Improving a Centroid-Based Clustering by Using Suitable Centroids from Another Clustering

Mohammad Rezaei

School of Computing, University of Eastern Finland, Joensuu, Finland

Abstract: Fast centroid-based clustering algorithms such as *k*-means usually converge to a local optimum. In this work, we propose a method for constructing a better clustering from two such suboptimal clustering solutions based on the fact that each suboptimal clustering has benefits regarding to including some of the correct clusters. We develop the new method COTCLUS to find two centroids from one clustering and replace them by two centroids from the other clustering so that the maximum decrease in the mean square error of the first clustering is achieved. After modifying centroids, *k*-means algorithm with few iterations is applied for fine-tuning. In an iterative algorithm, the resulting clustering is further improved using a new given clustering. The proposed method can find optimal clustering in a very small number of iterations for datasets with well-separated clusters. We demonstrate by experiments that the proposed method outperforms the selected competitive methods.

Keywords: Clustering, centroid-based, *k*-means, global optimization

1. Introduction

The goal of data clustering is to partition a set of unlabeled objects into homogeneous groups so that specific clustering criteria are optimized (Jain, Murty and Flynn, 1999). The *sum of the within-cluster variances* or *total squared error* (TSE) is the most wide used criterion in which the squared distance between each data point and its nearest cluster centroid is calculated and all of those results are added together (Likas, Vlassis and Verbeek, 2003). The *mean squared error* (MSE) is equal to TSE divided by the size of dataset. Other objective functions may be more appropriate for datasets with different cluster structures. However, they are out of the scope of this paper. For example, Min-Max *k*-means (Tzortzis and Likas, 2014) tackles the initialization problem of *k*-means by changing the objective function to be maximum within cluster variances.

K-means is the most popular clustering algorithm that minimizes the sum of the within-cluster variances. First, it randomly selects initial centroids from data points and each data point is then assigned to its nearest centroid. The centroids are updated by averaging the data points belonging to each cluster. The result of the *k*-means algorithm highly depends on the initial choice of the centroids, and the algorithm often converges to a local optimum. *Repeated k-means* is recommended by several researchers where the *k*-means algorithm is run with several different initial configurations, and the best result is taken. Steinley (2003) recommends repeating *k*-means several thousand times because the number of local optima solutions even for a small dataset (N = 200) can run into the thousands. This approach, however, would be time consuming and still does not guarantee to yield the global optimum.

Several methods exist that are able to obtain the near-optimal solution. Stochastic evolutionary approaches such as *Genetic algorithm* (Hruschka, Campello, and Freitas, 2009; Maulik and Bandyopadhyay, 2000) and *simulated annealing* (Klein and Dubes, 1989, Selim and Alsultan, 1991) take a large amount of time. Genetic algorithm starts with a population of random clustering solutions which are represented as chromosomes. Each chromosome is evaluated by the fitness function to check how good it is for the problem. A subset of chromosomes is selected, where fitter ones are more likely to be selected. Crossover and mutation operators are then applied to the selected chromosomes. The new good chromosomes are replaced with some of the weak old chromosomes. The entire process is repeated for a fixed number of iterations (Maulik and Bandyopadhyay, 2000). Simulated annealing is a sequential stochastic search that avoids local optima. To accomplish this, the algorithm with some probability allows a solution with lower quality is used in the next generation. The probability is determined by a parameter called temperature. Simulated annealing, in general, is very computationally expensive because it requires a very slow cooling rate (Klein and Dubes, 1989).

Agglomerative clustering is a bottom-up approach in which each object is initially considered as its own cluster. Two closest clusters are then iteratively merged. Among several criteria for selecting the next two clusters, *Ward's* criterion can be used to minimize the sum of squared errors (Ward, 1963). Agglomerative clustering is, in general, much slower than divisive algorithms such as *k*-means. Several methods have been proposed to speed up the process. For example, de Amorim et al. (2016) propose to start the agglomerative algorithm with a sufficiently large number of clusters instead of a partition formed by singleton clusters.

Random swap is a simple evolutionary algorithm that selects one centroid at each iteration and replaces it with a random data point. It then applies *k*-means and uses the new solution for the next iteration if a better MSE is achieved (Fränti and Kivijärvi, 2000). *J-means* performs a broader search comparing to random swap, where each centroid is relocated with each data point which does not already coincide with a centroid (Hansen and Mladenović, 2001). *Global k-means* algorithm starts with one cluster when the optimal solution is trivially known, and in an incremental way, it adds one cluster at a time. To solve the problem with k clusters, k-1 initial centroids are provided from the clustering solution for the problem with k-1 clusters. The last centroid, in N different runs, is set equal to each of N data points. *K*-means is applied with each set of centroids and the clustering with minimum MSE is considered as the solution for the problem with k clusters (Likas, Vlassis and Verbeek, 2003). *Ensemble clustering* is another approach in which the partitions from multiple clusterings are combined to infer statistically about final clusters from all partitions (Fred and Jain, 2005). All these methods, in general, have heavy computational load comparing to the standard *k*-means algorithm.

Several techniques try to improve *k*-means by intelligent selection of the initial centroids instead of random selection (Celebi, Kingravi and Vela, 2013). Steinley and Brusco compare 12 of such techniques, and conclude that their approach, which is similar to repeated *k*-means, outperforms others (Steinley and Brusco, 2007). Instead of random selection of centroids from data points, they divide the data randomly into k clusters and calculate the centroids as the initial configuration. *K-means++* is a well-known technique that aims at spreading the initial centroids over the whole range of a given dataset, and at the same time avoiding to select outliers. Specifically, it chooses the first centroid randomly from data points, and the distance of each data point x to its nearest centroid is calculated as $D(x)$. In an iterative way, the next centroid is chosen randomly from data points (excluding the previously selected ones) with probability proportional to $D^2(x)$ (Arthur and Vassilvitskii, 2007).

In this paper, we propose the novel approach *COTCLUS* to improve a centroid-based clustering of a dataset by an intelligent selection of two centroids and replacing them with two centroids from another clustering in such a way that maximum decrease in the clustering error is achieved. The new clustering can further be improved by iteratively using another clustering. The basic idea is to find a couple of badly allocated centroids from one clustering and replace them with two centroids from the other clustering so that a smaller MSE is achieved. We formulate the problem in Section 2, where we calculate the increase in MSE if a cluster is removed and the decrease in MSE if a cluster is split into two clusters. In splitting a cluster, one centroid is replaced by two corresponding centroids from the other clustering. Accordingly, the best couple of clusters for removing and splitting, which lead to maximum decrease in MSE, are selected. Section 3 provides the experimental results on a few publicly available and our own generated datasets. The datasets include a broad range of conditions including data size, dimensionality, relative density of clusters, overlap, and number of clusters. Our method demonstrates promising performance against the selected methods. Finally, conclusion and future work are given in Section 4.

2. Combining Two Centroid-Based Clusterings

K-means and many other centroid-based clustering algorithms usually fall into a local optimum. In each suboptimal clustering of a dataset with well-separated clusters, we observe that some clusters and their corresponding centroids have been correctly identified. For example, consider two clusterings of the artificially generated dataset S2 with 15 clusters in Figure 1. Without loss of generality, we use *k*-means as a centroid-based clustering algorithm to generate initial clusterings. The clusterings *A* and *B* both have two wrongly allocated centeroids. Removing incorrect centroids from *A* and replacing them by the two centroids inside the ellipse from *B* (see Figure 1) will turn *A* into a correct clustering of the dataset. Applying *k*-means with few iterations would help to fine-tune the resulting clustering. In the following, we formulate the problem, and describe our method for identifying and replacing two centroids of *A* with two centroids of *B* in order to reduce the clustering error.

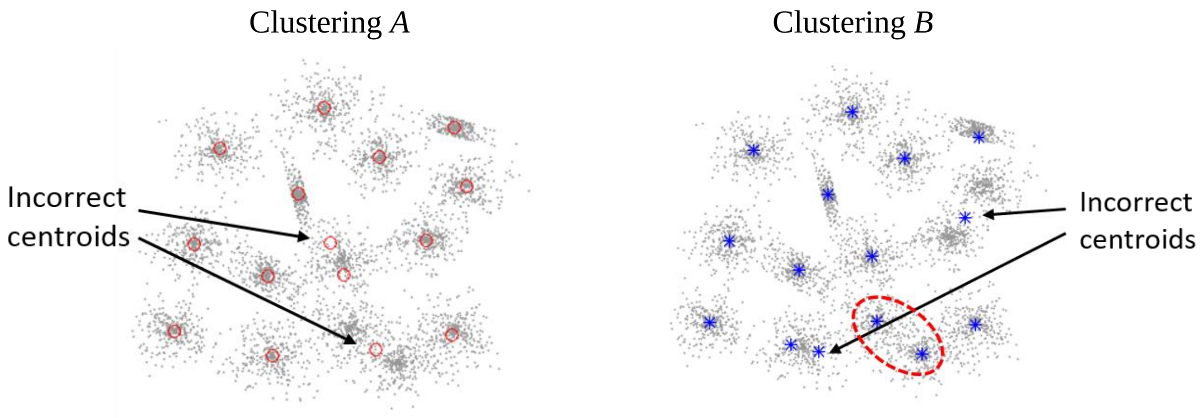


Figure 1. Applying *k*-means algorithm to S2 dataset with two different initial configurations

Let *S* be a clustering solution including *K* clusters of a set of data points $X = \{x_i\}_{i=1}^{N}$ in the *d* dimensional Euclidean space. Each cluster C_j is represented by its centroid c_j. The objective function is the typical mean square error:

$$MSE = \frac{1}{N} \sum_{j=1}^{K} \sum_{i \in C_j} \left\| x_i - c_j \right\|^2, \qquad c_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i \tag{1}$$

Given two different clusterings $S^a = \{c_j^a\}_{j=1}^{K}$ and $S^b = \{c_j^b\}_{j=1}^{K}$ with associated MSE values M^a and M^b, the goal is to improve S^a using the knowledge from S^b and build a new clustering $S^n = \{c_j^n\}_{j=1}^{K}$ with a reduced MSE, i.e., $M^n < M^a$. We found empirically that it is better to also improve S^b using S^a and take the better result. We define two actions *removing center* and *splitting cluster* for each cluster in S^a, and calculate the removal and split costs. The cost values are used for selecting the best couple of centroids for swapping that results in maximum decrease in MSE.

2.1. Removing Center (RC)

RC action removes one centroid c_k^a at a time from solution S^a and distributes the data points $\{x_i\}_{i\in C_k^a}$ to neighboring clusters $c_j^a, j \in (1,\dots,K), j \neq k$. Each data point is assigned to the cluster with the second nearest centroid, see Figure 2. Since the distance of each object to the new centroid is larger than the distance to c_k^a, MSE value increases. The resulting increase in MSE is calculated as:

$$\Delta M^{RC}(k) = \frac{1}{N} \sum_{i\in C_k^a} \left(\left\| x_i - c_{k'}^a \right\|^2 - \left\| x_i - c_k^a \right\|^2 \right) \tag{2}$$

where $c_{k'}^a$ is the second nearest centroid to data point $x_i, i \in C_k^a$.

Clustering *A*

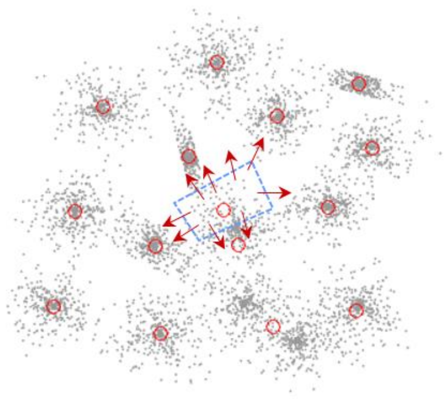


Figure 2. Removing one centroid

2.2. Splitting Cluster (SC)

SC action considers the data points associated with cluster C_k^a in S^b and locates the two clusters in S^b, with corresponding centroids c_q^b, $q = 1,2$, that own most of the data points of C_k^a. These two centroids are considered to be representative of data points in C_k^a, see Figure 3. As a result of having two centroids instead of one to represent data points in C_k^a, the MSE value for data points in C_k^a will decrease. The MSE decrease is calculated as:

$$\Delta M^{SC}(k) = \frac{1}{N} \left(\sum_{i\in C_k^a} \left(\left\| x_i - c_k^a \right\|^2 - \left\| x_i - c_{k'}^b \right\|^2 \right) \right) \tag{3}$$

where $c_{k'}^b$ (one of the two corresponding centroids to C_k^a) is the nearest centroid in the solution S^b to the data point $x_i, i \in C_k^a$. We set $\Delta M^{SC}(k) = 0$ if C_k^a has a very similar cluster $C_{k'}^b$ in S^b for which $\left| C_k^a \cap C_{k'}^b \right| / \left| C_k^a \right| > 0.95$.

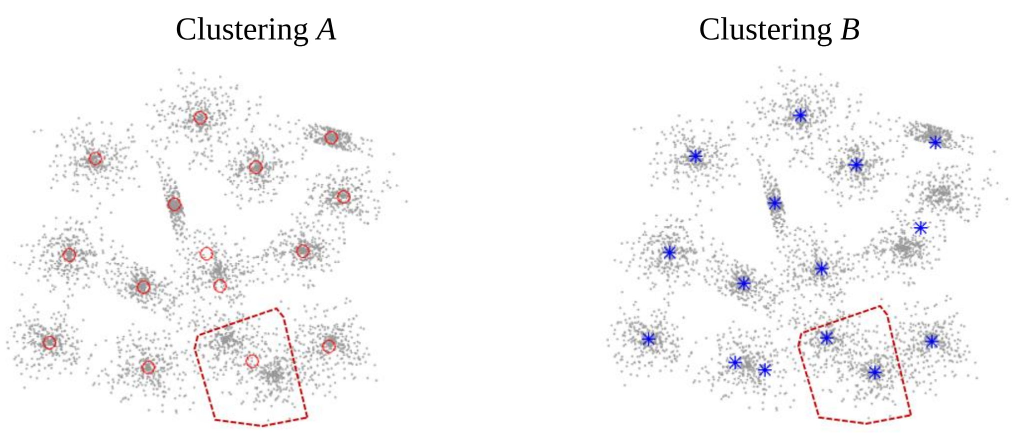


Figure 3. Splitting a cluster

2.3. Swapping

In this stage, ΔM^{RC} and ΔM^{SC} values are sorted in ascending and descending order as ΔM^{RC}_{sorted} and ΔM^{SC}_{sorted}, respectively. The pair $\Delta M^{RC}_{sorted}(1)$ and $\Delta M^{SC}_{sorted}(1)$ is the first choice for swapping. Two centroids of S^a corresponding to $\Delta M^{RC}_{sorted}(1)$ and $\Delta M^{SC}_{sorted}(1)$ are replaced by two centroids of S^b corresponding to $\Delta M^{SC}_{sorted}(1)$.

The swapping is performed only if $\Delta M^{SC}_{sorted}(1) > \Delta M^{RC}_{sorted}(1)$, or in other words, the decrease in MSE as a result of SC action is larger than the increase in MSE as a result of RC action. The same process is then repeated for the next pair in the list of ΔM^{RC}_{sorted} and ΔM^{SC}_{sorted} values. After the swapping centroids completed, *k*-means algorithm with few iterations (from 2 to 4) is applied to fine-tune the location of the new centroids. The pseudo code of the overall algorithm is as follows:

```
1.  Input:
    Data set X
    number of clusters K
    clustering solutions $S^a$ and $S^b$
2.  for each cluster $C^a_k$, k = 1 to K in $S^a$ do
3.      compute $\Delta M^{RC}(k)$ using Eq. 2
4.      find first corresponding centroid of $S^b$ to $C^a_k$ : $j_1$
5.      If $|C^b_{j_1} \cap C^a_k| / |C^a_k| < 0.95$
6.        find second corresponding centroid of $S^b$ to $C^a_k$ : $j_2$
7.        $i_1(k) = j_1$ and $i_2(k) = j_2$
8.        compute $\Delta M^{SC}(k)$ using Equation 3
9.      else
10.       $\Delta M^{SC}(k) = 0$
11. Sort $\Delta M^{RC}$ in ascending order: $idx^{RC}(i)$, i = 1..K
12. Sort $\Delta M^{SC}$ in descending order: $idx^{SC}(i)$, i = 1..K
13. i = 1
```

```
14. $k_1 = idx^{SC}(i)$ and $k_2 = idx^{RC}(i)$
15. while $\Delta M^{SC}(k_1) > \Delta M^{RC}(k_2)$ do
16.     Replace the centroids $k_1$ and $k_2$ of $S^a$ with the centroids $i_1(k_1)$ and $i_2(k_1)$ of $S^b$
17.     $i = i + 1$
18.     $k_1 = idx^{SC}(i)$ and $k_2 = idx^{RC}(i)$
19. Apply $k$-means algorithm to $S^a$
20. Output: modified clustering solution $S^a$
```

2.4. Iterating the Process

The process of combining two clusterings may be repeated for a certain number of times or until no significant improvement is obtained. Several scenarios are possible for iterating the process. A simple approach is to use two clustering at first iteration, and combine them to produce an improved clustering. At each following iteration, a new clustering is generated and combined with the resulting clustering from the previous iteration. Another approach is to use a larger number of initial solutions. At first iteration, every two clusterings are combined, and at each following iteration, the resulting clusterings from the previous iteration are combined.

We should note that the focus of this paper is on the new approach for combining two clusterings. We use the simple iterating scenario in our experiments, and generate the initial solutions using k-means algorithm. This configuration gives promising performance for datasets with separated clusters. However, finding a better iterating scenario appropriate and efficient for complicated datasets remains an open problem for future work.

3. Experimental Results

We examine the performance of COTCLUS using two sets of datasets *I* and *II*. Datasets *I* include seven publicly available datasets with known ground truth. We generated Datasets *II* with varying clustering complexity in order to arrange controllable experiments, and examine the proposed method on different data structures.

We compare COTCLUS to five different approaches for improving *k*-means including repeated *k*-means, random swap, *k*-means++, *J*-means, and genetic algorithm. We have set the number of iterations to 500 and 5000 for repeated *k*-means and random swap, respectively, and 100 for the rest of iterative methods. To make the results statistically valid, we repeat the overall process 100 times for each method, and report the average values. All the methods were implemented in MATLAB, and the experiments were performed on a 2.7-GHz Intel(R) core i5-6400 with 8.00 GB RAM.

The best MSE for each dataset is available from the provided ground truth. Therefore, for the iterative algorithms including repeated *k*-means, random swap, *J*-means, genetic algorithm, and COTCLUS, we stop the iteration as soon as the best MSE with an error less than 1% is achieved. Accordingly, the average time and the average number of iterations needed to obtain the optimal MSE are reported.

3.1. Results on Datasets *I*

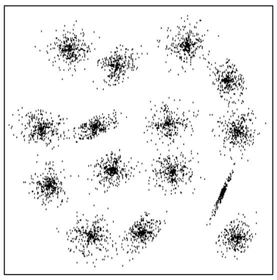
We first evaluate our proposed method on datasets *I* which include seven datasets with known ground truth summarized in Table 1, and depicted in Figure 4. They include varying cluster overlap (S sets) (Fränti and Virmajoki, 2006), large number of clusters with two different structures (Birch sets) (Zhang, Ramakrishnan, and Livny, 1997), and high-dimensional data (DIM032 set) (Fränti, Virmajoki, and Hautamaki, 2006).

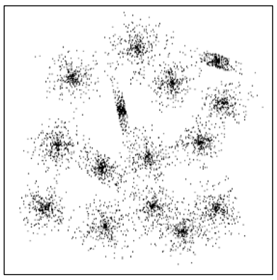
Tables 2 and 3 show the quality of clustering results measured by MSE and *adjusted Rand index* (ARI) (Hubert and Arabie, 1985, Rezaei and Fränti, 2016). The green-shaded cells in the table indicate the cases when the methods are able to find optimal clustering. Random swap, *J*-means, and COTCLUS are the methods that can achieve optimal solution for all the datasets. Genetic algorithm is not practical for Birch datasets due to large processing time. Repeated *k*-means yields a poor performance when the number of clusters or the data dimensionality is large.

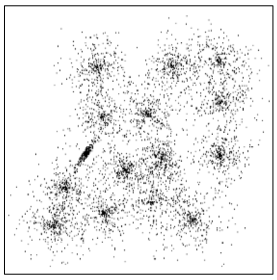
We report in Table 4, the average time and the average number of iterations needed to reach the optimal MSE. *K*-means++ is faster than COTCLUS, but it does not provide correct clustering for any of the datasets, see Table 3. For all datasets, COTCLUS provides less processing time than other iterative algorithms. However, more significant results are achieved for the high-dimensional dataset and the datasets with large number of clusters (highlighted by green-shaded cells in Table 4). COTCLUS is 16, 31, and 39 times faster than random swap for Birch1, Birch2, and DIM032 datasets, respectively.

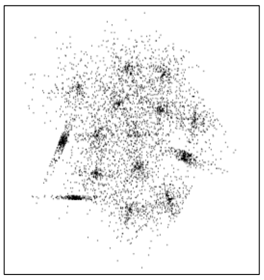
Table 1. Summary of the datasets

Dataset	Number of data points	Number of clusters	Dimension of data
S1-S4	5000	15	2
Birch1	100,000	100	2
Birch2	100,000	100	2
DIM032	1024	16	32


*S*1


*S*2


*S*3


*S*4

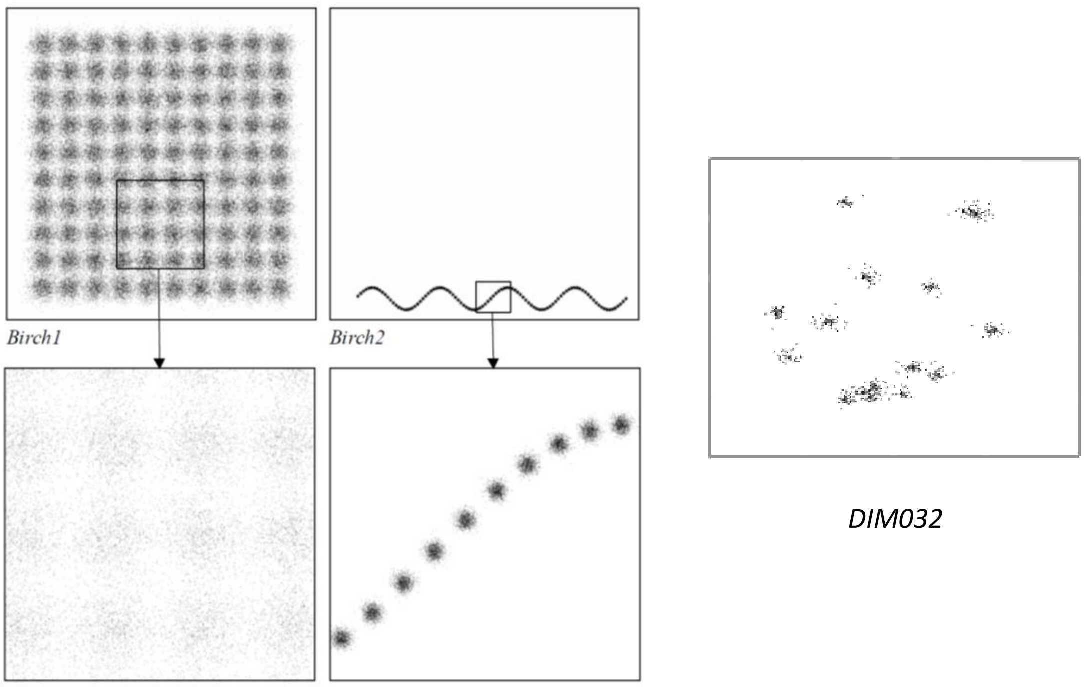


Figure 4. Example datasets

Table 2. Clustering quality of various clustering methods and different datasets, measured by TSE

Algorithm	**Dataset**						
	S_1 $\times 10^8$	*S_2* $\times 10^8$	*S_3* $\times 10^8$	*S_4* $\times 10^8$	*Birch* $_1$ $\times 10^8$	*Birch* $_2$ $\times 10^6$	dim032 $\times 1$
K-means	19.80	19.95	20.14	17.51	5.71	8.58	490.17
K-means++	16.97	19.18	19.04	17.10	5.51	5.60	136.73
Repeated KM	8.92	13.28	16.89	15.72	5.11	5.68	76.80
Genetic Algorithm	9.14	13.29	16.97	15.79	-	-	7.096
Random swap	8.92	13.28	16.89	15.72	4.64	2.28	7.096
J-means	8.92	13.28	16.89	15.72	4.64	2.28	7.096
COTCLUS	8.92	13.28	16.89	15.72	4.64	2.28	7.096

Table 3. Clustering quality of various clustering methods and different datasets, measured by ARI

Algorithm	**Dataset**						
	S_1	*S_2*	*S_3*	*S_4*	*Birch* $_1$	*Birch* $_2$	dim032
K-means	0.84	0.86	0.83	0.84	0.83	0.79	0.76
K-means++	0.87	0.87	0.87	0.86	0.85	0.86	0.93

Repeated KM	1.00	1.00	0.99	0.97	0.91	0.86	0.95
Genetic Algorithm	0.99	1.00	0.97	0.94	-	-	1.00
Random swap	1.00	1.00	0.99	0.97	1.00	1.00	1.00
J-means	1.00	1.00	0.99	0.97	1.00	1.00	1.00
COTCLUS	1.00	1.00	0.99	0.97	1.00	1.00	1.00

Table 4. Required time (sec) and number of repeats to find optimal solution for various clustering methods and different datasets

Algorithm		**Dataset**						
		S_1	S_2	S_3	S_4	*Birch* $_1$	*Birch* $_2$	dim032
K-means		0.198	0.207	0.224	0.238	0.5296	0.2916	0.176
K-means++		0.195	0.221	0.221	0.229	9.25	6.58	0.184
Repeated KM	time	7.52	2.85	2.74	1.39	544.7	298.4	69.46
	# repeats	39	14	13	6	100	100	417
Genetic Algorithm	time	100.9	13.27	17.38	20.57	-	-	68.83
	# repeats	36	12	11	6	-	-	325
Random swap	time	5.72	2.77	2.37	1.40	435.1	760.8	41.15
	# repeats	30	13	11	6	165	489	247
J-means+	time	68.31	3.84	5.81	3.24	5034	11500	29.40
	# repeats	89	5	7	3	55	112	37
COTCLUS	time	0.887	0.816	0.838	1.70	25.80	24.15	1.05
	# repeats	2	2	2	4	3	5	3

3.2. Results on Datasets *II*

The second experiment was performed on the synthetic datasets which contain spherical clusters with varying data size, number of clusters, relative cluster density, cluster overlap, and dimensionality. We selected the levels of each factor similar to the works presented in (Steinley and Brusco, 2007) and (Brusco and Steinley, 2007):

1. Size of dataset: N = 200, 1000, and 5000
2. Number of clusters: K = 4, 6, and 8
3. Dimensionality: d = 2, 4, 6, 8, and 10.
4. Relative cluster density: three types of datasets are considered including (*I*) all clusters with the same size, (*II*) one cluster includes 60% of data points, and other clusters have the same

size, and (*III*) one cluster includes 10% of data points, and other clusters have the same size.

5. Cluster overlap: three levels of overlap between clusters are considered including 0, 0.2, and 0.4. The overlap value 0.2 means that if we assign data points to their nearest centroid, 20% of the points are assigned to different clusters than the expected one.

In a series of experiments, we vary one parameter while fixing others to compare the clustering methods. The default settings of parameters are: N = 1000, K = 8, d = 2, no overlap (o = 0), and all clusters with the same size (type *I*). Tables 5 to 14 suggest the following conclusions:

(i) All methods except *k*-means and *k*-means++ are capable of finding optimal solution for all datasets and all levels of parameters if enough iterations are performed.

(ii) Regarding the time required for finding the optimal solution, the proposed method COTCLUS significantly outperforms other methods. It needs, in average, only two iterations to converge to the optimal solution. Repeated *k*-means and random swap have almost the same speed, and perform much faster than genetic algorithm and *J*-means.

(iii) Increasing data size and dimensionality does not affect much on the processing time. The reason is that the maximum data size N = 5000, and the maximum dimensionality d = 10, were not big enough to show a significant increase in the processing time. Increasing the number of clusters and overlap results in a more considerable increase in the required processing time and number of iterations to obtain the optimal solution. All evolutionary methods perform slower for relative cluster density type *II* than type *I* and *III*. However, the increase in the time for COTCLUS is much lower than other algorithms specially repeated *k*-means and random swap.

Table 5. Clustering quality for various clustering methods and datasets with different sizes

Algorithm	*N*=200		*N*=1000		*N*=5000	
	TSE	ARI	TSE	ARI	TSE	ARI
K-means	68.37	0.81	69.80	0.82	72.79	0.80
K-means++	55.58	0.85	56.37	0.86	58.53	0.85
Repeated KM	32.09	1.00	36.56	1.00	37.20	1.00
Genetic Algorithm	32.09	1.00	36.56	1.00	37.20	1.00
Random swap	32.09	1.00	36.56	1.00	37.20	1.00
J-means	32.09	1.00	36.56	1.00	37.20	1.00
COTCLUS	32.09	1.00	36.56	1.00	37.20	1.00

Table 6. Required time (sec) and number of repeats to find optimal solution for various clustering methods and datasets with different sizes

Algorithm		*N* = 200	*N* = 1000	*N* = 5000
K-means		0.173	0.172	0.191
K-means++		0.174	0.180	0.188
Repeated KM	time	1.38	1.10	1.39
	# repeats	8	6	7
Genetic algorithm	time	6.92	7.28	7.97
	# repeats	6	5	6
Random swap	time	1.47	1.09	1.50
	# repeats	8	6	8
J-means	time	1.32	5.60	5.76
	# repeats	11	5	14
COTCLUS	time	0.756	0.667	0.787
	# repeats	2	2	2

Table 7. Clustering quality for various clustering methods and datasets with different number of clusters

Algorithm	*K* = 4		*K* = 6		*K* = 8	
	TSE	ARI	TSE	ARI	TSE	ARI
K-means	56.53	0.84	78.34	0.84	69.80	0.82
K-means++	54.38	0.86	65.81	0.89	56.37	0.86
Repeated KM	37.45	1.00	39.13	1.00	36.56	1.00
Genetic Algorithm	37.45	1.00	39.13	1.00	36.56	1.00
Random swap	37.45	1.00	39.13	1.00	36.56	1.00
J-means	37.45	1.00	39.13	1.00	36.94	1.00
COTCLUS	37.45	1.00	39.13	1.00	36.56	1.00

Table 8. Required time (sec) and number of repeats to find optimal solution for various clustering methods and datasets with different number of clusters

Algorithm		$K = 4$	$K = 6$	$K = 8$
K-means		0.168	0.172	0.172
K-means++		0.172	0.174	0.180
Repeated KM	time	0.268	0.445	1.10
	# repeats	2	3	6
Genetic algorithm	time	6.67	6.79	7.28
	# repeats	1	2	5
Random swap	time	0.377	0.509	1.09
	# repeats	2	3	6
J-means	time	0.501	1.36	5.60
	# repeats	1	2	5
COTCLUS	time	0.331	0.489	0.667
	# repeats	1	1	2

Table 9. Clustering quality for various clustering methods and datasets with different dimensionalities

Algorithm	$d = 2$		$d = 4$		$d = 6$		$d = 8$		$d = 10$	
	TSE	ARI	TSE	ARI	TSE	ARI	TSE	ARI	TSE	ARI
K-means	69.80	0.82	69.33	0.80	49.36	0.82	47.18	0.81	50.60	0.79
K-means++	56.37	0.86	56.22	0.86	45.66	0.84	44.20	0.84	43.43	0.85
Repeated KM	36.56	1.00	39.90	1.00	37.29	0.99	33.62	1.00	32.43	1.00
Genetic Algorithm	36.56	1.00	39.90	1.00	37.29	0.99	33.62	1.00	32.43	1.00
Random swap	36.56	1.00	39.90	1.00	37.29	0.99	33.62	1.00	32.43	1.00
J-means	36.94	1.00	39.90	1.00	37.29	0.99	33.62	1.00	32.43	1.00
COTCLUS	36.56	1.00	39.90	1.00	37.29	0.99	33.62	1.00	32.43	1.00

Table 10. Required time (sec) and number of repeats to find optimal solution for various clustering methods and datasets with different dimensionalities

Algorithm		$d = 2$	$d = 4$	$d = 6$	$d = 8$	$d = 10$
K-means		0.172	0.167	0.174	0.171	0.168
K-means++		0.180	0.168	0.178	0.176	0.167
Repeated KM	time	1.10	0.830	0.837	1.19	1.02
	# repeats	6	5	5	7	6
Genetic algorithm	time	7.28	7.10	7.11	7.45	7.35
	# repeats	5	4	4	6	5
Random swap	time	1.09	1.01	1.16	1.22	1.27
	# repeats	6	6	6	7	7
	Time	5.60	1.82	3.21	2.46	1.78

J-means	*# repeats*	5	5	9	6	4
COTCLUS	time	0.667	0.558	0.662	0.571	0.666
	# repeats	2	1	2	1	2

Table 11. Clustering quality for various clustering methods and datasets with different relative density of clusters

Algorithm	*TYPE I*		*TYPE II*		*TYPE III*	
	TSE	ARI	TSE	ARI	TSE	ARI
K-means	69.80	0.82	52.65	0.51	76.38	0.84
K-means++	56.37	0.86	42.12	0.73	63.10	0.88
Repeated KM	36.56	1.00	27.55	1.00	37.90	1.00
Genetic Algorithm	36.56	1.00	27.55	1.00	37.90	1.00
Random swap	36.56	1.00	27.55	1.00	37.90	1.00
J-means	36.94	1.00	27.55	1.00	37.90	1.00
COTCLUS	36.56	1.00	27.55	1.00	37.90	1.00

Table 12. Required time (sec) and number of repeats to find optimal solution for various clustering methods and datasets with different relative density of clusters

Algorithm		**Relative cluster density**		
		Type *I*	Type *II*	TYPE *III*
K-means		0.172	0.178	0.175
K-means++		0.180	0.181	0.175
Repeated KM	time	1.10	8.58	1.37
	# repeats	6	50	8
Genetic algorithm	time	7.28	13.13	7.44
	# repeats	5	39	7
Random swap	time	1.09	6.21	1.61
	# repeats	6	36	10
J-means	time	5.60	14.31	110.1
	# repeats	5	13	5
COTCLUS	time	0.667	1.76	1.25
	# repeats	2	5	3

Table 13. Clustering quality for various clustering methods and datasets with different levels of overlap between clusters

Algorithm	$o = 0$		$o = 0.2$		$o = 0.4$	
	TSE	ARI	TSE	ARI	TSE	ARI
K-means	69.80	0.82	43.87	0.76	32.49	0.76
K-means++	56.37	0.86	40.34	0.78	29.67	0.80

Repeated KM	36.56	1.00	30.50	0.97	26.10	0.98
Genetic Algorithm	36.56	1.00	30.55	0.96	26.19	0.96
Random swap	36.56	1.00	30.50	0.97	26.10	0.98
J-means	36.94	1.00	30.50	0.98	26.10	0.99
COTCLUS	36.56	1.00	30.50	0.98	26.09	0.99

Table 14. Required time (sec) and number of repeats to find optimal solution for various clustering methods and datasets with different levels of overlap between clusters

Algorithm		**Overlap between clusters**		
		$o = 0$	$o = 0.2$	$o = 0.4$
K-means		0.172	0.178	0.181
K-means++		0.180	0.181	0.183
Repeated KM	time	1.10	1.18	1.44
	# repeats	6	7	8
Genetic algorithm	time	7.28	7.35	9.84
	# repeats	5	6	7
Random swap	time	1.09	1.13	1.49
	# repeats	6	6	8
J-means	time	5.60	4.78	4.99
	# repeats	5	4	4
COTCLUS	time	0.667	0.721	1.21
	# repeats	2	2	3

4. Conclusion and Future Work

We have proposed a method for combining two centroid-based clusterings of a dataset with the intent of reducing the clustering error. The process can be repeated by combining the resulting clustering with a new clustering. For only few iterations, the algorithm shows good convergence to optimal or near-optimal solution for datasets with well-separated clusters. In contrary to alternative methods such as repeated *k*-means and random swap, our experimental results show that the performance of COTCLUS remains strong for datasets with large number of clusters or high-dimensional feature space.

The main open issue that should be addressed in the future is how to develop a new iterating scenario. Currently, we combine the resulting clustering of the previous iteration with a new clustering. However, after a few iterations, the new weak clustering cannot make a significant improvement. With a better iterating scenario, the proposed method seems to work fine for datasets with overlapping clusters as well as well-separated clusters.

References

Arthur, D., & Vassilvitskii, S. (2007). *K-means++: The advantages of careful seeding.* Paper presented at the Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.

Brusco, M. J., & Steinley, D. (2007). A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning. *Psychometrika, 72*(4), 583-600.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the *k*-means clustering algorithm. *Expert Systems with Applications, 40*(1), 200-210.

de Amorim, R. C., Makarenkov, V., & Mirkin, B. (2016). A-Ward pβ: Effective hierarchical clustering using the Minkowski metric and a fast *k*-means initialisation. *Information Sciences, 370*, 343-354.

Fränti, P., & Kivijärvi, J. (2000). Randomised local search algorithm for the clustering problem. *Pattern Analysis & Applications, 3*(4), 358-369.

Fränti, P., & Virmajoki, O. (2006). Iterative shrinking method for clustering problems. *Pattern Recognition, 39*(5), 761-775.

Franti, P., Virmajoki, O., & Hautamaki, V. (2006). Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(11), 1875-1881.

Fred, A. L., & Jain, A. K. (2005). Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 835-850.

Hansen, P., & Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2): 405-413.

Hruschka, E. R., Campello, R. J., & Freitas, A. A. (2009). A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 39*(2), 133-155.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193-218.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR), 31*(3), 264-323.

Klein, R. W., & Dubes, R. C. (1989). Experiments in projection and clustering by simulated annealing. *Pattern Recognition, 22*(2), 213-220.

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global *k*-means clustering algorithm. *Pattern Recognition, 36*(2), 451-461.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition, 33*(9), 1455-1465.

Naldi, M. C., de Carvalho, A. C., & Campell, R. J. G. B. (2008). Genetic clustering for data mining *Soft computing for knowledge discovery and data mining* (pp. 113-132): Springer.

Rezaei, M., & Fränti, P. (2016). Set matching measures for external cluster validity. *IEEE Transactions on Knowledge and Data Engineering, 28*(8), 2173-2186.

Selim, S. Z., & Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition, 24*(10), 1003-1008.

Steinley, D. (2003). Local optima in *k*-means clustering: what you don't know may hurt you. *Psychological Methods, 8*(3), 294-304.

Steinley, D., & Brusco, M. J. (2007). Initializing *k*-means batch clustering: A critical evaluation of several techniques. *Journal of Classification, 24*(1), 99-121.

Tzortzis, G., & Likas, A. (2014). The MinMax *k*-means clustering algorithm. *Pattern Recognition, 47*(7), 2505-2516.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association, 58*(301), 236-244.

Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, *1*(2), 141-182.