

A Binary Method for Fast Computation of Inter and Intra Cluster Similarities for Combining Multiple Clusterings

Selim Mimaroglu and A. Murat Yagci
Department of Computer Engineering
Bahcesehir University
Ciragan Caddesi 34353 Besiktas, Istanbul, Turkey
selim.mimaroglu@bahcesehir.edu.tr
murat.yagci@bahcesehir.edu.tr

Abstract

In this paper, we introduce a novel binary method for fast computation of an objective function to measure inter and intra class similarities, which is used for combining multiple clusterings. Our method has the advantages of using less memory and CPU time. Moreover, compared with the conventional technique we reduce the time complexity of the problem considerably. Experimental test results demonstrate the effectiveness of our new method.

1. Introduction

Clustering is the process of organizing objects into groups which have similar members; a distance metric is used for evaluating the similarity. Clustering, which is an important problem, is also known as unsupervised classification in the literature. In a cluster, hopefully, objects are similar to each other and they are dissimilar to other objects in other clusters.

Some of the clustering algorithms partition objects into groups, therefore known as partitional clustering algorithms. k -means [5] is the best known partitional clustering algorithm. k -means divides a set of objects into k clusters, where k is a user specified parameter. Another type of clustering is hierarchical clustering algorithms; agglomerative methods are well known techniques for this type. Yet another type of clustering is density based clustering; *DBScan* is a popular algorithm that can correctly cluster arbitrary shaped objects, when provided right parameters.

An expert, by using some of the attributes and his/her experience, can provide a clustering. Therefore, for the same data set there may be multiple clusterings.

An important problem, which is known as *Combining Multiple Clusterings*, is combining existing clusterings into

a final clustering. In the literature there are some good methods and techniques for solving this problem. Also, some methods are provided for evaluating the quality of the final clustering; such as entropy, F-measure, intra-cluster similarity, and inter-cluster similarity. Main contribution of this paper is a new technique for efficiently computing intra-cluster, and inter-cluster similarities for combining multiple clusterings.

The paper is structured as follows. In Section 2, the problem of combining multiple clusterings is introduced. Section 3 includes intra-cluster and inter-cluster similarities. Section 4 presents our new binary method for computing intra-cluster and inter-cluster similarities. Experimental Results section provides comparison of our new method with the conventional technique for varying size data sets. In the final section, we present conclusions and future work.

2. Combining Multiple Clusterings

Combining multiple clusterings, which is also known as the Cluster Ensemble problem, refers to combining multiple clusterings (partitions) into a new, final clustering. This problem has been studied by many researchers in data mining. Some of the interesting resources on this topic can be listed as [7], [8], [4], [3], [2], [6], and [1]. When combining multiple clusterings, it is expected that the final clustering has better overall quality. Combining multiple clusterings provide reusing pre-existing knowledge, employing distributed data mining methods, and producing a final clustering having better overall quality. Some solutions for combining multiple clusterings are based on genetic algorithms such as [2, 6]. Some other techniques are based on simulated annealing [3]. A greedy optimization, info-theoretical method is presented in [7]. And, [1] presents a method based on ant colony optimization. All these methods operate on a consensus function.

Let D be a data set. A clustering (partition) of D , $\pi(D)$, can be stated as follows:

$$\pi(D) = \{C_1, C_2, \dots, C_{|\pi(D)|}\},$$

where C_i is a cluster (block) of $\pi(D)$, $1 \leq i \leq |\pi(D)|$, and

$$D = \bigcup_{i=1}^{|\pi(D)|} C_i$$

Given a set of clusterings $\Pi = \{\pi_1, \pi_2, \dots, \pi_m\}$, the problem of combining multiple clusterings is defined as finding a new clustering $\pi^* = \{C_1^*, C_2^*, \dots, C_{|\pi^*|}^*\}$ by using the information provided by Π . A consensus function ϕ ,

$$\forall i (\phi(\pi^*(D)) \geq \phi(\pi_i(D))), 1 \leq i \leq |\Pi| \quad (1)$$

is used for determining quality of the final clustering π^* . Exhaustively searching all the possible clusterings for finding *the best* clustering is not an option, since there are too many possible clusterings (see [7]).

Many consensus functions have been proposed in the literature. In [4], proposed consensus function is based on a co-association measure (simultaneous occurrences) defined by

$$coassoc(i, j) = votes_{ij}/m, \quad (2)$$

where m is the number of clusterings, $votes_{ij}$ is the number co-occurrences of o_i and o_j in a cluster, such that $o_i \in D$ and $o_j \in D$. In this approach, the co-occurrences are held in a $|D| \times |D|$ co-association matrix. In general, it is costly to construct, store and populate this matrix for large data sets, because of its $O(|D|^2)$ complexity. Besides, searching cluster similarity continuously at the object level is a computationally expensive task.

In [3] and [2], consensus functions based on median partition approach have been proposed. This approach searches differences between the clusterings, by working on a coarser level. At another direction, in [7], consensus functions based on hypergraphs have been proposed. In this technique, a hyperedge represents a cluster, and a hypergraph represents a clustering. We focus on a specific consensus function and optimize its computation time and memory footprint. This consensus function is presented in the next section.

3. Inter and Intra Cluster Similarities

Intra-cluster similarity measures how near the data objects are in a cluster. For a clustering, $\pi(D) = \{C_1, C_2, \dots, C_{|\pi(D)|}\}$, intra-cluster similarity is measured as follows:

$$ICS(\pi(D)) = \sum_{i=1}^{|\pi(D)|} \frac{1}{|C_i|^2} \sum_{d, d' \in C_i} similarity(d, d') \quad (3)$$

For the same clustering, inter-cluster similarity is defined as follows:

$$ECS(\pi(D)) =$$

$$\sum_{i=1}^{|\pi(D)|} \sum_{j=i+1}^{|\pi(D)|} \frac{1}{|C_i||C_j|} \sum_{d \in C_i, d' \in C_j} similarity(d, d') \quad (4)$$

Finally, the consensus function is

$$\phi(\pi(D)) = k_1 \cdot ICS(\pi(D)) + k_2 \cdot ECS(\pi(D)) \quad (5)$$

Final clustering is expected to have compact, and close clusters, therefore for a good clustering Formula 3 should provide large values. Similarly, separated (isolated) clusters are expected in a good clustering, therefore Formula 4 shall supply small values. k_1 , and k_2 parameters in Formula 5 are user defined values, which satisfy $k_1 > 0$ and $k_2 < 0$.

Inspection of Formula 5 reveals that its time complexity is quadratic with respect to the number of objects in the data set, $O(|D|^2)$. This complexity is due to pairwise similarity calculations performed in Formulas 3 and 4. Our aim is to reduce this complexity, and compute Formula 5 faster using binary methods.

It is very important to note that in combining multiple clustering problem, $similarity(d, d')$ is not computed from the data set D . $similarity(d, d')$ in Formulas 3 and 4 refer to number of co-occurrences of object d , and d' in the same cluster (related to Formula 2). Therefore, it is crucial to utilize the information provided by the pre-existing multiple clusterings.

4. Fast Computation of Inter and Intra Cluster Similarities

We represent each cluster with a bit vector. Existence of an object in a cluster is shown by 1, similarly absence of an object is captured by 0. Each cluster representation is as long as the size of the database, $|D|$. Three clusterings, each having four clusters are shown in Figure 1. An example of a cluster is shown below: C_{11} cluster has d_1, d_3 , and d_6 objects.

$$\begin{array}{c} C_{11} \\ \boxed{1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0} \end{array}$$

In order to compute Formulas 3 and 4, each cluster has to be examined for pairwise objects, and corresponding entries in the co-association matrix has to be updated. For example C_{11} cluster increments following object pairs in the co-association matrix: (d_1, d_1) , (d_1, d_3) , (d_1, d_6) , (d_3, d_3) , (d_3, d_6) , and (d_6, d_6) . Because of the pairwise increment nature, this computation has quadratic time complexity. We present our novel method below for reducing the time complexity of this operation.

Let Π be multiple clusterings, and π^* be a new final clustering for a data set D . In following we define intra-cluster, and inter-cluster similarities of $\pi^*(D)$. Intra-cluster similarity of $\pi^*(D)$ is shown in Formula 6

$$ICS_{\Pi}(\pi^*(D)) = \sum_{k=1}^{|\pi^*(D)|} \frac{1}{|C_{\star k}|^2} \sum_{i=1}^{|\Pi|} \sum_{j=1}^{|\pi_i|} \binom{|C_{\star k} \wedge C_{ij}|}{2} \quad (6)$$

And, Formula 7 shows the inter-cluster similarity of $\pi^*(D)$.

$$ECS_{\Pi}(\pi^*(D)) = \sum_{k=1}^{|\pi^*(D)|} \sum_{l=k+1}^{|\pi^*(D)|} \frac{1}{|C_{\star k}| |C_{\star l}|} \sum_{i=1}^{|\Pi|} \sum_{j=1}^{|\pi_i|} \left(\binom{|(C_{\star k} \vee C_{\star l}) \wedge C_{ij}|}{2} - \binom{|C_{\star k} \wedge C_{ij}|}{2} - \binom{|C_{\star l} \wedge C_{ij}|}{2} \right) \quad (7)$$

Finally, consensus function is described as:

$$\phi_{\Pi}(\pi^*(D)) = k_1 \cdot ICS_{\Pi}(\pi^*(D)) + k_2 \cdot ECS_{\Pi}(\pi^*(D)) \quad (8)$$

$ICS_{\Pi}(\pi^*(D))$ is computed as follows; every cluster of $\pi^*(D)$ is logically ANDed with every cluster in Π in order to find pairwise co-occurrences of the objects in the same cluster. Similarly, when computing $ECS_{\Pi}(\pi^*(D))$ every cluster pair in $\pi^*(D)$ is logically ORed in order to find all pairs of objects, this result is ANDed with every cluster in Π in order to find pairwise co-occurrences of objects. So far we obtained the pairwise co-occurrences of objects in the same clusters and in two different clusters. By subtracting (last 2 components in Formula 7) pairwise co-occurrences of the objects in the same clusters, we obtain pairwise co-occurrences of objects in different clusters as shown in the formula. When computed on the same clustering, Π , Formula 6 is equivalent to Formula 3 and Formula 7 is equivalent to Formula 4. Although equivalent formulas yield same result, it is very important to note that Formulas 6 and 7 are computed in cluster level, not in object level for each pairwise object. Therefore, Formulas 3 and 4 are very efficient when compared to the Formulas 6 and 7. As a result, time complexity of Formula 8 is reduced

to $O(|\pi^*(D)| |\Pi|_{tc} + |\pi^*(D)|^2 |\Pi|_{tc})$, where $|\Pi|_{tc}$ represents total number of clusters in $|\Pi|$ (e.g. $|\Pi|_{tc}$ in Figure 1 is 12) and $|\Pi|_{tc} \ll |D|$, and $|\pi^*(D)|^2 \ll |D|$. In the next section effectiveness of our new technique is demonstrated.

Example 4.1 Let us compute intra-cluster similarity, $ICS_{\Pi}(\pi^*(D))$, of $\pi^*(D)$ of Figure 2 using multiple clusterings shown in Figure 1 by using the Formula 6

$$ICS_{\Pi}(\pi^*(D)) = \frac{1}{3^2} \left(\binom{2}{2} + \binom{0}{2} + \dots + \binom{1}{2} \right) + \frac{1}{2^2} \left(\binom{1}{2} + \binom{0}{2} + \dots + \binom{0}{2} \right) + \dots + \frac{1}{1^2} \left(\binom{0}{2} + \binom{0}{2} + \dots + \binom{0}{2} \right)$$

In a similar manner, Formula 7 can be used for computing $ECS_{\Pi}(\pi^*(D))$.

5. Experimental Results

We have conducted experiments on varying size data sets using a computer having 2.0GHz processor with 2GB main memory. Our choice of implementation language is Java, which provides built-in support for bit vectors, and operations on bit vectors.

In Figure 3 execution time results of a data set having 5,000 objects is shown. Clearly, our *Binary Method* is superior to the conventional method. We provided the time axis in logarithmic scale, because our execution time results are very small when compared to the conventional case. Figures 4, and 5 present similar results. Overall, in the worst case our new method is 281.2 times faster and in the best case our new method is 2,369.05 times faster than the conventional technique.

Figure 6 displays execution time data in detail. And, Figure 7 shows memory consumption of our method and the conventional technique. Our binary method uses remarkably low memory, which enables working with very large data sets.

6. Conclusions and Future Work

In this paper, we presented a novel binary method for computing intra-cluster and inter-cluster similarities. By representing each cluster using a bit vector, we utilize fast operations and low memory requirements of binary operations. Our method is especially useful as a consensus function when combining multiple clusterings.

| | | d_1 | d_2 | d_3 | d_4 | d_5 | d_6 | d_7 | d_8 |
|---------|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| π_1 | C_{11} | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| | C_{12} | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | C_{13} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | C_{14} | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| π_2 | C_{21} | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | C_{22} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | C_{23} | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | C_{24} | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| π_3 | C_{31} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | C_{32} | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| | C_{33} | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| | C_{34} | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Figure 1. Binary representation of multiple clusterings, Π

| | | d_1 | d_2 | d_3 | d_4 | d_5 | d_6 | d_7 | d_8 |
|---------|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| π^* | C_{*1} | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | C_{*2} | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | C_{*3} | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | C_{*4} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Figure 2. Binary representation of π^*

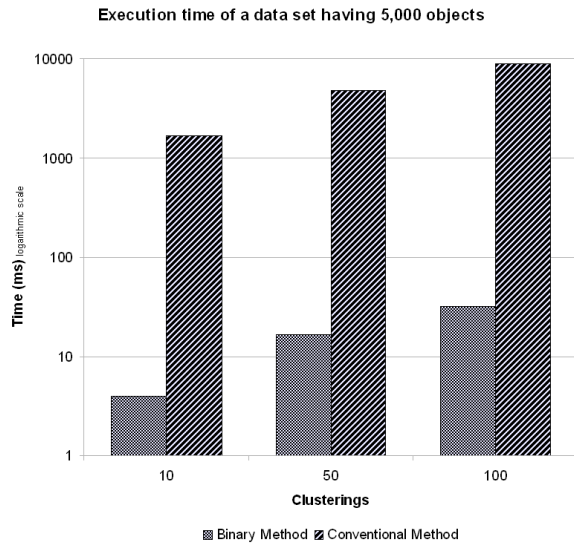


Figure 3. Execution time on 5,000 objects

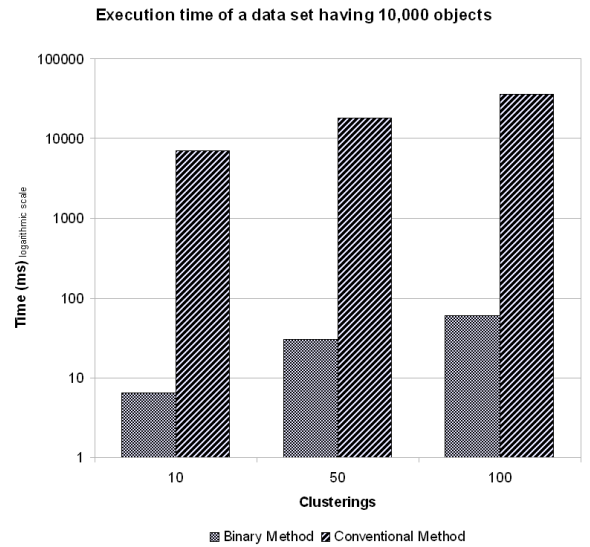


Figure 4. Execution time on 10,000 objects

Test results on multiple data sets and clusterings demonstrate the effectiveness of our method. Our method is superior to the conventional technique; it is much faster and it uses considerably less memory.

By using the consensus function here, we will investigate designing a novel method for combining multiple clusterings as a future work. Evolutionary based techniques can

use our consensus function presented here as a fitness function. Generalizing our technique for broader use will also be a future work.

| No. of data objects | No. of clusterings | No. of clusters | ICS with binary operations Run Time (ms) | ICS with conventional operations Run Time (ms) | ECS with binary operations Run Time (ms) | ECS with conventional operations Run Time (ms) | Time for co-association matrix generation (ms) |
|---------------------|--------------------|-----------------|--|--|--|--|--|
| 5,000 | 10 | 4 | 1.54 | 156.12 | 2.45 | 437.34 | 1,085.42 |
| 5,000 | 50 | 4 | 3.88 | 153.19 | 12.62 | 438.63 | 4,216.18 |
| 5,000 | 100 | 4 | 6.62 | 153.66 | 24.99 | 432.91 | 8,302.05 |
| 10,000 | 10 | 4 | 1.93 | 595.18 | 4.58 | 1,764.97 | 7,088.23 |
| 10,000 | 50 | 4 | 6.32 | 598.15 | 24.28 | 1,789.65 | 18,109.66 |
| 10,000 | 100 | 4 | 11.99 | 591.54 | 48.05 | 1,746.57 | 36,027.45 |
| 20,000 | 10 | 4 | 2.87 | 2,296.57 | 9.23 | 6,986.60 | 19,382.31 |
| 20,000 | 50 | 4 | 11.21 | 2,294.12 | 46.82 | 6,903.30 | 91,530.38 |
| 20,000 | 100 | 4 | 21.68 | 2,318.60 | 93.95 | 6,929.19 | 184,259.79 |

Figure 6. Execution time results

| No. of data objects | No. of clusterings | No. of clusters | Bitset operations JVM memory footprint (MB) | Integer type Matrix based conventional operations JVM memory footprint (MB) | Approximate Memory need for data structures in Bitset operations (MB) | Approximate Memory need for data structures in Matrix based operations (MB) |
|---------------------|--------------------|-----------------|---|---|---|---|
| 5,000 | 100 | 4 | 25 | 150 | 0.25 | 100 |
| 10,000 | 100 | 4 | 25 | 420 | 0.50 | 400 |
| 20,000 | 100 | 4 | 25 | 1,650 | 1.00 | 1,600 |

Figure 7. Memory consumption results

References

- [1] J. Azimi, P. Cull, and X. Fern. Clustering Ensembles Using Ants Algorithm. In *Proceedings of the 3rd International Work-Conference on The Interplay Between Natural and Artificial Computation: Part I: Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy*, page 304. Springer, 2009.
- [2] D. Cristofor and D. Simovici. Finding median partitions using information-theoretical-based genetic algorithms. *Journal of Universal Computer Science*, 8(2):153–172, 2002.
- [3] V. Filkov and S. Skiena. Heterogeneous data integration with the consensus clustering formalism. *Lecture notes in computer science*, pages 110–123, 2004.
- [4] A. Fred and A. Jain. Data clustering using evidence accumulation. In *International Conference on Pattern Recognition*, volume 16, pages 276–280. Citeseer, 2002.
- [5] J. MacQueen et al. Some methods for classification and analysis of multivariate observations, 1966.
- [6] M. Mohammadi, A. Nikanjam, and A. Rahmani. An Evolutionary Approach to Clustering Ensemble. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 3, 2008.
- [7] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.
- [8] A. Topchy, A. Jain, and W. Punch. Combining multiple weak clusterings. In *Third IEEE International Conference on Data Mining, 2003. ICDM 2003*, pages 331–338, 2003.

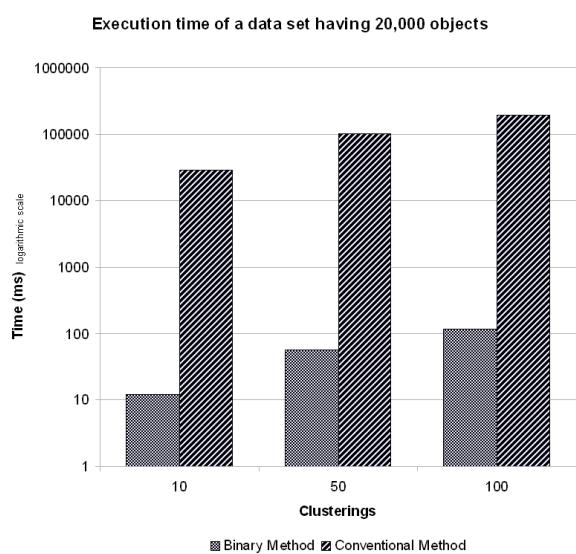


Figure 5. Execution time on 20,000 objects