MSc Data Science & Information Technologies

Bioinformatics - Biomedical Data Science Specialization

Course: Clustering Algorithms

Professor: Konstantinos Koutroumbas

# Clustering Algorithms 1st Homework


*Name*
Konstantinos Giatras

*Student ID*
7115152300005



2023-2024

# Contents

# Exercise 1

## Problem Statement

Given a set of real numbers $x_1, x_2, \ldots, x_n$ ordered such that $x_1 < x_2 < \ldots < x_n$, we define $A$ as:

$$A = \sum_{i=1}^{n} |x_i - \mu|$$

We aim to show that $A$ is minimized when $\mu$ is the median of $x_1, x_2, \ldots, x_n$, i.e.:

$$\mu = \text{med}(x_1, x_2, \ldots, x_n)$$

## Solution

The **median** is the middle value separating the higher half from the lower half of a data sample. For an odd number of observations ($n$), it is the center value. For an even number of observations, it is the average of the two central values.

### Case for Odd $n$

Assume $n$ is odd, and the median $\mu \equiv x_q$, where $q = \frac{n+1}{2}$. For any $k \neq q$, we compare the sum of absolute deviations from $x_k$ ($\equiv k$) and $x_q$:

$$A_1 = \sum_{i=1}^{n} |x_i - x_q|, \quad A_2 = \sum_{i=1}^{n} |x_i - k|$$

For any $k > x_q$, we will express $A_2$ in terms of $A_1$ and show that $A_2$ is greater than $A_1$, indicating that $A_1$ is indeed the minimum value of $A$. We consider the absolute value of the difference between each $x_i$ and $k$ and express it in terms of $|x_i - x_q|$ and $|x_q - k|$.

Case Analysis ($k > x_q$):

(a) For $x_i < x_q < k$, we have:

$$\begin{aligned} k - x_i &= (k - x_q) + (x_q - x_i), \\ |x_i - k| &= |(x_q - x_i) + (x_q - k)| \end{aligned}$$

Since $x_q - k$ is negative and $x_q - x_i$ is positive (but smaller in magnitude than $x_q - k$), their sum is negative. Thus, we get:

$$\begin{aligned} |x_i - k| &= -(x_q - x_i) - (x_q - k) \\ &= |x_i - x_q| + |x_q - k| \end{aligned}$$

(b) For $k > x_i > x_q$, we have:

$$\begin{aligned} k - x_i &= (k - x_q) - (x_i - x_q), \\ |x_i - k| &= |(x_i - x_q) - (x_q - k)| \end{aligned}$$

As both $x_i - x_q$ and $x_q - k$ are positive, the absolute value of their difference is the larger minus the smaller:

$$|x_i - k| = |x_q - k| - |x_i - x_q|$$

(c) For $x_i > k > x_q$, we have:

$$\begin{aligned} x_i - k &= (x_i - x_q) + (x_q - k), \\ |x_i - k| &= |(x_i - x_q) + (x_q - k)| \end{aligned}$$

Both $x_i - x_q$ and $x_q - k$ are positive, thus their sum is also positive, which implies:

$$|x_i - k| = |x_i - x_q| + |x_q - k|$$

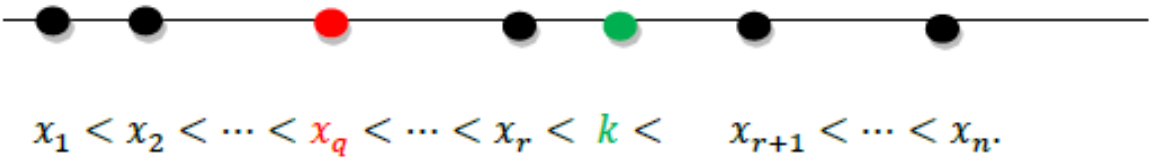
So, for odd $n$ and $k > x_q$, the previous case analysis showed that:

- For $x_i < x_q$, the term $|x_i - k|$ is greater than $|x_i - x_q|$ as $k$ is further from $x_i$ than $x_q$ is.
- For $x_q < x_i < k$, the term $|x_i - k|$ is still greater than $|x_i - x_q|$, since moving from $x_q$ to $k$ increases the distance.
- For $x_i > k$, the term $|x_i - k|$ is again greater than $|x_i - x_q|$ because $k$ is closer to $x_i$ than $x_q$ is, but the difference $|x_i - k| - |x_i - x_q|$ is less than the difference $|x_q - k|$.

For each term $i$, the expression $|x_i - k|$ can be decomposed into the expression $|x_i - x_q| + |x_q - k|$ or $|x_q - k| - |x_i - x_q|$. By summing over all terms $i$, we can then express $A_2$ as:

$$A_2 = A_1 + n \cdot |x_q - k|$$

Given that $|x_q - k|$ is a positive quantity and $n$ is the number of terms in the sum, it is evident that $A_2$ is strictly greater than $A_1$ by the total sum of $n$ times the absolute distance between $x_q$ and $k$. Therefore, we conclude that the sum of absolute deviations $A$ is minimized when the median $x_q$ is chosen as $\mu$, when $k > x_q$.




For any $k < x_q$, we follow the same procedure. We will express $A_2$ in terms of $A_1$ and show that $A_2$ is greater than $A_1$, indicating that $A_1$ is indeed the minimum value of $A$. We consider the absolute value of the difference between each $x_i$ and $k$ and express it in terms of $|x_i - x_q|$ and $|x_q - k|$.

Case Analysis ($k < x_q$):

(a) For $x_i < k < x_q$, we have:

$$x_q - x_i = (x_q - k) + (k - x_i),$$
$$|x_i - k| = |(k - x_i) - (x_q - k)|$$

Since $x_q - k$ is positive and $k - x_i$ is positive, their difference is the larger minus the smaller:

$$|x_i - k| = |x_q - k| - |x_i - x_q|$$

(b) For $k < x_i < x_q$, we have:

$$x_i - k = (x_i - x_q) + (x_q - k),$$
$$|x_i - k| = |(x_i - x_q) + (x_q - k)|$$

Both $x_i - x_q$ and $x_q - k$ are positive, thus their sum is also positive, which implies:

$$|x_i - k| = |x_i - x_q| + |x_q - k|$$

(c) For $x_i > x_q > k$, we have:

$$x_i - k = (x_i - x_q) + (x_q - k),$$
$$|x_i - k| = |(x_i - x_q) + (x_q - k)|$$

Since both $x_i - x_q$ and $x_q - k$ are positive, their sum is also positive, leading to:

$$|x_i - k| = |x_i - x_q| + |x_q - k|$$

So, for odd $n$ and $k < x_q$, the previous case analysis showed that:

- For $x_i < k$, the term $|x_i - k|$ is greater than $|x_i - x_q|$ since $k$ is closer to $x_i$ than $x_q$ is.
- For $k < x_i < x_q$, the term $|x_i - k|$ is still greater than $|x_i - x_q|$, as moving from $k$ to $x_q$ increases the distance.
- For $x_i > x_q$, the term $|x_i - k|$ is again greater than $|x_i - x_q|$ since $k$ is farther from $x_i$ than $x_q$ is.

For each term $i$, the expression $|x_i - k|$ can be decomposed into the expression $|x_i - x_q| + |x_q - k|$ or $|x_q - k| - |x_i - x_q|$. By summing over all terms $i$, we can then express $A_2$ as:

$$A_2 = A_1 + n \cdot |x_q - k|$$

Since $|x_q - k|$ is a positive quantity and $n$ is the number of terms in the sum, it is clear that $A_2$ is strictly greater than $A_1$ by the total sum of $n$ times the absolute distance between $x_q$ and $k$. Thus, we conclude that the sum of absolute deviations $A$ is minimized when the median $x_q$ is chosen as $\mu$, when $k < x_q$.

Therefore, for odd $n$ and any $k \neq x_q$, it is concluded that $A_2$ exceeds $A_1$ by $n \cdot |x_q - k|$, a positive quantity, thus affirming that $A$ is minimized when $\mu$ equals the median $x_q$.

**Case for Even $n$**

For an even $n$, the median $\mu$ is defined as $\mu = \frac{x_q + x_{q+1}}{2}$, where $q = \frac{n}{2}$. Unlike the odd $n$ case, the median is not the only minimizer, since all points within the range $[x_q, x_{q+1}]$ minimize $A$.

We analyze the sum of absolute deviations when $k > x_{q+1}$.

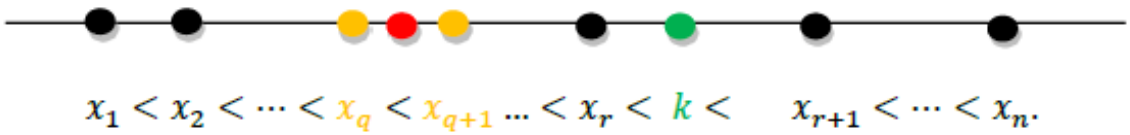
Case Analysis ($k > x_{q+1}$):

(a) For $x_i < x_q$, $|x_i - k|$ is greater than both $|x_i - x_q|$ and $|x_i - x_{q+1}|$ since $k$ is further from $x_i$ than both $x_q$ and $x_{q+1}$.

(b) For $x_q \leq x_i \leq x_{q+1}$, $|x_i - k|$ is simply $|x_{q+1} - k|$, which is a positive quantity since $k > x_{q+1}$.

(c) For $x_i > x_{q+1}$, $|x_i - k|$ equals $x_i - k$ and is smaller than the case when $x_i$ is between $x_q$ and $x_{q+1}$, but $|x_i - k|$ is still greater than $|x_i - x_{q+1}|$.

Thus, for any $k > x_{q+1}$, the sum of absolute deviations from $k$, denoted as $A_2$, is greater than the sum of absolute deviations from any point within the median range $[x_q, x_{q+1}]$, denoted as $A_1$. Therefore, for an even $n$, and any $k > x_{q+1}$, $A_2$ exceeds $A_1$ by at least $n \cdot |x_{q+1} - k|$, affirming that $A$ is minimized when $\mu$ falls within the median range $[x_q, x_{q+1}]$.




We also analyze the sum of absolute deviations when $k < x_q$.

Case Analysis ($k < x_q$):

(a) For $x_i < x_q$ and $k < x_q$, $|x_i - k|$ is simply $|x_q - k|$, which is a positive quantity since $k < x_q$.

(b) For $x_q \leq x_i \leq x_{q+1}$, $|x_i - k|$ is greater than both $|x_i - x_q|$ and $|x_i - x_{q+1}|$ as $k$ is further from $x_i$ than both $x_q$ and $x_{q+1}$.

(c) For $x_i > x_{q+1}$, $|x_i - k|$ is greater than $|x_i - x_{q+1}|$ since $k < x_q$ and $x_i > x_{q+1}$, making $k$ farther from $x_i$ than $x_{q+1}$.

Thus, for any $k < x_q$, the sum of absolute deviations from $k$, denoted as $A_2$, is greater than the sum of absolute deviations from any point within the median range $[x_q, x_{q+1}]$, denoted as $A_1$. Consequently, for an even $n$, and any $k < x_q$, $A_2$ exceeds $A_1$ by at least $n \cdot |x_q - k|$, verifying that $A$ is minimized when $\mu$ is within the median range $[x_q, x_{q+1}]$.

We can conclude that, for an even $n$ and any $k \notin [x_q, x_{q+1}]$, the sum of absolute deviations $A$ from any number outside the median range $[x_q, x_{q+1}]$ is greater than the sum from any point within this range, affirming that $A$ is minimized when $\mu$ falls within $[x_q, x_{q+1}]$.

## Conclusion

Our analysis has shown that for a given set of real numbers $x_1, x_2, \ldots, x_n$ ordered in ascending order, the sum of absolute deviations from the median, $A$, is minimized. This result holds true for both odd and even numbers of observations.

For odd $n$, the median is the central value, $x_q$, and we demonstrated that any deviation from this median increases $A$. For even $n$, the median is the average of the two central values, $x_q$ and $x_{q+1}$, and we showed that any point within this median range minimizes $A$, with deviations outside this range resulting in a larger sum.

Therefore, the median, whether as a single value for odd $n$ or a range for even $n$, uniquely minimizes the sum of absolute deviations in a set of ordered real numbers. This underscores the robustness and significance of the median as a measure of central tendency in statistical analysis.

# Exercise 2

## Hard k-medians Clustering Algorithm

The hard k-medians algorithm is a part of the Generalized Hard Algorithmic Scheme (GHAS) for clustering. This scheme is characterized by a hard assignment of data points to clusters, with the objective of minimizing a defined cost function.

In hard k-medians clustering, each cluster is represented by a point representative, commonly denoted as $\boldsymbol{\theta}_j$. This representative is a central point of the cluster, which ideally minimizes the sum of distances to all the points assigned to the cluster. In the context of k-medians, the point representative is typically the median of the points in the cluster.

The cost function $J(U, \Theta)$ for the hard k-medians clustering is given by:

$$J(U, \Theta) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} \|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1,$$

where $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{il}]^T \in \mathbb{R}^l, i = 1, \ldots, N$, are the $N$ $l$-dimensional data vectors, $\boldsymbol{\theta}_j = [\theta_{j1}, \ldots, \theta_{jl}]^T, j = 1, \ldots, m$ are the $m$ $l$-dimensional point representatives of the clusters, $u_{ij}$ are the binary membership indicators, and $\|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1$ denotes the L1 norm (Manhattan distance):

$$\|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1 = \sum_{r=1}^{l} |x_{ir} - \theta_{jr}|$$

### Algorithm Pseudocode

**Algorithm 1** Hard k-medians Clustering Algorithm as part of the Generalized Hard Algorithmic Scheme (GHAS)

Initialize point representatives $\boldsymbol{\theta}_j^{(0)}$ for $j = 1, \ldots, m$
Set iteration counter $t = 0$
**repeat**
  **for** $i = 1$ to $N$ **do** ▷ Determination of the partition, estimate $u_{ij}$'s using Manhattan distance
    **for** $j = 1$ to $m$ **do**
      $$u_{ij}^{(t)} \leftarrow \begin{cases} 1, & \text{if } \sum_{r=1}^{l} |x_{ir} - \theta_{jr}^{(t)}| = \min\limits_{k=1,\ldots,m} \sum_{r=1}^{l} |x_{ir} - \theta_{kr}^{(t)}| \\ 0, & \text{otherwise} \end{cases}$$
    **end for**
  **end for**
  $t \leftarrow t + 1$
  **for** $j = 1$ to $m$ **do** ▷ Parameter updating, estimate $\boldsymbol{\theta}_j$'s using Manhattan distance
    $\boldsymbol{\theta}_j^{(t)} \leftarrow \arg\min_{\boldsymbol{\theta}_j} \sum_{i=1}^{N} u_{ij}^{(t-1)} \sum_{r=1}^{l} |x_{ir} - \theta_{jr}|, j = 1, \ldots, m$
  **end for**
**until** a termination criterion is met

The hard k-medians clustering algorithm, based on the Generalized Hard Algorithmic Scheme (GHAS), iteratively assigns data points to clusters with the goal of minimizing the sum of Manhattan distances from each point to the central point representative of its cluster. Each cluster's representative is updated to be the median of the assigned points. This process repeats, alternating between assigning points based on the least distance to the median of each cluster and updating the cluster medians based on current assignments, until the cluster representatives stabilize, indicating that the arrangement of points into clusters no longer significantly changes, fulfilling the termination criterion of the algorithm.

## Possibilistic k-medians Clustering Algorithm

The possibilistic k-medians algorithm adapts the k-medians framework for use in situations where each data point can belong to multiple clusters with different degrees of membership. This method is part of the Generalized Possibilistic Algorithmic Scheme (GPAS), which revises the clustering process to consider the potential membership of each point in all clusters independently.

In this clustering approach, the point representatives $\boldsymbol{\theta}_j$ are the central points of the clusters, aiming to minimize a cost function that differs from the hard k-medians. The cost function $J(U, \Theta)$ for possibilistic k-medians is:

$$J(U, \Theta) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^{q} \|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1 + \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q,$$

where $q > 1$ is a parameter that influences the fuzziness of the cluster boundaries. The term $u_{ij}^{q} \|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1$ reflects the weighted distance from data points to cluster centers, with $u_{ij}$ being the membership degree of $\boldsymbol{x}_i$ in cluster $C_j$, and the term $\eta_j \sum_{i=1}^{N} (1 - u_{ij})^q$ acting as a regularization to prevent trivial solutions, promoting significant membership of each point in at least one cluster.

The degree of membership $u_{ij}$ is derived from the distances to the point representatives and parameter $\eta_j$, which governs the cluster spread. The Manhattan distance $\|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|_1$ is the sum of absolute differences between the components of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_j$.

### Algorithm Pseudocode

**Algorithm 2** Possibilistic k-medians Clustering Algorithm as part of the Generalized Possibilistic Algorithmic Scheme (GPAS1)

1: Fix $\eta_j$'s, $j = 1, \ldots, m$
2: Choose $\boldsymbol{\theta}_j^{(0)}$ as initial estimates for $\boldsymbol{\theta}_j$, $j = 1, \ldots, m$
3: Set iteration counter $t = 0$
4: **repeat**
5: **for** $i = 1$ to $N$ **do** ▷ Estimation of the $u_{ij}$'s using Manhattan distance
6: **for** $j = 1$ to $m$ **do**
7: $u_{ij}^{(t)} \leftarrow \dfrac{1}{1 + \left( \dfrac{\sum_{r=1}^{l} |x_{ir} - \theta_{jr}^{(t)}|}{\eta_j} \right)^{\frac{1}{q-1}}}$
8: **end for**
9: **end for**
10: $t \leftarrow t + 1$
11: **for** $j = 1$ to $m$ **do** ▷ Parameter updating, estimate $\boldsymbol{\theta}_j$'s using Manhattan distance
12: $\boldsymbol{\theta}_j^{(t)} \leftarrow \arg\min_{\boldsymbol{\theta}_j} \sum_{i=1}^{N} u_{ij}^{q(t-1)} \sum_{r=1}^{l} |x_{ir} - \theta_{jr}|, j = 1, \ldots, m$
13: **end for**
14: **until** a termination criterion is met

The possibilistic k-medians algorithm seeks to optimize its cost function $J(U, \Theta)$ through an iterative process that starts with initial estimates of cluster centers and alternates between updating membership degrees $u_{ij}$ for each data point based on these centers and a cluster extent parameter, and adjusting the cluster centers $\boldsymbol{\theta}_j$ considering the new membership degrees to minimize the cost. The algorithm refines membership degrees and cluster centers in a loop, proceeding until the cluster centers stabilize, as indicated by changes falling below a threshold or after a predetermined number of iterations. This flexibility in membership assignment, allowing points to belong to multiple clusters to varying degrees, renders the algorithm particularly adept at handling datasets with overlapping clusters and non-uniform densities, a significant advantage over hard clustering methods.

# Exercise 3

Consider the data set $Y = \{x_1, x_2, x_3, x_4, x_5\}$, where $x_1 = [0,0]^T$, $x_2 = [0,3]^T$, $x_3 = [6,0]^T$, $x_4 = [7,0]^T$, $x_5 = [7,-3]^T$. We will run the hard k-means and the hard k-medians algorithms using this data set for the following cases:

(a) For two representatives $\theta_1$ and $\theta_2$, whose initial positions are $\theta_1(0) = [6,1]^T$ and $\theta_2(0) = [8,0]^T$ respectively.

(b) For two representatives $\theta_1$ and $\theta_2$, whose initial positions are $\theta_1(0) = [6,1]^T$ and $\theta_2(0) = [20,0]^T$ respectively.

(c) For three representatives.

At each iteration, we will calculate the (i) $U = [u_{ij}]$ matrix, (ii) $\theta_j$'s and (iii) the formed clusters.

## Hard k-means Clustering Algorithm

**Case (a)**

Iteration 1:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their initialized positions, using the squared Euclidean distance:

$$
\begin{aligned}
d(x_1, \theta_1^{(0)}) &= \|x_1 - \theta_1^{(0)}\|^2 = (0-6)^2 + (0-1)^2 = 36 + 1 = 37 \\
d(x_2, \theta_1^{(0)}) &= \|x_2 - \theta_1^{(0)}\|^2 = (0-6)^2 + (3-1)^2 = 36 + 4 = 40 \\
d(x_3, \theta_1^{(0)}) &= \|x_3 - \theta_1^{(0)}\|^2 = (6-6)^2 + (0-1)^2 = 0 + 1 = 1 \\
d(x_4, \theta_1^{(0)}) &= \|x_4 - \theta_1^{(0)}\|^2 = (7-6)^2 + (0-1)^2 = 1 + 1 = 2 \\
d(x_5, \theta_1^{(0)}) &= \|x_5 - \theta_1^{(0)}\|^2 = (7-6)^2 + (-3-1)^2 = 1 + 16 = 17 \\
d(x_1, \theta_2^{(0)}) &= \|x_1 - \theta_2^{(0)}\|^2 = (0-8)^2 + (0-0)^2 = 64 + 0 = 64 \\
d(x_2, \theta_2^{(0)}) &= \|x_2 - \theta_2^{(0)}\|^2 = (0-8)^2 + (3-0)^2 = 64 + 9 = 73 \\
d(x_3, \theta_2^{(0)}) &= \|x_3 - \theta_2^{(0)}\|^2 = (6-8)^2 + (0-0)^2 = 4 + 0 = 4 \\
d(x_4, \theta_2^{(0)}) &= \|x_4 - \theta_2^{(0)}\|^2 = (7-8)^2 + (0-0)^2 = 1 + 0 = 1 \\
d(x_5, \theta_2^{(0)}) &= \|x_5 - \theta_2^{(0)}\|^2 = (7-8)^2 + (-3-0)^2 = 1 + 9 = 10
\end{aligned}
$$

We assign each data point to the nearest representative. The membership matrix $U$ is:

$$
\begin{aligned}
d(x_1, \theta_1^{(0)}) < d(x_1, \theta_2^{(0)}) &\Rightarrow u_{11}^{(0)} = 1, u_{12}^{(0)} = 0 \\
d(x_2, \theta_1^{(0)}) < d(x_2, \theta_2^{(0)}) &\Rightarrow u_{21}^{(0)} = 1, u_{22}^{(0)} = 0 \\
d(x_3, \theta_1^{(0)}) < d(x_3, \theta_2^{(0)}) &\Rightarrow u_{31}^{(0)} = 1, u_{32}^{(0)} = 0 \\
d(x_4, \theta_1^{(0)}) > d(x_4, \theta_2^{(0)}) &\Rightarrow u_{41}^{(0)} = 0, u_{42}^{(0)} = 1 \\
d(x_5, \theta_1^{(0)}) > d(x_5, \theta_2^{(0)}) &\Rightarrow u_{51}^{(0)} = 0, u_{52}^{(0)} = 1
\end{aligned}
$$

$$
U^{(0)} = \begin{bmatrix} u_{11}^{(0)} & u_{12}^{(0)} \\ u_{21}^{(0)} & u_{22}^{(0)} \\ u_{31}^{(0)} & u_{32}^{(0)} \\ u_{41}^{(0)} & u_{42}^{(0)} \\ u_{51}^{(0)} & u_{52}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}
$$

This indicates that $x_1, x_2$, and $x_3$ are closer to $\theta_1$, while $x_4$ and $x_5$ are closer to $\theta_2$.

*Update Step*:

We calculate the new positions of the representatives based on the mean of the assigned points:

$$\boldsymbol{\theta}_1^{(1)} = \frac{u_{11}^{(0)}\boldsymbol{x}_1 + u_{21}^{(0)}\boldsymbol{x}_2 + u_{31}^{(0)}\boldsymbol{x}_3 + u_{41}^{(0)}\boldsymbol{x}_4 + u_{51}^{(0)}\boldsymbol{x}_5}{u_{11}^{(0)} + u_{21}^{(0)} + u_{31}^{(0)} + u_{41}^{(0)} + u_{51}^{(0)}} = \frac{[0,0]^T + [0,3]^T + [6,0]^T}{3} = [2,1]^T$$

$$\boldsymbol{\theta}_2^{(1)} = \frac{u_{12}^{(0)}\boldsymbol{x}_1 + u_{22}^{(0)}\boldsymbol{x}_2 + u_{32}^{(0)}\boldsymbol{x}_3 + u_{42}^{(0)}\boldsymbol{x}_4 + u_{52}^{(0)}\boldsymbol{x}_5}{u_{12}^{(0)} + u_{22}^{(0)} + u_{32}^{(0)} + u_{42}^{(0)} + u_{52}^{(0)}} = \frac{[7,0]^T + [7,-3]^T}{2} = [7,-1.5]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}, C_2 : \{\boldsymbol{x}_4, \boldsymbol{x}_5\}$

Iteration 2:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their new positions, using the squared Euclidean distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(1)}\|^2 = (0-2)^2 + (0-1)^2 = 4 + 1 = 5$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(1)}\|^2 = (0-2)^2 + (3-1)^2 = 4 + 4 = 8$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(1)}\|^2 = (6-2)^2 + (0-1)^2 = 16 + 1 = 17$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(1)}\|^2 = (7-2)^2 + (0-1)^2 = 25 + 1 = 26$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(1)}\|^2 = (7-2)^2 + (-3-1)^2 = 25 + 16 = 41$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(1)}\|^2 = (0-7)^2 + (0-(-1.5))^2 = 49 + 2.25 = 51.25$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(1)}\|^2 = (0-7)^2 + (3-(-1.5))^2 = 49 + 20.25 = 69.25$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(1)}\|^2 = (6-7)^2 + (0-(-1.5))^2 = 1 + 2.25 = 3.25$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(1)}\|^2 = (7-7)^2 + (0-(-1.5))^2 = 0 + 2.25 = 2.25$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(1)}\|^2 = (7-7)^2 + (-3-(-1.5))^2 = 0 + 20.25 = 20.25$$

We assign each data point to the nearest representative. The new membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(1)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{11}^{(1)} = 1, u_{12}^{(1)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(1)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{21}^{(1)} = 1, u_{22}^{(1)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{31}^{(1)} = 0, u_{32}^{(1)} = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{41}^{(1)} = 0, u_{42}^{(1)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{51}^{(1)} = 0, u_{52}^{(1)} = 1$$

$$U^{(1)} = \begin{bmatrix} u_{11}^{(1)} & u_{12}^{(1)} \\ u_{21}^{(1)} & u_{22}^{(1)} \\ u_{31}^{(1)} & u_{32}^{(1)} \\ u_{41}^{(1)} & u_{42}^{(1)} \\ u_{51}^{(1)} & u_{52}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

This indicates a change in the cluster assignment for $\boldsymbol{x}_3$, which is now closer to $\boldsymbol{\theta}_2$.

*Update Step*:

We calculate the new positions of the representatives based on the mean of the assigned points:

$$\boldsymbol{\theta}_1^{(2)} = \frac{u_{11}^{(1)}\boldsymbol{x}_1 + u_{21}^{(1)}\boldsymbol{x}_2 + u_{31}^{(1)}\boldsymbol{x}_3 + u_{41}^{(1)}\boldsymbol{x}_4 + u_{51}^{(1)}\boldsymbol{x}_5}{u_{11}^{(1)} + u_{21}^{(1)} + u_{31}^{(1)} + u_{41}^{(1)} + u_{51}^{(1)}} = \frac{[0,0]^T + [0,3]^T}{2} = [0, 1.5]^T$$

$$\boldsymbol{\theta}_2^{(2)} = \frac{u_{12}^{(1)}\boldsymbol{x}_1 + u_{22}^{(1)}\boldsymbol{x}_2 + u_{32}^{(1)}\boldsymbol{x}_3 + u_{42}^{(1)}\boldsymbol{x}_4 + u_{52}^{(1)}\boldsymbol{x}_5}{u_{12}^{(1)} + u_{22}^{(1)} + u_{32}^{(1)} + u_{42}^{(1)} + u_{52}^{(1)}} = \frac{[6,0]^T + [7,0]^T + [7,-3]^T}{3} \simeq [6.66, -1]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, C_2 : \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$

Iteration 3:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their new positions, using the squared Euclidean distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(2)}\|^2 = (0-0)^2 + (0-1.5)^2 = 0 + 2.25 = 2.25$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(2)}\|^2 = (0-0)^2 + (3-1.5)^2 = 0 + 2.25 = 2.25$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(2)}\|^2 = (6-0)^2 + (0-1.5)^2 = 36 + 2.25 = 38.25$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(2)}\|^2 = (7-0)^2 + (0-1.5)^2 = 49 + 2.25 = 51.25$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(2)}\|^2 = (7-0)^2 + (-3-1.5)^2 = 49 + 20.25 = 69.25$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(2)}\|^2 = (0-6.66)^2 + (0-(-1))^2 = 44.36 + 1 = 45.36$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(2)}\|^2 = (0-6.66)^2 + (3-(-1))^2 = 44.36 + 16 = 60.36$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(2)}\|^2 = (6-6.66)^2 + (0-(-1))^2 = 0.44 + 1 = 1.44$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(2)}\|^2 = (7-6.66)^2 + (0-(-1))^2 = 0.12 + 1 = 1.12$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(2)}\|^2 = (7-6.66)^2 + (-3-(-1))^2 = 0.12 + 4 = 4.12$$

We assign each data point to the nearest representative. The new membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(2)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{11}^{(2)} = 1, u_{12}^{(2)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(2)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{21}^{(2)} = 1, u_{22}^{(2)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{31}^{(2)} = 0, u_{32}^{(2)} = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{41}^{(2)} = 0, u_{42}^{(2)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{51}^{(2)} = 0, u_{52}^{(2)} = 1$$

$$U^{(2)} = \begin{bmatrix} u_{11}^{(2)} & u_{12}^{(2)} \\ u_{21}^{(2)} & u_{22}^{(2)} \\ u_{31}^{(2)} & u_{32}^{(2)} \\ u_{41}^{(2)} & u_{42}^{(2)} \\ u_{51}^{(2)} & u_{52}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

This indicates that there is no change in the cluster assignment for any of the data points.

*Update Step*:

We calculate the new positions of the representatives based on the mean of the assigned points:

$$\boldsymbol{\theta}_1^{(3)} = \frac{u_{11}^{(2)}\boldsymbol{x}_1 + u_{21}^{(2)}\boldsymbol{x}_2 + u_{31}^{(2)}\boldsymbol{x}_3 + u_{41}^{(2)}\boldsymbol{x}_4 + u_{51}^{(2)}\boldsymbol{x}_5}{u_{11}^{(2)} + u_{21}^{(2)} + u_{31}^{(2)} + u_{41}^{(2)} + u_{51}^{(2)}} = \frac{[0,0]^T + [0,3]^T}{2} = [0,1.5]^T$$

$$\boldsymbol{\theta}_2^{(3)} = \frac{u_{12}^{(2)}\boldsymbol{x}_1 + u_{22}^{(2)}\boldsymbol{x}_2 + u_{32}^{(2)}\boldsymbol{x}_3 + u_{42}^{(2)}\boldsymbol{x}_4 + u_{52}^{(2)}\boldsymbol{x}_5}{u_{12}^{(2)} + u_{22}^{(2)} + u_{32}^{(2)} + u_{42}^{(2)} + u_{52}^{(2)}} = \frac{[6,0]^T + [7,0]^T + [7,-3]^T}{3} \simeq [6.66,-1]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, C_2 : \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$

Since there is no change in the clusters between iterations 2 and 3, the algorithm has converged (the cost function has reached a local minimum), and these are the final clusters.

**Case (b)**

Iteration 1:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their initialized positions, using the squared Euclidean distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(0)}\|^2 = (0-6)^2 + (0-1)^2 = 36 + 1 = 37$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(0)}\|^2 = (0-6)^2 + (3-1)^2 = 36 + 4 = 40$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(0)}\|^2 = (6-6)^2 + (0-1)^2 = 0 + 1 = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(0)}\|^2 = (7-6)^2 + (0-1)^2 = 1 + 1 = 2$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(0)}\|^2 = (7-6)^2 + (-3-1)^2 = 1 + 16 = 17$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(0)}\|^2 = (0-20)^2 + (0-0)^2 = 400 + 0 = 400$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(0)}\|^2 = (0-20)^2 + (3-0)^2 = 400 + 9 = 409$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(0)}\|^2 = (6-20)^2 + (0-0)^2 = 196 + 0 = 196$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(0)}\|^2 = (7-20)^2 + (0-0)^2 = 169 + 0 = 169$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(0)}\|^2 = (7-20)^2 + (-3-0)^2 = 169 + 9 = 178$$

We assign each data point to the nearest representative. The membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{11}^{(0)} = 1, u_{12}^{(0)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{21}^{(0)} = 1, u_{22}^{(0)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{31}^{(0)} = 1, u_{32}^{(0)} = 0$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{41}^{(0)} = 0, u_{42}^{(0)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{51}^{(0)} = 0, u_{52}^{(0)} = 1$$

$$U^{(0)} = \begin{bmatrix} u_{11}^{(0)} & u_{12}^{(0)} \\ u_{21}^{(0)} & u_{22}^{(0)} \\ u_{31}^{(0)} & u_{32}^{(0)} \\ u_{41}^{(0)} & u_{42}^{(0)} \\ u_{51}^{(0)} & u_{52}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

This indicates that all data points are closer to $\boldsymbol{\theta}_1$ than to $\boldsymbol{\theta}_2$.

*Update Step*:

We calculate the new positions of the representatives based on the mean of the assigned points:

$$\boldsymbol{\theta}_1^{(1)} = \frac{u_{11}^{(0)}\boldsymbol{x}_1 + u_{21}^{(0)}\boldsymbol{x}_2 + u_{31}^{(0)}\boldsymbol{x}_3 + u_{41}^{(0)}\boldsymbol{x}_4 + u_{51}^{(0)}\boldsymbol{x}_5}{u_{11}^{(0)} + u_{21}^{(0)} + u_{31}^{(0)} + u_{41}^{(0)} + u_{51}^{(0)}} = \frac{[0,0]^T + [0,3]^T + [6,0]^T + [7,0]^T + [7,-3]^T}{5} = [4,0]^T$$

$$\boldsymbol{\theta}_2^{(1)} = \boldsymbol{\theta}_2^{(0)} = [20,0]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}, C_2 : \emptyset$

Due to the bad initialization of the position of representative $\boldsymbol{\theta}_2$, cluster $C_1$ ended up containing all the points in the data (with representative $\boldsymbol{\theta}_1$ being the center of the data set) and cluster $C_2$ ending up empty (hence not being a cluster at all), after a single iteration of the algorithm. Thus, the algorithm has reaches a termination criterion.

Some ways of dealing with the bad initialization problem could be reinitializing the empty cluster (by randomly selecting a new data point or assigning to it the furthest point from all other representatives), or redistributing the data points based on the remaining non-empty clusters, before continuing the algorithm.

**Case (c)**

When running the hard k-means clustering algorithm on the given dataset with three representatives employed, we will ideally obtain 3 clusters. This is because the number of clusters in k-means is determined by the number of representatives (or centroids) we choose to initialize.

However, if the initialization of the positions of these representatives is not done properly (like in case b), some clusters may end up being empty. As a result, the actual number of clusters obtained might be less than 3, depending on how the empty clusters are handled (whether they are reinitialized or the algorithm stops). If we ensure that the initialization of the representatives is well done, we are likely to obtain 3 distinct clusters. Good initialization is key to achieving the maximum potential number of clusters and avoiding issues like empty clusters.

It's also important to note that even if the dataset contains more than 3 physical clusters, using this method with only three representatives means we cannot obtain more than 3 clusters. The k-means algorithm with a fixed number of representatives (in this case, three) will always partition the data into that specified number of clusters, which might lead to an oversimplified representation if the true structure of the data is more complex.

## Hard k-medians Clustering Algorithm

**Case (a)**

Iteration 1:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their initialized positions, using the Manhattan distance:

$$
\begin{aligned}
d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) &= \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(0)}\| = |0-6| + |0-1| = 6+1 = 7 \\
d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) &= \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(0)}\| = |0-6| + |3-1| = 6+2 = 8 \\
d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) &= \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(0)}\| = |6-6| + |0-1| = 0+1 = 1 \\
d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) &= \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(0)}\| = |7-6| + |0-1| = 1+1 = 2 \\
d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) &= \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(0)}\| = |7-6| + |-3-1| = 1+4 = 5 \\
d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) &= \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(0)}\| = |0-8| + |0-0| = 8+0 = 8 \\
d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) &= \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(0)}\| = |0-8| + |3-0| = 8+3 = 11 \\
d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) &= \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(0)}\| = |6-8| + |0-0| = 2+0 = 2 \\
d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) &= \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(0)}\| = |7-8| + |0-0| = 1+0 = 1 \\
d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) &= \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(0)}\| = |7-8| + |-3-0| = 1+3 = 4
\end{aligned}
$$

We assign each data point to the nearest representative. The membership matrix $U$ is:

$$
\begin{aligned}
d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) &\Rightarrow u_{11}^{(0)} = 1, u_{12}^{(0)} = 0 \\
d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) &\Rightarrow u_{21}^{(0)} = 1, u_{22}^{(0)} = 0 \\
d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) &\Rightarrow u_{31}^{(0)} = 1, u_{32}^{(0)} = 0 \\
d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) > d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) &\Rightarrow u_{41}^{(0)} = 0, u_{42}^{(0)} = 1 \\
d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) > d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) &\Rightarrow u_{51}^{(0)} = 0, u_{52}^{(0)} = 1
\end{aligned}
$$

$$
U^{(0)} = \begin{bmatrix} u_{11}^{(0)} & u_{12}^{(0)} \\ u_{21}^{(0)} & u_{22}^{(0)} \\ u_{31}^{(0)} & u_{32}^{(0)} \\ u_{41}^{(0)} & u_{42}^{(0)} \\ u_{51}^{(0)} & u_{52}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}
$$

This indicates that $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}_3$ are closer to $\boldsymbol{\theta}_1$, while $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ are closer to $\boldsymbol{\theta}_2$.

*Update Step*:

We calculate the new positions of the representatives based on the median of the points assigned to that cluster, separately for each dimension:

$$
\begin{aligned}
\boldsymbol{\theta}_1^{(1)} &= \text{med}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \text{med}([0,0]^T, [0,3]^T, [6,0]^T) = [0,0]^T \\
\boldsymbol{\theta}_2^{(1)} &= \text{med}(\boldsymbol{x}_4, \boldsymbol{x}_5) = \text{med}([7,0]^T, [7,-3]^T) = [7,-1.5]^T
\end{aligned}
$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}, C_2 : \{\boldsymbol{x}_4, \boldsymbol{x}_5\}$

Iteration 2:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their new positions, using the Manhattan distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(1)}\| = |0-0| + |0-0| = 0+0 = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(1)}\| = |0-0| + |3-0| = 0+3 = 3$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(1)}\| = |6-0| + |0-0| = 6+0 = 6$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(1)}\| = |7-0| + |0-0| = 7+0 = 7$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(1)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(1)}\| = |7-0| + |-3-0| = 7+3 = 10$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(1)}\| = |0-7| + |0-(-1.5)| = 7+1.5 = 8.5$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(1)}\| = |0-7| + |3-(-1.5)| = 7+4.5 = 11.5$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(1)}\| = |6-7| + |0-(-1.5)| = 1+1.5 = 2.5$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(1)}\| = |7-7| + |0-(-1.5)| = 0+1.5 = 1.5$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(1)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(1)}\| = |7-7| + |-3-(-1.5)| = 0+1.5 = 1.5$$

We assign each data point to the nearest representative. The new membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(1)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{11}^{(1)} = 1, u_{12}^{(1)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(1)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{21}^{(1)} = 1, u_{22}^{(1)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{31}^{(1)} = 0, u_{32}^{(1)} = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{41}^{(1)} = 0, u_{42}^{(1)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(1)}) > d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(1)}) \Rightarrow u_{51}^{(1)} = 0, u_{52}^{(1)} = 1$$

$$U^{(1)} = \begin{bmatrix} u_{11}^{(1)} & u_{12}^{(1)} \\ u_{21}^{(1)} & u_{22}^{(1)} \\ u_{31}^{(1)} & u_{32}^{(1)} \\ u_{41}^{(1)} & u_{42}^{(1)} \\ u_{51}^{(1)} & u_{52}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

This indicates a change in the cluster assignment for $\boldsymbol{x}_3$, which is now closer to $\boldsymbol{\theta}_2$.

*Update Step*:

We calculate the new positions of the representatives based on the median of the points assigned to that cluster, separately for each dimension:

$$\boldsymbol{\theta}_1^{(2)} = \text{med}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \text{med}([0,0]^T, [0,3]^T) = [0, 1.5]^T$$
$$\boldsymbol{\theta}_2^{(2)} = \text{med}(\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5) = \text{med}([6,0]^T, [7,0]^T, [7,-3]^T) = [7,0]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, C_2 : \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$

Iteration 3:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their new positions, using the Manhattan distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(2)}\| = |0-0| + |0-1.5| = 0 + 1.5 = 1.5$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(2)}\| = |0-0| + |3-1.5| = 0 + 1.5 = 1.5$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(2)}\| = |6-0| + |0-1.5| = 6 + 1.5 = 7.5$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(2)}\| = |7-0| + |0-1.5| = 7 + 1.5 = 8.5$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(2)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(2)}\| = |7-0| + |-3-1.5| = 7 + 4.5 = 11.5$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(2)}\| = |0-7| + |0-0| = 7 + 0 = 7$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(2)}\| = |0-7| + |3-0| = 7 + 3 = 10$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(2)}\| = |6-7| + |0-0| = 1 + 0 = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(2)}\| = |7-7| + |0-0| = 0 + 0 = 0$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(2)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(2)}\| = |7-7| + |-3-0| = 0 + 3 = 3$$

We assign each data point to the nearest representative. The new membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(2)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{11}^{(2)} = 1, u_{12}^{(2)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(2)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{21}^{(2)} = 1, u_{22}^{(2)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{31}^{(2)} = 0, u_{32}^{(2)} = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{41}^{(2)} = 0, u_{42}^{(2)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(2)}) > d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(2)}) \Rightarrow u_{51}^{(2)} = 0, u_{52}^{(2)} = 1$$

$$U^{(2)} = \begin{bmatrix} u_{11}^{(2)} & u_{12}^{(2)} \\ u_{21}^{(2)} & u_{22}^{(2)} \\ u_{31}^{(2)} & u_{32}^{(2)} \\ u_{41}^{(2)} & u_{42}^{(2)} \\ u_{51}^{(2)} & u_{52}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

This indicates that there is no change in the cluster assignment for any of the data points.

*Update Step*:

We calculate the new positions of the representatives based on the median of the points assigned to that cluster, separately for each dimension:

$$\boldsymbol{\theta}_1^{(3)} = \text{med}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \text{med}([0,0]^T, [0,3]^T) = [0, 1.5]^T$$
$$\boldsymbol{\theta}_2^{(3)} = \text{med}(\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5) = \text{med}([6,0]^T, [7,0]^T, [7,-3]^T) = [7, 0]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, C_2 : \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$

Since there is no change in the clusters between iterations 2 and 3, the algorithm has converged (the cost function has reached a local minimum), and these are the final clusters.

**Case (b)**

Iteration 1:

*Assignment Step*:

We calculate the distances of all points from all cluster representatives based on their initialized positions, using the Manhattan distance:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_1^{(0)}\| = |0-6| + |0-1| = 6+1 = 7$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_1^{(0)}\| = |0-6| + |3-1| = 6+2 = 8$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_1^{(0)}\| = |6-6| + |0-1| = 0+1 = 1$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_1^{(0)}\| = |7-6| + |0-1| = 1+1 = 2$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_1^{(0)}\| = |7-6| + |-3-1| = 1+4 = 5$$
$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_1 - \boldsymbol{\theta}_2^{(0)}\| = |0-20| + |0-0| = 20+0 = 20$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_2 - \boldsymbol{\theta}_2^{(0)}\| = |0-20| + |3-0| = 20+3 = 23$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_3 - \boldsymbol{\theta}_2^{(0)}\| = |6-20| + |0-0| = 14+0 = 14$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_4 - \boldsymbol{\theta}_2^{(0)}\| = |7-20| + |0-0| = 13+0 = 13$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) = \|\boldsymbol{x}_5 - \boldsymbol{\theta}_2^{(0)}\| = |7-20| + |-3-0| = 13+3 = 16$$

We assign each data point to the nearest representative. The membership matrix $U$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_1, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{11}^{(0)} = 1, u_{12}^{(0)} = 0$$
$$d(\boldsymbol{x}_2, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_2, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{21}^{(0)} = 1, u_{22}^{(0)} = 0$$
$$d(\boldsymbol{x}_3, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_3, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{31}^{(0)} = 1, u_{32}^{(0)} = 0$$
$$d(\boldsymbol{x}_4, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_4, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{41}^{(0)} = 0, u_{42}^{(0)} = 1$$
$$d(\boldsymbol{x}_5, \boldsymbol{\theta}_1^{(0)}) < d(\boldsymbol{x}_5, \boldsymbol{\theta}_2^{(0)}) \Rightarrow u_{51}^{(0)} = 0, u_{52}^{(0)} = 1$$

$$U^{(0)} = \begin{bmatrix} u_{11}^{(0)} & u_{12}^{(0)} \\ u_{21}^{(0)} & u_{22}^{(0)} \\ u_{31}^{(0)} & u_{32}^{(0)} \\ u_{41}^{(0)} & u_{42}^{(0)} \\ u_{51}^{(0)} & u_{52}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

This indicates that all data points are closer to $\boldsymbol{\theta}_1$ than to $\boldsymbol{\theta}_2$.

*Update Step*:

We calculate the new positions of the representatives based on the mean of the assigned points:

$$\boldsymbol{\theta}_1^{(1)} = \text{med}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5) = \text{med}([0,0]^T, [0,3]^T, [6,0]^T, [7,0]^T, [7,-3]^T) = [6,0]^T$$
$$\boldsymbol{\theta}_2^{(1)} = \boldsymbol{\theta}_2^{(0)} = [20,0]^T$$

Formed Clusters: $C_1 : \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}, C_2 : \emptyset$

Due to the bad initialization of the position of representative $\boldsymbol{\theta}_2$, cluster $C_1$ ended up containing all the points in the data (with representative $\boldsymbol{\theta}_1$ being the median of the data set) and cluster $C_2$ ending up empty (hence not being a cluster at all), after a single iteration of the algorithm. Thus, the algorithm has reaches a termination criterion.

In these cases, the hard k-means and k-medians algorithms produced the same clustering results. However, this is not generally the case. If we started from a different or larger data set, or if the initialization of the positions of the representatives was different, it is likely that we would get different results from each algorithm.

**Case (c)**

When employing the hard k-medians clustering algorithm on the given dataset with three representatives, the ideal outcome is the formation of 3 clusters. This is because, similar to k-means (and partition algorithms in general), the number of clusters in k-medians is directly determined by the number of representatives (or medians) initialized at the beginning.

However, the effectiveness of the k-medians algorithm is highly dependent on the proper initialization of these representatives. If the initialization is poorly executed (as in case b), some clusters might end up empty. This occurs when one or more representatives do not end up being the median of any data points after the initial assignment. Consequently, the actual number of clusters obtained could be fewer than 3, depending on the specific approach used to handle these empty clusters - whether they are reinitialized or if the algorithm terminates prematurely. Ensuring a good initialization of the representatives is crucial to realize the full potential of the k-medians algorithm, aiming to obtain 3 distinct clusters.

It's also significant to mention that the limitation of the k-medians algorithm, similar to k-means, lies in its fixed number of clusters based on the number of representatives. Therefore, even if the actual data contains more than 3 natural clusters, employing the k-medians method with only three representatives constrains the output to a maximum of 3 clusters. This restriction could potentially lead to an oversimplified clustering of the data if the inherent structure of the dataset is more nuanced and complex.

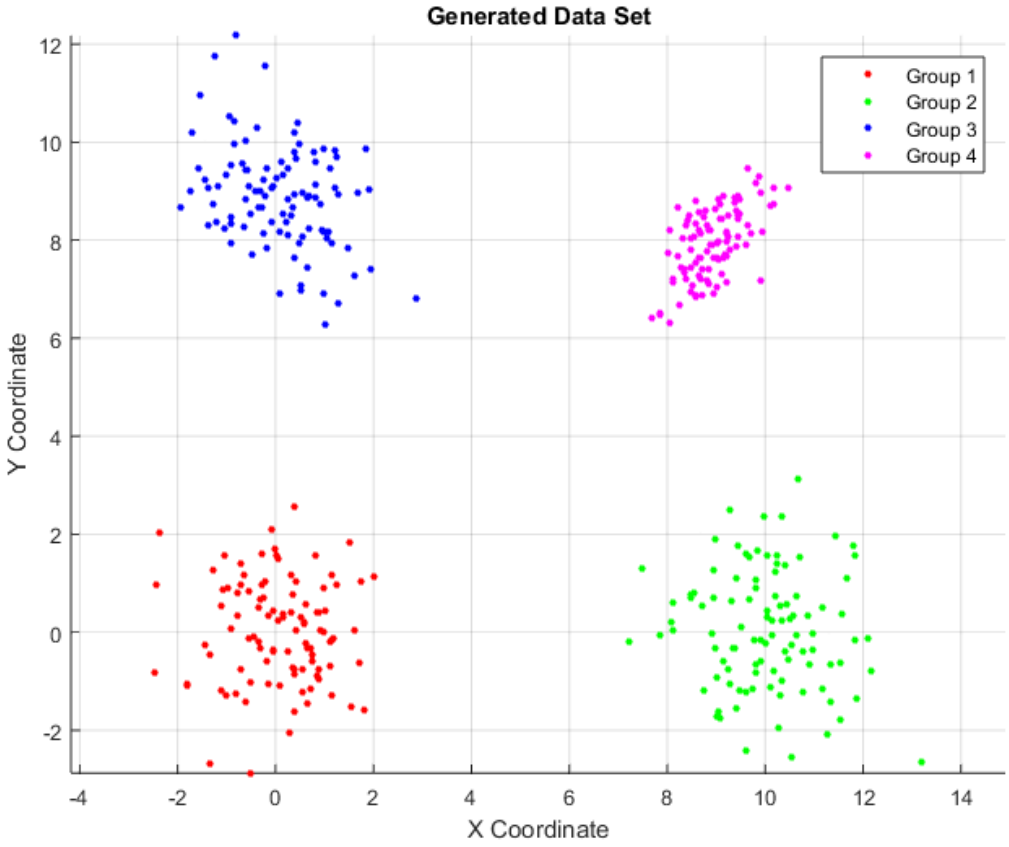
## Exercise 4

We generate a data set consisting of 400 2-dimensional points. The 1st, 2nd, 3rd and 4th groups of 100 points stem from 2-dimensional normal distributions with means $\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$, and covariance matrices $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\Sigma_2 = \begin{bmatrix} 1 & -0.2 \\ -0.2 & 1.5 \end{bmatrix}$, $\Sigma_3 = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 1.1 \end{bmatrix}$, $\Sigma_4 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$, respectively. To achieve this, we run the following Matlab script:

```
% Set the random seed for reproducibility
randn('seed', 0);
% Define the means for each of the 4 distributions
md = [0 0; 10 0; 0 9; 9 8];
% Define the covariance matrices for each distribution
S(:, :, 1) = eye(2); % Identity matrix for the 1st distribution
S(:, :, 2) = [1 -0.2; -0.2 1.5]; % Covariance matrix for the 2nd
    distribution
S(:, :, 3) = [1 -0.4; -0.4 1.1]; % Covariance matrix for the 3rd
    distribution
S(:, :, 4) = [0.3 0.2; 0.2 0.5]; % Covariance matrix for the 4th
    distribution
% Number of points to generate for each distribution
n_points = 100 * ones(1, 4);
% Initialize the array to hold all the points and the labels
X = [];
labels = [];
% Generate the points for each distribution and concatenate them
for i = 1:4
    X_temp = mvnrnd(md(i, :), S(:, :, i), n_points(i));
    X = [X; X_temp]; % Do not transpose, keep it as 400x2
    labels = [labels; i * ones(n_points(i), 1)]; % Create labels
end
```

We plot the data points using different colors for points stemming from different distributions, using the following Matlab script:

```
% Plot the dataset
figure, hold on;
colors = ['r', 'g', 'b', 'm']; % Define the colors for each group
markers = ['.', '.', '.', '.']; % Define the markers for each group
for i = 1:4
    plot(X(labels==i,1), X(labels==i,2), [colors(i) markers(i)], '
        MarkerSize', 12);
end
% Set the axis to be equal and add grid
axis equal;
grid on;
% Add title and labels to the plot
title('Generated Data Set');
xlabel('X Coordinate');
ylabel('Y Coordinate');
% Add a legend
legend('Group 1', 'Group 2', 'Group 3', 'Group 4');
% Hold off to prevent further plotting on the same figure
hold off;
```

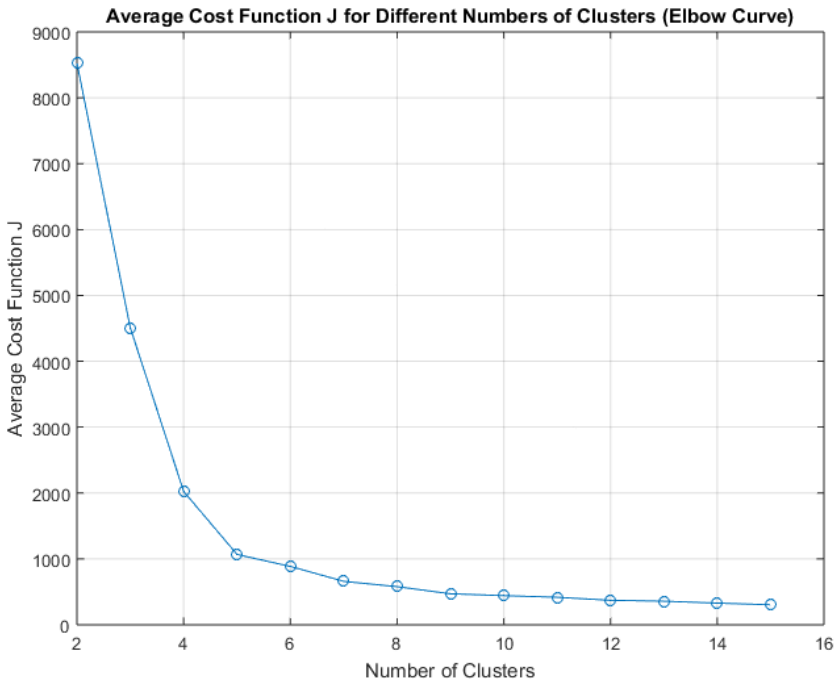



## Hard k-means Clustering Algorithm

We run the hard k-means clustering algorithm (Matlab function that was provided) on the generated data set. To identify the number of physical clusters, we run the algorithm for different values of the number of clusters $m$, computing the value of the cost function $J$ for each result, and we plot $J$ versus the number of clusters $m$ and identify the most significant knee in the graph:

```
X = X';  % Make it an lxN matrix to prepare it as input for the k-means
    algorithm
% Clustering of the data with k-means
% For each m we perform s=100 different runs, with different
    initializations
N = size(X, 2); % Total number of data points
J_tot = [];
for m = 2:15
    J_temp = [];
    for s = 1:100
        te = randperm(N);
        theta_ini = X(:, te(1:m));
        theta = theta_ini;
        [theta, bel, J] = k_means(X, theta);
        J_temp = [J_temp J];
    end
    J = mean(J_temp);
    J_tot = [J_tot J];
end
% Plot J_tot to see how the cost function varies with the number of
    clusters
figure;
```

```
plot(2:15, J_tot, '-o');
title('Average Cost Function J for Different Numbers of Clusters (Elbow Curve)');
xlabel('Number of Clusters');
ylabel('Average Cost Function J');
grid on;
```



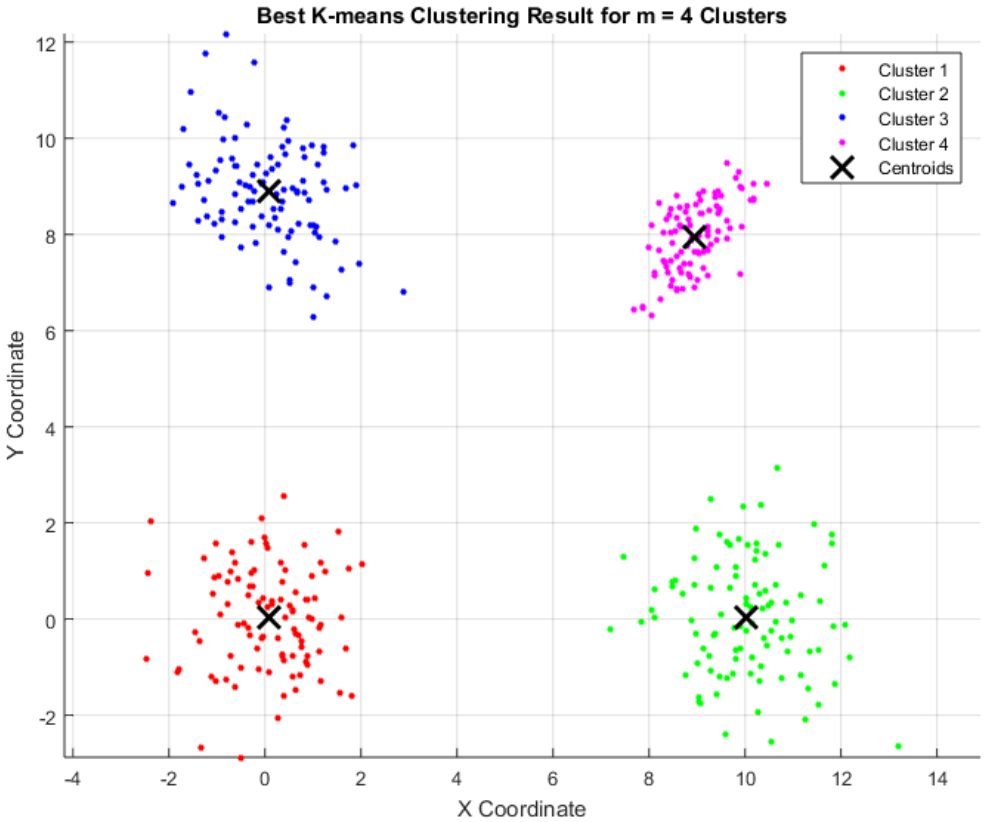

We can observe a knee for $m = 4$, which is an indication that the true number of physical clusters in the data set is 4, which we already know in this case, since we generated the data using 4 different normal distributions.

We proceed by running the k-means algorithm for $m = 4$, in order to get the best possible estimates for the cluster representatives:

```
% Run k-means for the best clustering m=4
m = 4; % Best number of clusters determined from the elbow curve
J_temp = [];
% Initialize bel_temp as an empty matrix to store the labels from each run
bel_temp = [];
for s = 1:50
    te = randperm(N);
    theta_ini = X(:, te(1:m)); % Initial centroids
    theta = theta_ini;
    [theta, bel, J] = k_means(X, theta); % Run k-means
    J_temp = [J_temp J];
    bel_temp = [bel_temp; reshape(bel, 1, N)]; % Reshape bel to be a row vector and concatenate it to bel_temp
end
% Find the run with the minimum cost function J
[J_min, pos] = min(J_temp);
bel_best = bel_temp(pos, :); % Extract the best set of labels corresponding to the minimum J
```

After determining our best clustering result, we plot it:

```
% Visualize the best clustering result
figure, hold on;
colors = ['r', 'g', 'b', 'm']; % Colors for each cluster
cluster_handles = zeros(1, m); % Handles for the clusters for use in
    the legend
centroid_handle = []; % Handle for the centroids for use in the legend
% Plot clusters
for i = 1:m
    cluster_points = X(:, bel_best == i);
    cluster_handles(i) = plot(cluster_points(1, :), cluster_points(2,
        :), [colors(i) '.'], 'MarkerSize', 12);
end
% Plot centroids
centroid_handle = plot(theta(1, :), theta(2, :), 'kx', 'MarkerSize',
    15, 'LineWidth', 2);
% Create legend
legend_handles = [cluster_handles centroid_handle]; % Combine cluster
    and centroid handles
legend(legend_handles, {'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster
    4', 'Centroids'});
title(sprintf('Best K-means Clustering Result for m = %d Clusters', m))
    ;
xlabel('X Coordinate');
ylabel('Y Coordinate');
axis equal;
grid on;
hold off;
```




The final estimates for our representatives are:

$$\boldsymbol{\theta}_1 = [0.0727, 0.0264]^T, \boldsymbol{\theta}_2 = [10.0351, 0.0443]^T, \boldsymbol{\theta}_3 = [0.0752, 8.8939]^T, \boldsymbol{\theta}_4 = [8.9551, 7.9494]^T$$

To quantitatively compare these estimates with the respective means of the distributions ($\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$), we calculate each corresponding pair's squared Euclidean distance:

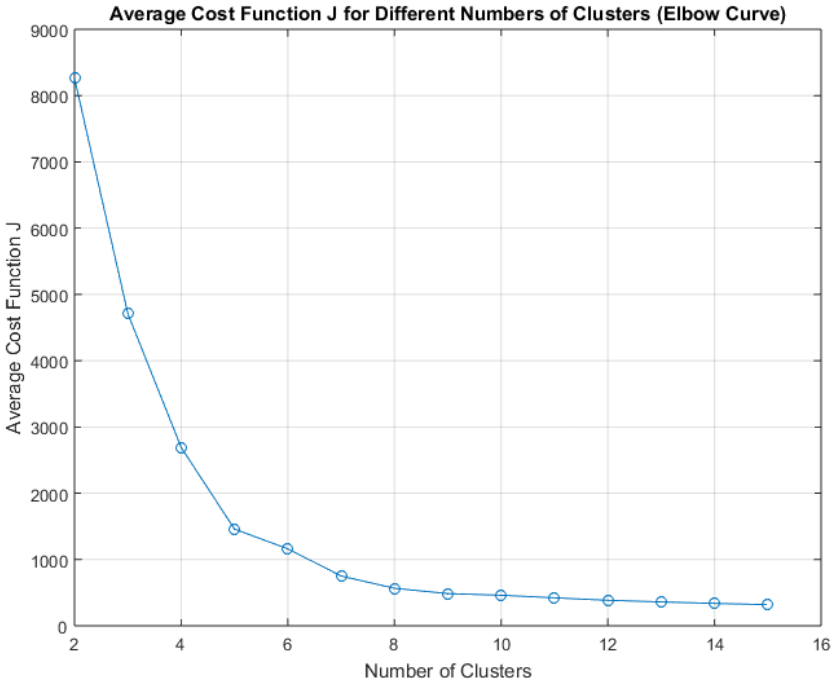
$$\begin{aligned} d(\boldsymbol{m}_1, \boldsymbol{\theta}_1) &= \|\boldsymbol{m}_1 - \boldsymbol{\theta}_1\|^2 = 0.00598225 \\ d(\boldsymbol{m}_2, \boldsymbol{\theta}_2) &= \|\boldsymbol{m}_2 - \boldsymbol{\theta}_2\|^2 = 0.0031945 \\ d(\boldsymbol{m}_3, \boldsymbol{\theta}_3) &= \|\boldsymbol{m}_3 - \boldsymbol{\theta}_3\|^2 = 0.01691225 \\ d(\boldsymbol{m}_4, \boldsymbol{\theta}_4) &= \|\boldsymbol{m}_4 - \boldsymbol{\theta}_4\|^2 = 0.00457637 \end{aligned}$$

## Hard k-medians Clustering Algorithm

We run the hard k-medians clustering algorithm (Matlab function that was provided) on the generated data set. To identify the number of physical clusters, we run the algorithm for different values of the number of clusters $m$, computing the value of the cost function $J$ for each result, and we plot $J$ versus the number of clusters $m$ and identify the most significant knee in the graph:

```
X = X';  % Make it an lxN matrix to prepare it as input for the k-means
    algorithm

% Clustering of the data with k-medians
% For each m we perform s=50 different runs, with different
   initializations
N = size(X, 2); % Total number of data points
J_tot = [];
for m = 2:15
    J_temp = [];
    for s = 1:50
        te = randperm(N);
        theta_ini = X(:, te(1:m)); % Initial centroids
        theta = theta_ini;
        [theta, bel, J] = k_medians(X, theta);
        J_temp = [J_temp J];
    end
    J = mean(J_temp);
    J_tot = [J_tot J];
end
% Plot J_tot to see how the cost function varies with the number of
   clusters
figure;
plot(2:15, J_tot, '-o');
title('Average Cost Function J for Different Numbers of Clusters (Elbow
    Curve)');
xlabel('Number of Clusters');
ylabel('Average Cost Function J');
grid on;
```




We can observe a knee for $m = 5$, which is an indication that the true number of physical clusters in the data set is 5, even though we already know that in this case, since we generated the data using 4 different normal distributions, we expect 4 physical clusters.

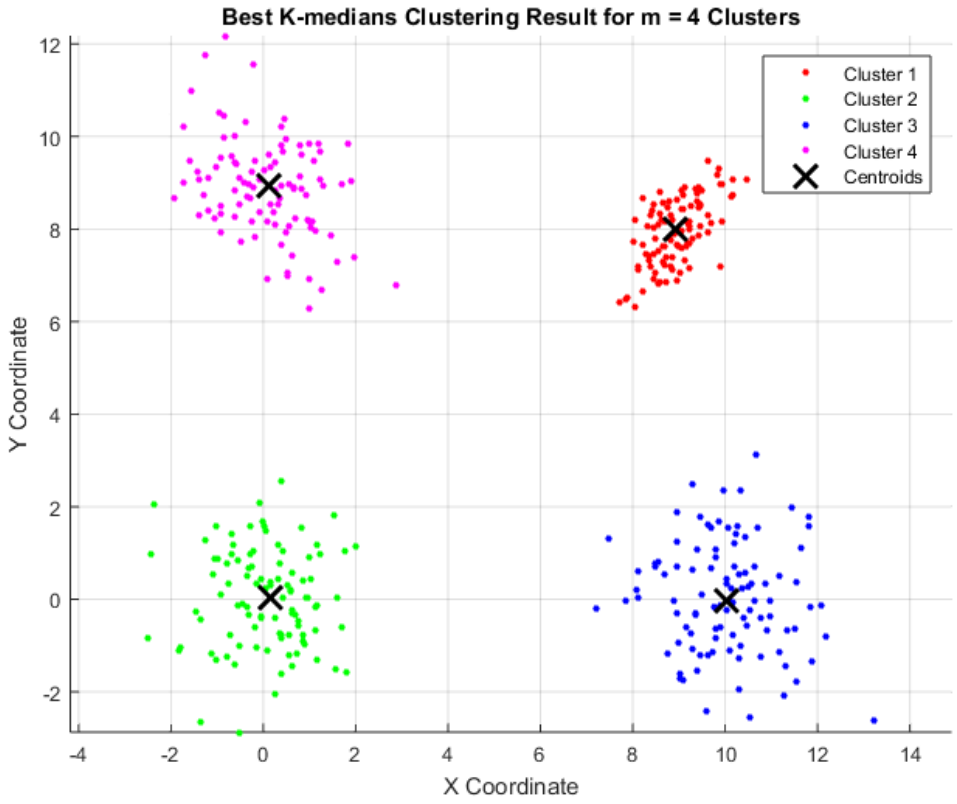
We proceed by running the k-medians algorithm for $m = 4$, in order to get the best possible estimates for the physical cluster representatives:

```
% Run k-medians for the best clustering m=4
m = 4; % Best number of clusters determined from the elbow curve
J_temp = [];
% Initialize bel_temp as an empty matrix to store the labels from each
    run
bel_temp = [];
for s = 1:50
    te = randperm(N);
    theta_ini = X(:, te(1:m)); % Initial centroids
    theta = theta_ini;
    [theta, bel, J] = k_medians(X, theta);
    J_temp = [J_temp J];
    bel_temp = [bel_temp; reshape(bel, 1, N)]; % Reshape bel to be a
        row vector and concatenate it to bel_temp
end
% Find the run with the minimum cost function J
[J_min, pos] = min(J_temp);
bel_best = bel_temp(pos, :); % Extract the best set of labels
    corresponding to the minimum J
```

After determining our best clustering result, we plot it:

```
% Visualize the best clustering result
figure, hold on;
colors = ['r', 'g', 'b', 'm']; % Colors for each cluster
cluster_handles = zeros(1, m); % Handles for the clusters for use in
    the legend
centroid_handle = []; % Handle for the centroids for use in the legend
% Plot clusters
```

```
for i = 1:m
    cluster_points = X(:, bel_best == i);
    cluster_handles(i) = plot(cluster_points(1, :), cluster_points(2, :), [colors(i) '.'], 'MarkerSize', 12);
end
% Plot centroids
centroid_handle = plot(theta(1, :), theta(2, :), 'kx', 'MarkerSize', 15, 'LineWidth', 2);
% Create legend
legend_handles = [cluster_handles centroid_handle]; % Combine cluster and centroid handles
legend(legend_handles, {'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Centroids'});
title(sprintf('Best K-medians Clustering Result for m = %d Clusters', m)); % Updated to say K-medians
xlabel('X Coordinate');
ylabel('Y Coordinate');
axis equal;
grid on;
hold off;
```




The final estimates for our representatives are:

$$\boldsymbol{\theta}_1 = [0.1489, 0.0301]^T, \boldsymbol{\theta}_2 = [10.0443, -0.0348]^T, \boldsymbol{\theta}_3 = [0.1375, 8.9308]^T, \boldsymbol{\theta}_4 = [8.9249, 8.0012]^T$$

To quantitatively compare these estimates with the respective means of the distributions ($\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$), we calculate each corresponding pair's squared Euclidean distance:

$$
\begin{aligned}
d(\boldsymbol{m}_1, \boldsymbol{\theta}_1) &= \|\boldsymbol{m}_1 - \boldsymbol{\theta}_1\|^2 = 0.02307722 \\
d(\boldsymbol{m}_2, \boldsymbol{\theta}_2) &= \|\boldsymbol{m}_2 - \boldsymbol{\theta}_2\|^2 = 0.00317353 \\
d(\boldsymbol{m}_3, \boldsymbol{\theta}_3) &= \|\boldsymbol{m}_3 - \boldsymbol{\theta}_3\|^2 = 0.02369489 \\
d(\boldsymbol{m}_4, \boldsymbol{\theta}_4) &= \|\boldsymbol{m}_4 - \boldsymbol{\theta}_4\|^2 = 0.00564145
\end{aligned}
$$

## Exercise 5

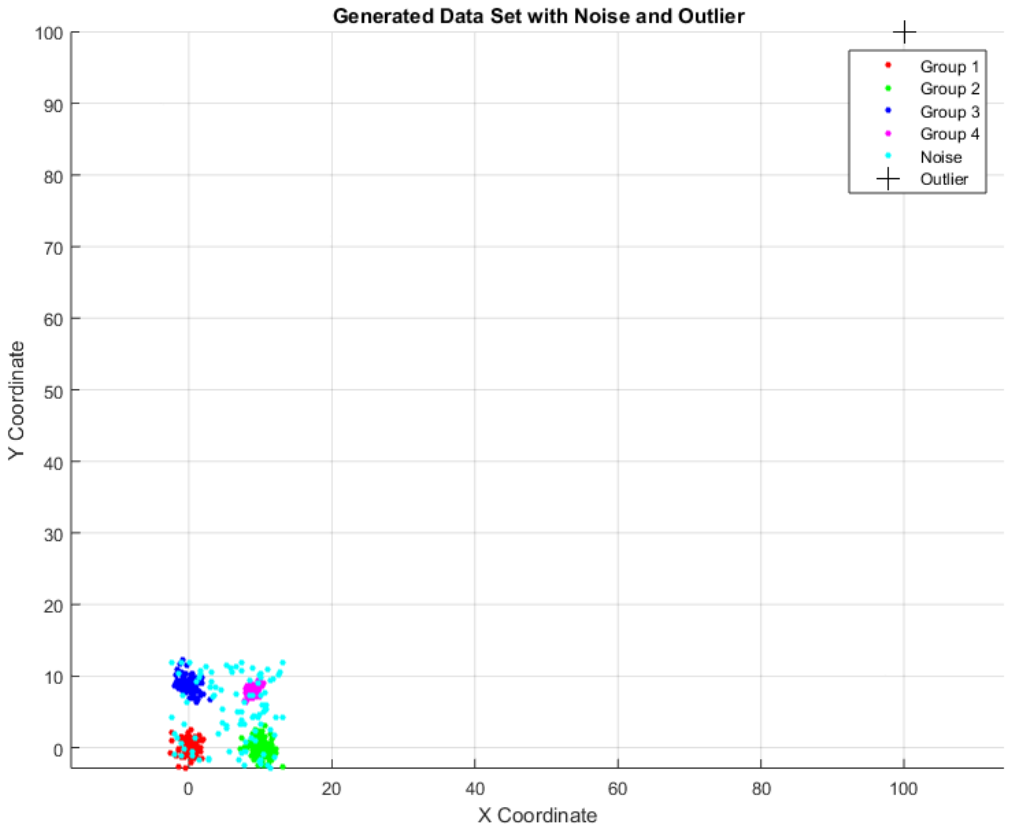
We generate a data set consisting of 500 2-dimensional points. The 1st, 2nd, 3rd and 4th groups of 100 points stem from 2-dimensional normal distributions with means $\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$, and covariance matrices $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\Sigma_2 = \begin{bmatrix} 1 & -0.2 \\ -0.2 & 1.5 \end{bmatrix}$, $\Sigma_3 = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 1.1 \end{bmatrix}$, $\Sigma_4 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$, respectively. The next 99 points are noisy points that are uniformly spread among the points resulting from the above four distributions and the last point is the $\boldsymbol{x}_{500} = [100,100]$. To achieve this, we run the following Matlab script:

```
% Set the random seed for reproducibility
randn('seed', 0);
% Define the means for each of the 4 distributions
md = [0 0; 10 0; 0 9; 9 8];
% Define the covariance matrices for each distribution
S(:, :, 1) = eye(2); % Identity matrix for the 1st distribution
S(:, :, 2) = [1 -0.2; -0.2 1.5]; % Covariance matrix for the 2nd
    distribution
S(:, :, 3) = [1 -0.4; -0.4 1.1]; % Covariance matrix for the 3rd
    distribution
S(:, :, 4) = [0.3 0.2; 0.2 0.5]; % Covariance matrix for the 4th
    distribution
% Number of points to generate for each distribution
n_points = 100 * ones(1, 4);
% Initialize the array to hold all the points and the labels
X = [];
labels = [];
% Generate the points for each distribution and concatenate them
for i = 1:4
    X_temp = mvnrnd(md(i, :), S(:, :, i), n_points(i));
    X = [X; X_temp]; % Keep as 400x2
    labels = [labels; i * ones(n_points(i), 1)]; % Create labels
end
% Determine the range of the current dataset to define the bounds for
    the noise
min_vals = min(X, [], 1);
max_vals = max(X, [], 1);
% Generate 99 noisy points uniformly spread among the existing points
noise_points = bsxfun(@plus, bsxfun(@times, rand(99, 2), (max_vals -
    min_vals)), min_vals);
X = [X; noise_points];
labels = [labels; 5 * ones(99, 1)]; % Label for noise
% Add the last point [100, 100]
X = [X; [100, 100]];
labels = [labels; 6]; % Label for the outlier
```

We plot the data points using different colors for points stemming from different distributions, as well as the noisy points and the outlier, using the following Matlab script:

```
% Plot the dataset
figure, hold on;
colors = ['r', 'g', 'b', 'm', 'c', 'k']; % Define the colors for each
    group, including noise and outlier
markers = ['.', '.', '.', '.', '.', '+']; % Define the markers for each
     group, including noise and outlier
for i = 1:6
    plot(X(labels==i,1), X(labels==i,2), [colors(i) markers(i)], '
        MarkerSize', 12);
```

```
end
% Set the axis to be equal and add grid
axis equal;
grid on;
% Add title and labels to the plot
title('Generated Data Set with Noise and Outlier');
xlabel('X Coordinate');
ylabel('Y Coordinate');
% Add a legend
legend('Group 1', 'Group 2', 'Group 3', 'Group 4', 'Noise', 'Outlier');
% Hold off to prevent further plotting on the same figure
hold off;
```

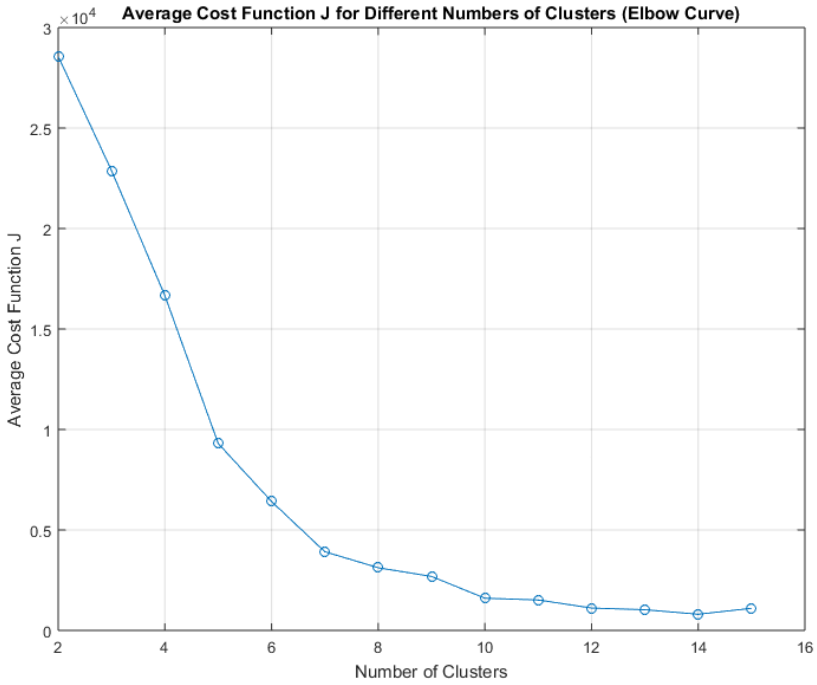



## Hard k-means Clustering Algorithm

We run the hard k-means clustering algorithm (Matlab function that was provided) on the generated data set. To identify the number of physical clusters, we run the algorithm for different values of the number of clusters $m$, computing the value of the cost function $J$ for each result, and we plot $J$ versus the number of clusters $m$ and identify the most significant knee in the graph:

```
X = X';  % Make it an lxN matrix to prepare it as input for the k-means
    algorithm
% Clustering of the data with k-means
% For each m we perform s=100 different runs, with different
    initializations
N = size(X, 2); % Total number of data points
J_tot = [];
for m = 2:15
```

```
    J_temp = [];
    for s = 1:100
        te = randperm(N);
        theta_ini = X(:, te(1:m));
        theta = theta_ini;
        [theta, bel, J] = k_means(X, theta);
        J_temp = [J_temp J];
    end
    J = mean(J_temp);
    J_tot = [J_tot J];
end
% Plot J_tot to see how the cost function varies with the number of
    clusters
figure;
plot(2:15, J_tot, '-o');
title('Average Cost Function J for Different Numbers of Clusters (Elbow
    Curve)');
xlabel('Number of Clusters');
ylabel('Average Cost Function J');
grid on;
```




We can observe a (not very clear) knee for $m = 5$, which is an indication that the true number of physical clusters in the data set is 5, even though we already know that in this case, since we generated the data using 4 different normal distributions, we expect 4 physical clusters.

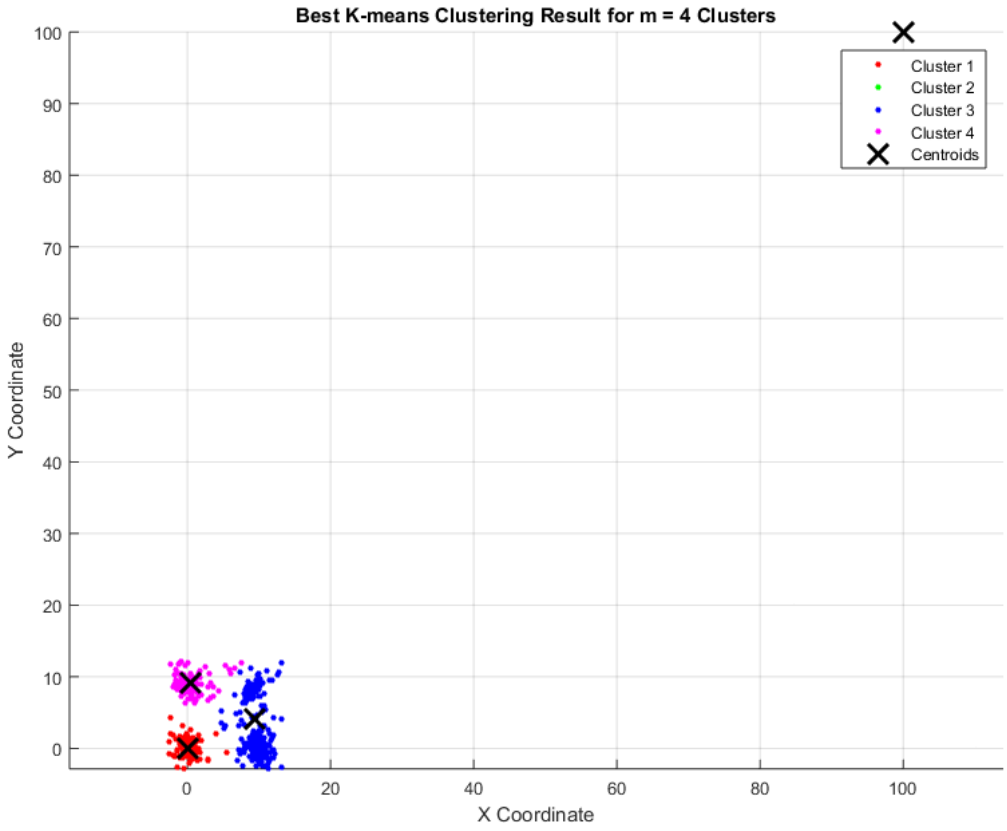
We proceed by running the k-medians algorithm for $m = 4$, in order to get the best possible estimates for the physical cluster representatives:

```
% Run k-means for the best clustering m=4
m = 4; % Best number of clusters determined from the elbow curve
J_temp = [];
% Initialize bel_temp as an empty matrix to store the labels from each
    run
bel_temp = [];
for s = 1:50
```

```
    te = randperm(N);
    theta_ini = X(:, te(1:m)); % Initial centroids
    theta = theta_ini;
    [theta, bel, J] = k_means(X, theta); % Run k-means
    J_temp = [J_temp J];
    bel_temp = [bel_temp; reshape(bel, 1, N)]; % Reshape bel to be a
        row vector and concatenate it to bel_temp
end
% Find the run with the minimum cost function J
[J_min, pos] = min(J_temp);
bel_best = bel_temp(pos, :); % Extract the best set of labels
    corresponding to the minimum J
```

After determining our best clustering result, we plot it:

```
% Visualize the best clustering result
figure, hold on;
colors = ['r', 'g', 'b', 'm']; % Colors for each cluster
cluster_handles = zeros(1, m); % Handles for the clusters for use in
    the legend
centroid_handle = []; % Handle for the centroids for use in the legend
% Plot clusters
for i = 1:m
    cluster_points = X(:, bel_best == i);
    cluster_handles(i) = plot(cluster_points(1, :), cluster_points(2,
        :), [colors(i) '.'], 'MarkerSize', 12);
end
% Plot centroids
centroid_handle = plot(theta(1, :), theta(2, :), 'kx', 'MarkerSize',
    15, 'LineWidth', 2);
% Create legend
legend_handles = [cluster_handles centroid_handle]; % Combine cluster
    and centroid handles
legend(legend_handles, {'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster
    4', 'Centroids'});
title(sprintf('Best K-means Clustering Result for m = %d Clusters', m))
    ;
xlabel('X Coordinate');
ylabel('Y Coordinate');
axis equal;
grid on;
hold off;
```




The final estimates for our representatives are:

$$\boldsymbol{\theta}_1 = [0.1471, 0.0542]^T, \boldsymbol{\theta}_2 = [9.4738, 4.1055]^T, \boldsymbol{\theta}_3 = [0.5321, 9.0729]^T, \boldsymbol{\theta}_4 = [100, 100]^T$$

To quantitatively compare these estimates with the respective means of the distributions ($\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$), we calculate each corresponding pair's squared Euclidean distance:

$$d(\boldsymbol{m}_1, \boldsymbol{\theta}_1) = \|\boldsymbol{m}_1 - \boldsymbol{\theta}_1\|^2 = 0.02457605$$
$$d(\boldsymbol{m}_2, \boldsymbol{\theta}_2) = \|\boldsymbol{m}_2 - \boldsymbol{\theta}_2\|^2 = 17.13201669$$
$$d(\boldsymbol{m}_3, \boldsymbol{\theta}_3) = \|\boldsymbol{m}_3 - \boldsymbol{\theta}_3\|^2 = 0.28844482$$
$$d(\boldsymbol{m}_4, \boldsymbol{\theta}_4) = \|\boldsymbol{m}_4 - \boldsymbol{\theta}_4\|^2 = 16,745$$

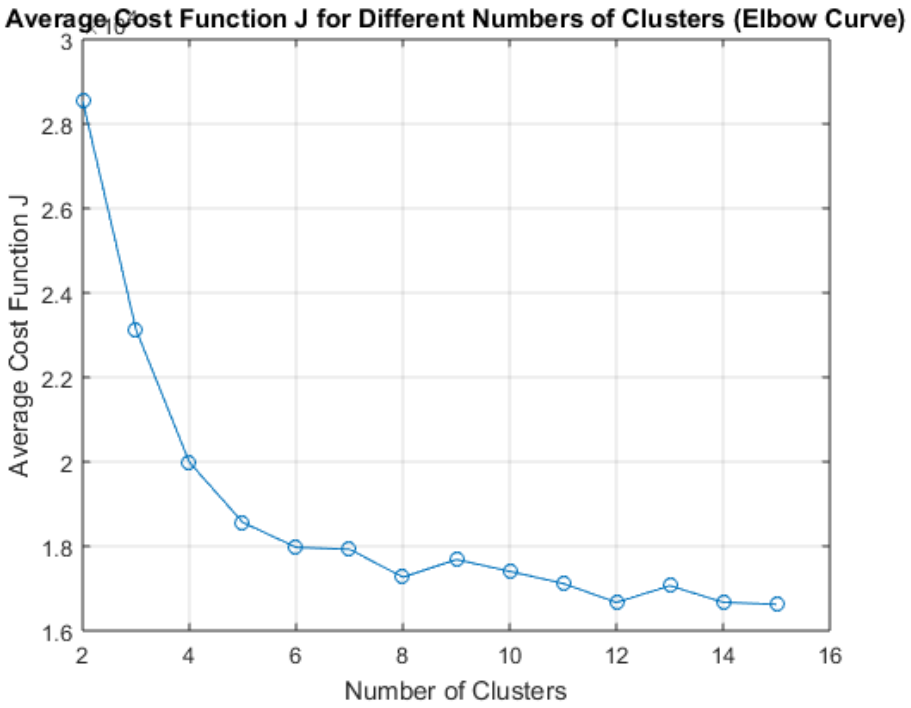
## Hard k-medians Clustering Algorithm

We run the hard k-medians clustering algorithm (Matlab function that was provided) on the generated data set. To identify the number of physical clusters, we run the algorithm for different values of the number of clusters $m$, computing the value of the cost function $J$ for each result, and we plot $J$ versus the number of clusters $m$ and identify the most significant knee in the graph:

```
X = X';  % Make it an lxN matrix to prepare it as input for the k-means
    algorithm

% Clustering of the data with k-medians
% For each m we perform s=50 different runs, with different
    initializations
N = size(X, 2); % Total number of data points
J_tot = [];
for m = 2:15
```

```
    J_temp = [];
    for s = 1:50
        te = randperm(N);
        theta_ini = X(:, te(1:m)); % Initial centroids
        theta = theta_ini;
        [theta, bel, J] = k_medians(X, theta);
        J_temp = [J_temp J];
    end
    J = mean(J_temp);
    J_tot = [J_tot J];
end
% Plot J_tot to see how the cost function varies with the number of
    clusters
figure;
plot(2:15, J_tot, '-o');
title('Average Cost Function J for Different Numbers of Clusters (Elbow
    Curve)');
xlabel('Number of Clusters');
ylabel('Average Cost Function J');
grid on;
```




We can observe a (not very clear) knee for $m = 5$, which is an indication that the true number of physical clusters in the data set is 5, even though we already know that in this case, since we generated the data using 4 different normal distributions, we expect 4 physical clusters.

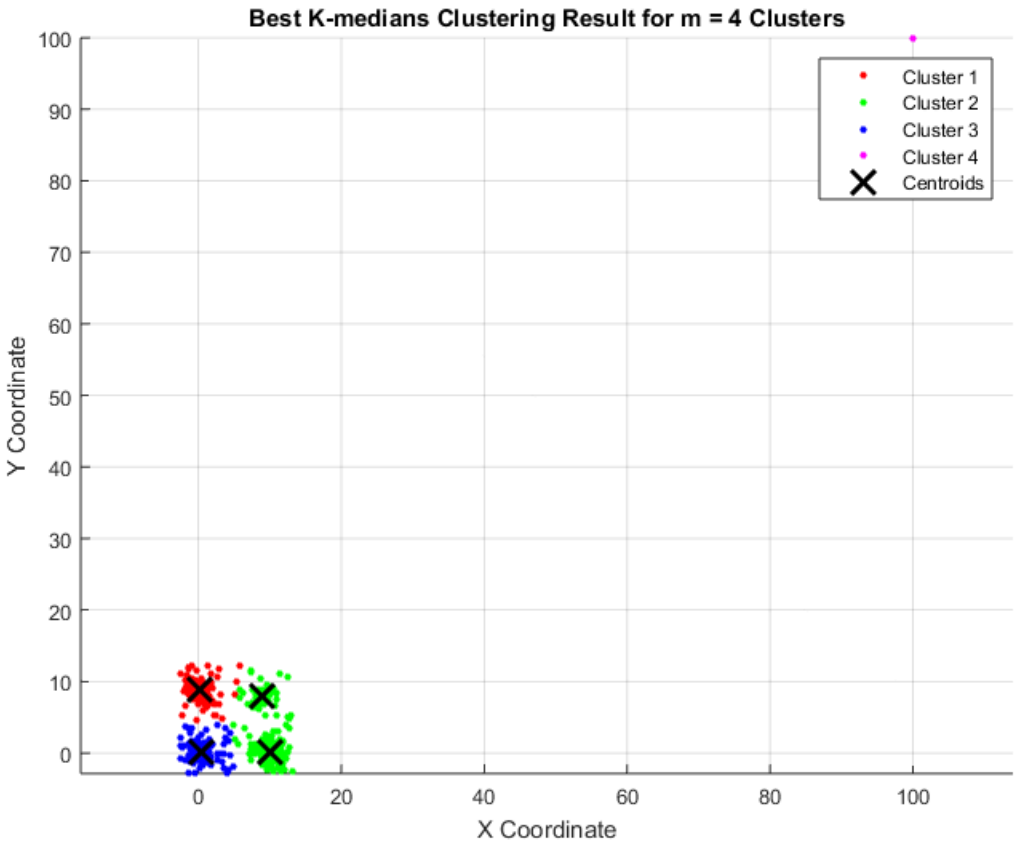
We proceed by running the k-medians algorithm for $m = 4$, in order to get the best possible estimates for the physical cluster representatives:

```
% Run k-medians for the best clustering m=4
m = 4; % Best number of clusters determined from the elbow curve
J_temp = [];
% Initialize bel_temp as an empty matrix to store the labels from each
    run
bel_temp = [];
for s = 1:50
```

```
    te = randperm(N);
    theta_ini = X(:, te(1:m)); % Initial centroids
    theta = theta_ini;
    [theta, bel, J] = k_medians(X, theta);
    J_temp = [J_temp J];
    bel_temp = [bel_temp; reshape(bel, 1, N)]; % Reshape bel to be a
        row vector and concatenate it to bel_temp
end
% Find the run with the minimum cost function J
[J_min, pos] = min(J_temp);
bel_best = bel_temp(pos, :); % Extract the best set of labels
    corresponding to the minimum J
```

After determining our best clustering result, we plot it:

```
% Visualize the best clustering result
figure, hold on;
colors = ['r', 'g', 'b', 'm']; % Colors for each cluster
cluster_handles = zeros(1, m); % Handles for the clusters for use in
    the legend
centroid_handle = []; % Handle for the centroids for use in the legend
% Plot clusters
for i = 1:m
    cluster_points = X(:, bel_best == i);
    cluster_handles(i) = plot(cluster_points(1, :), cluster_points(2,
        :), [colors(i) '.'], 'MarkerSize', 12);
end
% Plot centroids
centroid_handle = plot(theta(1, :), theta(2, :), 'kx', 'MarkerSize',
    15, 'LineWidth', 2);
% Create legend
legend_handles = [cluster_handles centroid_handle]; % Combine cluster
    and centroid handles
legend(legend_handles, {'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster
    4', 'Centroids'});
title(sprintf('Best K-medians Clustering Result for m = %d Clusters', m
    )); % Updated to say K-medians
xlabel('X Coordinate');
ylabel('Y Coordinate');
axis equal;
grid on;
hold off;
```




The final estimates for our representatives are:

$$\boldsymbol{\theta}_1 = [0.3516, 0.0414]^T, \boldsymbol{\theta}_2 = [10.0173, 0.0848]^T, \boldsymbol{\theta}_3 = [0.1894, 8.8765]^T, \boldsymbol{\theta}_4 = [8.9249, 7.9745]^T$$

To quantitatively compare these estimates with the respective means of the distributions ($\boldsymbol{m}_1 = [0,0]^T, \boldsymbol{m}_2 = [10,0]^T, \boldsymbol{m}_3 = [0,9]^T, \boldsymbol{m}_4 = [9,8]^T$), we calculate each corresponding pair's squared Euclidean distance:

$$d(\boldsymbol{m}_1, \boldsymbol{\theta}_1) = \|\boldsymbol{m}_1 - \boldsymbol{\theta}_1\|^2 = 0.12533652$$
$$d(\boldsymbol{m}_2, \boldsymbol{\theta}_2) = \|\boldsymbol{m}_2 - \boldsymbol{\theta}_2\|^2 = 0.00749033$$
$$d(\boldsymbol{m}_3, \boldsymbol{\theta}_3) = \|\boldsymbol{m}_3 - \boldsymbol{\theta}_3\|^2 = 0.05112461$$
$$d(\boldsymbol{m}_4, \boldsymbol{\theta}_4) = \|\boldsymbol{m}_4 - \boldsymbol{\theta}_4\|^2 = 0.00629026$$

# Exercise 6

Both of the previous two exercises involve generating datasets with known distributions and applying the hard k-means and k-medians clustering algorithms to cluster the data.

The k-means algorithm is known for its simplicity and efficiency on large datasets due to its $O(q \cdot m \cdot N)$ complexity, where $q$ is the number of iterations, $m$ is the number of clusters, and $N$ is the number of data points. However, it assumes the prior knowledge of the number of clusters, is sensitive to outliers, and can converge to local minima, which may lead to different results with different initial partitions.

The k-medians algorithm, on the other hand, is less sensitive to outliers as it uses medians instead of means as the cluster representatives, which can be advantageous in the presence of noise and outliers. However, it shares some of the drawbacks of k-means, such as the requirement of specifying the number of clusters beforehand and the potential for converging to local minima.

From the results and the two previous exercises, the following observations can be made:

- In Exercise 4, both algorithms effectively identified the true clusters, with the k-means representatives being a little closer to the means of the distributions, since the data was well-separated and there were no outliers.
- In Exercise 5, the algorithms' performance diverged; k-means was more affected by the outlier, which resulted in one of the centroids being placed on the outlier itself, while k-medians was less affected due to its robustness to outliers.
- The presence of noise and an outlier in the second exercise significantly impacts the clustering results. The k-means algorithm showed a clear weakness in handling outliers in Exercise 5 compared to Exercise 4, where no outliers were present. The k-medians algorithm, while more computationally intensive (due to the more complicated calculation of medians versus means), maintained its performance level in the presence of outliers from Exercise 4 to Exercise 5.
- For the k-means algorithm, the squared Euclidean distances between the actual means of the distributions and the estimated cluster centroids increased in the presence of noise and outliers, whereas for k-medians, the increase in distances was less pronounced, indicating its relative robustness.

To improve the results, especially for k-means, several strategies could be employed, such as:

- Utilizing sequential or online versions of k-means for better initial centroid placement.
- Applying stochastic optimization techniques to escape local minima.
- Discarding small clusters that likely formed due to outliers.
- Using k-medoids to represent clusters by actual data points rather than means, which might be more suitable for datasets with nominal features or significant noise and outliers.

In summary, while k-means is efficient for large datasets with well-separated clusters, k-medians is more suitable for datasets prone to noise and outliers. The choice between the two algorithms should be made based on the specific characteristics of the data and the problem at hand.