Data Mining project report

Lorenzo Paciotti

August 7, 2015

Abstract

The goal of this project was to expand and test the capabilities of my DEFC evolutionary clustering algorithm. DEFC is an implementation of Differential Evolution as proposed by Reiner and Storn. DE is used here to optimize the fuzzy c-means objective function, therefore to accomplish clustering of data points in a n-dimensional space.

1 Fuzzy C-Means Clustering

Fuzzy C-means clustering is a clustering algorithm which thoroughly resembles K-means. FCM differs from KM in its definition of membership of a point to a cluster. While k-means follows a crisp partitioning of the input when assigning memberships, C-means allows for each point to be part to a certain degree of all the clusters. This is done following this restriction to the possible values found in the membership matrix $U_{c\times n}$:

$$\sum_{j=1}^{N} u_{ij} = 1, \forall i = 1, ..., C$$

FCM objective function is much like K-means' with the addition of a weight (the aforementioned degree of membership) named μ to the power of m (the *fuzziness* factor) applied to the intra-cluster distances, meaning that distant points from the cluster have reduced influence on its compactness, i.e. are less similar to the centroid of that particular cluster.

$$\mathbf{J}_m = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^m ||\mathbf{v}_i - \mathbf{x}_j||^2$$

2 Xie-Beni index of medoid-based clustering quality

In this project, evaluation of clustering quality is made using a slightly modified Xie-Beni index which was created accounting for the fuzzy logic used in Fuzzy C-means.
Shown here is the standard Xie-Beni index:

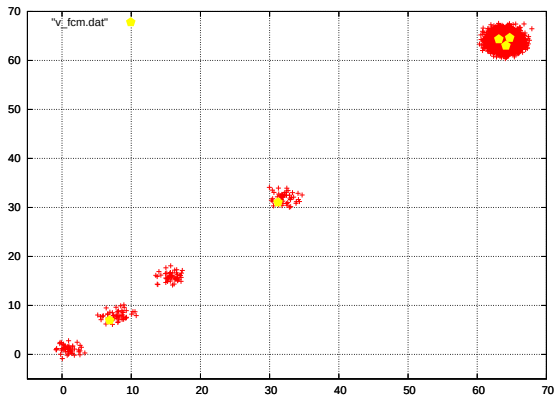
$$XB(U, V; X) = \frac{\sigma(U, V; X)}{n \times min\_sep(V)}$$

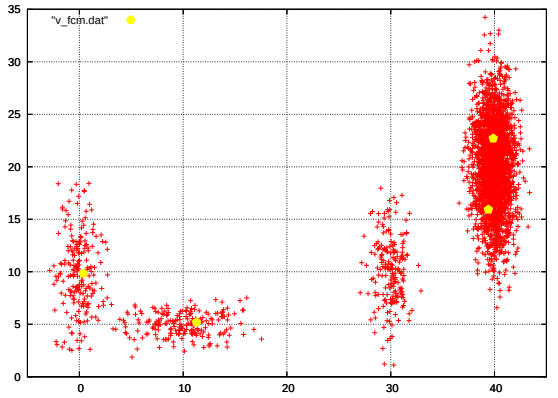
$$\sigma(U, V; X) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{j,i}^{m} ||v_j - x_i||^2$$

$$min\_sep(V) = \min_{i \neq j} ||v_i - v_j||^2$$

3 Issues with Fuzzy C-Means Clustering

When it comes to finding cluster of well separated, globularly arranged and balanced data FCM shows exceptional speed and performance when compared to a non specific algorithm like DEFC. The main issues with FCM come to show when the data is not equally distributed in the space. Highly unbalanced clusters can easily fool the algorithm into positioning two centroids very close to each other, while neglecting smaller, but not less important, clusters. Furthermore, the shape of clusters of data, when not globular, does compromise FCM functionality.
This is caused by the objective function which consists of sums of distances between each center and all the points in the input. When minimizing the distances sum, having many close points will induce the algorithm in positioning more than one cluster center in the same spot, which reduces the objective function value the most.





4 Optimizing with Differential Evolution

Differential Evolution is an evolutionary optimization strategy. It uses the principles of evolutionary computation in order to optimize a real-valued objective function.

In DEFC population is made of individuals, each one with its particular disposition of cluster centroids, randomly initialized in space then left to evolve.
Differential evolution generates a population of mutating individuals from the existing population of solutions by the process of *mutation* at each of its iterations (named generation)

$$V_{mutant}(c) \longleftarrow V_{p3}(c) + f * (V_{p1}(c) - V_{p2}(c))$$

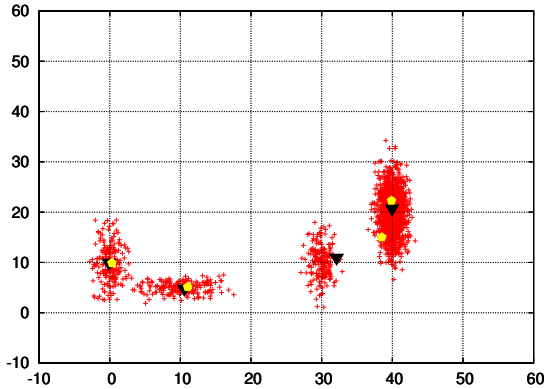
Vectors which make up the V centroid matrix of a *mutant* individual are constructed starting from three different individuals of the population *p1,p2,p3*. The strength of the mutation is specified by the parameter *f*, which is part of the genome of the individuals and is selected along with the best fit individuals (cfr. jDE)

After the mutation step, chances are the mutant inherits some of the characteristics of the target individual, in the so-called *crossover* process. The emphcrossover ratio (CR) parameter sets how likely it is for the crossover between the target and the mutant individual to happen. Like the f parameter, also CR parameter is randomly regenerated at each birth of a mutant and brought to the new generation if it contributed successfully in creating a better fit individual.

Final phase of a generation is the *selection* stage. In the selection stage the target individual and the trial individual (name of which was mutant until the crossover step) compete for survival. Fitness values of both are calculated against the objective function, whichever individual sports the lowest value of fitness will take the other's place in the population.
The algorithm terminates after a preset number of generations, the best individual's values are returned as result.

5 Adapting to variable density clusters

DEFC presented the same problems, although less prominently, as FCM when clustering unbalanced and/or non-globular data (DEFC centroids are shown in black, FCM centroids in yellow).



At this point, it was evident that the objective function did not fit this problem very well, so I made a modification to it which adds a reduced scale to the points of very large, very compact clusters.

When the fitness value of an individual is being calculated, a counter is increased each time a point of the input is found to be belonging to the current centroid more than the *counting threshold* value. This is to represent the fact that we consider the point to be more similar to the current centroid than to the others. When the computation of a centroid ends, the counter value is compared to the *cardinality threshold* value which had been generated earlier (more on this in the next paragraph) and if it is greater than the latter the *sigma weight* is applied to the sigma value of the current cluster.

In the initialization phase every individual of the original population is provided with its own randomly generated counting threshold and cardinality threshold values. Then its fitness value is computed.
The same happens when a new mutant is generated, so there are many different randomly generated values which will influence, for better or for worse, the fitness values of the population during evolution. Eventually, the best values will survive along with the host individual as the best fit to solve the problem.
Ranges of initialization are [0.5,1] for the counting threshold, [0,n/c] for the cardinality threshold.

5.1 Calculation of sigma weight value

Sigma weight value is calculated for each cluster as the ratio of the number of highly belonging elements over the total number of elements in the dataset.

$$\sigma_W = 1 - \left(\frac{|V_c|}{n} \right)$$

Where $|V_c|$ is the cardinality of centroid c, n is the cardinality of the input.

5.2 Adjusted Xie-Beni values

As the Xie-Beni value is based on the ratio of SSE over minimum separation, the modification that was made implies that the found Xie-Beni value is different from the standard one. Both values, standard and adjusted are presented in the experimental results section as a

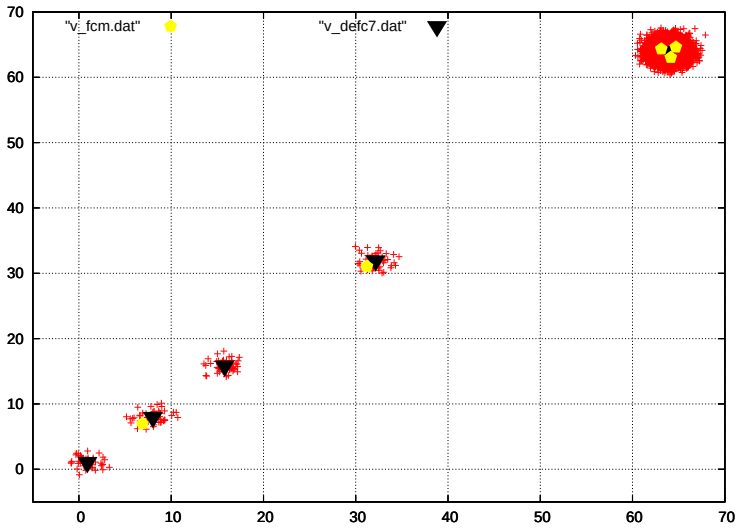
6 Experimental results

Following are experimental results on artificially generated datasets, for which means of clusters are known to the author of this work, this allowed to verify the correctness of the Xie-Beni measure for clustering quality.

All results are verified to be the same after 10 runs, DEFC and FCM always converge to the same minimum for these particular datasets.

6.1 Dataset 1

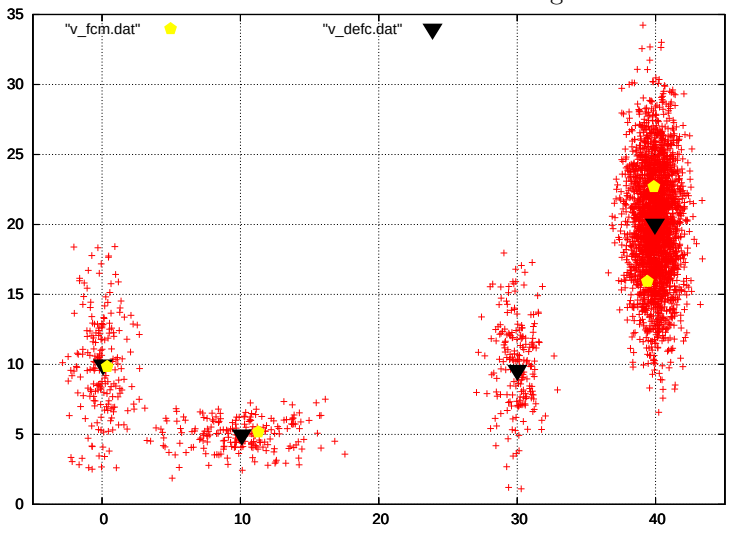
10000 points in 5 clusters, the last cluster holds 9600 points while the first four hold only 100 points each.



Algorithm	adj XB Value	XB Value	# of fitness evals	Counting T.	Cardinality T.
DEFC	0.001310	0.020399	50000	0.613474	15
FCM		0.578815			

6.2 Dataset 2

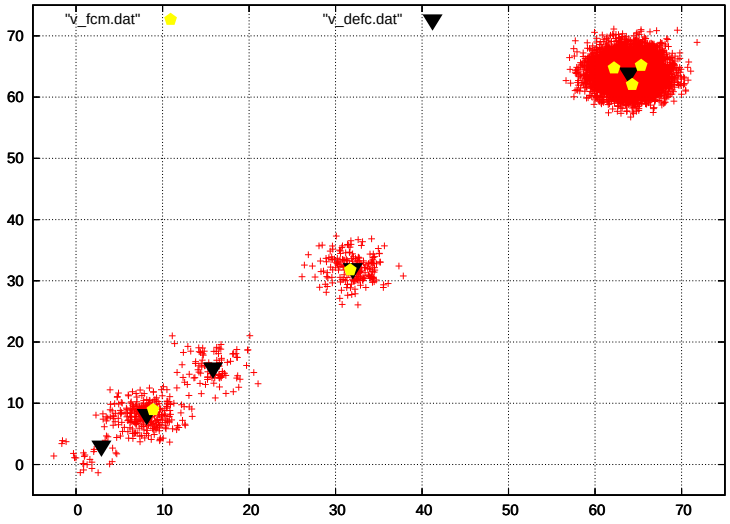
Four non globular clusters. Fourth centroid is much denser than the others. When compared to globular-shaped clusters datasets this kind of data arrangement causes much more trouble to a centroid-based algorithm.



Algorithm	adj XB Value	XB Value	# of fitness evals	Counting T.	Cardinality T.
DEFC	0.064777	0.094680	50000	0.524960	97
FCM		0.210802			

Table 1: results on a 3200 points dataset

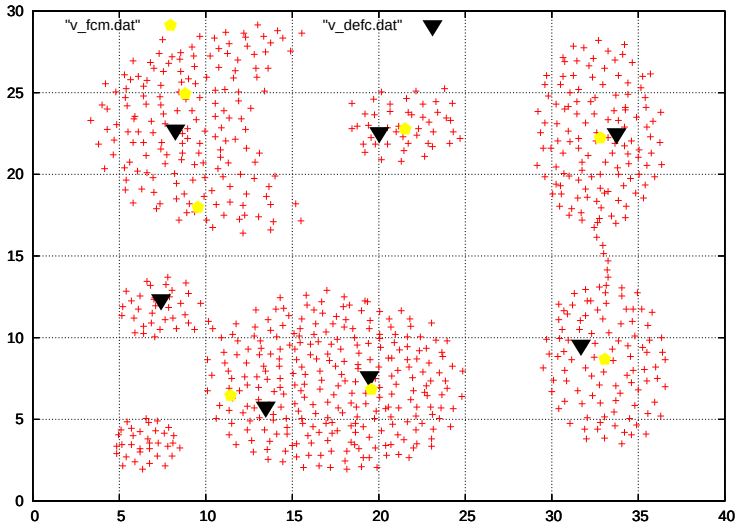
6.3 Dataset 3



Algorithm	adj XB Value	XB Value	# of fitness evals	Counting T.	Cardinality T.
DEFC	0.025377	0.133782	50000	0.654804	28
FCM		0.352798			

6.4 Dataset 4

From http://cs.joensuu.fi/sipu/datasets/



Algorithm	adj XB Value	XB Value	# of fitness evals	Counting T.	Cardinality T.
DEFC	0.261704	0.157123	50000	0.560025	30
FCM		0.182673			

7 Source code

7.1 Adapting objective function

```
double calcolaSigma(double **V, double **U, double count_t, int weight_t) {
    //for each centroid
    for (i = 0; i < c; i++) {
        V[i][d + 1] = 0; //cluster counter reset
        for (j = 0; j < n; j++) {
            V[i][d] += pow(U[i][j], m) * pow(calcDistanza(V[i], X[j]), 2.0);

            //counting highly belonging points
            if (U[i][j] > count_t)
                V[i][d + 1]++;
        }

                //adding sigma weight
        if (V[i][d + 1] > weight_t) {
            weight_v = 1 - (V[i][d + 1] / n);
            V[i][d] = V[i][d] * weight_v;
        }
    }

    //sum of individual sigma values
    for (i = 0; i < c; i++) {
        sigma += V[i][d];
    }
    return sigma;
}
```