Richard Vargas

005859278

CSE 516 Machine Learning

Dr. Zhang

Problem 1

Part 1 – Min-max normalization

Although Weka could be used to preprocess data, I decided to perform this in excel in order to understand the Min-max normalization a little bit better. I used the following equation on both vectors,

$$x_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

This resulted in the following dataset,

ID	A1_NORM	A2_NORM
1	0.08	0.533333
2	0.28	0.8
3	0.04	0.466667
4	0.2	0.833333
5	0.12	0.733333
6	0.4	0.4
7	0.24	0.433333
8	0.16	0.166667
9	0.32	0.266667
10	0.36	0.233333
11	0	0
12	0.48	0.066667
13	1	0.866667
14	0.6	0.9
15	0.44	0.566667
16	0.56	0.633333
17	0.52	1
18	0.8	0.3
19	0.88	0.8
20	0.84	0.5

Part 2 – Iterating K-means algorithm

First Iteration

First, we must assign each point to its closest centroid, since we picked instance 1 ("x") and 20("o") as our centroids, we must calculate the distance of each point to the centroid, square our result, and whichever distance is smaller we will assign to that cluster. We will calculate distance using the following equation,

$$d(c_i, z_j) = \sqrt{((c_{i1} - z_{j1})^2 + (c_{i2} - z_{j2})^2)}$$

Such that,

$$c_i \in C, \qquad z_j \in D$$

Where c is a centroid in our set of K centroids, and z is an instance in our dataset D

Calculate relative to both centroids, and choose minimum value to classify to a cluster,

ID	C1_DIST	C2_DIST	MIN_VAL
1	0	0.760731	0
2	0.333333	0.635295	0.333333
3	0.077746	0.800694	0.077746
4	0.32311	0.721603	0.32311
5	0.203961	0.756865	0.203961
6	0.346667	0.451221	0.346667
7	0.18868	0.603692	0.18868
8	0.375292	0.757305	0.375292
9	0.358763	0.569951	0.358763
10	0.410366	0.5491	0.410366
11	0.5393	0.977548	0.5393
12	0.614636	0.563363	0.563363
13	0.978525	0.400056	0.400056
14	0.636274	0.466476	0.466476
15	0.36154	0.405518	0.36154
16	0.490306	0.310125	0.310125
17	0.641387	0.593633	0.593633
18	0.756865	0.203961	0.203961
19	0.843274	0.302655	0.302655
20	0.760731	0	0

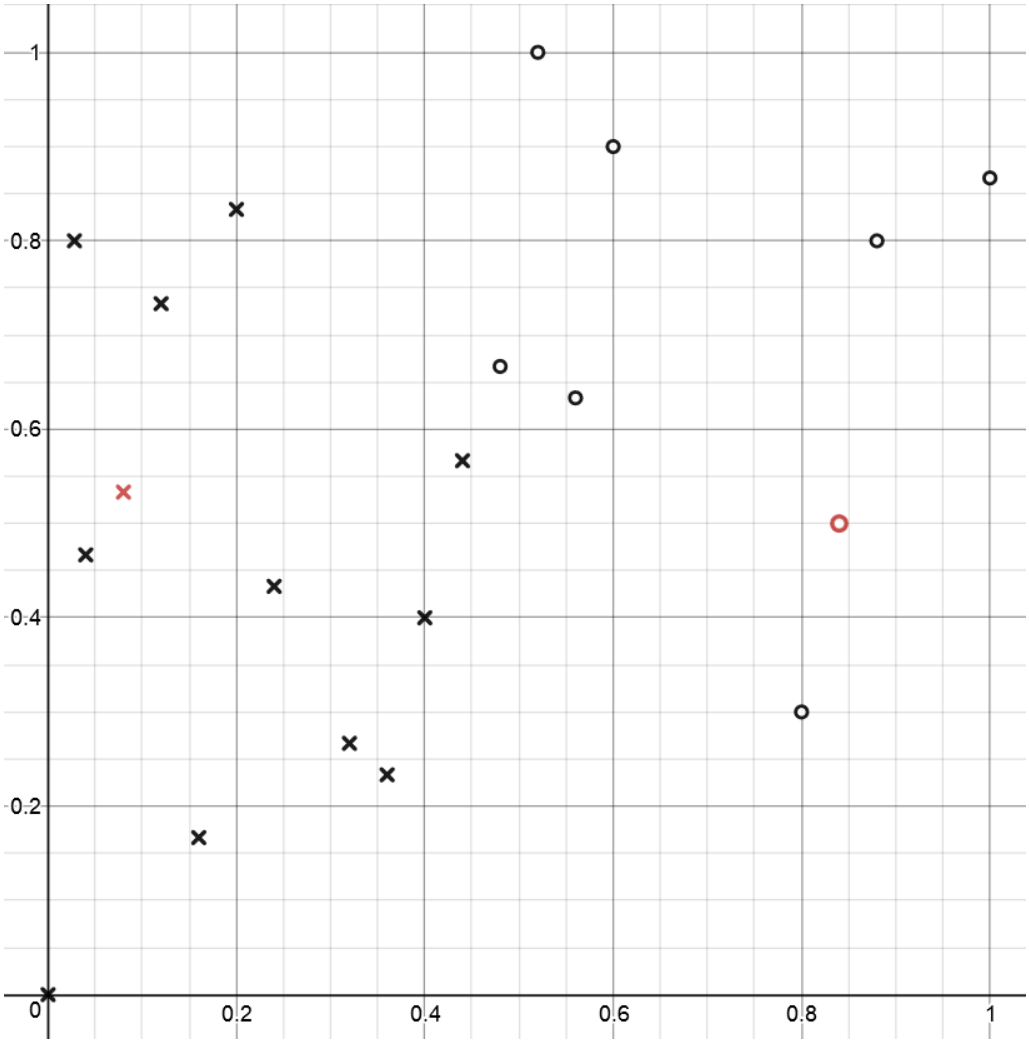
We will now have two clusters,

C1

ID	A1_Norm	A2_Norm
1	0.08	0.533333
2	0.28	0.8
3	0.04	0.466667
4	0.2	0.833333
5	0.12	0.733333
6	0.4	0.4
7	0.24	0.433333
8	0.16	0.166667
9	0.32	0.266667
10	0.36	0.233333
11	0	0
15	0.44	0.566667

C2

ID	A1_Norm	A2_Norm
12	0.48	0.066667
13	1	0.866667
14	0.6	0.9
16	0.56	0.633333
17	0.52	1
18	0.8	0.3
19	0.88	0.8
20	0.84	0.5



For each of these clusters we must find the average and set that to be the new Centroids for the second iteration,

$$c_1 = (0.22, 0.4528)$$

$$c_2 = (0.71, 0.6333)$$

Second Iteration

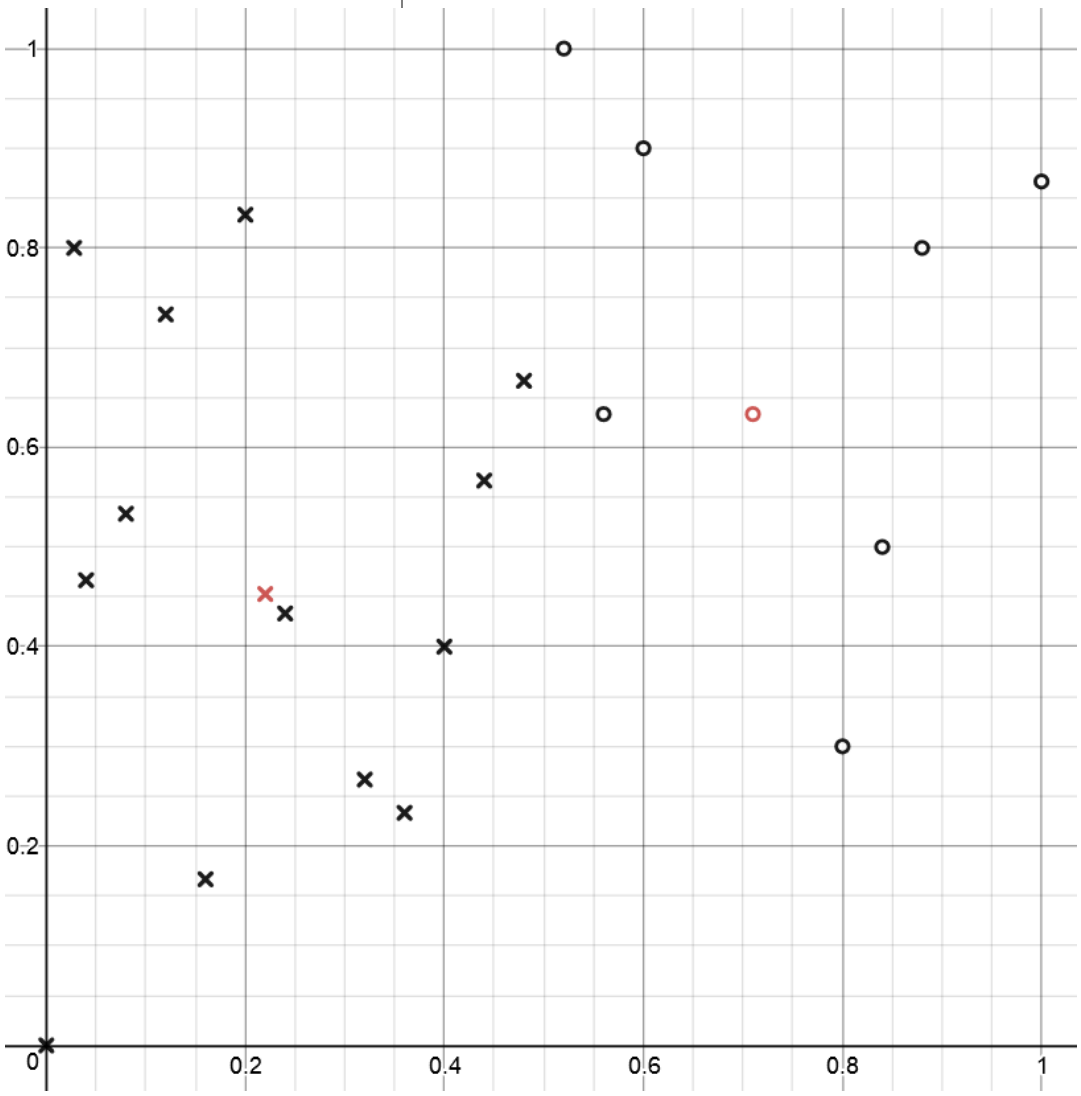
Calculate the distance of each instance from the new centroids using the same formulas as above,

ID	C1_DIST	C2_DIST	MIN_VAL
1	0.161522	0.637887	0.161522
2	0.352368	0.46117	0.352368
3	0.180535	0.690419	0.180535
4	0.381081	0.547814	0.381081
5	0.297845	0.598415	0.297845
6	0.187578	0.388001	0.187578
7	0.027894	0.510784	0.027894
8	0.292335	0.721303	0.292335
9	0.211276	0.535298	0.211276
10	0.2603	0.531507	0.2603
11	0.503396	0.951426	0.503396
12	0.465491	0.611564	0.465491
13	0.883009	0.372216	0.372216
14	0.586863	0.288463	0.288463
15	0.247731	0.278109	0.247731
16	0.384968	0.15	0.15
17	0.624061	0.41297	0.41297
18	0.599784	0.34527	0.34527
19	0.745764	0.238071	0.238071
20	0.621796	0.18622	0.18622

Gives us our new clusters

	C1	
ID	A1_Norm	A2_Norm
1	0.08	0.533333
2	0.28	0.8
3	0.04	0.466667
4	0.2	0.833333
5	0.12	0.733333
6	0.4	0.4
7	0.24	0.433333
8	0.16	0.166667
9	0.32	0.266667
10	0.36	0.233333
11	0	0
12	0.48	0.066667
15	0.44	0.566667

C2		
ID	A1_Norm	A2_Norm
13	1	0.866667
14	0.6	0.9
16	0.56	0.633333
17	0.52	1
18	0.8	0.3
19	0.88	0.8
20	0.84	0.5



New centroids based on these averaged instances

$$c_1 = (0.24, 0.4231)$$

$$c_2 = (0.7429, 0.7143)$$

Third Iteration -

Second Iteration

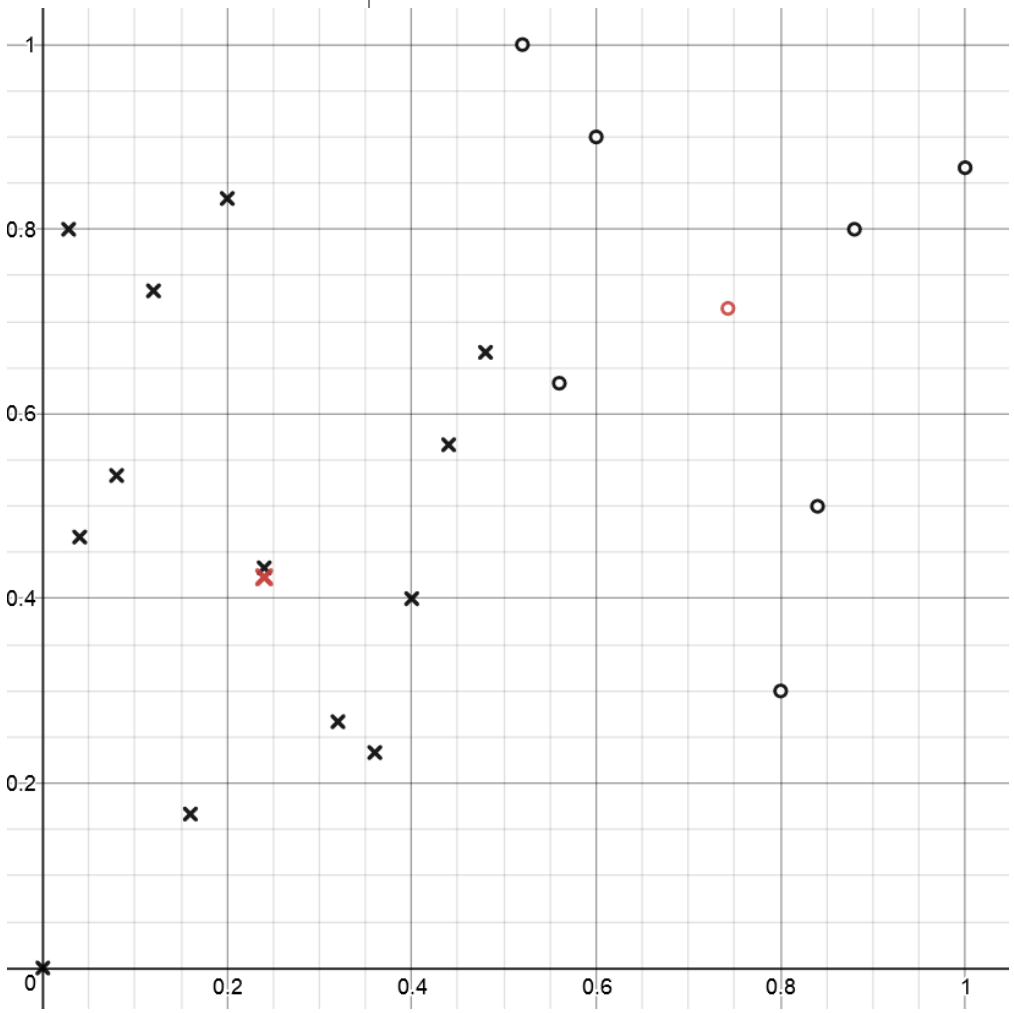
Calculate the distance of each instance from the new centroids using the same formulas as above,

ID	C1_DIST	C2_DIST	MIN_VAL
1	0.19431	0.687112	0.19431
2	0.37904	0.470727	0.37904
3	0.204695	0.7452	0.204695
4	0.412202	0.555757	0.412202
5	0.332655	0.623148	0.332655
6	0.161656	0.465109	0.161656
7	0.010256	0.57602	0.010256
8	0.2686	0.799756	0.2686
9	0.175682	0.615769	0.175682
10	0.224505	0.614731	0.224505
11	0.486409	1.030554	0.486409
12	0.429684	0.698931	0.429684
13	0.879984	0.298902	0.298902
14	0.597541	0.234303	0.234303
15	0.246207	0.336918	0.246207
16	0.382894	0.199975	0.199975
17	0.64128	0.362351	0.362351
18	0.573365	0.418208	0.418208
19	0.742746	0.161725	0.161725
20	0.604911	0.235277	0.235277

Which gives us the following clusters,

	C1	
ID	A1_Norm	A2_Norm
1	0.08	0.533333
2	0.28	0.8
3	0.04	0.466667
4	0.2	0.833333
5	0.12	0.733333
6	0.4	0.4
7	0.24	0.433333
8	0.16	0.166667
9	0.32	0.266667
10	0.36	0.233333
11	0	0
12	0.48	0.066667
15	0.44	0.566667

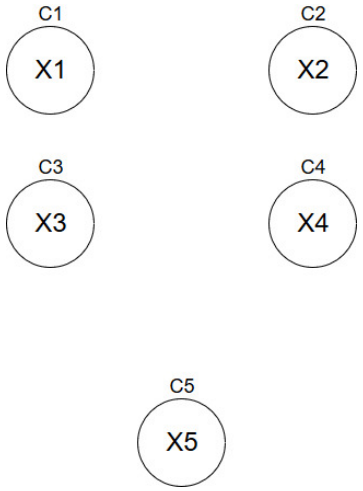
	C2	
ID	A1_Norm	A2_Norm
13	1	0.866667
14	0.6	0.9
16	0.56	0.633333
17	0.52	1
18	0.8	0.3
19	0.88	0.8
20	0.84	0.5



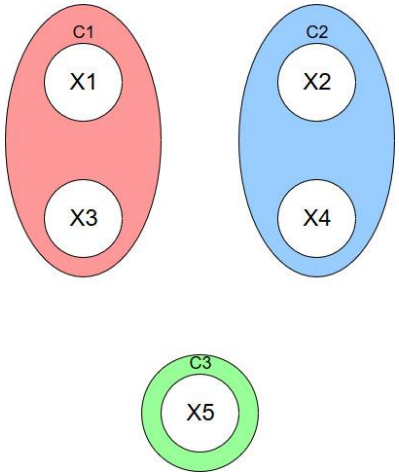
Since our clusters stayed the same, our algorithm would exit here for the fitting portion. We would end up with the same centroids as the previous iteration if we attempted to update them again.

Problem 2

Agglomerative Hierarchical Clustering starts by having each point as its own cluster. In this example we have the following cluster sets C1={X1}, C2={X2}, C3={X3}, C4={X4}, and C5={X5}.



From there it starts to create new clusters, based on instances being closest together. The new cluster sets will be C1={X1,X3}, C2={X2,X4}, and C3={X5}

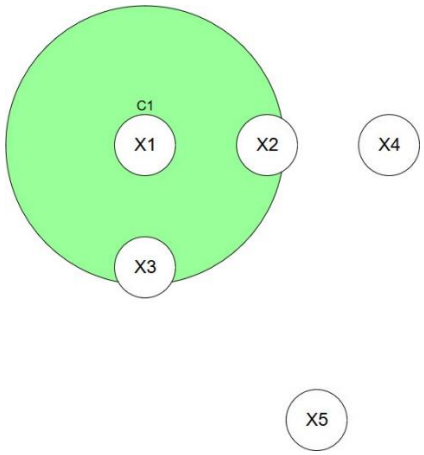


This process can continue until a desired K-clusters are met.

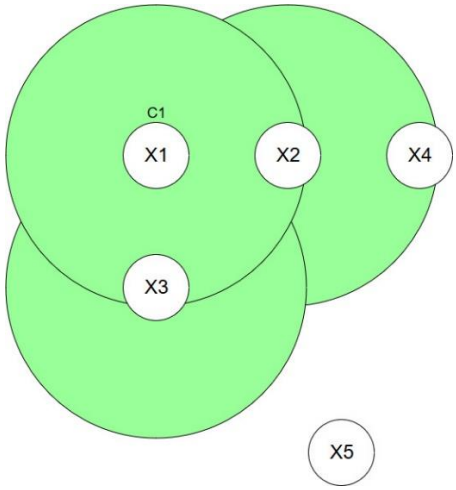
Problem 3

Density Based Clustering starts by setting a value ε , which the radial distance from some point x is the ε-neighborhood. If a point has a certain neighbor threshold (For example, 3 within its ε-neighborhood) then it becomes part of the cluster. In this algorithm there are three main kinds of points (or instances). First there are those which are core points, implying they are in the cluster and have the minimum number of points required and are directly reachable from our starting point. Then there are border points, which are still part of the cluster by being directly-reachable, however do not have the minimum number of points to be core points. Finally, there are noise points which are not a core point and are not directly reachable. These points are thought of similarly to outliers.

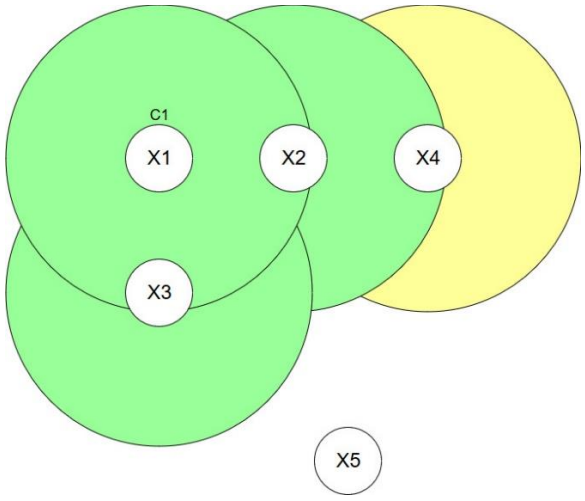
To start the algorithm, an arbitrary point is picked, and nearby instances are checked for. If a minimum number of instances exist in the ε-neighborhood then a cluster is started. By example, we check X1 first, since the minimum number of instances in the ε-neighborhood is 3, we have a core point which starts a cluster, and add all instances in the ε-neighborhood to the cluster set. C1={X1,X2,X3}



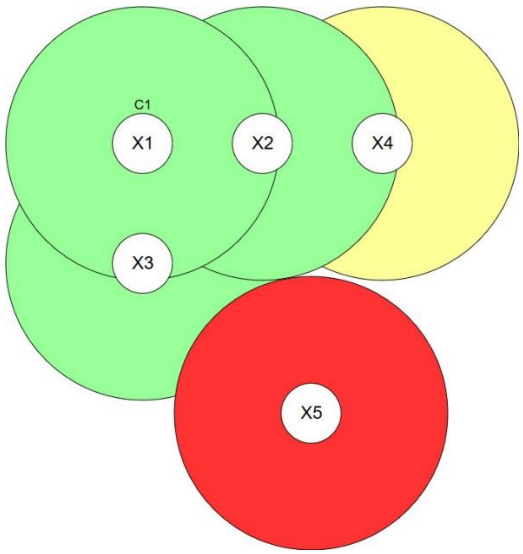
Now that we have other instances in our C1 set, we want to check the other unvisited elements X2 and X3, adding them to their ε-neighborhoods to the cluster we have started (if they have enough minimum instances in their respective ε-neighborhoods).



Since X2 is a core point, any instances in its ε-neighborhood are considered part of the cluster, and thus added to the C1 Set. Therefore C1 = {X1,X2,X3,X4}. Since instance X4 is an unvisited element we must inspect it with our algorithm. Since X4 only has 2 instances in its ε-neighborhood we cannot consider it a core point, thus it is a border point and still part our cluster.



Our final instance in our data is X5, which is not reachable by any of the core points for our C1 cluster and does not have a minimum number of instances in the ε-neighborhood to start its own cluster, so it is considered a noise point.



If another point were to be added within the ε-neighborhood of X4, then we could then change the status of X4 from a border point to a core point. On the same note, if we add 2 points to the ε-neighborhood of X5, then X5 would become a cluster and no longer a noise point.