Homework Assignment 2 – [30 points]

STAT437 Unsupervised Learning – Fall 2023

Due: Friday, September 8 on Canvas

Questions #1-#3: Refer to the attached Jupyter notebook to complete questions #1-#3 of this assignment.

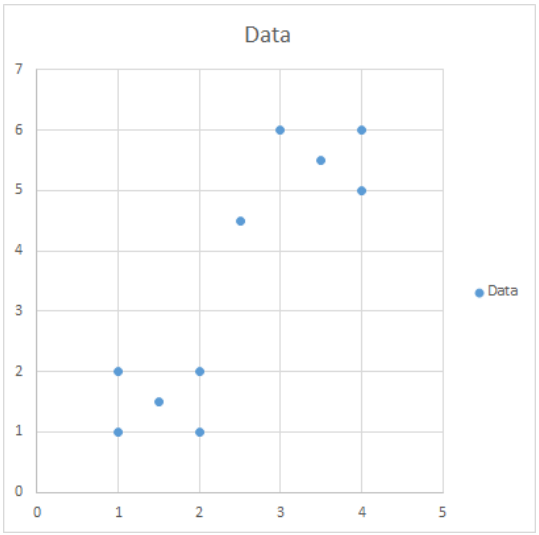
Question #4: [3 pt] Plotted and shown below is a two-dimensional dataset with 10 objects. Your two **initial medoids** have been randomly initialized to be **object 1** and **object 2**. What will be the NEXT two medoids in the k-medoids algorithm? Show your work.

Note 1: In this k-medoids algorithm, we will be using the Euclidean distance metric. A pairwise Euclidean distance matrix for each pair of objects in the dataset is also given below.

Note 2: Many algorithms that we will learn in this class may encounter a "tie" scenario for certain datasets. For instance, when it comes to assigning a particular object to a particular cluster based on the rules stipulated by the *general algorithm*, the object could *technically* be assigned to more than one cluster. In scenarios such these, it's often useful to designate a "tie-breaker" rule. For instance, let's designate the following "tie-breaker rule" for the cluster assignment step of the k-Medoids algorithm.

Tie-Breaker Rule: If an observation is equally close to two or more centroids (medoids), assign it to the centroid with the "lowest index" (ie. "object 3" < "object 4").

	Data	
	x	**y**
Object 1	1	1
Object 2	2	2
Object 3	1	2
Object 4	2	1
Object 5	1.5	1.5
Object 6	2.5	4.5
Object 7	3	6
Object 8	4	5
Object 9	4	6
Object 10	3.5	5.5



Euclidean Distance Between Each Pair of Objects

	Object 1	**Object 2**	**Object 3**	**Object 4**	**Object 5**	**Object 6**	**Object 7**	**Object 8**	**Object 9**	**Object 10**
Object 1	0.00	1.41	1.00	1.00	0.71	3.81	5.39	5.00	5.83	5.15
Object 2	1.41	0.00	1.00	1.00	0.71	2.55	4.12	3.61	4.47	3.81
Object 3	1.00	1.00	0.00	1.41	0.71	2.92	4.47	4.24	5.00	4.30
Object 4	1.00	1.00	1.41	0.00	0.71	3.54	5.10	4.47	5.39	4.74
Object 5	0.71	0.71	0.71	0.71	0.00	3.16	4.74	4.30	5.15	4.47
Object 6	3.81	2.55	2.92	3.54	3.16	0.00	1.58	1.58	2.12	1.41
Object 7	5.39	4.12	4.47	5.10	4.74	1.58	0.00	1.41	1.00	0.71
Object 8	5.00	3.61	4.24	4.47	4.30	1.58	1.41	0.00	1.00	0.71
Object 9	5.83	4.47	5.00	5.39	5.15	2.12	1.00	1.00	0.00	0.71
Object 10	5.15	3.81	4.30	4.74	4.47	1.41	0.71	0.71	0.71	0.00

Question #5 [3 pt]

Designing our Own Clustering Algorithms

Suppose that we'd like to *design ourselves* three different *types* of clustering algorithms.

- Clustering Algorithm A: a clustering algorithm that is designed to detect **density-based clusters**
- Clustering Algorithm B: a clustering algorithm that is designed to detect **contiguity-based clusters**
- Clustering Algorithm C: a clustering algorithm that is designed to detect **prototype-based clusters**

Three Datasets and their *Desired* Clusters

Three datasets are shown in the scatterplots below. The clusters of observations that we'd ideally *like* for the chosen clustering algorithm to find are color-coded.

(One exception to this is dataset 2. We can assume that the observations represented with green diamonds (ie. label 6) actually represent "noise" that we would NOT like for the algorithm to put in an "official cluster" and simply just label as "noise".)

Match the Algorithm to the Dataset

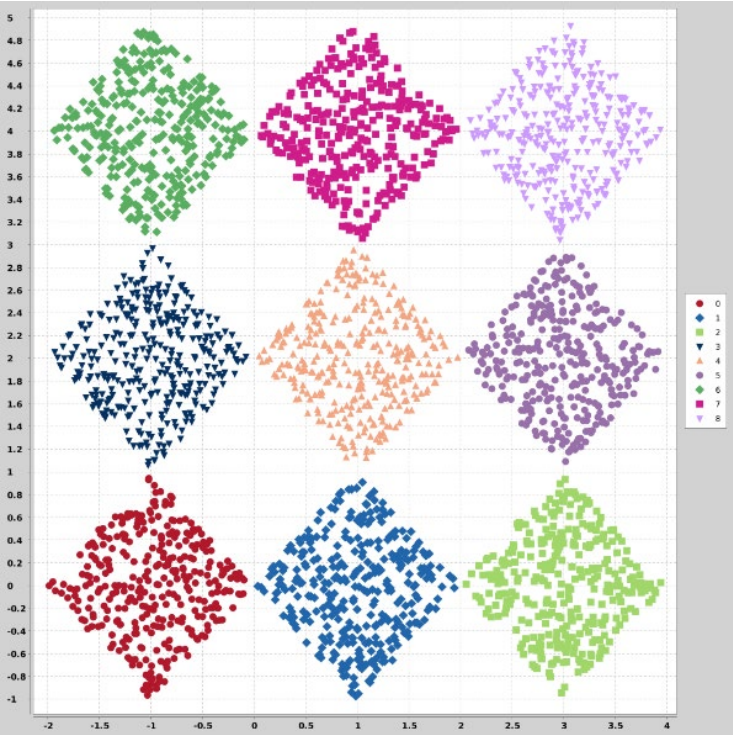
1. Select the algorithm (from the A,B,C above) that would be the best at detecting the cluster that we're looking for in **dataset 1.**
2. Select the algorithm (from the A,B,C above) that would be the best at detecting the cluster that we're looking for in **dataset 2.**
3. Select the algorithm (from the A,B,C above) that would be the best at detecting the cluster that we're looking for in **dataset 3.**

Match the Algorithm to the Dataset

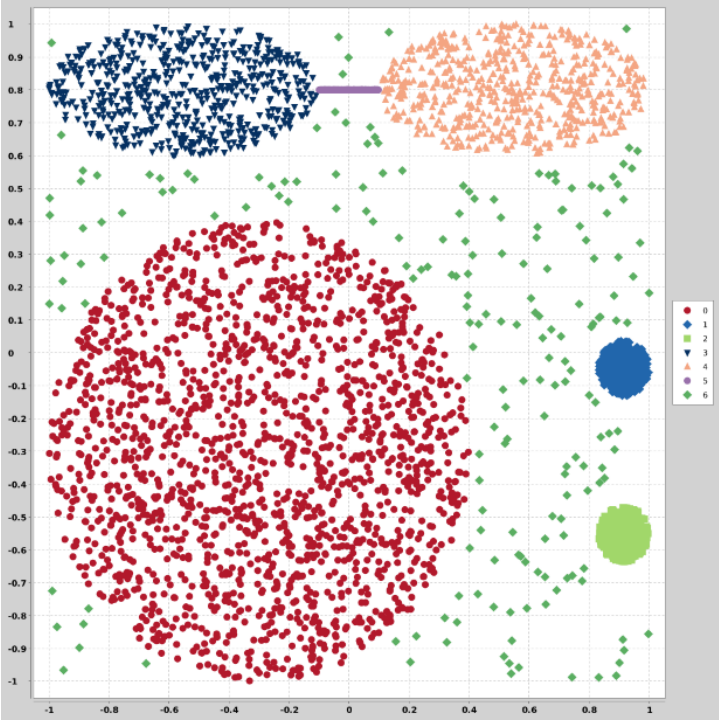
For questions (1, 2, and 3) above *also* answer the following.

- If you selected a clustering algorithm designed to detect **density-based clusters** *for a given dataset*, how might you *specifically* determine if a point should belong to a **density-based cluster** as opposed to being considered as **noise** *in this given dataset?*
- If you selected a clustering algorithm designed to detect **contiguity-based clusters** *for a given dataset*, how might you *specifically* determine if two points are **connected** *in this given dataset? (We can also assume that if A and B connect, and B and C connected, then A and C connect as well).*
- If you selected a clustering algorithm designed to detect **prototype-based clusters** *for a given dataset*, what prototype might you use *for this given dataset?*

Dataset 1



Dataset 2



Dataset 3