13. Unsupervised Learning – Clustering

-- When we don't have the output of our training examples… we just have different input features… this is called UNLABELLED DATASET.

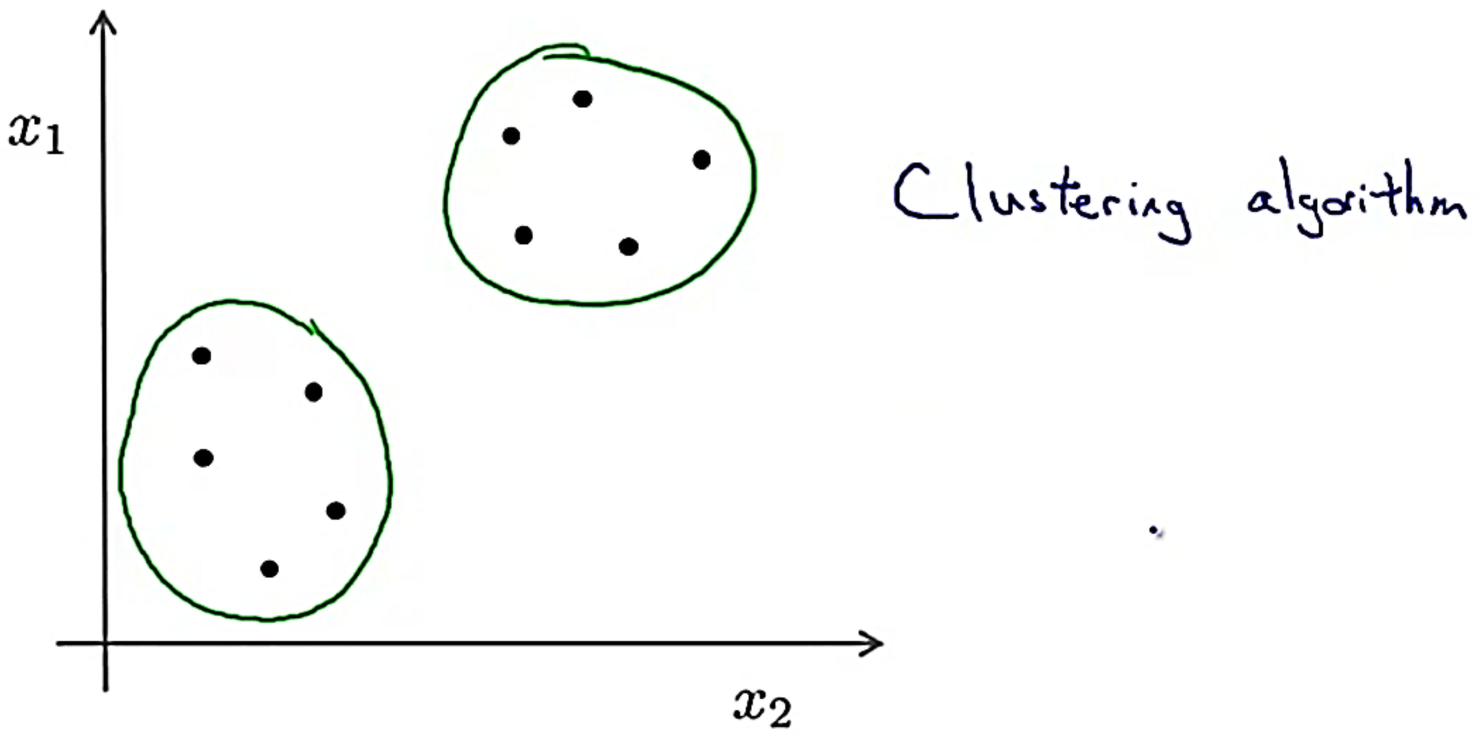
All we want is to group those inputs into different **clusters**

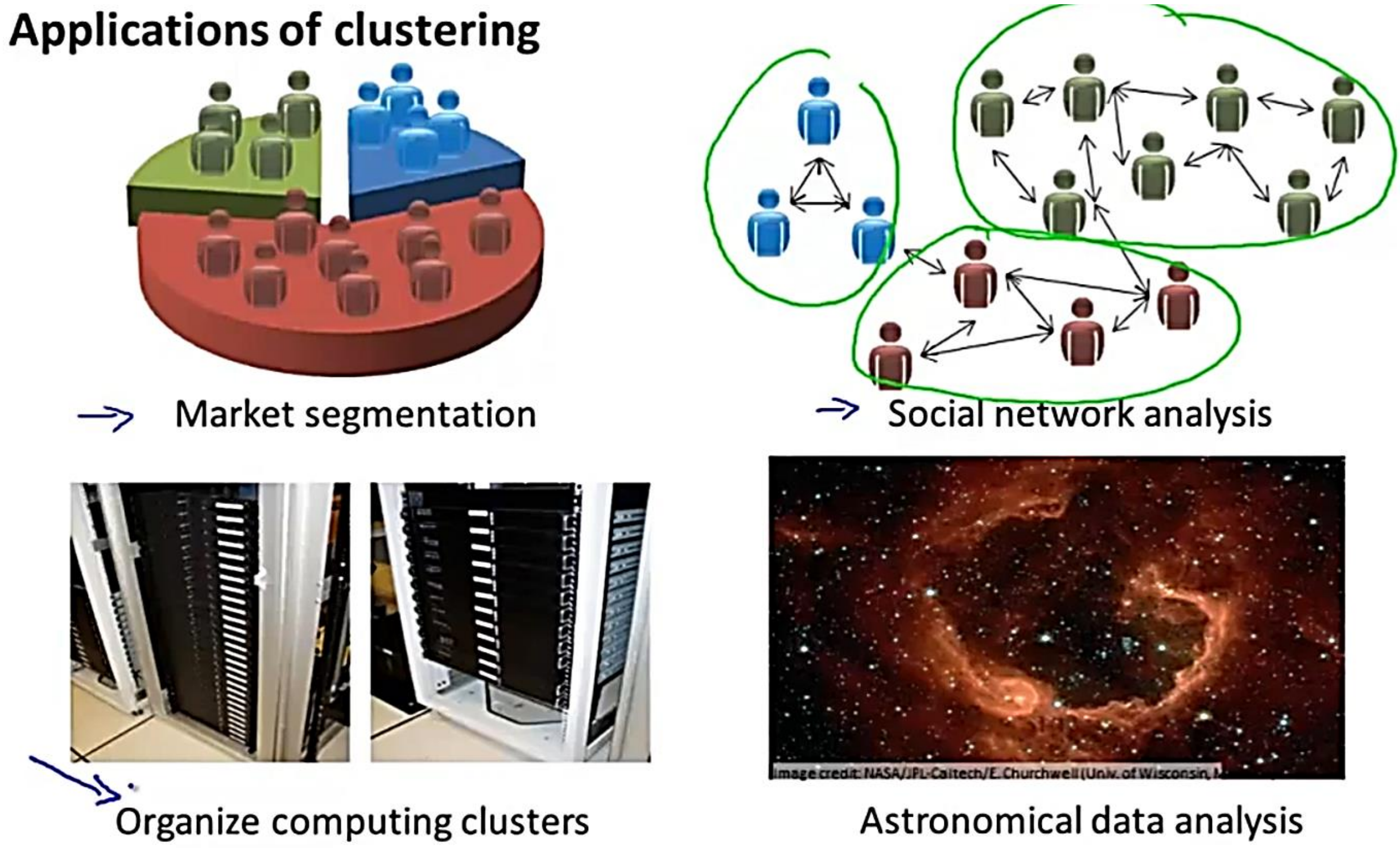
≫ In SUPERVISED LEARNING:

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

≫ In UNSUPERVISED LEARNING:

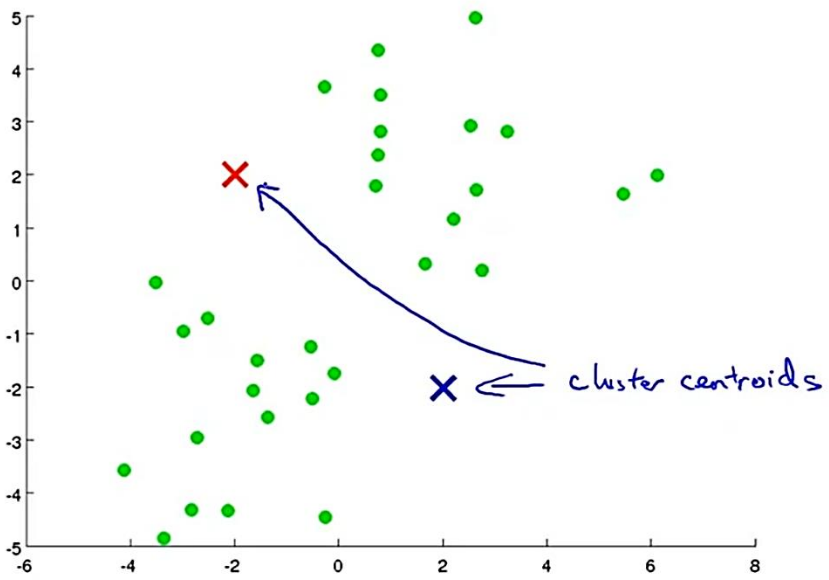
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$



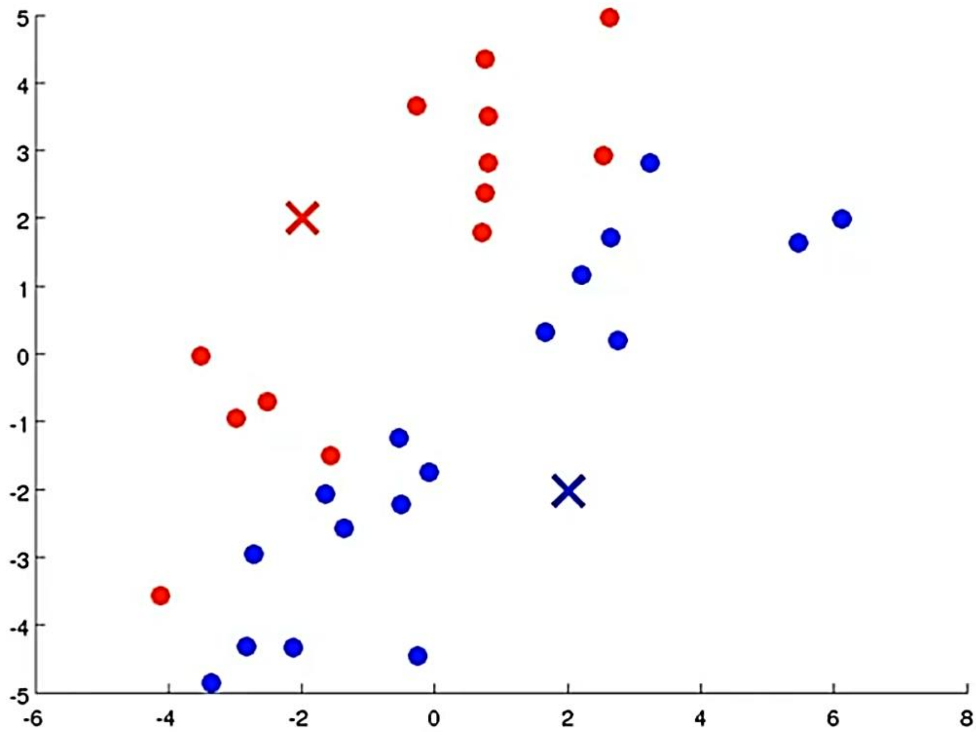


≫ K-MEANS ALGORITHM:

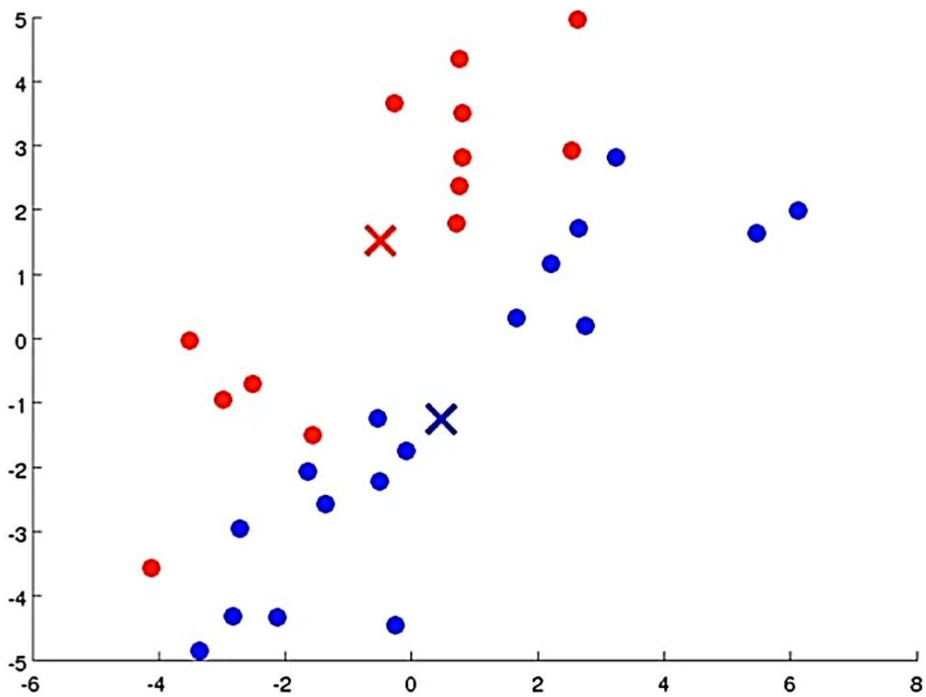
- ➔ First, we **randomly** initialize two **cluster centroids** in the data plot:



➔ Now, we assign each data point to one of the cluster centroids, whichever is **closer**

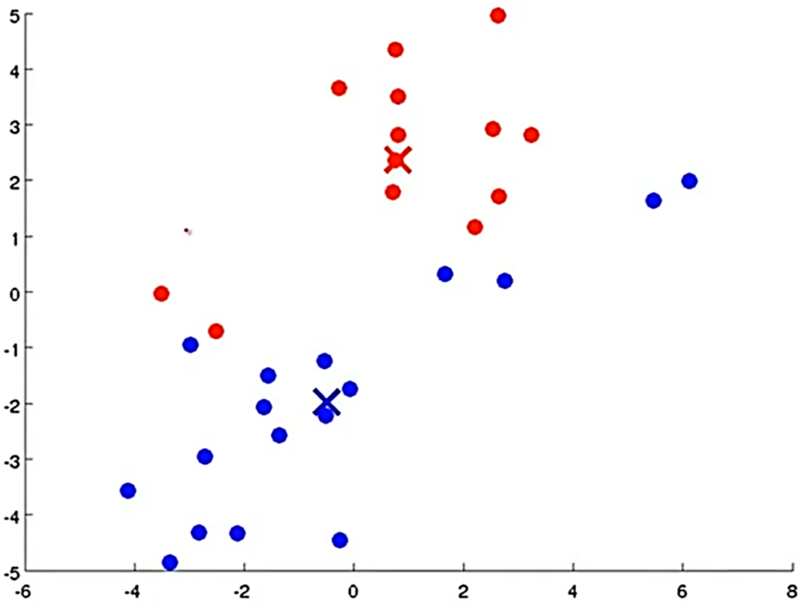


➔ Next, we **move the cluster centroids** to the **average** of that colour points

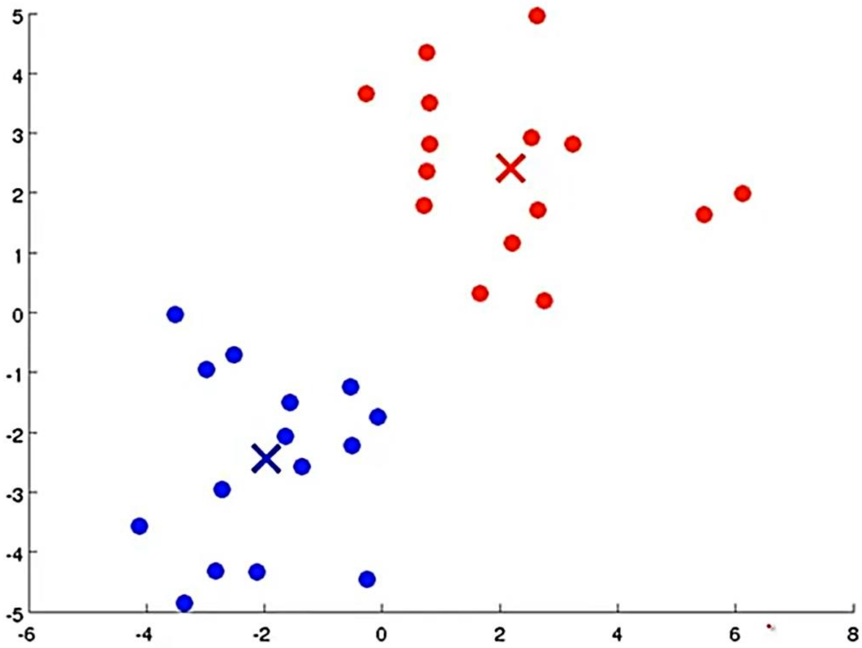


➔ Next, we **re-assign** each data point to one of the cluster centroids…

➔ Then we move the cluster centroids again.. to the average of that colour points



➔ We repeat these steps, until the cluster centroids **remain at the same point**, with each iteration.



ALGORITHM:

K-means algorithm

$\mu_1 \times \quad \mu_2 \times$

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 $x^{(i)}$

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

Move centroid

for k = 1 to K

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

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

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

}

In "cluster assignment" step → We assign each example to a cluster:

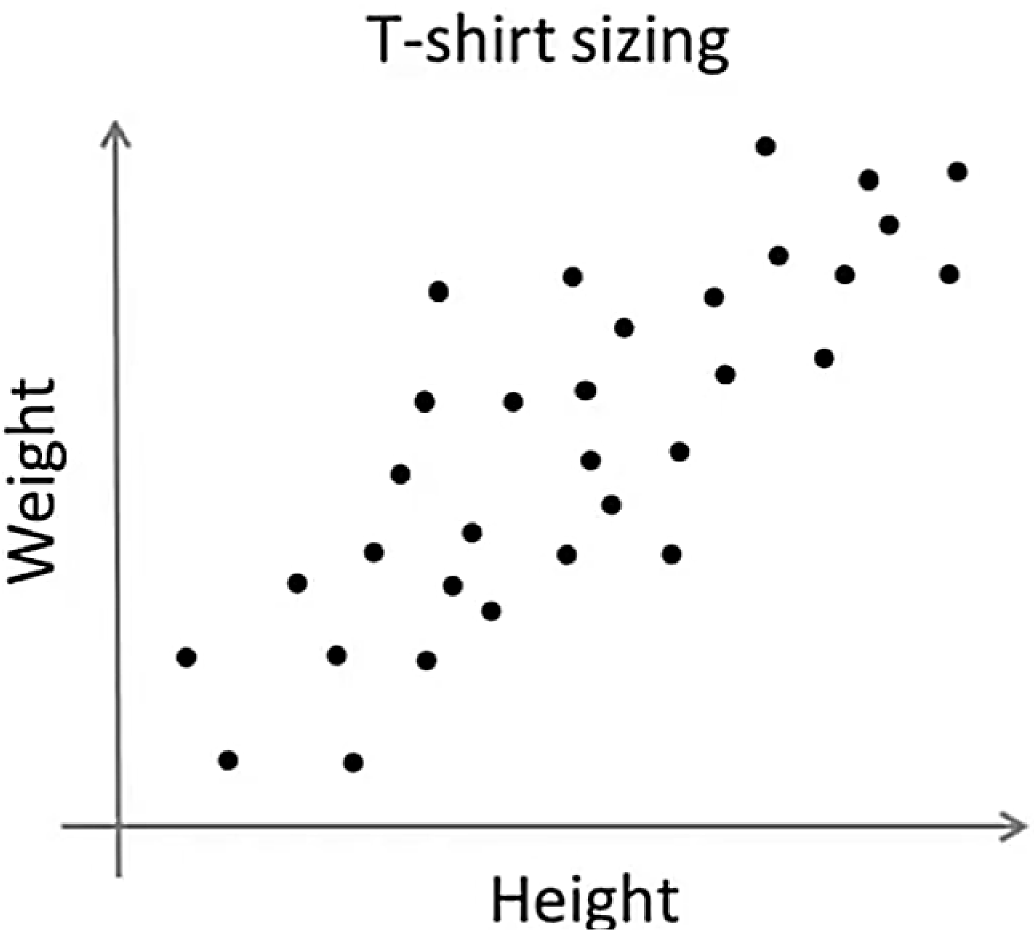
$c^{(i)}$ → holds the value from 1 to K.. whichever gives the smallest value of:

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

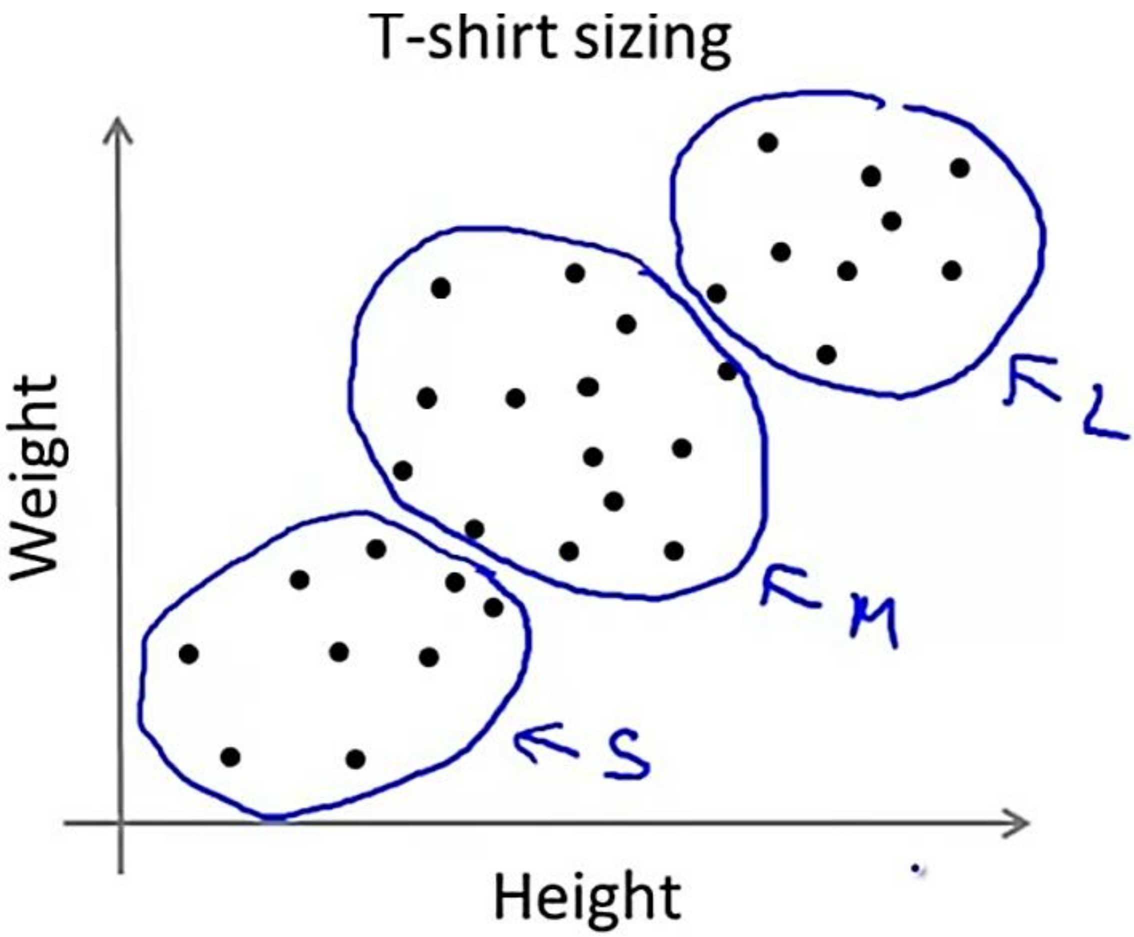
In "move centroid" step → We find average of all the points assigned to that centroid.

Each μ_k → kth cluster centroid → is an **n-dimensional vector** → corresponding to no of features.

K-MEANS FOR NON-SEPARATED CLUSTERS:



The algo will make 3 clusters → Small, Medium, Large



OPTIMIZATION OBJECTIVE OF K-MEANS ALGORITHM:

K-means optimization objective

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

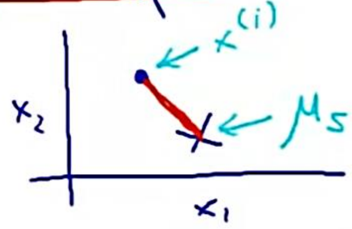
→ μ_k = cluster centroid $\underline{k}$ ($\mu_k \in \mathbb{R}^n$) K $k \in \{1, 2, \ldots, K\}$

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned $x^{(i)} \to \underline{5}$ $\underline{c^{(i)} = 5}$ $\mu_{c^{(i)}} = \mu_5$

Optimization objective:

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

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



Here, the algo is trying to **minimize the squared distance** b/w data points and the cluster centroid assigned to them. → **by changing the values of "c" and "μ".**

- ➢ We are choosing the value of **"c"** for each data point which is minimum for that data point.
- ➢ Then we are finding the value of "**μ**" for each centroid so that we can move the centroid.

The cost fxn **J(c, μ)** is also called the **Distortion Cost function**

So, what the algorithm is actually doing is:

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

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

}

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

This means:

→ **In the cluster assignment step:** we are **minimizing J wrt c** so as to choose a centroid for each data point

→ **In move centroid step:** we are **minimizing J wrt μ** so as to find the mean of points associated with each centroid and move the centroid to that point

---

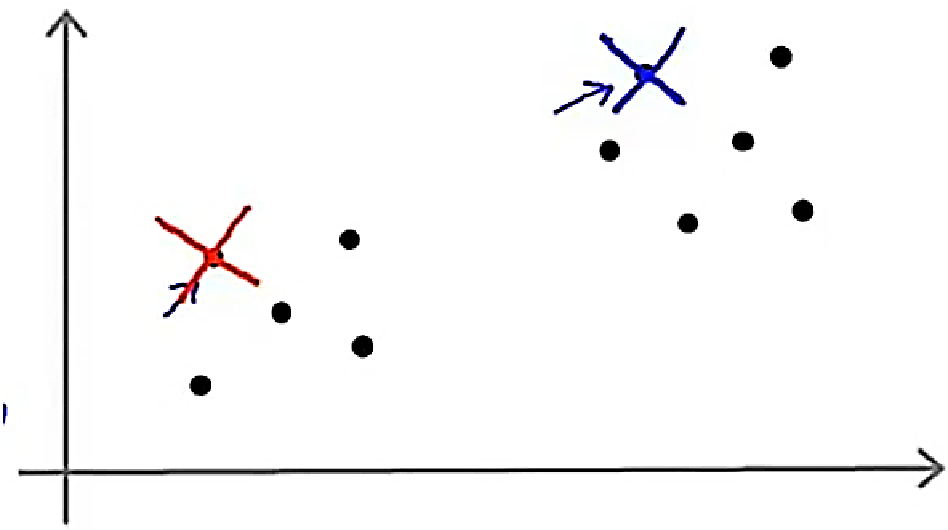
INITIALIZING THE CLUSTER CENTROIDS:

Random initialization

$K=2$

Should have $K < m$

Randomly pick K training examples.

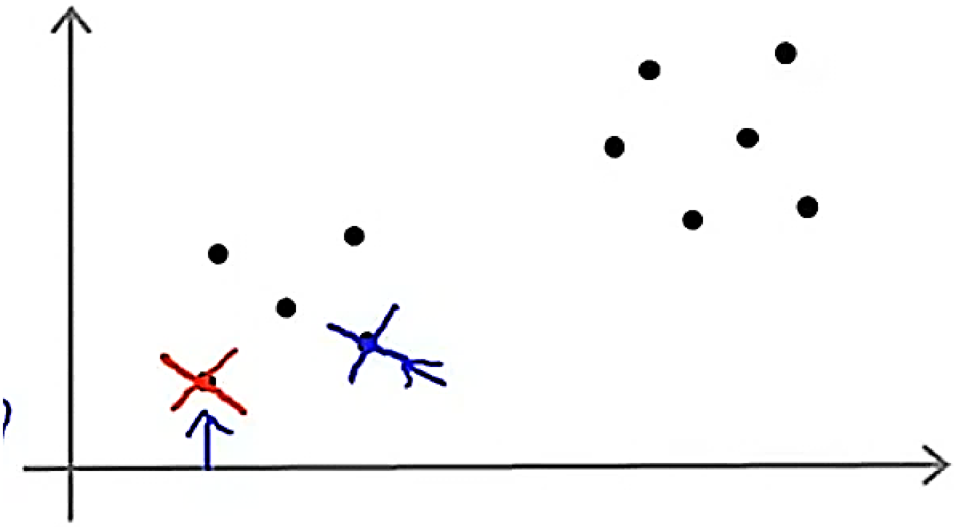


Set $\mu_1, \ldots, \mu_K$ equal to these K examples.

$\mu_1 = x^{(i)}$
$\mu_2 = x^{(j)}$
$\vdots$

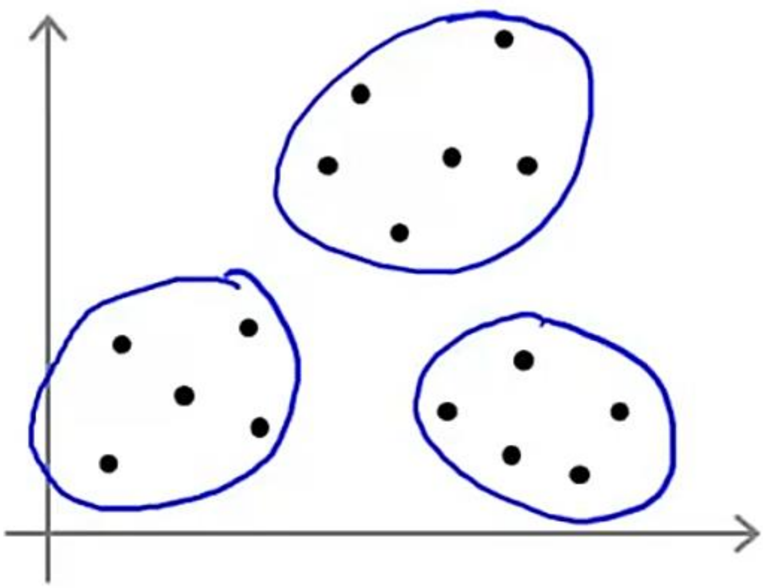
➢ Different values of centroids may lead to different results in the solution

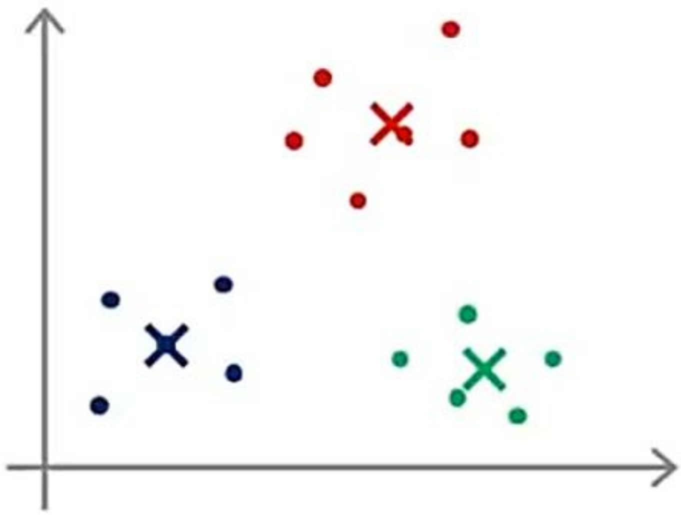
➢ Sometimes, we may end up picking **close examples**, which will make the algorithm **converge to local optimum** instead of global optimum:



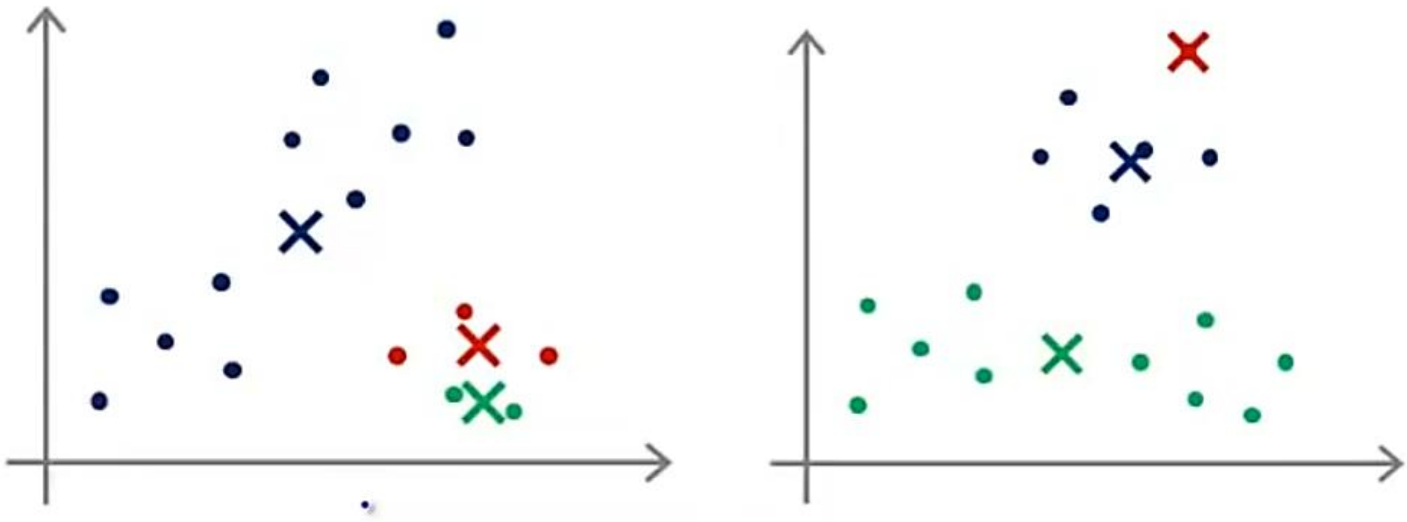
Example:

→ With a good choice of initial centroids, we get global optimum:





→ With other bad choices: we get local optimums like:



Solution for this: **initialize** K-clusters **many times** and choose the one which gives global optimum

For i = 1 to 100 { 50 - 1000

→ Randomly initialize K-means.

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

Compute cost function (distortion)

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

}

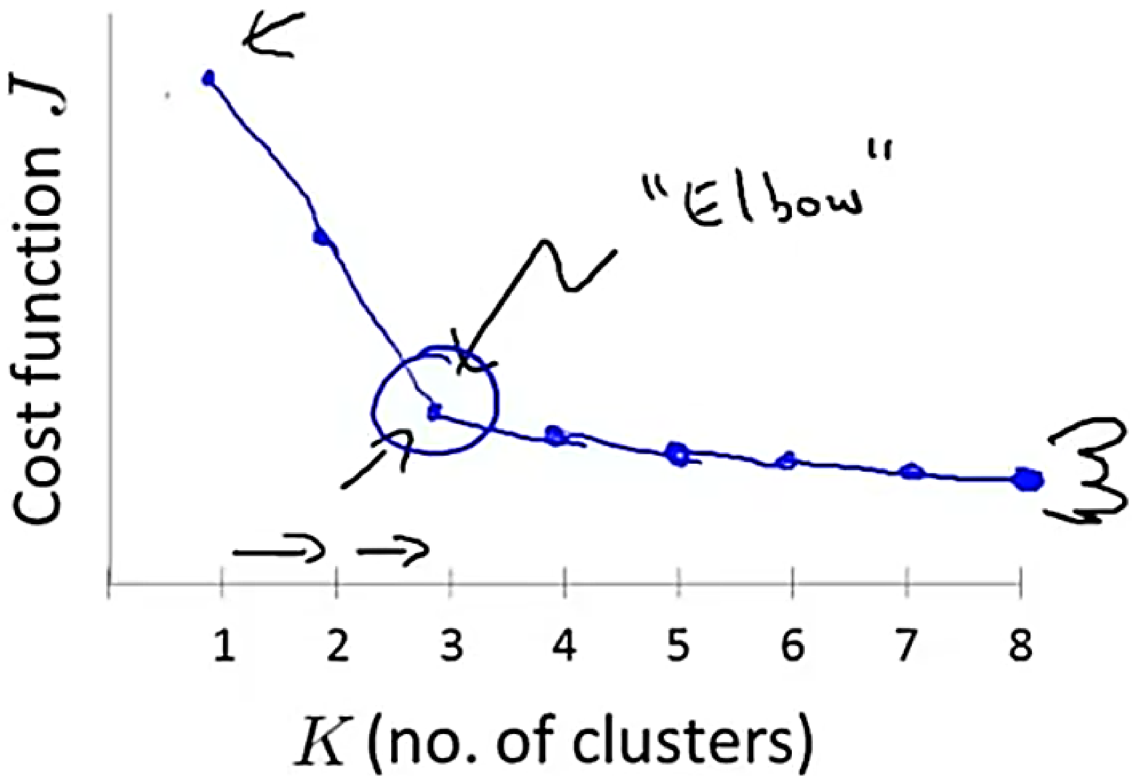
→ After this we have 100 different centroids and **their costs:**

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

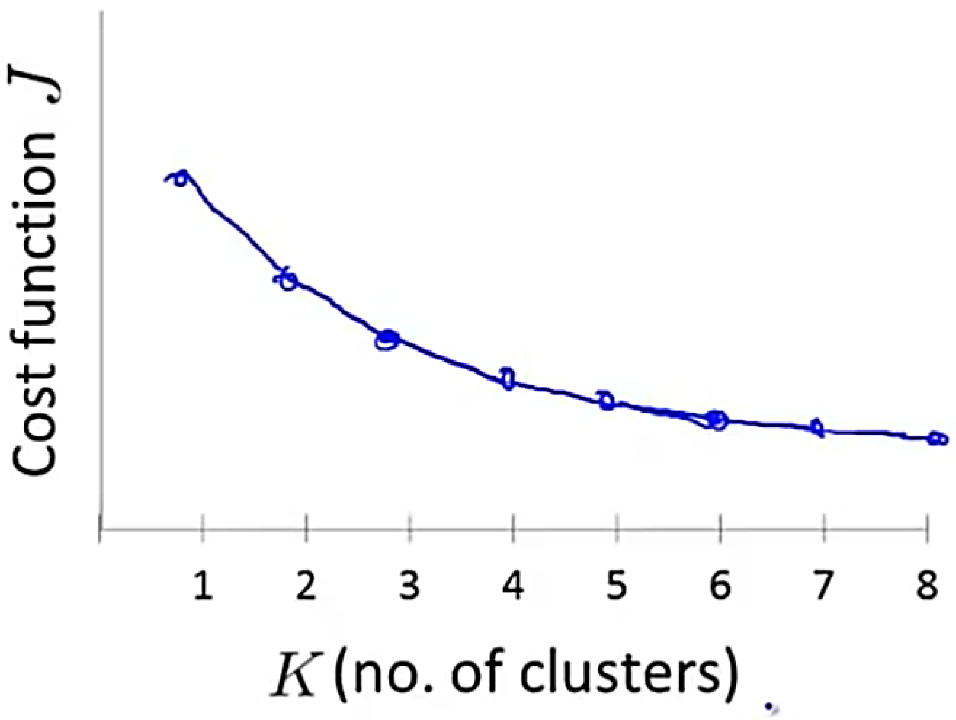
CHOOSING THE NO OF CLUSTERS:

Best way to choose the value of **"K"** is to **look at the visualizations** of data and **choose manually**.

Elbow method:



Elbow method is **not commonly used** because:



Sometimes, there is no clear elbow!

The more usual and reliable way: choose based on the problem

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.