Scalable Clustering: A Distributed Approach

Prodip Hore
Department of Computer Science and Engineering, ENB 118 University of South Florida, Tampa FL 33620
E-mail: phore@csee.usf.edu

Lawrence O. Hall
Department of Computer Science and Engineering, ENB 118 University of South Florida, Tampa FL 33620
E-mail: hall@csee.usf.edu

***Abstract*---The ever-increasing size of data sets and poor scalability of clustering algorithms has drawn attention to distributed clustering for partitioning large data sets. In this paper we propose an algorithm to cluster large-scale data sets without clustering all the data at a time. Data is randomly divided into almost equal size disjoint subsets. We then cluster each subset using the hard-k means or fuzzy k-means algorithm. The centroids of subsets form an ensemble. A centroid correspondence algorithm transitively solves the correspondence problem among the ensemble of centroids. The centroids are combined to form a global set of centroids. Experimental results show that most of the time the pattern of clusters generated by our algorithm is similar to the pattern of clusters generated by clustering all the data at a time. We have shown that the disputed examples between the clusters generated by our algorithm and clustering all the data at a time lay on the spatial border of clusters.**

I. INTRODUCTION

Clustering has long been used to group similar types of unlabeled data. Similarity among data can be measured by, for example, the Euclidean distance. Clustering may optimize an objective function iteratively to generate clusters within a data set. Hard k-means results in a crisp partition of the data while fuzzy k-means assigns a degree of membership to each example in a data set. Clustering doesn't scale well with very large data sets and there may be situations, where data may be geographically distributed [1]. We define clustering of all the data at time in a single memory as *global clustering* of the *global data*.

We propose an algorithm to cluster large data sets without clustering all the data at a time in a single computer memory. In [1], an ensemble of partitions has been combined without accessing the features of examples (Knowledge reuse framework [3]). Although, our algorithm doesn't provide a complete Knowledge reuse framework, it can form the global set of centroids without accessing any feature vector, provided knowledge of centroids exist in each subset.

We randomly divide the data (global) into equal size disjoint subsets. Each subset is formed by randomly selecting the data from the global data set. The smallest size of the subset has yet to be found. Similar questions have been raised in [4]. As each example from the global data set is equally likely to be selected, probability theory tells us that each subset is likely to have examples from each region of the global data. Thus centroids obtained by clustering data in each subset will contain partial information about the global partition, forming an ensemble of centroids. If one or more of the subset don't have enough representative examples from all the region of the global data, our algorithm may then combine non-representative centroids forming a global partition, which otherwise would not exist (fail to form a partition similar to global partition). We are currently developing centroid-filtering algorithms to avoid situation like this. In future we also plan to test our algorithm on high dimensional data. As discussed in [1], [2], the absence of labels for data in unsupervised learning makes the problem of combining an ensemble of partitions harder than combining an ensemble of classifiers in supervised learning. Attempts have been made to combine multiple partitions of a data set [1], [2], [4], [5]. In [1] Strehl, and Ghosh proposed an objective function to combine an ensemble of partitions but it is computationally prohibitive. They later used heuristic algorithms to combine an ensemble of partitions. In [2], Topchy etc. formed weak clusters of a data by random hyper planes and multiple views of a sample of the data and then combined them. Davidson and Satyanarayana [4] took bootstrap samples (with replacement) from the global data and clustered each sample to form a representative model. Then they combine all bootstrap models using a signature based cluster grouping. Our approach also combines multiple models (subsets), where the partition in each subset is represented by centroids. The key difference in our approach is the heuristic algorithm to combine the ensemble of centroids. Our subsets are neither bootstrap samples nor formed by a partial view of the data. They are disjoint subsets formed by randomly selecting the examples from the global data set, and thus provide a framework for distributed clustering of very large data sets. Each subset will then contain less data, and thus can be clustered faster [4]. The subsets can also be clustered in parallel. Our algorithm provides the platform to cluster a large dataset distributively and parallely. One may enhance clustering speed of individual subsets by using any standard speed-up techniques [6], [7], [8].

Generally large real data sets are likely to have lots of extremas. We have *observed* that if the J (objective function)

value of these extremas differs among themselves significantly in the global data, then the subsets also inherit this property. Thus there is always a probability that we might attempt to combine partitions (represented by centroids), which are significantly different and produce global partition, which couldn't otherwise occur during global clustering (fail to form a partition similar to global partition). This type of diversity among ensembles of centroids is a curse rather than a boon in unsupervised learning. Even if the multiple partitions (represented by label vectors) are combined using the objective function proposed by Strehl and Ghosh [1], this problem can occur. The partition formed by our Distributed-combining algorithm is compared to the partition formed by global clustering, and we call the examples that are placed in different clusters in the above two partitions disputed *examples*. We have plotted the disputed examples between the clusters formed by our algorithm and the global partition. The plot shows that these disputed examples lie on the border of global clusters.

II. CENTROID CORRESPONDENCE

After the subsets are formed, we cluster the data in each subset using a hard k-means or fuzzy k-means algorithm. Each partition on a subset generates a set of centroids. We assume each subset has the same number of clusters. To combine the partitions of all subsets (represented by centroids) we need to find out approximately which cluster in which subset corresponds to which cluster in other subsets. We propose a heuristic algorithm to solve the centroid correspondence problem. We define three essential data structures, which will be used in the algorithm. The algorithm assumes that the correspondence relation among centroids of subsets is approximately transitive in nature.

A. Matrices used for centroid correspondence

The Distance matrix stores the Euclidean distance among the centroid vectors of any two selected subsets. If there are C clusters in each subset, the dimension of the Distance matrix will be C by C.

The Local chain matrix stores the correspondence relation between any two selected subsets i.e. which cluster in the first selected subset corresponds to which cluster in the second subset. If there are C clusters in each subset, the dimension of the Local chain matrix will be C by 2.

The Global chain matrix uses the information from the local chain matrices to build the global correspondence relation of centroids in all subsets i.e. which cluster in a subset corresponds to which other clusters in other subsets. If there are C clusters in each subset and there are M such subsets, the dimension of the Global chain matrix will be C by M.

The centroid correspondence algorithm first picks up the first two subsets and computes the Euclidean distance among the centroid vectors in the distance matrix. The rows of the distance matrix represent the clusters of the first subset and the columns represent the clusters of the second subset. The smallest distance in the matrix is found and the correspondence relation (i.e. row and column number of the distance matrix) is stored in the local chain matrix. We then find the next minimum distance in the distance matrix. If the row and column number of this next minimum has not been stored earlier in the Local chain matrix, this new correspondence relation is stored in the local chain matrix. If either the row or column has been used earlier, we say that a *collision* has occurred. If a collision occurs, we skip that clusters centroid and find the distance to the next closest centroid and again test for collision. The correspondence relation is stored in the local chain matrix if and only if it is with *no collision.*

B. Illustrative Example

Consider the case that there are 3 subsets of data and each subset is grouped into 3 clusters. Let S1, S2, and S3 be the subsets. Fig. 1(a) shows the two local chain matrices and Fig. 1(b) shows the global chain matrix. Fig. 2 shows the centroid correspondence algorithm.

S1	S2
2	2
1	1
3	3

S2	S3
3	1
1	3
2	2

(a) The arrow shows the transitive relation between local chain matrixes.

S1	S2	S3
2	2	2
1	1	3
3	3	1

(b) The Global chain matrix formed from above two local chain matrixes**.**

Fig. 1. Chain matrices

We call each row of the Global chain matrix a *centroid chain*. It tells us which centroid in which subset corresponds to which other centroid(s) in other subsets. If the global chain matrix is formed without any collision at any stage, we say that the centroids of subsets have *mapped perfectly*. The whole process is known as *centroid mapping*. The input to the algorithm (Fig. 2) is only the set of centroids of subsets. The number of centroids of a data set is not generally a function of the number of examples, so the above algorithm generally takes constant time. After solving the correspondence problem, a centroid chain contains *similar types* of centroids. We find the weighted (according to number of examples in each subset) arithmetic mean of centroids in each centroid chain to represent a global centroid of the global partition

III. DISTRIBUTED COMBINING ALGORITHM

Now, we present the complete algorithm (Fig. 3) for clustering large-scale data without clustering all the data at once.

***Input**: centroids of subset partitions*
***Output**: Global chain matrix*
1. M=number of subsets, C=clusters in each subset, I=1, J=2. Global chain matrix is initialized to zero.
2. While (J! = M)
2.1 Local chain matrix initialized to zero. Select subset I and subset J and compute the Euclidean distance among the centroid vectors and store them in the distance matrix such that position (row, col) of the distance matrix is the Euclidean distance between row centroid vector of subset I and col centroid vector of subset J.
2.2 Find the minimum value in the distance matrix and record it's position (row, col) value in the local chain matrix
2.3 While (local chain matrix is not completely filled up)
2.3.1 Find the next minimum value in the distance matrix. If a collision doesn't occur, record the position (row, col) in the local chain matrix. Otherwise find the smallest distance for which no collision occurs and record that position
end while.
2.4 If (I==1)
The correspondence relation in the local chain matrix forms the first two columns of the global chain matrix.
Else
Fill the J^{th} column of the Global chain matrix using the local chain matrix and the transitive relation it has with the I^{th} column in the Global chain matrix..
2.5 I=I+1,J=J+1.
end while.

Fig. 2. Centroid correspondence Algorithm

***Input**: Global data*
***Output**: Partition of the global data*
1. Divide the data into M number of subsets.
2. Cluster each subset by a standard fuzzy k means or hard k-means algorithm.
3. Call the centroid correspondence algorithm to form the global chain matrix.
4. For each centroid chain, compute the weighted (according to the subset size) arithmetic mean of centroids in that centroid chain. The Arithmetic mean of centroids in each centroid chain represents the centroid of a cluster of the global partition.
5. Compute the Euclidean distance between each example and the global set of centroids. Assign the example to the nearest cluster centroid.

Fig. 3. Distributed (D)-Combining algorithm

Since, we are clustering a part of the data in subsets, a speedup is expected [4]. Each subset can be partitioned on a different computer simultaneously. As mentioned earlier, step 3 generally takes constant time. Step 5 will take O (n) time, where n=number of examples in the global data.

IV. CENTROID INITIALIZATION

Iterative clustering algorithms are sensitive to initialization. There are two ways one might approach initialization in a distributed clustering problem. Each subset of data might be given its own random initialization (Pure-random). Alternatively, all subset of data could be given the same random initialization (Semi-random). This is a low-cost approach to providing a potentially more uniform set of partitions. We will report experimental results with both initialization approaches.

V. EXPERIMENTAL RESULTS

We have performed experiments on the Iris Plant (from UCI repository of machine learning data base) data set and on an MRI data set.

A. Iris Plant Database

The Iris plant data set [9] consists of 150 examples each with 4 numeric attributes. It consists of 3 classes of 50 examples each. One class is linearly separable from the other two; the others have some overlap.

B. MRI Database

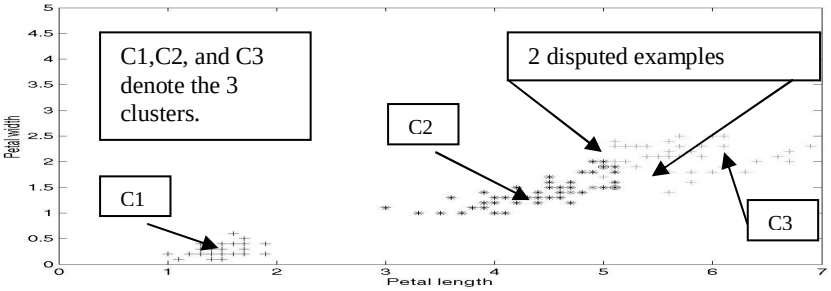
The MRI data set consists of 22,320 examples, each consisting of 3 numeric attributes. The attributes are T1 weighted, T2 weighted, and PD (proton density) weighted.

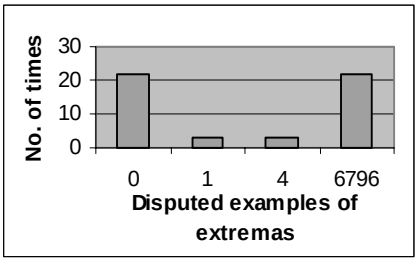
C. Experiments

We cluster the global data 50 times in a single memory with random centroid initialization within the range of the data. We keep a count of the extrema, which occurs most often. The average disputed examples and their standard deviation are computed. As the most typical extrema is chosen as the reference frame, the average disagreement will be zero a maximum number of times. We kept two

TABLE 1
Results of our D-combining algorithm. Fuzzy k means applied to each subset

(K=3 in each subset) Iris data	Average disputed examples (Semi-random)	Standard Deviation of disputed examples (Semi-random)	Average disputed examples (Pure-random)	Standard Deviation of disputed examples (Pure-random)
2-Subsets (Fuzzy)	0	0	0	0
3-Subsets (Fuzzy)	2	0	2	0



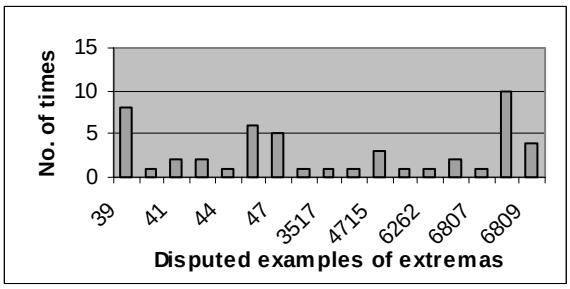


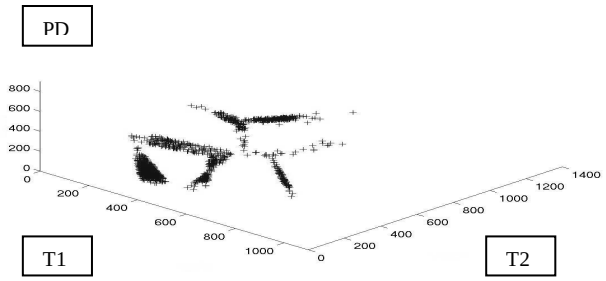
(a) (b)

Fig. 4. (a) 2 disputed examples (encircled) for 3 subsets-hard-k means in each subset**.** (b) Disputed examples of extremas of MRI data (global fuzzy clustering).

TABLE 2
Results of our D-combining algorithm (Fuzzy clustering applied to each subset). Also clustering results with all 22,320 total examples

(7 clusters) MRI Data set	Average disputed examples (Semi-random)	Standard Deviation of disputed examples (Semi-random)	Average disputed examples (Pure-random)	Standard Deviation of disputed examples (Pure-random)
1-Subset (Fuzzy)	2990.54	3407.42		
2-Subsets (Fuzzy)	3040.68	3229.14	3288.82	2665.02
3-Subsets (Fuzzy)	2980.76	3268.57	3380.84	2381.51
4-Subsets (Fuzzy)	3175.38	3213.51	3848.04	1815.08
5-Subsets (Fuzzy)	3073.8	3160.03	3469.36	1927.57





(a) (b)

Fig. 5. (a) The disputed examples of extremas obtained using 5 subsets of MRI data (fuzzy, k=7 in each subset, semi-random initialization). (b) Plotting of 1384 disputed examples of MRI data using 5 subsets-Hard-k-means in each subset.

types records: one in which we count these zero values only once and ignore other zeros ("Zero counted once"), and in the other we count everything ("All counted"). Result comparisons are done taking into account everything ("All counted").

The global data is clustered using our Distributed-combining algorithm using 50 pure random and 50 semi-random centroid initializations. We computed the average disputed examples and their standard deviation found by our Distributed-combining algorithm with the most typical extrema found during global clustering. Thus we compare the results of our D-combining algorithm with the most typical global partition. Since there can be more than one extrema in a data set, we have chosen the clusters of the most typical extrema as the reference frame, and plotted the disputed examples of similar types of extremas produced by our D-combining algorithm. Disputed examples of other types of extremas produced by our D-combining algorithm can also be plotted by choosing the closest extrema (by J value) from the global clustering and making that the reference frame. For the Iris data we have used 2 attributes (petal length and petal width) to plot the disputed examples because these features contain most of the information about the data. For MRI data we have used all the 3 attributes.

We performed experiments with the Iris data by dividing it randomly into 2 and 3 subsets and clustering each subset using the fuzzy-k means and the hard k-means algorithm. The number of clusters in each subset has been assigned to be 3. For all initializations, the exact same set of clusters was obtained using Fuzzy clustering using all the data. Table 1 shows the results of our D-Combining algorithm. Using 2 subsets and fuzzy-k means on each subset, it gives the same good clusters as global clustering and with 3 subsets we got only 2 disputed examples on average. As the average standard deviation of disputed examples is 0, like global clustering, clustering of 3 subsets by the D-Combining algorithm also produces 1 type of extrema. We plotted these 2 disputed examples in Fig. 4(a) and found that these disputed examples (encircled ones) lie on the border of global clusters.

Similar results have been obtained using hard-k-means in each subset, where clustering subsets by pure-random centroid initialization produces more extremas than semi-random initialization (keeping the number of subsets the same). It can be expected because the probability of combining extremas from different subsets, whose centroids differ significantly, is more during pure-random initialization than in semi-random initialization. This is because every subset has been initialized with centroids independent of the other, thus the probability of convergence of the objective function to a J value, which differs significantly is high. It has been observed that under this condition the centroids of subsets have more collisions in the centroid correspondence algorithm.

We have performed experiments on the 22,320 example MRI data. MRI data has clusters of varying size and density. One could also utilize other clustering algorithms on each subset, which better deal with differing size and density clusters. We performed tests using 2, 3, 4, and 5 subsets. We conducted experiments by initializing the value of k to 7 in each subset. The results of combining MRI data using 2, 3, 4, and 5 subsets and clustering using fuzzy k-means are shown in Table 2. The first line of Table 2 is the average over 50 random initializations using all the data. The global clustering results are reported as "All counted". The results show that the average number of disputed examples and their standard deviation obtained by our D-Combining algorithm using semi-random initialization is quite consistent with the average number of disputed examples and standard deviation obtained during global clustering of the MRI data (using k=7). This means that the pattern of extremas generated by our

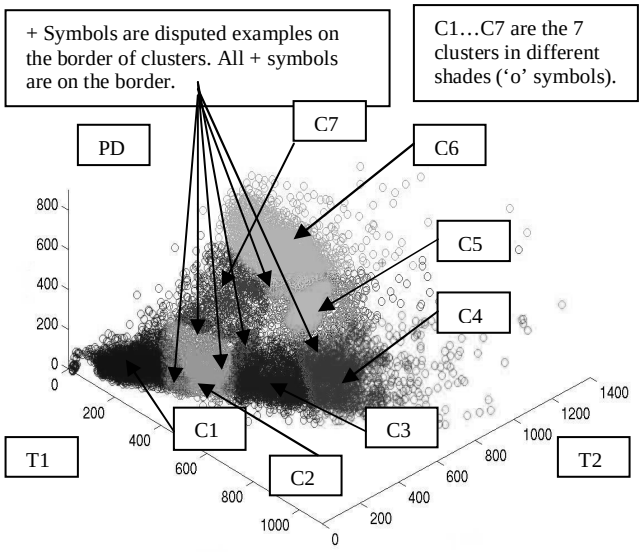


Fig. 6. The disputed examples of Fig 5 b (1384 examples) lie on the border (+ symbols) of clusters. Clusters have been represented using different shades of 'o' symbol.

D-Combining algorithm using semi-random initialization is quite similar to the pattern of extremas generated by clustering all the data at a time. Fig 4(b) shows the histogram of the disputed examples of all extremas that occurred during global clustering of MRI data (using fuzzy k means, k=7). It has lots of extremas whose disputed examples are 0 (most typical) or close to zero. There is also an equal amount of extremas whose disputed examples are large (6796 at the other end in Fig. 4(b)). The extremas generated by our algorithm using 5-subsets (fuzzy k means in each) generates a similar pattern of extremas (Fig. 5(a)). We also found some other types of extremas in the middle (Fig. 5(a)) i.e. (no such similar extrema exist in Fig. 4 b). This is expected because the J values of lots of extremas in the global data are quite different (Fig. 4(b)). Thus, sometimes our algorithm combines a highly diverse

ensemble of centroids producing partitions, which do not exist in global partition. With pure-random initialization this occurs more often. We got similar results using hard clustering also. We plotted the disputed examples found by our D-combining algorithm in Fig. 5(b)(using 5 subsets-hard k means on each subset). Fig. 6 shows that the disputed examples of Fig. 5(b) lie on the border of clusters. Similar plots have been obtained by clustering subsets with fuzzy-k means. Assigning k=10 also gave more or less similar results.

VI. DISCUSSION

Experimental results show that our D-Combining algorithm produces partitions quite similar to those generated by global clustering. We have observed that the D-combining algorithm produces extremas (or clusters) more similar to the global extremas, if the centroids in different subsets map without any collision. If the global data set has lots of extremas and if their J1 (J2) value differs significantly, then it is more probable that the centroids of different subsets will collide during centroid mapping. This likelihood becomes higher during pure-random initialization of centroids in each subset. It has been observed that our D-Combining algorithm produces more distinct extremas using pure-random centroid initialization than using semi-random centroid initialization.

The Iris data set when globally clustered using fuzzy k-means had only 1 extrema in our experiments similarly, our D-Combining algorithm generates 1 extrema. The MRI data when clustered globally using the fuzzy k-means algorithm produces extremas whose J2 (J1) value differ a lot i.e. they disagreed among themselves significantly (Fig. 4 (b)). Similarly, our Distributed-combining algorithm also produces similar patterns of extrema, which differ significantly (Fig. 5(a)). The disputed examples between the clusters generated by our Distributed-Combining algorithm and the global partition lie, as might be expected, on the border of clusters (Fig. 4(a), Fig 5(b), Fig. 6).

Sometimes, our algorithm produced extremas, which didn't exist in the global partition. This happens when the J value of extremas differs significantly in the global data and subsets also inherit that property. In future we plan to develop an algorithm to detect centroids, whose removal will make the final partition better. We also plan to test with larger high-dimensional data sets.

VII. CONCLUSIONS

Clustering very large-scale data using hard-k means or fuzzy k-means is very time consuming. Sometimes, the data may be geographically distributed or may be too large to fit in a single memory. In this paper we proposed a Distributed-combining algorithm that first divides the data randomly into almost equal size disjoint subsets and then clusters each subset using a hard or fuzzy-k means algorithm. The centroids of subsets form an ensemble. After solving the correspondence problem among the ensemble of centroids, we combine them. The number of clusters in each subset has been kept the same.

Our results on two real data sets show that most of the time the pattern of extremas (or clusters) produced by our D-combining algorithm is similar to the pattern of extremas generated by global clustering. The clusters produced by our D-combining algorithm and the clusters produced by global clustering of the data, have disputed examples on the border of the global clusters. If the initial centroid assignments of subsets are semi-random, our D-combining algorithm tends to produce an extrema pattern closer to the pattern of extremas found during global clustering. If pure random initialization is used, the similarity of extremas decreases a little. This is because the likelihood of a perfect mapping of centroids of disjoint subsets is higher during semi-random centroid initialization than in pure random centroid initialization. Data sets having lots of extremas, whose J value differs heavily among them, sometimes, produce clusters dissimilar to the global cluster.

ACKNOWLEDGEMENT

This research partially supported by The National Institutes of Health via a bioengineering research partnership under grant number 1 R01 EB00822−01.

REFERENCES

[1] A. Strehl and J. Ghosh, "Clusters ensembles- a knowledge reuse framework for combining multiple partitions". Journal of Machine learning Research, 3, 2002, pp. 583-617

[2] A. Topchy, A.K Jain, and W. Punch, "Combining Multiple Weak Clusterings". Proceedings of IEEE Intl. Conf. on Data Mining, 2003, pp. 331-338.

[3] K. D Bollacker and Joydeep Ghosh, "A Supra-Classifier Architecture for Scaleable Knowledge Reuse". In *Proc. Int'l Conf. on MachineLearning (ICML-98)*, 64–72.bor

[4] I. Davidson and Ashwin Satyanarayana, "Speeding up K-means clustering by Bootstrap averaging".. To Appear in the Workshop on Clustering Large Data Sets, IEEE ICDM 2004.

[5] A.L.N Fred, "Finding Consistent Clusters in Data Partitions". In Proc. 3d Int. Workshop on multiple Classifier Systems. Eds. F. Roli, J. Kittler, LNCS 2364, 2002, pp. 309-318.

[6] F. Farnstrom, J. Lewis, and Charles Elkan, Scalability for Clustering Algorithms Revisited, SIGKDD Explorations, 2, 1, 2000, pp. 51-57.

[7] F. Hoppner, Speeding up fuzzy c-means: using a hierarchical data organization to control the precision of membership calculation, Fuzzy Sets and Systems, 128, 2002, pp. 365-376.

[8] S. Eschrich, J. Ke, L.O. Hall and D.B. Goldgof, Fast Accurate Fuzzy Clustering through Data Reduction, IEEE Transactions on Fuzzy Systems, 11, 2, 2003, pp. 262-270.

[9] C.J. Merz and P.M. Murphy. UCI Repository of Machine Learning Databases Univ. of CA., Dept. of CIS, Irvine, CA http://www.ics.uci.edu/~mlearn/MLRepository.html.