International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 10, October - 2013

Hard and Fuzzy Clustering Algorithms Using Normal Distribution of Data Points: a Comparative Performance Analysis

Yasobanta Karali,D Kodamasingh,R B Lyngdoh H.S. Behera,
Department of Computer Science and Engineering,
Veer Surendra Sai University of Technology(VSSUT), Burla
Sambalpur, Orissa, India

Abstract: With its huge applications, 'Data Clustering' which is one the main task in data mining, is getting popular day by day. In simple words it can be defined as the process of organizing similar data objects into groups. It is considered as one among the unsupervised learning methods, which yields result without using any prior external supervision. Any Clustering technique, organized a given data set into several groups such that the similarity within a group is greater as compared to among other groups. In this paper, a review of different Hard and Fuzzy Clustering technique is done such as: K-means, K-Medoid, Fuzzy C-Means, Gustafson-Kessel, and Gath-Geva. The different techniques are implemented and tested against a normally distributed motorcycle dataset. Their performances with respect to speed and accuracy of each of the techniques are compared accordingly which includes their validation parameters as well.

Index Terms— data clustering, k-means, k-Medoid, fuzzy c-means, Gustafson-Kessel, Gath-Geva.

I. INTRODUCTION

It is remarkable to think how this particular field has find it's ways in numerous uncountable applications such as Image processing, Biological research, web documentation and many more. Data Clustering itself is well known for its interesting approach of finding similarities between data and organizing them into groups. However it does not only means organizing data's into groups but they are used for data compression and model induction as well as extraction of meaningful information As mentioned earlier Clustering is nothing but partition of given data sets into clusters such that there is a larger similarity between data in the same group as compared to those among other groups. Clustering in its approach reflects very much from our daily lives. For example so often we group our books of similar sizes together in one side of our shelves, our mother used to organize similar household utensils one side of our almaris and so on. This is not the point to be noted, if then clustering would look very simple and predictable. But living in a digital world things normally don't go easy, data tends to accumulate and multiply rapidly over time. This leads to the challenge, of how to organize these data's. Here is where 'soft computing' methods come into usefulness which helps us in clustering those huge data's.. In this paper, a brief review is done on the performance of the various Hard as well as Fuzzy Clustering algorithms. Hard algorithms such as K-Means, K-Medoid and Fuzzy algorithms such as C-Means, GK(Gustafson-Kessel), GG(Gath-Geva). These five techniques are implemented and their performance comparison is comprehensively done to

II. RELATED WORKS

There has been a tremendous growth of interest in the field of soft computing, starting from the development of K-Means to the development of fuzzy C-Means algorithms. J. C. Bezdek [1] in his book "Pattern Recognition with Fuzzy Objective Function Algorithms" has given a comprehensive detail of all the fuzzy clustering algorithms. F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler[2] in another book gave a brief idea of Fuzzy Cluster Analysis. R. Babuška[3] formulate all the clustering algorithms. D.E. Gustafson and W.C. Kessel [4] proposed the Fuzzy clustering with fuzzy covariance matrix. T.Velmurugan and T.Santhanam [5] describe the affect of normal distribution of data points on the Hard clustering algorithms. R. Babuška, P. J. van der Veen, and U. Kaymak[6] proposed the Improved covariance estimation for Gustafson-Kessel clustering. .D .Napolean et al[6] proposed an efficient

Clustering Technique to Reduce the time Complexity Of K-Means Using Normal Distribution of data points. J.C. Bezdek and J.C. Dunn [8], I. Gath and A.B [9], A.M. Bensaid et al [10] are some of the other works that has explore finding out the validation parameters of the various clustering algorithms. The implementation and analysis id one with the used of fuzzy toolbox by Balazs Balasko[11]. .

III. DATA CLUSTERING OVERVIEW

So as to get a taste of the coming sections, let's just have an overlook, about data clustering. Let's start from the term cluster itself. Many definitions of cluster have been formulated, but all solely depend on the objective of clustering. Among those the most accepted one is that cluster, is nothing but a group of objects that are more similar to one another than to members of other clusters. Here the term 'Similarity' here should be understood clearly that it is mathematical similarity, purely defined by distance Norm. In general, clusters are subsets of a data set. The major issue now is whether those subsets formed are hard (crisp) or fuzzy. Hard clustering methods are based on classical set theory, which requires an object either does or does not belong to a cluster; i-e subsets are mutually exclusive. Fuzzy clustering methods on the other hands allow objects to belong to several clusters simultaneously, but with deferent degrees of membership. In many real situations, fuzzy clustering is more natural than hard clustering, as objects on the boundaries between several classes are not forced to fully belong to one of the classes, but rather are assigned membership degrees between 0 and 1 indicating their partial memberships.

The structure of the partition matrix U = [μ_{ik}]

$$U=\begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,c} \\ u_{2,1} & u_{2,2} & u_{2,3} & \cdots & u_{2,c} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{N,1} & u_{N,2} & u_{N,3} & \cdots & u_{N,c} \end{bmatrix}$$

A. Hard partition

As we are dealing with clustering, one must be clear with the objective that is to partition the data set X into c clusters. Any Hard Partition can be defined as a family of subsets. The classical set represents Hard Clustering as $\{Ai|1 \le i \le c \subset P(x)\}$. Its properties are as follows:

$$\cup_{i=1}^{c} Ai = X$$

$$Ai \cap Aj, 1 \le i \ne j \le c,$$

$$\emptyset \subset Ai \subset X,\ 1 \le i \le c.$$

The first condition above states that the subsets A_i contain all the data in X, second one states that they must be disjoint and the last none of them must be empty nor should contains all the data in X. The membership function of Hard partition is given below:

$$V_{i=1}^{c} \mu_{A_i} = 1,$$

$$\mu_{A_i} \vee \mu_{A_{j,}} 1 \le i \ne j \le c$$

$$0 < \mu_{A_i} < 1,\ 1 \le i \le c.$$

Here μA_i is the characteristic function of the subset A_i and can have values either zero or one. In matrix notation the partition can be represented as follows: A N×c matrix U = [μ_{ik}] represents the hard partition if and only if its elements satisfies

$$\mu_{A_{ij}} \in 0,1, 1 \le i \le N, 1 \le k \le c,$$

$$\sum_{k=1}^{c} \mu_{A_{ik}} = 1, 1 \le i \le N,$$

$$0 < \sum_{k=1}^{k} \mu_{A_{ik}} < N, 1 \le k \le c.$$

B. Fuzzy partition

Fuzzy partition can be viewed as a generalization of hard partition. The only difference is that it allows μ_{ik} attaining real values between the range [0, 1]. An N×c matrix U = [μ_{ik}] represents the fuzzy partitions. Its condition is given below:

$$\mu_{ij} \in [0,1], 1 \le i \le N, 1 \le k \le c,$$

$$\sum_{k=1}^{c} \mu_{ik} = 1, 1 \le i \le N,$$

$$0 < \sum_{k=1}^{k} \mu_{ik} < N, 1 \le k \le c.$$

One thing one needs to be clear that the distribution of memberships among the c fuzzy subsets in case of fuzzy partition is not constrained. Hence data at the borders are not forced to belong to only one cluster.

IV. DATA CLUSTERING TECHNIQUES

A. K-means and K-Medoid algorithms

The popular classes of Hard Partitioning algorithm are K-Means and K-Medoid. These classes of partitioning methods are simple, popular and very widely used. But, both have a big drawback that is both does not give any fair reliable results every time and have some numerical problems as well. For any N× n dimensional data set K-means and K-Medoid algorithms is purely based on finding data clusters in a data set by minimizing the within-cluster sum of squares given below which basically denotes a distance norm:

$$\sum_{i=1}^{c}\sum_{K\in A_i}\|X_i - V_i\|^2$$

Where A_i a set of data is points or objects and V_i is the mean for that data points over any cluster. In K-means clustering V_i is also called the cluster prototypes, its general form is

$$V_i = \frac{\sum_{K=1}^{N} X_K}{N_i} \qquad , X_K \in A_i$$

Where N_i is the number of objects in A_i.In K-medoid clustering the cluster centers are to the nearest objects to the mean of data in one cluster

$$V = \{V_i \in X | 1 \le i \le c\}.$$

B. Fuzzy C-means algorithm

Fuzzy C-means clustering (FCM), relies on the basic idea of Hard Clustering(HC), with the difference that in FCM each data point belongs to a cluster based on a degree of membership, while in HC every data point either it belongs or not to a certain cluster. So FCM employs fuzzy partitioning such that a given data point can belong to several groups with the degree of belongingness specified by membership grades between the range 0 and 1. However, FCM still uses a cost function that is to be minimized while trying to partition the data set.

The membership matrix U is allowed to have elements with values between 0 and 1. However, the summation of degrees of belongingness of a data point to all clusters is always equal to unity:

$$\sum_{i=1}^{c} \mu_{ij} \quad ,\forall\ j = 1 \dots n$$

The cost function for FCM is a generalization of Equation given for Hard clustering algorithms.

$$J(X;U,V) = \sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{ik})^m \|X_i - V_i\|^2$$

Where μ_{ik} is between 0 and 1; V_i is the cluster center of fuzzy group, $d_{ij}=\|X_i - V_i\|^2$ is the Euclidean distance between the ith cluster center and the jth data point; and m[1,∞) is a weighting exponent.

Where, $D_{ikA}^2 = \|x_k - v_i\|_A^2$=(xk-vi)TA(xk-vi) is a squared inner-product distance norm. The necessary conditions for the above equation to reach its minimum are:

$$V_i = \frac{\sum_{j=1}^{N}(\mu_{ij})^m X_j}{\sum_{j=1}^{N}(\mu_{ij})^m}$$

$$\text{And} \qquad \mu_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}}$$

The algorithm works iteratively through the preceding two conditions until the no more improvement is noticed. The FCM algorithm computes with the standard Euclidean distance norm, which induces hyper spherical clusters. Hence it can only detect clusters with the same shape and orientation.

C. The Gustafson-Kessel (GK) algorithm:

The basic problem of FCM algorithm is that it does not detect clusters of different shapes. In order to solve this problem Gustafson and Kessel extended the standard fuzzy c-means algorithm by employing an adaptive distance norm, in order to detect clusters of different geometrical shapes. Each cluster has its own norm-inducing matrix Ai, which yields the following inner-product norm:

$$D_{ikA}^2 = (Xk\text{-}vi)T\ Ai(xk\text{-}vi) \quad ,1 \le i \le c,$$
$$1 \le k \le N.$$

The matrices Ai are used as optimization variables in the c-means functional, thus allowing each cluster to adapt the distance norm to the local topological structure of the data. Let A denote a c-tuple of the norm-inducing matrices: A= (A1, A2 …Ac) the objective functional of the GK algorithm is defined by:

$$J(X; U, V, A) = \sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{ik})^m D_{ikA_i}^2$$

For any fixed A, fuzzy partition conditions can be directly applied. However, in the objective function given above, since it is linear in Ai it cannot be directly minimized w.r.t Ai. Which means J can be made as small as we want by simply making Ai less positive definite. Ai must be constrained in some way to get a feasible solution, the usual way to accomplish this is to constrain the determinant of Ai. Allowing the matrix Ai to vary with its determinant fixed corresponds to optimizing the cluster's shape while its volume remains constant:

$$\|A_i\| = \rho_i \ , \rho > 0$$

Where ρ_i is fixed for each cluster. Using the Lagrange multiplier method, the following expression for Ai is obtained:

$A_i = [\rho_i \det(F_i)]^{1/n} F_i^{-1}$ Where Fi is the fuzzy covariance matrix of the ith cluster defined by

$Fi = \frac{\sum_{k=1}^{N}(\mu_{ik})^m (x_k - v_i)(x_k - v_i)^T}{\sum_{k=1}^{N}(\mu_{ik})^m}$ Note that the substitution of F_i and A_i gives a generalized squared Mahalanobis distance norm between X_k and the cluster mean v_i, where the covariance is weighted by the membership degrees in U.

D. The Gath-Geva (GG) algorithm:

We have seen how GK algorithm has considerably extent its effort to solve FCM limitation. GG algorithm is yet another extension done on GK algorithm by taking the size and density of clusters into account. In here the Clustering algorithm, employs a distance norm based on the fuzzy maximum likelihood estimates, proposed by Bezdek and Dunn.

$$D_{ik}(x_k, v_i) = \frac{\sqrt{\det(F_{wi})}}{\alpha_i}\exp\left(\frac{1}{2}(x_k - v_i^{(l)})^T F_{wi}^{-1}(x_k - v_i^{(l)})\right)$$

In contrast to GK algorithm, this distance norm involves an exponential term thus decreases faster than the inner-product norm. F_{wi} denotes the fuzzy covariance matrix of the i-th cluster, given by:

$$F_{wi} = \frac{\sum_{k=1}^{N}(\mu_{ik})^w (x_k - v_i)(x_k - v_i)^T}{\sum_{k=1}^{N}(\mu_{ik})^w}, 1 \le i \le c$$

Where w = 1 in the original FMLE algorithm, but here we use the w = 2 weighting exponent, so as to compensate the exponential term of the distance norm and the partition to become more fuzzy. The difference between the matrix Fi in GK algorithm and the Fwi define above is that the latter does not involve the weighting exponent m, but instead consists of w. This is because the two weighted covariance matrices arise as generalizations of the classical covariance from two different concepts. The α_i is the prior probability of selecting cluster is given by

$$\alpha_i = \frac{1}{N}\sum_{k=1}^{N}\mu_{ik}$$

The membership degrees μ_{ik} are interpreted as the posterior probabilities of selecting the i-th cluster.

Algorithms:

A. K-Means algorithm:

Given the data set X, choose the number of clusters 1 < *c* < *N*.
Initialize with random cluster centers chosen from the data set.
Repeat for l=1,2,3…….
Step 1 Compute the distances
$D_{ikA}^2 = (x_k - v_i)^T A(x_k - v_i), 1 \le i \le c, 1 \le k \le N$

Step 2 Select the points for a cluster with the minimal distances,
they belong to that cluster.
Step 3 Calculate cluster centers

$$v_i^{(l)} = \frac{\sum_{i=1}^{N} x_i}{N_i}$$

Until

$$\prod_{k=1}^{n} max\left|v^{(l)} - v^{(l-1)}\right| \neq 0$$

Ending: Calculate the partition matrix.

B.K-Medoid Algorithm:

Given the data set X, choose the number of clusters $1 < c < N$.Initialize with random cluster centers chosen from the data set*X*.
Repeat for l= 1,2,3,…….
Step 1 Compute the distances
$D_{ikA}^{2*} = (x_k - v_i^*)^T A (x_k - v_i^*)$,

And $x_i^* = \mathrm{argmin}_i (D_{ik}^{2*}) v_i^{(l)} = x_i^*$

Step 2 Select the points for a cluster with the minimal distances,they belong to that cluster.
Step 3 Calculate fake cluster centers(the original K-means)

$$v_i^{(l)*} = \frac{\sum_{i=1}^{N} x_i}{N_i}$$

Step 4 Choose the nearest data point to be the cluster center
$D_{ikA}^2 = (x_k - v_i)^T A (x_k - v_i)$
Until $\prod_{k=1}^{n} max\left|v^{(l)} - v^{(l-1)}\right| \neq 0$
Ending Calculate the partition matrix

C. FCM Algorithm

Given the data set X, choose the number of clusters $1 < c < N$,
the weighting exponent $m > 1$, the termination tolerance $\epsilon > 0$
and the norm-inducing matrix A. Initialize the partition matrix randomly, such that$U^{(0)} \in M_{fc}$.
Repeat forl=1,2,3….
Step 1 Compute the cluster prototypes (means):

$$v_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m x_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \leq i \leq c$$

Step 2 Compute the distances:

$D_{ikA}^2 = (x_k - v_i)^T A (x_k - v_i), 1 \leq i \leq c, 1 \leq k \leq N$

Step 3 Update the partition matrix:

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} (D_{ikA} / D_{jkA})^{2/(m-1)}}.$$

Until $\left\|U^{(l)} - U^{(l-1)}\right\| < \epsilon$

D. GG Algorithm

Given a set of data *X* specify *c*, choose a weighting exponent $m > 1$ and a termination tolerance $\epsilon > 0$.
Initialize the partition matrix with a more robust method.
Repeat for $l = 1, 2, \ldots.$
Step 1 Calculate the cluster centers:

$$v_i^{(l)} = \frac{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^{\omega} X_k}{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^{\omega}}, 1 \leq i \leq c$$

Step 2 Compute the distance measure D_{ik}^2 The distance to the prototype is calculated based the fuzzyCovariance matrices of the cluster.

$$F_i^l = \frac{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^{\omega} (X_k - V_{il})(X_k - V_{il})^T}{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^{\omega}}, 1 \leq i \leq c$$

$$D_{ik}^2 (X_k, v_i) = \frac{(2\pi)^{\left(\frac{n}{2}\right)} \sqrt{\det(F_i)}}{\alpha_i} \exp\left(\frac{1}{2}\left(X_k - v_i^{(l)}\right)^T F_i^{-1} \left(X_k - v_i^{(l)}\right)\right)$$

with the *a priori* probability

$\alpha_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{ik}$
Step 3 Update the partition matrix

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c} (D_{ik}(X_k, V_i) / D_{jk}(X_k, V_j))^{2/(m-1)}}, 1 \leq i \leq c, 1 \leq k \leq N$$

Until

$$\left\|U^{(l)} - U^{(l-1)}\right\| < \epsilon$$

E.GK Algorithm

Given the data set X, choose the number of clusters $1 < c < N$,
the weighting exponent $m > 1$, the termination tolerance $\epsilon > 0$
and the norm-inducing matrix A.
Initialize the partition matrix randomly, such that $U^{(0)} \in M_{fc}$.
Repeat for l= 1, 2,3…….
Step 1 Calculate the cluster centers.

$$v_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m x_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \leq i \leq$$

Step 2 Compute the cluster covariance matrices.

$F_i^{(l)} = \frac{\sum_{k=1}^{N}(\mu_{ik}^{(l-1)})^m (x_k - v_i^{(l)})(x_k - v_i^{(l)})^T}{\sum_{k=1}^{N}(\mu_{ik}^{(l-1)})^m}$, $1 \leq i \leq c$

Add a scaled identity matrix: $F_i = (1-\gamma)F_i + \gamma (F_0)^{1/n} I$

Extract eigenvalue λ_{ij} and eigenvectors ϕ_{ij} ,Find $\lambda_{i,max} = max_j \lambda_{i,j}$ and set

$\lambda_{i,max} = \lambda_{ij}/\beta, \forall j$ for which $\lambda_{i,max}/\lambda_{i,j} \geq \beta$

Reconstruct *Fi* by:

$F_i = [\emptyset_{i,1} \ldots\ldots \emptyset_{i,n}] \mathrm{diag}(\lambda_{i,1} \ldots\ldots \lambda_{i,n}) [\emptyset_{i,1} \ldots\ldots \emptyset_{i,n}]^{-1}$

Step 3 Compute the distances.

$D^2_{ikA_i}(X_k, V_i) = (x_k - v_i^{(l)})^T [(\rho_i \det(F_i))^{1/n} F_i^{-1}] (X_k - V_i^{(l)})$

Step4 Udpate the partition matrix

$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c}(D_{ikA_i}(X_k, V_i)/D_{jk}(X_k, V_j))^{2/(m-1)}}$, $1 \leq i \leq c, 1 \leq k \leq N$

Until $\|U^{(l)} - U^{(l-1)}\| < \epsilon$

V. Validation Parameter:

Cluster validity refers to the alternate solution to find out whether a given clustering partition fits to the data well or not. The clustering algorithm always tries to find the best fit for a fixed number of clusters and the parameterized cluster shapes. However this does not mean that even the best fit is meaningful for all cases. Either the number of clusters might be wrong or the cluster shapes might not correspond to the groups in the data, if the data can be grouped in a meaningful way at all. There is different scalar validity measures have been proposed in the literature, and one should be clear none of them is perfect by oneself; therefore we used several of those validity measures known as indexes in our validation.

1. Partition Coefficient (PC): measures the amount of "overlapping" between clusters. It is defined by Bezdek as follows:

PC(c) $= \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} (\mu_{ij})^2$, where μ_{ij} is the membership of data point j in cluster i.The disadvantage of PC is lack of direct connection to some property of the data themselves. The optimal number of cluster is at the maximum value.

2. Classification Entropy (CE): it measures the fuzziness of the cluster partition only, which is similar to the Partition Coefficient

$$CE(c) = -\frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij} \log(\mu_{ij})$$

.**3. Partition Index (SC)**: is the ratio of the sum of compactness and separation of the clusters. It is a sum of individual cluster validity measures normalized through division by the fuzzy cardinality of each cluster.

$SC(c) = \sum_{i=1}^{c} \frac{\sum_{j=1}^{N}(\mu_{ij})^m \|(x_j - v_i)\|^2}{N_i \sum_{k=1}^{c}(\mu_{ij})^m \|(v_k - v_i)\|^2}$, SC is useful when comparing different partitions having equal number of clusters. A lower value of SC indicates a better partition.

4. Separation Index (S): on the contrary of partition index (SC), the separation index uses a minimum-distance separation for partition validity.

$$S(c) = \frac{\sum_{i=1}^{c} \sum_{j=1}^{N} \sum_{j=1}^{N} (\mu_{ij})^2 \|(x_j - v_i)\|^2}{N_{\min_{i,k}} \|(v_k - v_i)\|^2}$$

5. Dunn's Index (DI): this index is originally proposed to use at the identification of "compact and well separated clusters". So the result of the clustering has to be recalculated as it was a hard partition algorithm.

$$DI(c) = \min_{i \in c}\left\{\min_{j \in c, i \neq j}\left\{\frac{\min_{x \in C_i, y \in C_j} d(x,y)}{\max_{k \in c}\{\max_{x,y \in C} d(x,y)\}}\right\}\right\},$$

The main drawback of Dunn's index is computational since calculating becomes computationally very expansive as c and N increase.

VI. Implementation and Results:

As now we have a general idea of all the Hard as well as Fuzzy clustering, which include their basic mathematical foundations. We now turn our discussion to all these techniques on the basis of practical study. This study includes the practical implementation of all the five clustering techniques and testing each one of them on a set of data called motorcycledataset.

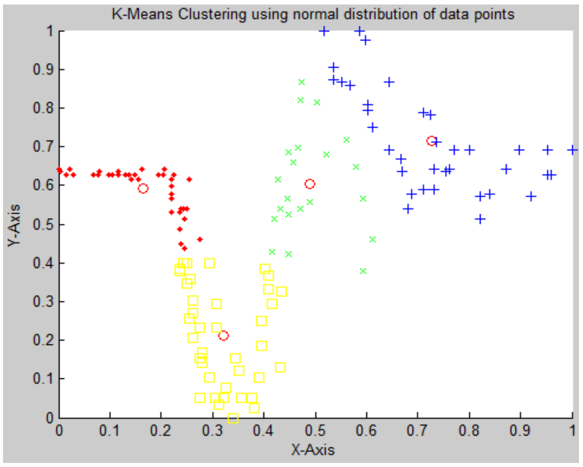

Fig-1 Clusters using K-means Algorithm

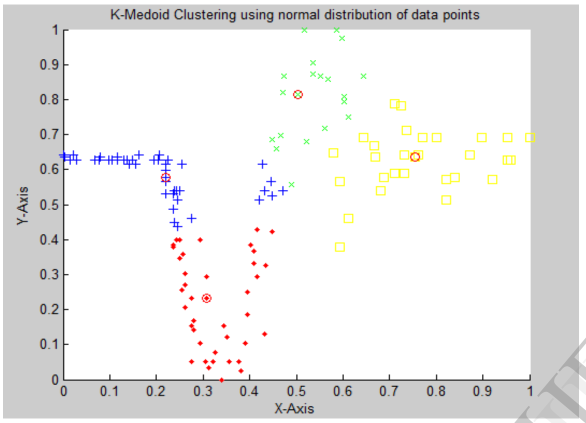

Fig-2 Clusters using K-medoid Algorithm

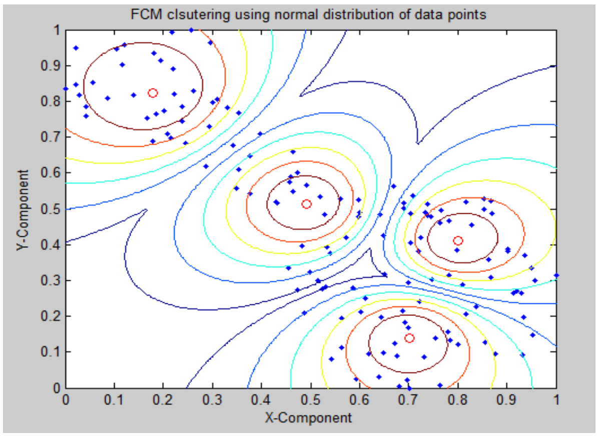

Fig-3 Cluster using FCM Algorithm

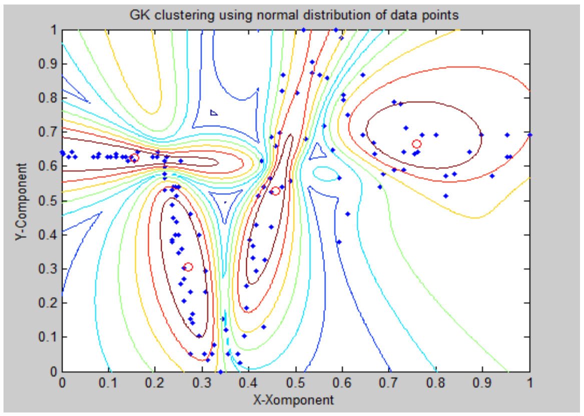

Fig-4 Clusters using GK Algorithm

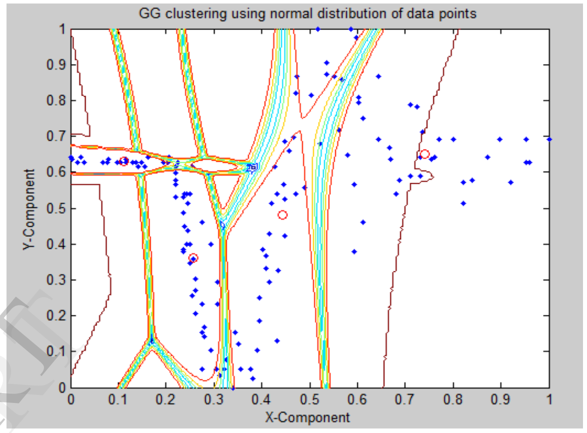

Fig-5 clusters using GG Algorithm

Table-1Comparision of Index parameters for different algorithms

	PC	CE	SC	S	DI
K-Means	1	NaN	0.085	0.0001	0.0129
K-Medoid	1	NaN	0.2384	0.0002	0.0031
FCM	0.8152	0.3480	0.9157	0.0007	0.0185
GK	0.8255	0.3278	0.8890	0.0007	0.0083
GG	0.9837	0.0287	2.2349	0.0020	0.0153

Table-2 Comparison of Speed and Accuracy for different algorithm

CLUSTERING ALGORITHM	Speed in terms of Number of Iteration	Root Mean Square Error (RMSE)	Accuracy
K-Means	3	0.443	81%
K-Medoid	2	0.441	78%
FCM	42	0.437	69%
GK	70	0.437	77%
GG	27	0.439	79%

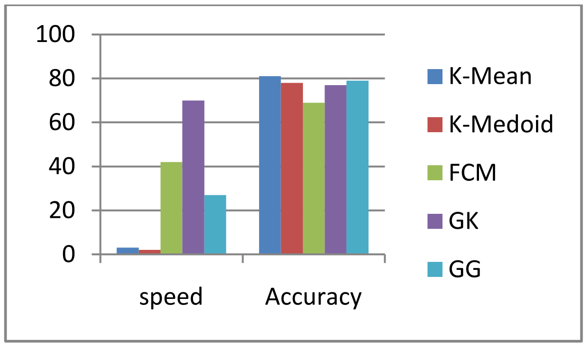

Fig-6 Comparison of speed and accuracy

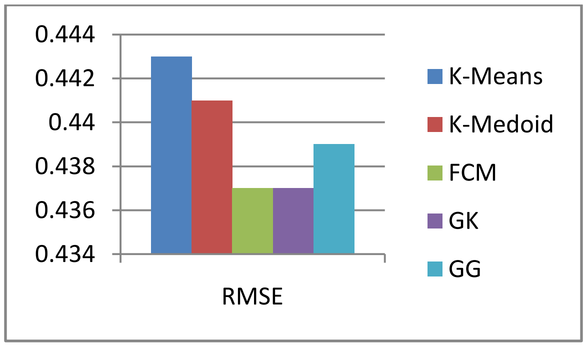

Fig-7 Comparison of RMSE

As mentioned earlier the similarity norm that is used is purely mathematical distance norm. However because most similarity metrics are very sensitive to large ranges of elements in the input vectors, so each of the input variable is normalized first within the unit interval [0, 1] or hypercube. After normalization each algorithm is tested on the normalized data set. The resultant cluster is then evaluated using the parameters mentioned above. The analysis of each of the algorithm is given below with some of their remarks:

K-Means runs considerably very fast by looking at the number of iteration it takes from above table 2, than other algorithms, but one limitation it has is that it is very sensitive to initial centroids. That is why it is recommended to run the algorithm repeatedly to yield better the best results. Moreover the value of K still needs to be given by the user however regarding accuracy it excel as compared with other algorithms.

K-Medoid performs almost same as that of K-Means algorithm. It's speed and accuracy are fairly encouraging and needless to say it is very very fast.

Coming to FCM it starts normally like the K-Means algorithm by assigning random values to the membership matrix U. Looking at the fig 3 above we can see that FCM can detect only clusters of circular shapes however the clusters are a little bit elongated because of the direct affect of one cluster over the other. In table 2 we can see it is comparatively slower than the Hard algorithms accuracy seems to be a problem too. Even here the number of clusters is not known before hand.

On running Gustafson-Kessel from fig 4 we can see it detect clusters not only of spherical shaped but of different geometrical shapes as well. The Mahalanobis distance used here adapt the topological structure because each cluster is forces to have its own norm inducing matrixA_i. since it needs to be initialized by PCA it takes more time and hence slower however accuracy seems to better than that of PCA with same RMSE.

Gath-Geva algorithm uses the PCA for initialization like Gk. However because of the exponential term in the distance norm, it decreases faster by increasing $|X_k - V_i|$ distance that is fairly visible from table 2 by looking at the number of iteration it takes.. In Fig. 4 The GG divides the data point into several disjoint clusters but the effects of the other "cluster-borders" distort this disjointness. It's accuracy is better than all the above fuzzy algorithms mentioned above.

VII. Comparative Analysis based on index Parameters:

From fig 1 to fig 5 shown above the various index values mentioned in Section 1V can be easily demarcated and compared for each clustering methods. All the validity measures are collected in Tab. 1. It must be noted, since all algorithms use random initialization, so different running's issue results in different partition results, i.e. values of the validation measures. On the other hand the results hardly depend on the structure of the data, and no validity index is perfect by itself for a clustering problem. The figures show that hard clustering methods can find a good solution for the clustering problem, when it is compared with the figures of fuzzy clustering algorithms. On the contrary in Fig.1 and Fig.2 one can see a typical example of the initialization problem of hard clustering. This caused the differences between the validity index values in Tab.1 PC and CE is useless for K-means and K-medoid, because they are hard clustering methods. But SC,S, and DI are useful to validate crisp and well separated clusters.. The only difference between Fig. 3 and Fig. 4 stands in the shape of the clusters, where the Gustafson-Kessel algorithm can find the elongated clusters better obvious by looking at the parameters in tab.1. Fig.5 shows that the Gath Geva algorithm returned with a result of three subspaces.

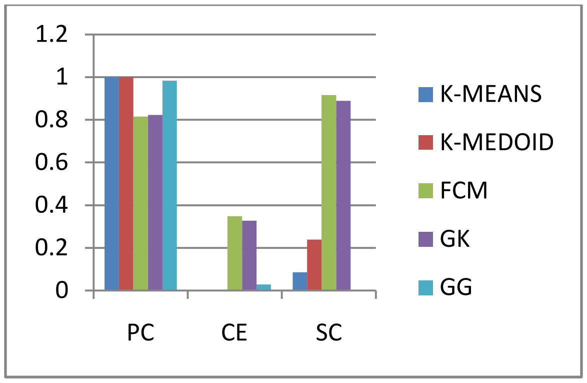

Fig-8Comparison of PC, CE and SC for different algorithms

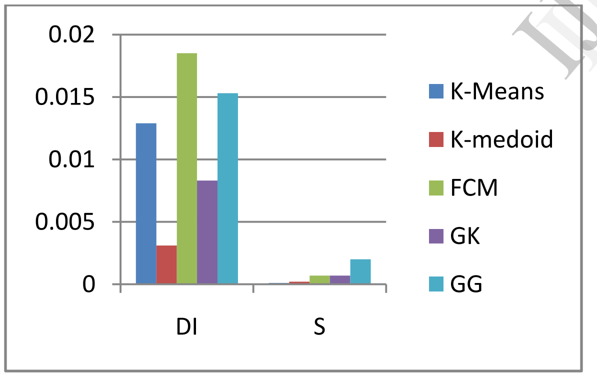

Fig-9 Comparison of DI and S for different algorithms

IX. Conclusion:

In this paper a brief review has been done on the various Hard and fuzzy clustering algorithm. It is worth knowing that each of them has their own share of pros and cons. Even though at times it may seems that Hard clustering algorithm may seems to have an upper hand regarding speed and other issues such as accuracy but it is important to keep in mind about the importance of fuzziness as well when it comes to application in real life. We have made efforts in the analyzing each clustering algorithm even in terms of index parameters and how they help in describing and explain the cluster behavior.

REFFERNCES:-

1. J. C. Bezdek. "Pattern Recognition with Fuzzy Objective Function Algorithms**".** Plenum Press, 1981.
2. F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. "Fuzzy Cluster Analysis*"*. Wiley, Chichester, 1999.
3. R. Babuska. "Fuzzy Modeling for Control*"*. Kluwer Academic Publishers,Boston, 1998.
4. D. E. Gustafson and W.C. Kessel. "Fuzzy clustering with fuzzy covariance matrix In a mixture of normal dustrubutions". IEEETransactions on Computers, pp. 835-838, 1975.
5. T. Velmurugan and T.Santhanam " Computational complexity between K-means and K-Medoids Clustering algorithm for Normal and Uniform Distrubution of data points." Journal of Computer science 6(3): pp.363-368,2010
6. R. Babuska, P. J. van der Veen, and U. Kaymak. "Improved covariance estimation for Gustafson-Kessel clustering". IEEE International Conference on Fuzzy Systems, pp.1081-1085, 2002.
7. D .Napolean et al. "An efficient Clustering Technique to Reduce the time Complexity Of K-Means Using Normal Distribution of data points" International Journal Of Advanced Research in Computer Science, vol.2 ,no-1, pp. 491-494,2011
8. J. C. Bezdek and J.C. Dunn. "Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal distributions". IEEETransactions on Computers, pp. 835-838, 1975
9. I. Gath and A.B. Geva. "Unsupervised optimal fuzzy clustering". IEEE Trans-actions on Pattern Analysis and Machine Intelligence, vol.7, pp.773-781, 1989.
10. A. M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A.Arrington, and R.F. Murtagh. "Validity-guided (Re)Clustering with applications to image segmentation". IEEE Transactions on Fuzzy Systems, vol.4, pp. 112-123, 1996.