Jeremy Palmerio
i6239656
PRA3021

Topics In Scientific Computing Week 2

Introduction

This week’s assignments were both harder and easier than last week's assignments. On one hand I was sick which made the theory behind the assignments harder to grasp alone, but on the other hand the tasks were more attractive to me. Some things that were apparently discussed in class, I had to come up with on my own which nonetheless gave me a deeper intuition for this clustering exercise.

Square distance function

The first of my problems was the square distance function. I spent a lot of time trying to figure out how to implement the given function, so much so that I opted instead to rewrite it myself. In doing so I managed to optimize a little bit more by the ability to calculate the distance from not only one center but an array of centers. This meant that all the other functions where I use this function are all a bit more optimized. In the end it's just a question of looping over multiple centers once in the square distance function rather than looping everytime I would use the function.

MyKmeans function

This function took me a long time to write and get working as I wanted it. Originally It worked by inputting the data set and a matrix of centers and it would return the local optimal location for those centers. The hardest part of this task was to somehow extract all the points in a similar cluster in an efficient way. It was while looking up the find() function that I stumbled upon the X(X==p) which creates a subset of X of all elements X=p. Thus for each index I could create a subset with all the points who are closest to that cluster. This method seemed better than trying to use structure or something else. After achieving good results with this function I attempted to make another function which uses the Kmeans function for a huge number of random starting centers and calculates the variance for each of them. The final output would be the collection of clusters with the smallest total variance. Then, just recently I merged the two functions into the MyKmeans which produced a more realistic clustering as it could try over hundreds of local optimums and pick the best. To automate how many iterations sthe clustering algorithm performs I used a quick trick of using a while loop and looking at the difference between the current iteration and the previous one. If the sum of differences is smaller than a tolerance the iterations are over and the centers have stopped moving.I haven’t had the time to optimize this newer function. Even though it is stable, I'm sure there are some redundancies that can be

Jeremy Palmerio
i6239656
PRA3021

eliminated. For more detailed information, see the comments in the .m file. To quantify how good it performed I used the total summed variance to pick the best iteration. Furthermore, I used matlab's built in kmean function to see how it compared and depending on the iteration, sometimes the clusters exactly overlap, sometimes one cluster perfectly replaces two others in the middle. EIther way the result provided shows that my function is comparable in result to the matlab one. Below is a figure comparing the two:

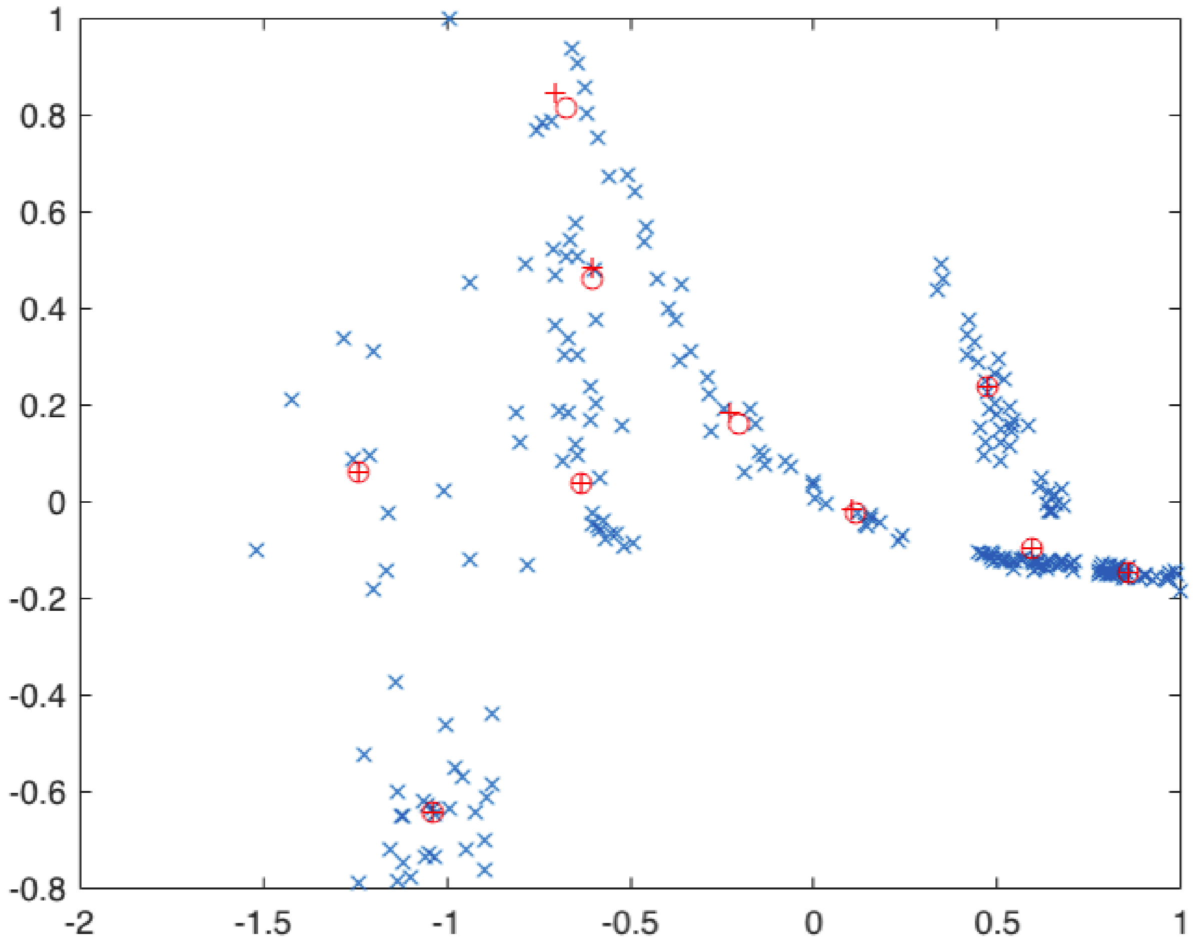

Figure 1: comparing my kmeans function to the matlab built in one. The red circles are my cluster centers, and the crosses and matlab's results.

Jeremy Palmerio
i6239656
PRA3021

MyKnn function

This function was dramatically simpler than the previous one. The important parts of this code were the use of the mink function which allows the extracting the kth minimum values from a matrix, which was paramount for getting the k closest points to a center using the square distance function. Then the next important step was using the mode function which collapses the array with all the closest neighbors to only the label with the most number of neighbors. Then to test the accuracy of the algorithm, I created a quick script which uses the myknn algorithm and inputs the whole data set as test points and then compares the label the function assigned versus the label the original mykmeans algorithm labeled them as. I then look at the average efficiency over 50 trials for different k (numbers of closest neighbors). The results are in the table below and the function can be seen in the zip file. For this data set it seems that the higher value for k doesn't directly imply a better algorithm, I suppose it's because this isn't the best data set to linearly cluster.

Neighbors k	3	5	10
Efficiency	0.9907	0.9930	0.9884

Table 1 : Efficiency of myknn function for different k. A higher k doesn't always mean a more accurate result.