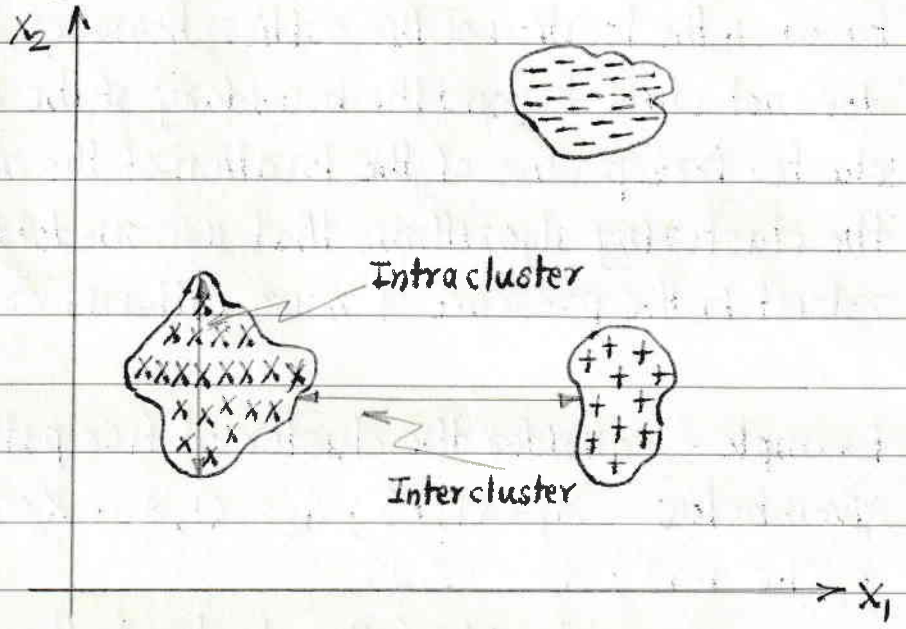
Clustering

Clustering is a process of grouping a set of patterns. It generates a partition consisting of cohesive groups or clusters from a given collection of patterns. Representations or descriptions of the clusters formed are used in decision making – classification is one of the important decision-making paradigms used.

The process of clustering is carried out so that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in a corresponding sense.



Consider the two-dimensional data set. Here each pattern is represented as a point in the two-dimensional space. It can be seen that there are three clusters. Also, the Euclidean distance between any two points belonging to the same cluster is smaller than that between any two points belonging to different clusters. Thus, this characterizes the fact that intracluster distance is small and intercluster distance is large.

Clustering is useful for generating data abstraction. A cluster of points is represented by it's centroid or it's medoid. The centroid stands for the sample mean of the points in cluster C. It Is given by $\frac{1}{N_C} \sum X_i \in C$, where N_C is the number of patterns in cluster C, and it need not coincide with one of the points in the cluster. The medoid is the most centrally located point in the cluster, i.e. medoid is that point in the cluster from which the sum of the distances from the points in the cluster is the minimum.

The centroid of the data can shift without any bound based on the location of the outlier, however, the medoid does not shift beyond the boundary of the original cluster irrespective of the location of the outlier. Hence, the clustering algorithms that use medoids are more robust in the presence of noisy patterns or outliers.

Example - Consider the cluster of five patterns as given below - $X_1 = (1,1)$; $X_2 = (1,2)$; $X_3 = (2,1)$; $X_4 = (1.6, 1.4)$; $X_5 = (2,2)$.

The centroid of the cluster is the sample mean. $\mu = \frac{1}{5}[(1,1)+(1,2)+(2,1)+(1.6,1.4)+(2,2)]$

$\therefore \mu = (1.52, 1.48)$. Note that μ is not one of the X_is. The medoid of the cluster is $m = (1.6, 1.4)$. It is a point such that sum of the distances from the points in the cluster is minimum. For the medoid identified, $d(X_1, m) = 0.52$, $d(X_2, m) = 0.72$, $d(X_3, m) = 0.32$, $d(X_4, m) = 0.0$, $d(X_5, m) = 0.52$ and sum $= 2.08$

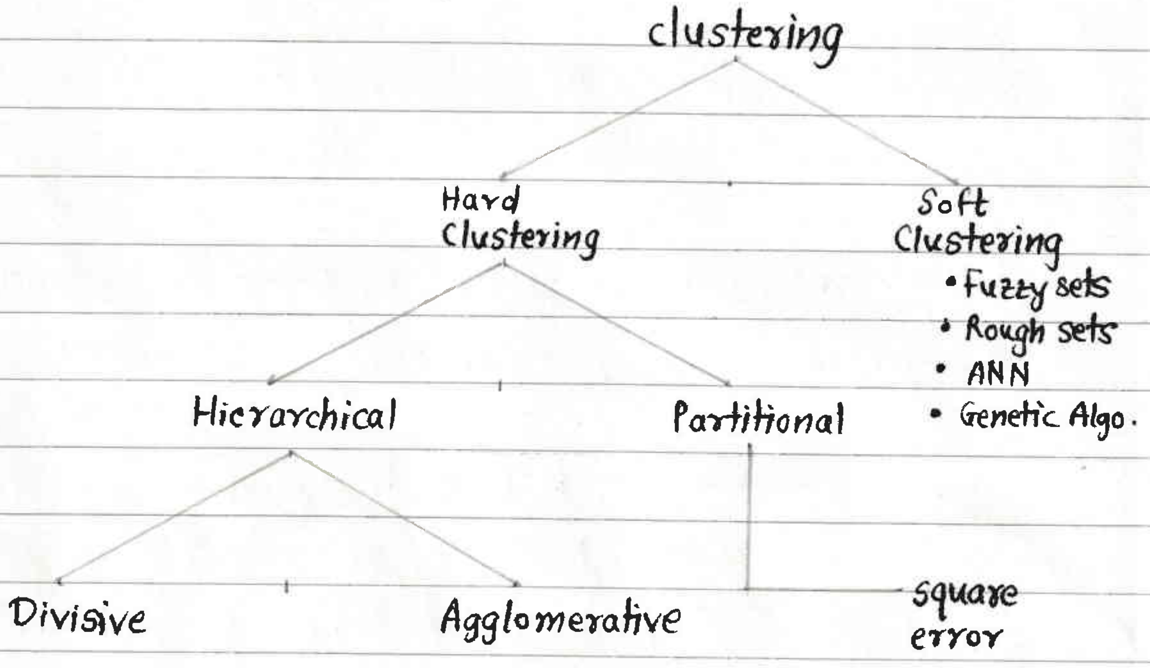
It is possible to show that the sum of the distances of of points in the cluster from x_1, x_2, x_3 or x_5 will be larger than 2·08, ruling out the possibility of any point other than x_4 being medoid.

Now, consider addition of a point (1·6, 7·4) to the cluster. The centroid of the updated cluster is $\mu_1 = (1\cdot53, 2\cdot47)$ The medoid shifts from $m = (1\cdot6, 1\cdot4)$ to $m_1 = (2\cdot2)$.

Now, consider the addition of a point (1·6, 17·4) to the original cluster instead of (1·6, 7·4). The centroid of the resulting cluster of 6 points is $\mu_2 = (1\cdot53, 4\cdot13)$. The medoid m_2 of the cluster remains unchanged.

Thus, the centroid keeps shifting away based on the location of the outlier pattern which is added to the cluster, however the medoid of the cluster remains much robust.

Taxonomy of clustering approaches



The distinction between hard and soft clustering is based on whether the partitions generated are overlapping or not.

Hierarchical Algorithms -

Hierarchical algorithms produce a nested sequence of data partitions. The sequence can be depicted using a tree structure known as dendrogram. The algorithms are either divisive or agglomerative.

The divisive algorithm starts with a single cluster having all the patterns; at each successive step, a cluster is split. This process continues till we end up with one pattern in a cluster or a collection of singleton clusters. It uses a top-down strategy for generating partitions of data.

Agglomerative algorithms on the other hand use a bottom-up strategy. The algorithm starts with 'n' singleton clusters when the input data is of size 'n', where each input pattern is in a different cluster. At successive levels, the most similar pair of clusters is merged to reduce the size of the partition by 1. An important property of the agglomerative algorithms is that once two patterns are placed in the same cluster at a level, they remain in the same cluster at all subsequent levels. Similarly, in the divisive algorithms, once two patterns are placed in two different clusters at a level, they remain in different clusters at all subsequent levels.

Divisive clustering-

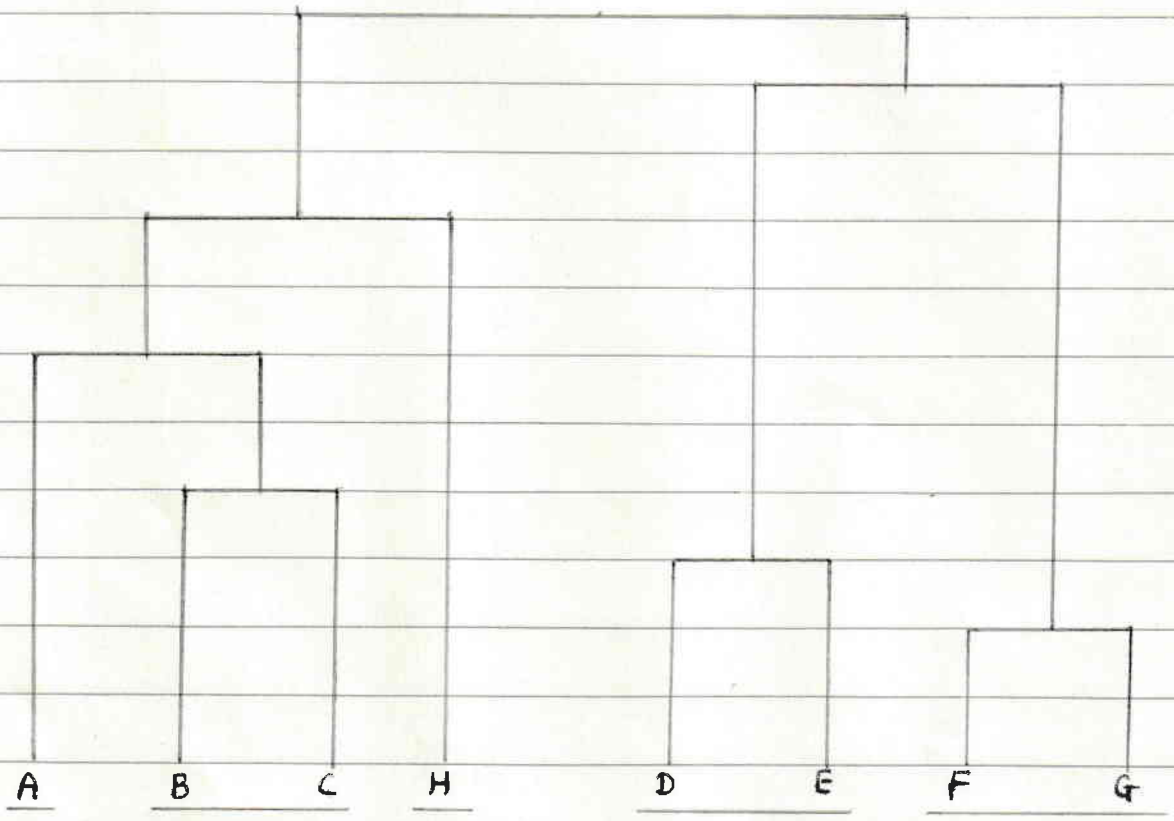
Divisive algorithms are either polythetic, where the division is based on more than one feature, or monothetic where only one feature is considered at a time. A scheme for polythetic clustering involves finding all possible 2-partitions of the data and choosing the best partition.

Here, the partition with the least sum of the sample variances of the two clusters is chosen as the best. From the resulting partition, the cluster with the maximum sample variance is selected and is split into an optimal 2-partition. This process is repeated till singleton clusters are obtained. The sum of the sample variances is calculated as follows-
If the patterns are split into two partitions with 'm' patterns $X_1, X_2, \ldots, X_m$ in one cluster and 'n' patterns $Y_1, Y_2, \ldots Y_n$ in the other cluster with the centroids of the two clusters being C_1 and C_2 respectively, then the sum of the sample variances will be $\sum_i (X_i - C_1)^2 + \sum_j (Y_j - C_2)^2$.

Example- 8 two-dimensional patterns are given as below-
$A = (0\cdot5, 0\cdot5)$; $B = (2, 1\cdot5)$; $C = (2, 0\cdot5)$; $D = (5, 1)$;
$E = (5\cdot75, 1)$; $F = (5, 3)$; $G = (5\cdot5, 3)$; $H = (2, 3)$.

At the top, there is a single cluster consisting of all the 8 patterns. By considering all possible 2-partitions $(2^7 - 1 = 127)$, the best 2-partition is given by $\{\{A, B, C, H\}, \{D, E, F, G\}\}$.
At the next level, the cluster $\{D, E, F, G\}$ is selected to split into two clusters $\{D, E\}$ and $\{F, G\}$
At the next level, we partition the cluster $\{A, B, C, H\}$ into $\{A, B, C\}$ and $\{H\}$ and at the subsequent level, the cluster $\{A, B, C\}$ is split into two clusters $\{A\}$, and $\{B, C\}$.
By the end of this level, we have 5 clusters given by $\{A\}$, $\{B, C\}$, $\{H\}$, $\{D, E\}$ and $\{F, G\}$
Similarly partitions having 6, 7 and 8 clusters of the data at successive levels will be created. Thus at the final level, each cluster has only one point.

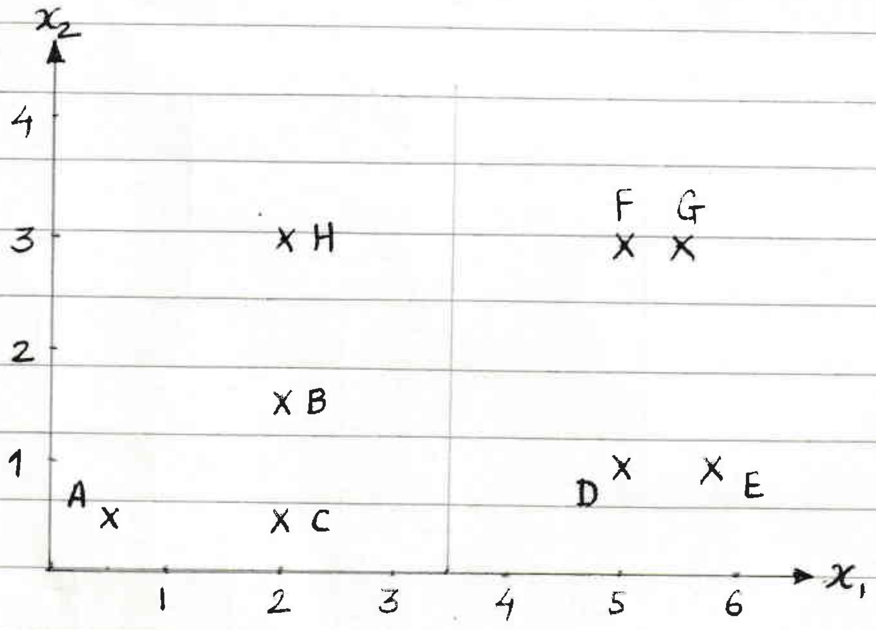
The dendrogram at the level of 5 clusters is as shown



It is possible to use one feature at a time to partition the given data set. In such a case, a feature direction is considered and the data is partitioned into two clusters based on the gap in the projected values along the feature direction. That is, the data set is split into two parts at a point that corresponds to the mean value of the maximum gap found among the values of the data's feature. Each of these clusters is further partitioned sequentially using the remaining features.

Example: Consider the earlier data set, with patterns $A=(0\cdot5,0\cdot5)$; $B=(2,1\cdot5)$; $C=(2,0\cdot5)$; $D=(5,1)$; $E=(5\cdot75,1)$; $F=(5,3)$; $G=(5\cdot5,3)$; $H=(2,3)$.

Monothetic clustering can be used for the partitioning as shown.



Here, we have 8 - two dimensional patterns and x_1 & x_2 are the two features used to describe these patterns. Taking feature x_1, the data is split into two clusters based on the maximum inter-pattern gap which occurred between two successive patterns in the x_1 direction. If we consider the x_1 values of the patterns in increasing order, they are A : 0·5, B : 2, H : 2, C : 2, D : 5, F : 5, G : 5·5 and E : 5·75.

So the maximum inter-pattern gap (5-2 = 3) units occurs between C and D. We select the mid-point between 2 and 5, which is 3·5 and use it to split the data into two clusters. They are $C_1 = \{A, B, C, H\}$ and $C_2 = \{D, E, F, G\}$

Each of these clusters is split further into two clusters based on the value of x_2; the split again depends on the maximum inter-pattern gap between two successive patterns in the x_2 direction. Ordering patterns in C_1 based on their x_2 values, we get A : 0·5, C : 0·5, B : 1·5 and H : 3. Hence the maximum gap of 3 - 1·5 = 1·5 units occurs between B and H. So, by splitting C_1 at the midpoint, 2·25, of the feature dimension x_2, we get 2 clusters $C_{11} = \{H\}$ and $C_{12} = \{A, B, C\}$

Similarly, by splitting C_2 using the value 2 for x_2, $C_{21} = \{F, G\}$ and $C_{22} = \{D, E\}$.

Agglomerative Clustering –

Typically, an agglomerative clustering algorithm goes through the following steps –

1) Compute the similarity / dissimilarity matrix between all pairs of patterns. Initialise each cluster with a distinct pattern.
2) Find the closest pair of clusters and merge them. Update the proximity matrix to reflect the merge.
3) Stop the process whenever desired number of clusters are obtained, or all the patterns are in one cluster, else repeat the earlier step.

Example – Consider the 8-data points of two dimensions that we had considered earlier.

There are 8 clusters to start with, where each of the clusters has one element. The distance matrix using Mahhattan distance is as shown below

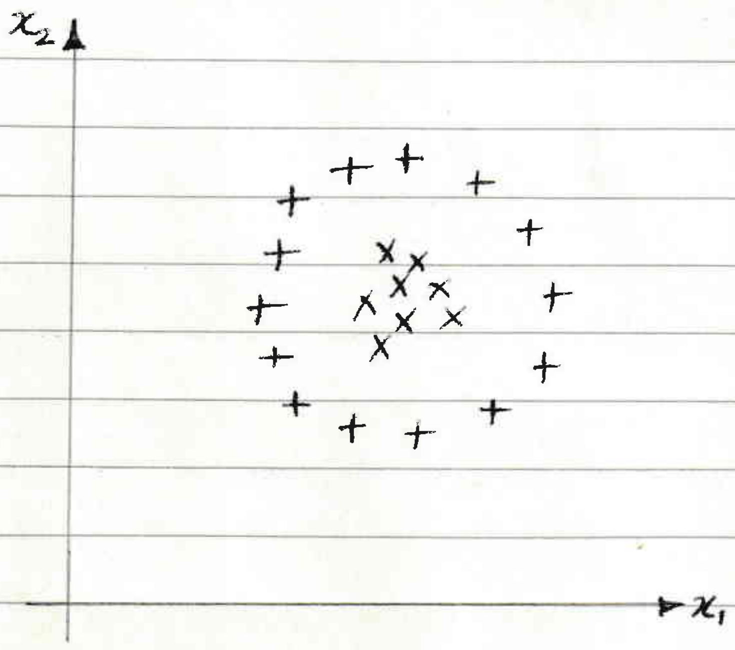
	A	B	C	D	E	F	G	H
A	0	2·5	1·5	5	5·75	7	7·5	4
B	2·5	0	1	3·5	4·25	4·5	5	1·5
C	1·5	1	0	3·5	4·25	5·5	6	2·5
D	5	3·5	3·5	0	0·75	2	2·5	5
E	5·75	4·25	4·25	0·75	0	2·75	2·25	5·75
F	7	4·5	5·5	2	2·75	0	0·5	3
G	7·5	5	6	2·5	2·5	0·5	0	3·5
H	4	1·5	2·5	2·5	5·75	3	3·5	0

Since the clusters {F} and {G} are the closest to each other, with a distance of 1·5 units, they are merged to realise a partition of 7 clusters. The updated similarity matrix after one merger is as shown below

	A	B	C	D	E	F, G	H
A	0	2·5	1·5	5	5·75	7	4
B	2·5	0	1	3·5	4·25	4·5	1·5
C	1·5	1	0	3·5	4·25	5·5	2·5
D	5	3·5	3·5	0	0·75	2	5
E	5·75	4·25	4·25	0·75	0	2·25	5·75
F, G	7	4·5	5·5	2	2·75	0	3
H	4	1·5	2·5	2·5	5·75	3	0

Clusters {D} and {E} are the next closest with a distance of 0·75 units and they are merged next. In a similar manner, clusters {B} and {C} are next merged. Then the cluster comprising B and C is merged with the cluster {A}. This cluster is then merged with the cluster {H}. Next, the cluster consisting of {D, E} and the cluster consisting of {F, G} are merged. At this stage, there are two clusters. The process can be stopped whenever we have the desired clusters.

This algorithm characterises the presence of a pattern in a cluster, based on it's nearest neighbor in the cluster. The algorithm is highly versatile, and can generate clusters of different shape. For example, the algorithm can generate the 2-partition of the data, forming concentric circular arrangement as shown.



Hierarchical clustering algorithms are computationally expensive. The agglomerative algorithms for example require computation and storage of a matrix in each step escalating time and space requirement. They can be used in applications where hundreds of patterns were to be clustered. However, when the data sets are larger in size, these algorithms are not feasible because of the non-linear time and space demands. It may not be easy to visualize a dendrogram corresponding to 1000 patterns.

Similarly, divisive algorithms require exponential time to analyse the number of patterns. They too do not scale up well in the context of large-scale problems involving a huge number of patterns.

Partitional clustering -

Partitional clustering algorithms generate a hard or soft partition of the data. The most commonly used algorithm in this category is the k-means algorithm.

k-means algorithm-

The algorithmic description is as follows-

1) Select 'k' out of the given 'n' patterns as the initial cluster centers. Assign each of the remaining n-k patterns to one of the 'k' clusters; a pattern is assigned to it's closest centre/cluster.

2) Compute the cluster centers based on the current assignment of patterns.

3) Assign each of the 'n' patterns to it's closest centre/cluster.

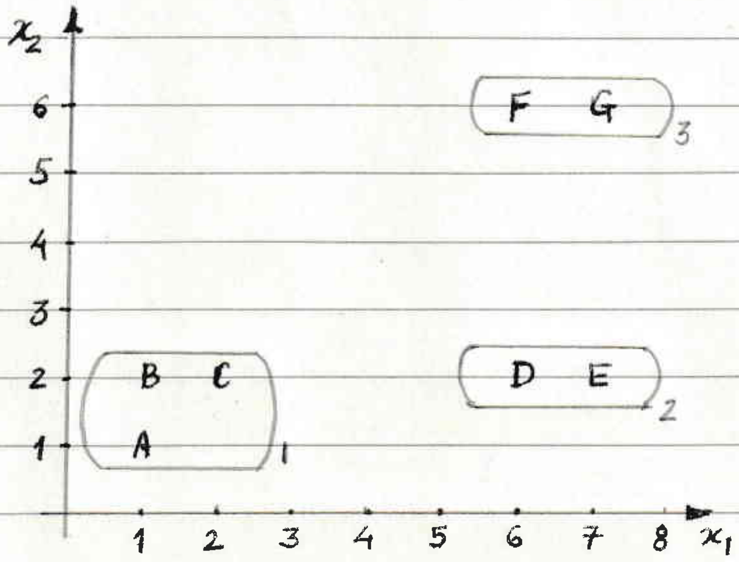
4) If there is no change in the assignment of patterns to clusters during two successive iterations, then stop else go to step 2.

There are a variety of schemes for selecting the initial cluster centers; these include selecting the first 'k' of the 'n' given patterns, selecting 'k' random patterns out of the given 'n' patterns, and viewing the initial cluster seed selection as an optimisation problem, and using a robust tool to search for the globally optimal solution. Selecting the initial cluster centre is a very important issue in deciding performance.

Example- Consider the patterns located as below-

A = (1,1) ; B = (1,2) ; C = (2,2) ; D = (6,2) ;
E = (7,2) ; F = (6,6) ; G = (7,6)

It is decided to generate a 3 partition output i.e. k = 3.



If A, D and F are selected as initial centres, cluster 1 has (1,1) as it's cluster centre, cluster 2 has (6,2) as it's cluster centre and cluster 3 has (6,6) as it's cluster centre. The remaining patterns are to be assigned to the cluster which is closest to them. Thus B and C are assigned to cluster 1; E is assigned to cluster 2 and G is assigned to cluster 3.

The new cluster centre of Cluster 1 will be the mean of patterns in cluster 1 (i.e. mean of A, B and C) which will be (1·33, 1·66). The centre of cluster 2 will be (6·5, 4) and the cluster centre of cluster 3 will be (6·5, 6). The patterns are again assigned to the closest cluster depending on the distance from the cluster centres. Now, A, B and C are assigned to cluster 1, D and E are assigned to cluster 2 & F and G are assigned to cluster 3. Since it is observed that there is no change in the clusters formed, this is the final set of clusters.

This gives a visually appealing partition of the 3 clusters as shown above. In mathematical terms, the variance in the clusters is acceptable.

An important property of the k-means algorithm is that it implicitly minimises the sum of squared deviations of patterns in a cluster from the centre. If C_i is the i^{th} cluster and μ_i is it's centre, then the criteria function minimised by the algorithm is

$$\sum_{i=1}^{k} \sum_{x \in C_i} (x-\mu_i)^t (x-\mu_i)$$

This is considered as sum of squared error criteria for clustering.

A frequently used heuristic is to select the initial 'k' centers in such a way that they are as far away from each other as possible. Such schemes are observed to work well in practice. The time complexity of the algorithm is of the order $O(nkdl)$, where 'l' is the number of iterations and 'd' is the dimensionality. The space requirement is of the order $O(kd)$. These features make the algorithm very popular for implementation in a variety of applications. Some of the applications involve large volumes of data, such as satellite image data. It is best to use k-means algorithm when the clusters are hyper-spherical.