K-means clustering

K-means tries to improve the inter group similarity while keeping the groups as far as possible from each other. Basically K-means runs on distance calculations, which again uses "Euclidean distance" for this purpose.

Euclidean distance calculates the distance between two given points using the following formula:

Euclidean distance = $\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$

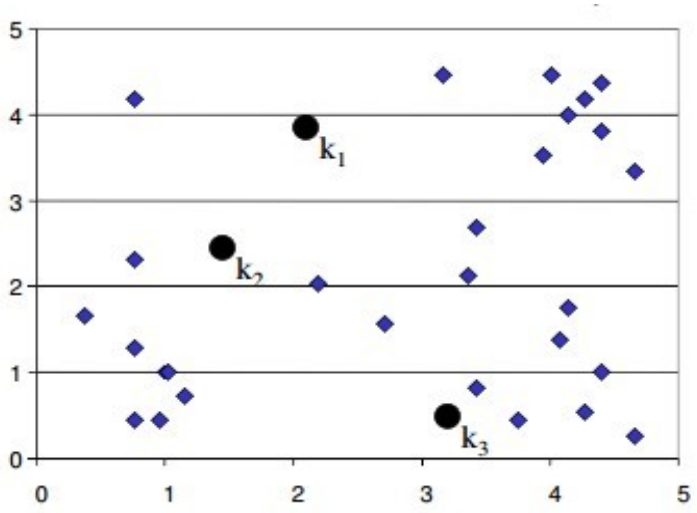
above formula captures the distance in 2-Dimensional space and in multi-dimensional space as well.

K in K-means represents the number of clusters in which we want our data to divide into.

Algorithm

K-means is an iterative process of clustering: which keeps iterating until it reaches the best solution or clusters in our problem space.

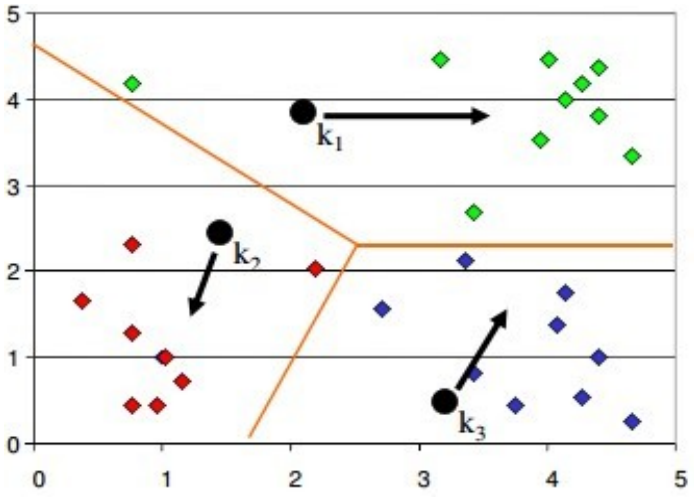
1 start with number of cluster we want e.g., 3 in this case.
K-means algorithm start the process with random centers in data, and then tries to attach the nearest points to these centers.



each data point is assigned to its nearest centroid, based on the Euclidean distance.

$$\underset{c_i \in C}{\arg\min}\ dist(c_i, x)^2$$

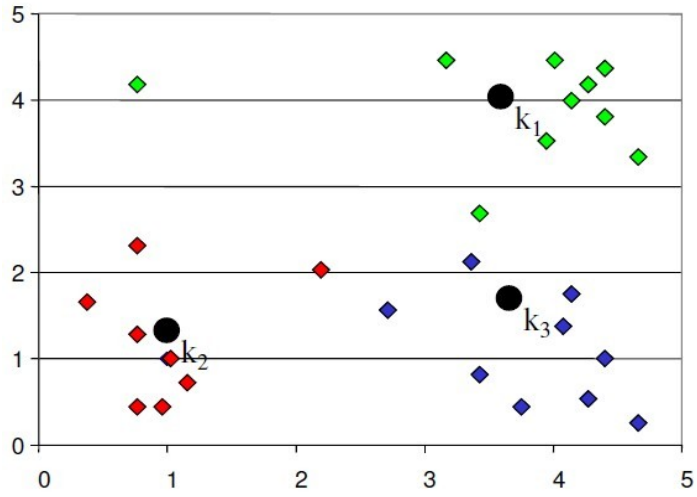
2 Algorithm then moves the randomly allocated centers to the means of created groups



Centroid update step:
In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

3 In the next step, data points are again reassigned to these newly created centers



4 Steps 2 & 3 are repeated until no member changes their association/groups

K-means: step-by-step example

as a simple illustration of a k-means algorithm, consider the following dataset consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This dataset is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	individual	Mean vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	**Individual**	**Vector (centroid)**	**Individual**	**Vector (centroid)**
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Cluster 1		**Cluster 2**	
x1	1.0	**x1**	5.0
Y1	1.0	**Y1**	7.0

subject2				
x2	1.5			
y2	2.0			
D_{c1}	**1.12**		D_{c2}	**5.22**
arg min {D_{c1}, D_{c2}		1.12		
Updated weight (online)		**Cluster 1**		
vector (centroid)				
Cluster 1	1.0	1.0		
subject2	1.5	2.0		
new centroid	**1.25**	**1.5**	(Mean)	

subject3				
x2	3.0			
y2	4.0			
D_{c1}	**3.05**		D_{c2}	**3.61**
arg min {D_{c1}, D_{c2}		3.05		
Updated weight (online)				
vector (centroid)		**Cluster 1**		
subject1	1.0	1.0		
subject2	1.5	2.0		
subject3	3.0	4.0		
new centroid	**1.83**	**2.33**	(Mean)	

subject5				
x2	3.5			
y2	5.0			
D_{c1}	**3.14**		D_{c2}	**2.50**
arg min {D_{c1}, D_{c2}		2.50		
Updated weight (online)				
vector (centroid)		**Cluster 2**		
subject4	5.0	7.0		
subject5	3.5	5.0		
new centroid	**4.25**	**6.00**	(Mean)	

subject6			
x2	4.5		
y2	5.0		
D_{c1}	**3.77**	D_{c2}	**1.03**
arg min {D_{c1}, D_{c2}		1.03	
Updated weight (online)			
vector (centroid)		**Cluster 2**	
subject4	5.0	7.0	
subject5	3.5	5.0	
subject6	4.5	5.0	
new centroid	**4.33**	**5.67**	(Mean)

subject7			
x2	3.5		
y2	4.5		
D_{c1}	**2.73**	D_{c2}	**1.43**
arg min {D_{c1}, D_{c2}		1.43	
Updated weight (online)			
vector (centroid)		**Cluster 2**	
subject4	5.0	7.0	
subject5	3.5	5.0	
subject6	4.5	5.0	
subject7	3.5	4.5	
new centroid	**4.13**	**5.38**	(Mean)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

new centroid

Cluster 1		**Cluster 2**	
1.8	**2.3**	**4.1**	**5.4**

But we cannot yet be sure that each individual has been assigned to the right cluster. So, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	**ean (centroid**	**ean (centroid) of Cluster 2**	
1	1.4	1.4	
2	2.5	2.5	
3	**5.0**	**5.0**	near to cluster 2
4	8.6	8.6	
5	6.1	6.1	
6	6.7	6.7	
7	5.7	5.7	

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller than the distance to the other cluster's mean (which is not the case with individual 3). This, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	**n Vector (centroid)**
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from this new partition until no more relocations occur. However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

Also, it is possible that the k-means won't find a final solution. In this case it would be a good idea to consider stopping the algorithm after a pre-chosen maximum of iterations.

http://blog.galvanize.com/introduction-k-means-cluster-analysis/
https://www.dezyre.com/data-science-in-r-programming-tutorial/k-means-clustering-techniques-tutorial
https://algobeans.com/2015/11/30/k-means-clustering-laymans-tutorial/
http://mnemstudio.org/clustering-k-means-example-1.htm