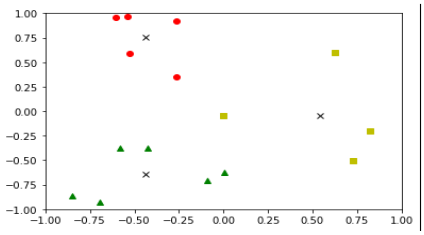
Q1) Clusters found by 10 Iterations on K-means:

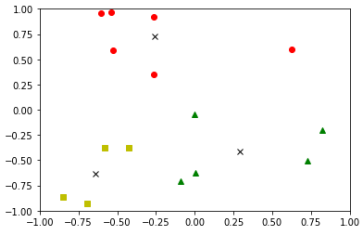
Code Snippet

```
data = init_board(15)
for i in range(1,10):
    out = find_centers(data, 3)
    parse_output(out)
```

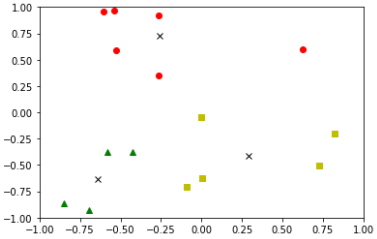
Iteration 1



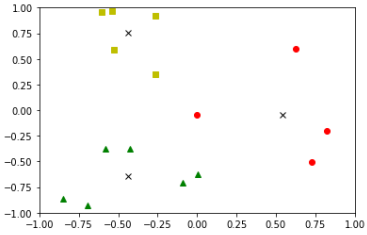
Iteration 2



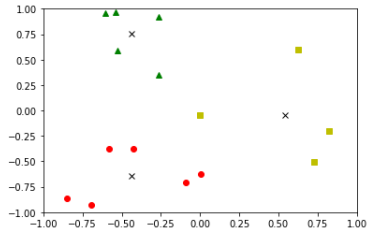
Iteration 3



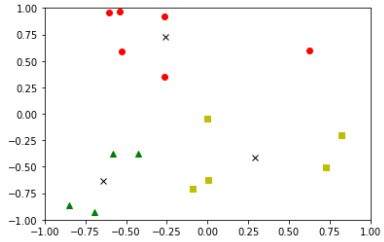
Iteration 4



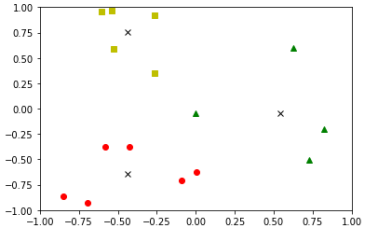
Iteration 5



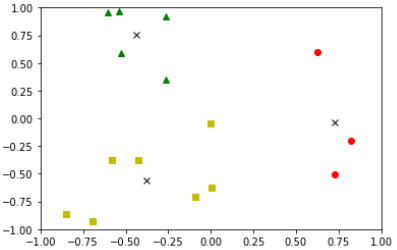
Iteration 6



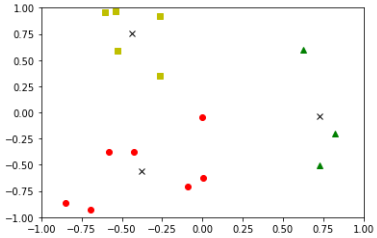
Iteration 7



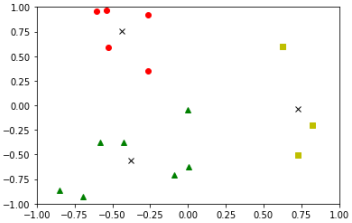
Iteration 8



Iteration 9



Iteration 10



Observations :

For a given set of datapoints very often the same clusters were produced, with minor fluctuations observed throughout the 10 iterations.

Three different sets of clusters are produced – The clusters 1 and 4 form first set of similar clusters. The clusters 2,3 and 6 form Second set and the clusters 5, 7,8,9 and 10 form the third set of similar clusters.

1. The K-means algorithm works on the basis of initially selected **random datapoints as centroids** and the further cluster assignments are done for the remaining datapoints based on their proximity(using Euclidian Distance) to different chosen centroids .
2. The selected centroids are further initialized to the mean of the cluster to which it belongs to.
3. In order to form the best possible clusters, the algorithm continues an iterative process till it gets a stable value of the centroids and no change in cluster assignment.

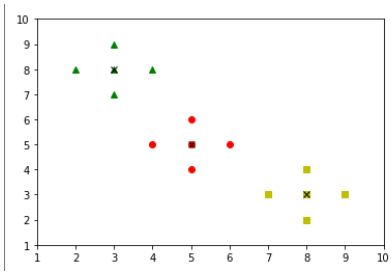
4. This random selection of centroid and the above steps which are dependent on centroid selection is the reason why K-means forms different clusters for each iteration.

Q2) Data set to construct 20 points to form well separated clusters:

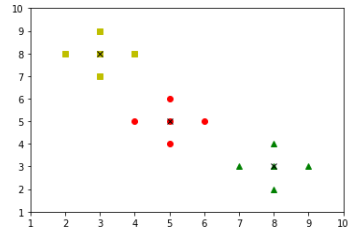
Datapoints used:

data = np.array([[5,5],[5,4],[5,6],[4,5],[6,5],[3,8],[2,8],[4,8],[3,7],[3,9],[8,3],[7,3],[9,3],[8,4],[8,2]])

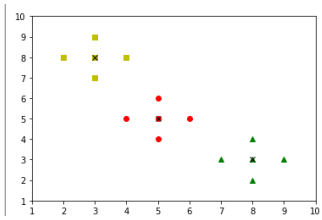
Iteration 1



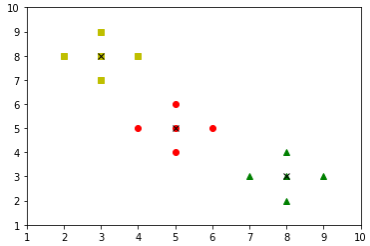
Iteration 1



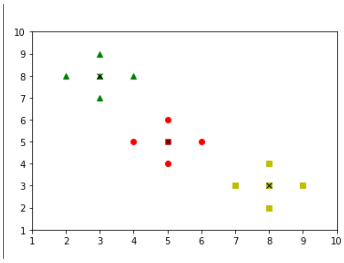
Iteration 1



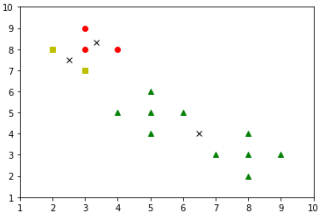
Iteration 4



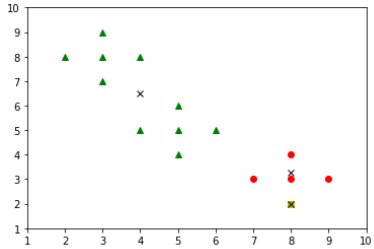
Iteration 5



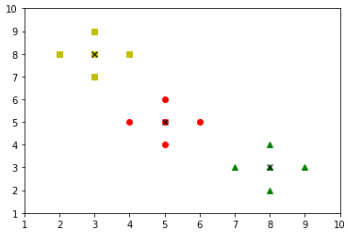
Iteration 6



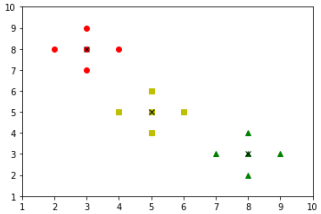
Iteration 7



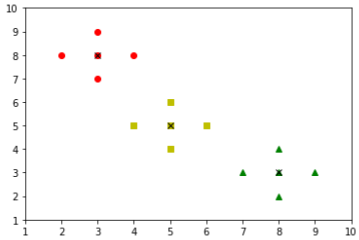
Iteration 8



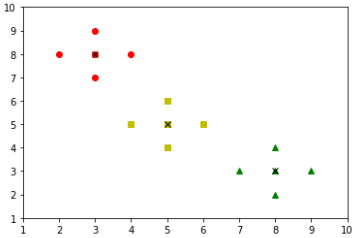
Iteration 9



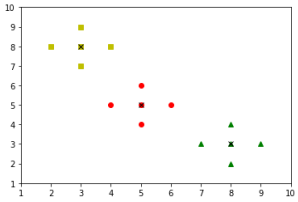
Iteration 10



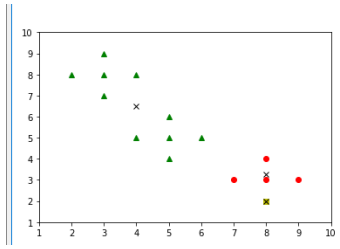
Iteration 11



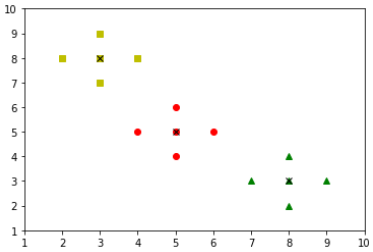
Iteration 12



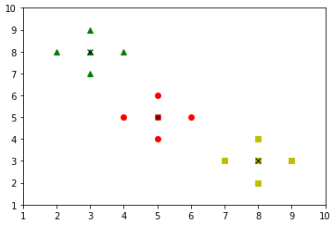
Iteration 13



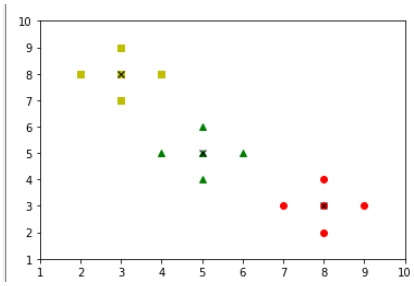
Iteration 14



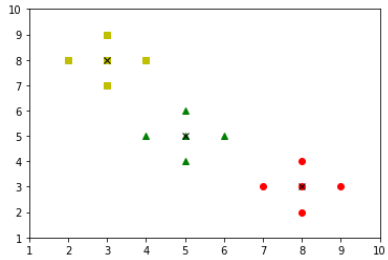
Iteration 15



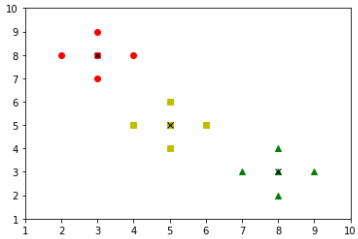
Iteration 16



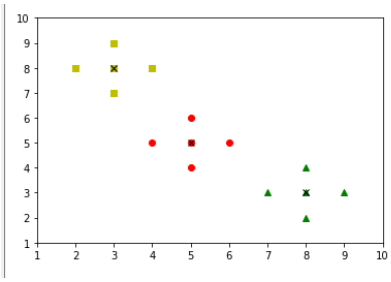
Iteration 17



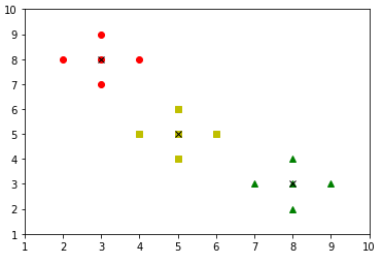
Iteration 18



Iteration 19



Iteration 20



Observation :

1. In all the 20 iterations almost similar plots are seen except for the 6th ,7th and 13th iterations which show a slight fluctuation and all other clusters are good clusters as there can be seen an exact separation between them.
2. Similar to Q1, all the clusters formed by k-means should have been similar but K-means selects different centroids randomly so the difference is observed in clusters even for the same datapoints.
3. But in this scenario, the accuracy is 85%.This is the case because the points for clustering are selected in such a way that they are equally spaced from each centroid(And the centroids are far from the every other centroid) and also the datapoints follow a linear path.

Q3) Problems with K-means :

The K-means algorithm starts with the initial step of selecting random centroids where each selected centroid is different from the other.

This selection of **random centroids** often leads to different clusters for each run and can be stabilized based on the use **of heuristic of selecting random points but iterate till a optimal output is obtained**. Further, the datapoints are assigned to a cluster based on their Euclidian distance from a particular centroid. **The noise and outliers in data often leads convergence to local optimum as noise often tends to shift the centroid from its actual position** which is one of the important drawback of K-means algorithm.

Improvement for K-means :

In order to improve the performance of K-means following two approaches can be used :

K-means and k-medoids.

A)**K-means++ :**

1. The k-means++ algorithm helps us select a particular way to select the centroids for k-means algorithm.
2. The first centroid is selected at a random from the given data points .
3. For calculating the next centroid the distance-squared for each of the datapoints from the initially selected mean is calculated.
4. The datapoints which have smaller distance from the first selected centroids will have a very less or no probability of getting selected as a next centroid and the ones with longer distance have the high probability of getting selected. This technique of selection is often termed as 'Proportional fitness selection'.
5. After the initial clusters are formed, the algorithm later works in a similar manner to k-means and to get best possible clusters
6. Thus k-means ++ algorithm solves the initial stage of selecting centroids which are very random and at the same time clusters with well -distinguished boundaries can be formed .

K-means++ algorithm The exact algorithm is as follows(Wikipedia):

1. Choose one center uniformly at random from among the data points.
2. For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x)^2.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard k-means clustering.

B)**K-Medoids :**

1. The k-Medoids algorithm uses medoids instead of mean value as a reference point for initial stage in the algorithm
2. The medoid is nothing but the most centrally located datapoint in the cluster.
3. This algorithm works well for noisy data as medoid is not affected by the biasness of the dataset unlike the mean .
4. Also the k-Medoids algorithm solves the problem of local optimum as it applies the swapping mechanism for medoid selection .
5. i.e. after assigning objects to a particular cluster, a non-medoid object is selected and its distance from the remaining points in the cluster is calculated if the distance is less the randomly selected object becomes the new medoid of the cluster.

K-Medoids Algorithm : The PAM Partitioning Around Medoids (PAM) is most commonly used flavor of K-medoids :

1. Initialize: randomly select *k* of the *n* data points as the medoids
2. Assignment step: Associate each data point to the closest medoid.
3. Update step: For each medoid *m* and each data point *o* associated to *m* swap *m* and *o* and compute the total cost of the configuration (that is, the average dissimilarity of *o* to all the data points associated to *m*). Select the medoid *o* with the lowest cost of the configuration.

Repeat alternating steps 2 and 3 until there is no change in the assignments.

References :

1. https://stats.stackexchange.com/questions/214323/why-choosing-proper-initial-centroids-is-very-important-for-k-means
2. https://www.coursera.org/lecture/cluster-analysis/3-4-the-k-medoids-clustering-method-nJ0Sb
3. https://msdn.microsoft.com/en-us/magazine/mt185575.aspx
4. https://en.wikipedia.org/wiki/K-means%2B%2B#Improved_initialization_algorithm
5. http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html