Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An accelerated *K*-means clustering algorithm using selection and erasure rules[*]

Suiang-Shyan LEE, Ja-Chen LIN
(*Department of Computer Science, National Chiao Tung University, Taiwan 30050, Hsinchu*)
E-mail: ignoreswing.cs98g@g2.nctu.edu.tw; jclin@cs.nctu.edu.tw

Received Mar. 22, 2012; Revision accepted July 23, 2012; Crosschecked Sept. 11, 2012

Abstract: The *K*-means method is a well-known clustering algorithm with an extensive range of applications, such as biological classification, disease analysis, data mining, and image compression. However, the plain *K*-means method is not fast when the number of clusters or the number of data points becomes large. A modified *K*-means algorithm was presented by Fahim *et al.* (2006). The modified algorithm produced clusters whose mean square error was very similar to that of the plain *K*-means, but the execution time was shorter. In this study, we try to further increase its speed. There are two rules in our method: a selection rule, used to acquire a good candidate as the initial center to be checked, and an erasure rule, used to delete one or many unqualified centers each time a specified condition is satisfied. Our clustering results are identical to those of Fahim *et al.* (2006). However, our method further cuts computation time when the number of clusters increases. The mathematical reasoning used in our design is included.

Key words: *K*-means clustering, Acceleration, Vector quantization, Selection, Erasure
doi:10.1631/jzus.C1200078 **Document code:** A **CLC number:** TP301.6

1 Introduction

Clustering (Lin *et al.*, 2005; Yue *et al.*, 2005; Fahim *et al.*, 2006) is a useful technique. It has been used to solve problems in many fields, such as biological classification (Wittkop *et al.*, 2010), disease analysis (Seligson *et al.*, 2005), data mining (Crespo and Weber, 2005), image compression (Lee *et al.*, 2007), and other fields (Kong and Zhu, 2007; Wang and Tsai, 2007). One of the prominent clustering methods is the *K*-means algorithm (Fahim *et al.*, 2006; Lu *et al.*, 2008). The *K*-means algorithm partitions a given set $X=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, where each point is m-dimensional, into a set of K clusters $\{S_1, S_2, \ldots, S_K\}$ as follows:

Initialization: Choose K initial (random) points as cluster centers $C=\{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_K\}$.

Iteration: Each iteration has stages A and B. Stop the algorithm when all K cluster centers no longer change in two consecutive iterations.

Stage A: For each data point $\boldsymbol{x}_i \in X$, assign $\boldsymbol{x}_i$ to the cluster whose center is the closest to $\boldsymbol{x}_i$.

Stage B: For k=1, 2, …, K, update the cluster center as $\boldsymbol{c}_k=(\sum_{\boldsymbol{x}\in S_k}\boldsymbol{x})/|S_k|$. Here, $|S_k|$ denotes the number of elements in cluster S_k.

The *K*-means algorithm is simple and easy to implement. As mentioned by Mahajan *et al.* (2009), it can reduce the mean square error (MSE) which is defined as

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n} d^2(\boldsymbol{x}_i,\boldsymbol{c}_j),\ \ \boldsymbol{x}_i \in \text{cluster } S_j, \tag{1}$$

in which $\boldsymbol{c}_j$ is the centroid of cluster j, and d is the Euclidean distance. However, to determine the closest cluster in Stage A, the plain *K*-means algorithm will calculate $n\times K$ sets of squared Euclidean distances in the m-dimensional space, namely,

$$d^2(\boldsymbol{x}_i,\boldsymbol{c}_k)=\sum_{l=1}^{m}(x_{il}-c_{kl})^2=\|\boldsymbol{x}_i-\boldsymbol{c}_k\|^2, \tag{2}$$

[*] Project (No. 100-2221-E-009-141-MY3) supported by the National Science Council

© Zhejiang University and Springer-Verlag Berlin Heidelberg 2012

where k=1, 2, …, K and i=1, 2, ..., n. When the number of clusters (K) or the number of points (n) is large, the algorithm is not fast. A modified K-means algorithm was presented by Fahim *et al.* (2006) to increase the speed. Here, we will accelerate it further. Our clustering results are identical to those of Fahim *et al.* (2006), but our method can further reduce the computation time.

2 Review of the efficient enhanced *K*-means clustering algorithm

An enhanced K-means algorithm was presented by Fahim *et al.* (2006) to reduce the computation time when the number of clusters (K) is large. The enhancement was based on making use of results from the previous iteration. For this purpose, in each iteration of K-means, immediately after obtaining the nearest centers of the data points, Fahim *et al.* (2006) recorded two values for each data point. The first was an integer id∈{1, 2, …, K} which can be considered as a pointer, used to identify which of the K clusters is the nearest cluster. The second was the (squared) Euclidean distance between the data point and the center of this nearest cluster. For n data points, these n+n=2n values were stored and updated using two arrays (Clusterid and Pointdis) each of size n×1. Then, in the next iteration of K-means, to determine the nearest center of a data point $\boldsymbol{x}_i$, the 'updated' center of the cluster whose id was recorded at Clusterid[i], i.e., at the ith entry of Clusterid, would be checked first. If the new distance between the recorded cluster centroid and data point $\boldsymbol{x}_i$ is shorter than or equivalent to the previous distance (i.e., shorter than Pointdis[i]), then Clusterid[i] is not modified. In this situation, only one cluster center is checked, and the remaining K–1 cluster centers are ignored immediately. Thus, computation time is saved by neglecting the remaining K–1 candidates. This might make the clustering result different from that of the plain K-means method. Nevertheless, this is acceptable if at the end of the algorithm, the final result gives cluster centers whose mean square error is similar (or even superior) to the mean square error of the plain K-means.

The method of Fahim *et al.* (2006) produced clusters whose mean square error was very similar to that of the plain K-means, but the execution time was shorter.

3 The proposed method

In this section, we propose a faster algorithm. Our obtained clusters are identical to those obtained by Fahim *et al.* (2006), but the execution time is reduced further.

3.1 Overview

In general, the most time-consuming part of the K-means method is Stage A (reassigning data points to clusters). For a data set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, …, \boldsymbol{x}_n\}$, the time complexity of an iteration is $O(nK)$ if the plain K-means method with full search is used. To many researchers it is of interest to design a more efficient way to locate the nearest cluster center of each data point, so that each data point can be quickly reassigned to a cluster.

To find the nearest cluster center of a data point $\boldsymbol{x}_i$, we need to determine quickly the potential qualification of a to-be-checked candidate cluster center. Thus, we derive a selection rule to judge roughly the to-be-checked center. The to-be-checked center is selected according to an organized (non-random) rule in our algorithm. An erasure rule is also used. Whenever the condition of the erasure rule is satisfied, one or more centers are erased simultaneously. Hence, the erasure rule can quickly find some more qualified clusters in a few tries. Since there are two rules in our method, namely selection and erasure, we call our method the selection-and-erasure (S&E) K-means method.

3.2 Faster assignment of data to clusters

3.2.1 Pre-processing

For each iteration, a norm-sorting pre-processing is executed before activating the algorithm. For all cluster centers $\boldsymbol{c}_k \in C$, compute the Euclidean norm $\|\boldsymbol{c}_k\| = \sqrt{\sum_{l=1}^{m} c_{kl}^{\,2}}$. Sort the norm values of $\{\|\boldsymbol{c}_k\|\}_{k=1,2,\dots,K}$ to obtain an ascending-order sequence $R=\{\boldsymbol{c}_j\}_{j=1,2,\dots,K}$, where $\|\boldsymbol{c}_1\| \le \|\boldsymbol{c}_2\| \le \dots \le \|\boldsymbol{c}_K\|$. This creates a new index sequence j=1, 2, …, K which has monotonic norm-order implied implicitly. Record the mapping from set C to sequence R, i.e., from k to j.

3.2.2 The function Select_and_Erase()

For each data point $\boldsymbol{x}_i$, to find its nearest cluster, we run the pseudo-code listed in Algorithm 1. Thus, Algorithm 1 is Stage A (assigning data to the nearest cluster) of our algorithm. Note that in line 11, we need to check whether the condition

$$(\|\boldsymbol{c}_j\| - \|\boldsymbol{x}_i\|)^2 \geq d_{\min} \tag{3}$$

is satisfied. Therefore, we recommend that the computed norm values $\{\|\boldsymbol{x}_i\|\}_{i=1,2,\ldots,n}$ should be preserved, for these values are used repeatedly in our method.

Algorithm 1 Function Select_and_Erase()

```
1  for $i$=1 to $n$
2    Compute squared Euclidean distance $d^2(\boldsymbol{x}_i$, Clusterid[$i$]);
3    if ($d^2(\boldsymbol{x}_i$, Clusterid[$i$])≤Pointdis[$i$])
4      then the point $\boldsymbol{x}_i$ stays in its original cluster (i.e.,
         no reassignment of $\boldsymbol{x}_i$);
5    else
6      Use binary search to locate a cluster center whose
         $\|\boldsymbol{c}_j\|$ is the closest to $\|\boldsymbol{x}_i\|$;
7      $d_{\min}=d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$; $\boldsymbol{c}_{\min}=\boldsymbol{c}_j$;
8      Delete this center $\boldsymbol{c}_j$ from $R$;
9      while ($R$ is not empty)
10       Choose from $R$ a to-be-checked center $\boldsymbol{c}_j$ whose
           $\|\boldsymbol{c}_j\|$ is the closest to $\|\boldsymbol{x}_i\|$;
11       if $(\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$
12         if $\|\boldsymbol{c}_j\| \geq \|\boldsymbol{x}_i\|$
13           Delete all of the centers whose indices in $R$
               are larger than $j$;
14         if $\|\boldsymbol{c}_j\| \leq \|\boldsymbol{x}_i\|$
15           Delete all of the centers whose indices in $R$
               are less than $j$;
16       else
17         Compute squared Euclidean distance $d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$;
18         if $d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)<d_{\min}$
19           $d_{\min}=d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$ and $\boldsymbol{c}_{\min}=\boldsymbol{c}_j$;
20         else
21           $d_{\min}=d_{\min}$ and $\boldsymbol{c}_{\min}=\boldsymbol{c}_{\min}$;
22     end while
23   Clusterid[$i$]=Cluster id of $\boldsymbol{c}_{\min}$;
24   Pointdis[$i$]=$d_{\min}$;
25 end for
```

3.2.3 Overlapped *K*-means with S&E and enhanced *K*-means with S&E

Fahim *et al.* (2006) designed two accelerated versions of *K*-means, namely overlapped *K*-means and enhanced *K*-means. Therefore, our new method was implemented in two versions: (1) overlapped *K*-means with S&E; (2) enhanced *K*-means with S&E. Fahim *et al.* (2006) found that enhanced *K*-means was faster than overlapped *K*-means. The reason was that the overlapped *K*-means inspected all possible clusters in odd cycles, but the enhanced version did not have such an adjustment. They concluded that the enhanced *K*-means algorithm was the one they recommended among the three algorithms they tested (i.e., the plain, the overlapped, and the enhanced *K*-means). The clustering quality of the enhanced *K*-means was similar to that of the others but the execution time of enhanced *K*-means was the shortest.

In each iteration, when we assign n data points to the K temporary clusters, our overlapped *K*-means with S&E will run Select_and_Erase() in each even-iteration, and run Select_and_Erase() without lines 2–5 in each odd-iteration. As for the enhanced *K*-means with S&E, this version will run Select_and_Erase() without lines 2–5 in the first two iterations, and then execute Select_and_Erase() for the remaining iterations.

3.3 Mathematical reasoning

3.3.1 Erasure rule

The condition used in the erasure rule is $(\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$. This inequality can erase the center $\boldsymbol{c}_j$ because

$$\begin{aligned} d^2(\boldsymbol{x}_i, \boldsymbol{c}_j) &= \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2 \\ &= \|\boldsymbol{x}_i\|^2 + \|\boldsymbol{c}_j\|^2 - 2\boldsymbol{x}_i \cdot \boldsymbol{c}_j \\ &\geq \|\boldsymbol{x}_i\|^2 + \|\boldsymbol{c}_j\|^2 - 2\|\boldsymbol{x}_i\|\|\boldsymbol{c}_j\| \\ &= (\|\boldsymbol{c}_j\| - \|\boldsymbol{x}_i\|)^2. \end{aligned} \tag{4}$$

Thus, we can conclude that if $(\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$, then $d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$ must be greater than or equal to $d_{\min}$. In addition, because the centers' norm value has been sorted already, we find that $d^2(\boldsymbol{x}_i, \boldsymbol{c}_{j+\sigma}) \geq (\|\boldsymbol{c}_{j+\sigma}\|-\|\boldsymbol{x}_i\|)^2 \geq (\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$ when $\|\boldsymbol{c}_{j+\sigma}\| \geq \|\boldsymbol{c}_j\| \geq \|\boldsymbol{x}_i\|$. The case of $\|\boldsymbol{c}_j\| \leq \|\boldsymbol{x}_i\|$ can be analyzed in an analogous manner.

3.3.2 Selection rule

The selection rule always picks the center $\boldsymbol{c}_j$ whose $\|\boldsymbol{c}_j\|$ is the closest to $\|\boldsymbol{x}_i\|$ as the one we try first. The reason is very simple. In any version of *K*-means, the objective of reassigning a data point $\boldsymbol{x}_i$ to a cluster is to minimize the distance from $\boldsymbol{x}_i$ to that center. We use the erasure rule $(\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$ in the proposed algorithm. Hence, if we can obtain a very small $d_{\min}$ in the very beginning of the while loop, then $(\|\boldsymbol{c}_j\|-\|\boldsymbol{x}_i\|)^2 \geq d_{\min}$ is easy to satisfy for the remaining runs of the while loop. Hence, lines 11–15 of Algorithm 1 can eliminate many candidate centers immediately. Therefore, it is important to make $d_{\min}$ very small when we pick from the set R the first center to be checked. To make $d_{\min}$ very small, since $d_{\min}=d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$ when the first $\boldsymbol{c}_j$ is picked, we want $\boldsymbol{x}_i$ to be very close to $\boldsymbol{c}_j$. Therefore, their norms $\|\boldsymbol{c}_j\|$ and $\|\boldsymbol{x}_i\|$ cannot be too far away from each other. So, $\|\boldsymbol{c}_j\|$ and $\|\boldsymbol{x}_i\|$ should be very close.

4 Experiments and comparisons

4.1 Experiments

4.1.1 Data sets

Four experiments were conducted to evaluate our method. The input of Experiment 1 was the letter recognition data set from the UCI Machine Learning Repository (Frank and Asuncion, 2010). It consists of 20 000 samples. Each sample point has 16 integer attributes ranging from 0 to 15. There are 26 classes and each class is one of the capital letters in the English alphabet. Each class has several commercial fonts. The original purpose of this data set was to identify the alphabetical letters. From the viewpoint of letter recognition, this data set is too hard for algorithms of the K-means type (unless post-processing such as split and merge is used (Lin, 1996)). However, we can still evaluate the acceleration gained by our method.

Experiment 2 was related to codebook generation in vector quantization (VQ) of images (Chen *et al.*, 2009). A total of 98 304 blocks were partitioned into K clusters. The input 98 304 blocks were generated as follows: six 512×512 images {Lena, Peppers, Baboon, Jet, Boat, Tank} were divided into 4×4 blocks. We thus had 6×(512×512)/(4×4)=98 304 blocks. First, we used K-means to train the (512×512)/(4×4) blocks of Lena to create a codebook of K vectors (each vector was a 4×4 block). The K vectors were then used as the K initial blocks of the K-means to partition the 98 304 blocks into K clusters. In terminology of VQ, Experiment 2 created a global codebook (for six images) by using Lena's local codebook as the initial try. Notably, when we used K-means to train the (512×512)/(4×4) blocks of Lena, the initials of the K-means method were selected by running the CURE algorithm described below. When we ran the CURE algorithm 30 times, we obtained an initial set each time for K-means and thereby created a local codebook of Lena, which in turn created a global codebook of the six images. Therefore, there were 30 global codebooks of the six images. The MSE and execution times shown are both the averages of the 30 tests.

The number of clusters was always a whole power of 2 for the data in Experiment 2 because it is used for communication purposes. In other words, after running K-means, each image block is compressed as a VQ index containing $\log_2 K$ bits. For example, if each image block is compressed as a 5-bit VQ-index, then there should be $2^5=32$ VQ codewords, and each codeword is the representative block (the center) of a cluster containing multiple image blocks similar to this representative block. Therefore, $2^5=32$ VQ codewords imply 32 clusters. An analogous argument shows that a 6-bit VQ-index implies $2^6=64$ clusters; a 7-bit VQ-index implies $2^7=128$ clusters; etc.

Experiment 3 used the Statlog (Landsat satellite) data set (Frank and Asuncion, 2010). The data were purchased from NASA by the Australian Centre for Remote Sensing. The Statlog data set consists of 6435 data points, and each data point has 36 attributes. The data have seven classes (red soil, cotton crop, grey soil, damp grey soil, etc.), and the data have been cited more than one hundred times by researchers, e.g., Leng and Hong (2010).

Experiment 4 used an artificial Gaussian data set in the 3D space. We created 12 800 data points using the following rules: (1) Locate eight positions (±7, ±7, ±7) in the 3D space; (2) For each 3D position (u, v, w), create eight Gaussian centers $(u\pm2, v\pm1, w\pm0.5)$. (3) For each of the 8×8=64 Gaussian centers, generate 200 points around the center using a Gaussian distribution whose standard deviation is 0.25. Notably, when this data set was designed, there were eight major groups which were far away from each other. Each group contained two subgroups which were close to each other. Then each subgroup had two branches which were considerably closer to each other, and each branch had two Gaussian clusters that almost touched each other. Therefore, the data set can be regarded as 8, 16, 32, or 64 classes.

4.1.2 Initial points

The configuration of initial centers has an impact on determining the results of the K-means algorithm.

To obtain reasonable and overall consideration, we obtain K initial points by running an algorithm called the clustering using representatives (CURE) algorithm in pattern recognition textbook (Theodoridis and Koutroumbas, 2009). The following are the steps for obtaining a K-point representative set R_C from a given set C:

Step 1: Select a vector $\boldsymbol{x}_1 \in C$, with the maximum distance from the mean of C. Let $R_C=\{\boldsymbol{x}_1\}$.

Step 2: For i=2 to K, pick an $\boldsymbol{x}_i \in C-R_C$ that maximizes the closest-distance to points of R_C. Then, $R_C=R_C \cup \{\boldsymbol{x}_i\}$.

Step 3: Shrink all points $\boldsymbol{x} \in R_C$ toward the mean $\boldsymbol{M}_C$ of C by a factor a. That is, $\boldsymbol{x}=(1-a)\boldsymbol{x}+a\boldsymbol{M}_C$.

The initial centers in our experiments were obtained using the CURE algorithm with a from 0.05 to 0.7. In other words, we ran CURE several times using distinct values of a. So, we obtained multiple sets of initials and each initial set had K points. Then, for each of the five clustering methods being compared, we ran the method using each set of initials and took the average time and the average MSE. This should have reduced the influence of the initials because it was a multi-run test and each test used the representatives which roughly represented the data set.

4.2 Comparisons

We compared the results of the five algorithms: the plain K-means algorithm, the overlapped and enhanced algorithms of Fahim *et al*. (2006), our overlapped algorithm with S&E, and our enhanced algorithm with S&E. All five algorithms used the same initial centers in the test.

To determine the quality of the K centers found by each algorithm, when an algorithm stopped, we measured the MSE value defined in Eq. (1). We also compared the CPU time used by each algorithm. Because the number of executed iterations might differ among these algorithms, we compared not only the average execution time per iteration, but also the total execution time. Comparisons of MSE values (Tables 1 and 2: results from Experiments 3 and 4 not shown) showed that: (1) In Experiments 1–4, the MSE values were always very similar among the different methods. (2) The MSE of the overlapped algorithm was always equal to that of our overlapped algorithm with S&E. Likewise, the enhanced algorithm and our enhanced algorithm with S&E always had the same MSE value. (3) In Experiments 1–4, the benefit of our total execution time over those of the other methods became obvious when the number of clusters became larger. This statement is true for both the total execution time and the average time per iteration.

The details of the comparisons are as the following.

Table 1 Mean square errors (MSE) for the five algorithms in Experiment 1

Number of clusters, K	MSE				
	Plain K-means	Overlapped algorithm	Overlapped with S&E	Enhanced algorithm	Enhanced with S&E
26	30.9341	30.9000	30.9000	30.9792	30.9792
52	23.8053	23.7465	23.7465	23.8402	23.8402
78	20.1564	20.1778	20.1778	20.2296	20.2296
104	17.9715	17.9679	17.9679	18.0867	18.0867
130	16.3066	16.3408	16.3408	16.4666	16.4666

Underlined numbers show that the plain K-means algorithm does not always have the best MSE value

Table 2 Mean square errors (MSE) for the five algorithms in Experiment 2

Number of clusters, K	MSE				
	Plain K-means	Overlapped algorithm	Overlapped with S&E	Enhanced algorithm	Enhanced with S&E
64	2347.89	2346.95	2346.95	2349.75	2349.75
128	1959.51	1963.45	1963.45	1970.75	1970.75
256	1656.31	1657.05	1657.05	1664.92	1664.92
512	1411.41	1413.37	1413.37	1423.35	1423.35
1024	1220.91	1220.18	1220.18	1242.68	1242.68

Underlined numbers show that the plain K-means algorithm does not always have the best MSE value

1. Letter recognition data set

According to Table 1, for a given number (K) of clusters with the same initial centers, all five algorithms achieved a similar quality of clustering results. The MSE of the overlapped algorithm was equal to that of the overlapped algorithm with S&E. Likewise, the enhanced algorithm and the enhanced algorithm with S&E had the same MSE value. Note that the plain K-means did not always have the best MSE value (see K=26, 52, and 104 in Table 1), since it is well-known that K-means also cannot ensure a global minimum. Fig. 1 shows the total execution time for the five algorithms and Fig. 2 shows the average time per iteration. Our two algorithms used less time compared with the corresponding methods. In summary, the clustering quality was the same as that of Fahim *et al.* (2006) but the required execution was improved.

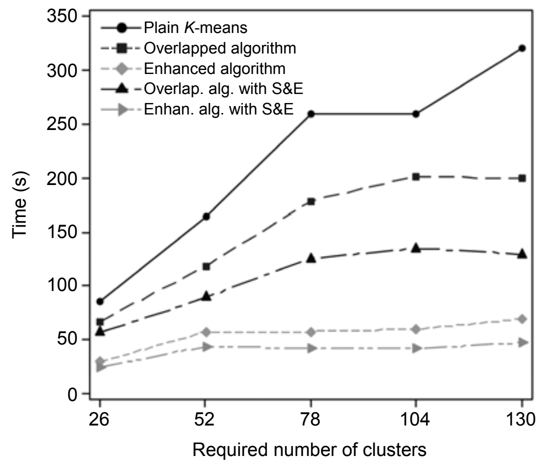

Fig. 1 Total execution time for Experiment 1

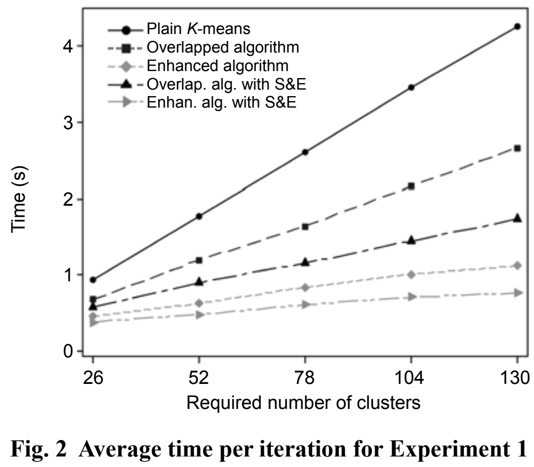

Fig. 2 Average time per iteration for Experiment 1

2. Vector quantization of images

Table 2 lists the MSE values for the five algorithms. The plain K-means did not always have the best MSE value (e.g., K=64 and K=1024). Fig. 3 shows the total execution time and Fig. 4 shows the average execution time per iteration. The curves indicate that the computation time of our algorithm increased slowly even if the number of clusters grew quickly.

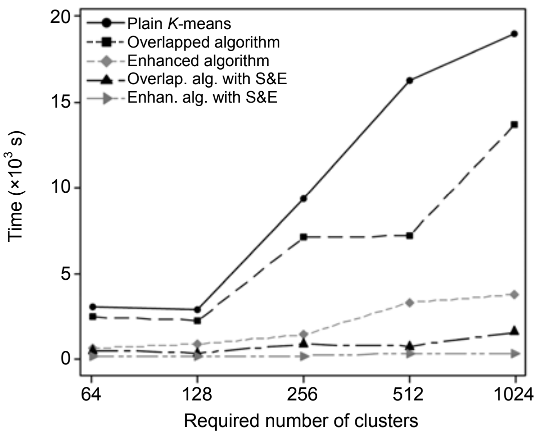

Fig. 3 Total execution time for Experiment 2

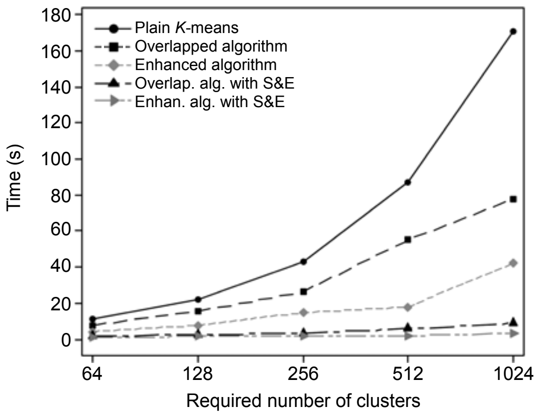

Fig. 4 Average time per iteration for Experiment 2

3. Statlog (Landsat Satellite) data set

Fig. 5 displays the curves of total execution time and Fig. 6 the curves of average execution time per iteration. Again, our methods improved the processing efficiency.

4. Artificial Gaussian data set

Figs. 7 and 8 show the execution time for Experiment 4. The actual number of clusters was 64 for this data set. Again, the benefit of our execution time over those of the other methods became apparent when the number of clusters increased.

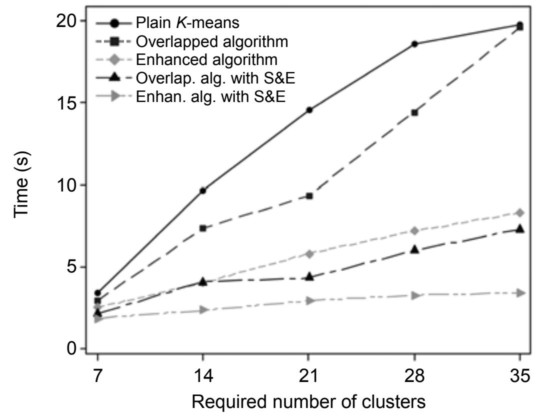

Fig. 5 Total execution time for Experiment 3

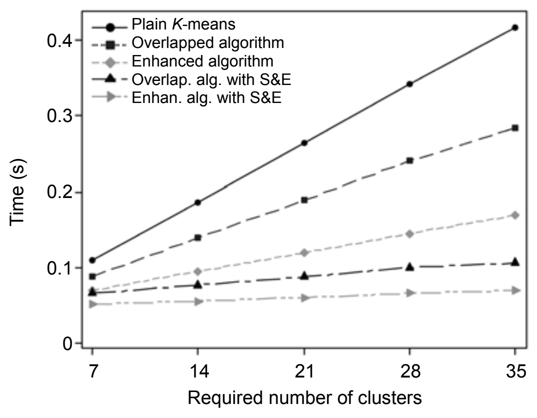

Fig. 6 Average time per iteration for Experiment 3

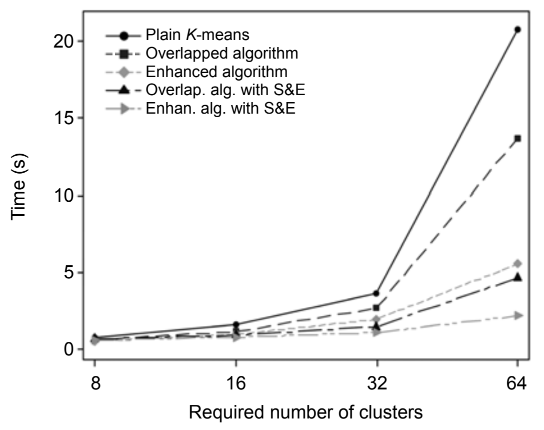

Fig. 7 Total execution time for Experiment 4

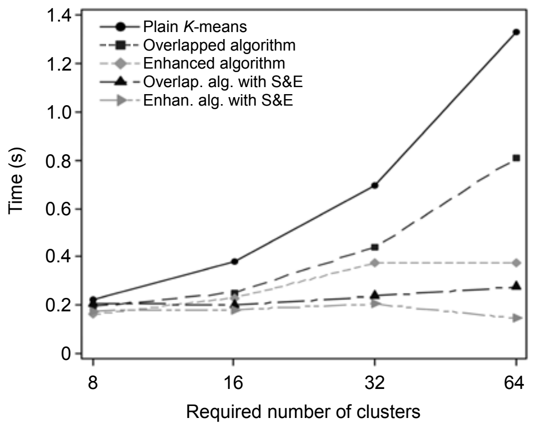

Fig. 8 Average time per iteration for Experiment 4

5 Conclusions

In this paper, we present a selection-and-erasure (S&E) *K*-means algorithm. Our clustering results were exactly the same as those of the method used by Fahim *et al*. (2006). However, we increased the efficiency of the method when the number of clusters became large. Experimental results showed that our algorithm works well.

References

Chen, L.S.T., Su, W.K., Lin, J.C., 2009. Secret image sharing based on vector quantization. *Int. J. Circ. Syst. Signal Process.*, **3**(3):137-144.

Crespo, F., Weber, R., 2005. A methodology for dynamic data mining based on fuzzy clustering. *Fuzzy Sets Syst.*, **150**(2): 267-284. [doi:10.1016/j.fss.2004.03.028]

Fahim, A.M., Salem, A.M., Torkey, F.A., Ramadan, M.A., 2006. An efficient enhanced *k*-means clustering algorithm. *J. Zhejiang Univ.-Sci. A*, **7**(10):1626-1633. [doi:10.1631/jzus.2006.A1626]

Frank, A., Asuncion, A., 2010. UCI Machine Learning Repository. Schools of Information and Computer Science, University of California, Irvine, CA. Available from http://archive.ics.uci.edu/ml [Accessed on July 19, 2012].

Kong, W.Z., Zhu, S.A., 2007. Multi-face detection based on downsampling and modified subtractive clustering for color images. *J. Zhejiang Univ.-Sci. A*, **8**(1):72-78. [doi:10.1631/jzus.2007.A0072]

Lee, W.J., Chung, J.S., Ouyang, C.S., Lee, S.J., 2007. Vector quantization of images using a fuzzy clustering method. *Cybern. Syst.*, **39**(1):45-60. [doi:10.1080/01969720701710139]

Leng, J., Hong, T.P., 2010. Mining outliers in correlated subspaces for high dimensional data sets. *Fundam. Inform.*, **98**(1):71-86. [doi:10.3233/FI-2010-217]

Lin, H.J., Yan, F.W., Kao, Y.T., 2005. An efficient GA-based clustering technique. *Tamkang J. Sci. Eng.*, **8**(2):113-122.

Lin, J.C., 1996. Multi-class clustering by analytical two-class formulas. *Int. J. Pattern Recogn. Artif. Intell.*, **10**(4):307-323. [doi:10.1142/S0218001496000220]

Lu, J.F., Tang, J.B., Tang, Z.M., Yang, J.Y., 2008. Hierarchical initialization approach for *K*-means clustering. *Pattern Recogn. Lett.*, **29**(6):787-795. [doi:10.1016/j.patrec.

2007.12.009]

Mahajan, M., Nimbhorkar, P., Varadarajan, K., 2009. The Planar *K*-means Problem is NP-Hard. 3rd Int. Workshop on Algorithms and Computation, p.274-285. [doi:10.1007/978-3-642-00202-1_24]

Seligson, D.B., Horvath, S., Shi, T., Yu, H., Tze, S., Grunstein, M., Kurdistani, S.K., 2005. Global histone modification patterns predict risk of prostate cancer recurrence. *Nature*, **435**(7046):1262-1266. [doi:10.1038/nature03672]

Theodoridis, S., Koutroumbas, K., 2009. Chapter 13—Clustering Algorithms II: Hierarchical Algorithms. *In*: Pattern Recognition (4th Ed.). Academic Press, Elsevier, London, p.653-700. [doi:10.1016/B978-1-59749-272-0.50015-3]

Wang, R.Z., Tsai, Y.D., 2007. An image-hiding method with high hiding capacity based on best-block matching and *k*-means clustering. *Pattern Recogn.*, **40**(2):398-409. [doi:10.1016/j.patcog.2006.07.015]

Wittkop, T., Emig, D., Lange, S., Rahmann, S., Albrecht, M., Morris, J.H., Bocker, S., Stoye, J., Baumbach, J., 2010. Partitioning biological data with transitivity clustering. *Nat. Methods*, **7**(6):419-420. [doi:10.1038/nmeth0610-419]

Yue, S.H., Li, P., Guo, J.D., Zhou, S.G., 2005. A statistical information-based clustering approach in distance space. *J. Zhejiang Univ.-Sci.*, **6A**(1):71-78. [doi:10.1631/jzus.2005.A0071]

Recommended paper related to this topic

An efficient enhanced *k*-means clustering algorithm

Authors: FAHIM A.M., SALEM A.M., TORKEY F.A., RAMADAN M.A.

doi:10.1631/jzus.2006.A1626

Journal of Zhejiang University-SCIENCE A, 2006 Vol.7 No.10 P.1626-1633

Abstract: In *k*-means clustering, we are given a set of *n* data points in *d*-dimensional space $\mathbb{R}^d$ and an integer *k* and the problem is to determine a set of *k* points in $\mathbb{R}^d$, called centers, so as to minimize the mean squared distance from each data point to its nearest center. In this paper, we present a simple and efficient clustering algorithm based on the *k*-means algorithm, which we call enhanced *k*-means algorithm. This algorithm is easy to implement, requiring a simple data structure to keep some information in each iteration to be used in the next iteration. Our experimental results demonstrated that our scheme can improve the computational speed of the *k*-means algorithm by the magnitude in the total number of distance calculations and the overall time of computation.