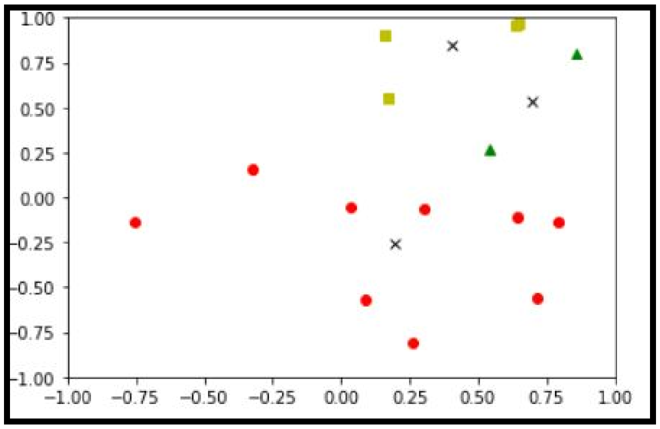
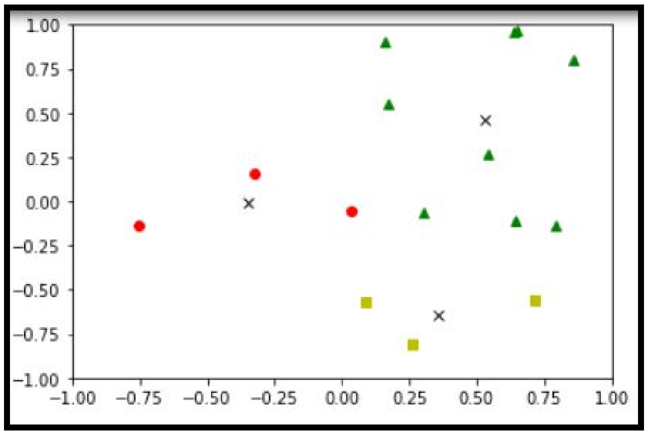
Chirag Shah **Text Analytics** **COMP47600**
19200072 **Practical Number 7** **06/10/2019**

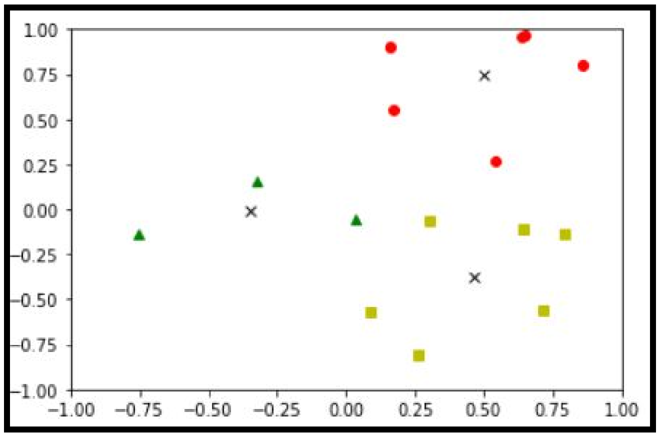
Q1.) Use the init_board fn to randomly generate 15 points; store this output and set the data variable to it Now run this set 10 times and note the clusters found by k-means Report the results of these runs and the extent to which the same clusters are found

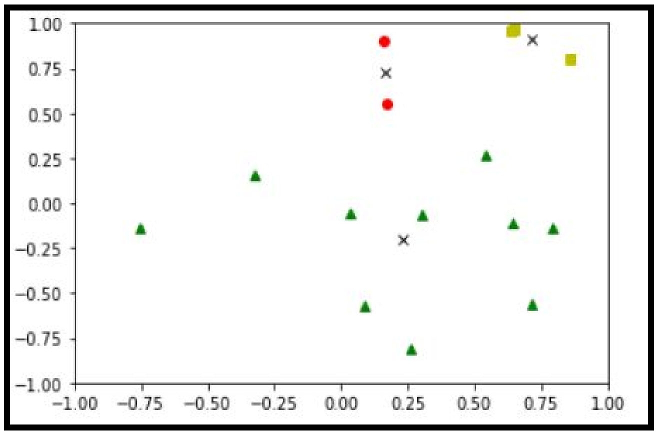
K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data. It tries to partition the dataset into *K*-pre-defined distinct non-overlapping subgroups randomly. It makes the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid is minimised. The less variation we have within clusters, the more homogeneous the data points are within the same cluster.

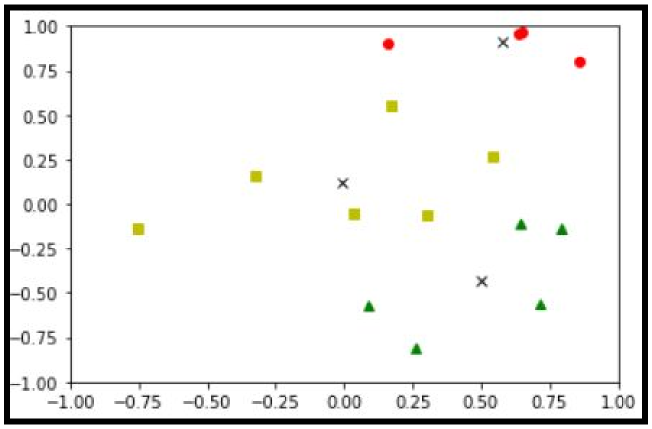
The provided k-means algorithm is made to run 10 times and following are the results of similar clusters obtained -

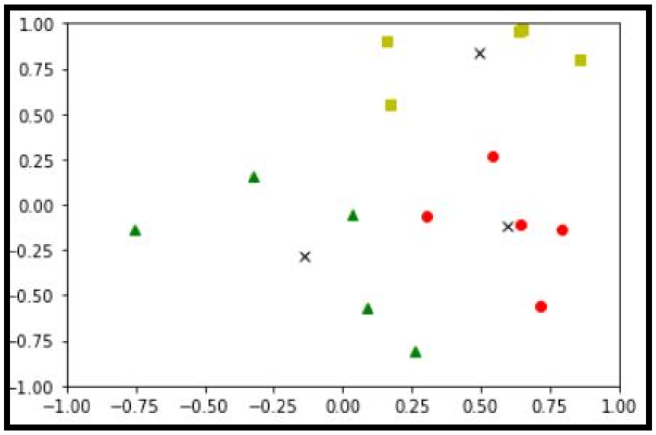








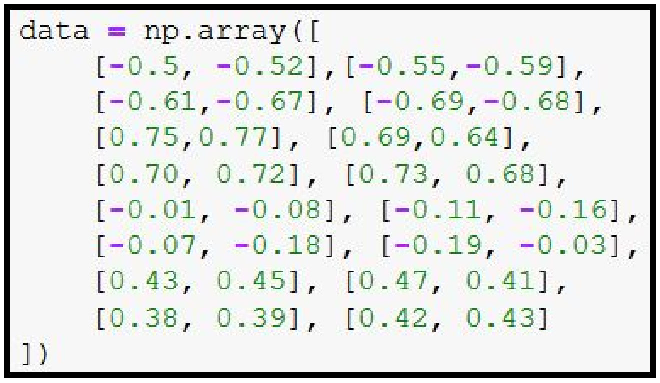




We can see that out of 10 runs there are 6 unique results of clusters obtained. The remaining 4 results produce same clusters from the above 6 unique results.

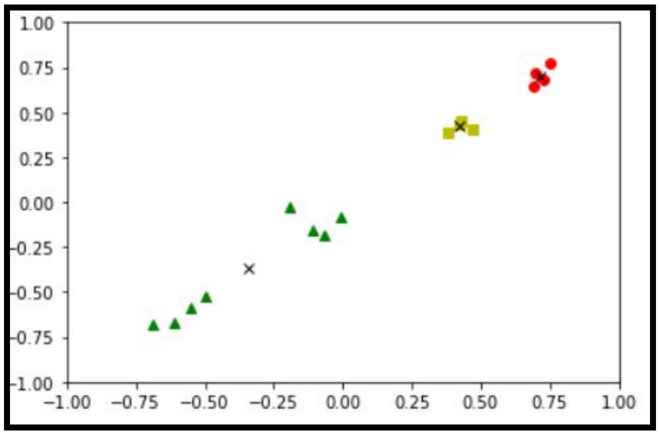
Q2.) Now, create your own set of data, again with 15 Points You should construct this data-set to have very clear clusters (a bit like the simple 6-point example shown) Now run this set 20 times and note the clusters found by k-means Report the results of these runs and the extent to which the same clusters are found

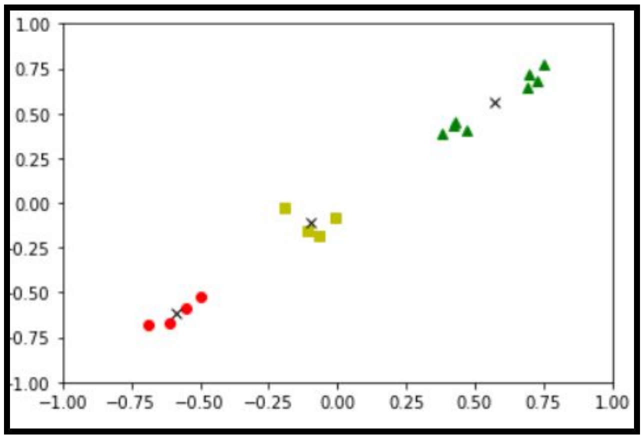
Following are the data points taken -



```
data = np.array([
    [-0.5, -0.52],[-0.55,-0.59],
    [-0.61,-0.67], [-0.69,-0.68],
    [0.75,0.77], [0.69,0.64],
    [0.70, 0.72], [0.73, 0.68],
    [-0.01, -0.08], [-0.11, -0.16],
    [-0.07, -0.18], [-0.19, -0.03],
    [0.43, 0.45], [0.47, 0.41],
    [0.38, 0.39], [0.42, 0.43]
])
```

The provided k-means algorithm is made to run 20 times and following are the results of similar clusters obtained -





From the above results, we infer that there are 2 variations produced in multiple runs. The remaining 18 runs had clusters that are same from among the above 2 variants. Thus we can infer that when we have points chosen closely to each other in a systematic manner, there are less variations in clusters produced.

Q3.) Do some research on the problem that k-means produces different answers on different runs Describe two typical solutions to this problem with references to the literature you read to answer his question.

The K-means algorithms does not guarantee unique clustering and every different choice of initial cluster centers may lead to different clustering results. Kmeans places the initial seeds (cluster centers) randomly. So each run will have a different initial set of seed locations, and as such (slightly) different outcomes. **Because of this random initialisation of seeds, K-means produces different answers on different runs.** (Why choosing proper initial centroids is very important for K-means?)

Since K-Means randomly assigns initial centroids and shifts them to minimize the sum of squares, the bad starting positions of initial centroids causes the algorithm to converge at a local optimum. With better initial seeds, the K-means algorithm converges faster and produces good quality clusters.

It is computationally intractable to produce a global optimum for K-means problem(S. Dasgupta. The hardness of k-means clustering)

Following are some of the techniques to improvise initialisation of initial seeds -

1. **Specific random seed value**.

This ensures that the same input samples are chosen across multiple runs and then the best result among multiple runs could serve a base for using initial cluster points. This results in better initialization of centroids and there is a good probability that it can produce clusters with same center points on multiple runs. However, this doesn't entirely solve the problem of different answers on different runs.

2. **Subsampling and choosing random.**

Randomly split the total sample into subsamples and to perform K-means on each, then again averaging the final centres and running clustering of the total sample.()

- RUNFP - *farthest points.*

  First k cases are taken as centres and then during the run through the rest of the cases of the dataset there progressively replacements among the centres are done; the aim of the replacements is to obtain in the end k points most distant from each other in the variable space.

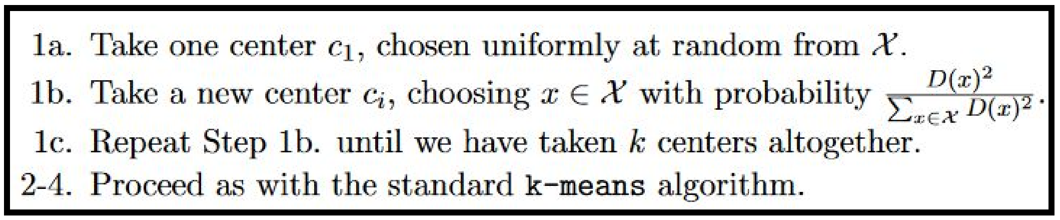
- GREP - *group representative points*.

  To collect as centres k most representative, "deputy" cases. The 1st centre is taken as the case closest to the general data centroid. Then the rest of the centres are selected from the data points in such a way that each point is considered as to whether it is closer (and how much, in terms of squared euclidean distance) to a set of points than each one of the latter is to any of the already existing centres.

- KMPP - *random farthest points, or k-means++.*

  The first centre is selected as a random case from the dataset. The 2nd centre is selected also randomly, but the probability of selection of a case is proportional to the distance (square euclidean) of it to that (1st) centre. The 3rd centre is selected also randomly with the probability of selection proportional to the distance of a case to the nearest of those two centres

Following is the K-means algorithm - (k-means++: The Advantages of Careful Seeding)



```
1a.  Take one center $c_1$, chosen uniformly at random from $\mathcal{X}$.
1b.  Take a new center $c_i$, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$.
1c.  Repeat Step 1b. until we have taken $k$ centers altogether.
2-4. Proceed as with the standard k-means algorithm.
```

All of the above mechanisms are non-deterministic in a sense that they do work on the factor of randomness to initialise results and produce different clusters always.

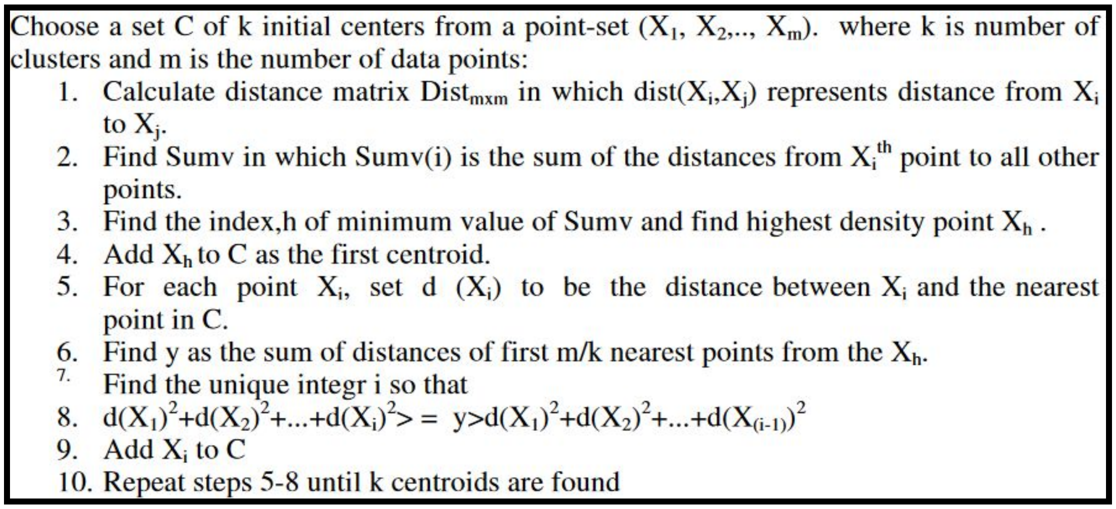
3. Single Pass Seed Selection (SPSS) algorithm

It is a modification to k-means++, in the sense that it is a method of initialization to k-means type algorithms. The SPSS initialize first seed and the minimum distance that separates the centroids based on highest density point, which is close to more number of other points in the data set. Following are the important steps that differentiate the initial seeding via SPSS from k-means++ :

→ Initialize first centroid with a point which is close to a number of other points in the data set.

→ Assume that m (total number of points) points are distributed uniformly to k (number of clusters) clusters then each cluster is expected to contain m/k points. Compute the sum of the distances from the selected point to first m/k nearest points and assume it as y.

Algorithm - (Robust seed selection algorithm for k-means type algorithm)



```
Choose a set C of k initial centers from a point-set ($X_1, X_2,.., X_m$). where k is number of clusters and m is the number of data points:
  1. Calculate distance matrix $Dist_{mxm}$ in which dist($X_i$,$X_j$) represents distance from $X_i$ to $X_j$.
  2. Find Sumv in which Sumv(i) is the sum of the distances from $X_i^{th}$ point to all other points.
  3. Find the index,h of minimum value of Sumv and find highest density point $X_h$ .
  4. Add $X_h$ to C as the first centroid.
  5. For each point $X_i$, set d ($X_i$) to be the distance between $X_i$ and the nearest point in C.
  6. Find y as the sum of distances of first m/k nearest points from the $X_h$.
  7. Find the unique integr i so that
  8. $d(X_1)^2+d(X_2)^2+...+d(X_i)^2>= y>d(X_1)^2+d(X_2)^2+...+d(X_{(i-1)})^2$
  9. Add $X_i$ to C
  10. Repeat steps 5-8 until k centroids are found
```

When SPSS is combined with k-means++, it tries to eliminate the pseudo-randomness in choosing the initial centroid.

4. Sorting based Heuristics -

Sort data points according to either of the following criterion -

a.) Distance to center point - Sort the data points according to their distance to the center of the data. The centroids are then selected as every N / k th point in this order. We include this variant in our tests. To have randomness, we choose a random data point as a reference point instead of the center. This heuristic fulfills our requirements: it is fast, simple, and requires no additional parameters. (A k-means clustering algorithm, J.A. Hartigan)

b.) Density - Here the density is based on the number of other points within a distance d1 . First centroid is the point with the highest density, and the remaining k -1 centroids are chosen at a decreasing order, with the condition that they are not closer than distance d2 from an already chosen centroid. (Initializing k-means batch clustering: a critical evaluation of several techniques)

c.) *Attribute with greatest variance* - sorts the data points according to the dimension with the largest variance. The points are then partitioned into k equal size clusters. Median of each cluster is selected instead of the mean. This approach belongs to a more general class of projection-based techniques where the objects are mapped to some linear axis such as diagonal or principal axis. (A new algorithm for cluster initialization, M. Al-Daoud) d.) *Centrality* - Here a primary criterion is used(cohesion) to estimate how central a point is (how far from boundary). Secondary threshold criterion (coupling) is used to prevent centroids from being neighbors. (An initialization method for the k-means algorithm using neighborhood model, F. Cao , J. Liang , G. Jiang)

After sorting, k points are selected from the sorted list using one of the following heuristics -

1. First k points.
2. First k points while disallowing points closer than ε to already chosen centroids.
3. Every (N / k)th point (uniform partition).

References:

S. Dasgupta. The hardness of k-means clustering. Technical report, University of California, San Diego,2008.

k-means++: The Advantages of Careful Seeding, David Arthur and Sergei Vassilvitskii, Available at: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

Robust seed selection algorithm for k-means type algorithm, Available at : https://arxiv.org/ftp/arxiv/papers/1202/1202.1585.pdf

J.A . Hartigan , M.A . Wong , Algorithm AS 136: a k-means clustering algorithm, J. R. Stat. Soc. C 28 (1) (1979) 100–108, Available at: http://refhub.elsevier.com/S0031-3203(19)30160-8/sbref0060

D. Steinley, M.J. Brusco, Initializing k-means batch clustering: a critical evaluation of several techniques, J. Classification 24 (2007) 99–121, Available at: http://refhub.elsevier.com/S0031-3203(19)30160-8/sbref0020

F. Cao , J. Liang , G. Jiang , An initialization method for the k-means algorithm using neighborhood model, Comput. Math. Appl. 58 (2009) 474–483, Available at: http://refhub.elsevier.com/S0031-3203(19)30160-8/sbref0063

M. Al-Daoud, A new algorithm for cluster initialization, in: World Enformatika Conference, 2005, pp. 74–76, Available at: http://refhub.elsevier.com/S0031-3203(19)30160-8/sbref0062

Methods of initialising k means clustering, Available at - https://stats.stackexchange.com/questions/317493/methods-of-initializing-k-means-clustering

Why choosing proper initial centroids is very important for K-means, Available at - https://stats.stackexchange.com/questions/214323/why-choosing-proper-initial-centroids-is-very-important-for-k-means