Volume 2, Issue 8, August 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Performance Analysis of Extended Shadow Clustering Techniques and Binary Data Sets Using K-Means Clustering

[1]**Dr.A.Senguttuvan,** Dean, Computer Sciences, SVEC, Tirupati, AP, India.
[2]**D.Pramodh Krishna,** Asst. Professor, CSE, SVEC, Tirupati, AP, India.
[3]**Dr.K.Venugopal Rao,** Professor, CSE, GNITS, Hyderabad, AP, India.

ABSTRACT: Data clustering is a process of putting similar data into groups. A clustering algorithm partitions a data set into several groups such that the similarity within a group is larger than among groups. This paper reviews five of the most representative off-line clustering techniques: K-means clustering, Fuzzy C-means clustering, Mountain clustering, Subtractive clustering and Extended Shadow Clustering. The techniques are implemented and tested against a medical problem of heart disease diagnosis. Performance and accuracy of the four techniques are presented and compared. General Terms: Data Mining, Image Processing, Artificial Intelligence.

Keywords — Clustering, Data clustering, K-means, Fuzzy mountain, subtractive. C-Means Clustering Extended Shadow Clustering.

I. INTRODUCTION

Data Clustering is considered an interesting approach for finding similarities in data and putting similar data into groups. Clustering partitions a data set into several groups such that the similarity within a group is larger than that among groups [1]. The idea of data grouping, or clustering, is simple in its nature and is close to the human way of thinking; whenever we are presented with a large amount of data, we usually tend to summarize this huge number of data into a small number of groups or categories in order to further facilitate its analysis. Moreover, most of the data collected in many problems seem to have some inherent properties that lend themselves to natural groupings. Nevertheless, finding these groupings or trying to categorize the data is not a simple task for humans unless the data is of low dimensionality.

Clustering algorithms are used extensively not only to organize and categorize data, but are also useful for data compression and model construction. By finding similarities in data, one can represent similar data with fewer symbols for example. Also if we can find groups of data, we can build a model of the problem based on those groupings.

Another reason for clustering is to discover relevance knowledge in data. Francisco Azuaje *et al.* [2] implemented a Case Based Reasoning (CBR) system based on a Growing Cell Structure (GCS) model. Data can be stored in a knowledge base that is indexed or categorized by cases; this is what is called a Case Base. Each group of cases is assigned to a certain category. Using a Growing Cell Structure (GCS) data can be added or removed based on the learning scheme used. Later when a query is presented to the model, the system retrieves the most relevant cases from the case base depending on how *close* those cases are to the query.

In this paper, five of the most representative off-line clustering techniques are reviewed:

- K-means (or Hard C-means) Clustering,
- Fuzzy C-means Clustering,
- Mountain Clustering,
- Subtractive Clustering and
- Extended Shadow Clustering.

These techniques are usually used in conjunction with radial basis function networks (RBFNs) and Fuzzy Modeling. Those four techniques are implemented and tested against a medical diagnosis problem for heart disease. The results are presented with a comprehensive comparison of the different techniques and the effect of different parameters in the process.

The remainder of the paper is organized as follows. Section II presents an overview of data clustering and the underlying concepts. Section III presents each of the four clustering techniques in detail along with the underlying mathematical foundations. Section IV introduces the implementation of the techniques and goes over the results of each technique, followed by a comparison of the results. A brief conclusion is presented in Section V. The MATLAB code listing of the five clustering techniques can be found in the appendix.

II. DATA CLUSTERING OVERVIEW

As mentioned earlier, data clustering is concerned with the partitioning of a data set into several groups such that the similarity within a group is larger than that among groups. This implies that the data set to be partitioned has to have an inherent grouping to some extent; otherwise if the data is uniformly distributed, trying to find clusters of data will fail, or will lead to artificially introduced partitions. Another problem that may arise is the overlapping of data groups. Overlapping groupings sometimes reduce the efficiency of the clustering method, and this reduction is proportional to the amount of overlap between groupings. Usually the techniques presented in this paper are used in conjunction with other sophisticated neural or fuzzy models. In particular, most of these techniques can be used as preprocessors for determining the initial locations for radial basis functions or fuzzy if-then rules.

The common approach of all the clustering techniques presented here is to find *cluster centers* that will represent each cluster. A cluster center is a way to tell where the heart of each cluster is located, so that later when presented with an input vector, the system can tell which cluster this vector belongs to by measuring a similarity metric between the input vector and al the cluster centers, and determining which cluster is the *nearest* or most similar one.

Some of the clustering techniques rely on knowing the number of clusters apriori. In that case the algorithm tries to partition the data into the given number of clusters. K-means and Fuzzy C-means clustering are of that type. In other cases it is not necessary to have the number of clusters known from the beginning; instead the algorithm starts by finding the first large cluster, and then goes to find the second, and so on. Mountain and Subtractive clustering are of that type. In both cases a problem of known cluster numbers can be applied; however if the number of clusters is not known, K-means and Fuzzy C-means clustering cannot be used. Another aspect of clustering algorithms is their ability to be implemented in on-line or offline mode. On-line clustering is a process in which each input vector is used to update the cluster centers according to this vector position. The system in this case *learns* where the cluster centers are by introducing new input every time. In off-line mode, the system is presented with a training data set, which is used to find the cluster centers by analyzing all the input vectors in the training set. Once the cluster centers are found they are fixed, and they are used later to classify new input vectors. The techniques presented here are of the off-line type. A brief overview of the four techniques is presented here. Full detailed discussion will follow in the next section.

The first technique is *K-means* clustering [6] (or *Hard C-means* clustering, as compared to *Fuzzy C-means* clustering.) This technique has been applied to a variety of areas, including image and speech data compression, [3, 4] data preprocessing for system modeling using radial basis function networks, and task decomposition in heterogeneous neural network architectures[5]. This algorithm relies on finding cluster centers by trying to minimize a cost function of dissimilarity (or distance) measure. The second technique is *Fuzzy C-means* clustering, which was proposed by Bezdek in 1973 [1] as an improvement over earlier Hard Cmeans clustering. In this technique each data point belongs to a cluster to a degree specified by a membership grade. As in K-means clustering, Fuzzy C-means clustering relies on minimizing a cost function of dissimilarity measure.

The third technique is *Mountain* clustering, proposed by Yager and Filev [1]. This technique builds calculates a mountain function (density function) at every possible position in the data space, and chooses the position with the greatest density value as the center of the first cluster. It then destructs the effect of the first cluster mountain function and finds the second cluster center. This process is repeated until the desired number of clusters have been found. The fourth technique is *Subtractive* clustering, proposed by Chiu [1]. This technique is similar to mountain clustering, except that instead of calculating the density function at every possible position in the data space, it uses the positions of the data points to calculate the density function, thus reducing the number of calculations significantly. The fifth technique is *Shadow* Clustering, proposed by Marco Muselli and Enrico Ferrari [2]. This technique is similar to Hamming Clustering, employs more refined heuristics for retrieving one at a time the logical products that contribute to form the resulting Boolean function. Different criteria can be adopted to pursue different targets for the synthesis. For example, if the number of operands is to be minimized, the Lowest Cube criterion must be used, whereas the Maximum covering Cube criterion has the aim of increasing the number of examples in the training set correctly classified by each AND operation.

III. DATA CLUSTERING TECHNIQUES

In this section a detailed discussion of each technique is presented. Implementation and results are presented in the following sections.

A. K-means Clustering

The K-means clustering, or Hard C-means clustering, is an algorithm based on finding data clusters in a data set such that a cost function (or an objection function) of dissimilarity (or distance) measure is minimized [1]. In most cases this dissimilarity measure is chosen as the Euclidean distance.

A set of n vectors $\mathbf{x}_j$, j =1,.., n , are to be partitioned into c groups G_i, i=1, .., c . The cost function, based on the Euclidean distance between a vector $\mathbf{x}_k$ in group j and the corresponding cluster center $\mathbf{c}_i$, can be defined by:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k,\, \mathbf{x}_k \in G_i} \|\mathbf{x}_k - \mathbf{c}_i\|^2 \right), \qquad (1)$$

where $J_i = \sum_{k,\, \mathbf{x}_k \in G_i} \|\mathbf{x}_k - \mathbf{c}_i\|^2$ is the cost function within group i.

The partitioned groups are defined by a c x *n* binary membership matrix **U**, where the element $\mathbf{u}_{ij}$ is 1 if the *j*th data point $\mathbf{x}_j$ belongs to group *i*, and 0 otherwise. Once the cluster centers $\mathbf{c}_i$ are fixed, the minimizing $\mathbf{u}_{ij}$ for Equation (1) can be derived as follows:

$$u_{ij} = \begin{cases} 1 \text{ if } \|\mathbf{x}_j - \mathbf{c}_i\|^2 \le \|\mathbf{x}_j - \mathbf{c}_k\|^2, \text{ for each } k \ne i, \\ 0 \text{ otherwise.} \end{cases} \qquad (2)$$

Which means that $\mathbf{x}_j$ belongs to group *i* if $\mathbf{c}_i$ is the closest center among all centers.

On the other hand, if the membership matrix is fixed, i.e. if $\mathbf{u}_{ij}$ is fixed, then the optimal center $\mathbf{c}_i$ that minimize Equation (1) is the mean of all vectors in group *i* :

$$\mathbf{c}_i = \frac{1}{|G_i|} \sum_{k,\, \mathbf{x}_k \in G_i} \mathbf{x}_k\,, \qquad (3)$$

where $|G_i|$ is the size of G_i, or $|G_i| = \sum_{j=1}^{n} u_{ij}$.

The algorithm is presented with a data set $\mathbf{x}_j$ =1, .., *n* ; it then determines the cluster centers $\mathbf{c}_i$ and the membership matrix **U** iteratively using the following steps:

Step 1: Initialize the cluster center, $\mathbf{c}_i$ =1, .., *c*. This is typically done by randomly selecting *c* points from among all of the data points.
Step 2: Determine the membership matrix **U** by Equation (2).
Step 3: Compute the cost function according to Equation (1). Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.
Step 4: Update the cluster centers according to Equation (3). Go to step 2.

The performance of the K-means algorithm depends on the initial positions of the cluster centers, thus it is advisable to run the algorithm several times, each with a different set of initial cluster centers. A discussion of the implementation issues is presented later in this paper.

B. Fuzzy C-means Clustering

Fuzzy C-means clustering (FCM), relies on the basic idea of Hard C-means clustering (HCM), with the difference that in FCM each data point belongs to a cluster to a degree of membership grade, while in HCM every data point either belongs to a certain cluster or not. 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 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:

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \forall j = 1, \ldots, n. \qquad (4)$$

The cost function for FCM is a generalization of Equation (1):

$$J(U,\mathbf{c}_1,\ldots,\mathbf{c}_c)=\sum_{i=1}^{c}J_i=\sum_{i=1}^{c}\sum_{j=1}^{n}u_{ij}^{m}d_{ij}^{2}\ , \qquad (5)$$

Where u_{ij} is between 0 and 1; $\mathbf{c}_i$ is the cluster center of fuzzy group *I*; $d_{ij} = ||\mathbf{c}_i - \mathbf{x}_j||$ is the Euclidean distance between the *i*th cluster center and the *j*th data point; and *m*□□1,□□ is a weighting exponent.

The necessary conditions for Equation (5) to reach its minimum are

$$\mathbf{c}_i=\frac{\sum_{j=1}^{n}u_{ij}^{m}\mathbf{x}_j}{\sum_{j=1}^{n}u_{ij}^{m}}\ , \qquad (6)$$

And

$$u_{ij}=\frac{1}{\sum_{k=1}^{c}\left(\dfrac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}}\ . \qquad (7)$$

The algorithm works iteratively through the preceding two conditions until the no more improvement is noticed. In a batch mode operation, FCM determines the cluster centers $\mathbf{c}_i$ and the membership matrix **U** using the following steps:

Step 1: Initialize the membership matrix **U** with random values between 0 and 1 such that the constraints in Equation (4) are satisfied.

Step 2: Calculate *c* fuzzy cluster centers, $\mathbf{c}_i$, $i= 1,\ldots,c$, using Equation (6).

Step 3: Compute the cost function according to Equation (5). Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

Step 4: Compute a new **U** using Equation (7). Go to step 2.

As in K-means clustering, the performance of FCM depends on the initial membership matrix values; thereby it is advisable to run the algorithm for several times, each starting with different values of membership grades of data points.

C. Mountain Clustering

The mountain clustering approach is a simple way to find cluster centers based on a density measure called the *mountain function*. This method is a simple way to find approximate cluster centers, and can be used as a preprocessor for other sophisticated clustering methods.

The first step in mountain clustering involves forming a grid on the data space, where the intersections of the grid lines constitute the potential cluster centers, denoted as a set *V*.

The second step entails constructing a mountain function representing a data density measure. The height of the mountain function at a point **v**□*V* is equal to

$$m(\mathbf{v})=\sum_{i=1}^{N}\exp\left(-\frac{\|\mathbf{v}-\mathbf{x}_i\|^2}{2\sigma^2}\right), \qquad (8)$$

where $\mathbf{x}_i$ is the *i*th data point and □ is an application specific constant. This equation states that the *data density* measure at a point **v** is affected by all the points $\mathbf{x}_i$ in the data set, and this density measure is inversely proportional to the distance between the data points $\mathbf{x}_i$ and the point under consideration **v**. The constant □□determines the height as well as the smoothness of the resultant mountain function.

The third step involves selecting the cluster centers by sequentially destructing the mountain function. The first cluster center $\mathbf{c}_1$ is determined by selecting the point with the greatest density measure. Obtaining the next cluster center requires eliminating the effect of the first cluster. This is done by revising the mountain function: a new mountain function is formed by subtracting a scaled Gaussian function centered at $\mathbf{c}_1$:

$$m_{\text{new}}(\mathbf{v})=m(\mathbf{v})-m(\mathbf{c}_1)\exp\left(-\frac{\|\mathbf{v}-\mathbf{c}_1\|^2}{2\beta^2}\right) \qquad (9)$$

The subtracted amount eliminates the effect of the first cluster. Note that after subtraction, the new mountain function new *m* (**v**) reduces to zero at **v** □ □$\mathbf{c}_1$. After subtraction, the second cluster center is selected as the point having the greatest value for the new mountain function. This process continues until a sufficient number of cluster centers is attained.

D. Subtractive Clustering

The problem with the previous clustering method, mountain clustering, is that its computation grows exponentially with the dimension of the problem; that is because the mountain function has to be evaluated at each grid point. Subtractive clustering solves this problem by using data points as the candidates for cluster centers, instead of grid points as in mountain clustering. This means that the computation is now proportional to the problem size instead of the problem dimension. However, the actual cluster centers are not necessarily located at one of the data points, but in most cases it is a good approximation, especially with the reduced computation this approach introduces.

Since each data point is a candidate for cluster centers, a *density measure* at data point $\mathbf{x}_i$ is defined as

$$D_i = \sum_{j=1}^{n} \exp\left(-\frac{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2}{(r_a/2)^2} \right), \qquad (10)$$

where r_a is a positive constant representing a neighborhood radius. Hence, a data point will have a high density value if it has many neighboring data points. The first cluster center $\mathbf{x}_{c1}$ is chosen as the point having the largest density value *c*1 *D*. Next, the density measure of each data point $\mathbf{x}_i$ is revised as follows:

$$D_i = D_i - D_{c_1} \exp\left(-\frac{\left\| \mathbf{x}_i - \mathbf{x}_{c_1} \right\|^2}{(r_b/2)^2} \right) \qquad (11)$$

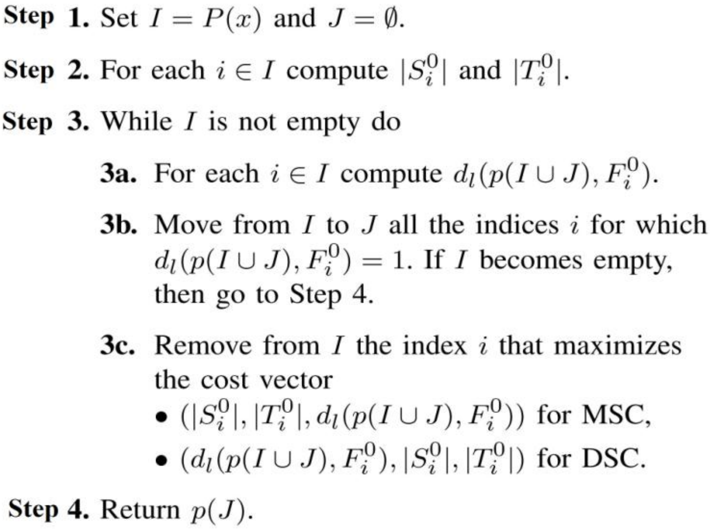
where r_b is a positive constant which defines a neighborhood that has measurable reductions in density measure. Therefore, the data points near the first cluster center *c*1 **x** will have significantly reduced density measure. After revising the density function, the next cluster center is selected as the point having the greatest density value. This process continues until a sufficient number of clusters is attainted.

E. Extended Shadow Clustering

The SC procedure offers a systematic way for performing the selection of the index i when the goal is to minimize a desired quality factor measuring the complexity of the final irredundant PDNF. In particular, the size of the antichain A and the number of literals □$_{a□A}$P(a) are two important quantities that must be minimized to make the resulting positive Boolean function simpler.

To pursue this aim, SC adopts two possible criteria: the first one, called Maximum covering Shadow Clustering (MSC), tries to increase the number of points in S and T covered by every generated bottom point; the second one, called Deepest Shadow Clustering (DSC), manages to reduce the degree of the prime implicant produced. Other quality factors can be minimized through SC by employing different ad hoc criteria; an analysis of some interesting choices will be the subject of a forthcoming paper. It should be noted that the goal of MSC is very important when the algorithm for pdpBf reconstruction is used in the solution of machine learning problems. In fact, any bottom point with high covering represents a relevant rule inferred from data.

The steps involved in this technique are as follows:



```
Step 1. Set $I = P(x)$ and $J = \emptyset$.
Step 2. For each $i \in I$ compute $|S_i^0|$ and $|T_i^0|$.
Step 3. While $I$ is not empty do
    3a. For each $i \in I$ compute $d_l(p(I \cup J), F_i^0)$.
    3b. Move from $I$ to $J$ all the indices $i$ for which
        $d_l(p(I \cup J), F_i^0) = 1$. If $I$ becomes empty,
        then go to Step 4.
    3c. Remove from $I$ the index $i$ that maximizes
        the cost vector
        • $(|S_i^0|, |T_i^0|, d_l(p(I \cup J), F_i^0))$ for MSC,
        • $(d_l(p(I \cup J), F_i^0), |S_i^0|, |T_i^0|)$ for DSC.
Step 4. Return $p(J)$.
```

Where MSC is Maximum covering Shadow Clustering and DSC is Deepest Shadow Clustering.

The lower distance $d_l(x,y)$ between x and y is defined as

$$d_l(x,y) = \sum_{i=1}^{n} |x_i - y_i|_+$$

where

$$|z|_+ = \begin{cases} z, & \text{if } z \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

In the same way, the upper distance $d_u(x,y)$ between x and y is given by

$$d_u(x,y) = \sum_{i=1}^{n} |y_i - x_i|_+$$

IV. IMPLEMENTATION AND RESULTS:

Having introduced the different clustering techniques and their basic mathematical foundations, we now turn to the discussion of these techniques on the basis of a practical study.This study involves the implementation of each of the four techniques introduced previously, and testing each one of them on a set of medical data related to heart disease diagnosis problem. The medical data used consists of 13 input attributes related to clinical diagnosis of a heart disease, and one output attribute which indicates whether the patient is diagnosed with the heart disease or not. The whole data set consists of 300 cases. The data set is partitioned into two data sets: two-thirds of the data for training, and one- third for evaluation. The number of clusters into which the data set is to be partitioned is two clusters; i.e. patients diagnosed with the heart disease, and patients not diagnosed with the heart disease. Because of the high number of dimensions in the problem (13-dimensions), no visual representation of the clusters can be presented; only 2-D or 3-D clustering problems can be visually inspected. We will rely heavily on performance measures to evaluate the clustering techniques rather than on visual approaches.

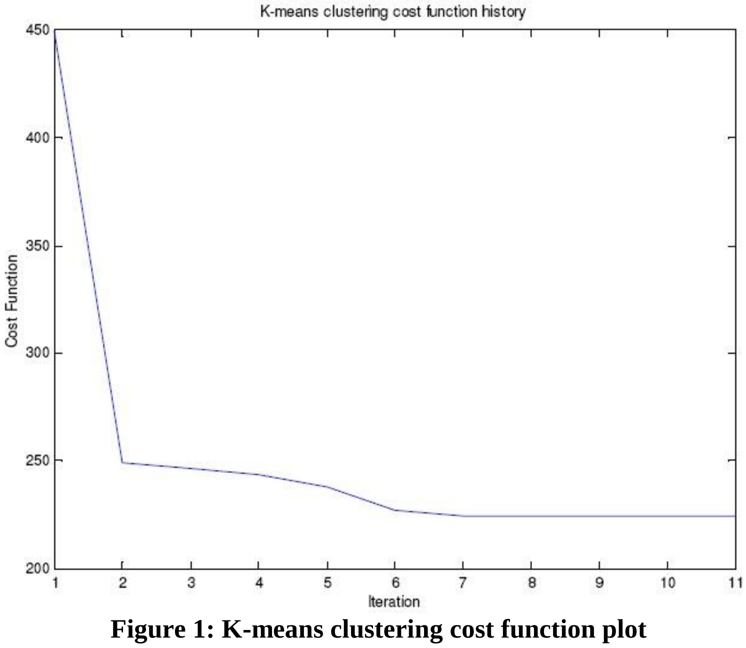


Figure 1: K-means clustering cost function plot

To further measure how accurately the identified clusters represent the actual classification of data, a regression analysis is performed of the resultant clustering against the original classification. Performance is considered better if the regression line slope is close to 1. Figure 2 shows the regression analysis of the best test case.

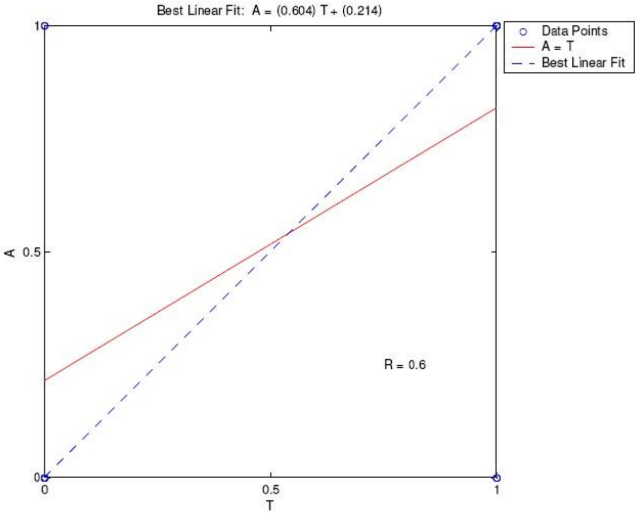


Figure 2: Regression Analysis of K-means Clustering

As seen from the results, the best case achieved 80% accuracy and an RMSE of 0.447. This relatively moderate performance is related to the high dimensionality of the problem; having too much dimensions tend to disrupt the coupling of data and introduces overlapping in some of these dimensions that reduces the accuracy of clustering. It is noticed also that the cost function converges rapidly to a minimum value as seen from the number of iterations in each test run.

However, this has no effect on the accuracy measure.

B. Fuzzy C-means Clustering

FCM allows for data points to have different degrees of membership to each of the clusters; thus eliminating the effect of hard membership introduced by K-means clustering. This approach employs fuzzy measures as the basis for membership matrix calculation and for cluster centers identification. As it is the case in K-means clustering, FCM starts by assigning random values to the membership matrix **U**, thus several runs have to be conducted to have higher probability of getting good performance. However, the results showed no (or insignificant) variation in performance or accuracy when the algorithm was run for several times. For testing the results, every vector in the evaluation data set is assigned to one of the clusters with a certain degree of belongingness (as done in the training set). However, because the output values we have are crisp values (either 1 or 0), the evaluation set degrees of membership are defuzzified to be tested against the actual outputs. The same performance measures applied in K-means clustering will be used here; however only the effect of the weighting exponent *m* is analyzed, since the effect of random initial membership grades has insignificant effect on the final cluster centers. Table 2 lists the results of the tests with the effect of varying the weighting exponent *m*. It is noticed that very low or very high values for *m* reduces the accuracy; moreover high values tend to increase the time taken by the algorithm to find the clusters. A value of 2 seems adequate for this problem since it has good accuracy and requires less number of iterations. Figure 3 shows the accuracy and number of iterations against the weighting factor.

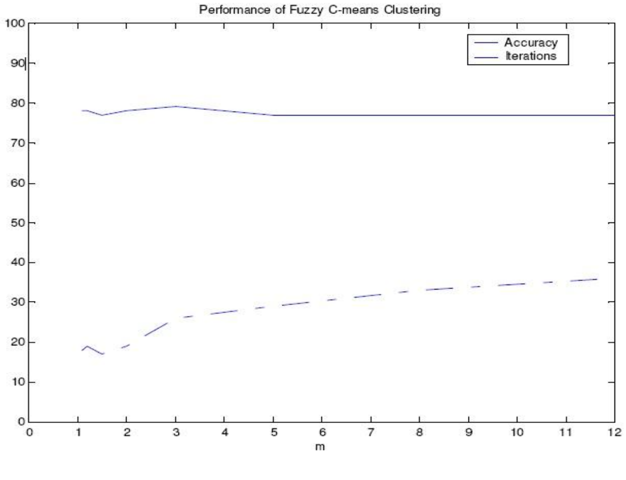


Figure 3: Fuzzy C-means Clustering Performance

In general, the FCM technique showed no improvement over the K-means clustering for this problem. Both showed close accuracy; moreover FCM was found to be slower than K-means because of fuzzy calculations.

C. Mountain Clustering

Mountain clustering relies on dividing the data space into grid points and calculating a mountain function at every grid point. This mountain function is a representation of the density of data at this point.
The performance of mountain clustering is severely affected by the dimension of the problem; the computation needed rises exponentially with the dimension of input data because the mountain function has to be evaluated at each grid point in the data space. For a problem with *c* clusters, *n* dimensions, *m* data points, and a grid size of *g* per dimension, the required number of calculations is:

$$N = m´ gn + (c -1)gn \quad \text{-------------(12)}$$

So for the problem at hand, with input data of 13- dimensions, 200 training inputs, and a grid size of 10 per dimension, the required number of mountain function calculation is approximately $2.01*10^{15}$ calculations. In addition the value of the mountain function needs to be stored for every grid point for later calculations in finding subsequent clusters; which requires g^n storage locations, for our problem this would be 10^{13} storage locations. Obviously this seems impractical for a problem of this dimension.

In order to be able to test this algorithm, the dimension of the problem have to be reduced to a reasonable number; e.g. 4-dimensions. This is achieved by randomly selecting 4 variables from the input data out of the original 13 and performing the test on those variables. Several tests involving differently selected random variables are conducted in order to have a better understanding of the results. Table 3 lists the results of 10 test runs of randomly selected variables. The accuracy achieved ranged between 52% and 78% with an average of 70%, and average RMSE of 0.546. Those results are quite discouraging compared to the results achieved in K-means and FCM clustering. This is due to the fact that not all of the variables of the input data contribute to the clustering process; only 4 are chosen at random to make it possible to conduct the tests. However, with only 4 attributes chosen to do the tests, mountain clustering required far much more time than any other technique during the tests; this is because of the fact that the number of computation required is exponentially proportional to the number of dimensions in the problem, as stated in Equation (12). So apparently mountain clustering is not suitable for problems of dimensions higher than two or three.

D. Subtractive Clustering

This method is similar to mountain clustering, with the difference that a density function is calculated only at every data point, instead of at every grid point. So the data points themselves are the candidates for cluster centers.

This has the effect of reducing the number of computations significantly, making it linearly proportional to the number of input data instead of being exponentially proportional to its dimension. For a problem of *c* clusters and *m* data points, the required number of calculations is:

$$N □ □m^2 □ □(c □1)m \text{------------------(13)}$$

As seen from the equation, the number of calculations does not depend on the dimension of the problem. For the problem at hand, the number of computations required is in the range of few ten thousands only.

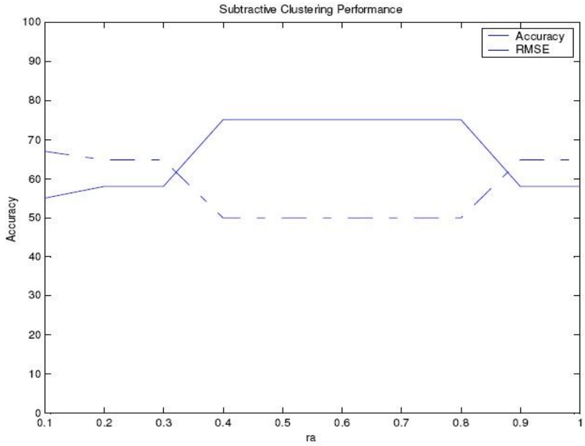


Figure 4: Subtractive Clustering Performance

Since the algorithm is fixed and does not rely on any randomness, the results are fixed. However, we can test the effect of the two variables r_a and r_b on the accuracy of the algorithm. Those variables represent a radius of neighborhood after which the effect (or contribution) of other data points to the density function is diminished. Usually the r_b variable is taken to be as 1.5 r_a. Table 4 shows the results of varying r_a . Figure 4 shows a plot of accuracy and RMSE against r_a.
It is clear from the results that choosing r_a very small or very large will result in a poor accuracy because if *a r* is chosen very small the density function will not take into account the effect of neighboring data points; while if taken very large, the density function will be affected account all the data points in the data space. So a value between 0.4 and 0.7 should be adequate for the radius of neighborhood. As seen from table 4, the maximum achieved accuracy was 75% with an RMSE of 0.5. Compared to K-means and FCM, this result is a little bit behind the accuracy achieved in those other techniques.

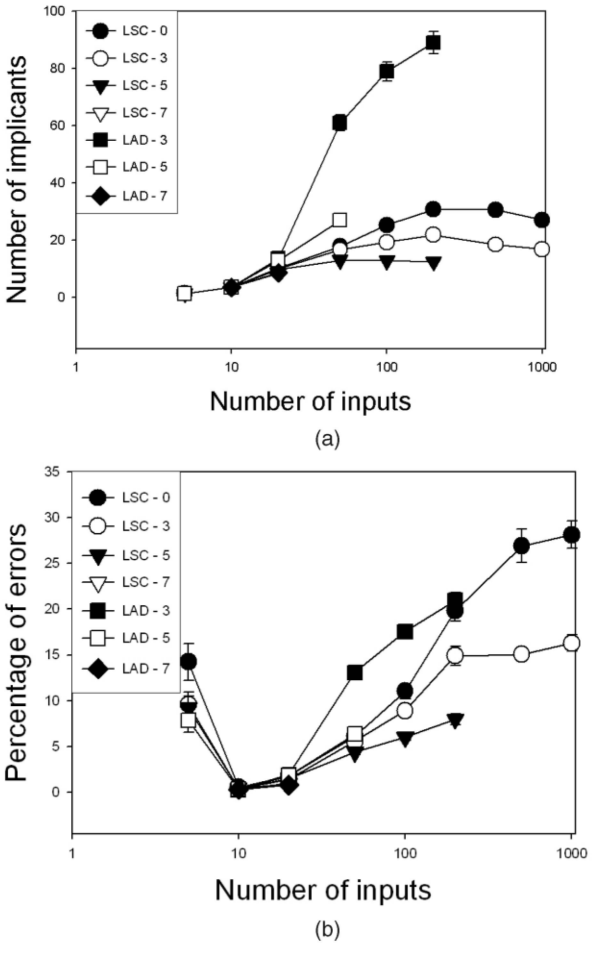
E. Extended Shadow Clustering

To measure the performance achieved by LSC in reconstructing a pdpBf, two different experiments have been carried out. The first one concerns the ability of retrieving the optimal expression of a positive Boolean function f, whose PDNF is randomly generated. In particular, we want to measure the difference between the considered algorithms LAD and LSC, in terms of complexity of the generated solution and computational cost required to produce the irredundant PDNF.

The second experiment evaluates the possibility of using SC in the synthesis of general Boolean functions. In fact, every mapping g: $\{0, 1\}^n$ -> $\{0, 1\}$ can be realized through a positive Boolean function f : $\{0, 1\}^{2n}$->$\{0, 1\}$ by applying the transformation □:$\{0, 1\}^n$ ->$\{0, 1\}^{2n}$ defined as

$$\square_{2i-1}(x) = 1-x_i,\ \square_{2i}(x) = x_i \qquad (14)$$

for i = 1, . . . , n, where □$_j$(x) is the jth bit of the binary string □(x) having length 2n. Once f has been generated, the associated Boolean function g is simply obtained by setting g(x) = f(□((x)) for every x □ $\{0, 1\}^n$, where□□□is given by (14). In this way, every prime implicant for f can be directly transformed into a minterm for g.



(a)

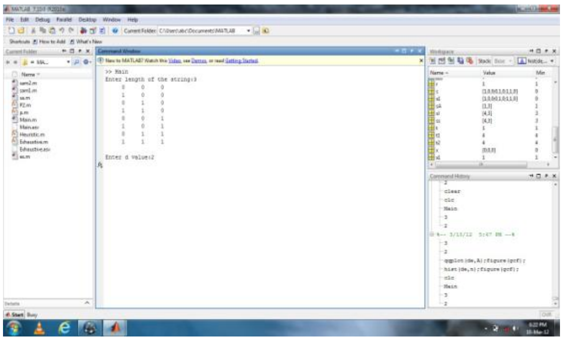
(b)

Thus, LSC can be used for reconstructing a Boolean function g from two sets of binary strings T’ and F’, with T’ □ F’=□, where we want to have g(x) = 1 for every x □ T’ and g(x) = 0 for every x □ F’. The first action to be performed is translating T’ and F’, which include binary strings with length n, into two sets T and F, containing elements with 2n bits, by applying the mapping □. It can be shown that the resulting set T is lower separated from F.

Target Boolean functions are generated by choosing, at random, the minimum true points to be included into the antichain A of the irredundant PDNF (2). By adopting proper constraints, the complexity of the resulting expression is controlled so as to avoid too simple or too specialized functions. This kind of functions allows a fair evaluation of the quality of the results produced by ESC and LAD. In addition, they lead to reconstruction problems similar to those arising from real-world applications, particularly in the machine learning field.

The performance of SC in reconstructing general Boolean functions is compared with those of the standard LAD algorithm. All the experiments have been carried out on a personal computer with a Pentium 4 (CPU 2.80 GHz, RAM 480 MB) running under the Windows XP operating system.

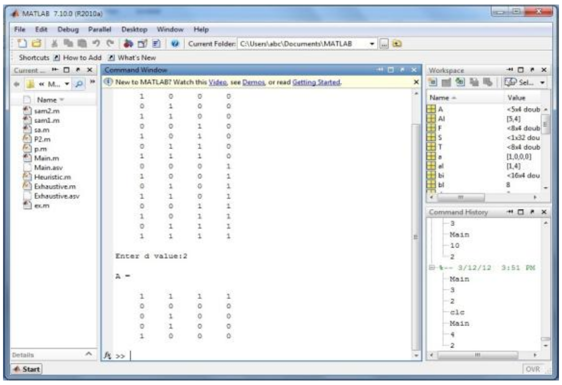
Results



```
>> Main
Enter length of the string:3
     0     0     0
     1     0     0
     0     1     0
     1     1     0
     0     0     1
     1     0     1
     0     1     1
     1     1     1

Enter d value:2
```

Fig: Input Screen

Clustering on Binary Data Sets using Extended Shadow Clustering 21

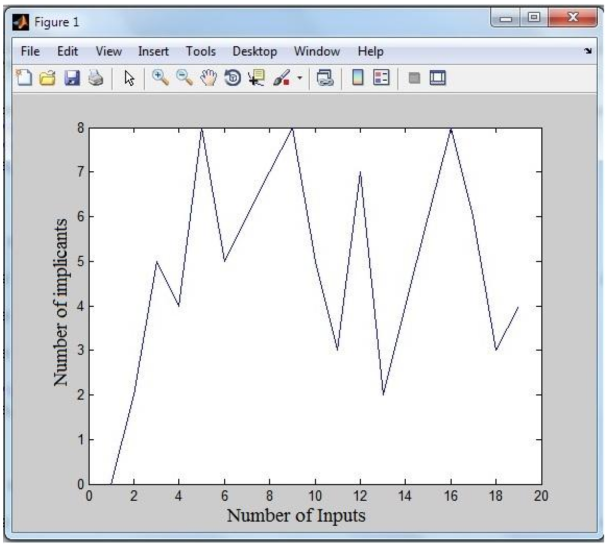


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\abc\Documents\MATLAB
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Command History

```
     1     0     0     0
     0     1     0     0
     1     1     0     0
     0     0     1     0
     1     0     1     0
     0     1     1     0
     1     1     1     0
     0     0     0     1
     1     0     0     1
     0     1     0     1
     1     1     0     1
     0     0     1     1
     1     0     1     1
     0     1     1     1
     1     1     1     1

Enter d value:2

A =

     1     1     1     1
     0     0     0     0
     0     1     0     0
     0     1     0     0
     1     0     0     0

>>
```

Fig: Generating Prime Implicant A

Clustering on Binary Data Sets using Extended Shadow Clustering 23



V. CONCLUSION

Five clustering techniques have been reviewed in this paper, namely: K-means clustering, Fuzzy C-means clustering, Mountain clustering, Subtractive clustering and Extended Shadow Clustering. These approaches solve the problem of categorizing data by partitioning a data set into a number of clusters based on some similarity measure so that the similarity in each cluster is larger than among clusters. The five methods have been implemented and tested against a data set for medical diagnosis of heart disease. The comparative study done here is concerned with the accuracy of each algorithm, with care being taken toward the efficiency in calculation and other performance measures. The medical problem presented has a high number of dimensions, which might involve some complicated relationships between the variables in the input data. It was obvious that mountain clustering is not one of the good techniques for problems with this high number of dimensions due to its exponential proportionality to the dimension of the problem. K-means clustering seemed to over perform the other techniques for this type of problem. However in other problems where the number of clusters is not known, Kmeans and FCM cannot be used to solve this type of problem, leaving the choice only to mountain or subtractive clustering. Subtractive clustering seems to be a better alternative to mountain clustering since it is based on the same idea, and uses the data points as cluster centers candidates instead of grid points; however, mountain clustering can lead to better results if the grid granularity is small enough to capture the potential cluster centers, but with the side effect of increasing computation needed for the larger number of grid points.

This Shadow Clustering decrease in computational cost is not achieved at the expense of a worse final solution. In contrast, positive Boolean functions generated by LSC exhibit, in general, a lower complexity and a higher accuracy than those built with LAD. This improvement is essentially due to the employment of SC in the heuristic part of the procedure and degrades when d increases. Note that in most applications, the high input dimension n forces the choice of low values for d; in these cases, the adoption of SC leads to resulting functions with better quality. However, the adoption of an exhaustive part improves the accuracy of the solution, as pointed out by the comparisons.

Finally, the clustering techniques discussed here do not have to be used as stand-alone approaches; they can be used in conjunction with other neural or fuzzy systems for further refinement of the overall system performance.

6. REFERENCES

[1] R.F. Ling, Comm. ACM, vol. 16, pp. 355-361, 1973, “A Computer Generated Aid for Cluster Analysis,”

[2] J. Huband, J.C. Bezdek, and R. Hathaway, Pattern Recognition, vol. 38, no. 11, pp. 1875-1886, 2005, “bigVAT: Visual Assessment of Cluster Tendency for Large Data Sets”.

[3] R. Hathaway, J.C. Bezdek, and J. Huband, Pattern Recognition, vol. 39, pp. 1315-1324, 2006, “Scalable Visual Assessment of Cluster Tendency”.

[4] W.S. Cleveland, Visualizing Data. Hobart Press, 1993. [6] J.C. Bezdek, R.J. Hathaway, and J. Huband, IEEE Trans. Fuzzy Systems, vol. 15, no. 5, pp. 890-903, 2007, “Visual Assessment of Clustering Tendency for Rectangular Dissimilarity Matrices”.

[5] R.C. Gonzalez and R.E. Woods, Prentice Hall, 2002, Digital Image Processing.

[6] I. Dhillon, D. Modha, and W. Spangler, Proc. 30th Symp. Interface: Computing Science and Statistics, 1998, “Visualizing Class Structure of Multidimensional Data”.

[7] T. Tran-Luu, PhD dissertation, Univ. of Maryland, College Park, 1996, “Mathematical Concepts and Novel Heuristic Methods for Data Clustering and Visualization”.

[8] J.C. Bezdek and R. Hathaway, Proc. Int’l Joint Conf. Neural Networks (IJCNN ’02), pp. 2225-2230, 2002, “VAT: A Tool for Visual Assessment of (Cluster) Tendency”.

[9] Liang Wang, Christopher Leckie, Kotagiri Ramamohanarao, and James Bezdek, Fellow, IEEE-MARCH 2009, Automatically Determining the Number of Clusters in Unlabeled Data Sets.

[10] Jang, J.-S. R., Sun, C.-T., Mizutani, E., “Neuro- Fuzzy and Soft Computing – A Computational Approach to Learning and Machine Intelligence,”Prentice Hall.

[11] Azuaje, F., Dubitzky, W., Black, N., Adamson, K., “Discovering Relevance Knowledge in Data: A Growing Cell Structures Approach,” IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics, Vol. 30, No. 3, June 2000 (pp.448).

[12] Lin, C., Lee, C., “Neural Fuzzy Systems,” Prentice Hall, NJ, 1996.[4] Tsoukalas, L., Uhrig, R., “Fuzzy and Neural Approaches in Engineering,” John Wiley & Sons, Inc., NY, 1997.

[13] Nauck, D., Kruse, R., Klawonn, F., “Foundations of Neuro-Fuzzy Systems,” John Wiley & Sons Ltd.,NY, 1997.

[14] J. A. Hartigan and M. A. Wong, “A k-means clustering algorithm,” Applied Statistics, 28:100--108, 1979.

[15] The MathWorks, Inc., “Fuzzy Logic Toolbox – For Use With MATLAB,” The Math Works, Inc., 1999.

[16] Srinivasulu Asadi, Dr. Ch.D.V.Subbarao, V. Saikrishna, “Finding the number of clusters using Dark Block Extraction”, IJCA International Journal of Computer Applications (0975 – 8887), Volume 7– No.3, September, 2010.

[17] A. Ahmad and L. Dey, (2007), A k-mean clustering algorithm for mixed numeric and categorical data’, Data and Knowledge Engineering Elsevier Publication, vol. 63, pp 503-527.

[18] Srinivasulu Asadi, Dr.Ch.D.V.SubbaRao, V.Saikrishna and Bhudevi Aasadi “Clustering the Labeled and Unlabeled Datasets using New MST based Divide and Conquer Technique,” International Journal of Computer Science & Engineering Technology (IJCSET), (0975 – 8887), IJCSET | July 2011 | Vol 1, Issue 6,302-306, ISSN:2231-0711, July, 2011.

[19] Xiaochun Wang, Xiali Wang and D. Mitchell Wilkes, IEEE Members, “A Divide-and-Conquer Approach for Minimum Spanning Tree-Based Clustering”, IEEE Knowledge and Data Engineering Transactions, vol 21, July 2009.

[20] Srinivasulu Asadi, Dr.Ch.D.V.Subba Rao, O.Obulesu and P.Sunil Kumar Reddy, “Finding the Number of Clusters in Unlabelled Datasets Using Extended Cluster Count Extraction (ECCE)”, ,” IJCSIT International Journal of Computer Science and Information Technology (ISSN: 0975 – 9646), Vol. 2 (4) , 2011, 1820-1824, August, 2011.

[21] S Deng, Z He, X Xu, 2005. Clustering mixed numeric and categorical data: A cluster ensemble approach. Arxiv preprint cs/0509011.

[22] Srinivasulu Asadi, Dr. Ch.D.V.Subba Rao, O.Obulesu and P.Sunil Kumar Reddy, “A Comparative study of Clustering in Unlabelled Datasets Using Extended Dark Block Extraction and Extended Cluster Count Extraction Extended Dark Block Extraction and Extended Cluster Count Extraction”, IJCSIT International Journal of Computer Science and Information Technology (ISSN:0975 – 9646), Vol. 2(4) , 2011, 1825-1831,August, 2011.

[23] S. Guha, R. Rastogi, and K. Shim, 2000. ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems, vol. 25, no. 5: 345-366.