Experiment 7
Data Normalization & Clustering
Lab Manual

Normalization is a data preprocessing technique used to scale the values of features in a dataset to a standard range. Decimal scaling, z-score (standardization), and min-max scaling are three common methods for normalization.

1. Decimal Scaling:
In decimal scaling, we divide each value by a power of 10 to scale it between -1 and 1.

```
import numpy as np

def decimal_scaling(data):
    max_value = np.max(np.abs(data))
    scaled_data = data / (10 ** (len(str(int(max_value))) - 1))
    return scaled_data

# Example usage:
data = np.array([25, 100, 1000, 5000])
scaled_data = decimal_scaling(data)
print(scaled_data)
```

2. Z-Score (Standardization):
Z-score scaling transforms the data to have a mean of 0 and a standard deviation of 1.

```
import numpy as np

def z_score_scaling(data):
    mean = np.mean(data)
    std_dev = np.std(data)
    standardized_data = (data - mean) / std_dev
    return standardized_data

# Example usage:
data = np.array([25, 100, 1000, 5000])
standardized_data = z_score_scaling(data)
print(standardized_data)
```

3. Min-Max Scaling:
Min-max scaling scales the data to a specified range, typically between 0 and 1.

```python
import numpy as np

def min_max_scaling(data, min_value=0, max_value=1):
    min_data = np.min(data)
    max_data = np.max(data)
    scaled_data = (data - min_data) / (max_data - min_data) * (max_value - min_value) + min_value
    return scaled_data

# Example usage:
data = np.array([25, 100, 1000, 5000])
scaled_data = min_max_scaling(data)
print(scaled_data)
```

You can use these functions to normalize your data using the respective method that suits your needs. Simply pass your data as a NumPy array to the appropriate function, and it will return the normalized data.

Clustering of Data(K-means Method):

The k-means clustering algorithm, a popular unsupervised machine learning technique used for clustering data points into groups (clusters) based on their similarity.

EXAMPLE:

Cluster the following eight points (with (x, y) representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).

The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

$$P(a, b) = |x2 – x1| + |y2 – y1|$$

Use K-Means Algorithm to find the three cluster centers after the second iteration.

Solution-

We follow the above discussed K-Means Clustering Algorithm-

Iteration-01:

- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.

The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-

Calculating Distance Between A1(2, 10) and C1(2, 10)-

P(A1, C1)

$= |x2 – x1| + |y2 – y1|$

$= |2 – 2| + |10 – 10|$

$= 0$

Calculating Distance Between A1(2, 10) and C2(5, 8)-

P(A1, C2)

$= |x2 – x1| + |y2 – y1|$

$= |5 – 2| + |8 – 10|$

$= 3 + 2$

$= 5$

Calculating Distance Between A1(2, 10) and C3(1, 2)-

P(A1, C3)

$= |x2 – x1| + |y2 – y1|$

$= |1 – 2| + |2 – 10|$

$= 1 + 8$

$= 9$

In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Next,

- We draw a table showing all the results.
- Using the table, we decide which point belongs to which cluster.
- The given point belongs to that cluster whose center is nearest to it.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

From here, New clusters are-

Cluster-01:

First cluster contains points-

- A1(2, 10)

Cluster-02:

Second cluster contains points-

- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)

Cluster-03:

Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01:

- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.

For Cluster-02:

Center of Cluster-02

= ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)

= (6, 6)

For Cluster-03:

Center of Cluster-03

= ((2 + 1)/2, (5 + 2)/2)

= (1.5, 3.5)

This is completion of Iteration-01.

Iteration-02:

- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.

The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-

Calculating Distance Between A1(2, 10) and C1(2, 10)-

P(A1, C1)

$= |x2 – x1| + |y2 – y1|$

$= |2 – 2| + |10 – 10|$

$= 0$

Calculating Distance Between A1(2, 10) and C2(6, 6)-

P(A1, C2)

$= |x2 – x1| + |y2 – y1|$

$= |6 – 2| + |6 – 10|$

$= 4 + 4$

$= 8$

Calculating Distance Between A1(2, 10) and C3(1.5, 3.5)-

P(A1, C3)

$= |x2 – x1| + |y2 – y1|$

$= |1.5 – 2| + |3.5 – 10|$

$= 0.5 + 6.5$

$= 7$

In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Next,

- We draw a table showing all the results.
- Using the table, we decide which point belongs to which cluster.
- The given point belongs to that cluster whose center is nearest to it.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

From here, New clusters are-

Cluster-01:

First cluster contains points-

- A1(2, 10)
- A8(4, 9)

Cluster-02:

Second cluster contains points-

- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)

Cluster-03:

Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01:

Center of Cluster-01

= ((2 + 4)/2, (10 + 9)/2)

= (3, 9.5)

For Cluster-02:

Center of Cluster-02

= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)

= (6.5, 5.25)

For Cluster-03:

Center of Cluster-03

= ((2 + 1)/2, (5 + 2)/2)

= (1.5, 3.5)

This is completion of Iteration-02.

After second iteration, the center of the three clusters are-

- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)

Program:

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

# Generate a sample dataset for demonstration purposes
#np.random.seed(0)
#X = np.random.rand(100, 2)

X = np.array([[2, 10], [2, 5], [8, 4],
              [5, 8], [7, 5], [6, 4],[1,2],[4,9]])

# Number of clusters (you can change this)

K = 3

# Maximum number of iterations
max_iterations = 100

# Initialize cluster centroids randomly
centroids = X[np.random.choice(range(len(X)), K, replace=False)]

# Helper function to calculate Euclidean distance
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

# Main K-Means algorithm
for _ in range(max_iterations):
    # Assign each data point to the nearest centroid
    cluster_assignments = []
    for x in X:
```

```
        distances = [euclidean_distance(x, centroid) for centroid in
centroids]
        closest_cluster = np.argmin(distances)
        cluster_assignments.append(closest_cluster)

    # Update centroids by calculating the mean of each cluster
    new_centroids = []
    for cluster_id in range(K):
        cluster_points = [X[i] for i, cluster in
enumerate(cluster_assignments) if cluster == cluster_id]
        new_centroid = np.mean(cluster_points, axis=0)
        new_centroids.append(new_centroid)

    # Check for convergence
    if np.all(np.array(centroids) == np.array(new_centroids)):
        break

    centroids = new_centroids
    for i in range(3):
        print("Centroids of Data Points",centroids)
# Convert cluster assignments to numpy array for plotting
cluster_assignments = np.array(cluster_assignments)

# Plot the data points and centroids
plt.scatter(X[:, 0], X[:, 1], c=cluster_assignments, cmap='rainbow')
plt.scatter(np.array(centroids)[:, 0], np.array(centroids)[:, 1], c='black',
marker='x', s=100)
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```