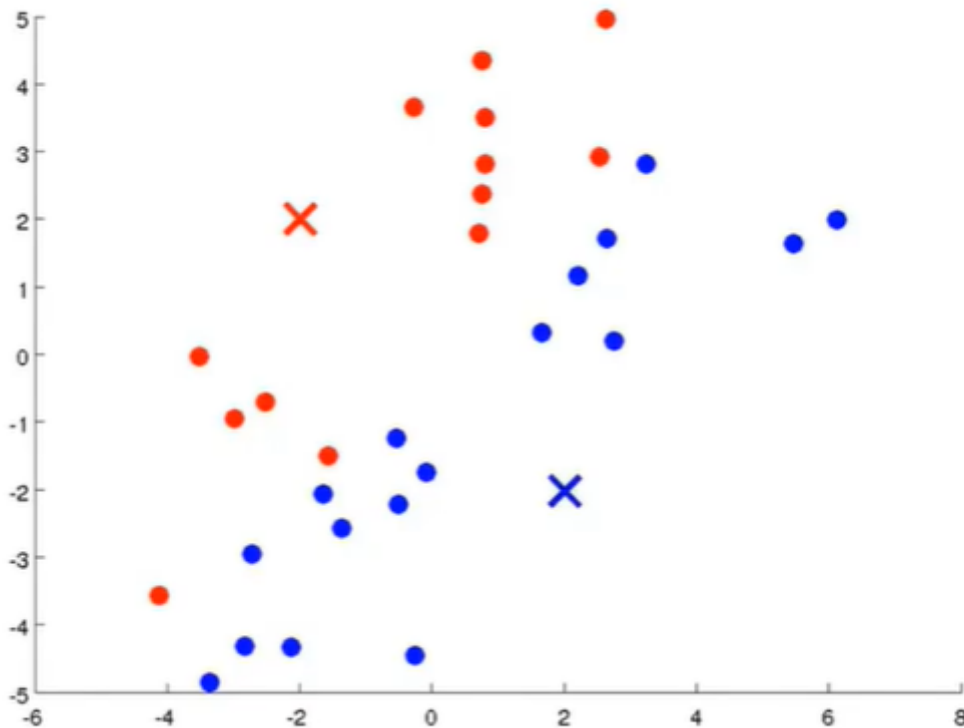


# K-means聚类算法

## 簇分配步骤：

- 1.选取好分类点，离某一个分类点更近的点都归类于那一类。
- 2.移动聚类中心：计算所有某一类的点的平均位置，再将那一类的分类点移动到平均位置。
- 3.再次将离新的分类点更近的点分配到那一类。
- 4.再次移动聚类中心
- ....直到聚类中心不发生改变。



# K-means algorithm

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接受两个输入，k是想要的簇个数，还有一个数据集

规定 $x(i)$ 是一个n维实数向量，所以训练样本是n维向量

## K-means algorithm

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  $x^{(i)}$

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

*Move centroid*

for  $k = 1$  to  $K$

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

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

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

Andrew Ng

第一步随机初始化k个聚类中心

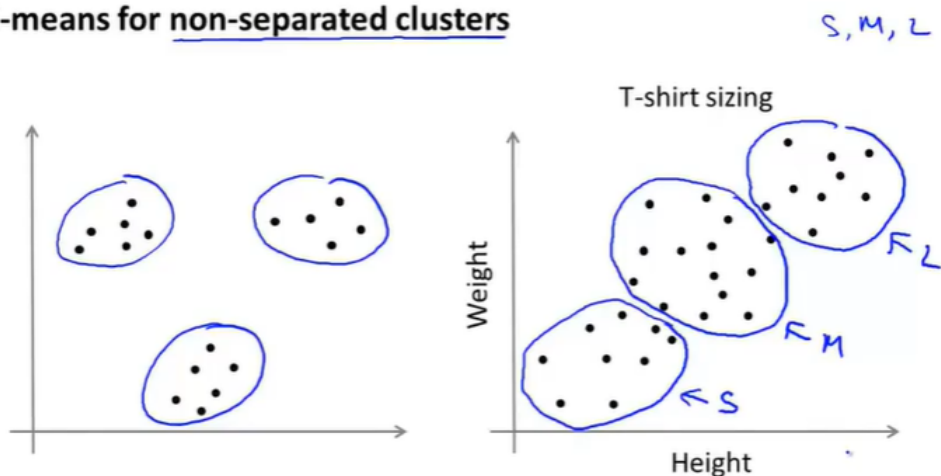
内循环中第一步进行簇分配，将每个样本根据它离聚类中心的距离，将它染成对应的颜色。 $c(i)$ 是1-k之间的数，表示它是属于哪个簇。

另一步是移动聚类中心，将聚类中心分配到该簇中所有点的均值处。

如果出现了一个没有点的聚类中心，这时可以直接移除这个聚类中心，但不会有 $k$ 个簇，而是有 $k-1$ 个簇。如果我们确实需要 $k$ 个簇，那么我们可以重新初始化这个没有点的聚类中心。

**K均值用于市场划分：**

**K-means for non-separated clusters**



## K-means优化目标函数

K-means中，大写的 $K$ 表示簇的数量，小写的 $k$ 表示聚类中心的下标(取值范围1- $K$ )。

$c(i)$ 表示 $x_i$ 所属的聚类中心的索引值

$\mu_k$ 表示第 $k$ 个聚类中心的位置

$\mu_{c(i)}$ 表示 $x_i$ 所属的聚类中心的位置

代价函数是每个数据样本到其聚类中心的距离的平方的加和。代价函数也被叫做算法的失真(distortion)函数。

## K-means optimization objective

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

→  $\mu_k$  = cluster centroid  $k$  ( $\mu_k \in \mathbb{R}^n$ )

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

$x^{(i)} \rightarrow 5$

$c^{(i)} = 5$

$\mu_{c^{(i)}} = \mu_5$

$K$

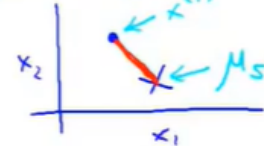
$k \in \{1, 2, \dots, K\}$

Optimization objective:

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

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

Distortion



Andrew Ng

第一步，簇分配步骤就是在保持聚类中心不变，选择 $c_1 \dots c_m$ 来最小化代价函数，把每个点分配给离它最近的距离中心，这样就可以减少它到聚类中心的距离。

第二步，移动聚类中心，就是选择 $\mu$ 值来最小化代价函数。

然后保持迭代。

## K-means随机初始化（如何避开局部最优）

如何初始化 $K$ 个随机聚类中心？

通常选取的方法：

随机将 $k$ 个训练样本当作聚类中心。

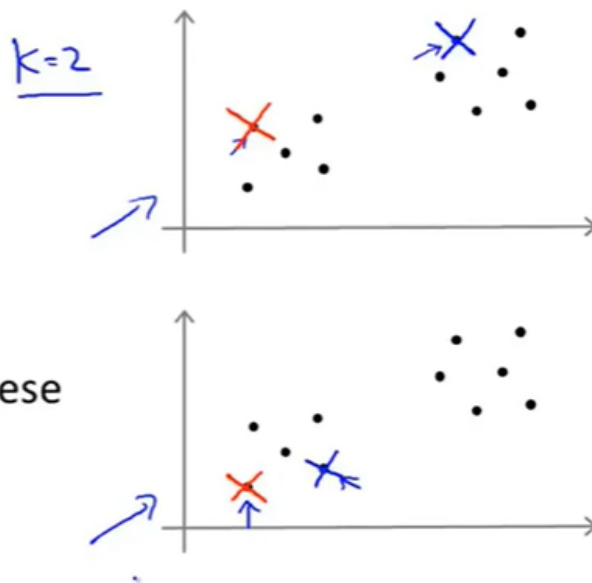
## Random initialization

Should have  $K < m$

Randomly pick  $K$  training examples.

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

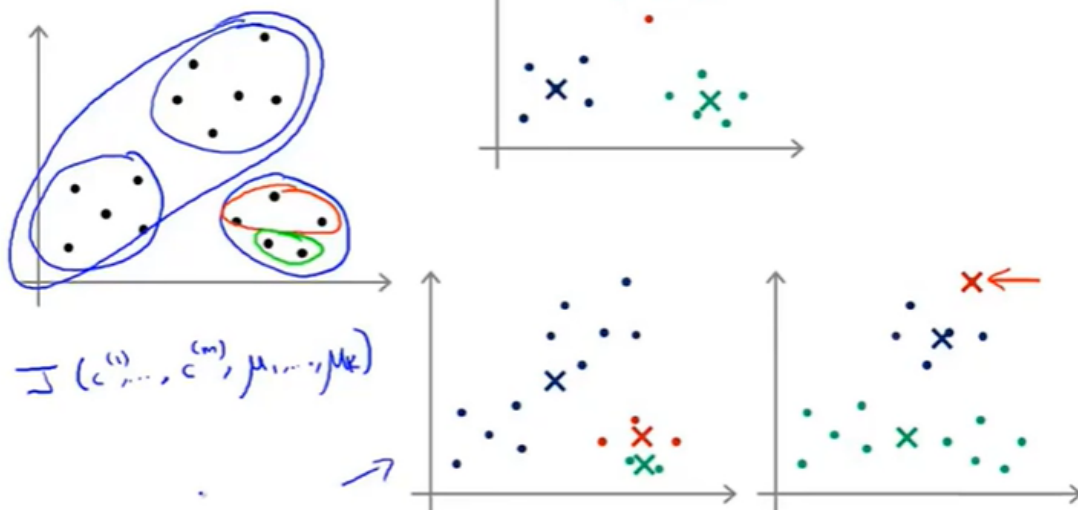
$$\begin{aligned}\mu_1 &= x^{(i)} \\ \mu_2 &= x^{(j)} \\ &\vdots\end{aligned}$$



Andrew Ng

K-means可能会出现效果很不好的局部最优解，也就是J失真函数的局部最优解。

## Local optima



为了避免这种情况，我们可以多次随机初始化，并多次运行kmeans算法，以此来保证获得一个较好的结果。

## 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  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$   
 $K = 2 - 10$

在聚类数K较小(2-10)时，多次随机初始化可以有很好的效果，但当K较大，多次随机初始化帮助很有限。

## K-means 选择聚类数量K

最常见的方法：根据图像手动选择

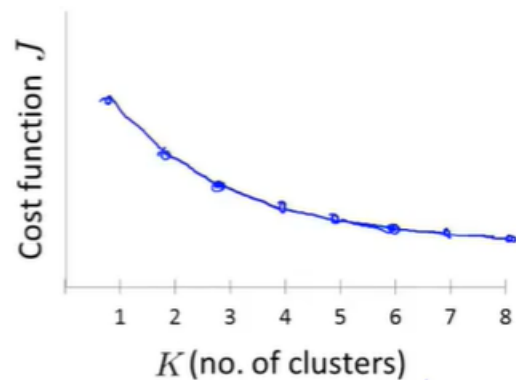
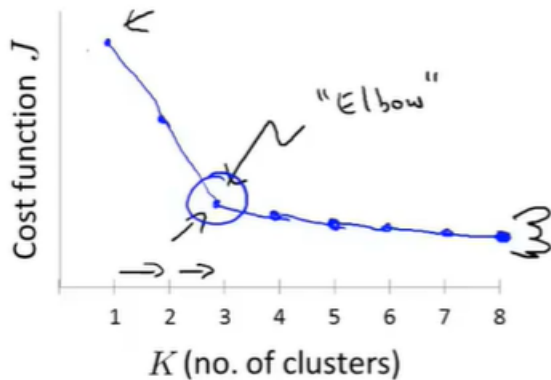
### 肘部法则：

“肘部”：指畸变值（代价函数的值）开始下降的缓慢的点。K选择肘部的点。  
如图一，K=3拐点。

但有些时候图像曲线均匀下降，很难找到清晰的拐点，如图二。

## Choosing the value of K

Elbow method:



另一种思路：看哪个聚类数量能更好地应用于后续目的。比如体恤尺寸的例子。

## 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.

