computing clusters



Astronomical data analysis

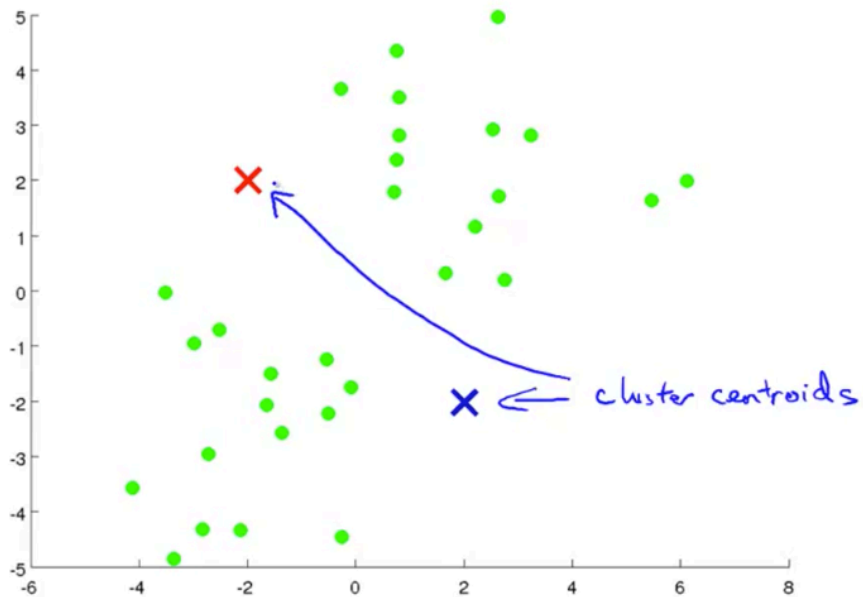
K-means Algorithm

The K-means algorithm is by far the most popular and widely used clustering algorithm.

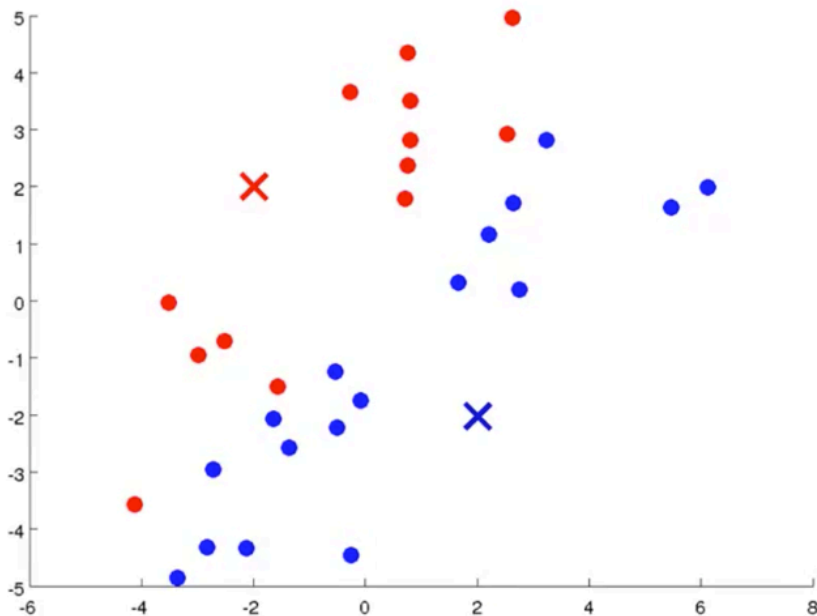
It is an iterative algorithm that works in two steps, the first is a cluster assignment step, the second is a move centroid step.

It works as follows:

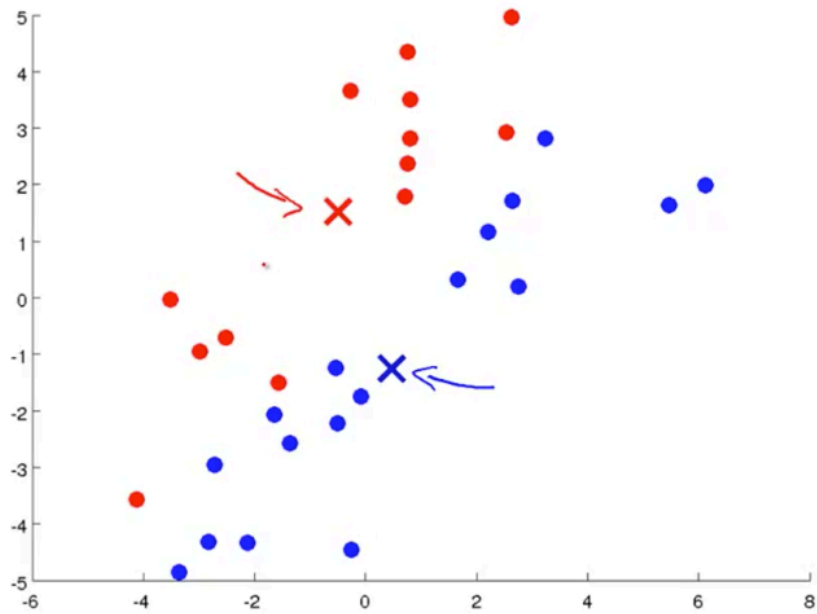
Start: Randomly initialize two points called the cluster centroids. (Blue and red Xs)



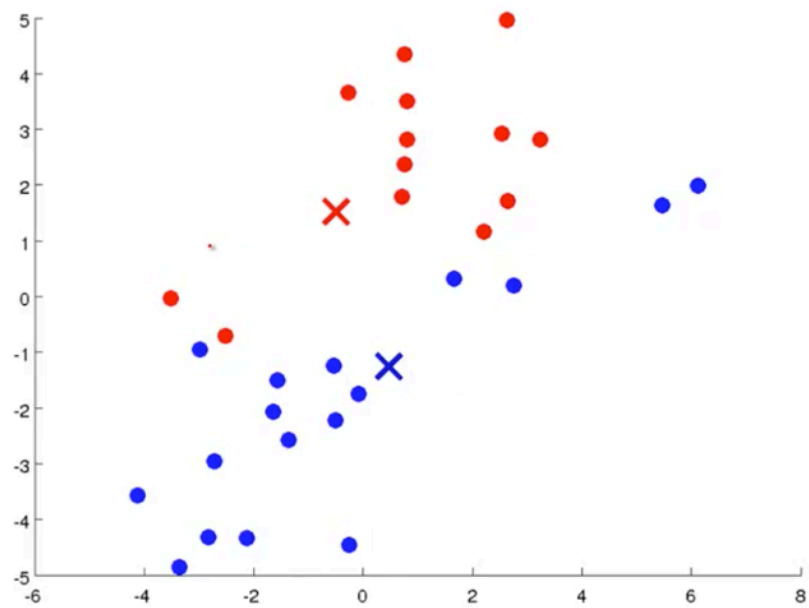
Cluster Assignment Step: Go through all points and mark each point as closer to either the red or blue X:



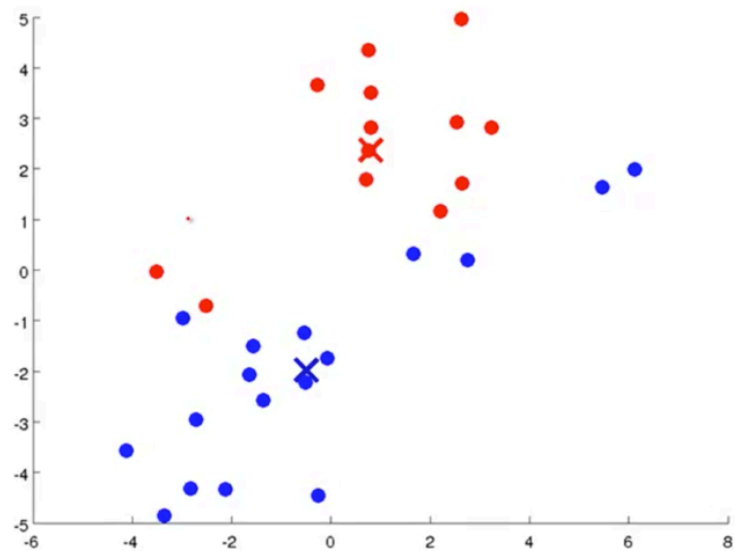
Move Centroid Step: Look at all the red dots, compute their mean, and then move the red centroid there. Same for blue.



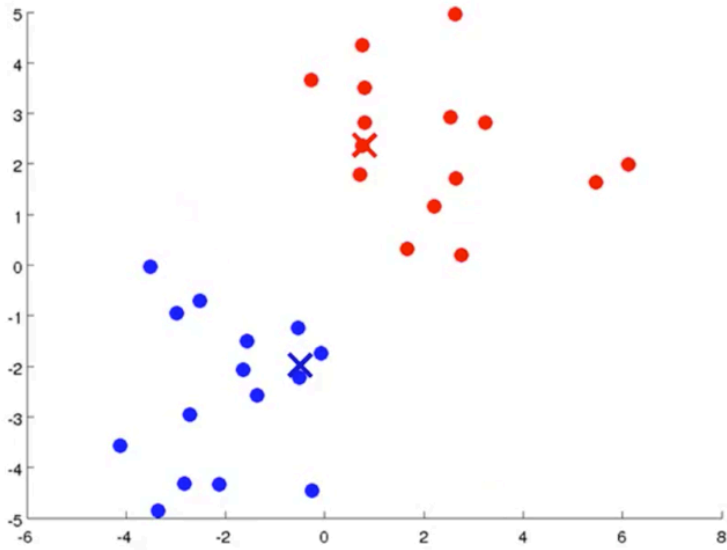
Repeat the Cluster Assignment Step:



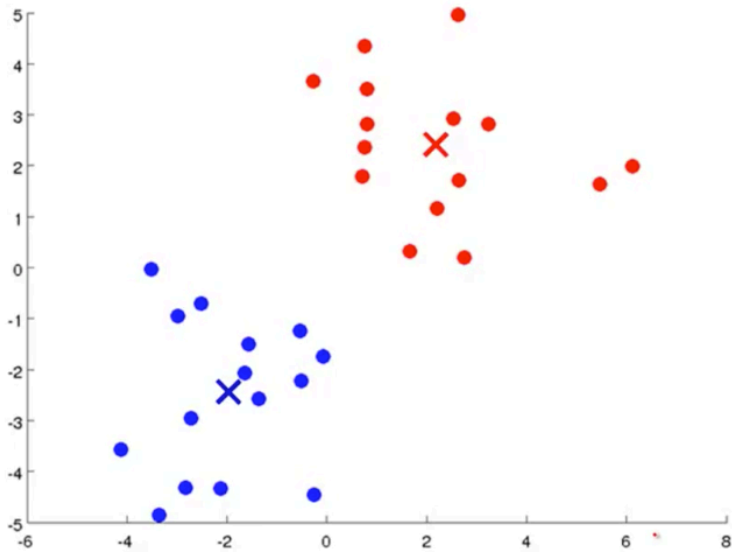
Repeat the Move Centroid Step:



Repeat the Cluster Assignment Step:



Repeat the Move Centroid Step:



And we are done. If the algorithm continues, the centroids won't move and the colors won't be reassigned.

=> K-means has converged.

Formally, the K-means algorithm is as follows:

Input:

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

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

K-means algorithm

μ_1 μ_2
x x

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

Repeat {

Cluster assignment step

for $i = 1$ to m

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

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

Move centroid

for $k = 1$ to K

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

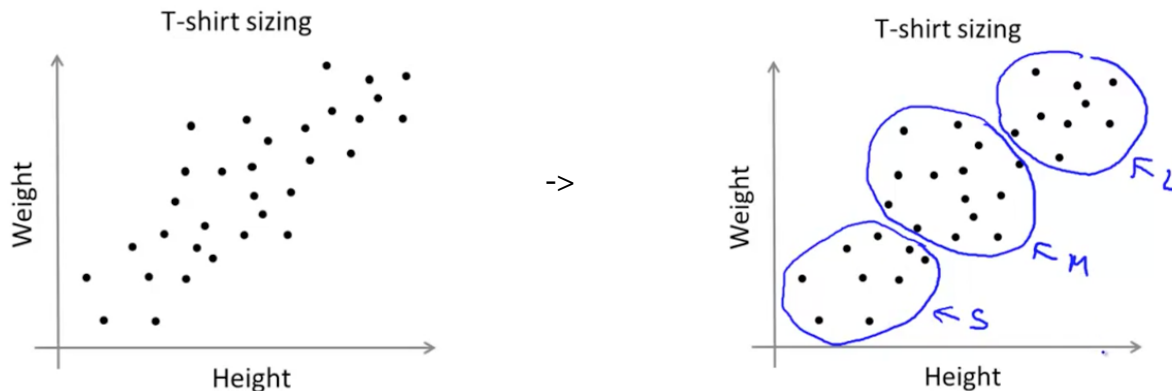
$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$

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

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

If for some reason there aren't any points assigned to a cluster k during the move centroid step, you can either get rid of that cluster and have $(K-1)$ -Means or reinitialize the algorithm with different initialize cluster centroid positions.

Another example application of K-means is for non-separated clusters:



This is a basic example of market segmentation.

Optimization Objective (For K-means algorithm)

What is the optimization objective for K-means algorithm?

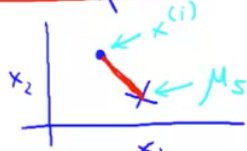
Note: While K-means is running, we are going to need to keep track of the following:

- $c^{(i)}$ = index of cluster $(1, 2, \dots, K)$ to which example $x^{(i)}$ is currently assigned
 - μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$) $k \in \{1, 2, \dots, K\}$
 - $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned
- $x^{(i)} \rightarrow 5$ $c^{(i)} = 5$ $\mu_{c^{(i)}} = \mu_5$

Our optimization objective of the K-means clustering algorithm is:

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

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



Sometimes this cost function J is also referred to as the Distortion of the K means algorithm.

K-means algorithm is as follows:

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

Repeat {

Cluster assignment step
Minimize $J(\dots)$ w.r.t $c^{(1)}, c^{(2)}, \dots, c^{(m)}$ (holding μ_1, \dots, μ_K fixed)

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 $J(\dots)$ w.r.t μ_1, \dots, μ_K

Random Initialization

How do we initialize k-means and avoid local optima?

We have not yet discussed randomly initializing the cluster centroids.

The best method (most recommend):

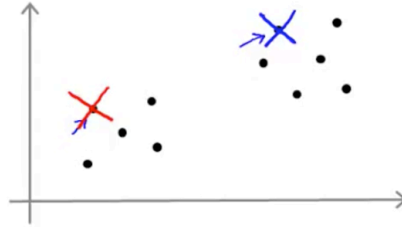
Random initialization

Should have $K < m$

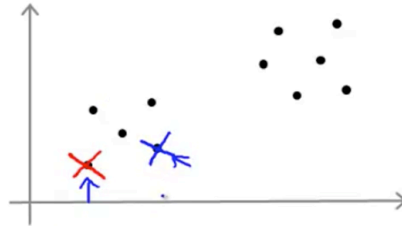
Randomly pick K training examples.

Set μ_1, \dots, μ_K equal to these K examples.

$K=2$



This first graph shows a good case in which we got 'lucky' with our random initializations.

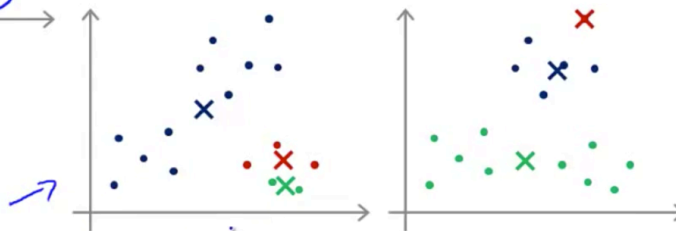
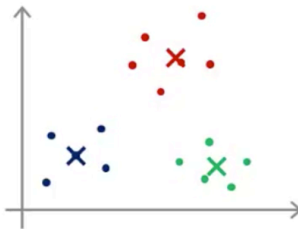
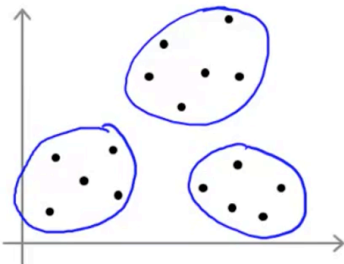


This second graph is less 'lucky' of a random initialization.

** Note in the first video where we saw an example of the algorithm, we used a different method by initializing two random points that weren't even elements of the training data set.

It is also possible for K-means to fall into a local optima if you get an 'unlucky' random initialization.

Local optima



Recall these local optima mean we have found a solution for $J(c \dots, \mu \dots)$ which don't do a good job minimizing the cost function J

If you want to increase the odds of K-means finding the best possible clustering, you can try multiple random initializations.

Typically we do the following: Iterate through many different random initializations, try different K

Random initialization

For $i = 1$ to 100 { 50 - 1000

 → Randomly initialize K-means.
 Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.
 Compute cost function (distortion)
 → $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$
 }

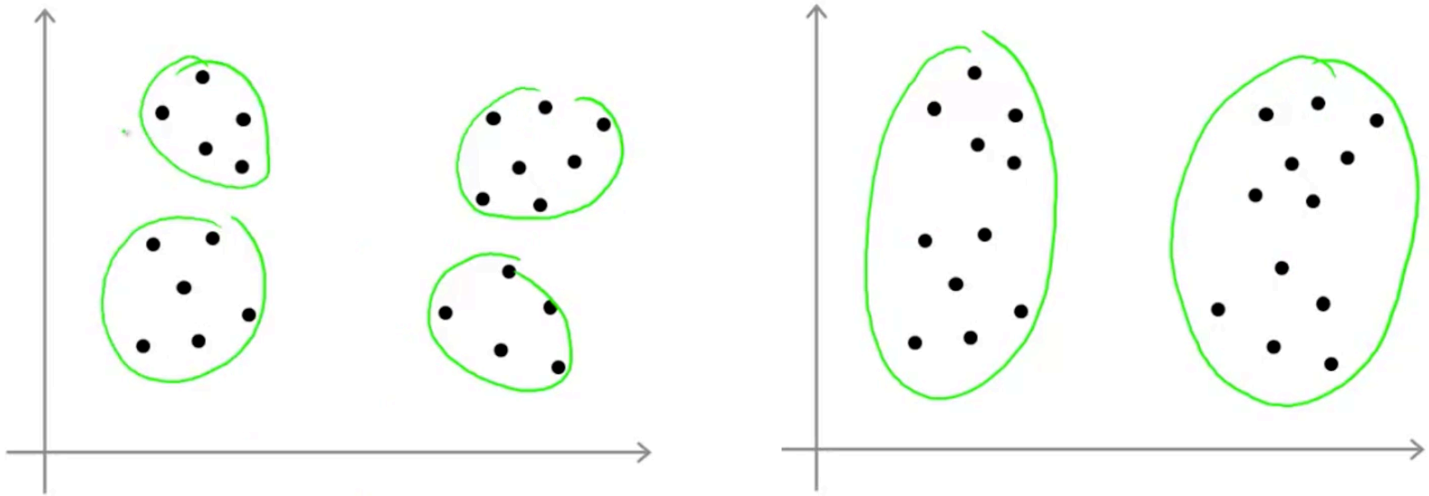
Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$
 $K = 2 - 10$ ↑

Choosing the Number of Clusters

How do we choose the number of clusters K ?

There isn't a great way to do this automatically, usually we try a few and check them manually. (Visually?)

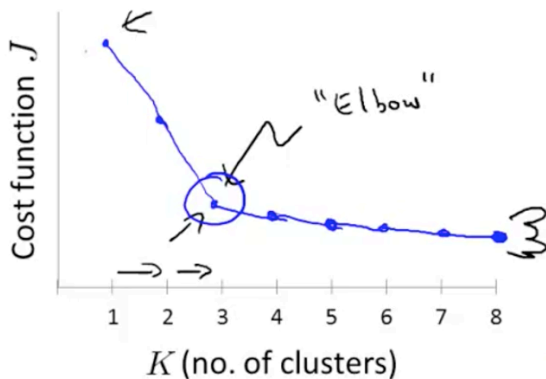
Ex. This could be 2, 3, or 4 clusters:



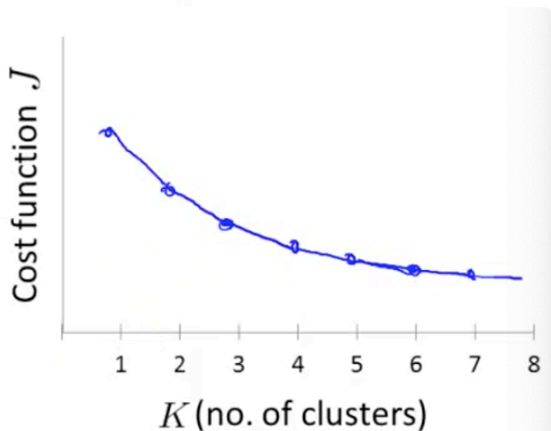
This is really one of the subjective parts of unsupervised learning, there isn't always a clear cut answer.

One possible method to choosing the number of clusters is called the Elbow Method.

Elbow method:



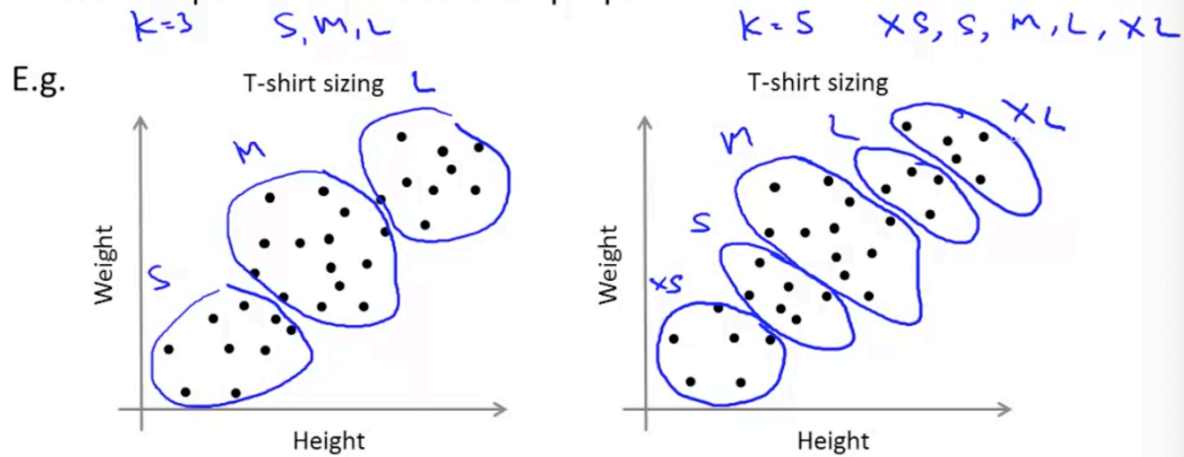
We can choose K in the 'Elbow' of the trend where using a higher K doesn't decrease the distortion much more.



In reality however we typically can't find a clear 'Elbow' which doesn't help us much in choosing a value of K .

One last way we can possibly use to find a value 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.



I.e. think in the perspective of the application / problem.

** For this weeks programming assignment we tackle an image compression problem, so in that instance we could use that metric of the quality of the image vs the file size to choose the number of clusters K.