Reducing Time Complexity of Fuzzy C Means Algorithm

Amrita Bhattacherjee [1], **Sugata Sanyal** [2], **Ajith Abraham** [3]

[1] Department of Statistics, St. Xavier's College, Kolkata – 700016, India
amritab1211@gmail.com

[2] School of Technology & Computer Science, Tata Institute of Fundamental Research,
Mumbai – 400005, India
sanyals@gmail.com

[3] Machine Intelligence Research Labs (MIR Labs) Scientific Network for Innovation and Research Excellence, Auburn, Washington 98071, USA
ajith.abraham@ieee.org

Abstract: **The Fuzzy C-Means clustering technique is one of the most popular soft clustering algorithms in the field of data segmentation. However, its high time complexity makes it computationally expensive, when implemented on very large datasets. Kolen and Hutcheson [1] proposed a modification of the FCM Algorithm, which dramatically reduces the runtime of their algorithm, making it linear with respect to the number of clusters, as opposed to the original algorithm which was quadratic with respect to the number of clusters. This paper proposes further modification of the algorithm by Kolen et. al., by suggesting effective seed initialisation (by Fuzzy C-Means++, proposed by Stetco et. al.[2]) before feeding the initial cluster centers to the algorithm. The resultant model converges even faster. Empirical findings are illustrated using synthetic and real-world datasets. Finally, we check the algorithm's robustness to perturbations in the data.**

Keywords: Clustering, Fuzzy partitions, Time complexity, Fuzzy C-Means algorithm, Unsupervised Machine Learning

I. Introduction

Cluster analysis or clustering is a method of grouping data points into different clusters or categories such that objects within the same cluster are more similar to each other than objects in different clusters. The objects are grouped together based on some similarity measure, which is specified depending on the data at hand and the objective of the task. This method has widespread application, ranging from pattern recognition and market segmentation to image processing and various other fields of data analysis.

The Fuzzy C-Means algorithm is one such clustering algorithm, which facilitates soft partitioning of the objects in the dataset. The earliest applications of clustering primarily focused on 'crisp' partitions of objects, where each point either fully belongs to a category or does not belong to a category at all. This approach relied on the idea that an object in a category does not bear any resemblance to any of the categories except to the one it belongs to. Soft partitions, on the other hand, rely on the idea that each object is characterised by the extent to which they belong to all the clusters/categories. A measure of this extent of an object's resemblance to each cluster is introduced by Zadeh (1965) [11] in the form of what is now known as a 'membership function'. The final goal is to create partitions or clusters with soft or fuzzy margins. As stated by Bezdek et. al. [3]: "A fuzzy c-partition of (the dataset) X is one which characterizes the membership of each sample point in all the clusters by a membership function which ranges between 0 and 1". The detailed definition of fuzzy c-means (FCM) partitioning and the corresponding algorithm, as proposed by Bezdek et al. [3], is given in Section 3.1.

The main limitation of this algorithm is its time complexity and memory requirements. The algorithm alternates between estimating cluster centers from the membership matrix and updating the membership matrix based on the cluster centers. As such, the membership matrix, which is of the order of the number of objects to be clustered, is repeatedly accessed and updated, on every iteration. This greatly affects the speed of the algorithm when the dataset is very large. This problem has been widely addressed in the literature. This paper focuses on the modification proposed by Kolen and Hutcheson (2002) [1], where the membership matrix is not generated (or updated) iteratively. This modification generates an algorithm which has a time complexity of $O(ncp)$ as opposed to Bezdek's original FCM Algorithm, which had a time complexity of $O(nc^2p)$, where n is the number of objects in the dataset, c is the number of clusters and p is the number of features of each object/point in the data. Let us call this algorithm FCM-U, where U refers to the membership matrix.

This paper employs the FCM-U algorithm and pairs it with the popular approach of effective seed initialisation for even faster convergence. Here, the FCM++ algorithm (proposed by Stetco et. al. [2]) is implemented for effective seed initialization. On clubbing these two algorithms together, the model runs faster and empirically converges earlier than the FCM-U algorithm. The following section discusses some related works in reducing time complexity of the FCM Algorithm, followed by short descriptions of the original FCM algorithm, the FCM++ approach and the FCM-U algorithm. Then, the proposed model is defined, followed by a comparative analysis of the results obtained when this algorithm is employed for clustering datasets. In Section 5, we check the robustness of the proposed algorithm against perturbations in the data. Finally, some further scopes of improvement are discussed.

II. Related works

Several researchers have proposed methods to tackle the problem of high computational cost that comes with implementation of the Fuzzy C-Means algorithm.

In 1986, Cannon, Dave and Bezdek [4] proposed an Approximate Fuzzy C-Means algorithm where the exact variates in the equation are replaced with integer/real-valued estimates. With their approach, they sped up the computation six times that of the original FCM implementation, while keeping the accuracy of cluster results unchanged. Tolias and Panas [5] applied spatial constraints on image segmentation problems using a fuzzy rule-based system, which showed reduced computational time. In 1994, Kamel and Selim [6] proposed two algorithms that converged faster than the FCM algorithm, having adopted a continuous process where the algorithm starts updating the membership values as soon as a part of cluster centers are updated. Their algorithm found manyfold applications in Pattern Recognition problems and were very effective in speeding up the algorithm. While [4], [5], [6] and [7] proposed modifications in the computation itself, some researchers applied modifications in handling the data instead. In 1998, Cheng et. al. [7] proposed a multi-stage random sampling approach where the cluster centers are estimated after taking repeated random samples from the data. Then, the centroids are initialised over the entire data. This process reported a speed-up of 2-3 times than the original algorithm. Note that the algorithmic iterations of the original FCM algorithm is preserved. The data feed is manipulated to obtain gain in time. The problem of time complexity is so influential in this algorithm that research in this field continued during the past two decades and is still ongoing. In 2007, Hore et. al. [8] proposed a single-pass fuzzy c-means algorithm using weighted point calculation. In 2002, Kolen and Hutcheson [1] proposed a modification which eliminates the task of repeatedly updating the membership matrix, this reducing the time complexity to a linear function of the number of clusters; as opposed to the original algorithm which was a quadratic function of the number of clusters. This was particularly beneficial for large datasets. In fact, this paper implements this approach in the proposed algorithm along with effective seed initialisation. In 2001, Hung and Yang [9] proposed the psFCM algorithm which used a simplified subset of the original data to speed up the convergence. Unlike [7], here, the subset of the data is further simplified to obtain approximate results. Several approaches were made to eliminate initial bias and reduce the time taken for convergence of the FCM algorithm. These research works mainly focused on modifying the initial centroids which are passed to the algorithm. Effective seed initialisation shows promising result in removing initial bias of the FCM algorithm. In 2015, Stetco, Zeng and Keane [2] extended the idea of K-Means++ [10] algorithm into the standard version of Fuzzy C-Means.

Research to speed up the Fuzzy C-Means algorithm is mainly motivated by its potential to be applicable in a myriad of disciplines. The concept of fuzzy sets has been a long-standing tool in different fields of study and research. Tlili et al. [18] applied this concept as fuzzy cognitive maps for effective risk management in software projects. They implemented fuzzy cognitive maps with Reinforcement Learning to obtain a model that could efficiently study project risk management as a decision validation tool. While this paper shows the application of fuzzy sets in computer applications and economics, Aggarwal et al. [19] presented another application of fuzzy logic in the field of medicine. They formulated a fuzzy logic based interface which predicted the risk of onset of depression, based on four predictor variables. This paper deals with a specific shortcoming of the algorithm and a possible algorithmic modification to tackle it from two different perspectives. For readers interested in more versatile research trends involving this algorithm, Nayak et al. [20] provides an excellent review report of the advances of this algorithm over the decade of 2000 to 2014.

III. Fuzzy C-Means (FCM) Algorithm and its variants

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of n points in $\mathcal{R}^p$, the p-dimensional Euclidean space. For $1 \leq c \leq n, c \in \mathcal{N}$, the set of natural number, a fuzzy c-partition of X is represented by (U, X) where, U is a matrix of order $n \times c$, that is –

$$U = \left((u_{ij})\right)_{n\times c}$$

where, u_{ij} denotes the membership value of the i^{th} point in X to the j^{th} fuzzy set. Here, $1 \leq i \leq n$ and $1 \leq j \leq c$. The values of the membership matrix are subject to the following conditions:

1. $0 \leq u_{ij} \leq 1, \qquad \forall\, i, j$
2. $\sum_{j=1}^{c} u_{ij} = 1, \qquad \forall\, i$
3. $0 < \sum_{i=1}^{n} u_{ij} < n, \qquad \forall\, j$

The FCM algorithm defines a constant m, which is called the fuzziness parameter and corresponds to the degree of fuzziness of the clusters.

By convention, we take $m > 1$. The FCM Algorithm then defines 'cluster centers' $\boldsymbol{v_j}$, $1 \leq j \leq c$ as:

$$v_j = \frac{\sum_{i=1}^{n} x_i u_{ij}^m}{\sum_{i=1}^{n} u_{ij}^m} \qquad (1)$$

The membership function is typically defined as:

$$u_{ij} = \left(\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}\right)^{-1},$$

$$for\ 1 \leq i \leq n\ and\ 1 \leq j \leq c \qquad (2)$$

where,

$d_{ij} = \|x_i - v_j\|$ is the distance of the i^{th} point in X to the j^{th} cluster center.

The cost function is defined as:

$$J_m(U, V; X) = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ji}^m \|x_i - v_j\|^2 \qquad (3)$$

Therefore, the Fuzzy C-means algorithm as proposed by Bezdek is given by:

Algorithm 1 : FCM	
1.	Fix c, m. Choose an initial membership matrix $U^{(0)}$
2.	At step k, compute the means $\boldsymbol{v_j}$, $\boldsymbol{1 \leq j \leq c}$ using equation 1
3.	Update membership matrix $U^{(k)}$, using equation 2
4.	Repeat steps 2 and 3 until: $\| U^{(k+1)} – U^{(k)}\| < \in$. Or, until k reaches the maximum number of permissible iterations

A. *Effective Seed Initialization and Eliminating the U-Matrix*

The Fuzzy C-Means ++ algorithm as proposed by Stetco et.al. uses effective seed initialisation to determine the starting values for the FCM algorithm. Before stating the algorithm, we state some notations :

c : number of clusters
p : dimension of the datapoints
s : the spreading factor
V : the c x p prototype matrix
X : the n x p data matrix

They defined a value P_i , corresponding to the ith data point in X, given by:

$$P_i = \frac{d^s(x_i, V)}{sum(d^s)}$$

where, $d^s(x_i, V)$ denotes the distance (raised to the power s) from a point $x_i \in X$ to its closest representative in R. The value of s controls the spreading factor of the algorithm. A small value of s will choose centers which are very close to each other, whereas a very large value of s might lead to the choice of outliers as cluster centers. When s is taken to be zero, the algorithm reduces to random seed initialisation. Further, the first point is randomly chosen and determines the selection of all the other centers. With the values and parameters defined above, the FCM++ algorithm by Stetco et.al. is as given in Algorithm 2.

Algorithm – 2: FCM++ initialisation

```
function FCM++(X,c)
begin
    V = V ∪ random point from dataset
    while sizeOfV<k do
    begin
        choose $x_i \in X$ with probability $P_i$
        V = V ∪ $x_i$
    end
    return V
end    //FCM++ ends here
```

We now state the algorithm as proposed by Kolen et.al. which constitutes the main body of the algorithm. In 2002, John Kolen and Tim Hutcheson proposed a modification in the algorithm which reduced the time of computation drastically. They eliminated the storage of the membership matrix at every iteration, and directly computed the updated cluster centers and is detailed in Algorithm 3.

We use the following notations:

c : number of clusters
p : dimension of the datapoints
n : number of data points
m : the fuzziness coefficient
V : the c x p prototype matrix
J : the current cost measure
X : the n x p data matrix

Algorithm – 3: Eliminating U-Matrix

```
function UpdateV(V,c,X,p,n,m)
begin
    //save the current V matrix
    oldV=V
    //Initialise cost at 0
    J = 0
    rowsumU = 0     //c-dimensional vector
    V = 0    //initialise new V matrix to zero
    for k=1 to n do
    begin
//Calculate the distances from the current datapoint X[i] to the centers in oldV
//Calculate the numerators and denominators of equation 2 for this data point
//initialise accumulator for denominator in equation 2
    denom3 = 0
    for i = 1 to c do
    begin
        //calculate distance between current datapoint and ith cluster center
        dsqr[i] = $(||X[k]-oldV[i]||)^2$
        //save numer3[i] for future use
        numer3[i] = $(dsqr[i])^{(1/m-1)}$
        //Update denom3
        denom3 = denom3 + 1/numer3[i]
    end
    for i = 1 to c do
    begin
        u = $(numer3[i]*denom3)^{(-m)}$
        //Update the cost (optional)
        J = J + dsqr[i]*u
        //Update numerator of prototype centers
        V[i] = V[i] + u*X[k]//p-vector operation
        //Update future denominators of centers
        rowsumU[i] = rowsumU[i] + u
    end    //for i = 1 to c
    end    //for k = 1 to n
    //Combine numerator and denominators
    for i = 1 to c do
        V[i] = V[i]/rowsumU[i]
    return V,J
end    //UpdateV ends here
```

We use the above algorithms to derive the proposed algorithm.

B. *Proposed Algorithm*

This paper implements an algorithm which combines the previous methods into a single implementation. In other words, we first generate a prototype matrix using effective seed initialisation (FCM++), and then use this initial prototype matrix as the starting point of the algorithm as stated in Section 3.1. Additionally, some modifications were made so that the algorithm works even when the cluster centers are points from the dataset itself. The algorithm is as stated below:

Step 1 : Run algorithm 2 to obtain initial cluster centers
Step 2 : Pass V obtained in Step 1 to algorithm 3. Run algorithm 3 with some modifications. The modified version is given below –

```
function ModifiedUpdateV(V,c,X,p,n,m)
begin
    oldV=V
    J = 0
    rowsumU = 0
    V = 0
    for k=1 to n do
    begin
        denom3 = 0
        flag = -1
        //flag to check the equality of points with cluster centers
        for i = 1 to c do
        begin
            $dsqr[i] = (||X[k]-oldV[i]||)^2$
            if dsqr[i]==0:
                flag=i
                continue to next i
            $numer3[i] = (dsqr[i])^{(1/m-1)}$
            denom3 = denom3 + 1/numer3[i]
        end
        for i = 1 to c do
        begin
            if i==flag:
            u=1
            V[i] = V[i] + X[k]
            rowsumU[i] = rowsumU[i] + 1
            else:
            $u = (numer3[i]*denom3)^{(-m)}$
            J = J + dsqr[i]*u
            V[i] = V[i] + u*X[k]
            rowsumU[i] = rowsumU[i] + u
        end    //for i = 1 to c
    end    //for k = 1 to n

    for i = 1 to c do
        V[i] = V[i]/rowsumU[i]
    return V,J
end
```

IV. Experimental Results

Kolen and Hutcheson [1] illustrated the performance impacts of their modification in great detail. The algorithm implemented in this paper shows further improvement in computation speed owing to effective seed initialisation. The results are illustrated on 4 datasets – the Iris Dataset [12], Wine Dataset [13] and 2 synthetic datasets generated from gaussian distributions. The time for convergence (to reach the same cost value) was measured (in seconds) for both the original FCM algorithm and the proposed modified algorithm while varying the number of clusters. The empirical findings are Tabulated in Tables 1 to 4.

Table 1: Time for convergence for the Iris dataset

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.058	0.021
3	0.147	0.047
4	0.193	0.072
5	0.299	0.093
6	0.384	0.101

Table 1 indicate that the proposed algorithm provides a considerable gain in time due to faster convergence with the same cost value. The time taken for convergence is plotted in Figure 1. The black points are the time taken (in seconds) by the original algorithm, plotted against the number of clusters specified to the algorithm. To compare the rate of change in time taken for each algorithm, a simple linear regression is fitted for each of them. The following graph gives a visual representation of the results obtained.

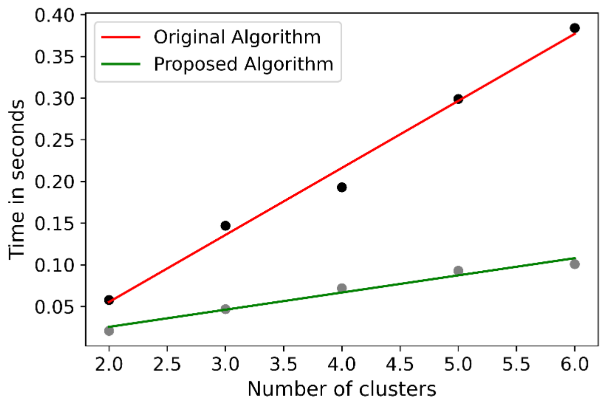

Figure 1. Iris Dataset performance

The regression equations obtained are:

$$Time_{Original} = (0.0804 \times N) - 0.1054$$
$$Time_{Proposed} = (0.0206 \times N) - 0.0156$$

where, N represents the number of clusters. The regression is done keeping the number of features in the dataset constant. It can be noted visually from the graph that the time taken by the original algorithm is consistently higher than that by the proposed algorithm. In addition, the rate of increase in time as the number of clusters increases can be obtained from the regression equations as follows:

$$Slope\ for\ Original\ Algorithm = 0.0804$$
$$Slope\ for\ Proposed\ Algorithm = 0.0206$$

Clearly, the rate of increase in time for a unit increase in the number of clusters is approximately 4 times higher for the original algorithm than that for the proposed algorithm. This validates a considerable amount of savings in time, especially for higher number of clusters.

The time taken are recorded while keeping the cost value constant for a given number of clusters, which enables a fair comparison. The cost is calculated using (3). For perspective, the performance of the proposed algorithm in predicting the correct clusters can be visually estimated by looking at the following graphs. Figure 2 represents the true clusters as available in ground truth labels of the dataset.

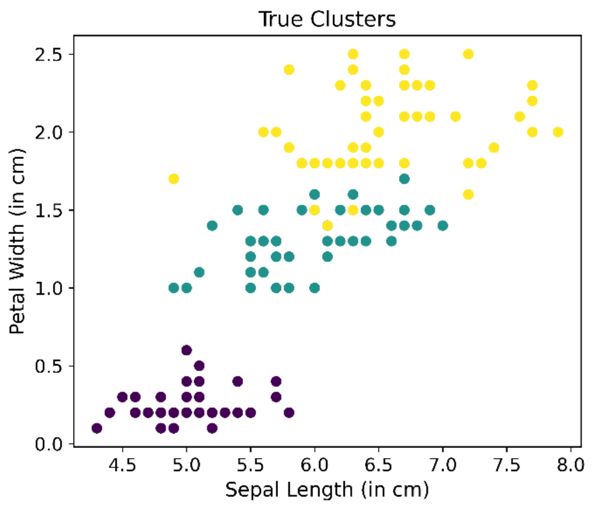

Figure 2(a). Iris dataset: True clusters

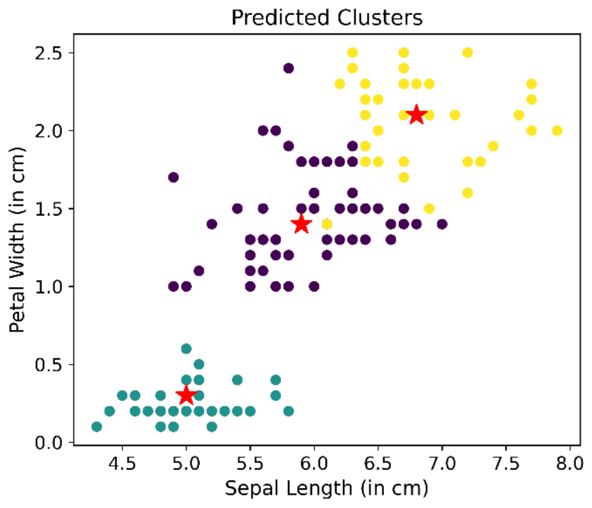

Figure 2(b). Iris dataset: Predicted clusters (red stars indicate the predicted cluster centers)

The Wine Dataset contains data on the results of a chemical analysis of 3 different types of wine grown in the same region in Italy. The 13 different features for each datapoint are actually the amount of each of the 13 different constituents found in the analysis. The attributes are real-valued numbers. There is a total of 178 datapoints. The time for convergence (to reach the same cost value) was measured (in seconds) for both the original FCM algorithm and the proposed modified algorithm while varying the number of clusters and the results are depicted in Table 2.

Table 2: Time for convergence for the Wine dataset

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.184	0.074
3	0.619	0.321
4	0.794	0.288
5	2.493	0.504
6	2.501	0.915

The proposed algorithm, once again, shows significant economy in terms of time taken till convergence. A similar study is done to obtain simple linear regression equations for each of the algorithms. The regression lines are plotted against the number of clusters in Figure 3.

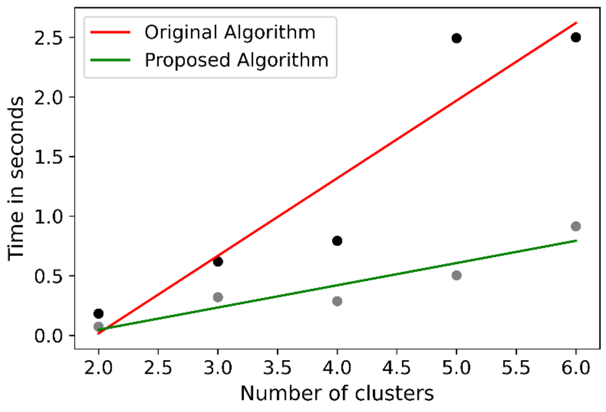

Figure 3. Wine Dataset performance

The regression equations obtained are:

$$Time_{Original} = (0.6508 \times N) - 1.285$$
$$Time_{Proposed} = (0.1865 \times N) - 0.326$$

where, N represents the number of clusters. The regression is done keeping the number of features in the dataset constant. It can be noted visually from the graph that the time taken by the original algorithm is consistently higher than that by the proposed algorithm. In addition, the rate of increase in time as the number of clusters increases can be obtained from the regression equations as follows:

$$Slope\ for\ Original\ Algorithm = 0.6508$$
$$Slope\ for\ Proposed\ Algorithm = 0.1865$$

Here, the rate of increase in time for a unit increase in the number of clusters is approximately 3.5 times more for the original algorithm than that for the proposed algorithm. The time taken are recorded while keeping the cost value constant for a given number of clusters, which enables a fair comparison. The cost is calculated using (3). Figure 4 illustrates the true clusters and the predicted clusters for the Wine dataset.

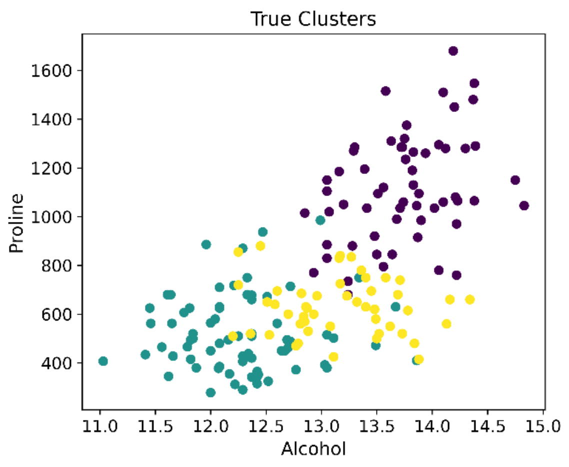

Figure 4(a). Wine data set : True clusters

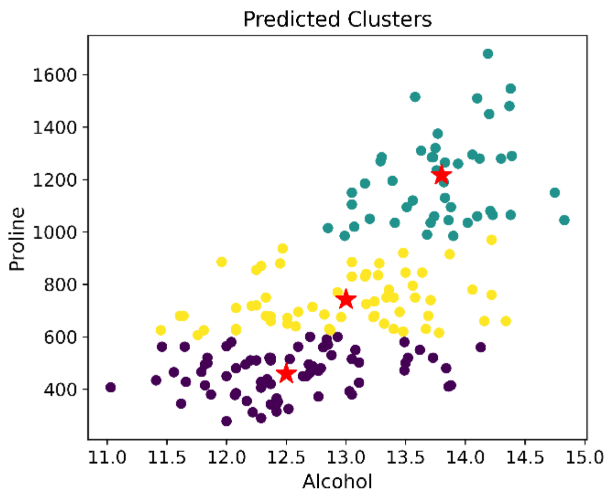

Figure 4(b). Wine data set: Predicted clusters (red stars indicate the predicted cluster centers)

Isotropic gaussian blobs are generated using Python's Scikit-learn library. The dataset generated for this problem contains 3 clusters where cluster centers are generated at random from the interval (-10, 10). The standard deviation for each cluster is set at 1 (to maintain homoscedasticity). The random state is fixed at '0'. Under the above conditions, 300 points are generated, each having 3 features. The points are plotted on a 2-dimensional space for visualisation in Figure 5.

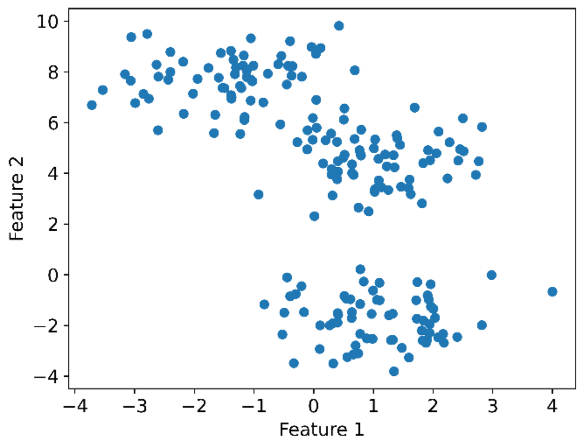

Figure 5. Isotropic gaussian blobs

The time for convergence (to reach the same cost value) was measured (in seconds) for both the original FCM algorithm and the proposed modified algorithm while varying the number of clusters and the results are illustrated in Table 3 and the time taken over the number of clusters is depicted in Figure 6.

Table 3: Time of convergence for Gaussian dataset (Type 1)

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.115	0.043
3	0.126	0.037
4	0.461	0.126
5	0.642	0.225
6	0.840	0.267

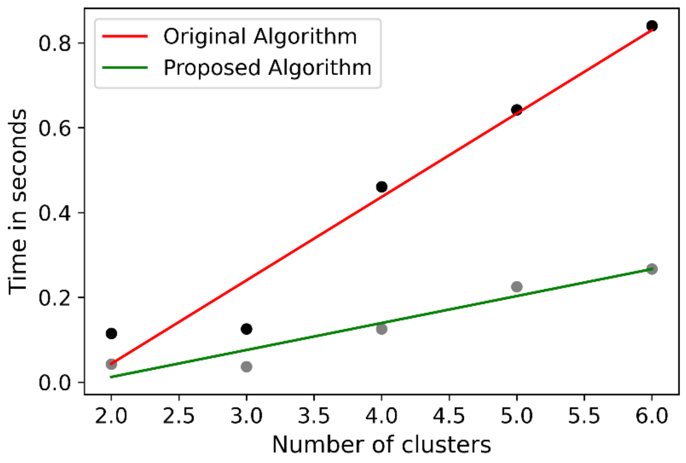

Figure 6. Isotropic gaussian blobs dataset performance

Referring to Figure 6, the regression equations obtained are:

$$Time_{Original} = (0.197 \times N) - 0.345$$
$$Time_{Proposed} = (0.064 \times N) - 0.115$$

where, N represents the number of clusters. The number of features in the dataset is kept constant. It can be noted visually that the time taken by the original algorithm is consistently higher than that by the proposed algorithm. In addition, the rate of increase in time as the number of clusters increases can be obtained from the regression equations as follows:

$$Slope\ for\ Original\ Algorithm = 0.197$$
$$Slope\ for\ Proposed\ Algorithm = 0.064$$

Here, the rate of increase in time for a unit increase in the number of clusters is approximately 3 times more for the original algorithm than that for the proposed algorithm. Hence, we can conclude that the proposed algorithm facilitates a significant amount of savings in time to converge to the same clustering result. Figure 7 illustrates the true and predicted clusters of this simulated dataset.

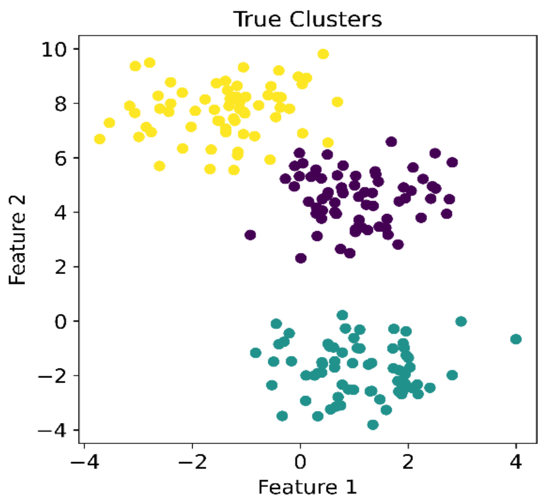

Figure 7(a). Isotropic gaussian blobs dataset: True clusters

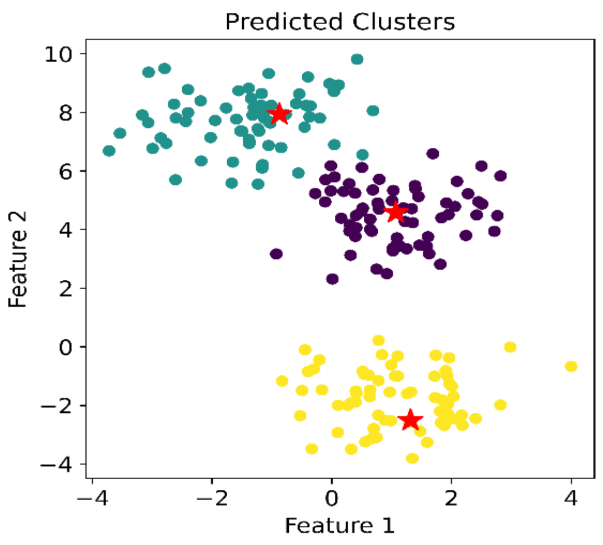

Figure 7(b). Isotropic gaussian blobs dataset: Predicted clusters (red stars indicate the predicted cluster centers)

As illustrated in Figure 8, samples from 4 gaussian distributions of varying means and standard deviations are taken to create overlapping clusters. For this particular evaluation, the means of the 4 distributions are taken as (-3,1), (2,2), (1,-3) and (5,4) with respective standard deviations 1, 0.5, 1.5 and 2 respectively. 250 points are generated from each of these distributions (making a total of 1000 datapoints). The time for convergence (keeping the cost same) is measured in seconds for both the original and the proposed algorithm are depicted in Table 4 and the clustering results are illustrated in Figures 9-10.

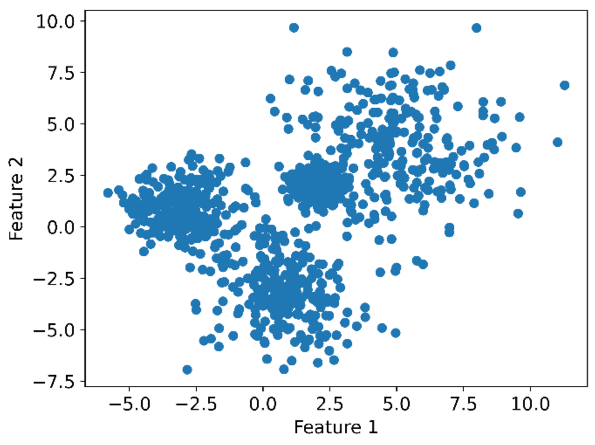

Figure 8. Gaussian Dataset (Type 2)

Table 4: Time of convergence for Gaussian dataset (Type 2)

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.738	0.321
3	1.973	0.619
4	1.264	0.442
5	5.288	0.910
6	12.016	2.402

The time taken by each of the 2 algorithms is regressed separately on the number of clusters, and two regression equations are obtained. Note that even though the regression lines (Figure 9) seem to suggest that, for 2 clusters, proposed algorithm takes more time than the original algorithm, it can be seen from the plotted points that, in the data, the proposed algorithm does in fact take less time for all clusters. The regression equations obtained are:

$$Time_{Original} = (2.587 \times N) - 6.093$$
$$Time_{Proposed} = (0.445 \times N) - 0.842$$

where, N represents the number of clusters. The number of features in the dataset is kept constant.
It can be noted that the rate of increase in time as the number of clusters increases can be obtained from the regression equations as follows –

$$Slope\ for\ Original\ Algorithm = 2.587$$
$$Slope\ for\ Proposed\ Algorithm = 0.445$$

Here, the rate of increase in time for a unit increase in the number of clusters is approximately 5 times more for the original algorithm than that for the proposed algorithm, which is especially pronounced for high number of clusters. Hence, we can conclude that the proposed algorithm facilitates a significant amount of savings in time to converge to the same clustering result.

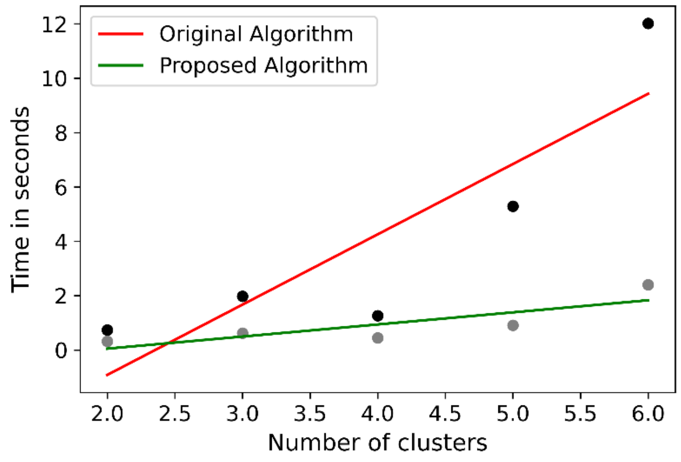

Figure 9. Gaussian Dataset (Type 2) performance

For visualisation, the true clusters are plotted below, followed by a graph illustrating the predicted clusters –

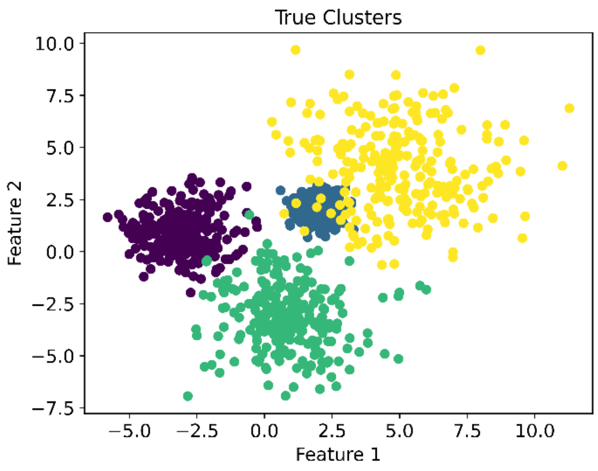

Figure 10(a). Gaussian Dataset (Type 2): True clusters

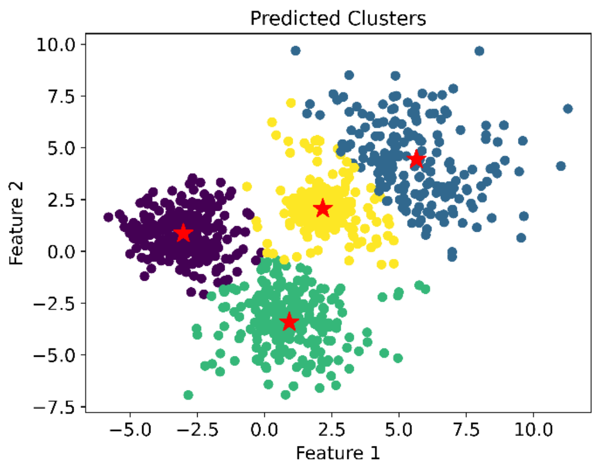

Figure 10(b). Gaussian Dataset (Type 2): Predicted clusters

The next dataset is the Breast Cancer (Wisconsin) Dataset from the UCI ML Repository [14]. This data has 569 instances, where each datapoint has 30 attributes (after eliminating the ID Number and Diagnosis). The target values are the diagnosis outcomes – Malignant or Benign. The features were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe characteristics of the cell nuclei present in the image. Some of the features, for example, include mean radius, mean texture, radial error and other measurements of the mass. The objective is to detect the diagnosis, given the feature vector. The time for convergence (to reach the same cost value) was measured (in seconds) for both the original FCM algorithm and the proposed modified algorithm while varying the number of clusters and the results are depicted in Table 5.

Table 5: Time of convergence for Breast Cancer Dataset

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.929	0.416
3	6.331	2.426
4	6.278	2.555
5	8.809	4.092
6	13.870	13.093

Clearly, the proposed algorithm converges much faster than the original one. As done with the previous datasets, a simple linear regression is implemented to obtain equations that indicate the rate of change in time. The regression lines are plotted against the number of clusters in Figure 11.

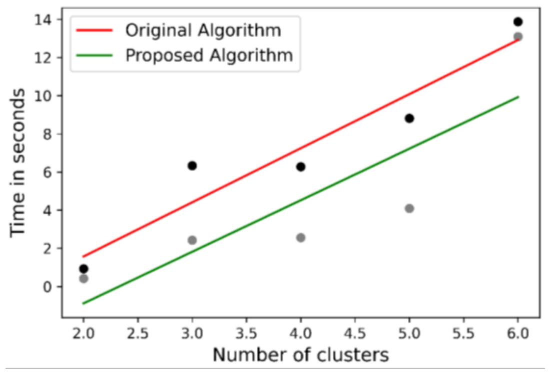

Figure 11. Breast Cancer Dataset Performance

The regression equations obtained are:

$$Time_{Original} = (2.836 \times N) - 4.1$$
$$Time_{Proposed} = (2.701 \times N) - 6.3$$

where, N represents the number of clusters. The number of features in the dataset is kept constant.
It can be noted that the rate of increase in time as the number of clusters increases are indicated by the slopes of the respective equations as follows –

$$Slope\ for\ Original\ Algorithm = 2.836$$
$$Slope\ for\ Proposed\ Algorithm = 2.701$$

Here, the rate of increase in time for a unit increase in the number of clusters is more for the original algorithm than that for the proposed algorithm. Therefore, here as well, the proposed algorithm converges much faster than the original one, which is especially advantageous for datasets with a high number of features such as the one used here. Further, the true vs. predicted clusters are indicated in Figure 12, to facilitate a visual representation of the clustering outcome. Here, each cluster represents a diagnosis, that is, one for benign and the other for malignant masses.

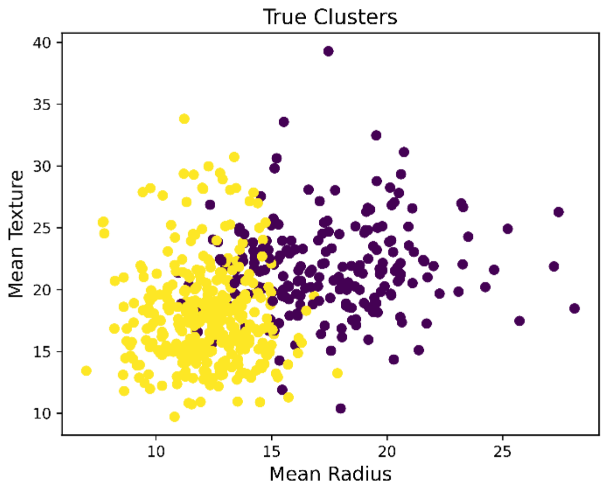

Figure 12(a). Breast Cancer Dataset : True clusters

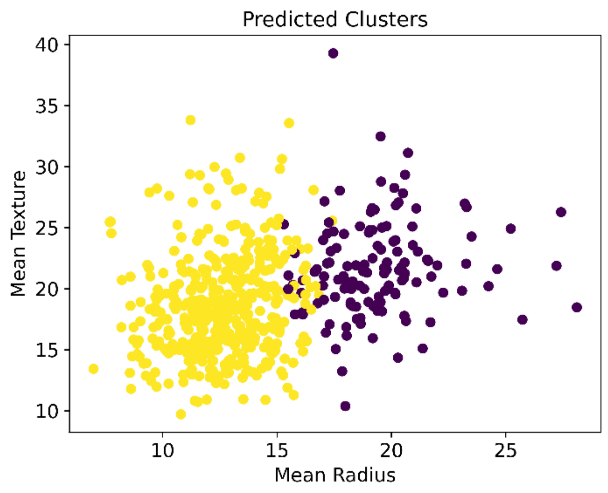

Figure 12(b). Breast Cancer Dataset : Predicted Clusters

The yellow points represent benign masses while the purple ones represent malignant ones. Since the clusters have a very high overlap, standard global clustering techniques are not ideal for this dataset. Therefore, although most of the points are correctly clustered, a considerable number of 'false positives' can be seen in terms of diagnosis. However, its performance is equivalent to the that of the traditional FCM algorithm, in terms of cluster accuracy. Therefore, the purpose of the study is fulfilled.

We finally illustrate the algorithm's performance over non-convex datasets as well, keeping in mind that the cluster accuracy is expected to be lower for non-convex datasets compared to performance of algorithms like DBSCAN. However, we include this dataset in our study to demonstrate that the time conservation is maintained irrespective of the type of dataset used. For this, two crescents are generated using Python Scikit-learn's make_moons() method. We generate 100 datapoints for each moon, making a total of 200 points in the 2-dimensional dataset. We added a noise quotient of 0.08. The resultant dataset is shown in Figure 13.

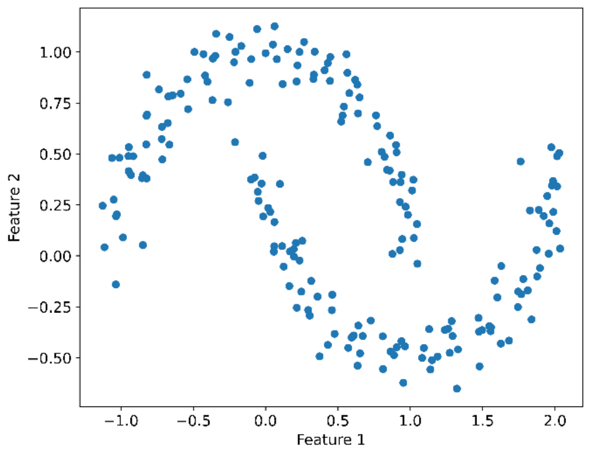

Figure 13. Crescents Dataset

The time for convergence (to reach the same cost value) was measured (in seconds) for both the original FCM algorithm and the proposed modified algorithm while varying the number of clusters and the results are depicted in Table 6.

Table 6 : Time of convergence for the crescents

Number of clusters	Algorithm Used	
	Original FCM	**Proposed FCM**
2	0.083	0.034
3	0.314	0.213
4	0.423	0.151
5	0.515	0.214
6	0.604	0.241

Time taken by the proposed algorithm is consistently lower than that by the original algorithm. Finally, a simple linear regression is implemented to obtain equations that indicate the rate of change in time. The regression lines are plotted against the number of clusters in Figure 14.

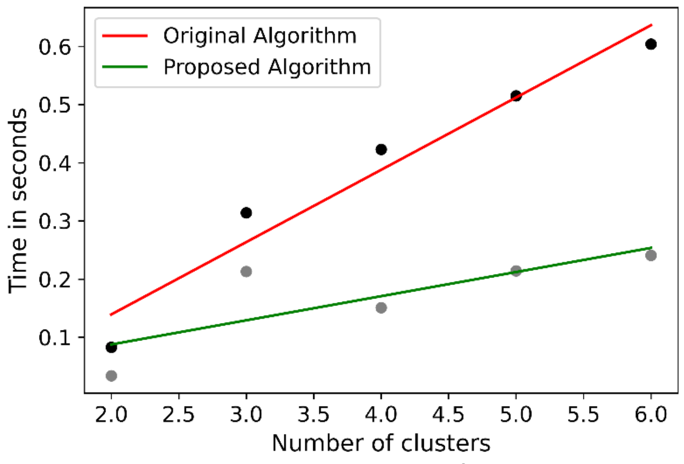

Figure 14. Crescents Dataset performance

The regression equations obtained are:

$$Time_{Original} = (0.124 \times N) - 0.109$$
$$Time_{Proposed} = (0.041 \times N) + 0.005$$

where, N represents the number of clusters. The number of features in the dataset is kept constant.
The rate of increase in time as the number of clusters increases are indicated by the slopes of the respective equations as follows –

$$Slope\ for\ Original\ Algorithm = 0.124$$
$$Slope\ for\ Proposed\ Algorithm = 0.041$$

Here, the rate of increase in time for a unit increase in the number of clusters is approximately 3 times more for the original algorithm than that for the proposed algorithm. Therefore, although a global clustering algorithm is not the most appropriate model to cluster a pair of crescents, the proposed algorithm converges much faster than the original one.

The true vs. predicted clusters are shown in Figure 15. It is evident that the clustering is not perfect. This is an expected result.

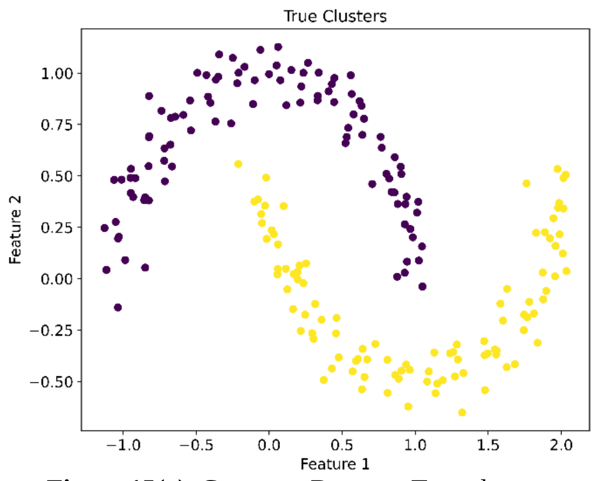

Figure 15(a). Crescents Dataset : True clusters

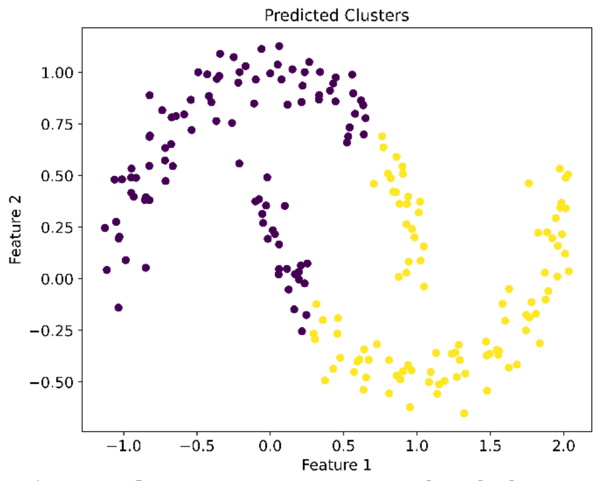

Figure 15(b). Crescents Dataset : Predicted Clusters

V. Check for Robustness

This final section checks the proposed algorithm's robustness to data perturbation. In other words, we check how resilient the proposed algorithm is to some noise/perturbation in the dataset. The procedure is explained as follows.

We randomly selected 20% points from the data at hand. To each point, we introduced some random noise. For each datapoint X_i, we generate p number of observations from a Normal distribution with mean at zero and standard deviation of 0.15. We then add this p-dimensional vector to the p-dimensional datapoint X_i, to obtain the final perturbed datapoint. Mathematically, this can be expressed as –

$$\begin{aligned}
&For\ point\ X_i\ with\ p\ features, we\ calculate\ - \\
&Y_i^{(j)} = \left(1 + z^{(j)}\right) X_i^{(j)}, \\
&where\ - \\
&X_i^{(j)}\ is\ the\ j^{th}\ feature\ of\ the\ i^{th}\ datapoint \\
&z^{(j)} \sim \mathcal{N}(0, 0.15) \\
&for\ j = 1, 2, \ldots, p
\end{aligned}$$

The vector Y_i is the final datapoint after necessary perturbation. The new dataset is so formed that it contains 80% points same as the original dataset and 20% perturbed points.

The new dataset X' is then fed to the proposed algorithm and the cluster outcome is noted. This output is then compared with that obtained with the original, unperturbed dataset and the degree of overlap is studied. If this degree of overlap is very high, we say that the algorithm is robust to perturbation. However, if the resulting clusters vary to a great extent from the clusters obtained from the original dataset, then we say that the algorithm is highly sensitive to perturbations. The degree of overlap is calculated by the adjusted rand score of the two clustering outputs for each dataset. The value of Rand Index, as defined in scikit-learn [15], computes a similarity between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the original and perturbed clusterings. This measure was first introduced by William Rand [16]. The Rand Index was then modified by Hubert and Arabie [17] and adjusted for chance, thus forming the ARI (or Adjusted Rand Index) score. The Adjust Rand Index score is therefore defined as follows –

$$Adjusted\ Rand\ Index = \frac{Rand\ Index - Expected\ Rand\ Index}{Max(Rand\ Index) - Expected\ Rand\ Index}$$

An ARI value of 1.0 indicates complete overlap between two clusters whereas an ARI value of 0 indicates no overlap. The results are tabulated in Table 7.

Table 7 : ARI for each dataset

Dataset	**Adjusted Rand Index**
Iris	0.912
Wine	0.874
Gaussian (Type 1)	0.997
Gaussian (Type 2)	0.983
Breast Cancer	0.974
Crescents	0.979

The ARI values are very close to 1, indicating that the algorithm is highly robust to perturbations. The next figures show cluster outputs on the original vs. perturbed datasets for each dataset mentioned in Table 7.

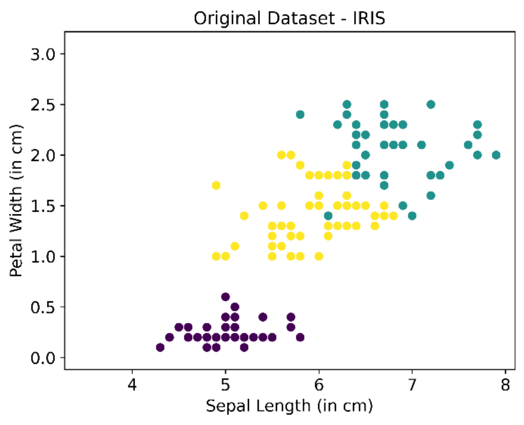

Figure 16(a). Iris Dataset – Cluster output on original dataset

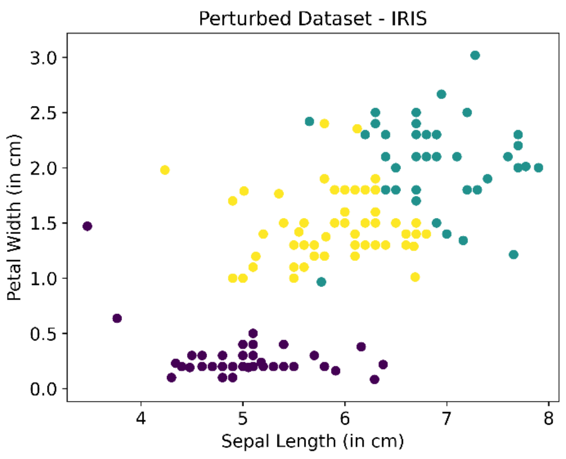

Figure 16(b). Iris Dataset – Cluster output on perturbed dataset

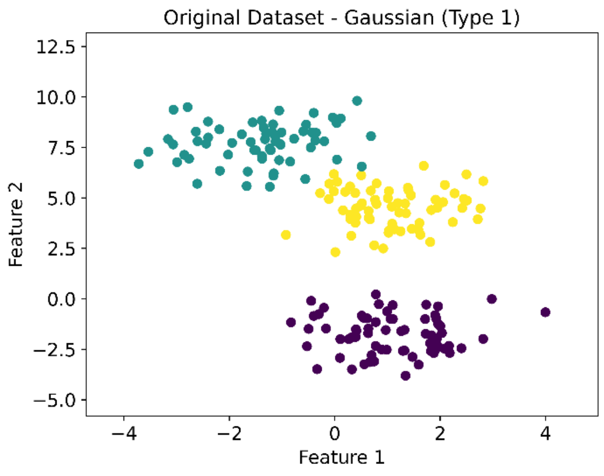

Figure 18(a). Gaussian Dataset (Type 1) – Cluster output on original dataset

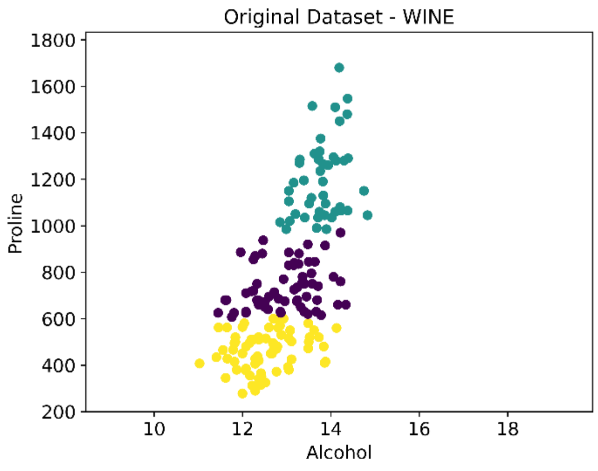

Figure 17(a). Wine Dataset – Cluster output on original dataset

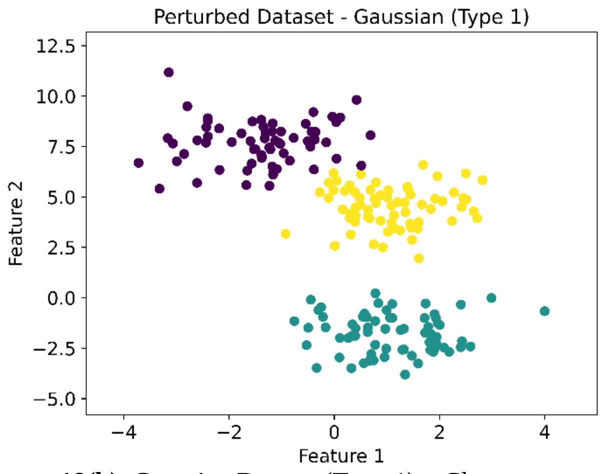

Figure 18(b). Gaussian Dataset (Type 1) – Cluster output on perturbed dataset

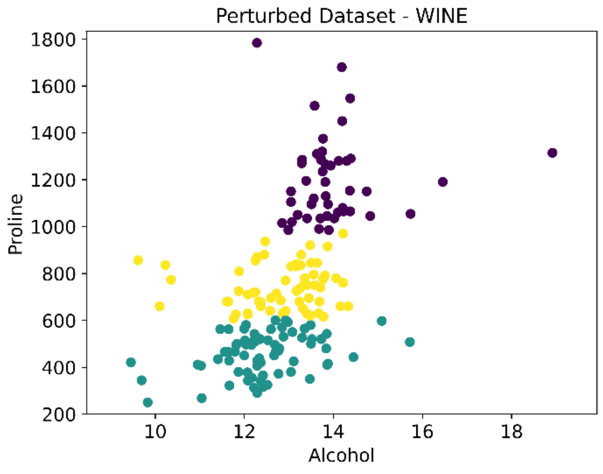

Figure 17(b). Wine Dataset – Cluster output on perturbed dataset

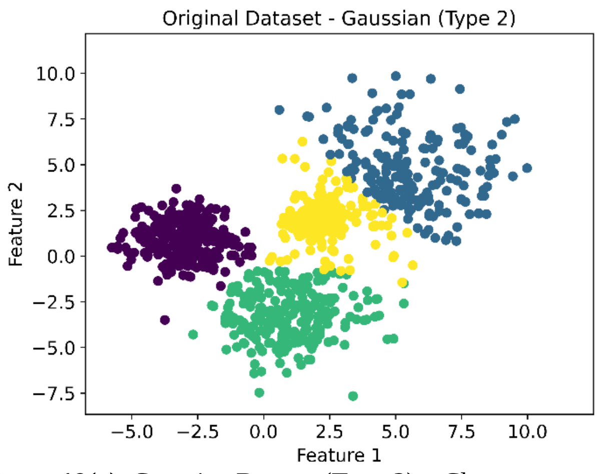

Figure 19(a). Gaussian Dataset (Type 2) – Cluster output on original dataset

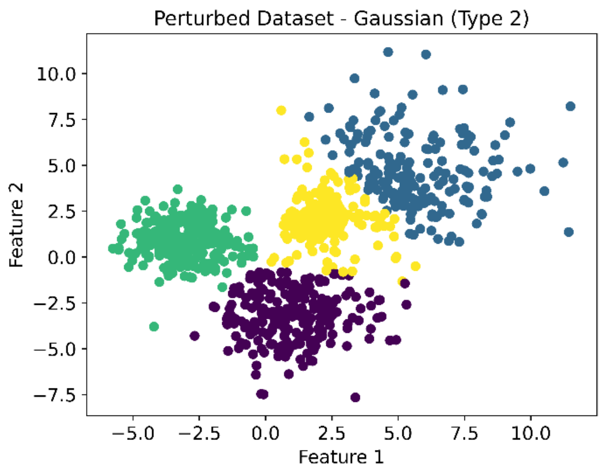

Figure 19(b). Gaussian Dataset (Type 2) – Cluster output on perturbed dataset

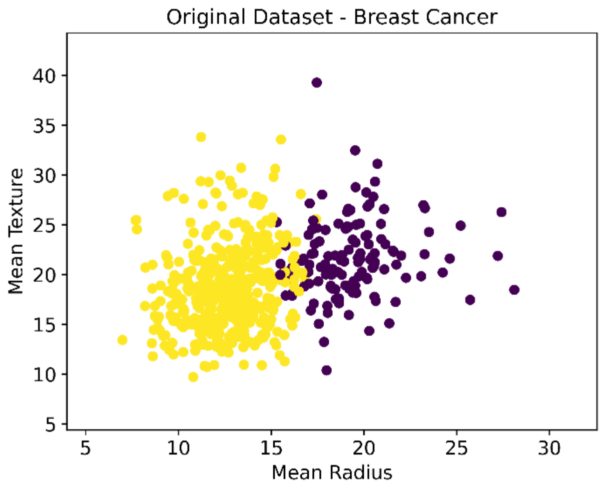

Figure 20(a). Breast Cancer Dataset – Cluster output on original dataset

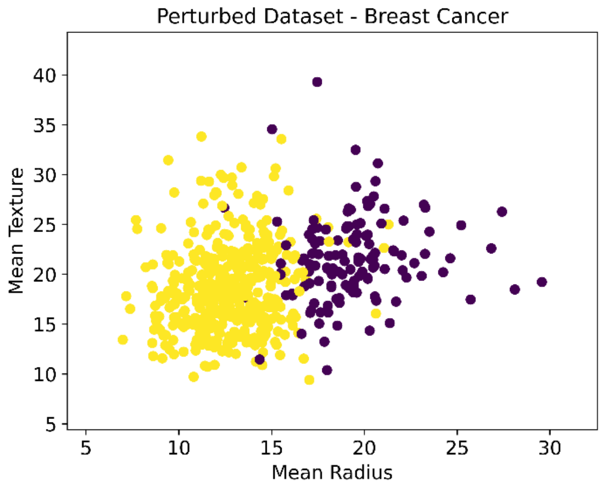

Figure 20(b). Breast Cancer Dataset – Cluster output on perturbed dataset

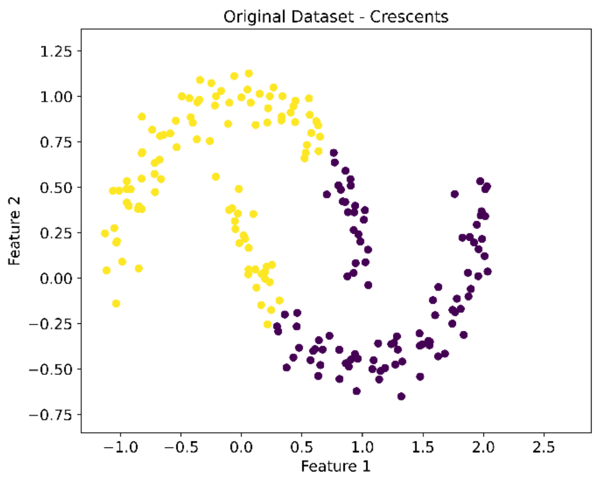

Figure 21(a). Crescents Dataset – Cluster output on original dataset

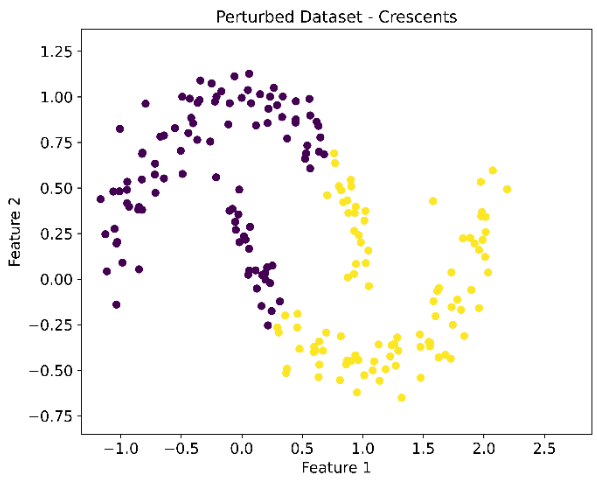

Figure 21(b). Crescents Dataset – Cluster output on perturbed dataset

VI. Conclusions

Comparative analyses of the time taken for Algorithm 2 and Algorithm 3, when implemented individually are already elaborated in [3] and [2] respectively. This paper combined these algorithms and compared its performance with the original Fuzzy C-Means algorithm to empirically confirm that it indeed accelerates the speed of the algorithm, which becomes more evident for larger datasets and higher number of clusters. In fact, the cluster accuracy stays intact (and in some cases, improves over the original FCM algorithm). Empirical results indicate faster convergence with very high cluster accuracy (as confirmed by Adjusted Rand Index during runtime). One can be interested in tailoring the algorithm to the specific data in hand. In this context, feature normalisation, feature engineering, sampling from the dataset could be viable options for further speeding up the convergence. The FCM algorithm largely depends on the initial centers selected. Further attempts could be made to eliminate the initial bias to ensure that the algorithm converges to a better solution. FCM++ has been proven to be a good approach in this context. However, testing other methods of effective seed initialisation (preferably along with Hutcheson and Kolen's [1] algorithm) might yield promising results. Combining other time-reduction approaches like random sampling of the datapoints or multi-stage random sampling [7] have been proven to be very successful. Pairing this strategy with the proposed algorithm is expected to perform extremely well for large datasets. Another open field of application is image segmentation. FCM algorithm finds manifold implementations in image segmentation problems, where the image sizes are quite high. In such a scenario, modifying the algorithm to accommodate image data and effectively reducing its runtime will open new avenues. The authors of this paper are looking into a similar implementation on image data, and tailor the time complexity reduction approach towards image-segmentation problems.

References

[1] Kolen, John F., and Tim Hutcheson. "Reducing the time complexity of the fuzzy c-means algorithm." *IEEE Transactions on Fuzzy Systems* 10.2 (2002): 263-267.

[2] Stetco, Adrian, Xiao-Jun Zeng, and John Keane. "Fuzzy C-means++: Fuzzy C-means with effective seeding initialization." *Expert Systems with Applications* 42.21 (2015): 7541-7548.

[3] Bezdek, James C., Robert Ehrlich, and William Full. "FCM: The fuzzy c-means clustering algorithm." *Computers & geosciences* 10.2-3 (1984): 191-203.

[4] Cannon, Robert L., Jitendra V. Dave, and James C. Bezdek. "Efficient implementation of the fuzzy c-means clustering algorithms." *IEEE transactions on pattern analysis and machine intelligence* 2 (1986): 248-255.

[5] Tolias, Yannis A., and Stavros M. Panas. "On applying spatial constraints in fuzzy image clustering using a fuzzy rule-based system." *IEEE Signal Processing Letters* 5.10 (1998): 245-247.

[6] Kamel, Mohamed S., and Shokri Z. Selim. "New algorithms for solving the fuzzy clustering problem." *Pattern recognition* 27.3 (1994): 421-428.

[7] Cheng, Tai Wai, Dmitry B. Goldgof, and Lawrence O. Hall. "Fast fuzzy clustering." *Fuzzy sets and systems* 93.1 (1998): 49-56.

[8] Hore, Prodip, Lawrence O. Hall, and Dmitry B. Goldgof. "Single pass fuzzy c means." *2007 IEEE International Fuzzy Systems Conference*. IEEE, 2007.

[9] Hung, Ming-Chuan, and Don-Lin Yang. "An efficient fuzzy c-means clustering algorithm." *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE, 2001.

[10] Arthur, David, and Sergei Vassilvitskii. *k-means++: The advantages of careful seeding*. Stanford, 2006.

[11] Zadeh, Lotfi A. "Fuzzy Sets, Information and Control, 8: 338-353." *MathSciNet zbMATH* (1965).

[12] https://archive.ics.uci.edu/ml/datasets/iris (accessed on November 20, 2021)

[13] https://archive.ics.uci.edu/ml/datasets/wine (accessed on November 20, 2021)

[14] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

[16] Rand, William M. "Objective criteria for the evaluation of clustering methods." *Journal of the American Statistical association* 66.336 (1971): 846-850.

[17] Hubert, Lawrence, and Phipps Arabie. "Comparing partitions." *Journal of classification* 2.1 (1985): 193-218

[18] Tlili, A., Chikhi, S. and Abraham, A. "Project Risks Management: Applying Extended Fuzzy Cognitive Maps with Reinforcement Learning". *International Journal of Computer Information Systems and Industrial Management Applications*. 12.182-192 (2020): 2150-7988

[19] Aggarwal, A., Choudhury, T., and Dewangan, B. "A Fuzzy Interface System to Predict Depression Risk". *International Journal of Computer Information Systems and Industrial Management Applications*. 12.286-296 (2020):2150-7988

[20] Nayak, Janmenjoy, Bighnaraj Naik, and HSr Behera. "Fuzzy C-means (FCM) clustering algorithm: a decade review from 2000 to 2014." *Computational intelligence in data mining-volume 2* (2015): 133-149.

Author Biographies

Amrita Bhattacherjee will complete her BSc. in Statistics from St. Xavier's College, Kolkata in June 2022. Her research areas include Machine Learning, Statistical Learning and Data Science. She plans to pursue her Masters in Mathematical Data Science and Analytics.

Sugata Sanyal completed his Bachelor of Engineering from Jadavpur University, Kolkata in Electronics and Tele-Communication Engineering in 1971, Master of Technology in Electronics and Electrical Engineering from Indian Institute of Technology, Kharagpur in 1973 and Ph. D. in Computer Science, from the University of Mumbai (through Tata Institute of Fundamental Research) in 1992. Sugata worked in the Tata Institute of Fundamental Research (1973-2012) He worked in the Tata Consultancy Services as a Research Advisor (2012-2015). Sugata was a Honorary Professor in the Indian Institute of Technology, Guwahati, till early 2020 and as an Adjunct Professor in the Indian Institute of Technology, Kharagpur (2013-2014). His research areas include Mobile Ad Hoc Networks, Multifactor Security Protocol for Wireless Payment, Intrusion Detection System, Multi-Factor Security Protocol, Data Hiding Technique, Steganography and Steganalysis, Jigsaw-based Secure Data Transfer, Whole Genome Comparison, Parallel Processing, Fault Tolerance and Coding Theory.

Third Author The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author's educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state or country, and year degree was earned. The author's major field of study should be lower-cased. If a photograph is provided, the biography will be indented around it. The photograph is placed at the top left of the biography.