Thomas Pollard
Programming 3

Programming assignment 3 report

Programming assignment 3 is a k-means and fuzzy c-means implementation. They both used the same data set and were intended to see how unsupervised learning algorithms do on clustering data points.

Assignment #1: K-means

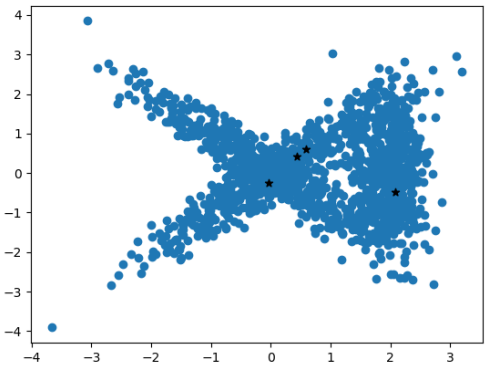
For the k-means algorithm I first picked K random points in the data to be the initial centroid locations where K is the target number of clusters. Then I classified each data point to its closest cluster centroid using Euclidean distance. After classifying each data point, I updated each centroids location to be the mean of each data point in that cluster. These expectation and maximization steps were repeated until the classifications between each repetition did not change. Last, I calculated the mean squared error for each data point to its corresponding cluster and then summed them up. I ran this algorithm R times where R is a hyperparameter and found the iteration that resulted in the least error. Table 1 below shows the K-means algorithm run with R = 5 and various cluster counts for K and the calculated error for each of them.

K (number of clusters)	Mean Squared Error
1	4058.27
2	2228.62
3	1539.26
4	1103.51
5	773.13
6	626.96

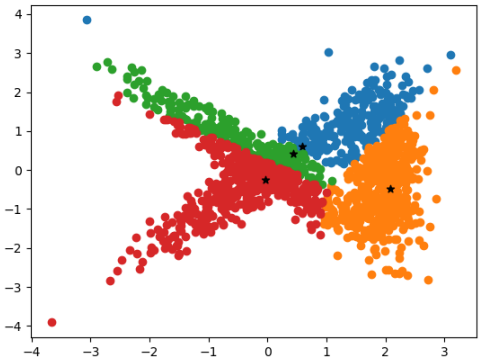
Table 1: The K-means algorithm run for various cluster counts and the errors for the best of 5 iterations

Table 2 shows some iterations of the algorithm for 4 clusters on the fourth random initialization which turned out to be the optimum configuration.

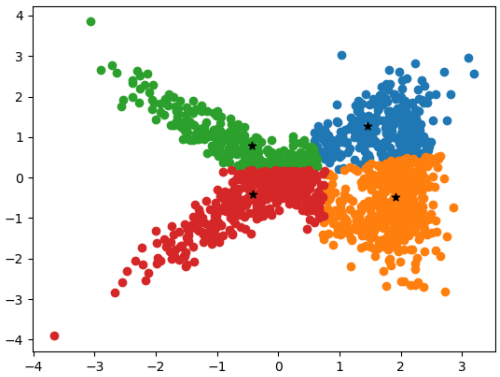
Thomas Pollard
Programming 3



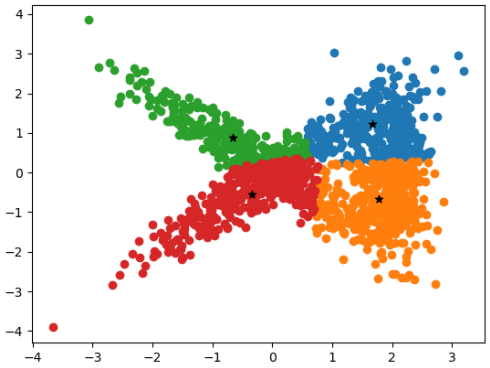
K = 4, r = 3, iteration 0



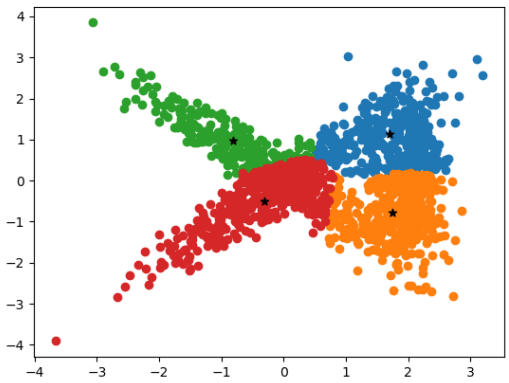
K = 4, r = 3, iteration 1



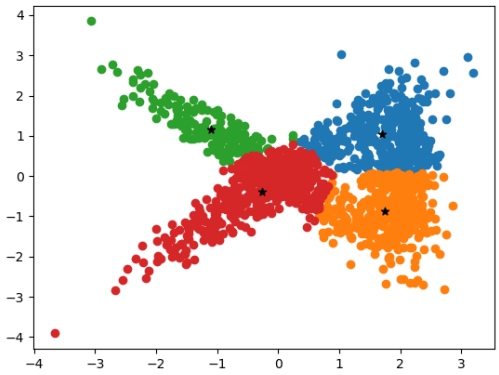
K = 4, r = 3, iteration 2



K = 4, r = 3, iteration 3



K = 4, r = 3, iteration 4



K = 4, r = 3, iteration 5

Table 2: Plots from the 4-cluster k-means on the data set of the optimum configuration

Thomas Pollard
Programming 3

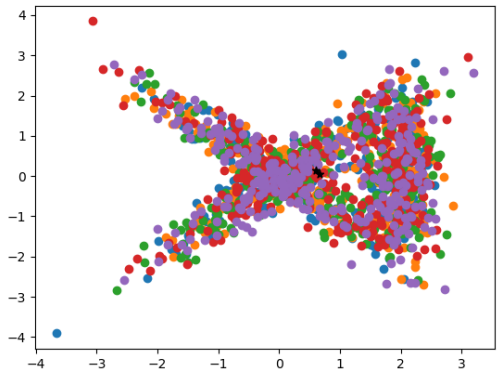
Assignment #2: Fuzzy c-means

For the fuzzy c-means algorithm I used a weight matrix to quantify the memberships grades of each point to each cluster. Then I calculated the location of the centroid of each cluster by dividing the weight times each data point divided by the total weights each to M power where M is a hyperparameter representing the fuzzifier constant. Then I recalculated the weights of each data point with respect to each data point. I repeated the steps until the algorithm converged and the classifications of the data points did not change. Last, I calculated the mean squared error for each data point to its corresponding cluster and then summed them up. I ran this algorithm R times where R is a hyperparameter and found the iteration that resulted in the least error. Table 2 below shows the fuzzy c-means algorithm run with R = 5 and various cluster counts for K and the calculated error for each of them. M is set to 5 for all runs as well.

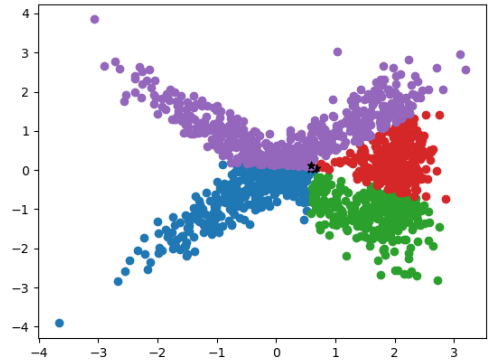
K (number of clusters)	Mean Squared Error
1	4058.27
2	2354.34
3	2124.68
4	1474.76
5	1227.02
6	1129.00

Table 3: The fuzzy c-means algorithm run for various cluster counts and the errors for the best of 5 iterations with M = 5

Table 4 shows some iterations of the algorithm for 4 clusters on the first random initialization



K = 5, r = 0, iteration 1, M = 5



K = 5, r = 0, iteration 2, M = 5

Thomas Pollard
Programming 3

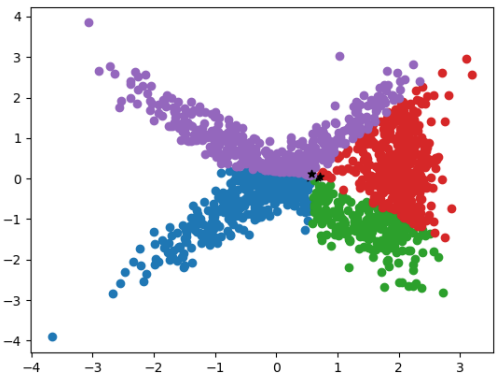
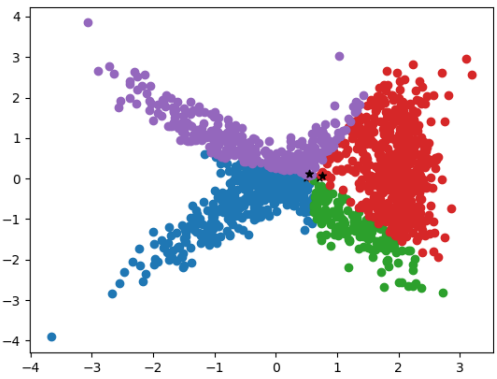
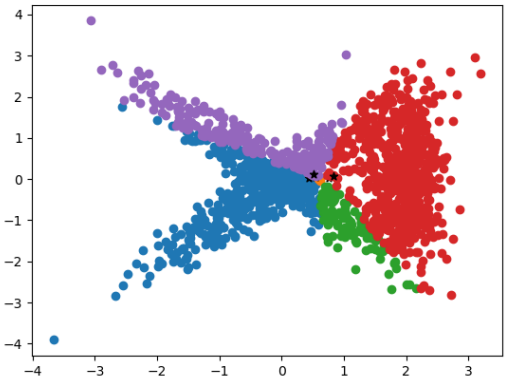
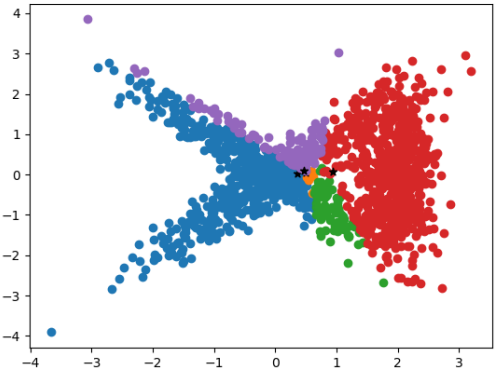
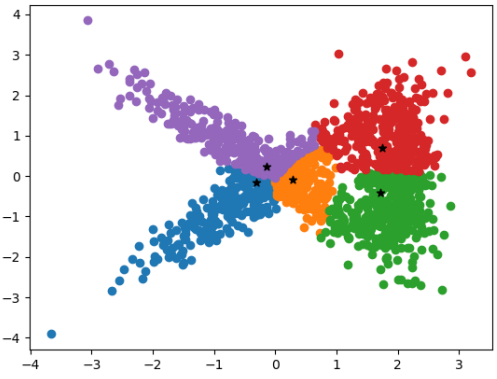
 K = 5, r = 0, iteration 3, M = 5	 K = 5, r = 0, iteration 4, M = 5
 K = 5, r = 0, iteration 5, M = 5	 K = 5, r = 0, iteration 6, M = 5
Skipping some iterations	 K = 5, r = 0, iteration 30, M = 5

Table 4: Plots from the 5-cluster k-means on the data set of the first initialization and M = 5