K-means clustering

1. Data

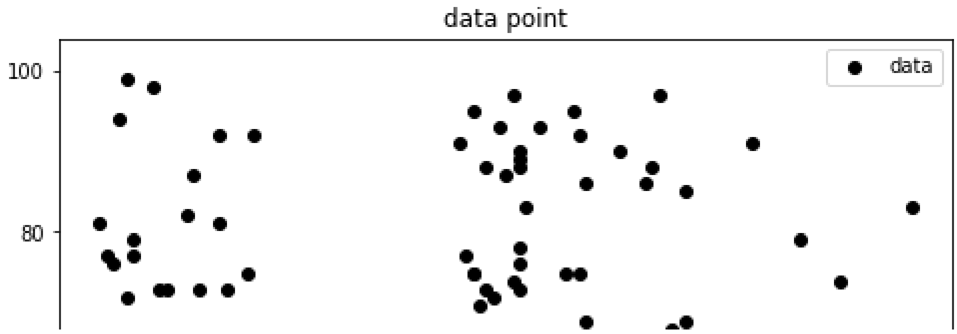
- the data are given by the file data-kmeans.csv
- the data consist of a set of points $\{(x_i, y_i)\}_{i=1}^{n}$ where $z_i = (x_i, y_i)$ denotes a 2-dimensional point in the cartesian coordinate and n is given as 200

load the data from the files

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random as rd

path = '/content/drive/My Drive/ML_Assignment/data/data-kmeans.csv'
dataset = pd.read_csv(path)
data = dataset.values
x_data = data[:,0] # x
y_data = data[:,1] # y
```

Plot the data points

```
fig_1 = plt.figure(figsize = (8,8))
plt.scatter(x_data, y_data, c='k', label='data')
plt.title('data point')
plt.legend()
plt.show()
fig_1.savefig('data point.png')
```



2. Loss

- the loss function $\mathcal{L}(C_1, C_2, \cdots, C_k, \mu_1, \mu_2, \cdots, \mu_k)$ with a given number of clusters k for a set of data $\{z_i\}_{i=1}^n$ is defined by:
$$\mathcal{L}(C_1, C_2, \cdots, C_k, \mu_1, \mu_2, \cdots, \mu_k) = \frac{1}{n} \sum_{i=1}^n \| z_i - \mu_{l(z_i)} \|_2^2 = \frac{1}{n} \sum_{j=1}^k \sum_{z_i \in C_j} \| z_i - \mu_j \|_2^2$$

> - $l(z) = k$ is a label function that defines a label k of point z
> - C_k denotes a set of points $\{z_i | l(z_i) = k\}$ of label k
> - μ_k denotes a centroid of points in C_k

define a function to compute a initial centroid

```
def init_centroid(k):
  centroids = np.array([]).reshape(2,0)
  for i in range(k):
    rand = rd.randint(0, 200)
    centroids = np.c_[centroids, data[rand]]

  return centroids.T
```

define a function to compute a distance between two points a and b

```
def compute_distance(data, c):

    dist = np.array([]).reshape(200,0)

    # distance between data and cluster
    for i in range(5):
      i_dist = np.sqrt(np.sum((data - c[i,:])**2, axis=1))
      dist = np.c_[dist, i_dist]

    return dist
```

```
def compute_centroid_distrance(c):
    dist = []
    # distance between data and cluster
    for i in range(5):
      i_dist = np.sqrt(np.sum((c[i,:])**2))
      dist.append(i_dist)
```

```
    return dist
```

define a function to compute a centroid from a given set of points Z

```
def compute_centroid(cluster):
    center = np.array([]).reshape(2,0)
    # centroid of a set of points in Z
    for i in range(5):
        idx = (cluster[:,2]==i)
        i_center = np.mean(data[idx],axis=0)
        center = np.c_[center, i_center]
    return center.T
```

define a function to compute the loss with a set of clusters C and a set of centroids M

```
def compute_loss(cluster, centroids):
    loss_list = []
    loss = 0

    for i in range(5):
      idx = (cluster[:,2]==i)
      i_loss = np.sqrt(np.sum((data[idx] - centroids[i,:])**2))
      loss += i_loss

    loss = loss / len(cluster)
    return loss
```

3. Optimization

- the label $l(z)$ of each point z is determined by: $l(z) = \arg\min_k \|z - \mu_k\|_2^2$
- the centroid μ_i of cluster k is determined by: $\mu_k = \frac{\sum_{z_i \in C_k} z_i}{|C_k|}$

define a function to determine the label of point z with a set of centroids M

```
def compute_label(dist):

    argmin_label = np.argmin(dist, axis=1) #label of point z with a set of centroids M#
    label = np.c_[data, argmin_label]

    return label
```

4. Clustering

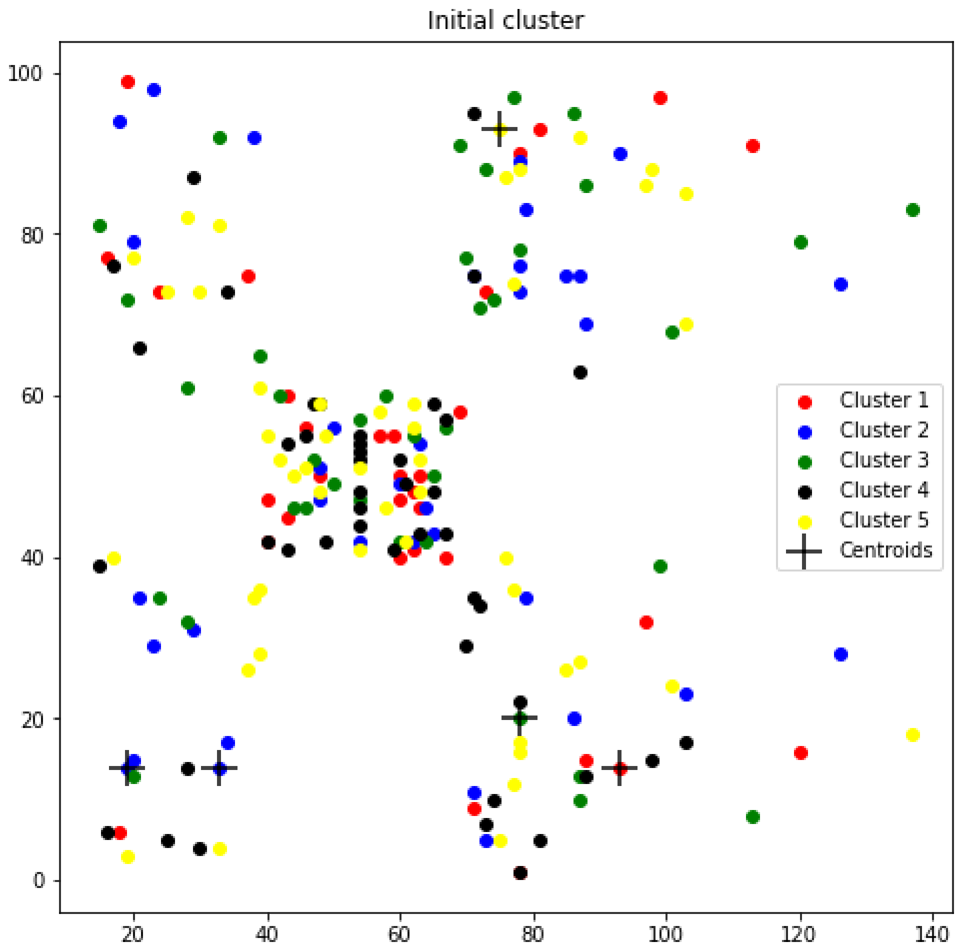
- initialise labels $l(z_i)$ for point z_i for all i randomly

- optimise the loss function with respect to the centroids and the clusters in an alternative way
- set the number of clusters $k = 5$

Visualise the initial condition of the point labels

```
k = 5   # set the number of clusters
n = len(data)
max_iter = 50
centroids = init_centroid(k)
labels = np.random.randint(low=0, high=k, size=n)
result = np.c_[data, labels]
```

```
fig_2 = plt.figure(figsize = (8,8))
color=['red','blue','green', 'black', 'yellow']
label=['Cluster 1','Cluster 2','Cluster 3', 'Cluster 4', 'Cluster 5']
for i in range(k):
    idx = (result[:,2]==i)
    plt.scatter(x_data[idx],y_data[idx], c=color[i],label=label[i])
plt.scatter(centroids[:,0],centroids[:,1],s=300, c='k', marker='+', label='Centroids')

plt.title('Initial cluster')
plt.legend()
plt.show()
fig_2.savefig('Initial cluster.png')
```



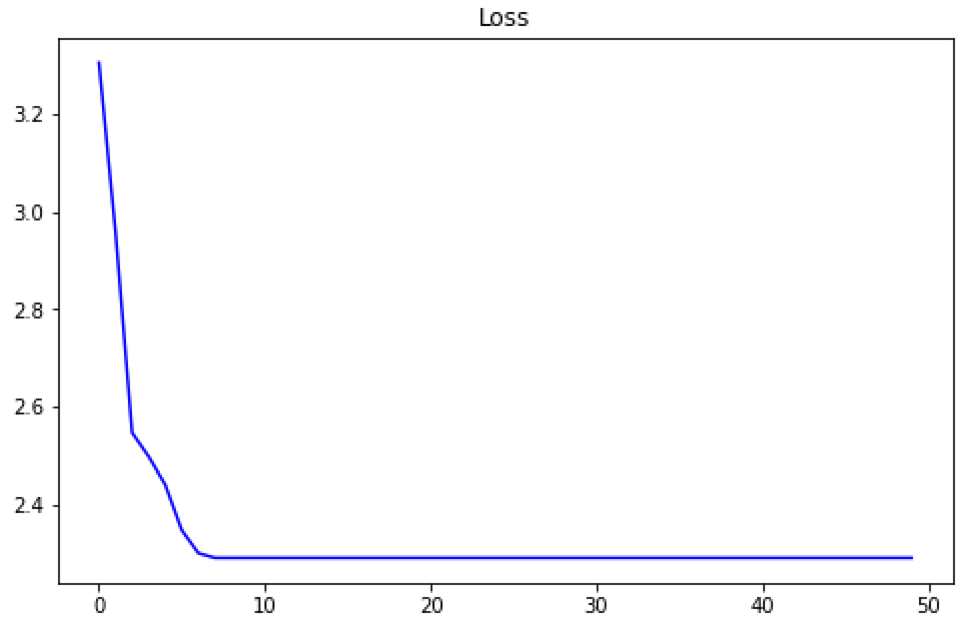

```
def k_means_clustering(max_iter, data, centroids):
```

```
    loss_iters = [] # record the loss values
    centroid_iters = []

    for i in range(max_iter):
        dist = compute_distance(data, centroids)
        cluster = compute_label(dist)
        centroids = compute_centroid(cluster)
        loss = compute_loss(cluster, centroids)
        c_dist = compute_centroid_distrance(centroids)
        loss_iters.append(loss)      # save the current loss value
        centroid_iters.append(c_dist)

    return cluster, centroids, loss_iters, centroid_iters
```

```
final_result, final_c, loss_iter, c_iter = k_means_clustering(max_iter, data, centroids)
```

Plot the loss curve

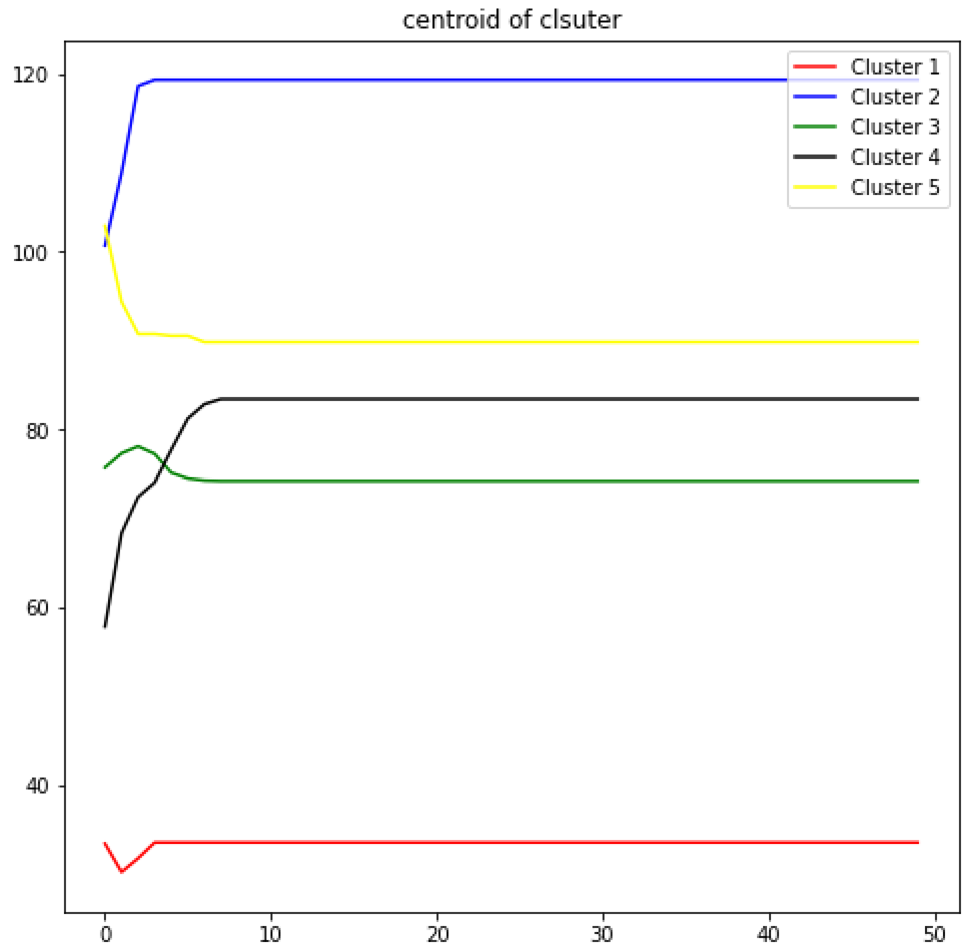
```
# Plot the loss curve
fig_3 = plt.figure(figsize = (8,5))
plt.plot(np.array(range(max_iter)),loss_iter, c = 'b')
plt.title('Loss')
plt.show()
fig_3.savefig('Loss.png')
```



Plot the centroid of each clsuter

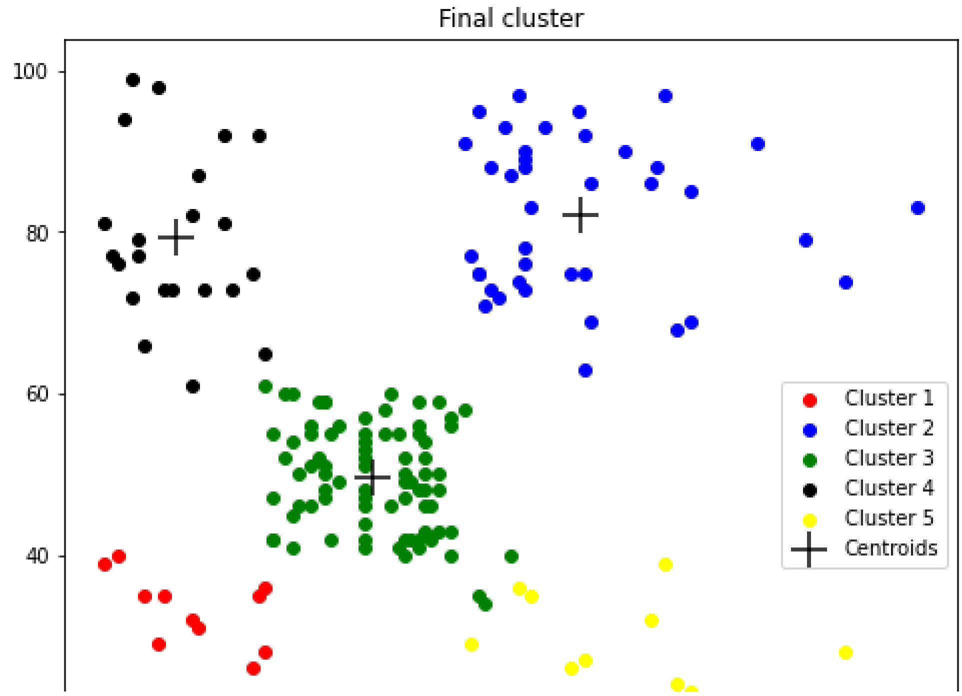
```
# Plot the centroid of each clsuter
fig_4 = plt.figure(figsize = (8,8))
color=['red','blue','green', 'black', 'yellow']
label=['Cluster 1','Cluster 2','Cluster 3', 'Cluster 4', 'Cluster 5']
np_c_iter = np.array(c_iter)

for i in range(k):
    idx = (final_result[:,2]==i)
    plt.plot(np.array(range(max_iter)), np_c_iter[:,i], c=color[i],label=label[i])
```

```
    plt.plot(np.array(range(max_iter)), np_c_iter[:,i], c=color[i],label=label[i])

plt.title('centroid of clsuter')
plt.legend(loc = 'upper right')
plt.show()
fig_4.savefig('centroid of clsuter.png')
```



Plot the final clustering result

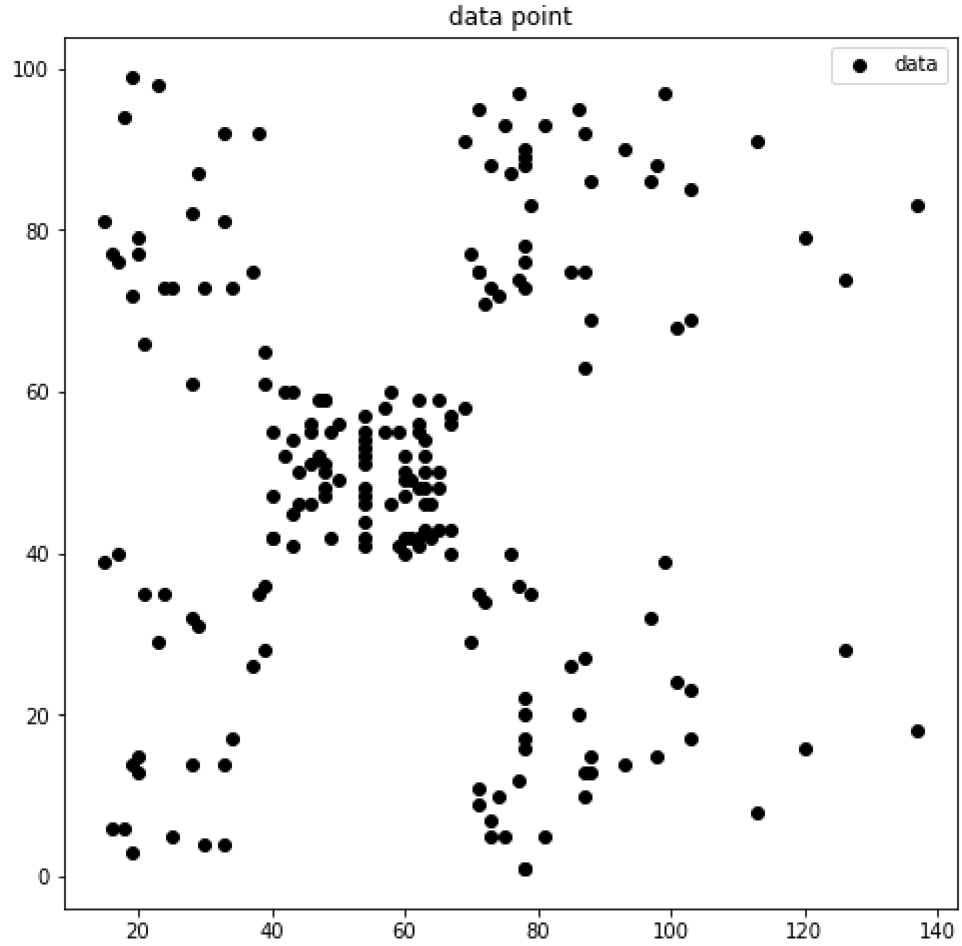
```
fig_5 = plt.figure(figsize = (8,8))
color=['red','blue','green', 'black', 'yellow']
label=['Cluster 1','Cluster 2','Cluster 3', 'Cluster 4', 'Cluster 5']
for i in range(k):
    idx = (final_result[:,2]==i)
    plt.scatter(x_data[idx],y_data[idx], c=color[i],label=label[i])
plt.scatter(final_c[:,0],final_c[:,1],s=300, c='k', marker='+', label='Centroids')

plt.title('Final cluster')
plt.legend()
plt.show()
fig_5.savefig('Final cluster.png')
```



Output

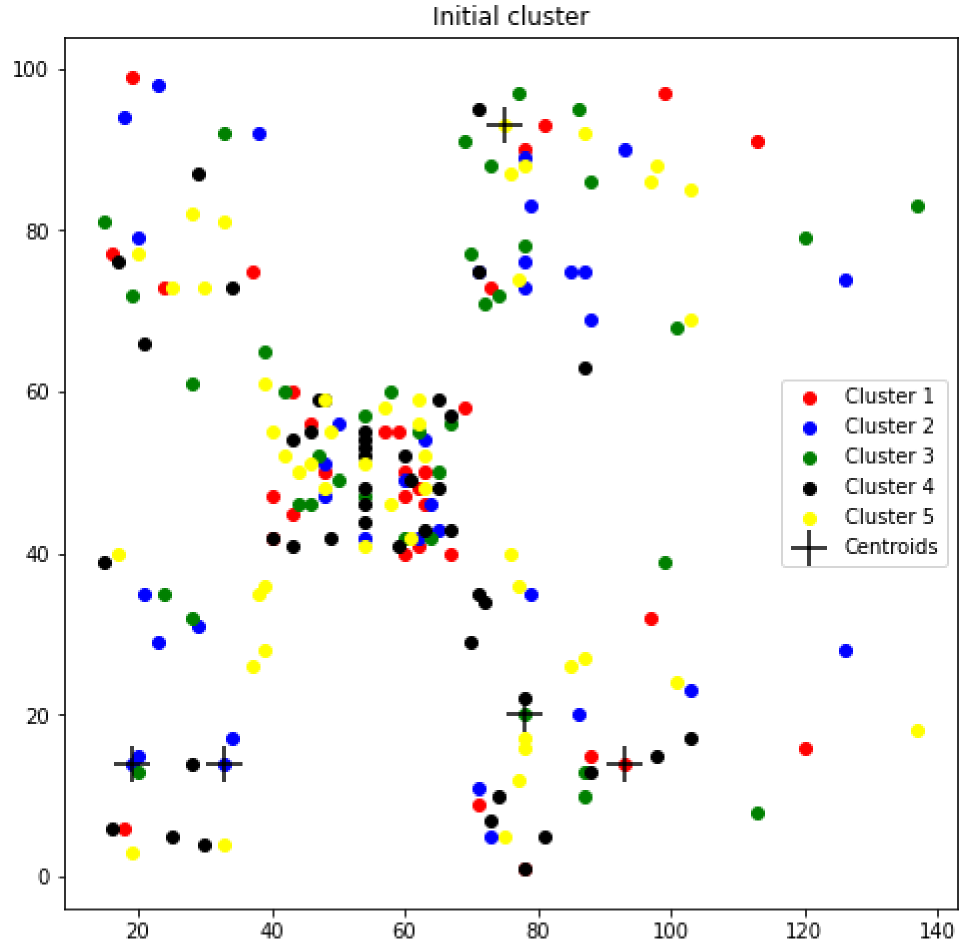
1. Plot the data points [1pt]

```
fig_1
```



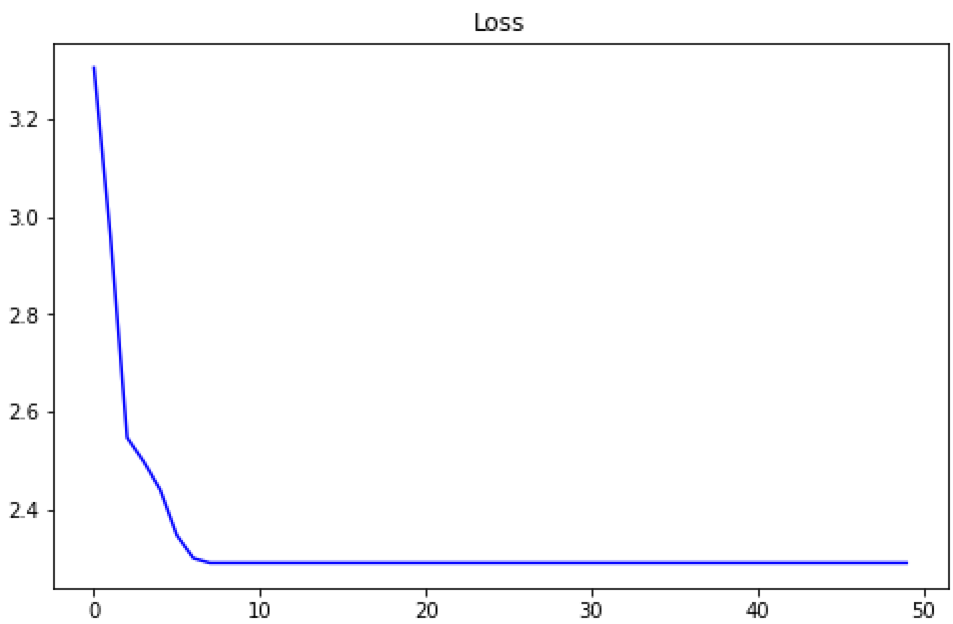
2. Visualise the initial condition of the point labels [1pt]

```
1    fig_2
```



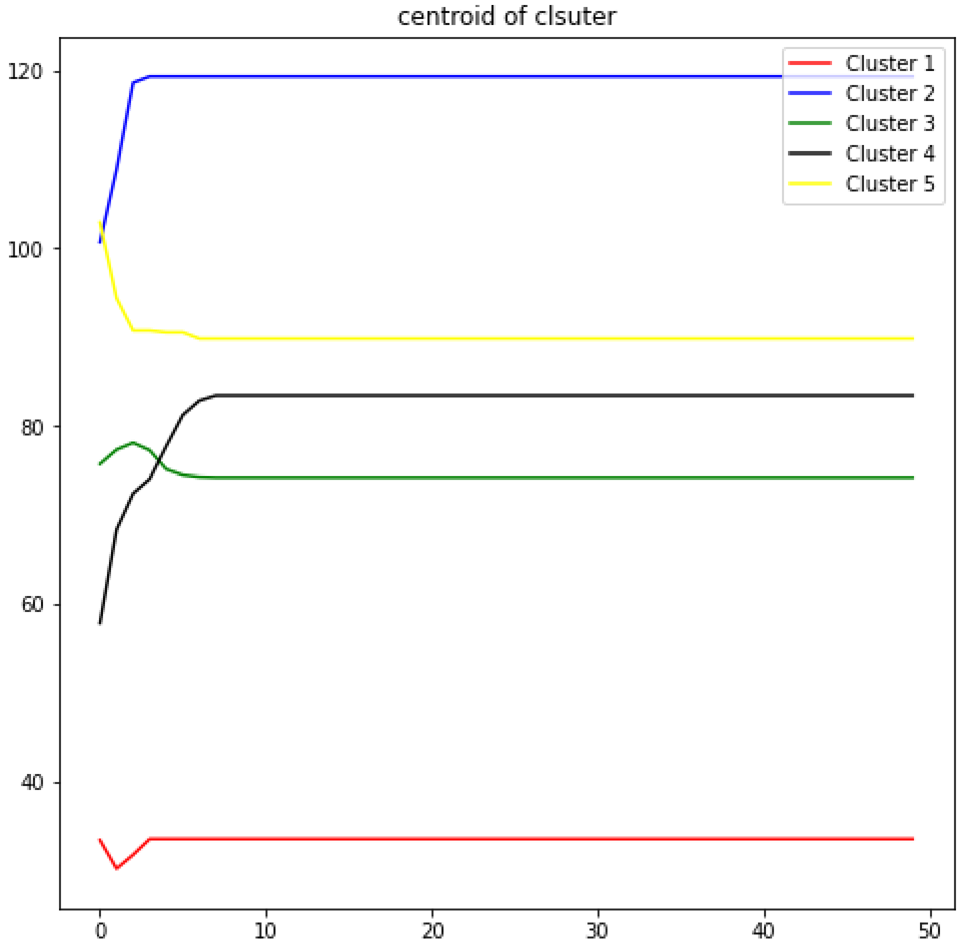
3. Plot the loss curve [5pt]

```
1    fig_3
```



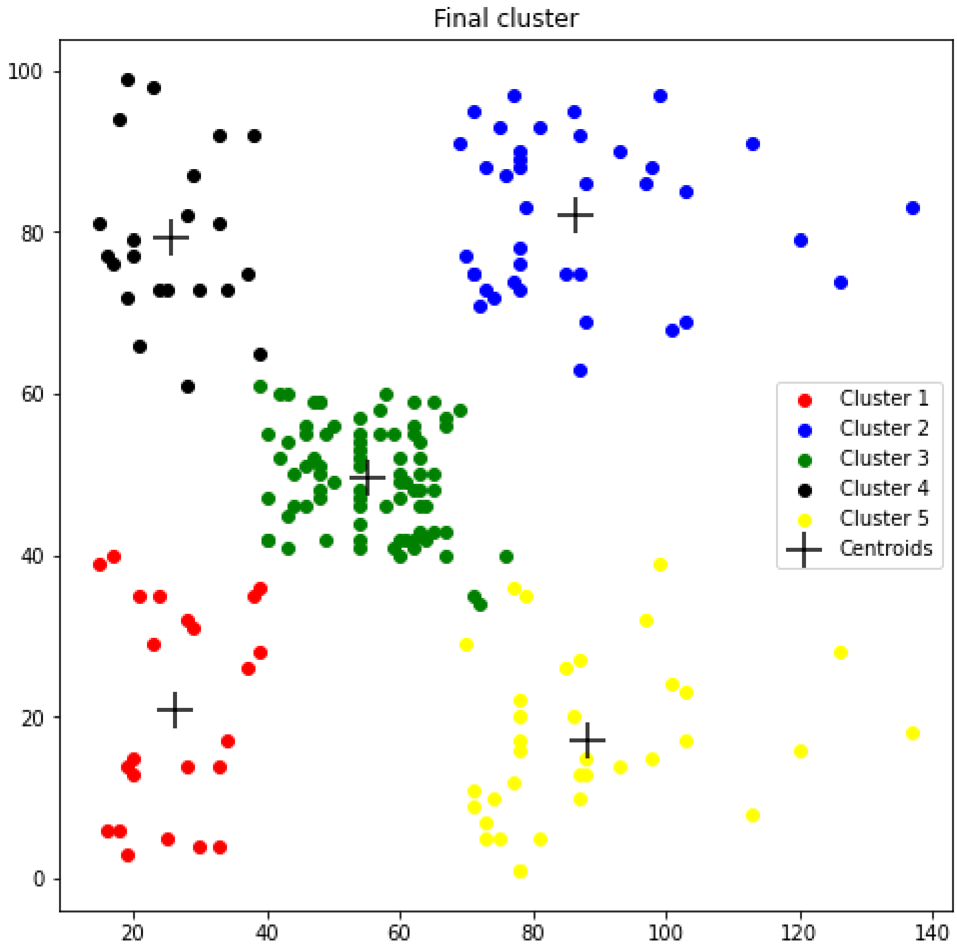
4. Plot the centroid of each clsuter [5pt]

```
1    fig_4
```



5. Plot the final clustering result [5pt]

```
1    fig_5
```


Final cluster
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
100
80
60
40
20
0
20
40
60
80
100
120
140