

A Cluster Ensemble Framework for Large Data sets

Prodip Hore, Lawrence Hall, *Fellow, IEEE*, and Dmitry Goldgof, *Member, IEEE*

Abstract—Combining multiple clustering solutions is important for obtaining a robust clustering solution, merging distributed clustering solutions, and scaling to large data sets. Combining multiple clustering solutions within a scalable and robust framework for large data sets is addressed. Merging multiple clustering solutions in a scalable framework requires both ensemble creation and merging to be efficient in terms of time and memory complexity. We also introduce the concept of filtering malformed clusters from the ensemble resulting from unfortunate initialization or unbalanced data distribution or noise. Experimental results on real data sets show that this approach will scale and provide cluster partitions which are functionally better or equivalent when compared to clustering all the data at once and clustering solutions in the ensemble. We have also compared our algorithm with other relevant ensemble merging and scalable algorithms to point out its strength and limitations.

I. INTRODUCTION

Clustering algorithms are used to partition data, where the examples are not labeled. Clustering data is often considered to be a slow process. This is especially true of iterative clustering algorithms such as the K-means family [16] or EM [23]. As larger unlabeled data sets become available, the scalability of clustering algorithms becomes important. It has become increasingly difficult to load large data sets into a single memory for clustering [11]. Data can be sequentially clustered in such a way that each data subset fits in memory and finally the clustering solution of all subsets can be merged. Sometimes, data is physically distributed and centrally pooling the data might not be feasible due to privacy issues and cost. Thus, merging of clustering solutions from different distributed sites is required. Moreover, iterative clustering algorithms are sensitive to initialization and produce different partitions for the same data with different initializations. Combining multiple partitions will provide a robust and stable solution [1, 7]. Also, combining multiple partitions might produce novel clustering solutions which might not be possible when clustering all the data [1, 7]. Thus, the importance of combining multiple clustering solutions has emerged as an important area of research to address the issue of scalability, the distributed nature of some data, robustness, novelty and the stability of clustering solution.

There has been research on combining ensembles of clustering solutions [1, 2, 5, 6, 7, 8, 12, 14] where each clustering solution is represented by a label vector whose size is equal

to the size of the data from which the solution was obtained. It has been shown that combining this ensemble of clustering solutions produces a good final global partition. But creating and combining multiple clustering solutions represented by label vectors could pose a scalability problem in terms of memory and time complexity for large data sets. In [7], it has been pointed out that existing merging algorithms suffer from a time complexity problem. In most of the approaches using label vectors the representation of each clustering solution in the ensemble is $O(n)$, where n is the number of examples. Also, label vector representation of clustering solutions in the ensemble might pose a natural problem to merging clustering solutions for object distributed data. In [1], it has been pointed out that combining clustering solutions from object distributed data is more difficult than feature distributed as there is no overlapping between labels from disjoint data sets. This could be an obstacle to scalability as clustering the data distributively is a way of scaling it. We believe for large data sets the representation of a cluster ensemble should be small so that it can be kept in memory and be merged efficiently while at the same time providing a robust and good quality final clustering solution. The merging algorithm should also be able to merge clustering solutions obtained from a set of distributed examples, so as to provide a way of scaling clustering algorithms on very large data sets.

In this paper, we propose a framework for creating and merging an ensemble of multiple clustering solutions, which is scalable in terms of time and memory complexity. In our approach multiple clustering solutions (an ensemble) are created from disjoint data sets, where each clustering solution in the ensemble is represented by its centroids. Creating the ensemble of centroids from disjoint data set also puts a restriction to the size of the ensemble i.e. if we divide a data set of 1000 examples into 10 equal subsets then there can be a maximum number of 10 sets of centroids from each subset in the ensemble. In comparison to label vectors the centroids are generally smaller in size as the number of clusters is generally much smaller than number of examples in the data. Thus ensemble formed by centroids can be kept in memory and merged efficiently. For a data set of very large dimensions keeping centroids in memory might be difficult also. However for many cases the size of centroid vectors is not likely to be large.

Now, the problem of finding a final clustering solution using the limited knowledge of the ensemble of centroids can be viewed as the problem of reaching a global consensus on the position of cluster centroids for the final clustering solution. A global consensus on the position of cluster centroids of the global data using only the centroids from each clustering solution is formed by grouping them

Prodip Hore, Lawrence Hall, and Dmitry Goldgof is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, USA (Prodip Hore phone: 813-472-3634; email:phore@csee.usf.edu), (Lawrence Hall phone: 813-974-4195; email:hall@csee.usf.edu), (Dmitry Goldgof phone: 813-974-4055; email:goldgof@csee.usf.edu)

into consensus chains and computing the weighted mean of centroids in each consensus chain to represent a global cluster centroid. A final partition in terms of label vectors can be obtained by computing the Euclidean distance of each example from the final set of centroids, and assigning it to the nearest centroid. Thus, the set of global centroids can be used to partition the data. Experimental results show that the quality of clusters generated by our algorithm is better than or similar to the average quality of partitions in the ensemble and the cluster quality generated by clustering all the data at a time. As in [1] for producing the final clustering solution we don't need access to the feature values of the examples, thus providing a knowledge reuse framework also. Our approach can be used also for scaling clustering algorithms because disjoint subsets of data can be clustered in parallel. Even in the case that only one machine is available; data can still be sequentially clustered in such a way that each data subset fits in memory and the ensemble of clustering solutions can be merged at the end to give a final partition. The main focus of this paper is to show how an ensemble of centroids created from object distributed data can be merged in a scalable framework, in terms of time and memory complexity, to produce a robust and good final clustering solution when compared to clustering all the data at a time. We have compared the quality of our algorithm with five other state of the art multiple partition combining algorithms and a scalable single pass clustering algorithm.

II. RELATED WORK

Creating multiple partitions of a given data set has been examined in several ways [1, 2, 5, 6, 7, 8, 12, 14]. In [1] hyperedges are formed from each clustering solution represented as label vector. Using these hyperedges a final partition is obtained by using graph partitioning algorithms like Cluster-based Similarity Partitioning (CSPA), Hypergraph-Partitioning (HGPA), and Meta-Clustering (MCLA). In [2] clustering solutions in the ensemble are represented by membership matrices of size n by k , where n is the number of examples in the data set and k is the number of clusters. A final clustering solution is then obtained by soft correspondence, where each cluster from one partition corresponds to clusters in other partitions with a different weight. In [12], weak clusters of a data set are formed by using random hyper planes and multiple views of a sample of the data. By creating many views of the data and merging them, they were able to show the final partitions were better. However, the approach might be complicated for large data sets because as many as 100 or more different partitions might be created to get a good overall partition. In [14] clusters have been merged using co-association matrices, but they are computationally expensive. In [6, 8, 22] combining clustering solutions from multiple bootstrap samples has been discussed.

As in our framework, disjoint subsets of data can be clustered in parallel and thus can be used independently to allow clustering to scale for large data sets, so we did a survey of scaling algorithms also. In recent years a number

of new clustering algorithms have been introduced to address the issue of scalability [13, 15, 16, 17, 18, 19, 20, 21]. All the referenced algorithms assume that the clustering algorithm is applied to all the data centrally pooled in a single location. Various methods for expediting fuzzy-k-means have been explored. In [31] data sub sampling is proposed and a good speed up of fuzzy-k-means is obtained. Similarly, in [32] a multistage random sampling algorithm was proposed to speed-up fuzzy-k-means. A parallel algorithm of k-means [27] or other similar algorithms is not quite applicable to cluster ensembles because they do not combine multiple clusterings but provide a way of clustering a data set using multiple processors, where synchronized communication during each iteration is required.

In [25, 26] distributed clustering has been discussed under limited knowledge sharing. In [26] a clustering solution has been represented by labels while in [25] generative or probabilistic model parameters are sent to a central site, where "virtual samples" are generated and clustered using an EM algorithm [23] to obtain the global model. In [24], local sites are first clustered using the DBSCAN algorithm and then representatives from each local site are sent to a central site, where DBSCAN is applied again to find a global model. Another density estimation based distributed clustering algorithm has been discussed in [25]. In [4] local models or prototypes are detected using EM [23] in a distributed environment and later merged by computing mutual support among the Gaussian models. So, there has been some work where a global centroid was formed by merging local models or prototypes using EM or density based algorithms in a distributed environment including our earlier work [11]. Our merging algorithm is different from them, as we optimally solve the correspondence problem between two partitions at a time by assuming one to one correspondence and build consensus chains. The formation of a consensus chain provides the framework for filtering malformed clusters in an ensemble and prevents "blind" merging of a clustering solution, which to our knowledge is novel concept in the cluster ensemble problem. This merging algorithm is mainly designed keeping in mind the ensemble formed from hard-k-means and fuzzy k means clustering and provides a way of scaling them to large data also.

III. ENSEMBLE MERGING

After clustering is applied to each disjoint subset there is a set of centroids available which describe each partitioned subset. Clustering or partitioning a subset $\{S_i\}$ will produce a set of centroids $\{C_{i,j}\}_{j=1}^k$, where k is the number of clusters. In this work, we set number of clusters for each partition of the ensemble to be the same as that of the final target clustering. For r subsets there will be r sets of centroids i.e. $\{C_{1,j}\}_{j=1}^k, \{C_{2,j}\}_{j=1}^k, \dots, \{C_{r,j}\}_{j=1}^k$ forming an ensemble of centroids. To produce the final partition, we need to reach a consensus of the position of the centroids in the target clustering. One way to reach a global consensus is to group the ensemble of centroids into k consensus chains, where each consensus chain will contain r centroids

$\{c_1^n, \dots, c_r^n\}$ one from each of the subsets, where n runs from 1 to k . The aim is to group similar centroids in each consensus chain by solving the cluster correspondence problem. The objective is to globally optimize the assignment ψ^* out of all possible families f of centroid assignments to k consensus chains:

$$\psi^* = \arg \min_{\psi \subset f} \{\psi\} \quad (1)$$

$$\psi = \sum_{n=1}^k \text{cost}(\text{consensus_chain}(n)) \quad (2)$$

$$\text{cost}(\text{consensus_chain}(n)) = \sum_{i=1}^r D^n(i) \quad (3)$$

$$D^n(i) = \frac{1}{2} \sum_{j=1}^r d(c_i^n, c_j^n)_{j=i} \quad (4)$$

where $d(\cdot, \cdot)$ is the distance function between centroid vectors in a consensus chain. We have used the Euclidean distance in computing the cost (4). In other words, in a graph theoretical formulation if the centroids of each partition are represented by non-adjacent vertices of a graph and the Euclidean distance between a centroid of a partition and other partitions as a weighted edge, then finding the globally optimum value for the objective function (1) reduces to the minimally weighted perfect r -partite matching problem, which is intractable for $r > 2$. So, we need another approach.

We know that for 2 partitions i.e. $r=2$, there is a polynomial time algorithm i.e. minimally weighted perfect bipartite matching to globally optimize the above objective function. For optimally matching centroids of 2 partitions we have used the Hungarian method of minimally weighted perfect bipartite matching [10]. So, we match two partitions at a time. After matching a pair of partitions we keep the centroids of one of its partitions as the reference and a new partition is randomly chosen and matched i.e. minimally weighted bipartite matching. Now, the centroids of this new matched partition are in the same consensus chain in which the centroids of the reference partition belong. In this way we continue grouping the centroids of new partitions into consensus chains one by one until they are exhausted. This is a modification of our earlier work [11], where we matched centroids of two partitions using a greedy algorithm. After the consensus chains are created, we simply compute the weighted arithmetic mean of centroids in a consensus chain to represent a global centroid, where the weights of a centroid are determined from the size of the subset from which it was created. Each consensus chain tells us which centroid in which subset is matched to which other centroids in other subsets. A final partition can be obtained by assigning an example to the nearest global centroid.

A. Example

Consider the case that there are 4 subsets S1, S2, S3, and S4 of a data set as shown in Figure 1 and each subset is grouped into 3 clusters. Let S1 first match with S2, then S2

with S3, and S3 with S4. After solving the matching problem, a consensus chain contains *similar types* of centroids as shown in Figure 2.

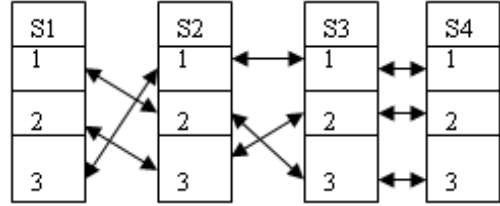


Fig. 1. Partitions matched

S1	S2	S3	S4
1	2	3	3
2	3	2	2
3	1	1	1

Fig. 2. Each row a consensus chain

As mentioned before, we are optimally matching the centroids of two randomly picked partitions at a time. But the question comes in which order the partitions should be matched so that the overall matching cost is minimum. We have found it to be equivalent to solving the traveling salesman problem. This is because we are optimally matching two partitions at a time and if we could also optimally select their order of selection then we are optimizing the whole objective function (1), which is an intractable problem. The implication of random pair-wise matching is that in a consensus chain between any three centroids there is a transitive relation i.e. if centroid C1 is matched to C2, and C2 is matched to C3, then C1 is matched to C3. We believe in the cluster correspondence problem this assumption of transitive relation is quite valid. One could also use an approximate traveling salesman algorithm to determine a sub-optimal minimum cost order. It would obviously add some computation time of the merging algorithm. We would evaluate later the sensitivity of the random selection of the order of partitions for merging.

IV. FILTERING BAD CENTROIDS

We introduce a novel concept of detecting and removing spurious or malformed clusters from the cluster ensemble, which may result from an unfortunate initialization or unbalanced data distribution or noise in the data set. As each consensus chain contains matched centroids, the problem is equivalent to removing outliers/noise from it, if present. But with a limited number of centroids in a consensus chain and no knowledge about the distribution of noise, detecting them is a non trivial task. It is reasonable to assume that “good” centroids would be spatially close to each other forming the largest compact cluster. Thus we have to search for this

cluster and the rest would be classified as outliers/noise. So, for each consensus chain we built a minimally weighted spanning tree from a complete graph whose vertices are the centroids in a consensus chain and the Euclidean distance between centroids as the weight of the edges. Next, we sort the edges of the tree and find the mode. Now, if all edges of intermediate or high lengths are removed this will reveal clusters [30] in the consensus chain. We have set the threshold to cut edges above $\text{mode} + 2.5 * \text{mode}$. After cutting the edges of the spanning tree it becomes a forest. Now, we take vertices of the largest tree/cluster from this forest for merging and the rest as outliers. This largest connected component/tree is found using a depth-first search algorithm. As stated earlier merging is then done by computing the weighted arithmetic mean of these centroids in a consensus chain, where the weights of a centroid are determined from the size of the subsets from where it has come. Thus, each consensus chain will result in a final centroid. The code for bipartite matching (Hungarian method), depth first search, and minimally weighted spanning is obtained from <http://reptar.uta.edu/>.

Example: Consider the example as mentioned in the previous section i.e. Figure 1 and Figure 2. Each consensus chain has 4 centroids. Figure 3 shows the example of building a spanning tree from such a consensus chain and Figure 4 shows the resulting forest after cutting edge(s) from such a minimally weighted spanning tree. The largest connected component C1, C2, and C3 is then selected for merging.

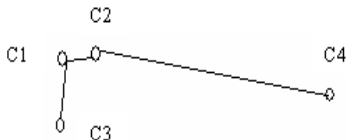


Fig. 3. Minimally weighted spanning tree formed from centroids in a consensus chain.

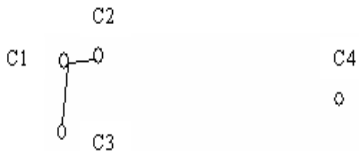


Fig. 4. After cutting edge(s) above threshold, the largest connected component (C1, C2, and C3) is used for merging.

V. EXPERIMENTAL SETUP

In [2] extensive comparison of Soft-Correspondence Ensemble Clustering (SCEC) has been done with four other state of the art algorithms on three real-world data sets. The algorithms which they compared with are CSPA [1], MCLA [1], QMI [7], and MMEC [12]. CSPA and MCLA are graph partitioning algorithms while QMI uses k-means,

and MMEC is based on mixture model based ensemble clustering. We conducted experiments on the same three real-world data sets used there [2]. This allows us to compare the quality of our algorithm with their results [2]. For comparing quality, we used the same information theoretic criterion they used- the Normalized Mutual Information (NMI) [1]. NMI computes the mutual information between two label vectors. Its values ranges from 0 to 1. An NMI value of 1 means two partitions are exactly the same. The label vector from a clustering solution represented by centroids can be computed by assigning the examples to the nearest centroid. For measuring quality in NMI, label vectors are compared with the true class labels of the data.

It is clear that our approach of creating the ensemble and merging is different. They used label vectors and we used centroids, but ultimate goal of all is to create a good, robust final partition. So, we believe comparing the quality of final partitions produced by our approach and the approach reported in [2] on the same data sets will provide an insight into the quality of our algorithm. We also compared the quality of our centroid based ensemble clustering algorithm (CBEC) with clustering all the data at once in a single memory i.e. global clustering (GC) and with the mean NMIs of the partitions in the ensemble (MNE) i.e. produced by applying k-means or the fuzzy k-means on the subsets. Clustering each subset will create a set of centroids. A label vector can then be obtained by assigning examples from global data set to the nearest centroid. This label vector is then compared with real class label of the global data to compute the NMI for that subset.

As mentioned earlier, even if a data set is centrally available, loading the whole data into memory and clustering it is difficult for large data sets. If we cannot load the entire data into memory, hard-k-means or fuzzy-k-means will have to make a disk access every iteration. To address this problem, in [15] a single pass clustering algorithm (SP) has been described for hard-k-means, where only part of the data is loaded into memory at a time and the whole data set is clustered in one disk access. We have also compared our algorithm with this approach. The code for [15] is available at <http://www-cse.ucsd.edu/users/elkan/skm.html>. For computing NMI of SP clustering solutions, a label vector is created in the same way by assigning examples from global data to the nearest centroid and then compared to the true class labels.

VI. DATA SETS USED

A. The Iris Data set

The Iris plant data set consists of 150 examples each with 4 numeric attributes [29] and 3 classes of 50 examples each. One class is linearly separable from the other two.

B. The ISOLET6 Data set

It is a subset of the ISOLET spoken letter recognition training set and has been prepared in the same way as done in [2]. Six classes out of 26 were randomly chosen and it consists of 1440 instances with 617 features [29].

C. The Pen-Based Recognition of Handwritten Digits

It consists of 3498 examples, 16 features, and 10 classes [29]. It is a pen-based handwritten digits with ten digits 0 to 9. We have named it as Pendig data set in our experiments.

VII. RESULTS AND DISCUSSION

In the experiments the number of partitions combined for the Iris data set was $r=3$ and 5 i.e. divided into 3 and 5 disjoint subsets. For ISOLET6 and Pendig it was 5 and 10. Each experiment is conducted 50 times with random initialization. Both hard-k-means and fuzzy-k-means were used as the clustering algorithm. The single pass clustering algorithm (SP) in [15] is for hard-k means only. So, its results are reported for hard-k-means experiments only. Table I and Table II show the results on the three data sets using hard-k-means and fuzzy-k-means respectively. In the Tables the quality metric is the NMI and the entry in the first column tells the name of data set used and the number of subsets in which it was divided i.e. for example Iris, 3S indicates Iris data set was divided into 3 subsets. So, there will be 3 partitions in the ensemble for CBEC and for SP it means it will load exactly 1/3 of the examples into memory at a time. For GC the whole data is loaded into memory and hard-k-means or fuzzy-k-means is applied. So, its value is the same for different subset experiments on a data set. Table I and Table II show that the quality of final clustering obtained by CBEC is better (higher) than or very close to the others. This is not a trivial accomplishment because each partition is formed from clustering disjoint data subsets which provides a scalable framework also.

Tables III, IV, and V show the result of comparing our algorithm with five others reported in [2] and SP. Except SP all are multiple partition combining algorithms. Still, we have kept the result of SP for comparison purposes. To provide a fair comparison we compare with the Random Initiation (RI) results in [2] only, where as done here the number of clusters in each clustering in the ensemble is same as the number of cluster in the consensus function, which is equal to the number of true classes in the data i.e. 3 for Iris, 6 for ISOLET, and 10 for Pendig. This provides a little built in advantage to them because the clustering solutions in the ensemble of RI experiments were formed from clustering all the data at a time whereas in our case ensemble is formed from disjoint subsets. Another advantage they had is on average they had more partitions in the ensemble than us. The NMI shown in Table III, Table IV, Table V is then the mean of those results for each data set. The NMI for CBEC and SP is also the mean over each data set. As they used hard-k-means only for ensemble creation the results of CBEC are of hard-k-means also.

Table III shows that CBEC achieved the best NMI on Iris when compared to the other algorithms. Although in Table IV CBEC achieves the best NMI, we are unable to do a complete comparison except SP on this data set because ISOLET6 was formed by selecting 6 classes randomly from 26 classes. So, for none of the RI experiments in [2], which 6 classes

they used was not known [33]. In our case all experiments have the following 6 classes chosen 2, 4, 8, 11, 17, and 25. For data set Pendig, Table V, CBEC is better than 3 other algorithms while 3 other are better than CBEC. In summary CBEC is better or competitive in quality to other state of the art multiple cluster combining and single pass algorithm.

TABLE I
NMI COMPARISON ON THREE DATA SETS USING HARD-K-MEANS AS THE CLUSTERING ALGORITHM

	GC	MNE	CBEC	SP
Iris, 3S	0.710896	0.747758	0.751706	0.715301
Iris, 5S	0.710896	0.721192	0.752983	0.712143
Isolet6, 5S	0.815032	0.811695	0.818239	0.804961
Isolet6, 10S	0.815032	0.790612	0.794605	0.790622
Pendig, 5S	0.682015	0.673280	0.654750	0.664840
Pendig, 10S	0.682015	0.674296	0.644739	0.659348

TABLE II
NMI COMPARISON ON THREE DATA SETS USING FUZZY-K-MEANS AS THE CLUSTERING ALGORITHM

	GC	MNE	CBEC
Iris, 3S	0.749638	0.765465	0.762541
Iris, 5S	0.749638	0.748654	0.756009
Isolet6, 5S	0.833225	0.811695	0.838923
Isolet6, 10S	0.833225	0.832535	0.833654
Pendig, 5S	0.682055	0.674656	0.673104
Pendig, 10S	0.682055	0.671220	0.666417

TABLE III
NMI COMPARISON OF CBEC WITH OTHER ENSEMBLE BASED ALGORITHM AND THE SINGLE PASS ALGORITHM ON THE IRIS DATA SET

SCEC	CSPA	MCLA	QMI	MMEC	CBEC	SP
0.7506	0.6938	0.7473	0.6687	0.7169	0.7523	0.7137

TABLE IV
NMI COMPARISON OF CBEC WITH OTHER ENSEMBLE BASED ALGORITHM AND THE SINGLE PASS ALGORITHM ON THE ISOLET6 DATA SET

SCEC	CSPA	MCLA	QMI	MMEC	CBEC	SP
0.7050	0.7011	0.7081	0.6154	0.6631	0.8064	0.7977

The three data sets we have used have true class labels, but labels will not typically be available. In the absence of real class labels one way of evaluating cluster quality is computing sum of squared error criterion, which is sum of the square of Euclidean distance of the examples in a cluster from their geometric mean. It intuitively says how compact or scattered the examples are in clusters. It is defined as follows [30]: For a data set $D = \{x_1, \dots, x_n\}$ of n examples, the task of clustering is to partition it into k disjoint subsets D_1, \dots, D_k . The sum of squared error (SE) of examples in

TABLE V

NMI COMPARISON OF CBEC WITH OTHER ENSEMBLE BASED ALGORITHM AND THE SINGLE PASS ALGORITHM ON THE PENDIG DATA SET

SCEC	CSPA	MCLA	QMI	MMEC	CBEC	SP
0.6756	0.6367	0.6791	0.6467	0.6469	0.6497	0.6620

the clusters is then $SE = \sum_{i=1}^k \sum_{x \in D_i} \|x - m_i\|^2$, where $m_i = \frac{1}{n_i} \sum_{x \in D_i} x$, n_i the number of examples in D_i . Generally this criteria is minimized in many iterative clustering algorithms. We have compared the quality of clusters from CBEC with the iterative GC and SP algorithms using the above criterion. In Table VI, it can be seen that CBEC is better (lower value) on Iris and competitive to GC and SP on ISOLET6 and Pendig in minimizing the sum of squared error criterion.

TABLE VI

COMPARISON OF CBEC WITH GLOBAL CLUSTERING AND SINGLE PASS ALGORITHM BY MEASURING SUM OF SQUARED-ERROR OF EXAMPLES IN CLUSTERS FROM THEIR MEANS

	GC	CBEC	SP
Iris, 3S	94.48	82.18	99.60
Iris, 5S	94.48	84.14	98.15
Isolet6, 5S	84697.03	84900.56	85280.31
Isolet6, 10S	84697.03	85998.40	85708.75
Pendig, 5S	15033257.03	15619284.38	15484834.63
Pendig, 10S	15033257.03	15859731.86	15563107.05

In all experiments, we assumed data is randomly divided into equal subsets i.e. each subsets contains a uniform distribution of all classes. But this is an ideal condition which will not always happen in a real life scenario. We created one type of unbalanced distribution of the Iris data as shown in Table VII. The data is divided into 5 equal sized subsets and except the first subset; all 4 other subsets have examples from one class missing. This data set was used to provide an indication of what might happen in the case that the random subsets got one type of poor distribution of examples. In Table VIII we show the result of CBEC without filtering (CBEC'') and with filtering (CBEC). As expected the average NMIs of the individual partitions (MNE) are quite low because 4 out of 5 subsets got a poor distribution. Still, CBEC with filtering succeeded in recovering a final clustering solution, which is much better compared to the average NMIs of the partitions in the ensemble. For hard-k-means the final clustering solution is better than GC and for fuzzy-k-means it is comparable to GC. In both cases result after filtering is much better than without filtering. So, it seems that with the filtering turned on we can recover a final clustering solution, which is better or comparable to GC even in the presence of a poor distribution in the majority of the subsets.

We analyze the time complexity of the multiple partitions

TABLE VII

UNBALANCED DISTRIBUTION OF IRIS DATA INTO 5 SUBSETS. S1,S2,...,S5 ARE THE SUBSETS.

	S1	S2	S3	S4	S5
Class1	10	0	20	10	10
Class2	10	20	0	0	20
Class3	10	10	10	20	0

TABLE VIII

RESULTS ON IRIS DATA SET SHOWING THE EFFECT OF FILTERING IN NMI

	GC	MNE	CBEC''	CBEC
Hard-k-means	0.7108	0.6168	0.6496	0.7244
Fuzzy-k-means	0.7496	0.6383	0.6681	0.7156

combining algorithm below. For SCEC it is $O(tnr^2k)$, where t is the number of iterations required in SCEC, CSPA is $O(n^2kr)$ [1], MCLA $O(nk^2r^2)$ [1]. It has been mentioned in [2] QMI and MMEC have the same time complexity as SCEC. Thus all of the above algorithms have the factor n in their time complexity. In CBEC, centroids were used to represent the ensemble, thus the time complexity of our algorithm is free from n i.e. $O(rk^2f + rk^3 + kr^2)$, where f is the dimension of the data. Thus compared to the other multiple clusterings combining algorithms our algorithm should scale extremely well for large data sets in terms of time and memory as it doesn't depend upon the size of the data and for most cases the number of centroids is not likely to be large. Even if for very large dimensional data sets if k remains small our algorithm will be fast, provided f is not as large as the size of data.

In comparison to iterative clustering algorithms GC and SP, Tables IX and X show how fast clustering would be if the subsets are clustered parallelly using hard-k-means and fuzzy-k-means respectively and then the partitions merged using CBEC. Time is computed in seconds and the results are on data set ISOLET6 and Pendig as the Iris data set is too small to report time accurately. SP is a single pass sequential algorithm, so it cannot be used to merge distributed partitions. For merging the partitions using CBEC, if the subsets are clustered parallelly using different machines then it has to wait until all the partitions are available. So, the time reported for CBEC includes the largest of the clustering time taken by the subsets and the time required for merging the partitions. All experiments were run on UltraSPARC-III+ with one 900 MHz processor with 1024 MB memory. We also reported the partitions merging time of CBEC separately in the last column (column 5) of the tables to show how fast merging of multiple partitions using centroids are. As expected, Tables IX and X show that the merging time of CBEC is very fast because the number of centroids are generally very small compared to the size of a data set. If the data is centrally stored, there would be some overhead in distributing the data. Also there would be some time spent in transferring the centroids through the network for merging. This overheads

are assumed to be very small compared to clustering time of large data sets. Tables IX and X mainly demonstrate the comparison in terms of clustering time. In summary, as expected, distributed clustering of the data will be several times faster than GC and SP. The clustering time reported for GC is by loading the whole data in memory. But for large data sets disk access will take place for every iteration making GC unacceptably slow.

TABLE IX
COMPARISON OF TIME (IN SECONDS) OF CBEC WITH GC AND SP FOR
HARD-K-MEANS EXPERIMENT

	GC	SP	CBEC	CBEC Merging Time
ISOLET6, 5S	2.8168	2.1412	0.6568	0.0062
ISOLET6, 10S	2.8168	2.0458	0.3438	0.0148
PENDIG, 5S	0.4152	0.1632	0.0834	0.0052
PENDIG, 10S	0.4152	0.1386	0.0414	0.0056

TABLE X
COMPARISON OF TIME (IN SECONDS) OF CBEC WITH GC FOR
FUZZY-K-MEANS EXPERIMENT

	GC	CBEC	CBEC Merging Time
ISOLET6, 5S	95.9	31.53	0.0054
ISOLET6, 10S	95.9	23.50	0.0144
PENDIG, 5S	47.77	11.70	0.0056
PENDIG, 10S	47.77	5.91	0.0068

Each experiment with CBEC consisted of 50 random initializations using hard-k-means or fuzzy-k-means, where the order of selecting partitions was random for each merging operation. We re-ran the experiment 32 times with the same set of centroids obtained from 50 random initializations; each time the centroid combination order was random as before to estimate how sensitive CBEC is to the random selection of order. Sensitivity is determined by computing the average of the absolute difference of quality in SE of each experiment from its mean. As, SE values vary by data sets, so we normalized it by dividing by its mean and then multiply by 100 to express it as a percentage. This will indicate how much the quality of partitions could vary from their mean quality on average. Sensitivity is The computed as below:

$$Sensitivity = \frac{1}{n} \sum_{i=1}^n \left(\frac{|p_i - m_i|}{m_i} \right) * 100, \text{ where } n \text{ is } 32, p_i \text{ is the average quality in SE of each experiment, and } m_i = \sum_{i=1}^n p_i.$$

TABLE XI
ESTIMATION OF THE SENSITIVITY OF CENTROIDS COMBINATION ORDER
ON AVERAGE

	Iris, 5S	Isolet6, 10S	Pendig, 10S
Hard-k-means	1.30	0.30	0.45
Fuzzy-k-means	1.13	0.18	0.31

It is seen in Table XI, the sensitivity of the quality of CBEC to the order of partitions is not much on average.

VIII. CONCLUSIONS

In this paper we proposed a method for merging clustering solutions in an ensemble within a framework that is scalable for large data sets in terms of time and memory complexity. Although we form the ensemble from disjoint subsets, we have shown that a final clustering of better or equivalent quality can be achieved compared to global clustering and the average NMIs of individual partition in the ensemble. We have also shown that our method is better or competitive with other relevant algorithms in the literature in terms of quality while maintaining superiority in terms of time and memory complexity. Our algorithm is also capable of detecting and removing malformed clusters from the ensemble to provide a robust framework. Future work could be done to handle different number of clusters in different clustering solution, experiments on large data sets, and automatically detecting the threshold value in centroid filtering.

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] B. Long, Z. (Mark) Zhang, "and Philip S. Yu, " Combining Multiple Clusterings by Soft Correspondence," *ICDM 2005*, pp. 282-289.
- [3] K. Punera and J. Ghosh, "A scaleable and Robust Framework for Structure Discovery," *ICDM 2005*, pp. 757-760.
- [4] H. Kriegel, P. Kroger, A. Pryakhin, M. Scubert, "Effective and Efficient Distributed Model-based Clustering," *ICDM 2005*, pp. 258-265.
- [5] X.Z Fern and C.E Brodley. "Solving cluster ensemble problem by bipartite graph partitioning," *ICML 2004*.
- [6] B. Fischer and J.H Buhmann, "Path-based clustering for grouping of smooth curves and texture segmentation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2003.
- [7] A. Topchy, A.K Jain, W. Punch, "A mixture model for clustering ensembles," *Proc. AIAM Data mining*, 2004
- [8] S. Dudoit and J. Fridlyand "Bagging to improve the accuracy of a clustering procedure," *Bioinformatics*, 2003
- [9] G. Forman and B. Zhang, "Distributed Data Clustering can be Efficient and Exact," *SIGKDD Explorations*, 2000
- [10] Kuhn, H.W., "The Hungarian Method for the Assignment Problem," *Naval. Res. Logist. Quart.*, 2, 1955, pp. 83-97.
- [11] P. Hore and L. Hall, "Scalable Clustering: A Distributed Approach," *FUZZ-IEEE*, 2004.
- [12] A. Topchy, A.K Jain, and W. Punch, "Combining Multiple Weak Clusterings," *Proc. IEEE Intl. Conf. on Data Mining*, 2003, pp. 331-338.
- [13] I. Davidson and A. Satyanarayana, "Speeding up K-means clustering by Bootstrap averaging," *IEEE ICDM 2004*.
- [14] A.L.N Fred, "Finding Consistent Clusters in Data Partitions," *Proc. Int. Workshop on multiple Classifier Systems*, Eds. F. Roli, J. Kittler, LNCS 2364, 2002, pp. 309-318.
- [15] F. Farnstrom, J. Lewis, and C. Elkan, "Scalability of Clustering Algorithms Revisited," *SIGKDD Explorations*, 2(1), 2000, pp. 51-57.
- [16] 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, pp. 262-270 2003.
- [17] Ganti, V., Gehrke, J. and Ramakrishnan, R., "Mining very large databases," *Computer*, August, 1999, 38-45.
- [18] Zhang, T., Ramakrishnan, R. and Livny, M., "BIRCH : An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, ACM Press, 1996 NY, 103-114.
- [19] Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A. L. and French, J. C., "Clustering Large Datasets in Arbitrary Metric Spaces," *Proc. 15th Int'l. Conf. on Data Engineering*, IEEE CS Press, Los Alamitos, CA, 1999 502-511.
- [20] Bradley, P., Fayyad, U. and Reina, C., "Scaling clustering algorithms to large databases," *Proc. 4th Int'l. Conf. Knowledge Discovery and Data Mining*, AAAI Press., Menlo Park, CA, 1998, 9-15.

- [21] Domingos, P. and Hulten, G., "A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering," *Proc. Eighteenth Int'l. Conf. on Machine Learning*, 2001, 106-113.
- [22] Behrouz Minaei-Bidgoli, Alexander Topchy, William F. Punch, "Ensembles of partitions via data resampling,"
- [23] AK Jain and RC Dubes, "Algorithms for Clustering Data. Prentice Hall," Englewood Cliffs NJ, USA, 1988
- [24] Inderjit S. Dhillon and Dharmendra S. Modha, "A Data-Clustering Algorithm On Distributed Memory Multiprocessors," *Proc. of Large-scale Parallel KDD Systems Workshop*, ot ACM SIGKDD, 1999, pp. 245-260.
- [25] Joydeep Ghosh and Srujana Merugu, "Distributed Clustering with Limited Knowledge Sharing," *Proc. 5th International Conference on Advances in Pattern Recognition*, pp. 48-53, Dec 2003.
- [26] Joydeep Ghosh, Alexander Strehl, and Srujana Merugu, "A Consensus Framework for Integrating Distributed Clusterings Under Limited Knowledge Sharing," *Proc. National Science Foundation (NSF) Workshop on Next Generation Data Mining*, 2002, pp. 99-108
- [27] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle, "Towards Effective and Efficient Distributed Clustering," *Proc. Int. Workshop on Clustering Large Data Sets, 3rd IEEE International Conference on Data Mining*, 2003, pp. 49-58
- [28] Matthias Klusch, Stefano Lodi, and Gianluca Moro, "Distributed Clustering Based on Sampling Local Density Estimates," *Proc. Eighteenth International Joint Conference on Artificial Intelligence*, 2003, pp. 485-490.
- [29] 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>.
- [30] R. Duda, P. Hart, and D. Stork "Pattern Classification"
- [31] N.R. Pal and J.C. Bezdek. "Complexity reduction for large image processing," *Transactions on Systems, Man, and Cybernetics - Part B*, 2002, pp. 598-611.
- [32] Cheng T.W., Goldgof D.B., and Hall L.O. "Fast fuzzy clustering," *Fuzzy Sets and Systems*, 1998, pp. 4956.
- [33] Personal communication: We communicated with one of the authors of [2]. We were told that their experiment might choose 6 classes randomly and they have no record of it except the last experiment.