A New Efficient Approach in Clustering Ensembles

Javad Azimi, Monireh Abdoos and Morteza Analoui

Computer Engineering Department- Iran University of Science and Technology, Tehran, Iran
{Ja_azimi, Abdoos}@comp.iust.ac.ir, Analoui@iust.ac.ir

Abstract. Previous clustering ensemble algorithms usually use a consensus function to obtain a final partition from the outputs of the initial clustering. In this paper, we propose a new clustering ensemble method, which generates a new feature space from initial clustering outputs. Multiple runs of an initial clustering algorithm like k-means generate a new feature space, which is significantly better than pure or normalized feature space. Therefore, running a simple clustering algorithm on generated feature space can obtain the final partition significantly better than pure data. In this method, we use a modification of k-means for initial clustering runs named as "Intelligent k-means", which is especially defined for clustering ensembles. The results of the proposed method are presented using both simple k-means and intelligent k-means. Fast convergence and appropriate behavior are the most interesting points of the proposed method. Experimental results on real data sets show effectiveness of the proposed method.

Keywords: Clustering ensemble, feature space, intelligent k-means, and initial points

1 Introduction

There is no clustering algorithm performing best for all data sets. Choosing a single clustering algorithm for each data set requires both expertise and insight. Therefore, instead of clustering algorithm, a cluster ensemble can be used [1, 2]. In order to integrate clustering ensembles in a robust and stable manner, one needs a diversity of component partitions for combination that usually obtained from several sources:

1) Using different clustering algorithms to produce partitions for combination [4].
2) Changing initialization or other parameters of a clustering algorithm [3, 5].
3) Using different features via feature extraction for subsequent clustering [1, 6, 7].
4) Partitioning different subsets of the original data [8, 9, 10, 11, 12, 13].

All above introduced mechanisms try to produce more diversity by considering data from different aspects. The major hardship in clustering ensembles is consensus function and partitions combination algorithm to produce final partition, or in the other words finding a consensus partition from the output partitions of various clustering algorithms. The combination of multiple clustering can also be viewed as finding a median partition with respect to the given partitions, which is proven to be NP-complete [14].

There are many type of consensus function such as Hypergraph partitioning [1, 6], Voting approach [5, 8, 15], Quadratic Mutual Information Algorithm [16] and Co-association based functions [2, 17, 18]. In this paper, we propose a new consensus function in clustering ensembles, which is named *Labeling Algorithm*. Instead of using previous consensus functions to maintain the results of each k-means and then obtain the final partition using the consensus function, we generate a feature as a result of each k-means run and then run a simple k-means on generated features. In fact, the proposed method can be also classified as a feature extraction method with high precision. Since all of the generated features are the outputs of initial clusterings, each generated feature can classify the samples as accurate as a k-mean algorithm by itself. Therefore, running a clustering algorithm over the generated features can improve the result significantly. Although previous feature extraction methods generate the good features from pure data, each generated feature cannot classify the samples as accurate as simple k-means by itself. Most of the previous studies in clustering ensembles use k-means for initial clustering. It has been reported that the solutions obtained from the k-means are strongly dependent on the initialization of cluster centers [19, 20].

There are many methods which select the initial samples wisely [19, 21, 22] which usually studies the whole feature space to select the initial samples. Since they should study the feature space and select the initial samples using probabilistic method, increasing the time complexity in previous studies is one of the unavoidable problems. In this study, we introduce an intelligent k-means, especially defined for clustering ensembles, which selects the initial samples wisely without any increasing in time complexity. In this paper, the experimental results of the proposed method are presented by both simple k-means and intelligent k-means as initial clustering. The proposed algorithm guarantees that increasing the number of partitions does not decrease the accuracy of clustering ensembles. Sometimes, increasing the number of partitions increases the error rate of the results. Using intelligent k-means instead of simple k-means, increasing the number of partitions usually causes an improvement in results.

The rest of the paper is organized as follows. Section 2 describes the clustering ensembles. The proposed consensus function, named *Labeling Algorithm*, is presented in Section 3. In Section 4, experimental results are presented. We study the complexity of the proposed method beside other methods in Section 5. The paper is concluded in Section 6.

2 Clustering Ensembles

Clustering ensembles usually are two stage algorithms. At the first, they store the results of some independent runs of k-means or other clustering algorithms. Then, they use the specific consensus function to find a final partition from stored results. The major hardship in clustering ensembles is consensus functions and partitions combination algorithm to produce final partition, or in other words finding a consensus partition from the output partitions of various clustering algorithms. There are many types of consensus function such as:

Hypergraph partitioning: The clusters could be represented as hyper edges on a graph whose vertices correspond to the objects to be clustered. The problem of consensus clustering is then reduced to finding the minimum-cut of a hypergraph. The minimum k-cut of this hypergraph into *k* components gives the required consensus partition [1, 6]. Three hypergraph algorithms, CSPA, HGPA, and MCLA, are described in [1] and their corresponding source code are available at http://www.strehl.com.

Voting approach (re-labeling): In the other algorithms, there is no need to explicitly solve the correspondence problem between the labels of known and derived clusters. The voting approach attempts to solve the correspondence problem and then uses a majority vote to determine the final consensus partition [5, 8, 15].

Quadratic Mutual Information (QMI) (also feature-based approach): treats the output of each clustering algorithm as a categorical feature. The collection of L features can be regarded as an ‘‘intermediate feature space’’ and another clustering algorithm can be run on it. A mixture model for this case is proposed in [16].

Co-association based functions (also pair wise approach): The consensus function operates on the co-association matrix. Similarity between a pair of objects simply counts the number of clusters shared by these objects in the partitions. Numerous hierarchical agglomerative algorithms (criteria) can be applied to the co-association matrix to obtain the final partition, including Single Link (SL), Average Link (AL), Complete Link (CL) and voting k-means[2,17,18].

3 The Proposed Method

In this paper, we present a new method for clustering ensemble. The proposed method generates a new feature space using the outputs of initial clustering algorithms. The stages of the proposed clustering ensemble are as follows:

1-Using special clustering algorithms to produce initial partitions for combination.

2- Generating new features by *labeling algorithm*.

3- Running a final clustering algorithm on the new generated features.

In the proposed method, both simple k-means and intelligent k-means, introduced in section 3.1, are studied to produce partitions for combination (Stage 1).

Suppose we are given a set of N data points, $X = \{x_1,...,x_N\}$ and a set of H partitions of X return a set of labels for each point $x_i, i = 1,...,N$:

$$x_i \rightarrow \{\pi_1(x_i), \pi_2(x_i)..., \pi_H(x_i)\} \quad \textbf{(1)}$$

Where $\pi_j(x_i)$ denotes a label assigned to x_i by the j-th initial clustering. $\pi_j(x_i)$ is converted into a new value by *labeling algorithm* described in section 3.2.

If y_{ij} denotes to a new label of x_i by j-th initial clustering, we have:

$$y_{ij} = l(\pi_j(x_i)) \ , i=1,..,N, j=1,..,H \quad \textbf{(2)}$$

Where *l* is the *labeling algorithm* function introduced in Section 3.2. A new feature space is generated by Eq. (2) for each point x_i ,*i=1…,N*. Each clustering algorithm output adds a new dimension in the new feature space. Therefore, each point has *H* components in the new generated feature space, that *H* is the number of initial partitions (Stage 2). A final clustering algorithm is run on the new feature space. We use k-means for both initial and final clustering (Stage 3).

3.1 Intelligent K-means

It has been reported that the solutions obtaining from k-means are dependent on the initialization of cluster centers [19, 20]. At the first step of the k-means algorithm, we must select *k* initial samples which *k* is the number of clusters. If there are *k* real clusters, then the chance of selecting one sample from each cluster is small. The chance is relatively small when the number of clusters is large. If *k* clusters have equal samples (*n*), then the chance of selection of one sample from each cluster is:

$$p = \frac{\alpha}{\beta} = \frac{k! n^k}{(kn)^k} = \frac{k!}{k^k} \tag{3}$$

Where α , is the number of ways to select one samples from each cluster and β is the number of ways to select *k* samples.

There are many methods which select the initial samples wisely [19, 21, 22]. They studied the whole feature space to select the initial samples and so they increase the complexity.

We propose a new algorithm to refine the initial samples of k-means especially for clustering ensembles without any increasing in complexity. In clustering ensembles, the k-means is run several times. In proposed algorithm, the first execution of k-means uses random initial seed points, but for other executions, we use the previous result of k-means algorithm to select the initial seed points. The initial points for execution *i* are selected from the result of execution *i-1* of k-means algorithm. One sample from each cluster is selected at random as the initial points of next execution of k-means. Therefore, the complexity of the proposed method is $O(1)$.

3.2 Labeling Algorithm

We introduce a new algorithm generating a new feature space based on the k-means outputs with intelligent initial points. Since each generated feature is as accurate as intelligent k-means, the clustering algorithm, which is run over the generated features, is expected to be significantly more accurate than original feature space. In fact, instead of using previous consensus functions to maintain the results of each k-means and then obtain the final partition using the obtained results, we generate a feature as a result of each k-means run and then run a simple k-means on generated features.

The labeling algorithm is run after each execution of k-means algorithm. Graph theory is used in the labeling algorithm. Assume that $G = < V, E >$ denotes a graph

obtained by a clustering algorithm output. *G* is a complete weighted graph with *V* vertices and E edges. *V* and *E* represent the cluster centers and edges between every two cluster centers, respectively. The weight of each edge is the Euclidean distance between two cluster centers.

We consider an approach to generate a spanning tree, $T =< V, E' >$, for a given graph. One of the approaches to generate a spanning tree is to choose a sequence of n-1 edges, in a graph with n nodes. We proposed a greedy algorithm, similar to Prim's algorithm, to generate a spanning tree as follows:

1. Set v_0 as v_i.

2- Select v_j from *V* which $edge_{ij}$ is smaller than $edge_{ik}$ for each k and set $label(v_j) = label(v_i) + f(w(i, j))$.

3- Remove v_j from *V*.

4- Set v_j as v_i.

5- Continue from step 2 until all vertices have been selected.

The new generated tree has two nodes of degree one while the degree of the others are two. The obtained spanning tree is used for labeling the samples of each cluster. All samples whose cluster is v_j are labeled with $label(v_j)$.

The proposed labeling method has following characteristics:

1- All samples of each cluster have the same label.

2- The function *f* is a heuristic function of the edges weight.

3- Different clusters have different labels.

4- The value of α is optional.

For more details on this process, consider an example of a data set $X = \{x_1, ..., x_{20}\}$ with two features assigned to four classes as shown in Fig. 1.a.

The first step in clustering ensembles is generating ensemble members. After each generation the labeling algorithm is run on obtained results.

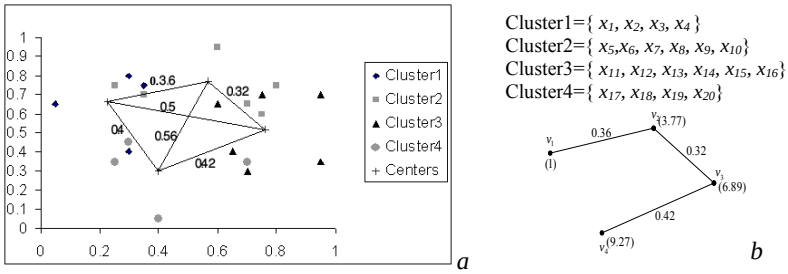

Fig. 1. a) The scatter plot of the samples **b)** Spanning and labeling of the graph

At the first we label the samples in cluster $l(v_1)$ as $1(l(v_1) = 1)$. Then the nearest cluster to v_1 is v_2. The label of the samples of cluster v_2 is set by using by $(l(v_1) + f(w(1,2)))$ that $f(w(i, j)) = 1/w(i, j)$. The nearest cluster to v_2 is v_3 which

their samples are labeled as $(l(v_2)+f(w(2,3)))$. At last, the samples of cluster 4 (v_4) are labeled as $(l(v_3)+f(w(3,4)))$ as you can see in Fig.1.b.

The cluster centers are labeled as follows:

$l(v_1)=1,$ $\quad l(v_2)=l(v_1)+f(w(1,2))=1+2.77=3.77$

$l(v_3)=l(v_2)+f(w(2,3))=3.77+3.12=6.89,$ $\quad l(v_4)=l(v_3)+f(w(3,4))=6.89+2.38=9.27$

The new feature space is generated according to $\pi_1(x_i)$. The output class and the assigned label of each sample are shown in Table 1.

Table 1. The first ensemble member result and assigned label.

x_i	Obtained class $\pi_1(x_i)$	Assigned label y_{i1}	x_i	Obtained class $\pi_1(x_i)$	Assigned label y_{i1}
x_1	1	1	x_{11}	2	3.77
x_2	4	9.27	x_{12}	3	6.89
x_3	1	1	x_{13}	3	6.89
x_4	2	3.77	x_{14}	3	6.89
x_5	1	1	x_{15}	3	6.89
x_6	2	3.77	x_{16}	4	9.27
x_7	2	3.77	x_{17}	3	6.89
x_8	2	3.77	x_{18}	4	9.27
x_9	3	6.89	x_{19}	4	9.27
x_{10}	3	6.89	x_{20}	1	1

The proposed mechanism is done after each initial clustering algorithm. Finally, the k-means algorithm is run on the new generated feature space to obtain the final partition.

4 Experiments

The experiments were performed on several data sets, including, four data sets from the UCI repository, “Iris”, “Wine”, “Soybean” and “Thyroid” data sets. A summary of data sets characteristics is shown in Table 2.

Table 2. A summary of data sets characteristics

Name	#of samples	#of features	#of classes	Samples per class
Thyroid	215	5	3	150-35-30
Iris	150	4	3	50-50-50
Wine	178	13	3	59-71-48
Soybean	47	35	4	10-10-10-17

4.1 Heuristic functions in Labeling

After finding the spanning tree, a heuristic function is used to label the clusters. In experience, we study three different functions for labeling as follows:

$$f_1(w(i,j)) = \frac{1}{w(i,j)} \quad (4)$$

$$f_2(w(i,j)) = w(i,j) \quad (5)$$

$$f_3(w(i,j)) = \frac{w(i,j)}{\arg \min_{i,j} w(i,j)} \quad (6)$$

Tables 4-7 report the mean error rate (%) of clustering combination from 50 independent runs. In tables 4-7, the mean error rate of four different available consensus functions are reported: Co-association function and Average Link (CAL), CSPA and HPGA (which are described in section 2) and the proposed method which is described in this paper with $f_3(w(i,j))$ as a heuristic functions. Parameter H represents the number of partitions and α =1.

In tables 3-6 we can see when the number of partitions is between 10 and 20, we can usually obtain the best results. The error variance value of the proposed method is usually less than the other methods; it means that the proposed method is fast convergence and *H*-independent compared with the others methods. Another fact is that the proposed method has usually an appropriate accuracy in common data sets.

Table 3. Mean error rate (%) for Iris

H	CAL	CSPA	HGPA	Proposed Method $f = f_3(w(i,j))$
5	12.7	6.38	19.81	8.36
10	9.97	5.23	7.97	4
15	7.73	4.32	5.05	4
20	6.17	4.23	4	4.89
25	5.03	4.3	4	4
30	5.57	4.3	4	4
35	5.07	4.33	4	4
40	5.53	4.27	4	4
45	5.6	4.4	4	4.87
50	5.5	4.5	4	4

Table 4. Mean error rate (%) for Soybean

H	CAL	CSPA	HGPA	Proposed method $f = f_3(w(i,j))$
5	7.02	15.74	18.51	14.04
10	7.01	13.4	15.96	6.38
15	7.52	12.55	14.57	8.09
20	6.55	13.09	14.57	9.08
25	6.88	13.19	15.21	7.94
30	6.21	14.26	15	9.86
35	4.55	14.15	14.47	5.11
40	5.21	13.94	15.11	7.8
45	4.22	13.94	15.85	5.04
50	4.51	13.51	15.85	3.48

Table 5. Mean error rate (%) for Thyroid

H	CAL	CSPA	HGPA	Proposed Method $f = f_3(w(i,j))$
5	24.3	49.35	43.77	12.25
10	20.9	49.26	39.72	12.51
15	9.86	48.6	40.47	13.32
20	7.19	48.47	38.26	13.04
25	16.7	48.88	37.47	12.85
30	15.9	48.6	38.12	13.58
35	16.0	48.84	38.4	13.22
40	16.5	49.09	37.56	13.18
45	16.6	49.14	39.87	13.04
50	15.9	48.65	36.65	13.13

Table 6. Mean error rate (%) for Wine

H	CAL	CSPA	HGPA	Proposed Method $f = f_3(w(i,j))$
5	11.7	10.65	15.98	11.87
10	13.3	9.97	10.06	9.91
15	9.19	10.03	8.57	9.49
20	11.3	10.39	7.42	8.11
25	10.6	10.37	9.04	9.97
30	10.5	10.25	8.15	9.82
35	9.97	10.53	7.39	8.69
40	10.1	10.67	8.09	9.18
45	9.65	10.51	7.84	9.4
50	9.88	10.11	7.53	9.59

The results of simple k-means and intelligent k-means with three different heuristic functions have been shown in Figs 2-5. All of the three functions have approximately

the same quality in labeling data. It is so clear that the intelligent k-means has a better behavior than simple k-means in different partition size. In addition, Figure 5.a, on soybean dataset, shows that the results of simple k-means has unexpected variations. But when we select the initial samples intelligently, the unexpected variation can not be seen (Fig. 5.b). An improvement in results is expected by increasing the number of partitions. But left side of Figs 2-5 demonstrates that increasing the number of partitions does not guarantee an improvement in final results. Sometimes, increasing the number of partitions increases the error rate of result (Simple k-means). Right side of Figs 2-5 shows that increasing the number of partitions usually improves results. Although some times increasing the number of partitions does not improve the accuracy of results, it does not decrease the accuracy.

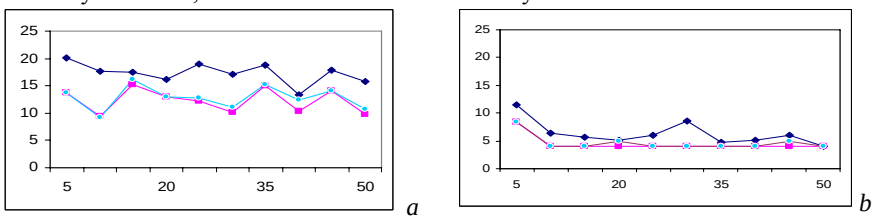

Fig. 2. The results on Iris a)with simple k-means, b) intelligent k-means (α =1).

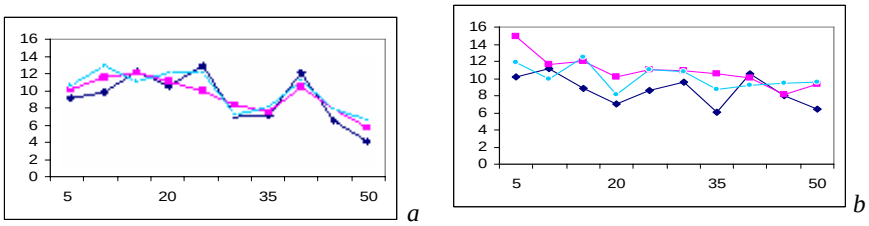

Fig. 3. The results on Wine a)with simple k-means, b) intelligent k-means (α =1).

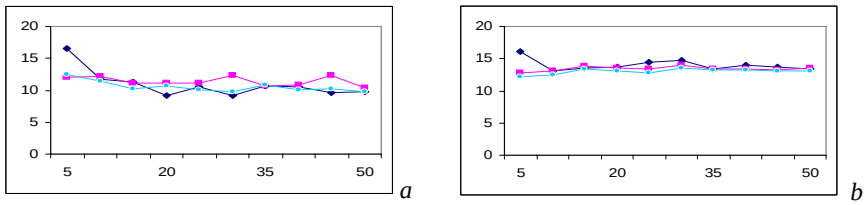

Fig. 4. The results on Thyroid a)with simple k-means, b) intelligent k-means (α =1).

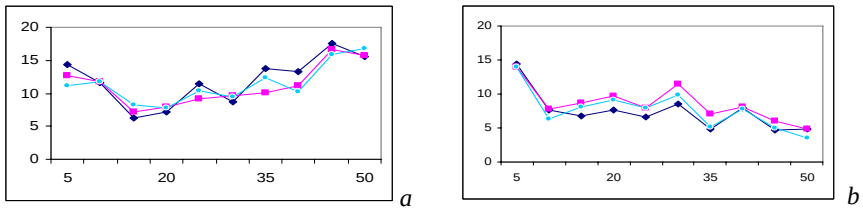

Fig. 5. The results on Soybean a)with simple k-means, b) intelligent k-means (α =1).

5 Complexities

Time complexity is one of the most important parameters in clustering ensembles algorithms. In this section, we compare the time complexity of different proposed consensus functions.

Hypergraph partitioning: Efficient heuristics to solve the *k* ways min-cut partitioning problem are known, some with computational complexity on the order of O(|ε|), where ε is the number of hyperedges.

Voting approach: all the partitions from the ensemble must be re-labeled according to a fixed reference partition. The complexity of this process is k!, which can be reduced to O(k^3) if the Hungarian method is employed for the minimal weight bipartite matching problem.

QMI: the complexity of this consensus function is O(kNB), where *B* is the number of partitions. Though the *QMI* algorithm can be potentially trapped in a local optimum, its relatively low computational complexity allows using multiple restarts in order to choose a quality consensus solution with minimum intra-cluster variance.

Co-association based functions: the computational complexity of co-association based consensus algorithms is very high, O (kN^2d^2).

The proposed method uses k-means for clustering data. The complexity of k-means is $O(kNId)$ where k is the number of clusters and N is the number of samples and I is the number of iterations of k-means to converge in each execution and d is the number of features (dimensions). Therefore, the complexity of the proposed method is $O(k!+kNIdd')$, where, d' is the number of partitions, in the other words, the number of generated features. $k!$ is the complexity time to generate spanning tree. Since k is a small number, $k!$ can be neglected. Therefore, we can see that the complexity of the proposed method is very low.

6 Conclusion

In this paper, we proposed an approach in clustering ensembles. The proposed approach generates a new feature space from the k-means outputs. Each k-means execution generates a new feature. Finally, the k-means algorithm is run on the new generated feature spaces to obtain the final partition.

The complexity of the proposed method is $O(k!+kNIdd')$ where, d' is the number of partitions, in the other words, the number of generated features. An intelligent k-means, which selects the initial samples wisely, has been proposed in this paper. The proposed selecting initial samples algorithm guaranteed that increasing the number of partitions do not decrease the accuracy of clustering ensembles. The complexity of the proposed algorithm for selecting initial points in k-means is $O(1)$. Fast convergence, the novelty and appropriate behaviors are the most interesting points of the proposed method. Using the introduced method, we can make the clustering ensembles as an incremental method, which will be very important in further studies.

References

1. Strehl, A., Ghosh, J.: cluster ensembles—a knowledge reuse framework for combining partitioning. In: Proc. of 11-th National Conf. on Artificial Intelligence, Edmonton, Alberta, Canada, 2002, pp. 93–98.
2. Fred, A.L.N., Jain, A.K.: Data Clustering Using Evidence Accumulation. Proc. of the 16th Intl. Conf. on Pattern Recognition, ICPR 2000, Quebec City, 2002, pp.276 – 280 .
3. Topchy, A., Jain, A.K., Punch, W.: Combining Multiple Weak Clustering, Proc. 3d IEEE Intl. Conf. on Data Mining, 2003, pp.331-338.
4. Hu, X., Yoo, I.: Cluster ensemble and its applications in gene expression analysis. In: Y.-P.P. Chen (Ed.). Proc. 2-nd Asia-Pacific Bioinformatics Conference, Dunedin, New Zealand, 2004, pp. 297–302 .
5. Fern, X.Z, Brodley, C.E.: Random projection for high dimensional data clustering: a cluster ensemble approach. Proc. 20th International Conference on Machine Learning, ICML, Washington, DC, 2003, pp.186–193.
6. Strehl, A., Ghosh, J.: Cluster ensembles a knowledge reuse framework for combining multiple partitions. Journal on Machine Learning Research, 2002, pp. 583-617.
7. Greene, D., Tsymbal, A., Bolshakova, N., Cunningham, P.: Ensemble clustering in medical diagnostics, In: R. Long et al. (Eds.). Proc. 17th IEEE Symp. on Computer-Based Medical Systems, 2004, pp. 576– 581.
8. Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics 19, 2003, pp.1090–1099.
9. Fischer, B., Buhmann, J.M.: Bagging for path-based clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, pp.1411–1415.
10. Fred, A.L.N., Jain, A.K.: Robust data clustering. Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, USA, 2003, vol. II, pp. 128–136.
11. Minaei, B., Topchy, A., Punch, W. F.: Ensembles of Partitions via Data Resampling. In: Proc. Intl. Conf. on Information Technology, ITCC 04, Las Vegas, 2004.
12. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: a resampling based method for class discovery and visualization of gene expression microarray data. Machine Learning 52, 2003, pp.91–118.
13. Topchy, A., Minaei-Bidgoli, B., Jain, A.K., Punch, W.: Adaptive Clustering ensembles. Proc. Intl. Conf on Pattern Recognition, ICPR'04, Cambridge, UK, 2004, pp.272-275.
14. Barthelemy, J.P., Leclerc, B.: The median procedure for partition. In: Partitioning Data Sets, AMS DIMACS Series in Discrete Mathematics, 1995, pp.3-34.
15. Weingessel, A., Dimitriadou, E., Hornik, K.: An ensemble method for clustering. Working paper, http://www.ci.tuwien.ac.at/Conferences/DSC-2003/, 2003.
16. Topchy, A., Jain, A.K., Punch, W.: A mixture model for clustering ensembles. Proceedings of SIAM Conference on Data Mining, 2004, pp.379–390.
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd Edition, John Wiley & Sons Inc. New York NY, 2001.
18. Aarts, E.H.L., Eiben, A.E., VanHee, K.M.: A general theory of genetic algorithms. Tech.Rep.89/08, Einndhoven University of Technology, 1989.
19. Bradley, P., Fayyad, U.: Refining initial points for k-means clustering. Proceedings 15th International Conf, on Machine Learning, San Francisco, CA, 1998, pp. 91-99.
20. Pena, J., Lozano, J., Larranaga, P.: An Empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters, Vol. 20, 1999, pp. 1027-1040.
21. Babu, G., Murty, M.: A near optimal initial seed value selection in k-means algorithm using a genetic algorithm. Pattern Recognition Letters Vol. 14, 1993, pp. 763-769.
22. Linde, Y., Buzo, A., Gray, R.: An algorithm for vector quantizer design. IEEE trans. Comm. Vol. 28, 1980, pp. 84-95.