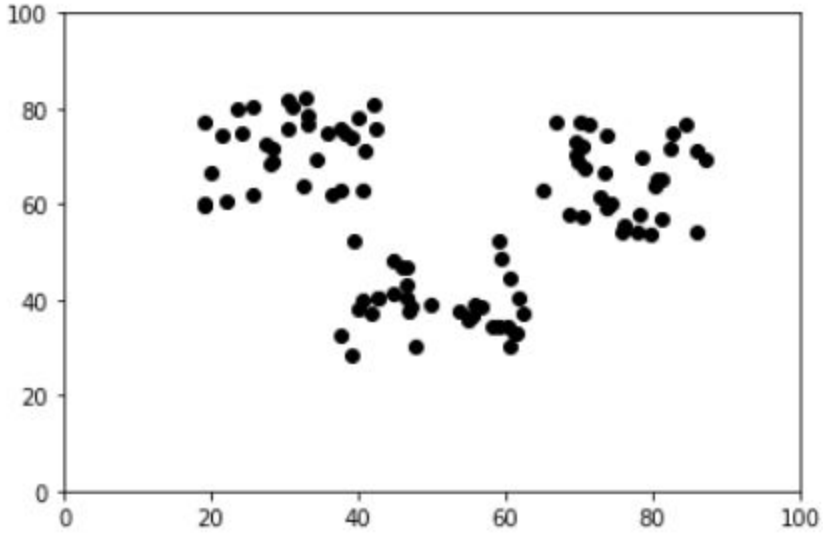
CSL 7020 - Assignment 3
K-means clustering

Vishakh.S (B16CS038)

---

1. DATASET

100 points have been considered in the two-dimensional XY space. For visualizing the clustering process clearly, these points have been generated by spreading points randomly around 3 specific points (pivots) in the 2D space .i.e. more specifically (30, 70), (75, 65) and (50, 40). The resulting distribution looks somewhat like this.



2. APPROACH

K-means clustering is one of the simplest and most popular unsupervised machine learning algorithms. A **cluster** is a set of data points aggregated together because of certain similarities. To begin with, you define a target number **k**, the number of clusters required in the target distribution. A **centroid** is an imaginary or real location representing the center of the cluster. K-means algorithm assigns each point to one of these k clusters (to be specific, the nearest) while striving to keep the in-cluster sum of squares minimum.

There are 3 steps:

- **Initialisation** – K initial “means” (centroids) are generated at random.

- **Assignment** – K clusters are created by associating each data point with the nearest centroid .i.e. each point is assigned a cluster label equal to that of the nearest cluster.
- **Update** – The centroid of the clusters is updated to be the new mean.

Assignment and Update are repeated iteratively until the centroids stabilize. The **iterative procedure** is shown below:

input: $\mathcal{X} \subset \mathbb{R}^n$; Number of clusters k
initialize: Randomly choose initial centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
repeat until convergence
$\forall i \in [k]$ set $C_i = \{\mathbf{x} \in \mathcal{X} : i = \text{argmin}_j \|\mathbf{x} - \boldsymbol{\mu}_j\|\}$
(break ties in some arbitrary manner)
$\forall i \in [k]$ update $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$

3. EXPERIMENTS

a. SETUP

➔ Setup Python3 virtual environment.

➔ Install copy, random, numpy, matplotlib and pandas.

➔ Generate 100 points in the 2D space.

➔ Fix a value for the number of clusters (k = 3) and initialize them with random values in the range.

➔ For each data point, compute the distances to each of the 3 centroids. Find the minimum of the 3 distances. The assigned cluster label is the label of the closest centroid.

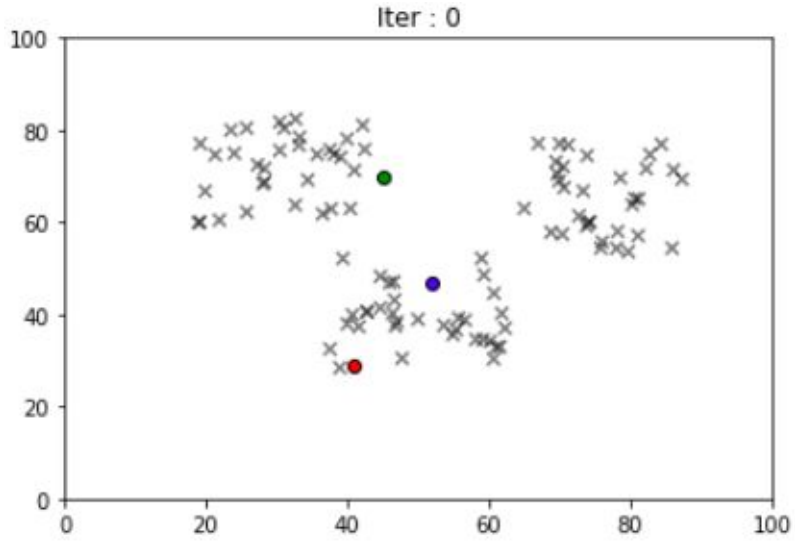
➔ Update the 3 centroids with the mean values of the newly obtained cluster points. This is the value of the centroid coordinate which minimizes the in-cluster sum of squares.

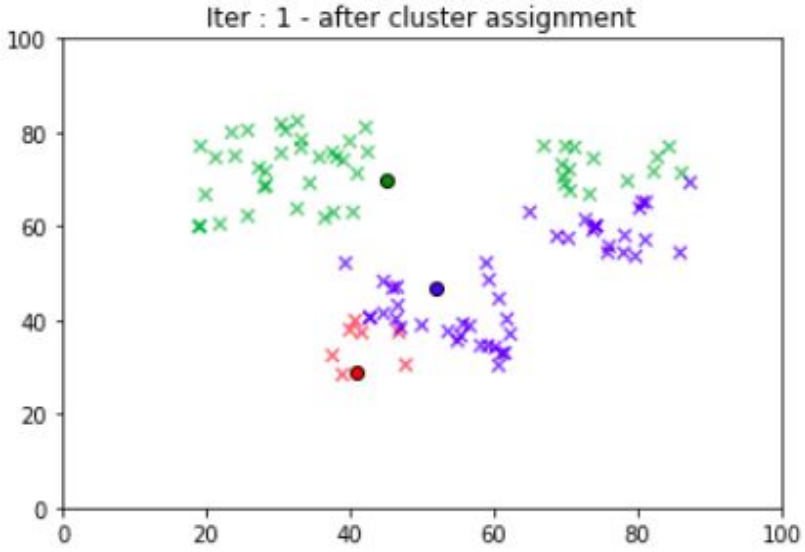
➔ Repeat the above two steps until the centroid value doesn't change.

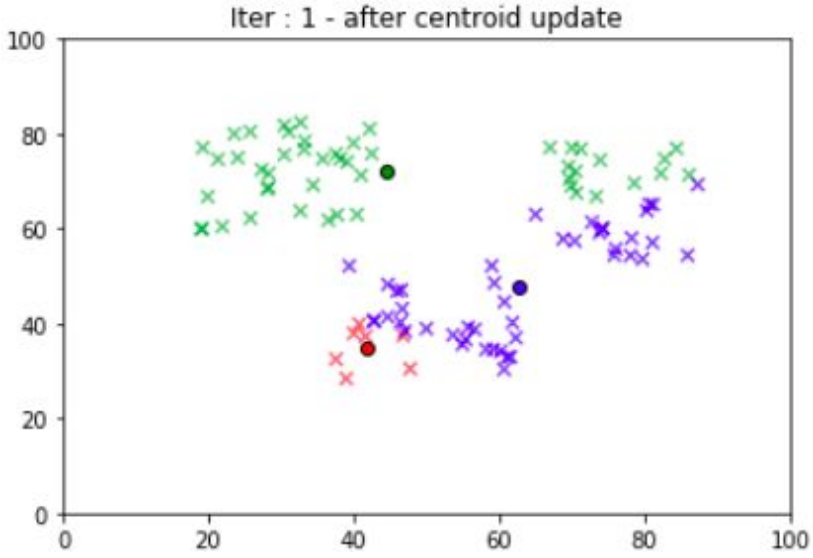
b. RESULTS

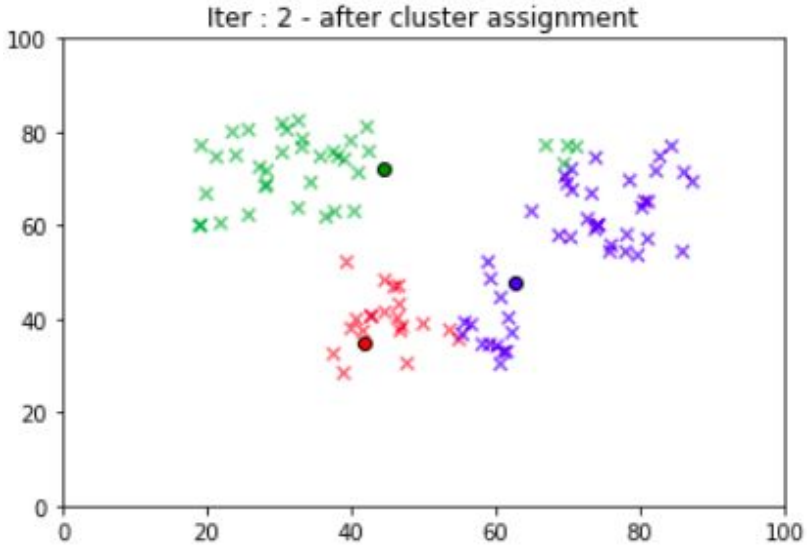
The centroid converges after a finite number of iterations.

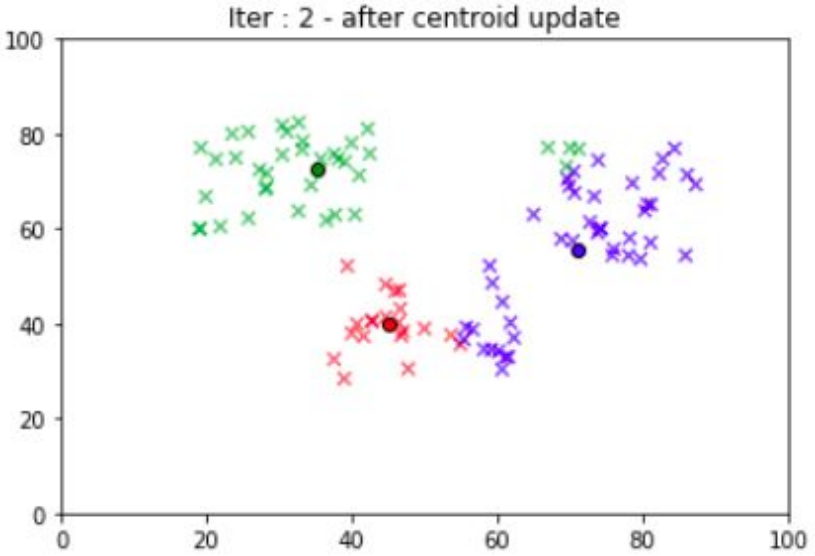
The following figures depict the process. (Here, for the given random selection of points, the centroid converges in 5 steps)

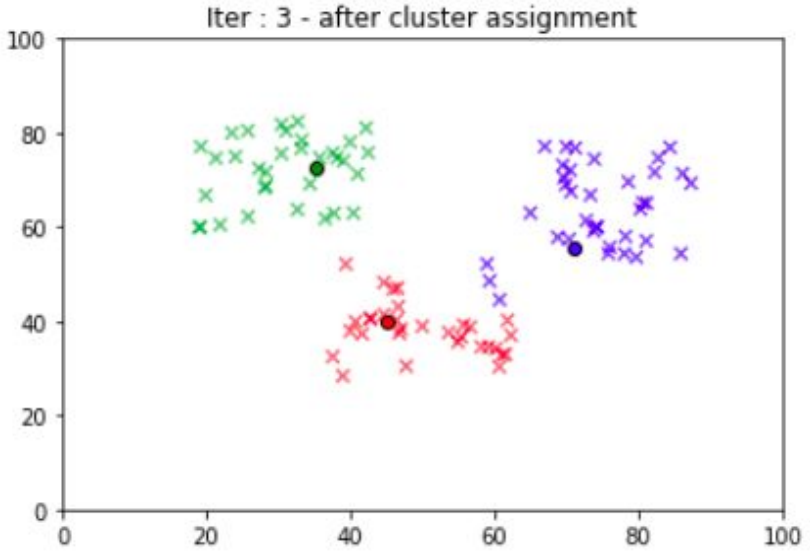

Iter : 0
100
80
60
40
20
0
0
20
40
60
80
100

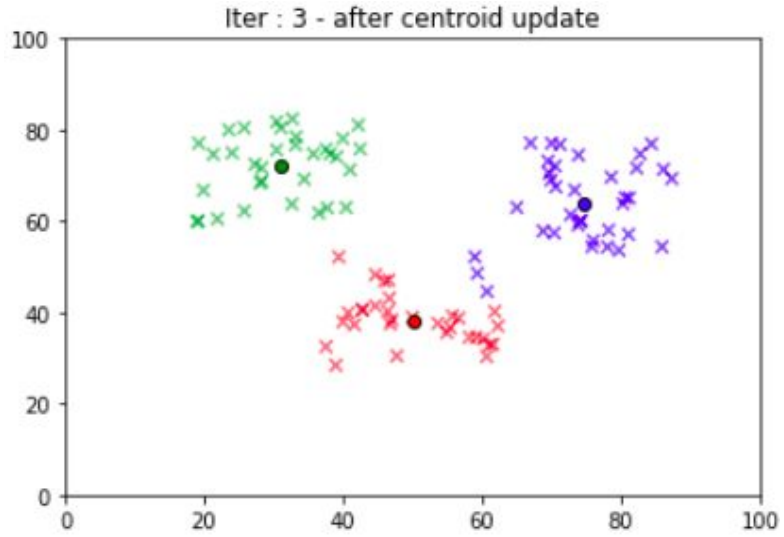

Iter : 1 - after cluster assignment
100
80
60
40
20
0
0
20
40
60
80
100

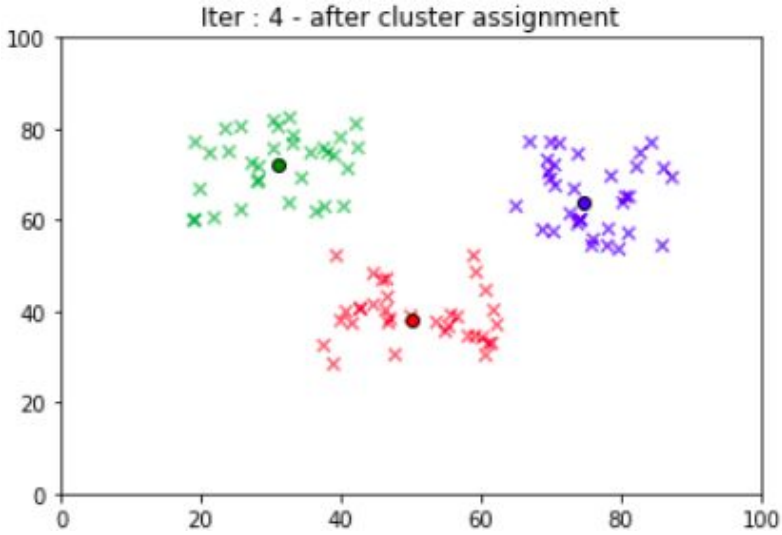

Iter : 1 - after centroid update
100
80
60
40
20
0
0
20
40
60
80
100

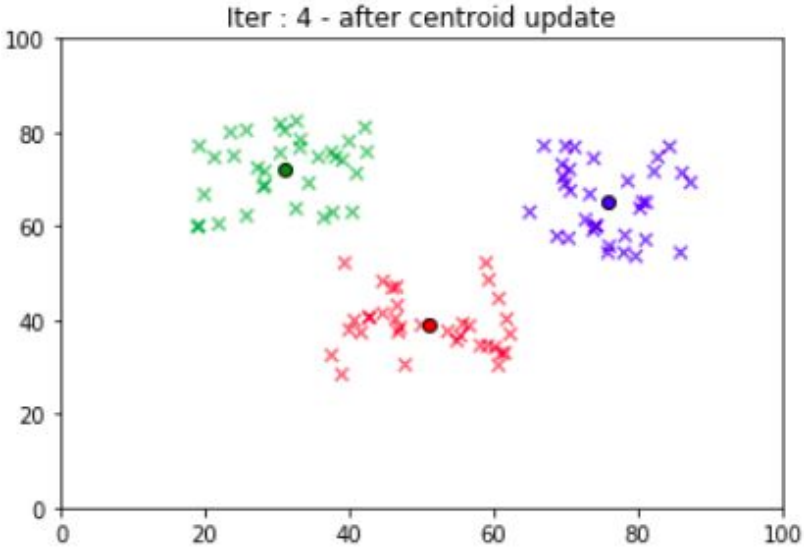

Iter : 2 - after cluster assignment
0
20
40
60
80
100

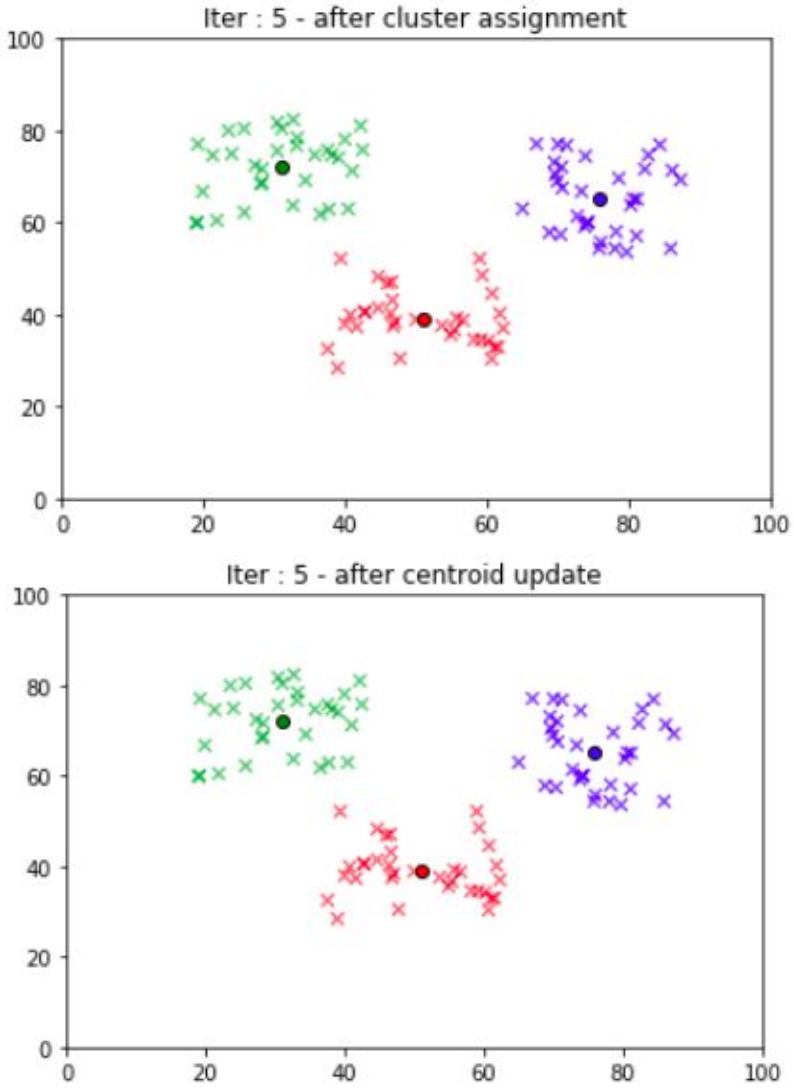

Iter : 2 - after centroid update
0
20
40
60
80
100


Iter : 3 - after cluster assignment
0
20
40
60
80
100


Iter : 3 - after centroid update
0
20
40
60
80
100

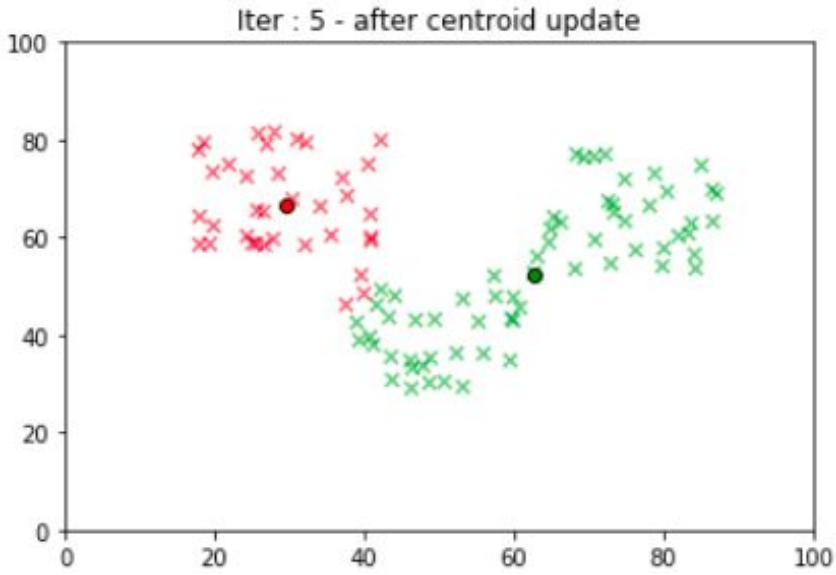

Iter : 4 - after cluster assignment
0
20
40
60
80
100


Iter : 4 - after centroid update
0
20
40
60
80
100



4. ANALYSIS

- The process is **heavily dependant on the initial values of the centroids**.
- If one of the initial centroids is too far away from the points, we occasionally end up with only 2 clusters as shown in the next figure. This is due to the fact that the initial value of the third centroid was so extreme that none of the data points were able to identify it as the closest centroid.


Iter : 5 - after centroid update
100
80
60
40
20
0
0
20
40
60
80
100