Everything you need to know about K-Means Clustering

Tanvi Penumudy · Follow
Published in Analytics Vidhya
9 min read · Jan 16, 2021

You're at the right place if you're wondering what K-means Clustering is all about! Let's quickly get started without further due!

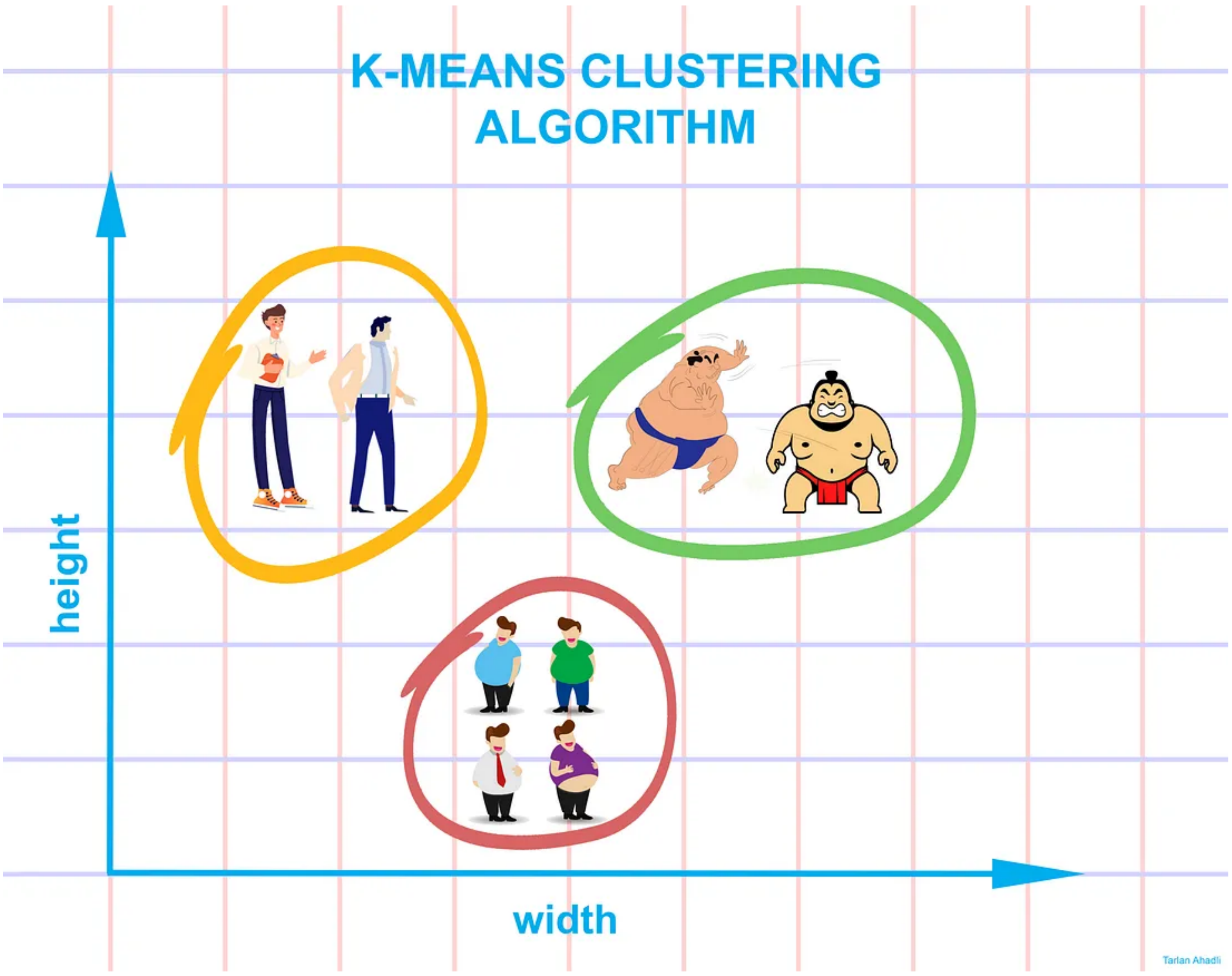


Image Source: Google Images

Introduction

Types of Clustering

Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.

*For more on Supervised/Unsupervised Learning — **A Beginner’s Guide for Getting Started with Machine Learning***

There are primarily two categories of clustering:

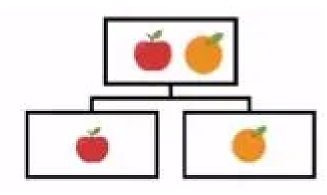
- *Hierarchical clustering*
- *Partitioning clustering*

Hierarchical Clustering

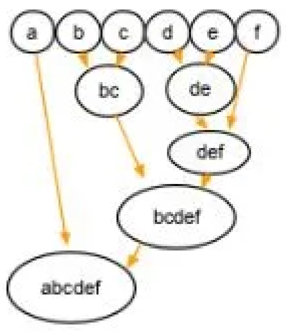
Hierarchical clustering is further subdivided into:

- *Agglomerative clustering*
- *Divisive clustering*

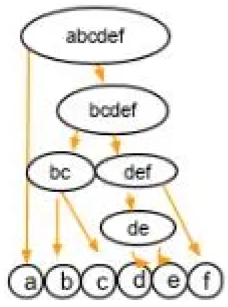
Hierarchical clustering uses a tree-like structure, like so:



In ***agglomerative clustering***, there is a ***bottom-up approach***. We begin with each element as a separate cluster and merge them into successively more massive clusters, as shown below:



Divisive clustering is a ***top-down approach.*** We begin with the whole set and proceed to divide it into successively smaller clusters, as you can see below:



Partitioning Clustering

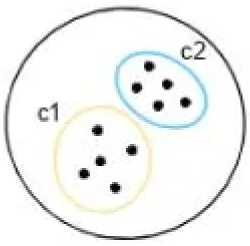
Partitioning clustering is further subdivided into:

- *K-Means clustering*
- *Fuzzy C-Means clustering*

Partitioning clustering is split into two subtypes — *K-Means clustering* and *Fuzzy C-Means.*

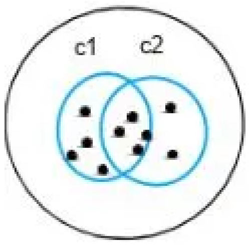
In K-means clustering, the objects are divided into several clusters mentioned by the number 'K.'

> *So if we say K = 2, the objects are divided into two clusters, c1 and c2, as shown:*



> *Here, the features or characteristics are compared, and all objects having similar characteristics are clustered together.*

Fuzzy c-means is very similar to k-means in the sense that it clusters objects that have similar characteristics together. In k-means clustering, a single object cannot belong to two different clusters. But in c-means, objects can *belong to more than one cluster,* as shown.



K-means Clustering

K-means is similar to ***KNN*** because it looks at distance to predict class membership. However, unlike KNN, K-means is an ***unsupervised learning algorithm***. Its goal is to discover how different points cluster together. The intuition behind this mathematical model is that similar data points will be closer together.

K-means then tries to determine different k-points called ***centroids***, which are at the centre *(least cumulative distance)* from other points of the same class, but further away from points of another class.

> *This algorithm is intuitive but computationally taxing, so it's mostly used for exploratory analysis on smaller data sets.*

For more on KNN — ***A Beginner's Guide to KNN and MNIST Handwritten Digits Recognition using KNN from Scratch***

> *For a better understanding of K-means, let's take an example from cricket. Imagine you received data on a lot of cricket players from all over the world, which gives information on the runs scored by the player and the wickets taken by them in the last ten matches.*

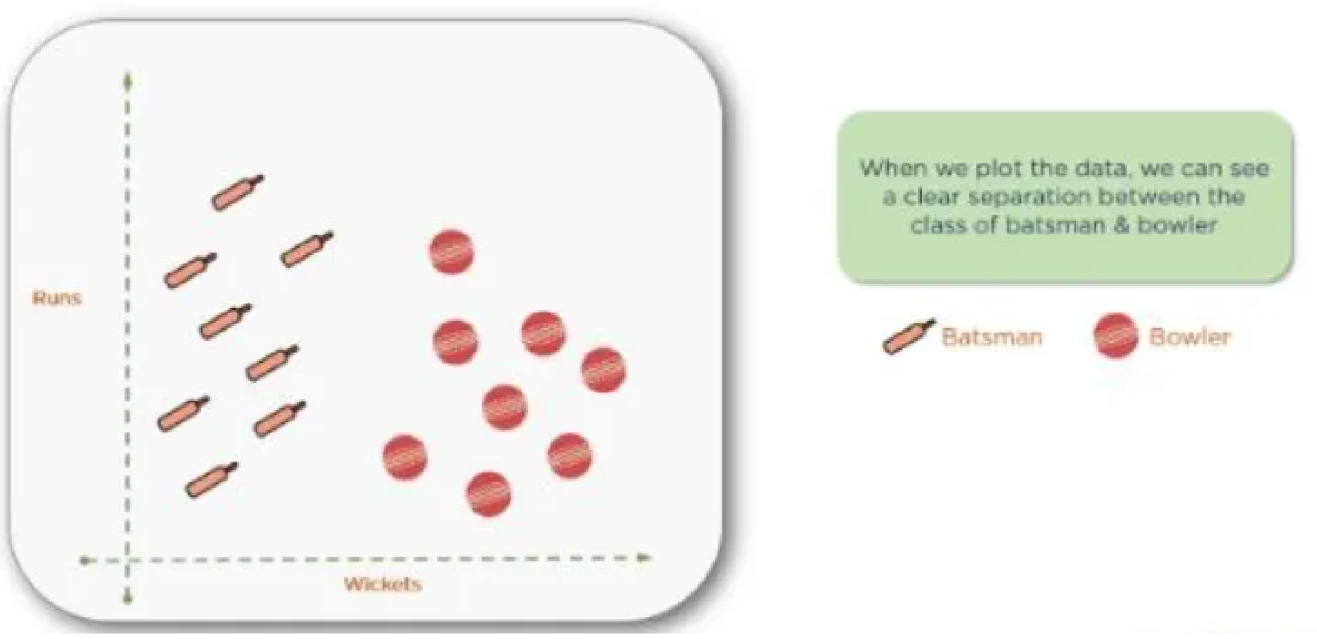
Based on this information, we need to group the data into two clusters, namely batsman and bowlers. *Let's take a look at the steps to create these clusters:*

Solution

Assign data points

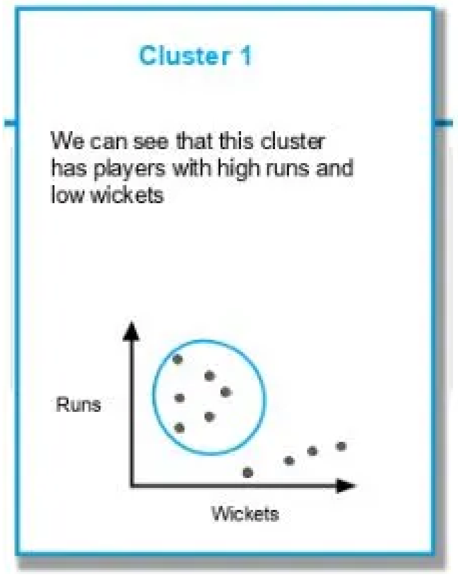
Here, we have our data set plotted on 'x' and 'y' coordinates. The information on the y-axis is about the runs scored, and on the x-axis about the wickets taken by the players.

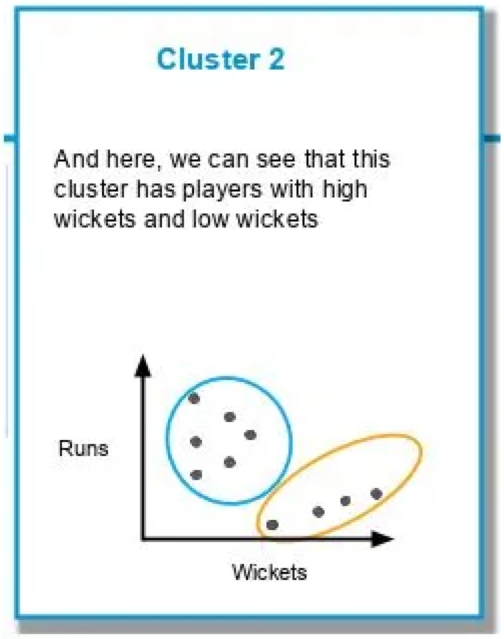
If we plot the data, this is how it would look:



Perform Clustering

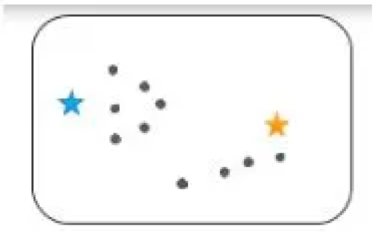
We need to create the clusters, as shown below:


Cluster 1
We can see that this cluster has players with high runs and low wickets
Runs
Wickets


Cluster 2
And here, we can see that this cluster has players with high wickets and low wickets
Runs
Wickets

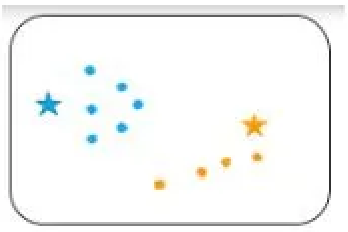
Considering the same data set, let us solve the problem using K-Means clustering (taking K = 2). The first step in k-means clustering is the allocation of two centroids randomly (as K=2). Two points are assigned as centroids.

> *They are called centroids, but initially, they are not the central point of a given data set.*

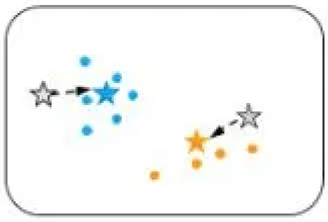


The next step is to determine the distance between each of the data points from the randomly assigned centroids. For every point, the distance is measured from both the centroids and whichever distance is less, that point is assigned to that centroid.

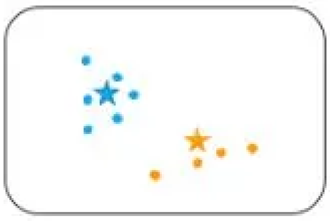
> *You can see the data points attached to the centroids and represented here in blue and yellow.*



The next step is to determine the actual centroid for these two clusters. The original randomly allocated centroid is to be repositioned to the actual centroid of the clusters.



This process of calculating the distance and repositioning the centroid continues until we obtain our final cluster. *Then the centroid repositioning stops.*



> *As seen above, the centroid doesn't need anymore repositioning, and it means the algorithm has converged, and we have the two clusters with a centroid.*

Applications

K-Means clustering is used in a variety of examples or business cases in real-life, like:

- *Academic performance*
- *Diagnostic systems*
- *Search engines*
- *Wireless sensor networks*

Measuring Distance

Distance measure determines the similarity between two elements and influences the shape of clusters.

K-Means clustering supports various kinds of distance measures, such as:

- ***Euclidean distance measure***
- ***Manhattan distance measure***

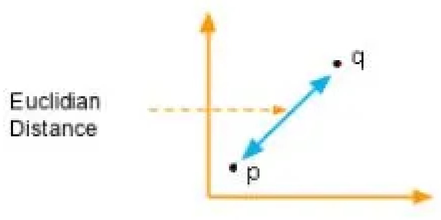
- *A squared euclidean distance measure*
- *Cosine distance measure*

Euclidean Distance Measure

The most common case is determining the distance between two points. If we have a point P and point Q, the euclidean distance is an ordinary straight line. ***It is the distance between the two points in Euclidean space.***

The formula for distance between two points is shown below:

$$d=\sqrt{\textstyle\sum_{i=1}^{n}(q_i-p_i)^2}$$



Squared Euclidean Distance Measure

This is identical to the Euclidean distance measurement but does not take the square root at the end. *The formula is shown below:*

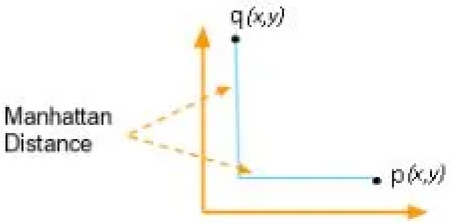
$$d=\textstyle\sum_{i=1}^{n}(q_i-p_i)^2$$

Manhattan Distance Measure

The Manhattan distance is the simple sum of the horizontal and vertical components or the distance between two points measured along axes at right angles.

The formula is shown below:

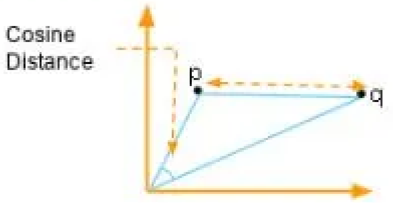
$$d = \sum_{i=1}^{n} | q_x - p_x | + | q_x - p_y |$$



Cosine Distance Measure

In this case, we take the angle between the two vectors formed by joining the points from the origin. The formula is shown below:

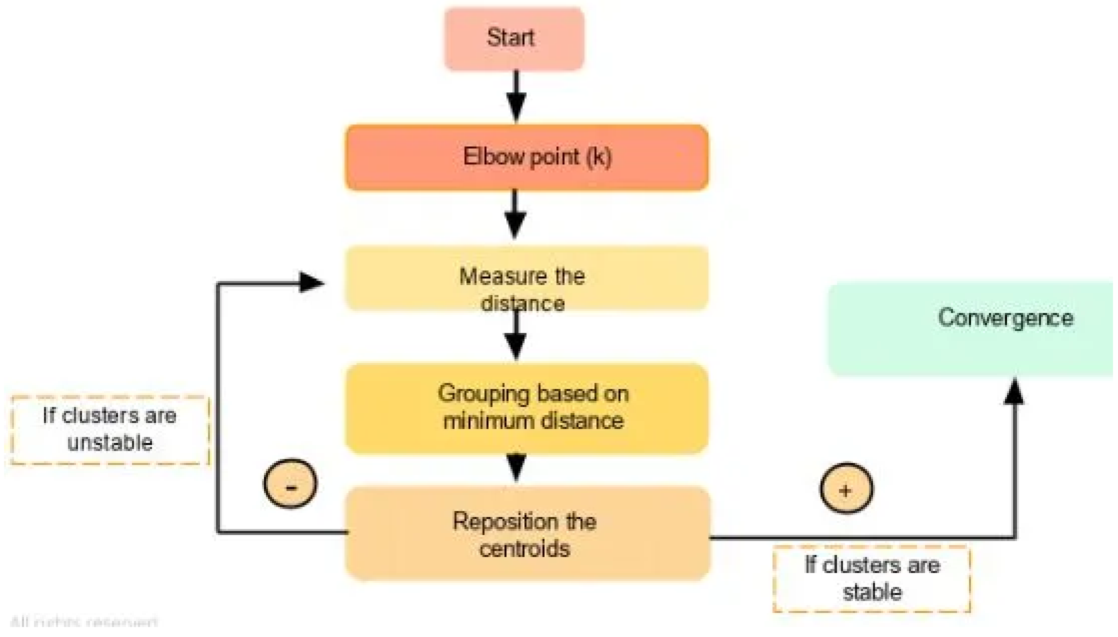
$$d = \frac{\sum_{i=0}^{n-1} q_i - p_x}{\sum_{i=0}^{n-1} (q_i)^2 \times \sum_{i=0}^{n-1} (p_i)^2}$$



For more on *distance measures*, refer — [*A Beginner's Guide to KNN and MNIST Handwritten Digits Recognition using KNN from Scratch*]

Working

The flowchart below summarises how ***K-means clustering*** *works:*



We can use the *trial and error method* by specifying the value of ***K (=3,4,5..)***. As we progress, we keep changing the value until we get the best clusters.

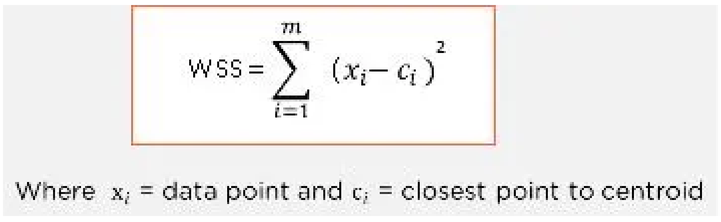
Another method is to use the ***Elbow technique*** to determine the value of K.

> *Once we get the value of K, the system will assign that many centroids randomly and measure the distance of each of the data points from these centroids.*

Accordingly, it assigns those points to the corresponding centroid from which the distance is minimum. So each data point will be assigned to the centroid, which is closest to it. Thereby we have a K number of initial clusters.

The ***Elbow method*** is the best way to find the number of clusters. The elbow method constitutes running K-Means clustering on the dataset.

Next, we use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. ***Within the sum of squares (WSS)*** is defined as *the sum of the squared distance between each member of the cluster and its centroid.*

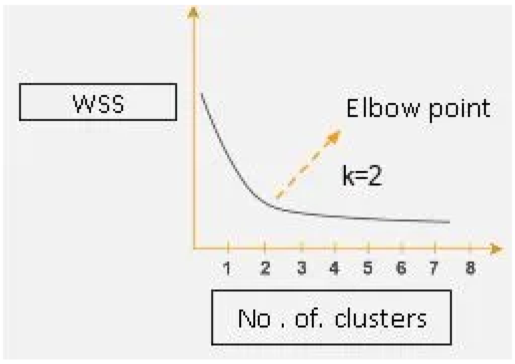


$$\boxed{WSS = \sum_{i=1}^{m} (x_i - c_i)^2}$$

Where x_i = data point and c_i = closest point to centroid

The WSS is measured for each value of K. *The value of K, which has the least amount of WSS, is taken as the* ***optimum value.***

Now, we draw a curve between WSS and the number of clusters.



> *Here, WSS is on the y-axis and number of clusters on the x-axis.*

You can see that there is a very ***gradual change*** in the value of WSS as the K value increases from 2.

So, you can take the elbow point value as the optimal value of K. It should be either two, three, or at most four. But, beyond that, increasing the number of clusters *does not dramatically change* the value in WSS, it gets ***stabilized.***

K-Means Clustering Algorithm

Let's say we have x1, x2, x3……… x(n) as our inputs, and we want to split this into K clusters.

The steps to form clusters are:

Step 1: Choose K random points as cluster centres called ***centroids.***

Step 2: Assign each x(i) to the closest cluster by implementing euclidean distance (i.e., calculating its distance to each centroid)

Step 3: Identify new centroids by taking the average of the assigned points.

Step 4: Keep repeating step 2 and step 3 until convergence is achieved!

Let's take a detailed look at it at each of these steps:

Step 1

We randomly pick K (centroids). We name them ***c1,c2,….. ck****, and we can say that:*

$$\boxed{C = c_1, c_2, \ldots, c_k}$$

Where C is the set of all centroids.

Step 2

We assign each data point to its nearest centre, which is accomplished by calculating the euclidean distance.

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

Where dist() is the ***Euclidean distance.***

> *Here, we calculate the distance of each x value from each c value, i.e. the distance between x1-c1, x1-c2, x1-c3, and so on. Then we find which is the lowest value and assign x1 to that particular centroid.*
>
> *Similarly, we find the minimum distance for x2, x3, etc.*

Step 3

We identify the actual centroid by taking the average of all the points assigned to that cluster.

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

Where Si is the set of all points assigned to the ith cluster.

> *It means the original point, which we thought was the centroid, will shift to the new position, which is the actual centroid for each of these groups.*

Step 4

Keep repeating step 2 and step 3 until convergence is achieved.

> *Once the cluster becomes* ***static****, the k-means algorithm is said to be* ***converged.***

Additional Resources and References

For more implementations on K-means Clustering: