Combining clustering solutions with varying number of clusters

Geeta Aggarwal[1], Saurabh Garg[2] and Neelima Gupta[3]
[1,2,3]Department of Computer Science
University of Delhi
INDIA

Abstract

Cluster ensemble algorithms have been used in different field like data mining, bioinformatics and pattern recognition. Many of them use label correspondence as a step which can be performed with some accuracy if all the input partitions are generated with same k. Thus these algorithms produce good results if this k is close to the actual number of clusters in the dataset. This puts great restriction if user has no idea of the number of clusters. In this paper we show through experimental studies that good ensembles can be generated even if the input solutions contain different number of clusters.

Keywords—Clustering, Cluster Ensemble

1. Introduction

Clustering deals with segregating data into well define groups. The aim is to group objects so as to maximize similarity between the objects within a group and minimize the similarity between the objects of different groups. Several algorithms have been developed to achieve the desired aim. All these clustering algorithms have their advantages and shortcomings. Different algorithms produce different clustering results.

Cluster ensembles combine different clustering solutions (called partitions) into a single consensus partition. Ensembling clustering solutions [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] have been successfully used in literature to improve the quality of clusterings. One of the major step in many of these algorithms is label correspondence. Most of these algorithms assume that the number of clusters in each partition is same. Experiments were performed with partitions containing equal number of clusters and this number was same as the number of the clusters implanted in the dataset. It can be shown that these algorithms may fair poorly when the number of output clusters is not known and the partitions are generated with the number of clusters way away from the actual number of clusters. In this paper, we show that the quality of the consensus partition could be improved by generating partitions with varying number of clusters, in these algorithms. Other approaches that deal with this problem include algorithms based on metaclustering and hyper graph partitioning [1].

We prove our claim through extensive experimental studies. We show that if we do not have an idea of the number k of clusters in the input data, generating partitions with varying values of k is better than at least half of the guesses one can make about k i.e. the probability with which a bad value is guessed for k is more than $1/2$. The aim of ensemble methods is not to provide clustering solution that are better than the best but rather it is to provide the user a good solution without having to worry about the input parameters like k, the number of clusters in the data.

We use two different measures to capture similarities between the clusters, Hungarian method for label correspondence and two different methods to generate the consensus. This provides us with four different algorithms. In another setting cumulative voting was used to combine the steps of label correspondence and consensus was used. Three different quality measures were used to measure the quality of the fina ensemble. We firs show our results on synthetic data sets with different number of implanted clusters. The data sets used for the purpose are $8dik$ as used in [1] where i denotes the number of implanted clusters, i was varied from 4 to 13. The k-means algorithm was used to generate the input partitions. The experiment was also performed on the real datasets of Iris and Wine [14].

In case we have some idea of the number of input clusters, we can narrow down our search. The ensemble is produced by varying k in a range close to the number of implanted clusters. The experimental results show that in this case also, results with varying k are better than 65 percent of the guesses for a fi ed k.

Main contributions of our work are :

1) Eliminates the requirement of equal number of clusters in the input partitions.
2) Eliminates the requirement of knowledge about the number of implanted clusters.

Remaining paper is organized as follows: Problem is define in section 2. Section 3 discusses the related work and the preliminaries. The setup required for the experiments is given in section 4. The experimental results are presented in section 5. The paper is concluded in section 6.

2. Problem Definition

Let X be a set of M objects and Y be a set of N samples. Let E be an $M \times N$ expression matrix. E is subjected to a clustering algorithm which delivers a clustering partition π_i consisting of k_i clusters. $\pi_i = \left(C_1^i, C_2^i, ..., C_{k_i}^i\right)$, Note that different clustering partitions may contain different number of clusters. Let $\pi_1, \pi_2, ..., \pi_H$ be H clustering partitions obtained by applying either same or different clustering algorithm(s) on E. We also denote as $\pi(X)$ the set of all possible partitions with the set of objects X.

Further, let $\lambda : E(G) \rightarrow \{1 \ldots k\}$ be a function that yields a label for each object. Let $\lambda_1, \lambda_2, ..., \lambda_H$ denote the H labellings of E. The problem of cluster ensemble is to derive a consensus function $\hat{\lambda}$, which combines the H clusterings and delivers a clustering $\hat{\pi}$ that achieves one or more of the following aims:

1) It improves the quality of the clusters.
2) It is more robust and stable than its constituent partitions.

3. Preliminaries and Related Work

The basic idea of combining different clustering solutions to obtain improved clustering has been explored under different names such as consensus clustering and evidence accumulation. Many different approaches for generating clustering solutions and combining them have been proposed in literature [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. The framework of cluster ensembles was formalized by Strehl and Ghosh [1]. A cluster ensemble method consists of two main steps as given the survey paper by Vega-Pons et al in [15] as *generation* of a set of input partitions and the integration of all the partitions to obtain a *consensus* partition.

Generation Process: There are various ways to generate the input partitions. We can generate partitions by running different clustering algorithms or by executing the same algorithm a number of times, each time with different initialization. k-means algorithm with different random initialization has been used in majority of the algorithms. The k-means clustering algorithm attempts to identify the best fit clusters by minimizing the within cluster sum of squared distance from cluster centers.

Consensus Process: Two main approaches are used to generate a consensus partition: *median partition* and *object co-occurrence*. In *median partition* approach, the consensus partition is obtained by solving an optimization problem. The median partition is defined as the partition that maximizes the similarity with all partitions in the cluster ensemble and is defined as:

$$\hat{\pi} = \arg \max_{\pi \in \pi(X)} \sum_{j=1}^{H} sim(\pi, \pi_j)$$

where $sim(\pi_i, \pi_j)$ is the similarity between two partitions π_i, π_j. The median partition problem defined with the Mirkin distance [16] has been proved to be NP-hard. Though no theoretical results are known for other similarity measures, this method is considered to be computationally expensive. Thus we do not include this approach in our study.

In the second approach, consensus partition is obtained depending upon the frequency with which two objects occur together or the frequency with which an object belongs to one cluster. One way to do this is by using *Co-association Matrix* based methods followed by some clustering algorithm and another is using *Relabelling and Voting* based methods.

All the co-association methods are based on the construction of a new similarity measure between objects whereas the Relabelling and Voting methods solve a label correspondence problem as a first step followed by a voting process to obtain the consensus partition. Different heuristics such as *bipartite matching* and *cumulative voting* have been used to solve the label correspondence problem. Experiments in these work use partition with equal number of clusters. In our study, we apply some form of these algorithms on partitions containing different number of clusters. We have used a representative of both the approaches(Hungarian algorithm [17] for bipartite matching and Andreas et al. [18] for cumulative voting) to solve the label correspondence problem. Figure 1 shows the steps involved in ensembling.

Other Consensus functions include *Graph and hyper graph algorithms [1], Information theory [3], finite mixture model [4], LAC [19], Genetic algorithms [20], NMF [21] and Kernel method [22]*. These methods either fall under object co-occurrence, which we have considered in our work or median partition.

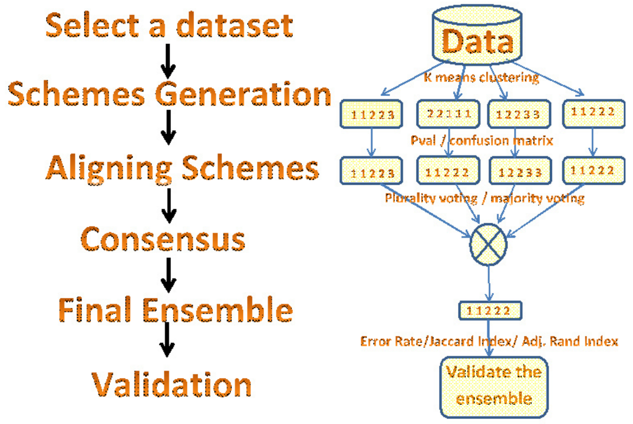

Fig. 1: Architecture showing cluster ensemble techniques using label correspondence.

4. Experimental Setup

In this section we discuss the datasets and the techniques used for our experiments.

Selecting a dataset To prove our claim we first performed experiments on synthetic supervised datasets with different number of implanted clusters. Borrowing the notation from [1], $8dik$ datasets were generated as follows: It contained i implanted clusters each consisting of 200 objects generated from 8 -dimensional Gaussian distributions Clusters have the same variance (0.1) and means were drawn from a uniform distribution within the unit hypercube.We generated 10 datasets of $8dik$, i varying from 4 to 13.

Experiments were also performed on the real datasets of Iris and Wine [14]. Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The three classes are Iris Setosa(1), Iris Versicolor(2), Iris Virginica(3). Wine dataset contains 178 objects and 3 classes. The data was generated as a result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The details of the datasets used in our study listed in Table 1.

Table 1: Details of the Datasets used.

Dataset	objects	examples	# implanted
			clusters
8dik	(200*i)	8	i
IRIS	150	4	3
Wine	178	13	3

Selecting a method to generate the partitions: To generate partitions, we used k-means algorithm as it is fast, robust and easier to understand. When data is distinct or well separated from each other k-means gives best results. The algorithm requires k, the number of output clusters, to be specified In our experiments, a different value of k is chosen randomly for each partition. This results in partitions with different number of clusters.

Selecting the similarity measure for label correspondence: In the absence of labeled training data, different partitions assign different labels to the clusters. To establish a correspondence between them one needs to solve a problem of k dimensional bipartite matching which is known to be NP-hard for $k >= 3$. To make the problem more tractable one of the clustering partition is fixed as a reference and other clusters are aligned with it. Hungarian method and cumulative voting were used for label correspondence. To align two partitions with different number of clusters the one with smaller number is assumed to have remaining clusters as empty.

Two similarity measures, confusion matrix and p-value have been used in this paper to capture the similarity between the clusters. Confusion matrix of two clusterings having k and k' clusters respectively is of size $k \times k'$. $(i,j)^{th}$ entry of the confusion matrix stores the number of objects that are in cluster i of first clustering and in cluster j of second clustering. Hungarian method for maximum weight matching is used in this case.

The measure defined by Krumpleman and Ghosh [13], is another method used to find similarity between two clusters. The p-value measures the probability of an event occurring by chance. It is defined as the total probability of seeing the observed overlap (S) or greater between the two clusters. This value essentially measures the likelihood of the observed overlap being a random event, hence a small p-value indicates a small probability of seeing the observation at random. Such pairs of clusters with smaller p-value show higher similarity so Hungarian method for minimum weight matching is used here.

The p-value between two clusters C_1 and C_2 is defined as:

$$p - value = \sum_{s=S}^{s=min(d1,d2)} P(s)$$

where

$$P(s = S) = \frac{\binom{d1}{S}\binom{M-d1}{d2-S}}{\binom{M}{d2}}$$

Here $d1$, $d2$ are the number of objects in cluster C_1 and C_2 respectively. S is the number of objects overlapping between C_1 and C_2 and M is the total number of objects in the dataset.

Selecting a method to generate the consensus: Once the partitions have been aligned, they need to be combined. We have used majority voting and plurality voting as two different methods for aggregating the results. Majority Voting involves selecting that label for an object whose count is greater than a fixed threshold whereas plurality voting works by taking the majority cluster label for each observation.

Measuring the quality of the consensus: Three different measures have been used to validate the accuracy of the consensus partition, they are Adjusted Rand Index, Jaccard Index and Error Rate.

Error Rate is the average number of misclassified objects. Lesser the error rate, more similar are the partitions. Jaccard Index is defined as $J = \frac{a}{a+b+c}$ where

- a, the number of pairs of elements that are in the same cluster in π_0 and in the same cluster in $\hat{\pi}$.
- b, the number of pairs of elements that are in the same cluster in π_0 and in different clusters in $\hat{\pi}$.
- c, the number of pairs of elements that are in different clusters in π_0 and in the same cluster in $\hat{\pi}$.

Jaccard Index disregards the pairs of elements that are in different clusters for both clusterings. Another quality measure ie Adjusted Rand Index(ARI) was considered for evaluating the clusters. ARI has become the most successful cluster validation index and is defined as:

$$ARI = \frac{\sum_{l,q}\binom{n_{lq}}{2} - \left[\sum_l \binom{n_l}{2}\sum_q \binom{n_q}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_l \binom{n_l}{2} + \sum_q \binom{n_q}{2}\right] - \left[\sum_l \binom{n_l}{2}\sum_q \binom{n_q}{2}\right]/\binom{n}{2}}$$

where

- n_{lq} denote the number of objects that are in both the l^{th} cluster of π_0, and the q^{th} cluster of $\hat{\pi}$.
- n_l the number of objects in l^{th} cluster of π_0.
- n_q the number of objects in q^{th} cluster of $\hat{\pi}$.

Adjusted Rand index and Jaccard index have large values for similar partitions. All the three measures have been used in the study to see the accuracy of the consensus formed.

5. Experimental Results

We implemented our algorithm on Intel Core i5-2430M CPU @2.40 Ghz with 4GB RAM using Windows 7 Home Basic Operating System. The code was created and executed in R version 2.15.2.

Three sets of experiments were performed for each dataset in $8dik$. In the first set, input partitions were generated for a fixed k. Thirteen such experiments were performed, one for each k varying from 3 to 15. In the second set, we performed one experiment in which input partitions were generated with different values of k ranging from 3 to 15. The third set of experiments is similar to the second one but in this set the variation in k is close to the number of clusters implanted. All

Table 2: Experimental setup

#implanted clusters	fixed k SET 1	varying k (wider range) SET 2	varying k (narrow range) SET 3
colour code	Blue	Red	Green
4	3-15	3-15	2-6
5	3-15	3-15	3-7
6	3-15	3-15	4-8
7	3-15	3-15	5-9
8	3-15	3-15	6-10
9	3-15	3-15	7-11
10	3-15	3-15	8-12
11	3-15	3-15	9-13
12	3-15	3-15	10-14
13	3-15	3-15	11-15

the three sets of experiments were performed on the datasets shown in Table 2.

In each experiment 20 partitions were generated, the experiment was iterated 20 times and the results were averaged over all iterations. For each experiment we applied 2 different similarity measures and 2 different consensus methods thereby resulting in four algorithms : $p$val-majority, $p$val-plurality, confusion-matrix-majority and confusion-matrix-plurality represented by four rows of the result graphs in figure 2- 11. The quality of consensus partition was measured using 3 different quality measures represented by 3 columns of the result graphs in figure 2- 11. Thus for every dataset 12 graphs are plotted, each showing the three sets of experiments. All the three experiments were also done using cumulative voting on all the datasets and the results are shown in figure 12- 14.

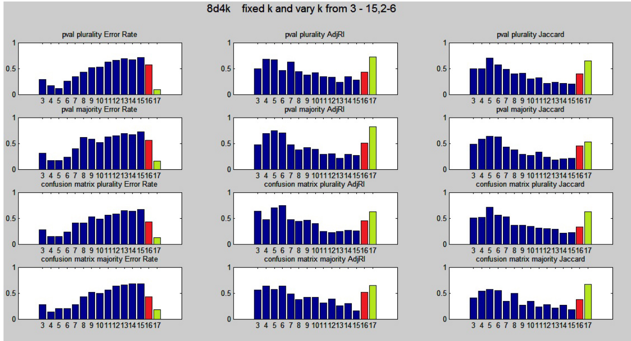

Fig. 2: fixed k versus varying k- $8d4k$

The experiments were performed on all the datasets of $8dik$ with i ranging from 4 to 13. Figures 2- 11 show the results on all the datasets $8dik$, i varying from 4- 13. Each figure corresponds to the results on one dataset. Further the blue bars show the results of the first set of experiments. A blue bar shows the ensemble produced when input partitions are generated with fixed k. Different blue bars show the results for different k's from 3-15. The result of second set of experiment is shown by the red bar. Here the partitions are all with different k varying in the range 3-15. The last bar(green in colour) shows the result of ensemble when k is varied in a range close to the number of implanted clusters.

It was observed that the ensemble produced by varying k between 3 and 15 produces better results than that produced by half of the guesses one can make for a fixed k in all the three quality measures used. In case we have some idea of the number of clusters, varying k in a range close to that provides results better than that produced by 65% of the guesses in that range. The percentage is calculated over all the datasets.

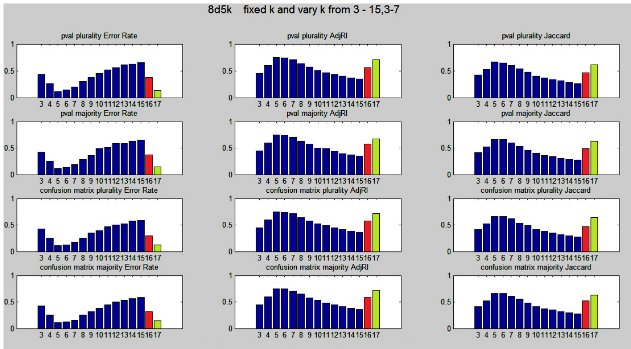

Fig. 3: fixed k versus varying k- $8d5k$

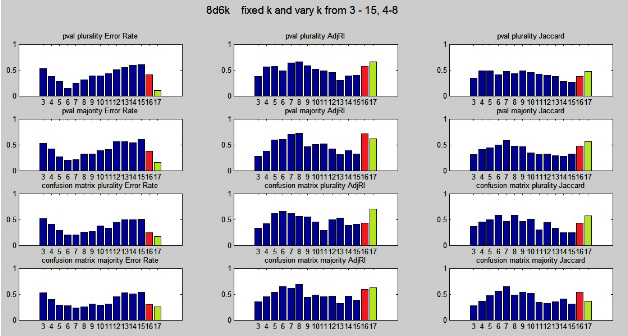

Fig. 4: fixed k versus varying k- $8d6k$

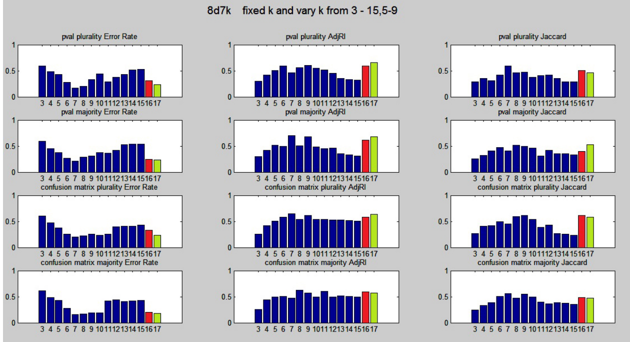

Fig. 5: fixed k versus varying k- $8d7k$

Experiments were also performed using cumulative voting on all the datasets; we present the results only on 3 of them. The figure 12- 14 show the results on the 3 datasets $8dik$, i=5, 9 and 13. It was once again found that the results produced by partitions with varying k in a wider range is better than that produced by 55% of ensembles of the input partitions with fixed k. Also for a smaller range of variation for k the enesemble is better than 70% of ensembles produced by fixed k.

Experiments were also performed on the real datasets of Iris and

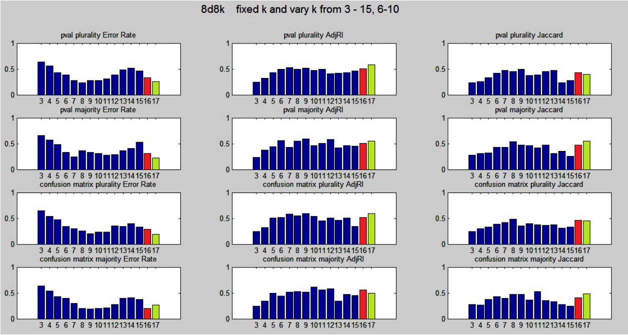

Fig. 6: fixed k versus varying k- $8d8k$

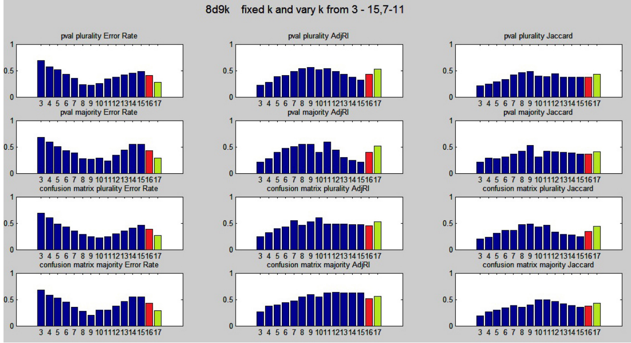

Fig. 7: fixed k versus varying k- $8d9k$

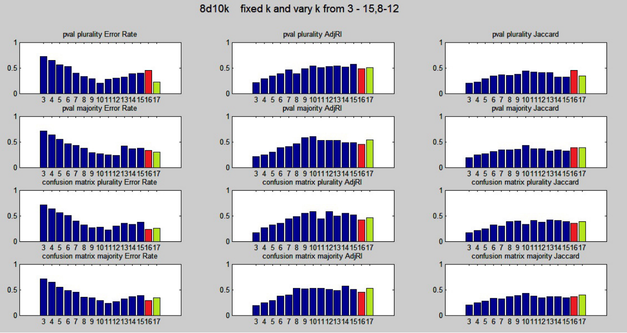

Fig. 8: fixed k versus varying k- $8d10k$

Wine [14]. Figures 15, 16 show the results of all the 12 experiments performed on these datasets. Figures 17, 18 show the results of experiments performed on these datasets using cumulative voting. It was found that on these real datasets only with 3 clusters, ensembles produced by varying k from 3 to 15 is better than at least 40% of ensembles with fixed k.

6. Conclusion

The paper focuses on the issue of guessing the number of clusters in the data. If user has no idea about that, he makes wild guesses and tries to ensemble input partitions all with the same guess. The guess may or may not be exact or close to exact. In this paper we have

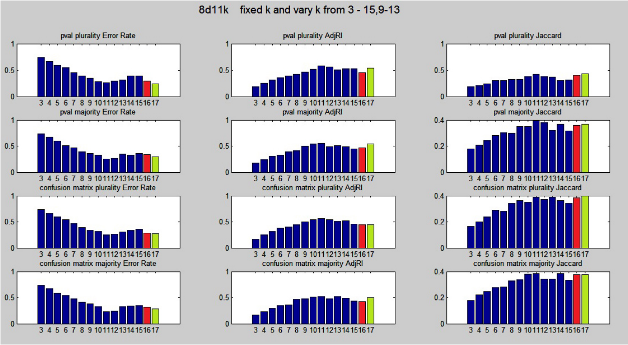

Fig. 9: fixed k versus varying k- $8d11k$

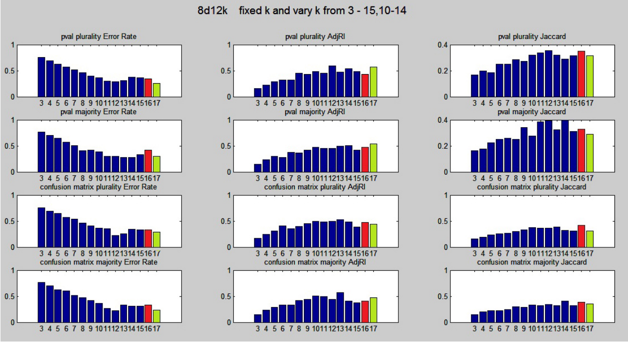

Fig. 10: fixed k versus varying k- $8d12k$

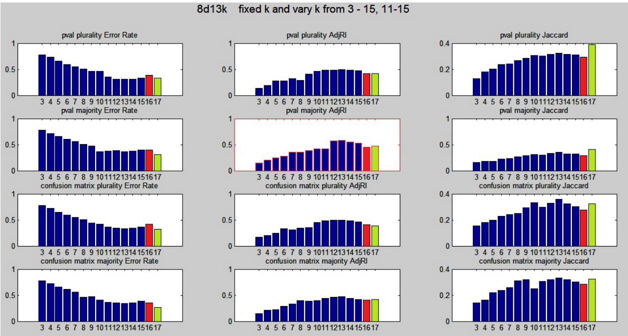

Fig. 11: fixed k versus varying k- $8d13k$

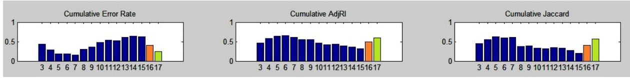

Fig. 12: fixed k versus varying k- $8d5k$

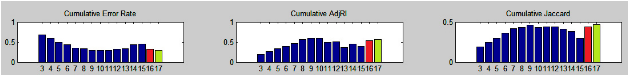

Fig. 13: fixed k versus varying k- $8d9k$

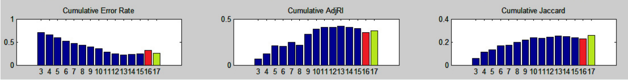

Fig. 14: fixed k versus varying k- $8d13k$

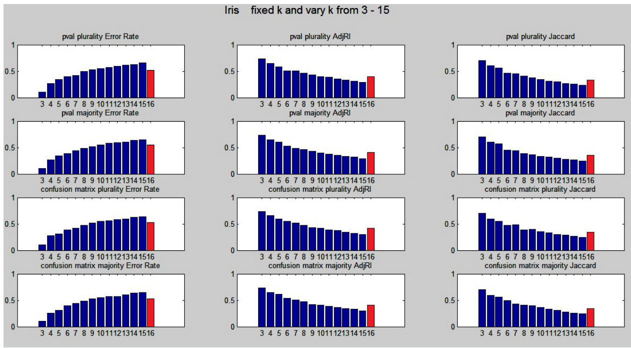

Fig. 15: fixed k versus varying k- Iris

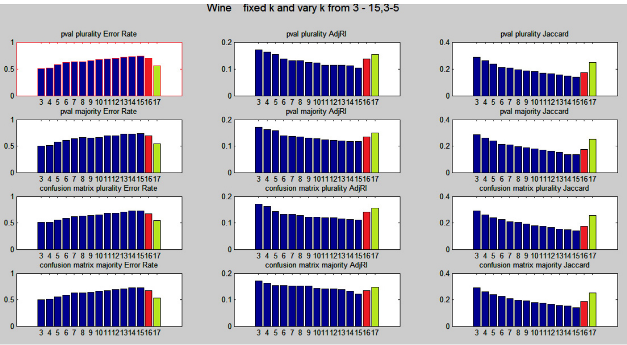

Fig. 16: fixed k versus varying k- Wine

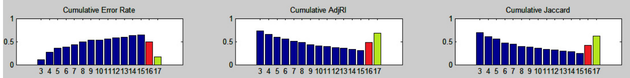

Fig. 17: fixed k versus varying k- Iris

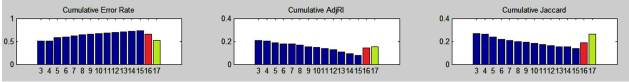

Fig. 18: fixed k versus varying k- Wine

shown that the ensemble produced by input partitions with varying number of clusters instead of all partitions with same k produces much better results in terms of error rate, adjusted rand index and jaccard index than the results produced by the guesses way away from the exact number. Moreover, if we have some idea about the number of clusters but do not know the exact number, we can narrow down the range in which we vary k thereby improving the results further.

REFERENCES

[1] Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. In: Journal on Machine Learning Research (JMLR). Volume 3., MIT Press (December 2002) 583–617

[2] Topchy, A.P., Bidgoli, B.M., Jain, A.K., Punch, W.F.: Adaptive clustering ensembles. In: ICPR. (2004) 272–275

[3] Topchy, A., Jain, A.K., Punch, W.: Combining multiple weak clusterings. In: ICDM. (2003) 331–338

[4] Topchy, A.P., Jain, A.K., Punch, W.F.: A mixture model for clustering ensembles. In Berry, M.W., Dayal, U., Kamath, C., Skillicorn, D.B., eds.: SDM, SIAM (2004)

[5] Ayad, H., Kamel, M.S.: Cumulative voting consensus method for partitions with variable number of clusters. IEEE Trans. Pattern Anal. Mach. Intell. **30**(1) (2008) 160–173

[6] Singh, V., Mukherjee, L., Peng, J., Xu, J.: Ensemble clustering using semidefinite programming with applications. Mach. Learn. **79**(1-2) (May 2010) 177–200

[7] Bhatnagar, V., Ahuja, S.: Robust clustering using discriminant analysis. In: Proceedings of the 10th industrial conference on Advances in data mining: applications and theoretical aspects. ICDM'10, Berlin, Heidelberg, Springer-Verlag (2010) 143–157

[8] Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. In: Bioinformatics. Volume 19. (2003) 1090–1099

[9] Hu, X., Yoo, I.: Cluster ensemble and its applications in gene expression analysis. In: Proceedings of the second conference on Asia-Pacific bioinformatics - Volume 29. APBC '04, Darlinghurst, Australia, Australian Computer Society, Inc. (2004) 297–302

[10] Fred, A.L.N.: Finding consistent clusters in data partitions. In: Proceedings of the Second International Workshop on Multiple Classifier Systems. MCS '01, London, UK, Springer-Verlag (2001) 309–318

[11] Fischer, B., Buhmann, J.M.: Bagging for path-based clustering. In: IEEE Trans. Pattern Anal. Mach. Intell. Volume 25., Washington, DC, USA, IEEE Computer Society (November 2003) 1411–1415

[12] Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: A cluster ensemble approach. (2003) 186–193

[13] Krumpelman, C., Ghosh, J.: Matching and visualization of multiple overlapping clusterings of microarray data. In: CIBCB'07. (2007) 121–126

[14] Bache, K., Lichman, M.: UCI machine learning repository (2013)

[15] Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. IJPRAI **25**(3) (2011) 337–372

[16] Mirkin, B.G.: Mathematical classification and clustering. Nonconvex optimization and its applications. Kluwer academic publ, Dordrecht, Boston, London (1996)

[17] Srinivasan, G.: Operations Research: Principles And Applications. Prentice-Hall of India (2002)

[18] Weingessel, A., Dimitriadou, E., Hornik, K.: Voting-merging: An ensemble method for clustering. icann. In: In Proc. Int. Conf. on Artificial Neural Networks, Springer Verlag (2001) 217–224

[19] Domeniconi, C., Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., Papadopoulos, D.: Locally adaptive metrics for clustering high dimensional data. Data Min. Knowl. Discov. **14**(1) (2007) 63–97

[20] Yoon, H.S., Ahn, S.Y., Lee, S.H., Cho, S.B., Kim, J.H.: Heterogeneous clustering ensemble method for combining different cluster results. In Li, J., Yang, Q., Tan, A.H., eds.: BioDM. Volume 3916 of Lecture Notes in Computer Science., Springer (2006) 82–92

[21] Li, T., Ding, C.H.Q., Jordan, M.I.: Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: ICDM, IEEE Computer Society (2007) 577–582

[22] Vega-Pons, S., Correa-Morris, J., Ruiz-Shulcloper, J.: Weighted cluster ensemble using a kernel consensus function. In Ruiz-Shulcloper, J., Kropatsch, W.G., eds.: CIARP. Volume 5197 of Lecture Notes in Computer Science., Springer (2008) 195–202

Geeta Aggarwal did her graduation in Computer Science from the University of Delhi, India in 1988. She did her masters in Computer Science from Banasthali Vidyapeeth in 1990. She worked as Systems Analyst in National Informatics Centre,Delhi for a short period following which she joined PGDAV College, University of Delhi in 1996. She is an associate professor and presently pursuing Ph.D. under Dr. Neelima Gupta in the fiel of Bioinformatics.

Saurabh Garg is a computer science graduate from Hansraj College, University of Delhi and is currently pursuing his MCA from the Department of Computer Sc., University of Delhi. His interests are algorithms and programming. He is an active freelance programmer for 4 years and has carried out several projects of his own.

Neelima Gupta graduated with a B.Sc. in Mathematics from the University of Delhi, India in 1985. She then went on to complete her M.Sc. in Mathematics in 1987 and M.Tech. in Computer Science in 1989 from the Indian Institute of Technology, Delhi (IITD), India. She received her Ph.D. in Computer Science from IITD in 1998, where she worked on designing randomized parallel algorithms for a number of problems in computational geometry. Earlier in her career, after her M.Tech. she briefl held the position of a Software Engineer at HCL Technologies Pvt. Ltd., in New Delhi, India in 1989. She then joined HansRaj college at University of Delhi in 1989. She is presently an associate professor in the Department of Computer Science at University of Delhi, which she joined in the year 2002. Her research interests include approximation algorithms for network design problems, networks, data mining and bioinformatics. She has published a number of papers in conferences and journals of high repute.