Pattern Recognition 42 (2009) 676 – 688

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A scalable framework for cluster ensembles ☆

Prodip Hore, Lawrence O. Hall *, Dmitry B. Goldgof

Department of Computer Science and Engineering, ENB118 University of South Florida, Tampa, FL 33620, USA

ARTICLE INFO

Article history:
Received 5 October 2007
Received in revised form 22 August 2008
Accepted 16 September 2008

Keywords:
Clustering
Hard/fuzzy-k-means
Large data sets
Ensemble
Scalability
Single pass algorithm

ABSTRACT

An ensemble of clustering solutions or partitions may be generated for a number of reasons. If the data set is very large, clustering may be done on tractable size disjoint subsets. The data may be distributed at different sites for which a distributed clustering solution with a final merging of partitions is a natural fit. In this paper, two new approaches to combining partitions, represented by sets of cluster centers, are introduced. The advantage of these approaches is that they provide a final partition of data that is comparable to the best existing approaches, yet scale to extremely large data sets. They can be 100,000 times faster while using much less memory. The new algorithms are compared against the best existing cluster ensemble merging approaches, clustering all the data at once and a clustering algorithm designed for very large data sets. The comparison is done for fuzzy and hard-k-means based clustering algorithms. It is shown that the centroid-based ensemble merging algorithms presented here generate partitions of quality comparable to the best label vector approach or clustering all the data at once, while providing very large speedups.

1. Introduction

Clustering algorithms are used to partition unlabeled data into groups or clusters. Clustering data is often time consuming. This is especially true of iterative clustering algorithms such as the k-means family [1] or EM [2]. As larger unlabeled data sets become available, the scalability of clustering algorithms becomes more important. There are now unlabeled data sets which vastly exceed the size of a typical single memory [3,4]. Subsets of data can be clustered in such a way that each data subset fits in memory and finally the clustering solution of all subsets can be merged. This enables extremely large data sets to be clustered. Sometimes, data is physically distributed and centrally pooling the data might not be feasible due to privacy issues and cost. Thus, merging clustering solutions from 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 may provide a robust and stable solution [5,6]. Also, combining multiple partitions may produce novel clustering solutions which might not be possible when clustering all the data [5,6]. In Ref. [7], it has been shown that consensus clustering converges to the true underlying distribution as the ensemble size grows, further justifying the use of cluster ensembles. So, combining multiple clustering solutions has emerged as an important area of research to address the issue of scalability, the distributed nature of some data, and the robustness or stability of the clustering solution.

Current research on combining ensembles of clustering solutions has focused on what we are going to call high-resolution representations [5,6,8–17]. Label vectors for each example or a co-association matrix or a similar representation have been used. These representations have a size which is on the order of the number of examples in a partition. As the number of examples becomes large to very large, both storage and time costs scale poorly with these approaches.

The work presented in this paper focuses on scalable methods of combining clustering solutions. We use a low resolution representation of the clustering solutions (partitions) in the form of cluster centroids. The centroids are insensitive to the number of examples from which they are created. Our approach does not require labels from examples which is an advantage when combining clustering solutions from object distributed data [5]. With object distributed data there may be no overlap between the arbitrary labels created from disjoint data sets making the label resolution difficult. This could prove an obstacle to creating scalable clustering solutions by applying a distributed clustering algorithm where solutions created on subsets must be merged.

There are many ways to create an ensemble of data partitions by clustering and we will touch on the most common. You can exploit

☆ This research was partially supported by the National Institutes of Health under Grant no. 1 R01 EB00822-01.

* Corresponding author. Tel.: +1 8139744195; fax: +1 8139745456.
E-mail addresses: usfprodip@gmail.com (P. Hore), hall@csee.usf.edu (L.O. Hall), goldgof@csee.usf.edu (D.B. Goldgof).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.09.027

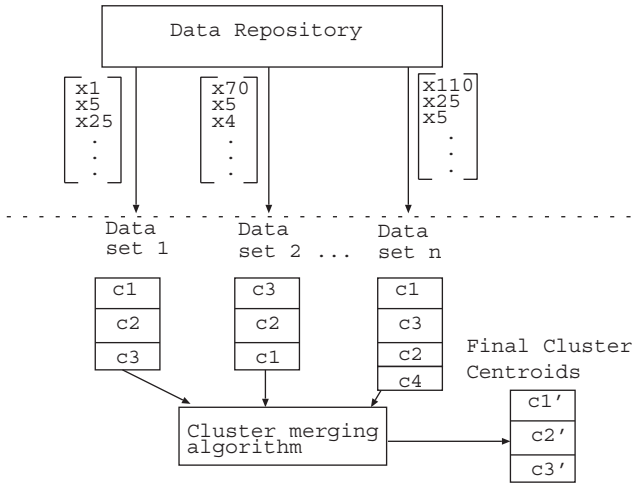

Fig. 1. An illustration of cluster ensemble creation and merging. Data may be extracted, with or without replacement, from a central repository. Alternatively, the data sets may exist already (no feed in from above the dotted line). The number of clusters may vary. Data are finally partitioned into three classes c1′–c3′.

the sensitivity of the clustering algorithms to initialization and simply create r partitions by initializing the algorithm differently each time. Disjoint subsets of the data may be clustered or overlapping subsets of the data may be clustered. In studying scalability of partition combining algorithms, we are primarily interested in disjoint data subsets, which may be created to reduce the clustering time required, or overlapping data sets. Also of interest are data sets that are by nature distributed, for example data of the same type that is owned by different companies. They might be willing to share their central tendencies, but could not/would not share the data itself. Fig. 1 illustrates the problem of creating a final partition from all of the data. Another example is when very large data sets of the same type are geographically distributed. The distributed solutions must be combined and we will analyze two approaches to combining centroids. The cluster to which any given example belongs can be determined by using the nearest centroid using any similarity metric (e.g. Euclidean distance). Certainly cluster centroids can be easily distributed to different companies or computers for determining to which cluster the examples belong.

We do not require the feature values of examples to produce the final clustering solution. Hence, we have a knowledge reuse and privacy preserving data mining framework which can be used to merge already clustered solutions without knowing the underlying data.

The main focus of this paper is to show how an ensemble of centroids created from object distributed data, using fuzzy-k-means and hard-k-means algorithms, can be merged in a scalable framework. We have proposed two methods to merge ensembles of centroids. We show a robust, quality final clustering solution was obtained compared to a relevant multiple partition combining algorithm (MCLA) [5], while providing superiority in terms of time and space complexity. The quality of our ensemble merging algorithms was also compared to the quality of clustering all the data at once in memory, the average base clustering (BC) solutions in the ensemble, and to a scalable single pass (SP) algorithm.

2. Related work

Creating and combining multiple partitions of a given data set has been examined in several ways [4–17]. In Ref. [6], it has been pointed out that existing merging algorithms suffer from a time complexity problem. In most of the approaches which use high resolution representations, each clustering solution in the ensemble will have n, the number of examples, in its time and memory complexity equations. In Refs. [16,17], to speed up their multiple partition combining algorithm they used a sample of the original data set and then extended the results to global data. However, for large or extreme data sets even a subsample may be quite large.

In Ref. [5] hyperedges were formed from each clustering solution represented as a label vector. Using these hyperedges, a final partition was obtained by using graph partitioning algorithms like cluster-based similarity partitioning (CSPA), hypergraph-partitioning (HGPA), and meta-clustering (MCLA). In Ref. [8] clustering solutions in the ensemble were 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 was then obtained by soft correspondence, where each cluster from one partition corresponded to clusters in other partitions with a different weight. In Refs. [12,14], weak clusters of a data set were formed by using random hyperplanes 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 representations of BC were in the form of label vectors, and the approach might be complicated for large data sets because a large number of partitions might have to be created to get a good overall partition. In Refs. [13,18] clusters have been merged using co-association matrices, but the approach is computationally expensive. In Refs. [10,11,19] combining clustering solutions from multiple bootstrap samples was discussed. In Ref. [20], a method for fast and efficient ensemble clustering was proposed using the Leader algorithm [21]. However, both the ensemble creation and merging algorithm use parameters particular to the Leader algorithm, and the method may not generalize to ensembles created from the k-means family or other clustering algorithms.

As in our framework, disjoint subsets of data can be clustered in parallel and used independently or merged to allow clustering to scale for large data sets, so we also survey scalable clustering algorithms. In recent years a number of new clustering algorithms have been introduced to address the issue of scalability [1,22–28]. All the algorithms assume that the clustering algorithm was applied to all the data centrally pooled in a single location. Various methods for expediting fuzzy-k-means have been explored. In Ref. [29] data subsampling was proposed and a good speedup of fuzzy-k-means was obtained. Similarly, in Ref. [30] a multistage random sampling algorithm was proposed to speed-up fuzzy-k-means. In Ref. [31], a density based data reduction method has been discussed, which condenses a data set into fewer examples. Recently two online clustering algorithms have been proposed [32]. A parallel implementation of k-means [33] or other similar algorithms is not applicable to cluster ensembles as they do not combine multiple clusterings.

In Refs. [34,35] distributed clustering has been discussed under limited knowledge sharing. In Ref. [35] a clustering solution has been represented by labels while in Ref. [34] generative or probabilistic model parameters were sent to a central site, where “virtual samples” were generated and clustered using an EM algorithm [2] to obtain the global model. In Ref. [36], data at local sites were first clustered using the DBSCAN algorithm and then representatives from each local site were sent to a central site, where DBSCAN was applied again to find a global model. Another density estimation based distributed clustering algorithm has been discussed in Ref. [37]. In Ref. [38], local models or prototypes were detected using EM [2] in a distributed environment and later merged by computing mutual support among the Gaussian models. In Ref. [39] a method for speeding up the EM algorithm by aggregating subclusters obtained from a data set was discussed. However, the objective functions optimized by the above algorithms are different from those optimized by the hard c means and the fuzzy c means algorithms. Moreover, the above algorithms

are not known to have been studied in the context of merging cluster ensembles for large or very large data sets in terms of time and space complexity.

Recently in Ref. [40], improving the clustering solution by weighting examples in a concept analogous to boosting in supervised learning was addressed, where less efficiently clustered points are given more weight. However, it does not fall under the class of multiple partition combining algorithms.

The contribution in this paper is twofold. First, we study scalability issues, in terms of time and space complexity, for merging ensembles for large or very large data sets. To our knowledge, algorithms for merging cluster ensembles have not been studied for large or very large data sets. Here, we address the merging of multiple partitions formed from the widely used fuzzy-k-means and hard-k-means algorithms in a scalable framework. Second, we propose an algorithm to filter noisy clusters to prevent "blind" merging of BC solutions, which to our knowledge has not been previously addressed. The performance of algorithms under noise was studied in Ref. [5].

An algorithm which merges clusters from multiple partitions by doing the best possible match without filtering may be said to do blind merging. We will discuss a simple method to remove outlier or noisy clusters which allows only a subset of clusters to be used to produce a final partition. As an example, if there are five partitions in an ensemble, each of three clusters, there are 15 clusters in the ensemble. One final centroid may be created from centroids from two of the five partitions, another from four of five and the last from all five. The unused centroids will be considered outliers or noisy. This approach provides a mechanism to deal with noise in the data or an unbalanced distribution (e.g. one or more classes is mostly missing in one data set). The essential idea is to identify the closely matching centroids and then filter out the outliers or "bad" centroids.

Although the concept of filtering and merging multiple clustering solutions (bipartite merger, BM algorithm) have been introduced in our earlier work [3,4], in this work we propose and evaluate another merging algorithm, the Metis merger (MM). We perform experiments on large or very large data sets from several domains, and quantitatively evaluate the performance of all of them compared to a well known multiple partition combining algorithm, clustering all data at once in memory, the average BC solution, and a SP algorithm.

3. Ensemble merging

Our first algorithm, BM, uses the assumption that the number of clusters in the BC, the clustering solutions in the ensemble, are the same for all partitions. However, the true number of classes in each partition may not always be the same.

So, if we set the number of clusters to k, the maximum number, what effect it will have on different mixtures?

Case A: If less than k classes are present in a subset, then we are overclustering. Overclustering may not cause any information loss as it will likely split homogeneous clusters. Information loss only occurs when we undercluster, which must group non-alike examples together.

Case B: If exactly all k classes are present, then we are neither overclustering nor underclustering.

The above holds if in each subset there exist enough examples of a class to form a cluster. In the case that a BC had a very small number of examples from a class which were not enough to form their own cluster, they will get mixed in with a cluster of another class. So, setting the number of clusters always equal to k, the maximum number of classes in the data set, may not cause any information loss in generating cluster ensembles in the cases that there are adequate numbers of each class present in a BC or classes are absent from a BC. Thus, we set the number of clusters to the maximum number of classes in all our experiments for generating ensembles for BM. Our second algorithm, MM, does not use the assumption that the number of clusters in each BC is the same. So for MM, one may also generate an ensemble with different numbers of clusters in the base clustering, provided the knowledge of the true number of classes is available. If that knowledge is not available, one may generate an ensemble by using the maximum possible number of classes, which should minimize any information loss. The knowledge of the true number of classes may not always be easily available. This is especially true for data divided in an unbalanced way. For example, knowing a data set has 12 clusters we may attempt to divide it into 10 disjoint subsets for generating ensembles. Now, if the data are divided randomly into 10 subsets, it is likely that the number of classes in each disjoint subset will also be 12; however, if the data are divided in an unbalanced way, it is difficult to know how many classes are in different subsets. We are certain that the maximum possible in any subset is 12.

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 it becomes a r-partite graph, where r is the number of partitions to combine. Our objective is to partition this r-partite graph into final target clusters, such that "similar" centroids are grouped together. If the base clusterings/partitions have an equal number of clusters and meaningful correspondences exist among them, the group size will tend to be equal. The centroids in each group will then be used to compute a global centroid. The partitioning of this r-partite group is like clustering the clustering solutions, and optimizing it is generally an NP hard problem.

So, our two heuristic algorithms, BM and Metis Merger, attempt to partition the ensemble of centroids, the r-partite graph. The BM algorithm partitions the ensemble of centroids into exactly equal size groups using the constraint of one to one centroid mapping between any two partitions. Hence, the number of target clusters must be the same as the number of clusters in each base partition. Although this assumption is somewhat restrictive, it has also been used for re-labeling purposes [10,11]. In many cases the underlying distributions of data in subsets slowly changes or is similar, and in those cases assumption of one to one correspondence generally holds. MM tends to partition ensembles of centroids, using the graph partitioning package METIS [41–43], into approximately equal size groups. Note, in MM the group sizes may not be the same and it does not require the number of clusters in the BC to be the same as the target clustering.

The assumption of an equal number of clusters in all base clustering solutions, especially in BM, will not cover all cases, for example the BC solutions may already exist with a different number of clusters, and in this case only a knowledge reuse framework is needed for merging. We do not specifically address this type of scenario for BM in this work. However, one may use our MM approach for an unequal number of clusters in the partitions.

The focus of this paper is on scalability, and we will show that for the same ensembles, a centroids based representation is better than or as good as the label vector based representation, while providing significant superiority in terms of time and space complexity.

In the first method, we merged the ensemble using the bipartite matching algorithm, and named it the BM. The second method uses the graph partitioning package METIS [41–43], and is called the MM. METIS was also used by Strelh and Ghosh [5] in their multiple partition combining algorithm, MCLA, to merge base clusterings represented in the form of label vectors. We have used METIS to merge BC solutions represented in the form of centroids. To our knowledge, we are the first to use this graph partitioning package to merge an ensemble of centroids. Below, we describe our two ensemble merging algorithms.

Between any two partitions, BM optimally matches clustering solutions, but the constraint of one to one mapping may force a matching of clusters without meaningful correspondence. This might

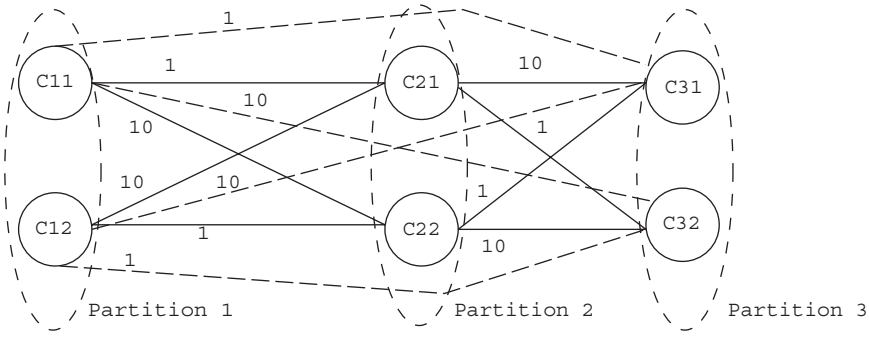

Fig. 2. Partitions to be matched with edges whose weights are inverses of the distances between the centroids. For Bipartite merging the raw distances would be used. The dotted lines (links) are only needed for Metis merger.

happen if one or more BC are relatively noisy or have originated from an unbalanced distribution. To address this problem and make our framework robust, we have proposed a filtering algorithm, described later, that filters out "bad centroids" from merging. The filtering concept has also been applied to MM to remove spurious centroids, if any.

We will evaluate our algorithms on ensembles created from both a balanced distribution, created by randomly dividing a data set, and an unbalanced distribution. The number of clusters has been kept at the maximum possible, k, for each clustering solution and is the same as the target number of clusters. We will be comparing BM with MM and also to MCLA, a well known relevant ensemble merging algorithm. Thus to provide a fair comparison, the same BC solutions were used both for MM and MCLA.

3.1. Bipartite merger

After clustering was applied to each disjoint subset, there was a set of centroids available which described each partitioned subset. Clustering or partitioning a subset $\{S_i\}$ produces a set of centroids, BC solutions, $\{C_{i,j}\}_{j=1}^{k}$, where k is the number of clusters. When r subsets are chosen there will be r sets of centroids i.e. $\{C_{1,j}\}_{j=1}^{k}$, $\{C_{2,j}\}_{j=1}^{k}, \ldots, \{C_{r,j}\}_{j=1}^{k}$ 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 partition the ensemble of centroids into k consensus chains, where each consensus chain will contain r centroids $\{c_1^l, \ldots, c_r^l\}$, one from each of the subsets, where l 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 the set f of all possible families of centroid assignments to k consensus chains:

$$\psi^* = \arg\min_{\psi \subset f}\{\psi\}, \tag{1}$$

$$\psi = \sum_{l=1}^{k} \cos t(consensus\_chain(l)), \tag{2}$$

$$\cos t(consensus\_chain(l)) = \sum_{i=1}^{r} D^l(i), \tag{3}$$

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

where $d(.,.)$ is the distance function between centroid vectors in a consensus chain. We used the Euclidean distance in computing the cost (4), as the underlying clustering solutions were also obtained using the Euclidean distance metric. So, in a graph theoretical formulation finding the globally optimum value for the objective function (1) reduces to the minimally weighted perfect r-partite matching problem, which is intractable for $r > 2$. Because the optimization problem is intractable (NP-hard), a heuristic method was used to group centroids.

We know that for two partitions, $r = 2$, 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 two partitions we used the Hungarian method of minimally weighted perfect bipartite matching [44]. After matching a pair of partitions, we keep the centroids of one of the pair 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. 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. In some cases, especially in the knowledge reuse framework, weights/importance of BC solutions may be difficult to obtain. In those cases, all the BC solutions may be considered to have the same weight/importance. Each consensus chain tells us which centroid from which subset is matched to a centroid in another subset. A final partition, in the form of label vectors, may be obtained by assigning an example to the nearest global centroid. It should be noted that the BM algorithm partitions the ensemble of centroids into k perfectly balanced chains, that is, each chain has the same number of centroids (one from each BC solution). More about consensus chains can be found in our earlier work [3,4].

In Refs. [9,15], both examples and clusters were modeled as a graph in which an example vertex can connect only to a cluster vertex, thus formulating the problem of finding a consensus partition as a bipartite matching problem. In our case, we formulated it using the centroids only, thus the time and space complexity will not involve n, the number of examples in a data set.

Example. Consider the case that there are three subsets S1, S2, and S3 of a data set and each subset is grouped into two clusters. Then three partitions are produced as shown in Fig. 2. To later also use this figure as an example for our MM algorithm, we have included some extra detail. For now, ignore the dotted lines with weights. The edges are between each centroid (C_{ij}) from different partitions where i is the partition number and j the cluster number for the partition. The weights on the edges are the inverse of the distances

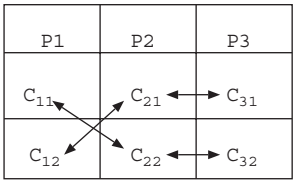

Fig. 3. A consensus chain is shown for each of two clusters. The arrows show cluster correspondences for the three partitions with Pi—partition *i*.

between the centroids. If we do the matching starting with partitions one and two and then matching partitions two and three we will get a consensus chain of *similar* centroids as shown in Fig. 3. It is clear from the edges that the highest weights (i.e. shortest distances) indicate which clusters would be grouped together. In BM, here, the raw distances are used as specified in Eq. 1.

3.2. Metis merger

In MM, the ensemble of centroids, the *r*-partite graph, is partitioned into *k* groups (target clusters). Fig. 2 shows an example of a weighted *r*-partite graph obtained from partitioning three subsets of data into two clusters. The MM algorithm is so named because it uses the graph clustering algorithms of the METIS package.

The algorithm does a multilevel *k*-way partitioning of the graph by coarsening the graph to reduce its size at each stage. It randomly selects a vertex and then adds an edge, which has the largest weight, to a vertex yet to be matched. This is continued until all vertices have been visited. Paired vertices are collapsed into a single vertex maintaining their edge weights. It is fast, $O(|E|)$. The coarsened graph is then *k*-way partitioned such that each partition contains approximately $1/k$ of the weight of the original graph. A fast multilevel bisection algorithm is used together with some refinements to produce the partition and is $O(k*m)$ for a graph with *m* edges. Details can be found in Refs. [41,42]. Note that our *r*-partite graph will have just $r*k$ vertices, where *k* is the number of clusters. Metis partitions the ensemble of centroids into approximately equal size groups; however, the groups may not be perfectly balanced, that is, the number of centroids in each group may not be the same. For example, consider a case in which a set of partitions had 2, 3, 2, and 3 clusters, respectively. We could require METIS to create three groups, by necessity, of unequal size. This is an important difference from BM, which guarantees a perfectly balanced grouping. We call each group from MM a consensus group. Similar to the concept of consensus chain in BM, it contains similar centroids grouped together. For our purposes the ability of MM to create groups of unequal sizes allows for classes of data that are split into two or more clusters to be handled properly, which can occur when one or more classes are not actually present in a particular data set.

For partitioning the ensemble of centroids using Metis, the weights of the *r*-partite graph have to be normalized so that the edge weights are proportional to the similarity between centroids, that is, similar centroids should be connected by high weighted edges and vice versa. The initial similarities can simply be the inverse of the Euclidean distance between centroids. Normalization is done by finding the maximum weight, *maxWeight*, of the *r*-partite graph and then all edge weights are subtracted from it. This will ensure that edge weights are proportional to the similarity between centroids. All edge weights are ensured to be non-zero by adding 1 to all weights so that the minimum weight is 1 and the maximum $maxWeight + 1$. After partitioning the centroids into *k* consensus groups, a global centroid is found by computing the weighted arithmetic mean of centroids in each group. Similar to BM, weights of a centroid are determined from the size of the subset from which it was created. A final partition, in the form of label vectors, may be obtained by assigning an example to the nearest global centroid.

In BM, we are optimally matching the centroids of two randomly picked partitions at a time. But the question arises, in which order should the partitions 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 would be 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 for the cluster correspondence problem the assumption of transitive relations is quite valid. One could also use an approximate traveling salesman algorithm to determine a sub-optimal minimum cost order. However, we will show sensitivity to different orders is low, implying the current algorithm is minimizing its objective function quite effectively. Many approximate algorithms may exist for solving the minimally weighted perfect *r*-partite matching problem, but we chose the bipartite matching algorithm because it is well known and has been used to solve correspondence problem in clustering [4,10,11].

MM uses the graph partitioning package METIS, which is a randomized algorithm. So, it is also sensitive to the order the vertices are numbered in the *r*-partite graph. We have conducted experiments to estimate the sensitivity of our algorithms. METIS is also a well studied graph partitioning package, and it has been used to combine multiple partitions represented by label vectors or their variants.

4. Filtering bad centroids

For very large data sets, it is possible for a class of data to get split into more than one cluster. It is also possible that an unfortunate initialization and the usual noise that will exist in any data set can cause a cluster to be formed that is not representative of any class of data. The centroid of that cluster will not fit well in any chain [4].

In BM and MM, each consensus chain/group contains matched centroids, and the problem is equivalent to removing outliers/noise from it, if present. Clusters are typically compact because this is a feature built into most clustering algorithms and particularly built into the k-means family of algorithms. In the rare case we have a very non-compact cluster, it is possible the centroid will fit into a chain to which it does not belong.

With a limited number of centroids in a consensus chain and no knowledge about the distribution of noise, detecting outliers is a non-trivial task. It seems reasonable to assume that "good" centroids are spatially close to each other. So, we expect a set of centroids that represent the same class to form a reasonably compact cluster. Thus for a consensus chain or, more broadly, a group of centroids believed to represent the same class we search for a compact cluster and any centroids that are distant from 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 was the weight of the edges. Next, we sorted the edges of the tree and found the statistical median. Now, if all edges of intermediate or high lengths are removed, this will reveal clusters [45] in the consensus chain. We have set the threshold to cut edges as: $\text{median} + 2.5 * \text{median}$. 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

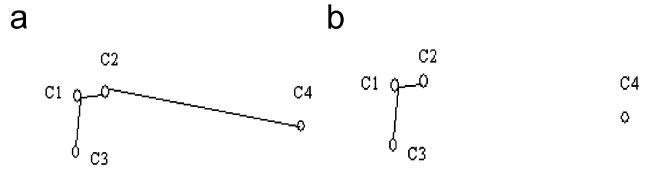

Fig. 4. (a) Minimally weighted spanning tree formed from centroids in a consensus chain/group. (b) After cutting edge(s) above threshold, the largest connected component (C1, C2, and C3) will be used for merging.

weighted arithmetic mean of these centroids in a consensus chain, where the weights of a centroid are determined by the size of the subsets it represents. Thus, each filtered consensus chain/group will result in a final centroid. In the case of MM, the consensus groups, formed after partitioning the r-partite graph, are similar in concept to the consensus chain in the BM. So, the concept of filtering is the same with a spanning tree built for each of the k consensus groups from the centroids in the kth group.

Example: Let each consensus chain (consensus group in the case of MM) have four centroids. As mentioned earlier, for MM the consensus group is not guaranteed to be of equal size; nevertheless, in this example we assume they are of equal size, that is, four centroids in each group. Fig. 4a shows an example of building a spanning tree from such a consensus chain/group and Fig. 4b 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.

5. Experimental setup

In Ref. [8] an extensive comparison of soft-correspondence ensemble clustering (SCEC) was done with four other state of the art algorithms on three real data sets Iris, Pen Digits, and Isolet6. The algorithms compared with were CSPA [5], MCLA [5], QMI [12], and MMEC [6]. CSPA and MCLA are graph partitioning algorithms while QMI uses k-means, and MMEC is based on mixture model based clustering. Although no single algorithm was a clear winner on all data sets, overall SCEC performed better in a majority of the cases followed closely by MCLA; however, they also reported that MCLA tends to give better partitions when each partition has the true number of clusters because that tends to result in nearly balanced clusters in the ensemble. In Ref. [5], it was stated that MCLA performs best, compared to HGPA and CSPA, when meaningful cluster correspondence is likely to exist. In Ref. [16], it was also reported that MCLA was a strong competitor to the PLA (probabilistic label aggregation) and wPLA (weighted probabilistic label aggregation) algorithms as compared to the three other algorithms CSPA [5], HGPA [5], and a mixture model based approach [6]. In our framework, we fixed the number of clusters in the BC solutions and target clustering to be the same, k (maximum possible). Hence, it is likely that cluster centroids in the ensemble will have meaningful correspondence (especially for a balanced distribution) in the BC solutions. So, we chose to compare our algorithms with MCLA, which is likely to perform well in this setting. Since MM and MCLA use the same graph partitioning package, METIS, a good framework was available for evaluating the quality of the ensemble formed from centroids versus an ensemble formed from label vectors. We re-implemented our algorithms in C. The METIS package (in C code), which MCLA and MM uses, is available at http://glaros.dtc.umn.edu/gkhome/views/metis.

MCLA requires representation of clustering solutions in the form of label vectors over all examples (global data) because the label vectors created from examples of disjoint subsets will not have any overlap and hence no meaningful correspondence. For MCLA a label vector equivalent representation of a BC solution in an ensemble was created, assigning an example to the nearest centroid in that partition, over all the examples.

It is well known that even if a data set is centrally available, loading all the data into memory and clustering it can be time consuming for large data sets. If we cannot load all data into memory, hard-k-means or fuzzy-k-means will have to make disk accesses every iteration. To address this problem, in Ref. [23] a SP clustering algorithm was described for hard-k-means, where only part of the data was loaded into memory at a time and the whole data set was clustered in one disk access. We compare to this approach because merging clustering solutions created from disjoint data is a way of scaling the clustering process of the full data set. It allows independent processors to cluster subsets simultaneously. Obviously, if data are geographically distributed (assuming centrally pooling the data is difficult) or have already been clustered (knowledge reuse), multiple partition combining algorithms must be used. The code for the SP hard-k-means algorithm [23] is available at http://www-cse.ucsd.edu/users/elkan/skm.html.

Measuring cluster quality is a non-trivial task. Unlike supervised learning, where the learning algorithm uses real class labels to minimize error over training examples, the true labels are not used to group data into different clusters. The sole purpose of a clustering algorithm is to try to optimize its objective function, which may result in a good set of clusters. Moreover, true labels will not typically be available. One way of evaluating cluster quality without using true labels is the sum of squared error (SE) criterion, which is, sum of the square of the Euclidean distance of the examples in a cluster from their geometric mean. It intuitively indicates how compact or scattered the examples are in clusters. We have used this metric to evaluate and compare clustering quality in our experiments.

It is defined as follows [45]: For a data set $D=\{x_1,\ldots,x_n\}$ of n examples, the task of clustering is to partition it into k disjoint subsets $D_i,\ldots,D_k$. The sum of SE of examples in the clusters is then $SE=\sum_{i=1}^{k}\sum_{x\in D_i}\|x-m_i\|^2$, where $m_i=1/n_i\sum_{x\in D_i}x$, $n_i=|D_i|$. Generally, this or a variant are minimized in many iterative clustering algorithms.

In summary, using the SE metric we compared the quality of our multiple partition combining algorithms, BM and MM, with MCLA against clustering with all the data at once in memory (global clustering, GC). The average quality of BC solutions, and a SP algorithm (for hard clustering only) were also compared to GC. We will show the ensemble formed by our centroid based approach has almost the same quality (sometimes even better) compared to the label vector based approach, MCLA. Further, it provides a speed up of the order of hundreds of thousand times on medium or large data sets. We will also show that compared to GC, the average BC solution, and a SP clustering algorithm, our centroids based approach performs better than or as good as the label vector based algorithm MCLA. We will quantitatively evaluate the difference in quality (DQ) in each case. Filtering (Section 4) was used to produce all results with BM and MM unless otherwise stated.

6. Data sets used

While this clustering approach is designed for data sets that are too large to fit in main memory, for comparison purposes we show results on tractable sized data sets for which we can compare against clustering all of the data. Some data sets were chosen to allow for comparison with published results. Table 1 describes the six data sets.

The Iris data set [46] has been heavily studied and has one class linearly separable from the other two. The ISOLET6 data set is a subset of the ISOLET spoken letter recognition training set and has been prepared in the same way as in Ref. [8] with six classes out of 26 randomly selected. Pen digits is the pen-based recognition of handwritten digits from the UC Irivne repository [46].

Table 1
Data sets

Name	Number of examples	Number of features	Number of classes
Iris	150	4	3
ISOLET6	1440	617	6
Pen Digits	3498	16	10
Plankton	419,358	26	12
KDD98	95,412	56	10
MRI-1	1,132,545	3	9
MRI-2	3,991,592	3	9

The plankton data consist of 419,358 samples of plankton from the underwater SIPPER camera which records eight gray levels. The samples were taken from the 12 most commonly encountered classes of plankton during the acquisition in the Gulf of Mexico. The class sizes range from about 11,337 to 74,053 examples. The KDD98 data set is the 1998 KDD contest data set [47]. This data set is about people who gave charitable donations in response to direct mailing requests and was processed as done in Ref. [23] (http://www-cse.ucsd.edu/users/elkan/skm.html).

The large MRI-1 data set was created by concatenating 45 slices of MR images of a human brain each of size 256×256 from modalities T1, PD, and T2. The magnetic field strength was 3 T. The largest data set, MRI-2, was created by concatenating 48 slices of MR images of a human brain each of size 512×512 from modalities T1, PD, and T2. The magnetic field strength was 1.5 T. Air was removed from both MRI-1 and 2 using a threshold.

The values of m used for fuzzy clustering were $m = 2$ for Iris, Plankton, MR1-1, MRI-2; $m = 1.2$ for ISOLET6, KDD98, and $m = 1.5$ for the Pen Digits data set. The different values enabled repeatable partitions with the full data to be obtained.

Experiments on the three small data sets, Iris, Pen Digits, and Isolet6 were run on an UltraSPARC-III+ with one 900 MHz processor with 1024 MB memory. As memory requirements with the other four medium or large data sets were greater, they were run on an UltraSPARC-III with eight processors each of 750 MHz with 32 GB of shared main memory. None of the programs were multi-threaded.

7. Results for ensembles created from a balanced distribution

In this section, we evaluate performance of algorithms on an ensemble created by randomly dividing the data into disjoint subsets. On the small data sets, Iris, ISOLET6, and Pen Digits, two types of experiments have been conducted. For the Iris data set, we used $r \in \{3, 5\}$ i.e. divided randomly into 3 and 5 disjoint equal size subsets. In the first experiment, an ensemble of base partitions from three subsets, will be merged, and in the second experiment base partitions from five subsets will be merged. Similarly, for ISOLET6 and Pen Digits it was $r = 5$ and 10. As the clustering time is larger for the other four large/medium large data sets, we split them in only one way. Plankton and KDD98 were divided into 15 disjoint equal size subsets ($r = 15$), while MRI-1 and MRI-2 were divided into 20 disjoint equal size subsets ($r = 20$). We chose these values because we believe a 10% ($r = 10$) to 20% ($r = 5$) sample size to be reasonable for small data sets and 5% ($r = 20$) for medium or large data sets. In each experiment, all the BC/partitions in an ensemble were formed either using hard-k-means or fuzzy-k-means. When all BC in an ensemble were formed by hard-k-means, we will denote that experiment as a hard-ensemble experiment, and when the ensemble was formed by fuzzy-k-means, we will call it a fuzzy-ensemble experiment. All experimental results were the average over 50 random initializations.

The evaluation of our experiments was done by the DQ as shown

$$DQ(X, Y) = \frac{(X_{SE} - Y_{SE})}{Y_{SE}} * 100, \tag{5}$$

where X_{SE} and Y_{SE} are the squared error for algorithm X and Y, respectively. For example, DQ(BM,MCLA), means we are comparing the SE of the BM (BM_{SE}) and MCLA ($MCLA_{SE}$) algorithms, respectively.

7.1. Hard-ensemble experiments

Table 2 shows the DQ of all approaches when compared to clustering with all the data (GC). We also show how the average BC in the ensemble (BC) compares to GC. The first column of the table is a data set name followed by the number of disjoint subsets into which it was divided. For example, Iris 3S means that a row contains results of the Iris data divided into three subsets.

For the SP experiment, each time the number of examples loaded was equal to the number of examples present in a subset. For example, for Iris 3S, SP will load exactly $\frac{1}{3}$ of the examples into memory at a time.

A bold negative value in Table 2, which shows the DQ values, indicates that the squared error criterion was less than the value achieved by clustering all the data for a particular algorithm. We can see that the single pass algorithm was generally not as good as the others. However, its average percentage difference from GC was only 1.79% making it comparable. The average quality of the base clustering solution in the ensemble is not as good as was achieved after combination with BM or MM. Hence, the combination approaches show utility. It is also the case that the final partitions produced after bipartite merging and Metis merging were comparable to that produced by MCLA. On average, all three combination methods were slightly better than GC with MM slightly better than BM.

It should be noted that getting better results, compared to GC, for a multiple partition combining algorithm may depend on the ensemble size and how the ensemble was created. In Ref. [8], the ensemble size was much bigger than we used. Our purpose is to show our algorithms using a low resolution representation of an ensemble produced quality which approximates that produced by a high resolution ensemble representation algorithm, MCLA.

In Table 3 the average speed up of our centroids based ensemble algorithms, BM and MM, compared to the label vector based algorithm, MCLA, is shown. We excluded the very small Iris data set. The results in Table 3 show that the speed-up from BM and MM is thousands of times on medium or large data sets. On the MRI-2 data set, which contains about 4 million examples, BM was more than six hundred thousand times faster than MCLA, while MM was over two hundred thousand times faster. It seems from the results that BM and MM have almost constant time complexity, that is, independent of data set size, while with MCLA it grows with data size. Theoretical time complexity analysis will be given later. It should be noted that for MRI-2, the largest data set, MCLA on average took about 3.84 h, while BM and MM took only 0.02 and 0.06 s, respectively. Looking at the contrast, it seems MCLA and other similar label vector based multiple partitioning algorithms, having similar time complexity, might not be scalable for large or very large data sets.

7.2. Fuzzy-ensemble experiments

The base partitions of the ensemble were generated using the fuzzy-k-means algorithm. We conducted comparison experiments similar to those with the hard ensembles. The DQ of the fuzzy results is evaluated using the same formula as used in the hard-ensemble experiments (5). We do not compare with SP because it is a crisp variant of the hard-k-means algorithm.

Table 2
Difference in quality of BM, MM, MCLA, BC and SP compared to GC

	DQ(BM,GC) (%)	DQ(MM,GC) (%)	DQ(MCLA,GC) (%)	DQ(BC,GC) (%)	DQ(SP,GC) (%)
Iris, 3S	**−13.018628**	**−10.171465**	**−11.049958**	**−6.7334**	5.4191
Iris, 5S	**−10.944115**	**−12.828112**	**−12.055461**	**−0.49746**	3.88442
Isolet6, 5S	0.240304	0.449673	**−0.083840**	0.679067	0.688666
Isolet6, 10S	1.536500	**−0.136227**	**−0.374712**	1.733862	1.194516
Pen Digits, 5S	3.898204	2.584418	3.535625	2.412399	3.003857
Pen Digits, 10S	5.497644	0.195674	1.814700	3.547871	3.524519
KDD, 15S	2.037422	3.022958	1.472665	0.404138	0.278379
Plankton, 15S	1.807285	5.134778	2.576170	0.316429	**−0.385283**
MRI-1, 20S	0.167477	**−0.210917**	**−0.196870**	0.271010	0.298877
MRI-2, 20S	1.240812	1.579382	2.168377	**−0.035630**	0.012238
Average	**−0.753710**	**−1.037984**	**−1.219330**	0.206835	1.791933

All values expressed in percentage.

Table 3
Time computed in seconds

	MCLA	BM	MM	SU-BM	SU-MM
Isolet6, 5S	0.56	0.0062	0.0562	89.87	9.91
Isolet6, 10S	1.32	0.0148	0.1244	89.40	10.63
Pen Digits, 5S	0.68	0.0052	0.0138	130.46	49.15
Pen Digits, 10S	2.87	0.0056	0.0298	511.85	96.18
KDD, 15S	223.63	0.0410	0.0798	5454.39	2802.38
Plankton, 15S	1424.38	0.0302	0.0760	47,164.90	18,741.84
MRI-1, 20S	3745.35	0.0214	0.0626	175,016.35	59,829.87
MRI-2, 20S	13,840.25	0.0222	0.0634	623,434.68	218,300.47

Speed up of BM and MM compared to MCLA. SU-BM, SU-MM mean the speed up using bipartite merger and Metis merger, respectively.

Table 4
Difference in quality of BM, MM, MCLA, and BC compared to GC

	DQ(BM,GC) (%)	DQ(MM,GC) (%)	DQ(MCLA,GC) (%)	DQ(BC,GC) (%)
Iris, 3S	1.73	1.97	1.78	3.72
Iris, 5S	0.42	0.66	0.34	9.22
Isolet6, 5S	0.69	1.26	0.28	0.54
Isolet6, 10S	1.20	1.19	0.98	0.99
Pen Digits, 5S	1.62	0.61	1.47	0.98
Pen Digits, 10S	3.01	**−0.50**	**−0.46**	1.76
KDD, 15S	1.30	1.45	1.23	**−0.01**
Plankton, 15S	1.24	**−1.45**	1.03	**−0.25**
MRI-1, 20S	1.77	1.16	1.82	0.29
MRI-2, 20S	0.055	0.055	0.058	0.06
Average	1.30	0.64	0.85	1.73

All values expressed in percentage.

As MCLA can only merge partitions in the form of label vectors, a crisp partition of the fuzzy centroids in the ensemble was obtained after assigning an example to the nearest centroid. For BM and MM, the fuzzy centroids in the ensemble were merged to obtain k global centroids. A crisp partition was then obtained by assigning an example to the nearest centroid. The quality of MCLA, BM, and MM was then evaluated using the SE metric.

Table 4 shows the quality difference in percentage of BM, MM, MCLA and BC compared to GC. In the case that one of the algorithms had a lower SE value than GC the number is negative in bold. MM was better than MCLA on average. It was quite a bit better on the plankton data set. Again, the average BC was the approach most different from clustering all the data. In the fuzzy case, all of the combination approaches resulted in slightly larger SE criterion values than clustering all the data at once. However, they were all less than 1.3% different and hence were comparable final partitions.

We also empirically evaluated the average speed up of BM and MM when compared to MCLA. Similar to the hard-ensemble experiments, the results in Table 5 show that speed-up for BM and MM was thousands of times.

7.2.1. Experimental summary

In summary, we compared the three multiple partition combining algorithms, BC and SP (for hard-ensemble experiments only) with GC. On average for all experiments BM, MM and MCLA were better than BC and SP. For the hard-ensemble experiments MCLA was slightly better than MM, and BM. For the fuzzy-ensemble experiments MM was slightly better than MCLA and BM. For both hard-ensemble and fuzzy-ensemble experiments, in all cases, our centroids based algorithms (MM and BM) produced similar quality to that of MCLA (a label vector based algorithm); however, our algorithms were hundreds of thousands times faster on medium/large data sets.

Looking at the time taken and speed up from using the centroids based algorithms, BM and MM, it seems that for very large or extreme data sets label vector based approaches may not be scale. It is true that in many cases BC in the form of a centroid vector may not exist (nominal features); however, there are many domains for which all features are numeric. For example, besides many generic data sets, image processing and many other multispectral imaging domains, including magnetic resonance and satellite imaging almost always have only numeric features. We have shown that our centroid based ensemble merging algorithms are better than or as good as a label vector based ensemble merging algorithm, MCLA, while being scalable. A centroid representation is likely to be a low resolution representation as the number of clusters is typically much less than number of examples in large or very large data sets. So, in all cases, for ensembles created from balanced distributions, the performance of our algorithms approximates MCLAs.

In some applications, a final partition in the form of label vectors may need to be created from the centroids, which would require another linear scan through the data on the disk. We empirically computed the time taken to create label vectors on the largest two data sets, MRI-1 and MRI-2. On average over 50 random experiments MRI-1 and MRI-2 took 2.3032 and 8.1654 s, respectively. If we add this time to the time taken by the BM and MM in Table 3, the speed up compared to MCLA is still more than a thousand times, that is, 1611.18 and 1583.12 for BM and MM, respectively, on the MRI-1 data set and 1690.39 and 1681.92 for BM and MM,

respectively, on the MRI-2 data set. Similar results would be obtained if we add this label vector creation time to the BM and MM times in Table 5. In summary, if one needs to create a final clustering solution in the label vector form, BM and MM still result in very large speed ups.

8. Performance under unbalanced distributions and the effect of filtering

In the previous section, data were randomly divided into equal size subsets i.e. each subset contains a uniform/slowly changing distribution from all classes. Skewed distributions may also exist in physically distributed data, and there may be no chance of homogenizing the distributed sites due to privacy preserving issues [5,34,35]. Moreover, in some cases one or more clustering solutions in the ensemble may be noisy. We created a type of unbalanced distribution using the three small data sets Iris, Pen Digits, and Isolet as shown in Tables 6, 7, and 8, respectively. We chose these data sets because being small they were easy to analyze plus they have true labels, which can be used to create known unbalanced distributions. All three data sets were divided into approximately equal sized subsets; $r = 5$ for Iris, $r = 10$ for Pen Digits, and $r = 10$ for Isolet6. For all the data sets, 80% of the subsets, four subsets for Iris, eight for Pen Digits and isolet6, have examples from at least one class missing, to create an unbalanced distribution of examples. We arbitrarily created the unbalanced distribution for each data set. The purpose is to provide an understanding of the performance of the algorithms under an unbalanced distribution. Figs. 5, 6, and 7 show the average results from 50 random experiments on the Iris, Pen Digits, and Isolet6 data sets, respectively. Experimental results on ensembles formed from both hard-k-means and fuzzy-k-means are provided. In the figures, BM″ means the ensemble was merged using the bipartite merger with filtering options turned off, MM″ means Metis merger with filtering option turned off, while BM and MM means the filtering option was on. MCLA has no notion of filtering, so it is unchanged. We compared the results with BC and GC. With the filtering option turned on, the quality of BM and MM improved in all experiments, justifying the concept of filtering. Because the distribution was heavily skewed, as expected the average SE value of BC is poor. In all hard-ensemble experiments, among the three multiple partition combining algorithms, MM was the best on all data sets. It managed to recover good partition quality, even better than GC on Iris, with the filtering option turned on. This is a non-trivial accomplishment because the distribution is heavily skewed, which is indicated by the poor SE value of BC. For fuzzy-ensemble experiments, MM was the best except on Isolet6, where MCLA was slightly better. BM with the filtering option turned on provided quality better than BC on all data sets; however, it performed poorly, except on Iris, compared to MM and MCLA. This might be because 80% of the subsets had an unbalanced distribution, which is not a favorable condition for grouping centroids using a one to one mapping constraint. BM optimally matches centroids between a pair of partitions by providing the best matches

Table 5
Time computed in seconds

	MCLA	BM	MM	SU-BM	SU-MM
Isolet6, 5S	0.56	0.0054	0.0566	103.92	9.91
Isolet6, 10S	1.33	0.0144	0.1270	92.01	10.43
Pen Digits, 5S	0.68	0.0056	0.01420	121.32	47.84
Pen Digits, 10S	2.87	0.0068	0.0304	421.38	94.25
KDD, 15S	223.49	0.0420	0.0762	5321.19	2932.93
Plankton, 15S	1424.83	0.0310	0.0796	45,962.25	17,899.87
MRI-1, 20S	3741.78	0.0194	0.0632	192,875.25	59,205.37
MRI-2, 20S	13,847.10	0.0212	0.0648	653,165.09	213,689.81

Speed up of BM and MM compared to MCLA. SU-BM means the speed up using Bipartite merger. Similarly for SU-MM.

Table 6
Unbalanced distribution of Iris data set into five subsets

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

S1, S2, …, S5 are the subsets.

Table 7
Unbalanced distribution of Pen Digit data set into 10 subsets

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Class1	35	35	0	35	35	70	35	5	65	48
Class2	35	35	70	35	35	0	35	65	5	49
Class3	35	35	35	0	35	35	70	35	35	49
Class4	35	35	35	35	35	35	35	35	35	21
Class5	35	35	35	70	0	35	0	70	35	49
Class6	35	35	35	35	35	35	35	35	35	20
Class7	35	35	5	35	35	65	35	35	0	56
Class8	35	35	35	35	70	35	35	0	70	14
Class9	35	35	65	65	35	5	5	35	56	0
Class10	34	34	34	4	34	34	64	34	13	51

S1, S2, …, S10 are the subsets.

Table 8
Unbalanced distribution of Isolet6 data set into 10 subsets

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Class1	24	24	0	24	4	48	24	44	48	0
Class2	24	24	24	0	44	24	48	4	24	24
Class3	24	24	48	24	24	0	24	24	0	48
Class4	24	24	24	48	24	24	0	24	24	24
Class5	24	24	4	24	48	44	24	0	24	24
Class6	24	24	44	24	0	4	24	48	24	24

S1, S2, ..., S10 are the subsets.

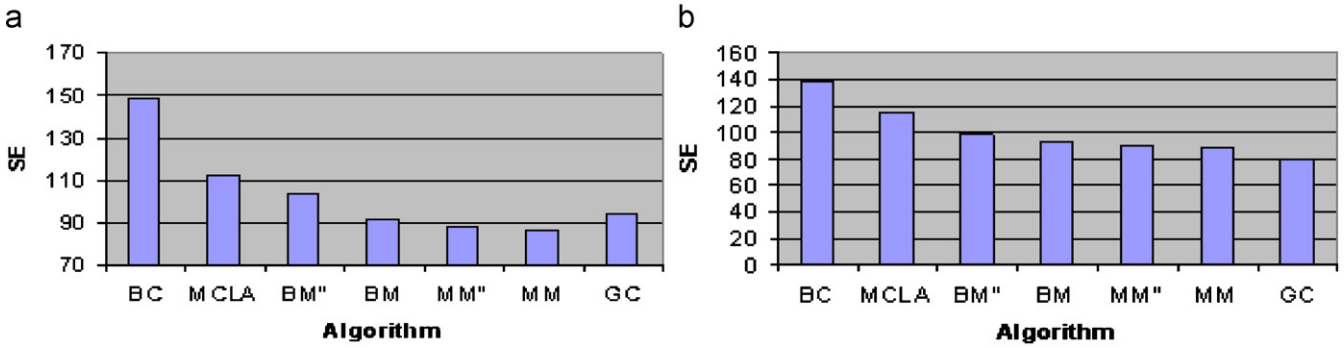

Fig. 5. Experiment on unbalanced Iris data set. Experimental results in SE both from (a) hard-k-means and (b) fuzzy-k-means are provided.

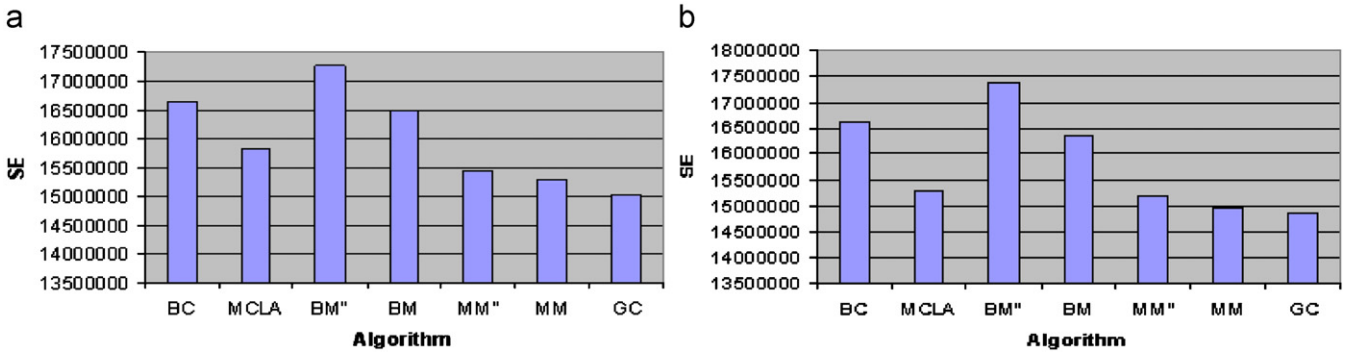

Fig. 6. Experiment on unbalanced Pen Digits data set. Experimental results in SE both from (a) hard-k-means and (b) fuzzy-k-means are provided.

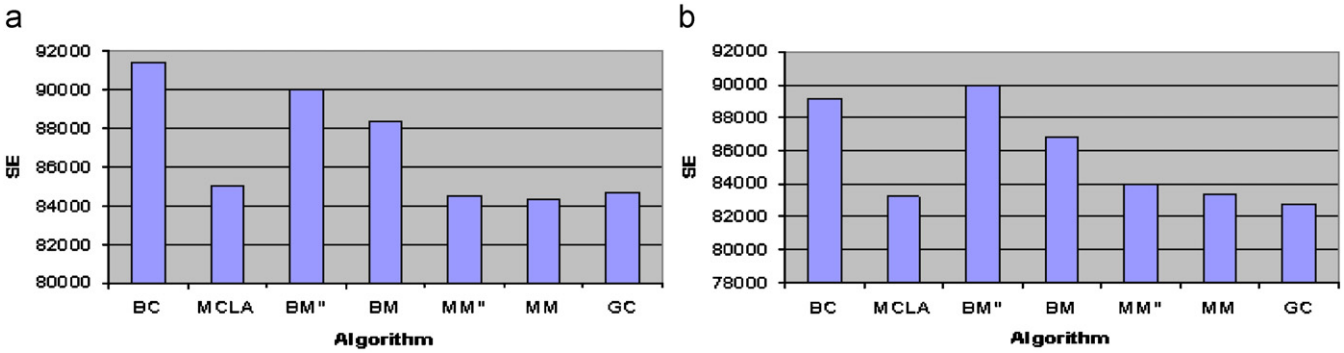

Fig. 7. Experiment on unbalanced Isolet6 data set. Experimental results in SE both from (a) hard-k-means and (b) fuzzy-k-means are provided.

locally. In many cases, this type of one to one constrained mapping might be useful, especially for mapping clusters of partitions whose distribution is slowly changing/evolving. For example, a one to one mapping constraint was used in Ref. [10] for addressing the clustering problem in perceptual organization, which was used to provide additional knowledge about a 3D object. They proposed path based clustering to group smooth curves and textured segmentations.

It should be noted that even without filtering, the quality of MM was always better (except on the fuzzy ensemble of isolet6) than MCLA.

9. Analysis of time and space complexity

We analyzed the time complexity of the multiple partition combining algorithms. Let n be the number of examples in the whole data set, k the number of clusters, and r the number of BC/partitions in the ensemble. Then, the time complexity of MCLA is $O(nk^2r^2)$ [5]. For BM and MM, centroids were used to represent the ensemble, thus the time complexity of our algorithms is free from n. The time complexity for BM is $O(rk^2f + rk^3 + kr^2)$, where f is the number of features in the data. The first term describing the time complexity of BM is the time required to compute the distances between cluster centers in adjacent partitions for an arbitrarily chosen ordering. The second term is the time required to apply the Hungarian algorithm for the bipartite matching to each of the $r-1$ pairs. The last term is the time required to build a spanning tree (using an adjacency matrix) for filtering (or just retrieval of the consensus chain) after bipartite matching has been done as there are r vertices. For filtering one must do depth first search which is also $O(|V|^2)$ or $O(|kr^2|)$ for k consensus chains.

The time complexity for MM is $O(r^2k^2f + kr^2)$. The first-term describing the time complexity of MM is the time required to obtain the distances between cluster centers and convert them to weights. The second term is the time required to create the partition using METIS which is bounded by the number of edges in the r-partite graph. It also is the time to do all of the filtering as noted above (which only changes the constant).

The time complexity of other related label vector based multiple partition algorithms is also of interest. For SCEC it is $O(tnr^2k)$, where t is the number of iterations required in SCEC and CSPA is $O(n^2kr)$ [5]. It has been noted in Ref. [8] that the QMI and MMEC have the same time complexity as SCEC. It seems that CSPA will be slower than MCLA as it depends on n^2. SCEC, hence QMI and MMEC, will be faster than MCLA if t, the number of iterations required in their algorithm, is less than k, which is the number of clusters. So, all have n in their time complexity equation, except BM and MM. However, it should be noted that if one needs to create a final solution in the form of label vector, a linear scan through the data is required. We have shown that our algorithms still produce significant speed up when the linear scan through the data is required. The time complexity for creating a label vector for the full data set is $O(nkf)$.

We also analyzed the space complexity of the representation of the two types of ensemble, that is, centroid based and label vector based. Assuming hyperedges are loaded in memory, the space complexity of the label vectors based algorithm, MCLA, is $O(knr)$. For BM and MM it is free from n, that is, $O(kfr)$, assuming all centroids are loaded in memory for merging. We did an empirical evaluation using our largest size data set, MRI-2. It contains 3,991,592 examples, three features, nine clusters, and was divided into 20 subsets, that is, ensemble size $r=20$. For representing hyperedges, an indicator vector, in MCLA, we used char type memory allocation (1 byte). In MM and BM, floating point precision (4 bytes) was used to store cluster centroids. For MCLA, $9*20=180$ hyperedges were kept in memory for efficient construction of the meta graph. So, the total memory requirement to store the hyperedges are $180*3,991,592*1=718,486,560$ bytes (approximately 685 MB). To keep the ensemble of centroids in memory, we need $9*3*20*4$ bytes, only 2160 bytes. Thus the memory requirement of the label vector based algorithm, MCLA, is more than three hundred thousand times larger, that is, 332,632.66. In a wide area network scenario, which might occur for merging clustering solutions of physically distributed data, transferring centroids would also be efficient in terms of network bandwidth cost. Thus, the centroid based ensemble algorithms not only provided a speed-up of hundreds of thousands of times but also required hundreds of thousands of times less memory compared to a label vector based algorithm, MCLA. As discussed earlier, if one needs to create a label vector from the final centroids created by BM and MM, it can be done by loading data incrementally based on available buffer size.

Compared to label vector based ensemble merger algorithms, our algorithm scales extremely well, in terms of time and space complexity, as it does not depend upon the size of the data, and for most cases the number of centroids is not likely to be large. Even for very large dimensional data sets if k remains relatively small our algorithm will be fast, provided f is not as large as the size of data.

Table 9
Sensitivity experiments of BM, MM, and MCLA on hard ensembles

	Iris, 5S (%)	Isolet6, 10S (%)	Pen Digits, 10S (%)	Average (%)
BM	1.304684	0.308255	0.453086	0.688675
MM	0.361843	0.145923	0.210061	0.239275
MCLA	0.096984	0.148035	0.369641	0.204886

All values expressed in percentage change of SE.

Table 10
Sensitivity experiments of BM, MM, and MCLA on fuzzy ensembles

	Iris, 5S (%)	Isolet6, 10S (%)	Pen Digits, 10S (%)	Average (%)
BM	1.139955	0.181290	0.315357	0.545534
MM	0.106712	0.178516	0.174228	0.153152
MCLA	0.040850	0.176785	0.254943	0.157526

All values expressed in percentage change of SE.

10. Sensitivity estimation

Because the multiple partition combining algorithms are heuristic solutions to the r-partite graph partitioning problem, they may be sensitive to the order of selecting partitions in the ensemble. In Ref. [5], it was not stated that MCLA could be sensitive to the order in which the hyperedges, actually the vertices, are numbered in the meta graph. Order matters because the METIS graph partitioning package is a randomized algorithm. In this section some results from experiments designed to estimate sensitivity of the multiple partition combining algorithms are reported.

Each experiment with multiple combining algorithms consisted of 50 random initializations, 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 to estimate how sensitive the algorithms were to the random selection of order. Sensitivity was 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, we normalized by dividing by the mean and then multiplying by 100 to obtain a percentage. This will indicate how much the quality of partitions could vary from their mean quality on average. Sensitivity is computed as:

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

Because each of the 32 experiments involved 50 random initializations, we used the three smaller data sets Iris, Isolet6, and Pen Digits for the sensitivity estimation experiments. The value of r, the ensemble size, chosen was maximum for each of the data sets, that is, five for Iris, 10 for Isolet6, and 10 for Pen Digits. Tables 9 and 10 show the results expressed in percentage for the multiple partition combining algorithms on each data set for hard and fuzzy ensembles, respectively. The average sensitivity of the algorithms over all data sets is also given. On average, for both hard- and fuzzy-ensemble experiments, BM was the most sensitive compared to MM and MCLA. In the hard-ensemble experiments, MCLA was the least sensitive,

while on the fuzzy-ensemble experiments MM was the least sensitive. Except on Iris, on which BM was much more sensitive compared to MM and MCLA, all three algorithms have small differences among them. On average the sensitivity of all the three algorithms was low, that is, for MM and MCLA much less than 0.5%, while for BM slightly above 0.5%. Because on average BMs sensitivity was low and also comparable to MCLA and MM, the order in which the partitions were matched in BM was not of significant importance.

11. Conclusions

In this paper we proposed methods for merging an ensemble of clustering solutions in a scalable framework in terms of time and space complexity. We evaluated our algorithms both under balanced and unbalanced distributions. Under a balanced distribution, the centroids based ensemble merger algorithms, bipartite merger (BM) and Metis merger (MM), were shown to result in partitions comparable to a label vector based ensemble merger algorithm, MCLA when compared against clustering all the data at once, global clustering (GC). BM, MM, and MCLA, were also compared against GC with BC (base clustering), and SP (a single pass clustering algorithm for hard ensembles only). On average the performance of BM and MM was better than BC, and SP. Quantitative evaluation on average indicates MM is slightly better than BM.

The centroid based ensemble merging algorithms provided partitions of quality comparable to the label vector based ensemble merging algorithm, *MCLA*, *while providing very large speed ups*. The memory requirements of our algorithms were also hundreds of thousand times less than MCLA. The speed of our algorithms coupled with small memory requirements will allow them to scale to extremely large data sets. In fact, they can be successfully applied to data sets which are distributed by necessity since they are too large to fit in any one main memory.

Under unbalanced distributions, our algorithms are also capable of detecting and removing spurious clusters from the ensemble to provide a robust framework, and MM outperformed MCLA in almost all cases. In this work, our focus was to show that using the same BC solutions a centroid based ensemble was competitive or better than a label vector based ensemble. It is not always necessary to create an ensemble from a disjoint data set, but we did it to restrict the ensemble size as a way of scaling the clustering process for the full data set. Future experiments could be done to create an ensemble in a different way, that is, from the full data set or overlapped examples to see how they affect quality compared to GC, average BC solutions, and SP clustering algorithm. Future work could also include partitioning the ensembles using different graph partitioning packages or methods, where grouping may be controlled more effectively. However, the purpose in this work was to show, using identical subsets whose clustering results were represented by low resolution (centroids) and by high resolution (label vector) approaches, that comparable final partitions can be obtained with much lower time and space complexity for the centroids based approaches.

Acknowledgements

This research was partially supported by the National Institutes of Health under grant number 1 R01 EB00822-01 and by the Department of Energy through the ASCI PPPE Data Discovery Program, Contract number: DE-AC04-76DO00789. The code for bipartite matching (Hungarian method), depth first search, and minimally weighted spanning were obtained from http://reptar.uta.edu/.

References

[1] S. Eschrich, J. Ke, L.O. Hall, D.B. Goldgof, Fast accurate fuzzy clustering through data reduction, IEEE Trans. Fuzzy Syst. 11 (2) (2003) 262–270.
[2] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, USA, 1988.
[3] P. Hore, L. Hall, Scalable clustering: a distributed approach, in: FUZZ-IEEE 2004, pp. 143–148.
[4] P. Hore, L. Hall, D. Goldgof, A cluster ensemble framework for large data sets, in: IEEE International Conference on Systems, Man, and Cybernetics, 2006.
[5] A. Strehl, J. Ghosh, Clusters ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2002) 583–617.
[6] A. Topchy, A.K. Jain, W. Punch, A mixture model for clustering ensembles, in: Proceedings of the SIAM International Conference on Data Mining, SDM, 2004, pp. 379–390.
[7] A.P. Topchy, M.H.C. Law, A.K. Jain, A.L. Fred, Analysis of consensus partition in cluster ensemble, in: ICDM, 2004, pp. 225–232.
[8] B. Long, Z.M. Zhang, P.S. Yu, Combining multiple clusterings by soft correspondence, in: ICDM, 2005, pp. 282–289.
[9] X.Z. Fern, C.E. Brodley, Solving cluster ensemble problem by bipartite graph partitioning, in: ICML, 2004.
[10] B. Fischer, J.H. Buhmann, Path-based clustering for grouping of smooth curves and texture segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 25 (4) (2003) 513–518.
[11] S. Dudoit, J. Fridlyand, Bagging to improve the accuracy of a clustering procedure, Bioinformatics 19 (9) (2003) 1090–1099.
[12] A. Topchy, A.K. Jain, W. Punch, Combining multiple weak clusterings, in: Proceedings of the IEEE International Conference on Data Mining, 2003, pp. 331–338.
[13] A.L.N. Fred, Finding consistent clusters in data partitions, in: F. Roli, J. Kittler (Eds.), International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, vol. 2364, 2002, pp. 309–318.
[14] A. Topchy, A.K. Jain, W. Punch, Clustering ensembles: models of consensus and weak partitions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (12) (2005) 1866–1881.
[15] M. Al-Razgan, C. Domeniconi, Weighted clustering ensembles, in: SDM, 2006.
[16] T. Lange, J.M. Buhmann, Combining partitions by probabilistic label aggregation, in: Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005, pp. 147–156.
[17] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, in: ICDE, 2005, pp. 341–352.
[18] A.L.N. Fred, A.K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (6) (2005) 835–850.
[19] B. Minaei-Bidgoli, A. Topchy, W.F. Punch, Ensembles of partitions via data resampling, in: ITCC, 2004, pp. 188–192.
[20] P. Viswanath, K. Jayasurya, A fast and efficient ensemble clustering method, in: ICPR, 2006, pp. 720–723.
[21] H. Spath, Cluster Analysis Algorithms for Data Reduction and Classification, Ellis Horwood, Chichester, UK, 1980.
[22] I. Davidson, A. Satyanarayana, Speeding up KMeans clustering using bootstrap averaging, in: Proceedings of the IEEE ICDM 2003 Workshop on Clustering Large Data Sets, 2003, pp. 16–25.
[23] F. Farnstrom, J. Lewis, C. Elkan, Scalability of clustering algorithms revisited, SIGKDD Explorations 2 (1) (2000) 51–57.
[24] V. Ganti, J. Gehrke, R. Ramakrishnan, Mining very large databases, Computer (1999) 38–45.
[25] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, New York, 1996, pp. 103–114.
[26] V. Ganti, R. Ramakrishnan, J. Gehrke, A.L. Powell, J.C. French, Clustering large datasets in arbitrary metric spaces, in: Proceedings of the 15th International Conference on Data Engineering, IEEE CS Press, Los Alamitos, CA, 1999, pp. 502–511.
[27] P. Bradley, U. Fayyad, C. Reina, Scaling clustering algorithms to large databases, in: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 1998, pp. 9–15.
[28] P. Domingos, G. Hulten, A general method for scaling up machine learning algorithms and its application to clustering, in: Proceedings of the 18th International Conference on Machine Learning, 2001, pp. 106–113.
[29] N.R. Pal, J.C. Bezdek, Complexity reduction for large image processing, Trans. Syst. Man Cybernet. Part B (2002) 598–611.
[30] T.W. Cheng, D.B. Goldgof, L.O. Hall, Fast fuzzy clustering, Fuzzy Sets Syst. (1998) 49–56.
[31] P. Mitra, C.A. Murthy, S.K. Pal, Density-based multiscale data condensation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (6) (2002) 734–747.
[32] J. Liu, J.P.Y. Lee, L. Li, Z.-Q. Luo, K.M. Wong, Online clustering algorithms for radar emitter classification, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1185–1196.
[33] I.S. Dhillon, D.S. Modha, A data-clustering algorithm on distributed memory multiprocessors, in: Proceedings of the Large-scale Parallel KDD Systems Workshop, ACM SIGKDD, 1999, pp. 245–260.
[34] J. Ghosh, S. Merugu, Distributed clustering with limited knowledge sharing, in: Proceedings of the 5th International Conference on Advances in Pattern Recognition, 2003, pp. 48–53.
[35] J. Ghosh, A. Strehl, S. Merugu, A consensus framework for integrating distributed clusterings under limited knowledge sharing, in: Proceedings of the National Science Foundation (NSF) Workshop on Next Generation Data Mining, 2002, pp. 99–108.
[36] E. Januzaj, H.-P. Kriegel, M. Pfeifle, Towards effective and efficient distributed clustering, in: Proceedings of the International Workshop on Clustering Large Data Sets, 3rd IEEE International Conference on Data Mining, 2003, pp. 49–58.

[37] M. Klusch, S. Lodi, G. Moro, Distributed clustering based on sampling local density estimates, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, 2003, pp. 485–490.
[38] H. Kriegel, P. Kroger, A. Pryakhin, M. Scubert, Effective and efficient distributed model-based clustering, in: ICDM, 2005, pp. 258–265.
[39] H. Jin, M.-L. Wong, K.-S. Leung, Scalable model-based clustering for large databases based on data summarization, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (2005) 1710–1719.
[40] R. Nock, F. Nielsen, On weighting clustering, IEEE Trans. Pattern Anal. Mach. Intell. 28 (8) (2006) 1223–1235.
[41] G. Karypis, V. Kumar, Multilevel algorithms for multi-constraint graph partitioning, Technical Report TR 98-019, Department of Computer Science, University of Minnesota, 1998.
[42] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, J. Parallel Distributed Comput. 48 (1) (1998) 96–129.
[43] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, in: International Conference on Parallel Processing, 1995, pp. 113–122.
[44] H.W. Kuhn, The Hungarian method for the assignment problem, Naval. Res. Logist. Q. 2 (1955) 83–97.
[45] R. Duda, P. Hart, D. Stork, Pattern Classification.
[46] C.J. Merz, P.M. Murphy, UCI Repository of Machine Learning Databases, Department of CIS, University of CA, Irvine, CA ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[47] KDD Cup 1998 Data ⟨http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html⟩.

About the Author—PRODIP HORE has received the M.S. degree in Computer Science from the University of South Florida in 2004 and the Ph.D. in Computer Science and Engineering in 2007. He is currently employed by Fair Isacc.

About the Author—DMITRY GOLDGOF has received the M.S. degree in Computer Engineering from the Rensselaer Polytechnic Institute in 1985 and the Ph.D. degree in Electrical Engineering from the University of Illinois at Urbana-Champaign in 1989.
He is currently Professor in the Department of Computer Science and Engineering and a member of H. Lee Moffitt Cancer Center and Research Institute where during 2002–2003 he held a position of Professor in Bioinformatics and Cancer Control.
Previously, Dr. Goldgof held visiting positions at the Department of Computer Science at the University of California at Santa Barbara and at the Department of Computer Science at University of Bern in Switzerland. Dr. Goldgof's research interests include motion and deformation analysis, image analysis and its biomedical applications, bioinformatics, and pattern recognition.

About the Author—LAWRENCE HALL is a Professor of Computer Science and Engineering at University of South Florida. He received his Ph.D. in Computer Science from the Florida State University in 1986 and a B.S. in Applied Mathematics from the Florida Institute of Technology in 1980. His research interests lie in distributed machine learning, data mining, pattern recognition and integrating AI into image processing.