ML Assignment 4 Report

Aditya Sharma (2022038)

Section-A (Theory)

(A)

(a) Dimensions of the resulting feature map:

Given:

- Input image dimensions: $M \times N$ with P channels.
- Kernel size: $K \times K$.
- Stride: 1.
- No padding.

Feature map dimensions (height and width):
$H' = M - K + 1, \quad W' = N - K + 1$

Thus, the resulting feature map has dimensions $(M - K + 1) \times (N - K + 1)$.

(b) Number of elementary operations for a single output pixel:

For a single output pixel:

1. The kernel of size $K \times K$ operates over the $K \times K$ region of the input image.
2. Each kernel has P channels, so the operation involves $K \times K \times P$ weights.
3. Each weight involves:
    - 1 multiplication.
    - 1 addition (to accumulate the result).

Thus, the total operations for computing one output pixel:

$$\text{Number of operations per output pixel} = (K \times K \times P) \text{ multiplications } + (K \times K \times P - 1) \text{ additions.}$$

$$= 2 \times (K \times K \times P) - 1 \quad \text{elementary operations.}$$

(c) Solution Part C: Computational Time Complexity

1. **Dimensions and Computation:**

- Input image dimensions: $M \times N$
- Input channels: P
- Number of kernels: Q
- Kernel size: $K \times K$
- Output image dimensions: $(M - K + 1) \times (N - K + 1)$

2. **Computation for a Single Output Pixel:**

- Per kernel: $K^2 \times P$ multiplications
- Per kernel: $(K^2 \times P) - 1$ additions
- Total operations per output pixel for Q kernels: $Q \times [2(K^2 \times P) - 1]$

3. **Total Computations for Entire Feature Map:**

- Number of output pixels: $(M - K + 1) \times (N - K + 1)$
- Computation per output pixel for Q kernels: $Q \times [2(K^2 \times P) - 1]$

4. **Computational Time Complexity:**

- First Big-O notation (exact):
  $O(Q \times (M - K + 1) \times (N - K + 1) \times K^2 \times P)$

5. **Approximation when $\min(M, N) \gg K$:**

- When the image is much larger than the kernel
- $(M - K + 1) \approx M$
- $(N - K + 1) \approx N$

Simplified Big-O notation:
$O(Q \times M \times N \times K^2 \times P)$

(B)

K-Means Algorithm: In-Depth Analysis

Part 1: Assignment Step

Key Characteristics

- **Objective**: Assign each data point to the nearest cluster centroid
- **Primary Mechanism**: Distance-based classification

Mathematical Formulation

1. **Distance Calculation**:

- Euclidean distance formula:
  $d(x, c_k) = \sqrt{\sum_{i=1}^{n} (x_i - c_{k,i})^2}$
- Where:
  - x is the data point
  - c_k is the k-th cluster centroid

2. **Cluster Assignment Rule**:

- Assign to closest centroid:
  $cluster(x) = \arg\min_k d(x, c_k)$

Part 2: Update Step

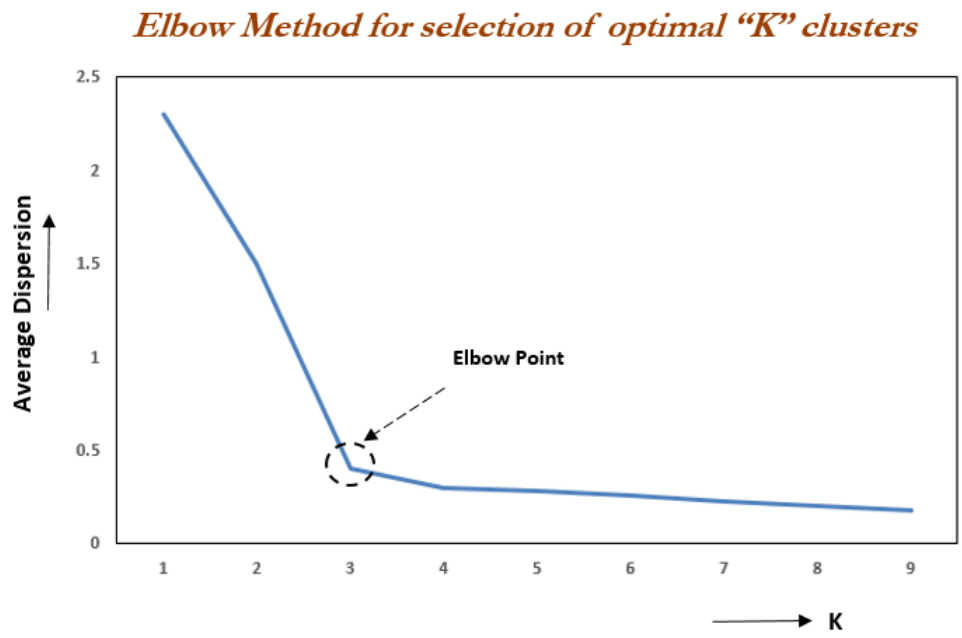
Centroid Recalculation

- Compute mean of all points in each cluster
- Formal representation:
  $c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$
- Where C_k is the set of points in cluster k

Part 3: Determining Optimal Clusters - Elbow Method

Methodology

- Plot within-cluster sum of squares (WCSS) against number of clusters
- Identify the "elbow point"



Key Characteristics

- WCSS decreases rapidly initially
- Becomes more gradual after optimal cluster count
- Represents trade-off between model complexity and variance

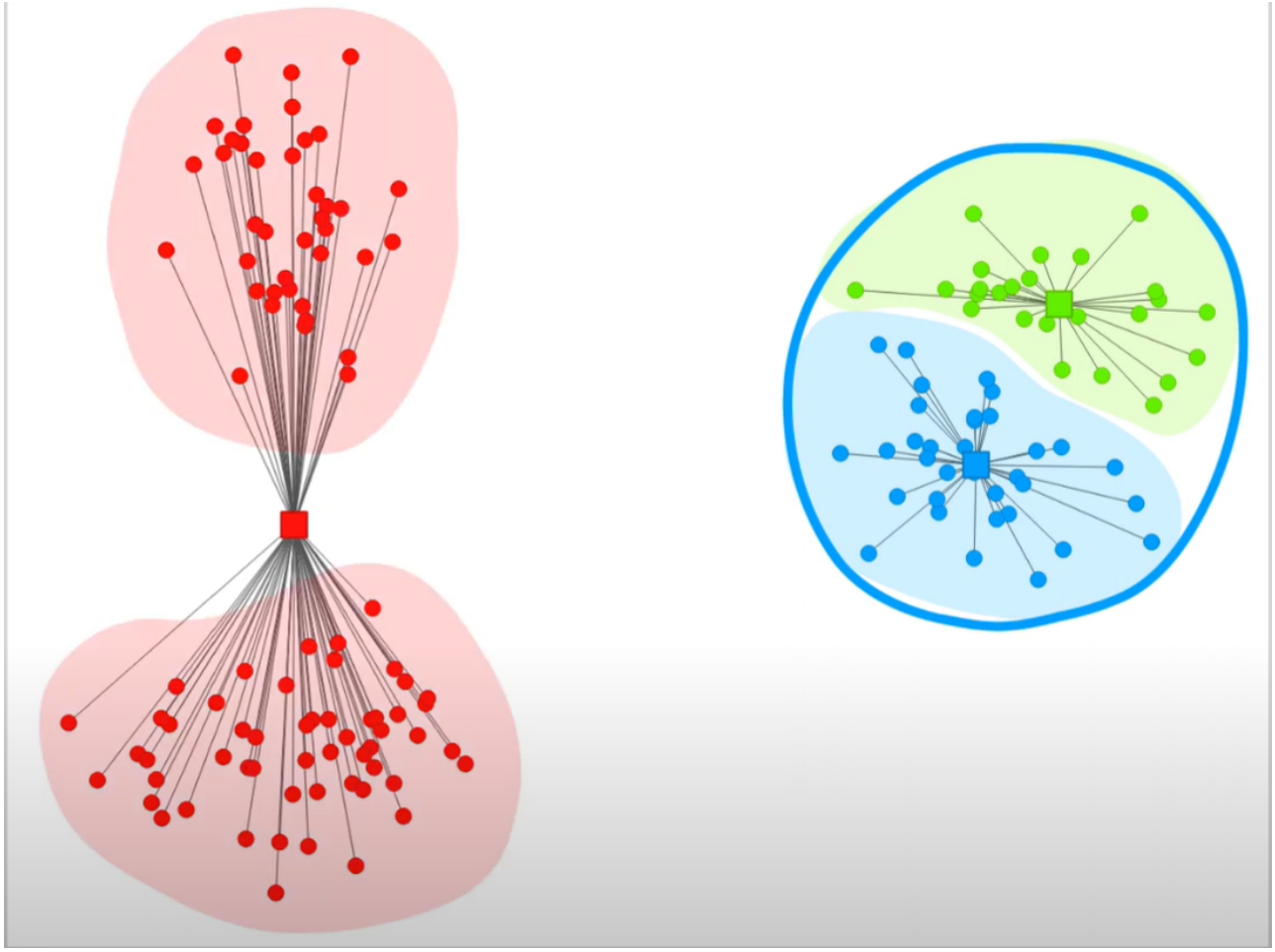
Part 4: Random Initialization and Global Minima

K-Means clustering in sensitive to initial position of centroids.

Challenges with Random Centroid Assignment

- **Non-Convex Optimization Problem**
- Multiple local minima exist

- Final clustering highly dependent on initial centroid positions



In this image we choose **3** centroid clusters on random. Instead of our "expected" out come of three clusters being top-left, bottom left and mid-right we see that:

- top-left and bottom-left clusters and **merged into one**.
- mid-right clusters has been **split into two**.

Fix μ optimize C

- Assign data points to closest cluster center

$$L(C^{t+1}, \mu^t) < L(C^t, \mu^t)$$

Fix C optimize μ

- Change the cluster center to the average of its assigned points

$$L(C^{t+1}, \mu^{t+1}) <= L(C^{t+1}, \mu^t)$$

Loss function is guaranteed to decrease monotonically in each iteration in each steps until convergence.

We will reach a local minima but,
Arriving at global minima through random initialization is not guaranteed, and in most cases, it is highly unlikely.

Mitigation Strategies

1. **Multiple Runs**

- Execute K-Means multiple times
- Select clustering with lowest total within-cluster variance

2. **Advanced Initialization**

- K-Means++: Probabilistic centroid placement
- Improved initial centroid distribution

Section-B (Scratch Implementation)

(A)

a)

```
# Data points
X = np.array([
    [5.1, 3.5], [4.9, 3.0], [5.8, 2.7], [6.0, 3.0], [6.7, 3.1] #.... // more data
points
])
# Initial centroids
initial_centroids = np.array([[3.0, 3.0], [2.0, 2.0]])
```

b)

```
for _ in range(self.max_iters):
            old_centroids = self.centroids.copy()
            # (b) => Assignment step
            """
            Assign each data point to the nearest centroid based
                        on the Euclidean distance
            """
            distances = np.array([[self.euclidean_distance(x, c) for c in
self.centroids] for x in X])
            self.labels = np.argmin(distances, axis=1)
```

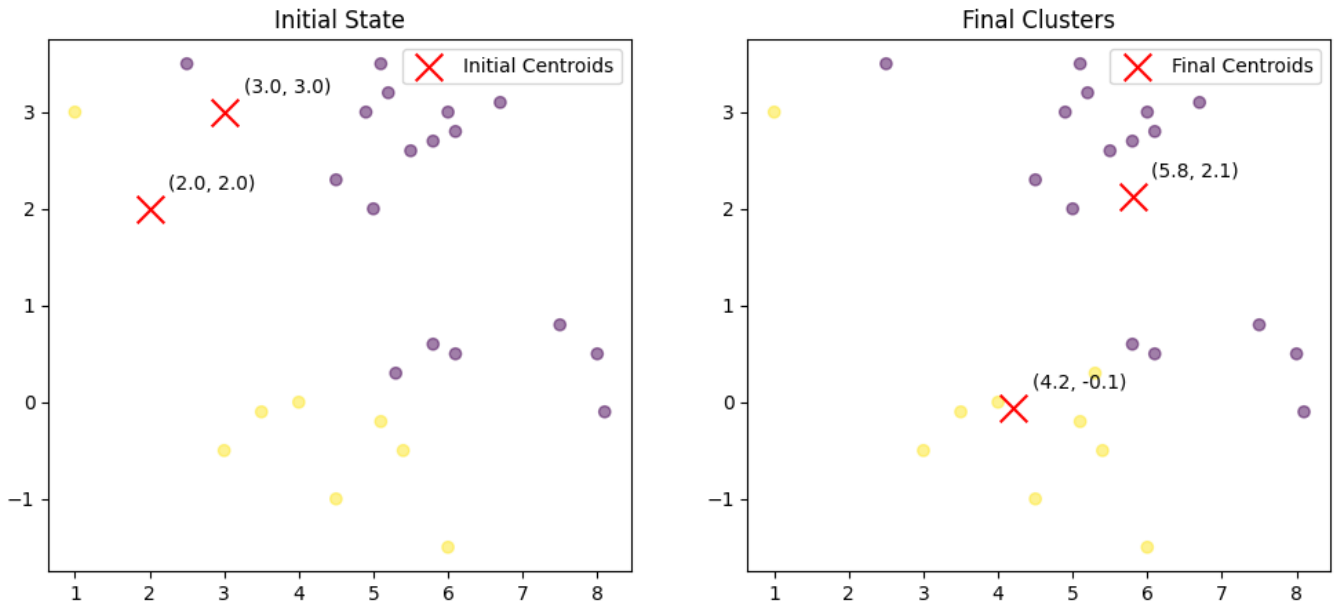
c)

```
            # (c) => Update step
            """
            Recalculate the centroids after each assignment by computing the mean
of all points assigned to each centroid.
            """
            for k in range(self.n_clusters):
                if sum(self.labels == k) > 0:
                    self.centroids[k] = np.mean(X[self.labels == k], axis=0)
```

d)

```
"""
max_iters=100, threshold=1e-4
"""
# (d) => Convergence check
                if np.all(np.abs(old_centroids - self.centroids) <
self.threshold):
                    break
```

(B)

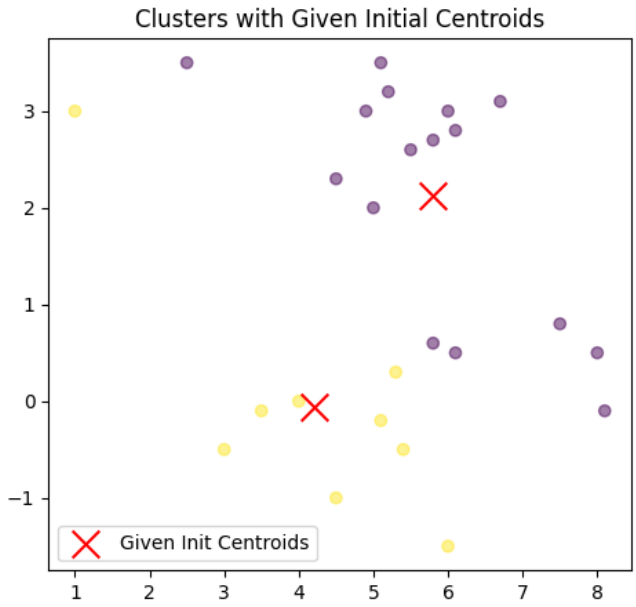


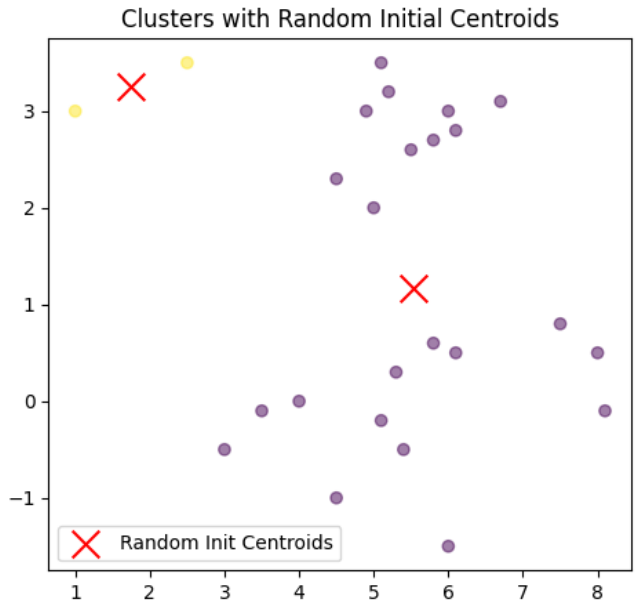
Initial centroids = (2,2) and (3,3)
Final Centroids = (5.8, 2.125) and (4.2, -0.05555556)

(C)

Problem with random initialization :

- **Poor Convergence:** If centroids are initialized far from the true cluster centers or poorly placed, the algorithm may take more iterations to converge.
- **Local Minima** : K-Means minimizes the sum of squared distances between points and their cluster centroids. Poor initialization can lead the algorithm to converge to a local minimum, resulting in a suboptimal clustering solution.
- Splitting of a cluster **OR** Merger of two clusters
- **Instability** : Results may vary significantly across different runs due to the randomness in initialization, making the clustering output unreliable.





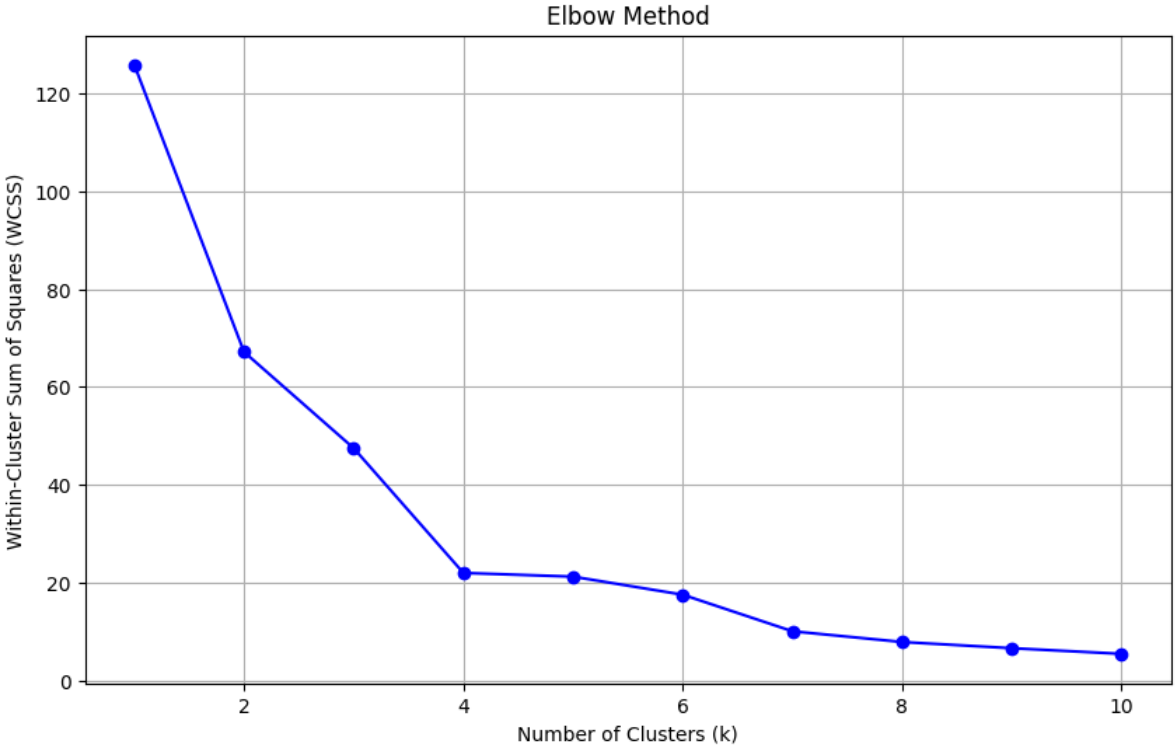
Initialization Method	x-coordinate	y-coordinate
Final Centroids with Initial	5.8	2.125
	4.2	-0.05555556
Final Centroids with Random	5.52608696	1.17391304
	1.75	3.25

We see that the division of points in the two clusters in **not always** optimal or as expected in case of random initialization of centroids

(D)

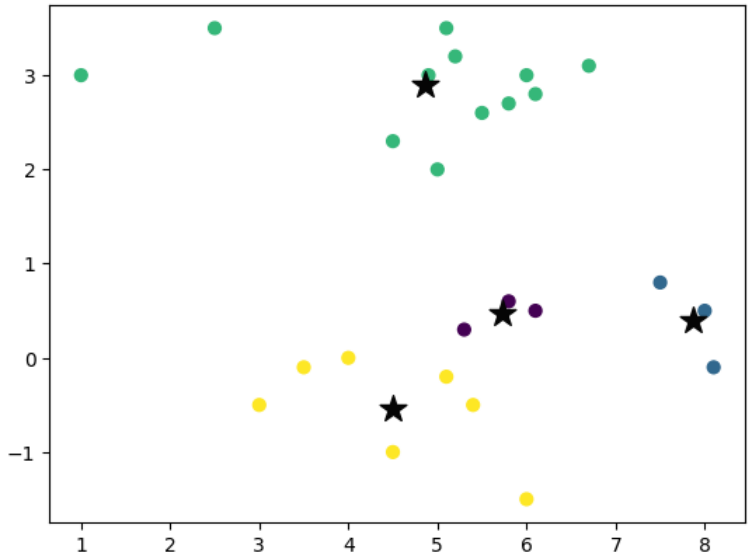
Finding optimal K value using Elbow methods by looking at plot of Within-Cluster Sum of Squares (WCSS) against different values of k to
find the elbow point

Identifying the Elbow Point: As we increase k, the WCSS typically decreases because we're creating more clusters, which tend to capture more data variations. However, there comes a point where adding more clusters results in only a marginal decrease in WCSS. This is where we observe an "elbow" shape in the graph.



We can see that the rate of decrease in distortion or inertia starts to slow down at around 4 clusters. This elbow point often indicates the optimal number of clusters. Therefore, using **4 clusters is a good choice**.

Final Centroids plotted with color coded points on 2-d scatter plot with 4 centroids



Final Centroids:

Index	x-coordinate	y-coordinate
1	5.73333333	0.46666667
2	7.86666667	0.4
3	4.85833333	2.89166667
4	4.5	-0.54285714