Distributed Clustering for Scaling Classic Algorithms

Prodip Hore
University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation
Distributed Clustering for Scaling Classic Algorithms

by

Prodip Hore

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Dr. Lawrence Hall, Ph.D.
Dr. Rafael Perez, Ph.D.
Dr. Dmitry Goldgof, Ph.D.

Date of Approval
July 1, 2004

Keywords: Ensemble, Merging, Filtering, Disputed Examples, Extrema

© Copyright, 2004, Prodip Hore
Acknowledgements

I am grateful to my major professor, Dr. Lawrence Hall, for his continuous support and help. I also appreciate the help of the other members of my committee: Dr. Rafael Perez and Dr. Dmitry Goldgof.

I am also indebted to my parents, Suvash Chandra Hore and Rekha Rani Hore, and other members of my family for their support and inspiration.
## Table of Contents

List of Tables .......................................................................................................................... ii
List of Figures .......................................................................................................................... iii
Abstract ..................................................................................................................................... v

Chapter 1  Introduction and Related Work ......................................................................... 1

Chapter 2  Background .......................................................................................................... 4

Chapter 3  Partition Merging ............................................................................................. 10
  3.1  Distance Matrix ............................................................................................................. 12
  3.2  Local Chain Matrix ....................................................................................................... 12
  3.3  Global Chain Matrix ..................................................................................................... 12
  3.4  Distributed Combining Algorithm ............................................................................... 15
  3.5  Computing Disputed Examples .................................................................................. 16

Chapter 4  Centroid Filtering ............................................................................................... 19

Chapter 5  Data and Experiments ....................................................................................... 20
  5.1  Iris Plant Database ........................................................................................................ 20
  5.2  MRI Database ............................................................................................................... 20
  5.3  Synthetic Data .............................................................................................................. 21
  5.4  Plankton Data .............................................................................................................. 21
  5.5  Initialization of Centroids ............................................................................................ 21
  5.6  Experimental Setup ..................................................................................................... 21
  5.7  Iris Experiments .......................................................................................................... 22
  5.8  MRI Experiments ........................................................................................................ 27
  5.9  Using Harmony to Filter Centroids in the MRI Data Set ........................................... 33
  5.10 Synthetic Data Experiments ....................................................................................... 36
  5.11 Plankton Data Experiments ....................................................................................... 42

Chapter 6  Discussion and Conclusions .............................................................................. 43

References ............................................................................................................................... 46
List of Tables

Table 5.1 Results of Clustering (Fuzzy) Iris Global Data Set and 50 Random Initializations of Centroids .............................................22
Table 5.2 Results of Clustering (Hard) Iris Global Data Set and 50 Random Initializations of Centroids .............................................23
Table 5.3 Results of our D-Combining Algorithm (Iris Data). Fuzzy-k-means Applied to Each Subset .............................................23
Table 5.4 Results of our D-Combining Algorithm (Iris Data). Hard-k-means Applied to Each Subset .............................................23
Table 5.5 Results of Clustering (Fuzzy) MRI Global Data Set and 50 Random Initializations of Centroids .............................................28
Table 5.6 Results of Clustering (Hard) MRI Global Data Set and 50 Random Initializations of Centroids .............................................28
Table 5.7 Results of our D-Combining Algorithm (MRI data). Fuzzy-k-means Applied to Each Subset .............................................29
Table 5.8 Results of our D-Combining Algorithm (MRI data). Hard-k-means Applied to Each Subset .............................................29
Table 5.9 Using Harmony to Filter Centroids in the MRI Data Set .........................................................................................33
Table 5.10 Results of Clustering (Fuzzy and Hard) Artificial Global Data Set and 50 Random Initializations of Centroids .............................................36
Table 5.11 Results of our D-Combining Algorithm without Filtering Cluster Centers .................................................................37
Table 5.12 Results of our D-Combining Algorithm after Filtering Cluster Centers Using Harmony ..................................................37
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Overall Procedure of Distributed Clustering</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Taxonomy of Clustering Algorithms</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of Data Set</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of Hierarchical Clustering</td>
<td>5</td>
</tr>
<tr>
<td>2.4</td>
<td>Example of Hard-Clustering</td>
<td>6</td>
</tr>
<tr>
<td>2.5</td>
<td>Example of Fuzzy Clustering</td>
<td>6</td>
</tr>
<tr>
<td>2.6</td>
<td>Hard-k-means Algorithm</td>
<td>8</td>
</tr>
<tr>
<td>2.7</td>
<td>Fuzzy-k-means Algorithm</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Example of Bipartite Matching</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of Formation of Global Chain Matrix from 3 Local Chain Matrices</td>
<td>13</td>
</tr>
<tr>
<td>3.3</td>
<td>Consensus Chain Algorithm</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Distributed (D)-Combining Algorithm</td>
<td>15</td>
</tr>
<tr>
<td>3.5</td>
<td>Algorithm for Confusion-Matrix</td>
<td>16</td>
</tr>
<tr>
<td>3.6</td>
<td>Example Label Vector of Global Partition of 10 Examples</td>
<td>17</td>
</tr>
<tr>
<td>3.7</td>
<td>Example Label Vector Formed by D-Combing Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>3.8</td>
<td>Confusion Matrix and the Matched Label Pairs</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of Perfect Harmony among Centroids in a Consensus Chain</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Synthetic 2-D Gaussian Distributed Data</td>
<td>20</td>
</tr>
<tr>
<td>5.2</td>
<td>Experiments with Iris Data</td>
<td>24</td>
</tr>
<tr>
<td>5.3</td>
<td>Result of D-Combining Algorithm on Iris Data with Semi-Random Initialization</td>
<td>25</td>
</tr>
<tr>
<td>5.4</td>
<td>Result of D-Combining Algorithm on Iris Data with Pure-Random Initialization</td>
<td>26</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of Global Fuzzy Clustering of MRI Data with D-Combining Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>5.6</td>
<td>Plotting of Disputed Examples of MRI Data (Hard-k means, k=7)</td>
<td>31</td>
</tr>
<tr>
<td>5.7</td>
<td>Plotting of Disputed Examples of MRI Data (Fuzzy-k means, k=7)</td>
<td>32</td>
</tr>
<tr>
<td>5.8</td>
<td>Effect of Applying Harmony Algorithm to Consensus Chains of MRI Data</td>
<td>34</td>
</tr>
<tr>
<td>5.9</td>
<td>Effect of Applying Harmony Algorithm to the Consensus Chains of Synthetic Data (Using Fuzzy-k-means, k= 4)</td>
<td>38</td>
</tr>
</tbody>
</table>
Figure 5.10 Effect of Applying Harmony Algorithm to the Consensus Chains of Synthetic Data (Using Hard-k-means, k= 4)  ------------------------------- 39
Figure 5.11 Extremas of Global Clustering of Synthetic Data  ------------------------------ 40
Figure 5.12 Effect of Applying Harmony Algorithm to the Extrema Patterns of Synthetic Data with Pure-random Initialization (Using Hard-k-means, k= 4)  ----------------------------------------------- 40
Figure 5.13 Effect of Applying Harmony Algorithm to the Extrema Patterns of Synthetic Data with Semi-random Initialization (Using Hard-k-means, k= 4)  ----------------------------------------------- 41
Distributed Clustering for Scaling Classic Algorithms

Prodip Hore

ABSTRACT

Clustering large data sets recently has emerged as an important area of research. The ever-increasing size of data sets and poor scalability of clustering algorithms has drawn attention to distributed clustering for partitioning large data sets. Sometimes, centrally pooling the distributed data is also expensive. There might be also constraints on data sharing between different distributed locations due to privacy, security, or proprietary nature of the data. In this work we propose an algorithm to cluster large-scale data sets without centrally pooling the data. Data at distributed sites are clustered independently i.e. without any communication among them. After partitioning the local/distributed sites we send only the centroids of each site to a central location. Thus there is very little bandwidth cost in a wide area network scenario. The distributed sites/subsets neither exchange cluster labels nor individual data features thus providing the framework for privacy preserving distributive clustering. Centroids from each local site form an ensemble of centroids at the central site. Our assumption is that data in all distributed locations are from the same underlying distribution and the set of centroids obtained by partitioning the data in each subset/distributed location gives us partial information about the position of the cluster centroids in that distribution. Now, the problem of finding a global partition 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. A global consensus on the position of cluster centroids of the global data using only the very limited statistics of the position of centroids from each local site is reached by grouping the centroids into consensus chains and computing the weighted mean of centroids in a consensus chain to represent a global cluster centroid. We compute the Euclidean distance of each example from the global set of centroids, and assign it to the centroid nearest to it. Experimental results show that quality of clusters generated by our algorithm is similar to the quality of clusters generated by clustering all the data at a time. We have shown that the disputed examples between the clusters generated by our algorithm and clustering all the data at a time lay on the border of clusters as expected. We also proposed a centroid-filtering algorithm to make partitions formed by our algorithm better.
Chapter 1 Introduction and Related Work

Unlabeled data can be grouped or partitioned into a set of clusters in many ways. There are hierarchal clustering algorithms [17], iterative clustering algorithms, single pass clustering algorithms and more [21]. Clustering data is often considered to be a slow process. This is especially true of iterative clustering algorithms such as the K-means family [25]. As larger unlabeled data sets become available, the scalability of clustering algorithms becomes important. In recent years a number of new clustering algorithms have been introduced to address the issue of scalability [8, 10, 11, 12, 13, and 14]. Various methods of accelerating k-means have also been studied [4, 24, 25, 26 and 30]. All the above algorithms assume that the clustering algorithm is applied to all the data centrally pooled in a single location.

Distributed computing has also emerged as an important area of research for scaling the clustering process and also due to some inherent problems in pooling data from distributed locations to a centralized location for extracting knowledge. Combining multiple partitions has been studied for quite a few times [1, 2, 5, 6, 7, 27, 28, 29, and 31]. In [9] a parallel version of k-means has been proposed which requires synchronized communication during each iteration [9], which might become difficult and costly in a wide area network. Moreover, there might be constraints like data could not be shared between different distributed locations due to privacy, security, or the proprietary nature of the data. Extracting knowledge from these types of distributed locations under restraints of data exchange is called privacy preserving data mining. There has been some work where data in distributed form has been clustered independently i.e. without any message passing among them and multiple partitions combined using limited knowledge sharing [6, 7, 27, 28, and 29]. Knowledge reuse framework [3] has also been explored, where label vectors of different partitions are combined without using any feature values [1 and 29]. In [6 and 27] distributed clustering has been discussed under two different strict settings that impose severe constraints on the nature of the data or knowledge shared between local data sites. In [7], 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 has been discussed in [28]. Some work on distributed data mining has also been done for association rule mining with limited knowledge sharing under the banner of privacy-preserving data mining [32, 34, and 35].
In this thesis we propose a distributed clustering framework under strict limited knowledge sharing similar to the setting in [6], where distributed sites don’t allow sharing of cluster labels and attributes of objects/examples among them. In [6] generative or probabilistic model parameters are sent to a central site, where “virtual samples” are generated and clustered using an EM algorithm [21 and 23] to obtain the global model. In our approach we cluster each distributed location/local sites using a hard-k-means or a fuzzy-k-means algorithm independently and send only the cluster centroids to a central location, where they form an ensemble of centroids. Our assumption is that data in all subsets or distributed locations are from the same underlying distribution and the set of centroids obtained by partitioning the data in each subset/distributed location gives us partial information about the position of the global cluster centroids. Now, the problem of finding a global partition 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. We reach a global consensus on the position of cluster centroids of the global data after integrating the very limited statistics of the position of centroids from each local site. Our approach also introduces an additional framework for filtering or removing inappropriate centroids from participating in the merging process, whose inclusion, otherwise, would have distorted the global partition. In the real life scenario this type of framework will be useful because one or more distributed sites might be very noisy. This framework will enable the central site to analyze the merging process rather than blindly combining all the partitions.

This approach can be applied to many existing algorithms, which might be labeled “centroidal”. That is, these algorithms iteratively produce cluster centroids, which are representative of the data assigned to each cluster. Examples of this type of clustering algorithm are the K-means clustering algorithms (hard and fuzzy), and the EM algorithm [15]. Almost any type of clustering algorithm can be made into a centroidal algorithm by simply creating cluster centroids to represent clustered data. Figure 1.1 shows the overall procedure of our algorithm.

In Chapter 2, the particular clustering algorithm used in the experiments is discussed. Chapter 3 describes how the centroids from different subsets/distributed sites are integrated to form the centroids of the global partition. In Chapter 4 a centroid-filtering algorithm has been discussed to filter out noisy centroids and Chapter 5 presents the experimental data sets and results. Chapter 6 is a summary and discussion.
Global data set

Subset 1

Apply Fuzzy-k-means or Hard-k-means

Subset 2

Apply Fuzzy-k-means or Hard-k-means

Subset m

Apply Fuzzy-k-means or Hard-k-means

Access the centroids formed by clustering partial data in the above subsets. Merge them to form the global set of centroids.

Final global partition of the data

Figure 1.1 Overall Procedure of Distributed Clustering
Chapter 2  Background

Clustering is one of the most important unsupervised learning problems. In unsupervised learning the data has no predefined label. Clustering could be viewed as the “process of organizing objects into groups whose members are similar in some way”. Objects or examples or patterns are defined by features, which describe them. Clustering algorithms group these examples into clusters such that examples in a cluster are more similar than examples in other clusters. There are many varieties of clustering algorithms but generally most of them fall under one of the two broad categories i.e. hierarchical or partitional algorithms. Below (Figure 2.1) is the taxonomy of clustering algorithms. There are also many other varieties of clustering algorithms [23] but we show the taxonomy of the most commonly used ones.

Figure 2.1  Taxonomy of Clustering Algorithms

Hierarchical clustering algorithms do not form a final partition of the data but yield a dendrogram representing the nested grouping of patterns and similarity levels at which grouping change [23].
Illustrative example:

Consider the following 5 examples E1, E2, E3, E4, and E5 in 2-dimensional space (Figure 2.2) and let’s assume that the spatial proximity among them is a measure of the similarity with each other.

![Figure 2.2 Example of Data Set](image)

Applying a hierarchical algorithm to the above data set could yield a dendrogram as shown in Figure 2.3.

![Figure 2.3 Example of Hierarchical Clustering](image)
We will not further discuss hierarchical algorithms because our work is based on scaling partitional algorithms in a distributed way. Partitional clustering algorithms produce a single partition of the data by iteratively optimizing a clustering criterion function or an objective function [23]. Hard k-means results in a crisp partition of the data while fuzzy k-means assigns a degree of membership to each example in a data set i.e. how much an example is associated with a cluster. Membership for a particular example in a cluster indicates how well it fits in that cluster. Applying hard-k-means to the data set (Figure 2.2) could yield the following two clusters (Figure 2.4).

![Figure 2.4  Example of Hard-Clustering](image)

Applying fuzzy-k-means to the data set (Figure 2.2) could yield the following two overlapping clusters (Figure 2.5). Here the example E5 belongs to both the clusters but its membership degree would be higher for cluster 2 than in cluster 1.

![Figure 2.5  Example of Fuzzy Clusters](image)
Similarity measures among examples or patterns are of fundamental importance and must be chosen carefully [23]. We have used a well-known metric i.e. Euclidean distance for measuring dissimilarity among examples whose features are all continuous values. If every example or pattern has dimension d i.e. d features, then dissimilarity among any examples \( x_i \) and \( x_j \) is measured as follows:

\[
d_{i,j}(x_i, x_j) = \left( \sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2 \right)^{\frac{1}{2}} = \| x_i - x_j \|_2
\]

………………..(1)

The widely used criterion function or objective function which hard-k-means and fuzzy-k-means try to optimize is the squared error function:

\[
\min_{(U,V)} \left\{ J_m(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^m D_{ik}(x_k, v_i) \right\}
\]

(2), where

\( U \) contains the cluster memberships with \( u_{ik} \in [0,1] \), \( m > 1 \) if fuzzy or \( u_{ik} \in \{0,1\} \) if hard; 
\( V = (v_1, v_2, ..., v_c) \in \mathbb{R}^p \), \( v_i \) specifies the \( i^{th} \) cluster center of dimension \( p \); 
\( m \geq 1 \) is a weighting exponent that controls the degree of fuzzification of \( U \); and \( D_{ik}(x_k, v_i) = D_{ik} \) is the deviation of \( x_k \) from the \( i^{th} \) cluster prototype and 

\[
D_{ik} = \| x_k - v_i \|_2^2
\]

We know describe the Hard-k-means and Fuzzy-k-means algorithms using the Euclidean distance metric in Figure 2.6 and Figure 2.7 respectively.
Input: Data set of n examples of s features and the value of k (number of clusters).

Output: Partition of the input data into k regions.

1. Declare an n x k size U membership matrix.

2. Randomly generate k cluster center locations within the range of the data or randomly select k examples as initial cluster centroids. Let the centroids be c₁, c₂,…, cₖ.

3. Calculate the distance measure $d_{i,j} = \|x_i - c_j\|_2$ according to equation (1), for all cluster centroids j=1 to k and data examples 1 to n.

4. Compute the U membership matrix as follows:

$$u_{i,j} = \begin{cases} 1; & \|x_i - v_j\|_2 \leq \|x_i - v_l\|_2 \text{, j ≠ l} \\ 0; & \text{otherwise} \end{cases} \quad \forall \ i, j; \text{ (ties are broken randomly)};$$

5. Compute new cluster centroids $c_j = \frac{\sum_{i=1}^{n} (u_{i,j})^m x_i}{\sum_{i=1}^{n} (u_{i,j})^m}$, for j=1 to k.

Note the cluster centers in hard-k-means are just the centroids of the points in a cluster.

6. Repeat steps 3 to 5 until the change in U in two successive iterations is less than a given threshold.

Figure 2.6  Hard-k-means Algorithm

8
Input: Data set of n examples, value of k (number of clusters), fuzzification value m>1 (we have used the value of 2).
Output: Partition of the input data into k regions.

1. Declare an n x k size U membership matrix.

2. Randomly generate k cluster center locations within the range of the data or randomly select k examples as initial cluster centroids. Let the centroids be c1, c2,…, ck.

3. Calculate the distance measure $d_{i,j} = \|x_i - c_j\|_2$ according to equation (1), for all cluster centroids j=1 to k and data examples 1 to n.

4. Compute the Fuzzy membership matrix as follows:

   $$u_{i,j} = \left[ \sum_{j=1}^{k} \left( \frac{d_{i,j}}{d_{i,j}^{m-1}} \right)^{\frac{2}{m-1}} \right]^{-1}$$

   if $d_{i,j} > 0$

   \[= 1 \quad \text{if} \quad d_{i,j} = 0\]

5. Compute new cluster centroids $c_j = \frac{\sum_{i=1}^{n} (u_{i,j})^m x_i}{\sum_{i=1}^{n} (u_{i,j})^m}$, for j=1 to k.

6. Repeat step 3 to 5 until the change in U in two successive iterations is less than a given threshold.

Figure 2.7  Fuzzy-k-means Algorithm

Although, we have not used EM clustering algorithm in this work, one can also apply it to cluster data. In fact any partitioning algorithm whose clusters can be represented by centroids can use our scaling method to cluster large-scale data. The EM algorithm makes a hypothesis that the pattern or examples of a data set are drawn from some distribution and proceeds with the goal to identify them by iteratively updating the hypothesis. The most commonly used assumption of the distribution is Gaussian.
Chapter 3  Partition Merging

After clustering is applied to each subset/distributed sites there are a set of centroids available which describe each partitioned subset. Our assumption is that data in all subsets or distributed locations are from the same underlying distribution and the set of centroids obtained by partitioning the data in each subset/distributed location gives us partial information about the position of the cluster centroids of the global data.

Clustering or partitioning a subset/distributed location (say i) will produce a set of centroids $\{C_{i,j}\}_{j=1}^{k}$, where k is the number of clusters. For m subsets/distributed locations we will have m sets of centroids i.e. $\{C_{1,j}\}_{j=1}^{k}, \ldots, \{C_{m,j}\}_{j=1}^{k}$ forming an ensemble of centroids at the central site. Now, the problem of findings a global partition using the limited knowledge of the ensemble of centroids can be viewed as the problem of reaching a global consensus on the position of the centroids for the global data. One way to reach a global consensus is to group the ensemble of centroids into k consensus chains, where each consensus chain will contain m (number of subsets/distributed locations) centroids $\{c_{i}^{n}, \ldots, c_{m}^{n}\}$ one from each of the partition, where n is from 1 to k. The aim is to group similar centroids in each consensus chain. The objective is to globally optimize the assignment $\psi^{*}$ out of all possible families $f$ of centroid assignments to k consensus chain:

$$\psi^{*} = \arg\min_{\psi \in f} \left\{ \sum_{n=1}^{k} \text{cost}(\text{consensus \_ chain}(n)) \right\} \quad \text{(1)}$$

$$\text{cost}(\text{consensus \_ chain}(n)) = \sum_{i=1}^{m} D(n) \quad \text{(2)}$$

where

$$D^{n}(i) = \frac{1}{2} \sum_{j=1}^{m} d(c_{i}^{n}, c_{j}^{n}) \quad \text{(3)}$$

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 (3).

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
centroids are determined from the size of the subsets/distributed location from where it has come.

In other words, if we represent the centroids of each partition 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 (2) reduces to the minimally weighted perfect m-partite matching problem, which is NP-complete for m>2. So, we need another approach.

If the consensus chains are formed after optimizing the objective function (1) i.e. m-partite minimally weighted perfect matching, then centroids in each consensus chain are assigned with minimum cost (2) having optimum consensus among centroid in a consensus for a global centroid position. But optimizing the objective function (1) is intractable. Thus we have used a heuristic algorithm to group the centroids into k consensus chains. We know that for 2 partitions we have a polynomial time algorithm i.e. minimally weighted perfect bipartite matching [33] to globally optimize the above objective function. We pick 2 partitions at random and group their centroids into k consensus chains after globally optimizing the above objective function using minimally weighted perfect bipartite matching (Figure 3.1 shows an example of it). Now, each consensus chain will contain 2 centroids (matched pairs) one from each partition. Next we pick the centroids of one of these already assigned partitions and a new partition randomly and again optimize the objective function for these two partitions and put the centroid of the new partition in the same consensus chain in which the matched centroid from the assigned partition belongs. In this way we continue grouping the centroids of partitions into the consensus chain one by one until they are exhausted. One can also use a greedy approach instead of minimally weighted bipartite matching for matching centroids of two partitions. We have observed that both the greedy approach and bipartite matching gives the same result on average. In 3.1 we have described the greedy approach and all experimental results are from using the greedy approach. But in future we plan to replace it by the Hungarian method of perfect bipartite matching [33].

Illustrative example:

![Figure 3.1 Example of Bipartite Matching. The numbers on the edges are weights or cost. The blue colored thick lines show the min-cost bipartite matching](image-url)
3.1 Distance Matrix

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

3.2 Local Chain Matrix

The Local chain matrix stores the matched centroids correspondence between any two selected subsets/distributed location i.e. which centroid in the first selected subset matched to which centroid in the second subset. If there are C clusters in each subset, the dimension of the Local chain matrix will be C X 2.

3.3 Global Chain Matrix

The Global chain matrix uses the information from local chain matrices to build the global consensus chain of centroids. Each row of the global chain matrix represents a consensus chain. If there are C clusters in each subset and there are M such subsets, the dimension of Global chain matrix will be C X M.

For a given pair of subsets, we first find the distance between all centroid vectors. This gives us a $c \times c$ distance matrix were each entry (i,j) is the distance from the $i^{\text{th}}$ centroid in one partition to the $j^{\text{th}}$ centroid in the other. The Euclidean distance is used here. The smallest distance entry provides the first pair of matched centroids. The next smallest distance will provide the second set of matching centroids, etc. until they are all paired. There is a difficulty if 2 centroids in one partition are closest to a single centroid in the other partition. In this case, the second centroid encountered will be matched with a centroid, which is not the closest. We say a collision has occurred.

So, every centroid in one partition is mapped to the closest centroid in the partition for which correspondence is being generated. In the case that the closest centroid has already been paired with another centroid, the mapping will be to the next closest centroid, which has not been previously paired. In this way, each centroid of a partition is mapped to a corresponding centroid in the other partition.

A global mapping is obtained by applying transitivity to matching pairs of partitions as described earlier. Consider four partitions, $p_1$, $p_2$, $p_3$, $p_4$, after finding a correspondence between $p_1$ and $p_2$, $p_2$ and $p_3$ and $p_3$ and $p_4$, by transitivity there is a mapping from each of the cluster centroids in $p_1$ to those in $p_4$.

Illustrative Example:

Consider the case that there are 4 subsets of data and each subset is grouped into 3 clusters. Let S1, S2, S3, and S4 be the subsets. Figure 3.2 (a) shows the two local chain matrices and Figure 3.2 (b) shows the global chain matrix. Figure 3.3 shows the
consensus chain algorithm, which merges the local chain matrices to form the global chain matrix.

(a) Arrow shows the transitive relation between 3 local chain matrices

(b) Global chain matrix formed from above three local chain matrixes. Each row is a consensus chain

Figure 3.2  Example of Formation of Global Chain Matrix from 3 Local Chain Matrices

We call each row of the Global chain matrix a consensus chain. It tells us which centroid in which subset is matched to which other centroid(s) in other subsets. If the global chain matrix is formed without any collision at any stage, we say that the centroids of subsets have mapped perfectly. The whole process is known as centroid mapping. The input to the algorithm (Figure 3.3) is only the set of centroids of subsets. The number of centroids of a data set is not generally a function of the number of examples, so the algorithm generally takes constant time. After solving the matching problem, a consensus chain contains similar types of centroids. In this work, as stated earlier we weight each centroid by the number of examples in its subset and then create a weighted average centroid. So, if we have 4 partitions and 3 clusters consider the first cluster of the first partition. Assume that the subset it was created from contains 100 examples; it matches the second cluster of the second partition which has 200 examples; let this matches the second cluster of the third partition which has 50 examples and this matches the first cluster of the fourth partition which has 100 examples. Let the cluster centers be denoted $\bar{c}_{ij}$ for the $j^{th}$ cluster of the $i^{th}$ partition. Each cluster center is a vector of dimension $s$ for $s$ features. So, we could create one cluster of the global partition call it

$$
\bar{c}_{gi} = \frac{100 \times \bar{c}_{i1} + 200 \times \bar{c}_{i2} + 50 \times \bar{c}_{i3} + 100 \times \bar{c}_{i4}}{450}
$$

After the global cluster centroids are created, all of the clustered examples can be assigned to the nearest centroid using a distance metric. In this work, the Euclidean distance is used. To avoid sending lots of data between processors, the cluster centroids themselves can be passed to other processors and the cluster assignment of the examples made locally.
Large data sets may have many extrema or saddle points to which a clustering algorithm will converge depending upon the initialization. If some of them are significantly different, this will present a problem in combining centroids from different partitions. Even if the two partitions had exactly the same data, but were initialized differently, the final partition could be significantly different. In order to filter clusters that come from very different partitions, we developed the harmony algorithm. It will remove noisy or poorly matching centroids from a chain of centroids.

---

**Input:** Centroids of subset partitions  
**Output:** Global chain matrix

1. M=number of subsets, C=clusters in each subset, I=1, J=2. Global chain matrix is initialized to zero. Number the subsets/distributed sites 1 to M randomly.

2. While (J! = M)
   
   2.1 Local chain matrix initialized to zero. Select subset I and subset J and compute the Euclidean distance among the centroid vectors and store it in the distance matrix such that position (row, col) of the distance matrix is the Euclidean distance between row\textsuperscript{th} centroid vector of subset I and col\textsuperscript{th} centroid vector of subset J.

   2.2 Find the minimum value in the distance matrix and record it’s position (row, col) value in the local chain matrix.

   2.3 While (local chain matrix is not completely filled up)
      
      2.3.1 Find the next minimum value in the distance matrix. If a collision doesn’t occur, record the position (row, col) in the local chain matrix.

   2.4 If (I==1)
      
      The correspondence relation in the local chain matrix forms the first two columns of the global chain matrix.

      Else
      
      Fill the J\textsuperscript{th} column of the Global chain matrix using the local chain matrix and the transitive relation it has with the I\textsuperscript{th} column in the Global chain matrix.

2.5 I=I+1, J=J+1.

---

Figure 3.3 Consensus Chain Algorithm
3.4 Distributed Combining Algorithm

Now, we present the complete algorithm (Figure 3.4) for clustering large-scale data without clustering all the data at once.

<table>
<thead>
<tr>
<th>Input: Global data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Partition of the global data</td>
</tr>
</tbody>
</table>

1. Divide the data into M subsets. In case of distributed sites the data is already divided. So, no needs to perform this step then.

2. Cluster each subset/distributed location by a standard fuzzy k-means or hard k-means algorithm.

3. Call the consensus chain algorithm to form the global chain matrix.

4. For each consensus chain, compute the weighted (according to the number of examples in each subset) arithmetic mean of centroids in that consensus chain. The arithmetic mean of centroids in each consensus chain represents the centroid of a cluster of the global partition.

5. Compute the Euclidean distance between each example and the global set of centroids. Assign the example to the nearest cluster centroid.

Figure 3.4 Distributed (D)-Combining Algorithm

Since, we are clustering a part of the data in subsets; speedup is expected [4]. As mentioned earlier, step 3 generally takes constant time. Step 5 will take O(n) time, where n=number of examples in global data.
3.5 Computing Disputed Examples

To measure the quality of the clusters the partition formed by our Distributed-combining algorithm is compared to the partition formed by global clustering, and we call the examples that are placed in different clusters in the above two partitions disputed examples. If we have C clusters formed in a partition then examples in a cluster can be assigned a label number. A collection of such label numbers is called a label vector. If a partition has C clusters then its label number varies from 1 to C. For computing disputed examples we need to solve the correspondence relation between the label vector produced by our D-combining algorithm and the label vector of the most typical global partition. The correspondence relation between these two label vectors could be solved using logic like the consensus chain algorithm with little modification. The distance matrix will be replaced by the confusion matrix formed by the two label vectors and instead of finding C (number of clusters) non-colliding smallest values from a distance matrix, it finds C non-colliding values from the confusion matrix to pair up the labels of the two partitions, such that their sum is globally optimally maximum (using the Hungarian method of max-cost assignment [33]). Disputed examples will provide a measure of the quality of the partition found by our D-combining algorithm i.e. how much our partition differs when compared to a global partition.

The complete algorithm for computing disputed examples is shown in Figure 3.5.

<table>
<thead>
<tr>
<th>Input: Two label vectors whose disputed examples are to be found.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Number of disputed examples between the 2 input label vectors.</td>
</tr>
</tbody>
</table>

1. Declare a C x C confusion matrix and initialize it to zero.
2. Read the first label number of the first input label vector and first label number of the second input label vector.
3. Increment the (row
\text{th}, col
\text{th}) position of the confusion matrix by one, where row=label number of the first input label vector and col=label number of the second input label vector.
4. Read the next label number from the first input label vector and the second input label vector and perform step 3.
5. Continue step 3 and 4 until all the labels are read from both file.
6. Select C non-colliding optimal matched labels by using the Hungarian method of assignment problem (max-cost).
7. Add up all the values outside the position of the matched labels in the confusion matrix to find the number of disputed examples.

Figure 3.5   Algorithm for Confusion Matrix
Illustrative Example:

Consider there are 10 examples in the global data and the most typical global partition produced 3 clusters and the label vector is shown is Figure 3.6. Let us assume that all examples of the first cluster have label 1, second cluster have label 2, and the third cluster have label 3. Please note that the cluster label numbers are symbolic in nature i.e. cluster label 1 of a cluster doesn’t mean that it is necessarily same as cluster label 1 of another cluster.

```
<table>
<thead>
<tr>
<th>Example Number</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Figure 3.6  Example Label Vector of Global Partition of 10 Examples

Now, let us assume that the label vector of our D-combining algorithm is shown in Figure 3.7:

```
<table>
<thead>
<tr>
<th>Example Number</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Figure 3.7  Example Label Vector Formed by D-Combing Algorithm
The confusion matrix formed (as described in Figure 3.5) by the above two label vectors (Figure 3.6 and Figure 3.7) is shown in Figure 3.8 (a) and Figure 3.8 (b) shows the matched label pairs. The number or entry of the table (Figure 3.8 a) is the number of times labels $i$ from the global partition matches label $j$ from the D-combining algorithm.

The row represents the label number of the global partition ($Label_g$) and column represents ($Label_d$) the label number of the D-combining algorithm.

<table>
<thead>
<tr>
<th></th>
<th>$Label_d$ 1</th>
<th>$Label_d$ 2</th>
<th>$Label_d$ 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Label_g$ 1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>$Label_g$ 2</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$Label_g$ 3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) Confusion matrix

(b) Matched label pairs

Figure 3.8  Confusion Matrix and the Matched Label Pairs

After applying the Hungarian-method of max-cost assignment algorithm [33] (taken from Knuth's Stanford Graph base) to the confusion matrix (Figure 3.8 a), it finds 3 non-colliding values such that their sum is optimally the maximum. The 3 selected values give the matched labels (Figure 3.8 b).

Number of disputed examples computed (using Figure 3.8) are: $0+0+0+0+1+0=1$
Chapter 4  Centroid Filtering

The centroids of the consensus-chain are clearly similar if each of them has the same distance from the other. This would be an unlikely extreme case. An information theoretic formulation is used to measure the harmony of the centroids in a chain as shown in Equation 4. Initially all centroids are active i.e. ready to participate in the merging process.

\[
H(CH_{i}) = \frac{1}{\log_A} \sum_{j=1}^{A} \left( \frac{d(j)}{\log_{\text{tot}_d} d(j)} \right),
\]

where \( A \) is a number of active centroids in a consensus chain, \( d(j) \) is the sum of the distances of the \( j^{th} \) active centroid vector from all other active centroid vectors in a consensus chain, \( \text{tot}_d = d(1) + d(2) + \ldots + d(A) \), for \( n \) centroids in a chain; and \( CH_{i} \) indicates the chain for the \( i^{th} \) global cluster.

After computing the harmony, \( H \), using all the centroids in the chain, we check whether the elimination of one of the centroids from the chain will increase the harmony value above a threshold. If it does, that centroid is made inactive. An inactive centroid will not be used in creating the global centroid. After a centroid is removed, the algorithm can be applied to this shortened chain. It halts when there is no choice that will increase the \( H \) value or there are only three centroids left. The filtering algorithm is applied when there is a collision in forming the consensus chain. If all centroids are exactly same in a consensus chain, then the default harmony=1.

Illustrative Example:

Let’s consider the case of three centroids (in a consensus chain). The three centroids will be in perfect harmony (value 1) if the Euclidean distance among them is equal.

![Figure 4.1](image)

Figure 4.1  Example of Perfect Harmony among Centroids in a Consensus Chain. Three centroids are equidistant from each other. Harmony is maximum i.e. 1
Chapter 5  Data and Experiments

We have performed experiments on the small Iris data set because it is tractable and on an MRI data set with air removed. Experiments have also been done on an artificial data set to examine the effects of subsets that do not have balanced class distributions. An experiment has been done on a medium-sized data set consisting of features extracted from underwater images of plankton [20].

5.1 Iris Plant Database

The Iris plant data set consists of 150 examples each with 4 numeric attributes [19]. It consists of 3 classes of 50 examples each. One class is linearly separable from the other two.

5.2 MRI Database

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

5.3 Synthetic Data

500 examples were generated from 4 slightly overlapping Gaussian distributions in 2-D. The mean of the Gaussian are (300, 300), (800,300), (300,800), and (800,800). The standard deviations of the mixtures are 100 (the same in both dimension). This is shown in Figure 5.1.

![Figure 5.1 Synthetic 2-D Gaussian Distributed Data. Different shades of “+”, “x”, “*”, and “o” symbols indicate different clusters](image)

20
## 5.4 Plankton Data

The plankton data consists of 350,000 samples of plankton from the underwater SIPPER camera which records 8 gray levels. There were 37 features extracted. The samples were taken from the twelve most commonly encountered classes of plankton during the acquisition in the Gulf of Mexico. The class sizes range from about 9,000 to 44,000 examples.

## 5.5 Initialization of Centroids

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

## 5.6 Experimental Setup

The full data was clustered 50 times in a single memory with random centroid initialization within the range of the data for all data sets except the plankton data. We kept a count of the extrema (or saddlepoints, but for the proceeding we will just discuss extrema), which occurs most often. The average number of examples put into different clusters (disputed examples) by clustering all the data and distributing the data into smaller subsets and then merging the partitions was recorded. The standard deviation of disputed examples from each of the extrema with the most often occurring extrema was computed.

Each data set, with the exception of the plankton data set, is broken into subsets and each one of them clustered and combined as described above using 50 pure random and 50 semi-random centroid initializations. We computed the average number of disputed examples and their standard deviation of the extrema found after combination with the most typical extrema found during global clustering. Thus we compare the results of our distributed algorithm with the most typical global partition. Since there can be more than one extrema in a data set, we have chosen the clusters of the most typical extrema as the reference frame, and plotted the disputed examples of similar types of extrema produced by our combining algorithm. Disputed examples of other types of extrema produced by the D-combining algorithm can also be plotted by choosing the closest extrema (by J value) from the global clustering and making that the reference frame. For the Iris data we have used 2 attributes (petal length and petal width) to plot the disputed examples because these features contain most of the information about the data.
5.7 Iris Experiments

We performed experiments with the Iris data by dividing it randomly into 2 and 3 subsets and clustering each subset using the fuzzy-k means and the hard k-means algorithm. The number of clusters in each subset was designated as 3. Tables 5.1 and 5.2 show the average number of disputed examples and their standard deviation found during global clustering and measured relative to the most typical extrema. Tables 5.3 and 5.4 show the results from applying the D-Combining algorithm. Using 2 subsets and fuzzy-k means on each subset, the same clusters were obtained as with global clustering. With 3 subsets, we got only 2 disputed examples on average. The average standard deviation of disputed examples is 0, like from global clustering. Hence clustering of 3 subsets by the combining algorithm also produces just 1 type of extrema. We plotted the 2 disputed examples in Figure 5.2 (a) and found that these disputed examples (encircled ones) lie on the border of global clusters in an area of overlap as you might expect because there little difference in the position of cluster centroids. Clustering the global data by hard-k means gives more than one extrema (Figure 5.2 b). Similarly, our combining algorithm also results in more than one extrema (Figure 5.3 a, b), and the patterns of extrema are quite similar to Figure 5.2 b (extremas of the global cluster). The disputed examples found using hard-k means in each subset also lie on the border (Figure 5.4 a) as expected.

Comparison of Figure 5.4 (b) and Figure 5.3 (b) show that clustering subsets by pure-random centroid initialization may produce more extremas than semi-random initialization for an equivalent number of subsets. This was because the likelihood of combining extremas from different subsets, whose centroids differ significantly, appears larger during pure-random initialization than in semi-random initialization. This is because every subset has been initialized with centroids independent of the other, thus the probability of convergence of the objective function to a J value, which differs significantly is high. It has been observed that under this condition the centroids of subsets have more collisions in the consensus chain algorithm.

<table>
<thead>
<tr>
<th>(3 clusters) (150 examples)</th>
<th>Average disputed examples</th>
<th>Standard Deviation of disputed examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.2  Results of Clustering (Hard) Iris Global Data Set and 50 Random Initializations of Centroids

<table>
<thead>
<tr>
<th>(3 clusters)</th>
<th>Average disputed examples</th>
<th>Standard Deviation of disputed examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>(150 examples)</td>
<td>7.9</td>
<td>20.63</td>
</tr>
</tbody>
</table>

Table 5.3  Results of Our D-combining Algorithm (Iris Data). Fuzzy-k-means Applied to Each Subset

<table>
<thead>
<tr>
<th>(K=3 in each subset)</th>
<th>Average disputed examples (Semi-random)</th>
<th>Standard Deviation of disputed examples (Semi-random)</th>
<th>Average disputed examples (Pure-random)</th>
<th>Standard Deviation of disputed examples (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Subsets (Fuzzy)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3-Subsets (Fuzzy)</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.4  Results of our D-Combining Algorithm (Iris Data). Hard-k-means Applied to Each Subset

<table>
<thead>
<tr>
<th>(K=3 in each subset)</th>
<th>Average disputed examples (Semi-random)</th>
<th>Standard Deviation of disputed examples (Semi-random)</th>
<th>Average disputed examples (Pure-random)</th>
<th>Standard Deviation Of disputed examples (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Subsets (Hard)</td>
<td>4.82</td>
<td>15.38</td>
<td>8.74</td>
<td>20.07</td>
</tr>
<tr>
<td>3-Subsets (Hard)</td>
<td>8.3</td>
<td>11.69</td>
<td>9.8</td>
<td>14.43</td>
</tr>
</tbody>
</table>
(a) 2 disputed examples (encircled) for 3 subsets-hard-k means in each subset (Iris data). Different shades of “+” and “*” symbols denote the clusters.

(b) Disputed examples of extremas of Iris (global hard clustering)

Figure 5.2 Experiments with Iris Data
(a) Disputed examples of extremas of Iris-2 subsets-hard-k means in each subset (semi-random initialization)

(b) Disputed examples of extremas of Iris-3 subsets-hard-k means in each subset (semi-random initialization)

Figure 5.3  Result of D-Combining Algorithm on Iris Data with Semi-Random Initialization
Different shades of “+” and “*” symbols represent different clusters.

(a) Encircled examples showing the 8 disputed examples of Iris data (Hard k means applied to data in each subset)

(b) Disputed examples of extremas of Iris data using 3 subsets-hard-k-means in each subset (Pure-random initialization)

Figure 5.4 Result of D-Combining Algorithm on Iris Data with Pure-Random Initialization
5.8 MRI Experiments

We have performed experiments on the 22,320 example MRI data using 2, 3, 4, and 5 subsets. MRI data has clusters of varying size and density. One could also utilize other clustering algorithms on each subset, which better deal with differing size and density clusters. We conducted experiments by initializing the number of clusters, \( k \), to 7 in each subset. Tables 5.5 and 5.6 show the result of globally clustering the MRI data using fuzzy-k-means and hard-k-means respectively. The results of combining MRI data using 2, 3, 4, and 5 subsets and clustering using fuzzy-k means and hard-k-means are shown in Tables 5.7 and 5.8 respectively. The results show that the average number of disputed examples and the standard deviation of extremas obtained by our D-Combining algorithm using semi-random initialization is quite consistent with the average number of disputed examples and standard deviation of extremas obtained during global clustering of the MRI data (using \( k=7 \)). This means that the pattern of extremas generated by our D-Combining algorithm using semi-random initialization is quite similar to the pattern of extremas generated by clustering all the data at a time. Figure 5.5 (a) shows the histogram of the disputed examples of all extremas that occurred during global clustering of the MRI data (using fuzzy k means, \( k=7 \)). It has lots of extremas whose disputed examples are 0 (most typical) or close to zero. There is also an equal number of extremas whose disputed examples are large (6796 at the other end of the histogram in Figure 5.5 a). The extremas generated by our algorithm using 5-subsets (fuzzy k means in each subset) generate a similar pattern of extremas (Figure 5.5 b). We also found some other types of extremas in the middle (Figure 5.5 b) i.e. (no such similar extrema exist in Figure 5.5 a). This is expected because the J values of many of the extremas in the global data are quite different (Figure 5.5 a). Thus, sometimes our algorithm combines a highly diverse ensemble of centroids. With pure-random initialization this occurs more often. We got similar results using hard clustering and with \( k=10 \). We plotted the disputed examples found by our D-Combining algorithm (using 5 subsets with hard k means in each subset). Figure 5.6 (a) shows the disputed examples only and Figure 5.6 (b) shows them with global clusters in the background. It shows that the disputed examples lie on the spatial border of clusters with small changes in the centroids’ locations. Similar plotting has been obtained by clustering subsets with fuzzy-k means. Figure 5.7 (a) shows 47 disputed examples using 5-subsets, fuzzy-k-means applied to examples of each subset, and \( k=7 \). Figure 5.7 (b) shows the disputed examples with global clusters in the background.
Table 5.5  Results of Clustering (Fuzzy) MRI Global Data Set and 50 Random Initializations of Centroids

<table>
<thead>
<tr>
<th>(7 clusters)</th>
<th>Average disputed examples</th>
<th>Standard Deviation of disputed examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy</td>
<td>2990.54</td>
<td>3407.42</td>
</tr>
</tbody>
</table>

Table 5.6  Results of Clustering (Hard) MRI Global Data Set and 50 Random Initializations of Centroids

<table>
<thead>
<tr>
<th>(7 clusters)</th>
<th>Average disputed examples</th>
<th>Standard Deviation of disputed examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard</td>
<td>4022.84</td>
<td>3720.89</td>
</tr>
</tbody>
</table>
Table 5.7  Results of Our D-Combining Algorithm (MRI data). Fuzzy-k-means Applied to Each Subset

<table>
<thead>
<tr>
<th>(7 clusters)</th>
<th>Average disputed examples (Semi-random)</th>
<th>Standard Deviation of disputed examples (Semi-random)</th>
<th>Average disputed examples (Pure-random)</th>
<th>Standard Deviation of disputed examples (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Subsets (Fuzzy)</td>
<td>3040.68</td>
<td>3229.14</td>
<td>3288.82</td>
<td>2665.02</td>
</tr>
<tr>
<td>3-Subsets (Fuzzy)</td>
<td>2980.76</td>
<td>3268.57</td>
<td>3380.84</td>
<td>2381.51</td>
</tr>
<tr>
<td>4-Subsets (Fuzzy)</td>
<td>3175.38</td>
<td>3213.51</td>
<td>3848.04</td>
<td>1815.08</td>
</tr>
<tr>
<td>5-Subsets (Fuzzy)</td>
<td>3073.8</td>
<td>3160.03</td>
<td>3469.36</td>
<td>1927.57</td>
</tr>
</tbody>
</table>

Table 5.8  Results of Our D-Combining Algorithm (MRI Data). Hard-k-means Applied to Each Subset

<table>
<thead>
<tr>
<th>(7 clusters)</th>
<th>Average disputed examples (Semi-random)</th>
<th>Standard Deviation of disputed examples (Semi-random)</th>
<th>Average disputed examples (Pure-random)</th>
<th>Standard Deviation of disputed examples (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Subsets (Hard)</td>
<td>4077.88</td>
<td>3630.29</td>
<td>3755.46</td>
<td>2947.79</td>
</tr>
<tr>
<td>3-Subsets (Hard)</td>
<td>4069.34</td>
<td>3661.02</td>
<td>3486.14</td>
<td>2355.76</td>
</tr>
<tr>
<td>4-Subsets (Hard)</td>
<td>4053.10</td>
<td>3278.49</td>
<td>3872.98</td>
<td>2434.57</td>
</tr>
<tr>
<td>5-Subsets (Hard)</td>
<td>4096.46</td>
<td>3282.17</td>
<td>3089.94</td>
<td>1765.54</td>
</tr>
</tbody>
</table>
(a) The disputed examples of extrema of clustering of all the MRI data (k=7) using fuzzy-k means

(b) The disputed examples of extrema obtained using 5 subsets of MRI data (fuzzy, k=7 in each subset, semi-random initialization)

Figure 5.5  Comparison of Global Fuzzy Clustering of MRI Data with D-Combining Algorithm
(a) Plotting of 1384 disputed examples of MRI data. No of subsets used is 5, k=7, and hard clustering in each subset

(b) Disputed examples lie on the spatial border (+green symbols) of clusters. Clusters have been represented using different colors

Figure 5.6  Plotting of Disputed Examples of MRI Data (Hard-k means, k=7)
(a) The plotting of 47 disputed examples of MRI data. No of subsets used is 5, k=7, and fuzzy clustering in each subset.

Figure 5.7 Plotting of Disputed Examples of MRI Data (Fuzzy-k means, k=7)
5.9 Using Harmony to Filter Centroids in the MRI Data

Table 5.9 shows the result of applying the Harmony algorithm with a 1% threshold. There’s a significant reduction in the number of disputed examples. It is most dramatic for the MRI data where the numbers of disputed examples go from the thousands to the hundreds. We believe that indicates that these partitions are essentially as good as those obtained from using all the centroids. Figure 5.8 (a) shows 1384 disputed examples obtained without using the Harmony algorithm. Figure 5.8 (b) shows that the number of disputed examples reduced to 346 from 1384 after applying harmony algorithm and Figure 5.8 (c) shows that the reduced disputed examples also lie on the spatial border of clusters of the global partition. This shows that Harmony algorithm effectively reduces the number of disputed examples on the spatial border of the clusters. We got similar plotting with fuzzy-k-means applied to each subset. It has been observed that the Harmony algorithm successfully removes noisy centroids i.e. when noise exists in minority proportion. It sometimes fails to filter centroids when a significant amount of noise is present in a consensus chain i.e. when there is no clear cut majority of a particular set of harmonious centroids in a consensus chain. In a real life scenario one or more distributed sites/subsets may be noisy. Thus it worth using it to fine tune the final partition.

Table 5.9 Using Harmony to Filter Centroids in the MRI Data Set

<table>
<thead>
<tr>
<th>(Boost Threshold used = 1%)</th>
<th>J1 (J2 for fuzzy) value of global cluster (most often occurring)</th>
<th>No. of Disputed examples from our D-combining algorithm</th>
<th>J1 value of D-combining algorithm.</th>
<th>No. of Disputed examples after applying Harmony booster</th>
<th>J1 value after applying Harmony booster</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRI (hard, 7 clusters, 5 subsets)</td>
<td>230359434.83</td>
<td>1384</td>
<td>234797809.09</td>
<td>346</td>
<td>231125304.72</td>
</tr>
<tr>
<td>MRI (fuzzy, 7 clusters, 5 subsets)</td>
<td>106134159.51</td>
<td>3395</td>
<td>245836863.33 (J1 value)</td>
<td>174</td>
<td>236170371.82 (J1 value)</td>
</tr>
</tbody>
</table>

33
(a) Left figure shows 1384 disputed examples of MRI data using 5 subsets-Hard-k-means in each subset (7 cluster, hard-k-means in each subset)

(a) Right figure shows the result after applying the Harmony-Booster. Disputed examples decreased to 346 (Boost threshold=1%)

Figure 5.8  Effect of Applying Harmony Algorithm to Consensus Chains of MRI Data (Using Hard-k-means, k= 7)
(c) Disputed examples (346 +green symbols) lie on the spatial border of clusters

Figure 5.8  Continued
5.10 Synthetic Data Experiments

This data set was used to provide an indication of what might happen in the case that the random subsets/distributed sites received a poor selection of examples. The data was broken into four subsets as follows. One subset was formed with 250 examples from class 1 and 250 examples from class 3, which correspond to the o and * respectively in Figure 5.1. So, this was a very non-representative subset. The other subsets got examples randomly selected from each of the classes with equal probability as long as an example was available. These data sets were slightly skewed. The data was clustered into four classes. In this case, one would expect the distributed partition to be at best a bit worse than the partition created with all of the data. Tables 5.10-5.12 show that this is so. However, after applying the harmony algorithm to smooth the cluster chains there are on average between 14 and 16 examples put in different classes as shown in Table 12 for hard k-means and there are 22 examples for fuzzy-k-means applied to each subset. While this is clearly more than the couple that changes with different random initializations using all the data, we believe it shows that the final partition would be quite usable because most times people are trying to get a general idea of how the data is grouped. Figure 5.9 (a) and (b) show the plotting of disputed examples before and after applying Harmony. Fuzzy-k-means has been used in each subset and it clearly shows that Harmony makes the partition better. Similar results have been plotted in Figure 5.10 (a)-(b), where hard-k-means has been used in each subset. Figure 5.11 shows the pattern of extemas formed by clustering all the data at once i.e. global clustering. It shows that all the extemas have disputed examples between 0-4 compared to the most typical global partition. Figure 5.12 (a) shows that combining partitions (hard-k-means, pure random initialization) without using Harmony resulted in extrema patterns whose disputed examples vary from 56 to 99 while after applying Harmony the number of disputed examples on average reduces to between 11 to 17 (Figure 5.12 b). Similar results have been obtained in Figure 5.13 (a)-(b) with semi-random centroid initialization. Please note in this experiment the result of semi-random and pure-random initialization doesn’t show much difference because the global data doesn’t appear to have multiple extrema (Figure 11), which differ among them significantly. Thus it is less sensitive to initialization.

Table 5.10 Results of Clustering (Fuzzy and Hard) Artificial Global Data Set and 50 Random Initializations of Centroids

<table>
<thead>
<tr>
<th></th>
<th>Average disagreement</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hard</strong></td>
<td>1.78</td>
<td>1.58</td>
</tr>
<tr>
<td><strong>Fuzzy</strong></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.11  Results of Our D-Combining Algorithm without Filtering Cluster Centers

<table>
<thead>
<tr>
<th>(K=4 in each subset) (500 examples in each subset)</th>
<th>Average disagreement (Semi-random)</th>
<th>Standard Deviation (Semi-random)</th>
<th>Average disagreement (Pure-random)</th>
<th>Standard Deviation (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-Subsets (Hard)</td>
<td>65.62</td>
<td>9.93</td>
<td>69.60</td>
<td>11.14</td>
</tr>
<tr>
<td>4-Subsets (Fuzzy)</td>
<td>92</td>
<td>0</td>
<td>92</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.12  Results of our D-Combining Algorithm after Filtering Cluster Centers Using Harmony

<table>
<thead>
<tr>
<th>(K=4 in each subset) (500 examples in each subset)</th>
<th>Average disagreement (Semi-random)</th>
<th>Standard Deviation (Semi-random)</th>
<th>Average disagreement (Pure-random)</th>
<th>Standard Deviation (Pure-random)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-Subsets (Hard)</td>
<td>15.66</td>
<td>1.75</td>
<td>14.5</td>
<td>2.31</td>
</tr>
<tr>
<td>4-Subsets (Fuzzy)</td>
<td>22</td>
<td>0</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>
(a) It shows 92 disputed examples (+ symbols) of synthetic data using 4 subsets-Fuzzy-k-means applied to each subset

(b) It shows the result after applying the Harmony algorithm. The number of Disputed is examples (+ symbols) decreased to 22 (Boost threshold=1%)
(a) 78 disputed examples (+ symbols) of the Synthetic data using 4 subsets-Hard-k-means in each subset

(b) The result after applying the Harmony algorithm. Disputed examples (+ symbols) are decreased to 17 (Boost threshold=1%)

Figure 5.10  Effect of Applying Harmony Algorithm to the Consensus Chains of Synthetic data (Using Hard-k-means, k= 4). One of the Subsets is Unbalanced. Different Colors of “.” Symbol Denotes Different Clusters
Count No. of disputed examples

Figure 5.11 Extrema of Global Clustering of Synthetic data. Shows the Disputed Examples of Extremas of Clustering of all the Synthetic Data (k=4) Using Hard-k means (Global clustering)

(a) Results after applying the D-Combining algorithm to Synthetic Data without using the Harmony algorithm to filter centroids (Pure-random initialization)

(b) Results after applying D-Combining algorithm to Synthetic Data with the Harmony algorithm to filter centroids (Pure-random initialization)

Figure 5.12 Effect of Applying the Harmony Algorithm to the Extrema Patterns of Synthetic Data with Pure-Random Initialization (Using Hard-k-means, k= 4)
Figure 5.13  Effect of Applying Harmony Algorithm to the Extrema Patterns of Synthetic Data with Semi-Random Initialization (Using Hard-k-means, k= 4)
5.11 Plankton Data Experiments

Due to the size of this data set, we are focusing on the time saved by clustering in a distributed way. All experiments were on a SUN Enterprise 3000 multiprocessor system using UltraSPARC II processors running at 248 MHz with 3 GB of RAM. All the data was clustered one time, which took approximately 407 minutes (nearly seven hours). It took approximately 43 minutes to cluster the data in a distributed way using 14 equal size subsets. The merging algorithm took near about 4.28 minutes to form the global partition. So, we got a speed up of 8.6 times. There were 17,028 (4.8%) disputed examples after merging the distributed partitions. Utilizing the harmony algorithm resulted in just 8147 (2.3%) disputed examples. Such an overall partition of the data should be quite useful given the typical instability of clustering algorithms and the fact that general cluster characteristics are likely preserved.
Chapter 6  Discussion and Conclusions

Experimental results show that the difference between the average number of disputed examples produced by our D-Combining algorithm and global clustering is 0.8% (averaged both for Fuzzy and Hard clustering, 3 subsets, semi-random initializations) on the Iris data and 0.34% (averaged both for Fuzzy and Hard clustering, 5 subsets, semi-random initializations) for the MRI data set. We have observed that the D-combining algorithm produces extremas (or clusters) more similar to the global extremas, if the centroids in different subsets map without any collision. If the global data set has lots of extremas and if their J1 (J2) values differ significantly, then it is more probable that the centroids of different subsets will collide during centroid mapping. This likelihood becomes higher during pure-random initialization of the centroids in each subset. It has been observed that our D-Combining algorithm produces more distinct extremas using pure-random centroid initialization than using semi-random centroid initialization.

The Iris data set when globally clustered using fuzzy k-means has only 1 extrema (Table 5.1), similarly, our D-Combining algorithm generates 1 extrema (Table 5.3). The MRI data when clustered globally using the fuzzy k-means algorithm produces extremas whose J2 (J1) value (thus also disputed examples) differ significantly (Figure 5.5 a). Similarly, our Distributed-combining algorithm also produces similar patterns of extremas, which differ significantly (Figure 5.5 b). Plotting the disputed examples shows that they lie on the spatial border of clusters of the global partition (Figure 5.2 a, 5.4 a, 5.6, 5.7, 5.8, 5.9, 5.10) as expected if the partitions are not radically different. Sometimes, our algorithm produced extremas, which didn’t exist in the global partition. This happens when the J value of extremas differs significantly in the global data and subsets also inherit that property. In summary, the quality of partition formed is more often better with semi-random initialization than with pure-random initialization.

Combining all the representatives in an ensemble of centroids in unsupervised learning is not always good due to the unstable characteristics of the clustering algorithms or non-representative subsets. We have shown with the artificial data that even if data in a subset is heavily skewed we can still recover good partitions i.e. average difference of disputed examples between D-Combining algorithm and global clustering is 3.85% (averaged both for Fuzzy and Hard clustering, 4 subsets, semi-random initializations) without using the filtering algorithm and 0.85% (averaged both for Fuzzy and Hard clustering, 4 subsets, semi-random initializations) after using the centroid filtering algorithm. Experimental
results show that minimizing the amount of diversity among an ensemble of centroids may give a better partition than merging all (Table 5.9, Figure 5.8, Table 5.12, Figure 5.9, Figure 5.10, Figure 5.12, and Figure 13). Thus on average the quality of partition improves by using Harmony algorithm. These partitions should be compared with single pass k-means in time and quality [8 and 13].

Clustering very large-scale data using hard-k means or fuzzy k-means is very time consuming. Sometimes, the data may be geographically distributed or may be too large to fit in a single memory. There may be constraints like data could not be shared between different distributed locations due to privacy, security, or the proprietary nature of the data. Extracting knowledge from these types of distributed locations under restraints of data exchange is called privacy preserving data mining. In this thesis we proposed a distributed clustering algorithm that provides a framework for integrating an ensemble of centroids to form a global partition. We reach a global consensus on the positions of the centroids after merging an ensemble of centroids.

As mentioned earlier the results on two real data sets show that the quality of clusters produced by our combining algorithm was similar (difference in the average number of disputed examples between the partitions of D-combining and global clustering is within 1%, averaged both for Fuzzy and Hard clustering ) to the quality of clusters generated by global clustering. The general cluster structure is maintained. If the initial centroid assignments of subsets are semi-random, our combining algorithm tends to produce an extrema pattern closer to the pattern of extrema found during global clustering. If pure random initialization is used, the similarity of extrema decreases a little. This is because the likelihood of a perfect mapping of centroids of disjoint subsets is higher during semi-random centroid initialization than in pure random centroid initialization.

Data sets having lots of extrema, whose J value differs heavily among them, sometimes, produce clusters dissimilar to those obtained from clustering all the data. To overcome this problem, we proposed a Harmony algorithm to “smooth” the cluster centroids to be combined. It tends to eliminate noisy or discordant centroids. Results show that use of the Harmony algorithm on the MRI data (Figure 5.8) reduces the number of disputed examples from 1384 to 346 on the border of global clusters.

A synthetic data set with 4 Gaussian classes was used to examine how this algorithm would perform in the case of a poor selection of data in the subsets. Results showed that even in this case, in conjunction with the Harmony algorithm, a partition that was a reasonable approximation (difference in the average number of disputed examples between the partitions of D-combining and global clustering is 0.85%, averaged both for Fuzzy and Hard clustering) of the data partition gotten from clustering with all the data was obtained, improving the quality of clusters.
A medium-sized real world data set of 350,000 examples, 37 dimensions and 12 classes was used to look at both the potential speed up of this approach and how well it matched clustering with all the data on a bigger set. The clustering was accomplished significantly faster, approximately 9 times, using 14 subsets of data. The final partition matched the partition obtained after clustering with all the data within 2.3%.

Our assumption behind merging the centroids of distributed partitions is that data in all distributed sites are from the same underlying distribution. If the above assumption is violated i.e. if all the distributed partitions are radically different from each other then our merging algorithm will still merge those partitions and the centroid filtering algorithm may not detect any noisy centroids because in the consensus chain there may be no consensus for a global centroid. For example, a problem would arise if there were 6 clusters and 3 sites each of which had data from 2 clusters with all pairs of clusters disjoint. In future we plan to detect a scenario like this.

The approach presented here is a scalable, distributed approach that can be applied to well-understood iterative clustering algorithms. It also provides a framework or privacy preserving data mining. We have shown it provides very representative clustering or data partitions. It allows partitions to be built from very large data sets, which will closely approximate those obtained from clustering with all the data and using algorithms such as hard k-means or fuzzy k-means.
References


