CSE381: Introduction to Machine Learning

Fall 2024

Lab Assignment 1

Name: Mostafa Hassan Mohamed Atya

ID : 21p0349

Instructor: Dr. Mahmoud Khalil

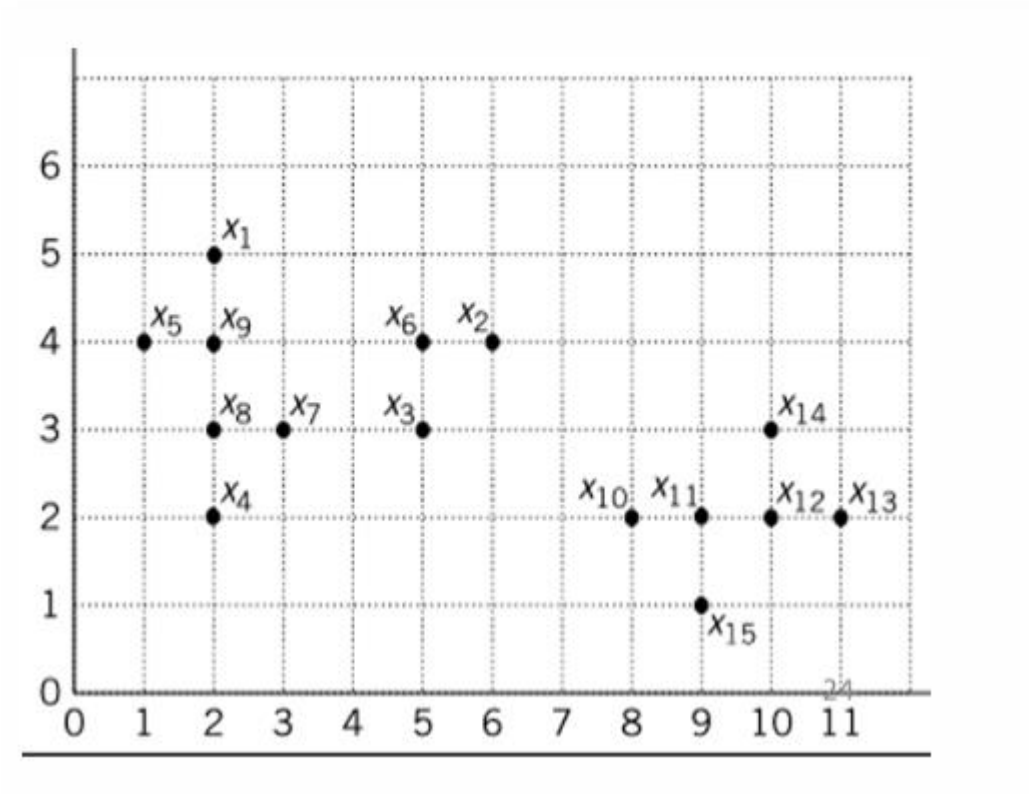
TA: Eng. Engy Ahmed Hassan

1 Problem definition:

Implement the BSAS (Basic Sequential Algorithmic Scheme) algorithm in Python based on the provided pseudocode. The output should be a JSON-formatted list, where each element represents a cluster containing:

- "Cluster": The cluster number.
- "Members": A list of data points in the cluster.
- "Mean": The mean value of the cluster

1.1 Data points to test my code :



1.2 Details of the task:

a) Apply the BSAS algorithm on X , presenting its elements in the

- **order x8, x6, x11, x1, x5, x2, x3, x4, x7, x10, x9, x12, x13, x14, x15**
- **for $\Theta = 2.5$**
- **And q = 15 .**

2 Code screenshots with comments

2.1 Code Screen shots :

```python
# Course code: CSE381
# Course Name: Introduction to Machine Learning
# Instructor: Dr. Mahmoud Khalil
# TA: Eng. Engy Ahmed Hassan
# Lab Assignment 1

# Task:
# Implement the BSAS algorithm in Python. The output should be a JSON-formatted list where each element represents a cluster containing:
# - "Cluster": The cluster number.
# - "Members": A list of data points in the cluster.
# - "Mean": The mean value of the cluster.

import numpy as np
import json
#NOTE : IF YOU WANT TO TEST MY CODE WITH DIFFERENT DATA POINTS CHANGE THE DATA POINTS LIST OF POINTS
# Input data points (in the order provided)
data_points = [
    (2, 3), (5, 4), (9, 2), (2, 5), (1, 4),
    (6, 4), (5, 3), (2, 2), (3, 3), (8, 2),
    (2, 4), (10, 2), (11, 2), (10, 3), (9, 1)
]

# Parameters
Theta = 2.5
q = 15

# Initialize variables
clusters = []  # To store clusters
representatives = []  # To store cluster representatives (centroids)
```

```python
# BSAS Algorithm Implementation
for point in data_points:
    if not clusters:  # Start the first cluster with the first data point
        clusters.append([point])
        representatives.append(np.array(point))
    else:
        # Calculate distances between the point and existing cluster representatives
        distances = [np.linalg.norm(np.array(point) - rep) for rep in representatives]
        min_distance = min(distances)
        closest_cluster_idx = distances.index(min_distance)

        # Check if the point should form a new cluster or be added to an existing cluster
        if min_distance > Theta and len(clusters) < q:
            clusters.append([point])  # Create a new cluster
            representatives.append(np.array(point))
        else:
            # Add the point to the closest cluster
            clusters[closest_cluster_idx].append(point)
            # Update the representative (mean) of the cluster
            cluster_points = np.array(clusters[closest_cluster_idx])
            representatives[closest_cluster_idx] = cluster_points.mean(axis=0)

# Prepare the JSON-formatted output
output = []
for i, cluster in enumerate(clusters, start=1):
    cluster_points = np.array(cluster)
    cluster_mean = cluster_points.mean(axis=0).tolist()
    output.append({
        "Cluster": i,
        "Members": [[float(p[0]), float(p[1])] for p in cluster],
        "Mean": [round(cluster_mean[0], 2), round(cluster_mean[1], 2)]
    })

# Format the output exactly as required
formatted_output = "[\n"
for cluster in output:
    formatted_output += f"    {{\n"
    formatted_output += f"        \"Cluster\": {cluster['Cluster']},\n"
    formatted_output += f"        \"Members\": {json.dumps(cluster['Members'])},\n"
    formatted_output += f"        \"Mean\": {json.dumps(cluster['Mean'])}\n"
    formatted_output += f"    }},\n"
formatted_output = formatted_output.rstrip(",\n") + "\n]"  # Remove trailing comma and close the list

# Print the formatted output
print(formatted_output)
```

2.2 Documentation for the code :

2.2.1 Importing Libraries:

- **`numpy`**: A powerful numerical library used for matrix operations and distance calculations.
- **`Json`**: Used to format the output as a JSON string.

2.2.2 Defining Input Data:

The input consists of a list of 2D data points. These points are provided in the form of tuples such as `(x, y)`. The list can be modified to test the algorithm with different datasets.

2.2.3 Setting Parameters:

- **Theta (Threshold)**: This value determines whether a new data point will form a new cluster or join an existing cluster. If the distance between a point and the closest cluster representative exceeds `Theta`, a new cluster is formed.
- **q (Max Clusters)**: The maximum number of clusters allowed. Once this limit is reached, no new clusters will be formed.

2.2.4 Cluster Initialization:

- **`clusters`**: A list that stores the clusters. Each cluster is represented by a list of data points.
- **`representatives`**: A list that stores the centroids (mean) of each cluster.

2.2.5 BSAS Algorithm Logic:

The algorithm processes each data point one by one:

- **First Data Point**: The first point forms the first cluster.
- **Subsequent Data Points**: For each point, the algorithm computes the Euclidean distance to the current cluster centroids. If the minimum distance is greater than `Theta`, a new cluster is created. Otherwise, the point is added to the closest cluster, and the cluster's centroid is updated to the new mean of the points in that cluster.

2.2.6 Preparing Output:

After clustering, the algorithm prepares the output as a list of dictionaries, where each dictionary represents a cluster with:

- **Cluster Number**: An integer indicating the cluster number.
- **Members**: A list of data points in the cluster, represented as lists of floats.

- **Mean**: The mean of the cluster points, represented as a list of two rounded floats.

2.2.7 Formatting the Output:

The output is formatted into a valid JSON structure. Each cluster is printed in a formatted JSON structure that includes the cluster's number, its members, and its mean.

2.2.8 Output Display*:*

The final output is displayed in JSON format, showing the clusters with their corresponding members and centroids.

3 Screenshots of the results when solving the problem


```
[
    {
        "Cluster": 1,
        "Members": [[2.0, 3.0], [2.0, 5.0], [1.0, 4.0], [2.0, 2.0], [3.0, 3.0], [2.0, 4.0]],
        "Mean": [2.0, 3.5]
    },
    {
        "Cluster": 2,
        "Members": [[5.0, 4.0], [6.0, 4.0], [5.0, 3.0]],
        "Mean": [5.33, 3.67]
    },
    {
        "Cluster": 3,
        "Members": [[9.0, 2.0], [8.0, 2.0], [10.0, 2.0], [11.0, 2.0], [10.0, 3.0], [9.0, 1.0]],
        "Mean": [9.5, 2.0]
    }
]
```