Comparison of different Clustering Algorithms

EE656A Course Project

Rishabh Katiyar
Roll Number:190702
EE Department, IIT Kanpur

***Abstract*—In this project, the performance of various clustering algorithms are being compared. The custering algorithms that are being compared are K-Means, Fuzzy-C Means, Improved Mountain Clustering - 1 (IMC-1), Improved Mountain Clustering -2 (IMC-2) and Self-Optimal Clustering. The quantitative and qualitative performances of all these well-known clustering techniques are presented and compared with the aid of examples on various benchmarked validation indices which are Global Silhouette Index(GSI), partition index(PI), separation index(SI), and Dunn index(DI).**

I. Introduction

CLUSTERS, by definition, are the collection of items that are comparable to one another. Each group or cluster is homogenous, which means that the things that belong to the same group are comparable to each other and to other things in that group. Also, each group or cluster ought to be distinguishable from the others; to put it another way, the items that belong to one cluster ought to be distinct from those that belong to other clusters. Clustering refers to the process of grouping together things that are related in some way, and this can be done in a hard or soft way. In the hard clustering algorithm, each element is assigned to a single cluster while it is operating. On the other hand, in the fuzzy clustering method, a degree of membership is assigned to each element based on the degree to which it is associated with several other clusters. This is in contrast to the hard clustering algorithm, which allocates each element to a single cluster while it is operating. When a fuzzy clustering is converted into a hard clustering by assigning each element to the cluster that has the largest membership, the result is a more accurate representation of the data.

Let's dicsuss about the clustering algorithms.

A. K-Means

k-means clustering is a method of vector quantization that was initially developed in the field of signal processing. Its primary objective is to partition n observations into k clusters in such a way that each observation belongs to the cluster that contains the nearest mean (cluster centres or cluster centroid), which acts as a prototype for the cluster. The k-means clustering algorithm reduces the amount of variation that exists within each cluster. The issue is computationally challenging, often known as NP-hard; yet, effective heuristic algorithms can converge to a local optimal solution in a short amount of time.

Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq$ n) sets S = $S_1, S_2, ..., S_k$ so as to minimize the within-cluster sum of squares (WCSS). Formally, the objective is to find:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{1}$$

where μ_i is the mean (also called centroid) of points in S_i, i.e.

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x \tag{2}$$

where $|S_i|$ is the size of S_i, and $||.||$ is the usual L^2 norm.

B. Fuzzy-C Means

Fuzzy c-means, also known as FCM, is a data clustering approach that involves the grouping of a data set into N clusters, with each data point in the dataset belonging to each cluster to a given degree. For instance, a data point with a location that is relatively close to the centre of a cluster will have a high degree of membership in that cluster, whereas a data point with a location that is relatively far away from the centre of a cluster will have a degree of membership that is significantly lower. It is also referred to as soft clustering or soft k-means.

Any point x has a set of coefficients giving the degree of being in the kth cluster $w_k(x)$. With fuzzy c-means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster, or, mathematically,

$$c_k = \frac{\Sigma_x w_k(x)^m x}{\Sigma_x w_k(x)^m} \tag{3}$$

where m is the hyper- parameter that controls how fuzzy the cluster will be. The higher it is, the fuzzier the cluster will be in the end.

The FCM algorithm attempts to partition a finite collection of n elements X=$x_1, ..., x_n$ into a collection of c fuzzy clusters with respect to some given criterion.

Given a finite set of data, the algorithm returns a list of c cluster centres C=$C_1, ..., C_k$ and a partition matrix W= $w_{i,j} \in [0,1]$, i=1,...,n, j=1,...,c, where each element, w_{ij} , tells the degree to which element, $\mathbf{x}_i$, belongs to cluster $\mathbf{c}_j$.

The FCM aims to minimize an objective function:

$$\sum_{i=1}^{n}\sum_{j=1}^{c} w_{ij}^{m}||x_i - c_j||^2 \tag{4}$$

where:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||x_i - c_j||}{||x_i - c_k||})^{\frac{2}{m-1}}} \tag{5}$$

C. IMC-1

Improved Mountain Clustering, also known as IMC-1, is an extension of the Mountain Clustering (MC) algorithm. MC is a density-based clustering algorithm that identifies clusters in a dataset by making use of density estimation. Improved Mountain Clustering, also known as IMC-1, is an improved version of MC. By incorporating a number of enhancements, the IMC-1 algorithm works towards the goal of overcoming a number of the shortcomings of the original MC algorithm.

It is possible for the original MC algorithm to be sensitive to the parameters that are used, such as the kernel bandwidth and the threshold value. This is one of the limitations of the algorithm. This issue is tackled head-on by the IMC-1 algorithm, which implements both an adaptable kernel bandwidth and an adaptive threshold value.

The adaptive kernel bandwidth and threshold value are given by the formula:

$$d_1 = d_2 = \frac{1}{2n}\sum_{j=1}^{n}(\frac{min(\mathbf{x}^j)}{\sum_{i=1}^{D} x_i^j}) \tag{6}$$

Here's a rough outline of the IMC-1 algorithm:

- Normalize your dataset using min max normalization.
- Calculate the distance between each pair of points using the adaptive kernel bandwidth and construct a distance matrix.
- Identify the local maxima in the distance matrix, which correspond to potential cluster centers.
- Assign each point to the nearest local maximum that exceeds the adaptive threshold value
- Repeat steps 2-4 until the clustering results converge or a predefined maximum number of iterations is reached.

D. IMC-2

IMC-2 (Improved Mountain Clustering 2) is an extension of IMC-1 which improves the clustering performance by using a more sophisticated bandwidth and threshold function which is

$$d_1 = d_2 = \frac{1}{2n}\sum_{j=1}^{n}(\frac{min(\mathbf{x}^j)}{\sum_{i=1}^{D} x_i^j}).(\alpha) \tag{7}$$

where α is:

$$\alpha = \frac{M}{M+1} \tag{8}$$

where M is the number of clusters.

Rest all the steps of the algorithm are same as IMC-1.

E. Self-Optimal Clustering

Self-Optimal Clustering is an advanced version of improved mountain clustering (IMC) technique. The threshold function is optimized using Lagrange's form of interpolation polynomial. The determination of the threshold function in SOC via interpolation method achieved a substantial improvement in cluster quality.

The threshold value is determined by:

$$\delta_m = \frac{1}{2n}\sum_{j=1}^{n}(\frac{min(\mathbf{x}^j)}{\sum_{i=1}^{D} x_i^j}).(\beta_m) \tag{9}$$

The threshold value δ_m is a positive value that defines the surrounding area of the data point that corresponds to the mth cluster. An expression derived through heuristics is multiplied by an optimising factor β_m to provide the value for the mth cluster.

The potential value P_m^r for the m^{th} cluster is calculated using the mountain function

$$P_m^r = \sum_{i=1}^{n} exp[-(\frac{d^2(x^r, x^j)}{\delta_m^2})] \tag{10}$$

The m^{th} cluster center c_m is the data point corresponding to the highest value among $P_m^1, P_m^2, ..., P_m^n$ and those data points in the data set are assigned to the m^{th} cluster whose Euclidean distance from the m^{th} cluster center is less than a threshold value δ_m i.e.,

$$d^2(\mathbf{x}^r, c_m) \leq \delta_m; \forall \text{r = 1, 2, . . . , n} \tag{11}$$

Repeat the above steps for all the m clusters.
In the next step we calculate the global silhouette value via the silhouette index (GSI) for each of the clusters. GSI values $S_1, ..., S_m$ close to unity indicates better cluster formation. Then, we form the lagrange polynomial using the the thresh- old δ_m and silhouette values S_m. We use the lagrange polynomial to find the roots and select the root(η) for which the value of the polynomial is found to be close to unity.

We divide η by δ_m for m = 1,2,3,.....,M to obtain the corresponding β_m values that will be substituted in (9) in the calculation of new δ_m while repeating the clustering process again, thus maximizing silhouette value S_m for the clusters formed.

$$\beta_m = \eta/\delta_m; \forall \text{m=1,2,3,....,M} \tag{12}$$

We repeat the whole algorithm for 4 iterations and then select the clusters according to that iteration which maximizes the GSI value.

F. Measures of Cluster Quality

- **GSI**
  The global silhouette index is a cluster validity measure that evaluates the quality of clustering by measuring how well separated clusters are from each other and how internally cohesive the clusters are. The silhouette score is a measure of how well each sample fits into its assigned cluster compared to other clusters, with higher values indicating better clustering results. Note that the global silhouette index should be maximized, so higher values indicate better clustering results. The silhouette width is a confidence indicator on the membership of the i^{th} sample in cluster X_m . It is defined as

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{13}$$

  where a(i) is the average distance between the i^{th} sample and all of the samples included in X_m ; b(i) is the minimum of the average distance between the i^{th} sample and all of the samples clustered in X_k (k = 1, ... , M ; k $\neq$ m).

  The silhouette value S_m for the m^{th} cluster is defined as:

$$S_m = \frac{1}{N_m} \sum_{i=1}^{N_m} s(i) \tag{14}$$

  The GSI is calculated as:

$$GSI = \frac{1}{M} \sum_{m=1}^{M} S_m \tag{15}$$

- **PI**
  The partition index is a cluster validity measure that evaluates the quality of clustering by measuring the compactness and separation of clusters.

$$PI = \sum_{i=1}^{M} \frac{\sum_{j=1}^{n} (\mu_{jm}^2) ||x^j - c_m||^2}{N_m \sum_{k=1}^{M} ||c_k - c_m||^2} \tag{16}$$

  where c_m is the cluster center, N_m is the fuzzy cardinality, μ_{jm} is the membership of data point j in cluster m.
  A lower value of PI indicates a better partition.

- **SI**
  SI uses a minimum-distance separation for partition validity. A lower value of separation index indicates a better partition.

$$SI = \frac{\sum_{m=1}^{M} \sum_{j=1}^{n} (\mu_{jm})^2 ||x^j - c_m||^2}{n.\min_{k,m} ||c_k - c_m||^2} \tag{17}$$

- **DI**
  Dunn's index is a cluster validity measure that evaluates the quality of clustering by measuring the compactness and separation of clusters.

$$\min_{1 \leq m \leq M} \left(\min_{1 \leq k \leq M; k \neq m} \frac{d(X_m, X_k)}{\max_{1 \leq m \leq M} (\triangle(X_m))} \right) \tag{18}$$

where $\triangle(X_m)$ is the complete diameter intracluster distance defined as:

$$\triangle(X_m) = \max_{x.y \in X_m} d(x, y) \tag{19}$$

where d(x, y) defines the distance between any two samples x and y belonging to X_m. The centroid linkage intercluster distance is defined as:

$$d(X_m, X_k) = \frac{1}{|X_m| + |X_k|} \left(\sum_{x \in X_m} d(x, v_t + \sum_{y \in X_k} d(y, v_s)) \right) \tag{20}$$

where

$$v_s = \frac{1}{|X_m|} \sum_{x \in X_m} x, \tag{21}$$

$$v_t = \frac{1}{|X_k|} \sum_{y \in X_k} y, \tag{22}$$

where $|X_m|$ and $|X_k|$ provide the number of samples included in clusters X_m and X_k.

II. Evaluation

The test image taken to test all the algorithms is shown below. Different algorithms were implemented on the following image for image segementation. The results of different algorithms is compiled below. The dataset is here.

Fig. 1. Flower Test Image

Fig. 2. K-Means Result

K-Means implementation on the above image is shown in Fig.2.

Fuzzy-C Means, IMC-1, IMC-2 and Self-Optimal Clustering results are shown in Fig.3, Fig.4, Fig.5, Fig.6 respectively:

Fig. 3. Fuzzy C-Means Result

Fig. 4. IMC-1 Result

Table I, II, III & IV shows the various evaluation scores for the different clustering algorithms on the test image. Table I shows the GSI scores, Table II shows the PI scores, Table III shows the Separation Index Scores, & Table IV shows the Dunn's Index scores.

The optimal number of clusters can be chosen based on the

Fig. 5. IMC-2 Result

Fig. 6. SOC Result

value of k which maximises the GSI value. The optimal value of k came out to be 3. A graph is shown in Fig.7 which shows the variation of GSI on varying k.

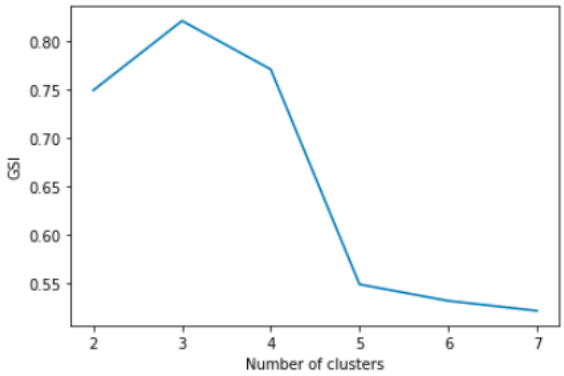

Fig. 7. Variation of GSI with number of clusters k for the test image

The algorithms were tested on another image shown in Figure 8 with optimal number of clusters as 3 and the results of various algorithms are compiled in the Table V. The result of Self-Optimal Clustering on Test Image 2 is shown in Figure 9.

The implementation of SOC on the test image is shown in Table VI for different number of iterations. The GSI values

TABLE I
GSI SCORES FOR TEST IMAGE I

Algorithm	GSI
K-Means	0.82113
Fuzzy C-Means	0.82106
IMC-1	0.82064
IMC-2	0.82065
SOC	0.82067

TABLE II
PI SCORES FOR TEST IMAGE I

Algorithm	PI
K-Means	0.00316
Fuzzy C-Means	0.02249
IMC-1	0.00235
IMC-2	0.00441
SOC	0.00002

TABLE III
SI SCORES FOR TEST IMAGE I

Algorithm	SI
K-Means	0.03670
Fuzzy C-Means	0.03471
IMC-1	0.00958
IMC-2	0.00972
SOC	0.00970

TABLE IV
DI SCORES FOR TEST IMAGE I

Algorithm	DI
K-Means	0.73224
Fuzzy C-Means	0.73239
IMC-1	0.75019
IMC-2	0.74427
SOC	0.74428

are changing and the cluster center used for segmentation is taken for that iteration for which the GSI value is the highest. It can be seen that for Iteration 9, highest GSI value is obtained. It can also be seen that the GSI values change and that the algorithm is improving itself in subsequent number of iterations as shown in Figure 10.

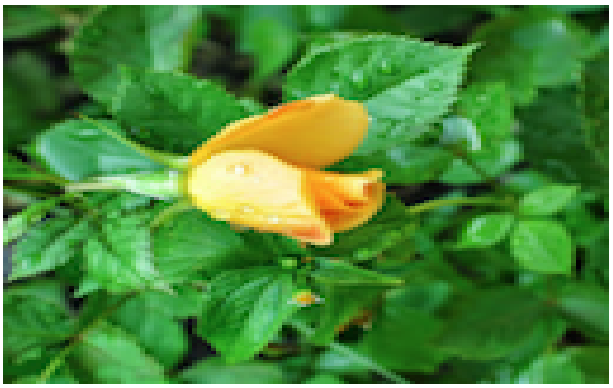

Fig. 8. Flower Test Image 2

III. CONCLUSION

By implementing various algorithms on the images, it was seen that Self-Optimal Clustering method gave the best results which were evident from the images and the validation indices. IMC-2 gave worse results compared to IMC-1 for the second test image. Thus, it can be said that IMC-2 does not always gives better results compared to IMC-1.

REFERENCES

[1] N.K. Verma, A. Roy, "Self-Optimal Clustering Technique Using Optimized Threshold Function" December 2014

[2] N.K. Verma, P. Gupta, P. Agrawal, M. Hanmandlu, S. Vasikarla, Y. Cui, "Medical Image Segmentation Using Improved Mountain Clustering Approach" 2009

[3] P. Gupta, N.K. Verma, S. Agrawal, S. Vasikarla, "Color Segmentation Using Improved Mountain Clustering Technique Version-2" 2011

Fig. 9. SOC Result on Test Image 2

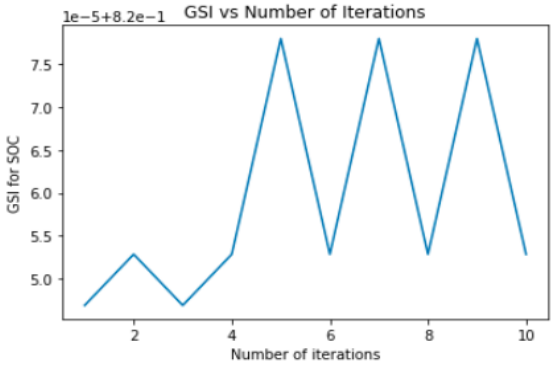

Fig. 10. Variation of GSI with number of iterations for Self Optimal Clustering

TABLE V
EVALUATIVE SCORES FOR TEST IMAGE II

Algorithm	GSI	PI	SI	DI
K-Means	0.51360	0.00604	0.19101	0.32708
Fuzzy C-Means	0.50880	0.00528	0.20875	0.32128
IMC-1	0.51236	0.00606	0.18341	0.34449
IMC-2	0.35922	0.00485	0.77608	0.27483
SOC	0.51432	2.246e-5	0.25090	0.32455

TABLE VI
VARIATION OF δ_m , S_m , GSI, η AND β_m VALUES WITH VARYING ITERATION FOR TEST IMAGE I

No. of Iterations	δ_1	δ_2	δ_3	S_1	S_2	S_3	GSI	η	β_1	β_2	β_3
1	0.07854	0.05456	0.06476	0.93793	0.66972	0.59792	0.82004	0.07993	1.0176	1.46496	1.23434
2	0.07993	0.07978	0.08026	0.93805	0.66955	0.59779	0.82005	0.08001	1.00098	1.00280	0.99689
3	0.07862	0.05471	0.06457	0.93792	0.66972	0.59791	0.82004	0.08001	1.01772	1.46237	1.23906
4	0.07993	0.07964	0.08058	0.93805	0.66955	0.59779	0.82005	0.08009	1.00191	1.00556	0.99388
5	0.07869	0.05484	0.06434	0.93792	0.66775	0.59987	0.82007	0.08011	1.01799	1.46079	1.24509
6	0.07995	0.07956	0.08098	0.93805	0.66955	0.59779	0.82005	0.08023	1.00346	1.00847	0.99079
7	0.07881	0.05499	0.06414	0.93792	0.66775	0.59987	0.82007	0.08024	1.01805	1.45892	1.25090
8	0.07996	0.07945	0.08135	0.93805	0.66955	0.59779	0.82005	0.08045	**1.00611**	**1.01250**	**0.98885**
9	**0.07902**	**0.05520**	**0.06400**	**0.93792**	**0.66775**	**0.59987**	**0.82007**	0.08045	1.01810	1.45739	1.25707
10	0.07996	0.07937	0.08176	0.93805	0.66955	0.59779	0.82005	0.08025	1.00357	1.01106	0.98156