# Programming Assignment #4 : Unsupervised Learning – Fuzzy C-Means Clustering

1. This assignment document comes as a part of an *assignment package* which contains, among other items, an excel file called *Data Sets*. The sheet Data Set 5 is the data set that you are supposed to work upon.

2. The mentioned data set is basically 2-D data implicitly arranged as clusters (in some sense) and also plotted for visualization. 2-D implies n = 2. Refer to the terminology of the attached power-point slides. The total number of points is 1000.

3. You are first to extract every 4 out of 5 data in a consistent interspersed manner (i.e. frequency of 4 out of 5 will be maintained for every block of 5 consistently) and save it in another file which will serve as your *mining data* *file*. Thus N = 800. The balance 1 out of 5 data you will save in a second file which you can call the *classification data file*.

4. Use the algorithm defined in slide nos 13 & 14 of the power-point file (use the current version and not those provided previously) to generate your own program (in any language – Python preferable due to the need for downstream graphics) for generating clusters using the data contained in the *mining data file*. In the codification process, take m = 2, A = I (Identity matrix which then makes A redundant in all calculations), ε = 0.001 (you may change this with one less or more zero).

5. Over and above the basic algorithm, you are to additionally program to perform the following tasks:

5.1 the process must be put in an outer loop which runs over different values of the parameter *c* representing the number of clusters, *1 < c < 11*, and the objective function *Jc* (eq. 6 of slides) evaluated for each cluster. Also a ratio *Rc* given as  should be evaluated for all *c* from 3 to 9. At the end of the outer loop, the program should compare all the *Rc* and find out at what value of *c* it minimizes. And finally, it should repeat the calculation of clustering (as in step 4) for the value of *c* identified as the minimum. All this should be done automatically in your code without manual intervention

5.2 plot in excel (less preferably in Python) the variation of *Jc*against *c*, and the number of iterations for convergence (ε should not change; the number of iterations will vary across c) against *c* in the same plot. Refer to the file “Plot specification by example” provided in the package (for a different data set) and you should do it similarly. Note this step if done in excel will be a part of post-processing activity and cannot be integrated with the rest of your code

5.3 at the end of step 5.1 you will have the final partition matrix U for the best clustering. This U will have 800 columns and c number of rows, with the value μ of each element between 0 and 1. You will create another data file C (a text file) with 800 rows and 3 columns, where the first two columns represent the x- and y- values for the corresponding data point (represented by the row), *and the third value contains the cluster identity number (from 0 to c-1) for which μ maximizes in the partition matrix U, for the column in the matrix (U) that represents that data sample*. In a physical sense, the italicized part above simply means that the third column of data file C contains the id-number of the cluster at which the data sample has the highest degree of belonging, compared to other clusters

5.4 Finally, you will plot the data points again in a 2-D plot using file C where each of the different values in the third column is mapped to a specific colour, hence all points in any cluster appear in the same colour. For an example, see the file “Final cluster illustration – by example”. Note that activities 5.3 and 5.4 can be performed within the same main program you are writing, *without intermediate manual intervention*. In that case you need not create a separate text file ‘C’ which you write and read into/from hard-disk, instead just use a 2-D list (array)

6. While steps 4 and 5 take care of clustering, the operational and practical downstream objective is that every new point that arrives should be inserted into a new cluster. This is the classification problem. To do this, you will need to write a separate program that reads the cluster centroid information from a file – which has to be written as a final output from the original clustering program. This is basically the file containing the vector *V* referred to in eqns. (8), (11) and again (A) in the slides. Next, this program will read all the 200 points that you have stored in your *classification data file*, and treat each of these new points as a vector zk that you shall use in eq. (B) of the slides to extract a distance measure *Dik* to each cluster centroid *i*. Then use these *D­ik* in eq. (C) to extract the corresponding μik, and evaluate the value of *i* (i.e. cluster id-number) at which μik maximizes. The new point then belongs to that cluster. Then repeat the code sections generated under 5.3 and 5.4 to plot the new points in their classified clusters. Check if the geometrical shapes of the clusters thus formed while performing classification are similar to the original shapes you had obtained while doing the clustering from your *mining data set*.

Submission:

While you would have carried out your internal development and testing of the program on the Data Set 5, you shall be carrying out your formal submissions on **Data Set 2**. In this data set there are 620 points. Carry out the interspersed approach of splitting into training and testing data sets at 4:1 ratio for the first 600 points only. The last 20 points will all go into the testing data set. So the total number in the training set is 480, while in the testing set is (120 +20) = 140 points.

You shall submit the following in a folder; the naming specifics of the folder will be provided to you before the time of submission. Any variation from these specifics will lose the assignment and you will get zero marks. The logistics of submission will be provided to you two days before the submission date. The matter for submission is a word/pdf:

1. A plot which has the number of clusters on the x-axis, and the value of Objective function and the number of iterations for convergence on the two y-axes
2. A plot in 2-D showing the locations of the sample points in the x-y frame with symbols, with **each symbol showing a specific colour according to the cluster to which the sample belongs**. This implies each cluster is given a unique colour. The belonging of each sample to a specific cluster is based on the largest value of the variable *degree of belonging* to each cluster. Importantly, this cluster plot shall be made for only the 140 points in the classification (testing) data set.

Each submission body should also contain your name and (again) HT number. Further, apart from the illustrations contained in the word / pdf, you MUST also submit the Python code with all relevant data in the same folder for the program to execute. A run of this code should generate exactly the same output that you are providing with your submission.