5-2018

k-means: A revisit

Wan-Lei ZHAO

Cheng-Hao DENG

Chong-wah NGO
Singapore Management University, cwngo@smu.edu.sg

Neurocomputing 291 (2018) 195–206

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

k-means: A revisit

Wan-Lei Zhao[a,*], Cheng-Hao Deng[a], Chong-Wah Ngo[b]

[a] Fujian Key Laboratory of Sensing and Computing for Smart City, and the School of Information Science and Engineering, Xiamen University, Xiamen 361005, PR China

[b] Department of Computer Science, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 3 December 2016
Revised 12 February 2018
Accepted 21 February 2018
Available online 28 February 2018

Communicated by Deng Cai

Keywords:
Clustering
k-means
Incremental optimization

ABSTRACT

Due to its simplicity and versatility, *k*-means remains popular since it was proposed three decades ago. The performance of *k*-means has been enhanced from different perspectives over the years. Unfortunately, a good trade-off between quality and efficiency is hardly reached. In this paper, a novel *k*-means variant is presented. Different from most of *k*-means variants, the clustering procedure is driven by an explicit objective function, which is feasible for the whole l_2-space. The classic egg-chicken loop in *k*-means has been simplified to a pure stochastic optimization procedure. The procedure of *k*-means becomes simpler and converges to a considerably better local optima. The effectiveness of this new variant has been studied extensively in different contexts, such as document clustering, nearest neighbor search and image clustering. Superior performance is observed across different scenarios.

1. Introduction

Clustering problems arise from variety of applications, such as documents/web pages clustering [1], pattern recognition, image linking [2], image segmentation [3], data compression via vector quantization [4] and nearest neighbor search [5–7]. In the last three decades, various clustering algorithms have been proposed. Among these algorithms, *k*-means [8] remains a popular choice for its simplicity, efficiency and moderate but stable performance across different problems. It was known as one of top ten most popular algorithms in data mining [9]. On one hand, *k*-means has been widely adopted in different applications. On the other hand, continuous efforts have been devoted to enhance the performance *k*-means as well.

Despite its popularity, it actually suffers from several latent issues. Although the time complexity is linear to data size, traditional *k*-means is still not sufficiently efficient to handle the web-scale data. In some specific scenarios, the running time of *k*-means could be even exponential in the worst case [10,11]. Moreover, *k*-means usually only converges to local optima. As a consequence, recent research has been working on either improving its clustering quality [12,13] or efficiency [2,13–19]. *k*-means has been also tailored to perform web-scale image clustering [2,20].

There are in general three steps involved in the clustering procedure. Namely, 1. initialize *k* cluster centroids; 2. assign each sample to its closest centroid; 3. recompute cluster centroids with assignments produced in Step 2 and go back to Step 2 until convergence. This is known as *Lloyd* iteration procedure [8]. The iteration repeats Step 2 and Step 3 until the centroids do not change between two consecutive rounds. Given $C_{1\ldots k} \in R^d$ are cluster centroids, $\{x_i \in R^d\}_{i=1\ldots n}$ are samples to be clustered, above procedure essentially minimizes the following objective function:

$$\min \sum_{q(x_i)=r} \| C_r - x_i \|^2. \quad (1)$$

In Eq. (1), function $q(\cdot)$ returns the closest centroid for sample x_i. Unfortunately, searching an optimal solution for the above objective function is NP-hard. In general *k*-means only converges to local minimum [21]. The reason that *k*-means maintains its popularity is mainly due to its linear complexity in terms of the number of samples to be clustered. The complexity is $O(t \cdot k \cdot n \cdot d)$, given t as the number of iterations to converge. Compared with other well-known clustering algorithms such as DBSCAN [22], Mean shift [23] and clusterDP [24], this complexity is considerably low. However, the efficiency of traditional *k*-means cannot cope with the massive growth of data in Internet. In particular, in the case that the size of data (n), the number of clusters (k) and the dimension (d) are all very large, *k*-means becomes unbearably slow. The existing efforts [16,18] in enhancing the scalability of *k*-means for web-scale tasks often come with price of lower clustering quality. On the other hand, *k*-means++ proposed in [12,17] focuses on enhancing the clustering quality by a careful design of the

* Corresponding author.
E-mail addresses: wlzhao@xmu.edu.cn (W.-L. Zhao), chenghaodeng@stu.xmu.edu.cn (C.-H. Deng), cscwngo@gapps.cityu.edu.hk (C.-W. Ngo).

https://doi.org/10.1016/j.neucom.2018.02.072
0925-2312/

initialization procedure. However, k-means slows down as a few rounds of scanning over the dataset is still necessary in the initialization.

In this paper, a novel variant of k-means is proposed, which aims to make a better trade-off between clustering quality and efficiency. Inspired by the work in [1], a novel objective function is derived from Eq. (1). With the development of this objective function, the traditional k-means iteration procedure has been revised to a simpler form, in which the costly initial assignment becomes unnecessary. In addition, driven by the objective function, sample is moved from one cluster to another cluster when we find this movement leads to higher objective function score, which is known as incremental clustering [1,25]. These modifications lead to several advantages.

- k-means clustering without initial assignment results in better quality as well as higher speed efficiency.
- k-means iteration driven by an explicit objective function converges to considerably lower clustering distortion in faster pace.
- Different from traditional k-means, it is not necessary to assign a sample to its closest centroid in each iteration, which also leads to higher speed.

In addition, when clustering undertaken in hierarchical bisecting fashion, the proposed method achieves the highest scalability among all top-down hierarchical clustering methods. Extensive experiments are conducted to contrast the performance of proposed method with k-means and its variants including tasks document clustering [1], nearest neighbor search (NNS) with product quantization [4] and image clustering.

The remainder of this paper is organized as follows. The reviews about representative works on improving the performance of traditional k-means are presented in Section 2. In Section 3, the clustering objective functions are derived based on Eq. (1). Based on the objective function, Section 4 presents the clustering method. Extensive experiment studies over proposed clustering method are presented in Section 5. Section 6 concludes the paper.

2. Related works

Clustering is a process of partitioning a set of samples into a number of groups without any supervised training. Due to its versatility in different contexts, it has been studied in the last three decades [26]. As the introduction of Web 2.0, millions of data in Internet has been generated on a daily basis. Clustering becomes one of the basic tools to process such big volume of data. As a consequence, traditional clustering methods have been shed with new light. People are searching for clustering methods that are scalable [16–18,27] to web-scale data. In general, boosting the performance of traditional k-means becomes the major trend due to its simplicity and relative higher efficiency over other clustering methods.

In general, there are two major ways to enhance the performance of k-means. For the first kind, the aim is to improve the clustering quality. One of the important work comes from Bahmani et al. [12,17]. The motivation is based on the observation that k-means converges to a better local optima if the initial cluster centroids are carefully selected. According to [12], k-means iteration also converges faster due to the careful selection on the initial cluster centroids. However, in order to adapt the initial centroids to the data distribution, k rounds of scanning over the data are necessary. Although the number of scanning rounds has been reduced to a few in [17], the extra computational cost is still inevitable.

In each k-means iteration, the processing bottleneck is the operation of assigning each sample to its closest centroid. The iteration becomes unbearably slow when both the size and the dimension of the data are very large. Considering that this is a nearest neighbor search problem, Kanungo et al. [14] proposed to index dataset in a KD Tree [28] to speed-up the sample-to-centroid nearest neighbor search. However, this is only feasible when the dimension of data is in few tens. Similar scheme has been adopted by Pelleg and Moore [29]. Unfortunately, due to the curse of dimensionality, this method becomes ineffective when the dimension of data grows to a few hundreds. A recent work [18] takes similar way to speed-up the nearest neighbor search by indexing dataset with inverted file structure. During the iteration, each centroid is queried against all the indexed data. Thanks to the efficiency of inverted file structure, one to two orders of magnitude speed-up is observed. However, inverted file indexing structure is only effective for sparse vectors.

Alternatively, the scalability issue of k-means is addressed by subsampling over the dataset during k-means iteration. Namely, methods in [16,30] only pick a small portion of the whole dataset to update the cluster centroids each time. For the sake of speed efficiency, the number of iterations is empirically set to small value. It is therefore possible that the clustering terminates without a single pass over the whole dataset, which leads to higher speed but also higher clustering distortion. Even though, when coping with high dimensional data in big size, the speed-up achieved by these methods is still limited.

Apart from above methods, there is another easy way to reduce the number of comparisons between the samples and centroids, namely performing clustering in a top-down hierarchical manner [1,31,32]. Specifically, the clustering solution is obtained via a sequence of repeated bisections. The clustering complexity of k-means is reduced from $O(t \cdot k \cdot n \cdot d)$ to $O(t \cdot log(k) \cdot n \cdot d)$. This is particularly significant when n, d and k are all very large. In addition to that, another interesting idea from [1,32] is that cluster centroids are updated incrementally [1,25]. Moreover, the update process is explicitly driven by an objective function (called as criterion function in [1,32]). Unfortunately, objective functions proposed in [1,31,32] are based on the assumption that input data are in unit length. The clustering method is solely based on *Cosine* distance, which makes the clustering results unpredictable when dealing with data in the general l_2-space.

In this paper, a new objective function is derived directly from Eq. (1), which makes it suitable for the whole l_2-space. In other word, objective function proposed in [1] is the special case of our proposed form. Based on the proposed objective function, conventional egg-chicken k-means iteration is revised to a simpler form. On one hand, when applying the revised iteration procedure in direct k-way clustering, k-means is able to reach to considerably lower clustering distortion within only a few rounds. On the other hand, as the iteration procedure is undertaken in top-down hierarchical clustering manner (specifically bisecting), it shows faster speed while maintaining relatively lower clustering distortion in comparison to traditional k-means and most of its variants.

3. Clustering objective functions

In this section, the clustering objective functions upon which our k-means method is built are presented. Basically, two objective functions that aim to optimize the clustering results from different aspects are derived. Furthermore, we also show that these two objective functions can be reduced to a single form.

3.1. Preliminaries

In order to facilitate the discussions that are followed, several variables are defined. Throughout the paper, the size of input data is given as n, while the number of clusters to be produced is given as k. The partition formed by a clustering method is represented as $\{S_1, \ldots, S_r \ldots, S_k\}$. Accordingly, the sizes of clusters are given as $n_1, \ldots, n_r, \ldots, n_k$. The composite vector of a cluster is defined as

$D_r = \sum_{x_i \in S_r} x_i$. The cluster centroid C_r[1] is defined by its members,

$$C_r = \frac{\sum_{i=1}^{n_r} x_i}{n_r} = \frac{D_r}{n_r} \tag{2}$$

The inner-product of C_r is given by $C_r'C_r = \frac{(\sum_{i=1}^{n_r} x_i)'(\sum_{i=1}^{n_r} x_i)}{n_r^2}$, which is expanded as following form.

$$\begin{aligned} C_r'C_r = \frac{1}{n_r^2}[&(x_1'x_1 + \cdots + x_1'x_i + \cdots + x_1'x_{n_r}) \\ &+ (x_2'x_1 + \cdots + x_2'x_i + \cdots + x_2'x_{n_r}) \\ &+ \cdots \\ &(x_i'x_1 + \cdots + x_i'x_i + \cdots + x_i'x_{n_r}) \\ &+ \cdots \\ &(x_{n_r}'x_1 + \cdots + x_n'x_i + \cdots + x_{n_r}'x_{n_r})] \\ = \frac{1}{n_r^2}&\left(\sum_{i=1}^{n_r} x_i^2 + 2\sum_{i,j=1\&i<j}^{n_r} < x_i, x_j > \right) \end{aligned}$$

Re-arrange the above equation, we have

$$\sum_{i,j=1\&i<j}^{n_r} < x_i, x_j > = \frac{1}{2}\left({n_r}^2 \cdot C_r'C_r - \sum_{i=1}^{n_r} x_i^2\right). \tag{3}$$

The sum of pairwise l_2-distance within one cluster is given as

$$S = (n_r - 1)\sum_{i=1}^{n_r} x_i^2 - 2\cdot \sum_{i,j=1\&i<j}^{n_r} < x_i, x_j > . \tag{4}$$

Plug Eq. (3) into Eq. (4), we have

$$\begin{aligned} S &= (n_r - 1)\sum_{i=1}^{n_r} x_i^2 - \left({n_r}^2 \cdot C_r'C_r - \sum_{i=1}^{n_r} x_i^2\right) \\ &= (n_r - 1)\sum_{i=1}^{n_r} x_i^2 - {n_r}^2 \cdot C_r'C_r + \sum_{i=1}^{n_r} x_i^2 \\ &= n_r \sum_{i=1}^{n_r} x_i^2 - {n_r}^2 \cdot C_r'C_r. \end{aligned} \tag{5}$$

Eq. (5) is rewritten as

$$S = n_r \sum_{i=1}^{n_r} x_i^2 - D_r'D_r. \tag{6}$$

3.2. Objective functions

In this section, two objective functions (also known as criterion functions [1]) are developed. In addition, with the support of the results obtained in Section 3.1, these objective functions will be reduced to simple forms, which enable them to be carried out efficiently in the incremental optimization procedure.

According to [1], objective functions are categorized into two groups. One group of the functions consider the tightness of clusters, while another focuses on alienating different clusters. In this paper, the focus is on producing a clustering solution defined over the elements within each cluster. It therefore does not consider the relationship between the elements assigned to different clusters.

The first objective function we consider is to minimize the distance of each element to its cluster centroid, which is nothing more than the objective function of k-means.

$$\begin{aligned} \text{Min. } \mathcal{I}_1 &= \sum_{q(x_i)=r} \| C_r - x_i \|^2 \\ &= \sum_{r=1}^{k}\sum_{x_i \in S_r} d(x_i, C_r). \end{aligned} \tag{7}$$

The above equation is simplified as

$$\begin{aligned} \text{Min. } \mathcal{I}_1 &= \sum_{r=1}^{k}\left(\sum_{i=1}^{n_r} x_i'x_i + n_r C_r'C_r - 2\sum_{i=1}^{n_r} x_i'C_r\right) \\ &= \sum_{r=1}^{k}\left(\sum_{i=1}^{n_r} x_i'x_i + \frac{D_r'D_r}{n_r} - 2\frac{D_r'D_r}{n_r}\right) \\ &= \sum_{r=1}^{k}\left(\sum_{i=1}^{n_r} x_i'x_i - \frac{D_r'D_r}{n_r}\right) \\ &= \sum_{r=1}^{k}\sum_{i=1}^{n_r} x_i'x_i - \sum_{r=1}^{k}\frac{D_r'D_r}{n_r} \\ &= E - \sum_{r=1}^{k}\frac{D_r'D_r}{n_r} \end{aligned} \tag{8}$$

Since the input data are fixed, E is a constant. As a result, minimizing Eq. (8) is equivalent to maximizing following function

$$\text{Max. } \mathcal{I}_1^* = \sum_{r=1}^{k}\frac{D_r'D_r}{n_r}. \tag{9}$$

Although objective function in Eq. (9) is in the same form as the first objective function in [1], they are derived from different initial objectives. More importantly, in our case, there is no constraint that input data should be in unit length.

The second internal objective function that we will study minimizes the sum of the average pairwise distance between the elements assigned to each cluster, weighted according to the size of each cluster.

$$\text{Min. } \mathcal{I}_2 = \sum_{r=1}^{k} n_r\left(\frac{2}{n_r\cdot(n_r-1)}\sum_{d_i,d_j\in S_r\&i>j} d(x_i, x_j)\right) \tag{10}$$

Plug Eq. (6) in, we have

$$\begin{aligned} \text{Min. } \mathcal{I}_2 &= \sum_{r=1}^{k} n_r\left(\frac{2}{n_r\cdot(n_r-1)}\left(n_r\sum_{i=1}^{n_r} x_i'x_i - D_r'D_r\right)\right) \\ &= \sum_{r=1}^{k}\frac{2n_r}{n_r-1}\sum_{i=1}^{n_r} x_i'x_i - 2\sum_{r=1}^{k}\frac{D_r'D_r}{n_r-1} \end{aligned} \tag{11}$$

In Eq. (11), $\frac{n_r}{n_r-1}$ is close to *1*, the above objective function is approximated as

$$\text{Min. } \mathcal{I}_2 \approx 2E - 2\sum_{r=1}^{k}\frac{D_r'D_r}{n_r}. \tag{12}$$

Similar as Eq. (8), since the input data are fixed, E is a constant. As as result, minimizing Eq. (12) is equivalent to maximizing function

$$\text{Max. } \mathcal{I}_2^* \approx \sum_{r=1}^{k}\frac{D_r'D_r}{n_r}. \tag{13}$$

Noticed that similar optimization objectives have been discussed under *Cosine* similarity measure in [1]. In the paper, two objective functions are reduced into different forms. This is different from the result obtained in our case (general l_2-space). As

[1] We refer to as column vector across the paper.

shown above, in l_2-space, the objective functions for $\mathcal{I}_1^*$ and $\mathcal{I}_2^*$ are approximately the same. The advantage that two objective functions are reduced to the same form is that, when we try to optimize one objective function, we optimize another in the mean time. Specifically, when we minimize the distances from elements to their cluster centroid, the average intra-cluster distance is minimized in the meantime. Since these two objective functions can be simplified to the same form, only objective function $\mathcal{I}_1^*$ is discussed in the rest of paper.

Although objective function in Eq. (9) is derived from Eq. (1), the former is much easier to operate in the incremental k-means procedure. As it will be shown in the next section, it is quite convenient to evaluate whether Eq. (9) attains a higher score (implies lower distortion in terms of Eq. (1)) when a sample x_i is moved from one cluster to another.

4. *k*-means driven by objective function

In this section, with the objective function developed in Section 3, two iterative clustering procedures are presented. Namely, one produces k clusters directly (called as direct k-way k-means), while another produces k clusters by bisecting input data sequentially k-1 times (called as bisecting k-means). Both clustering strategies are built upon incremental clustering [1,25] and driven by objective function $\mathcal{I}_1^*$ (Eq. (9)).

4.1. Clustering algorithm

The basic idea of incremental clustering is that one sample x_i is moved from cluster S_u to S_v as soon as this movement leads to higher score of objective function $\mathcal{I}_1^*$. To facilitate our discussion, the new function value as sample x_i is moved from S_u to S_v is formulated as following.

$$\begin{aligned}
&\Delta\mathcal{I}_1^*(x_i)\\
&= \frac{(D_v + x_i)'(D_v + x_i)}{n_v + 1} + \frac{(D_u - x_i)'(D_u - x_i)}{n_u - 1} - \frac{D_v'D_v}{n_v} - \frac{D_u'D_u}{n_u}\\
&= \frac{D_v'D_v + 2x_i'D_v + x_i'x_i}{n_v + 1} + \frac{D_u'D_u - 2x_i'D_u + x_i'x_i}{n_u - 1} - \frac{D_v'D_v}{n_v} - \frac{D_u'D_u}{n_u}\\
&= 2x_i'\frac{D_v}{n_v + 1} - 2x_i'\frac{D_u}{n_u - 1} + \frac{D_v'D_v}{n_v + 1} + \frac{D_u'D_u}{n_u - 1} + \frac{x_i'x_i}{n_v + 1}\\
&\quad + \frac{x_i'x_i}{n_u - 1} - \frac{D_v'D_v}{n_v} - \frac{D_u'D_u}{n_u} \qquad (14)
\end{aligned}$$

In each iteration of the clustering, sample x_i is randomly selected. The algorithm checks whether moving x_i from its current cluster to any other cluster will lead to higher $\mathcal{I}_1^*$ (i.e., $\Delta\mathcal{I}_1^* > 0$). If it is the case, x_i is moved to another cluster. The clustering procedure is detailed in Algorithm 1.

As seen from **Step 3** of Algorithm 1, the initialization of our method is different from most of the current practice of k-means, there is no assignment of each sample to its closest initial centroid. On the contrary, each sample x_i is assigned with a random cluster label (ranges from *1* to k). This allows to calculate an initial score of $\mathcal{I}_1^*$ and the composite vector D of each cluster. It is possible to do the initial assignment following the way of k-means or k-means++ [12]. However, as will be revealed in Section 5, initialization under either k-means manner or k-means++ manner improves the clustering quality slightly. However, extra computation is required in such kind of initial assignment.

During each iteration, each sample $x_i \in X$ is checked in random order. The optimization in **Step 8–10** seeks the movement of x_i that leads to the highest increase of function score. From the optimization view, the algorithm reduces the clustering distortion greedily. While from another view, the seeking process is comparable to the sample-to-centroid assignment in traditional k-means. They are actually on the same computational complexity level.

Algorithm 1 Direct k-way k-means$^{\#}$.

1: **Input**: matrix $X_{n\times d}$
2: **Output**: $S_1, \cdots, S_r, \cdots S_k$
3: Assign $x_i \in X$ with a random cluster label;
4: Calculate $D_1, \cdots, D_r, \cdots D_k$ and $\mathcal{I}_1^*$;
5: **while** not convergence **do**
6: **for** each $x_i \in X$ (in random order) **do**
7: Seek S_v that maximizes $\Delta\mathcal{I}_1^*(x_i)$;
8: **if** $\Delta\mathcal{I}_1^*(x_i) > 0$ **then**
9: Move x_i from current cluster to S_v;
10: **end if**
11: **end for**
12: **end while**

Whereas it is not necessary that we must seek the best movement for x_i. As we discover by experiment, it is feasible that moving x_i to another cluster as long as we find $\Delta\mathcal{I}_1^*(x_i)$ is greater than *0*. On one hand, this will speed-up the iteration. On the other hand such kind of scheme usually takes more rounds to reach to the same level of distortion. However, we discover that such kind of less greedy scheme results in lower clustering distortion if the iteration loops for sufficient number of times.

Moving x_i from one cluster to another (**Step 9**) is very convenient to take. It includes the operation that updates the cluster label of x_i and the operation that updates the composite vector for cluster S_v and S_u, viz., $D_v = D_v + x_i$, $D_u = D_u - x_i$.

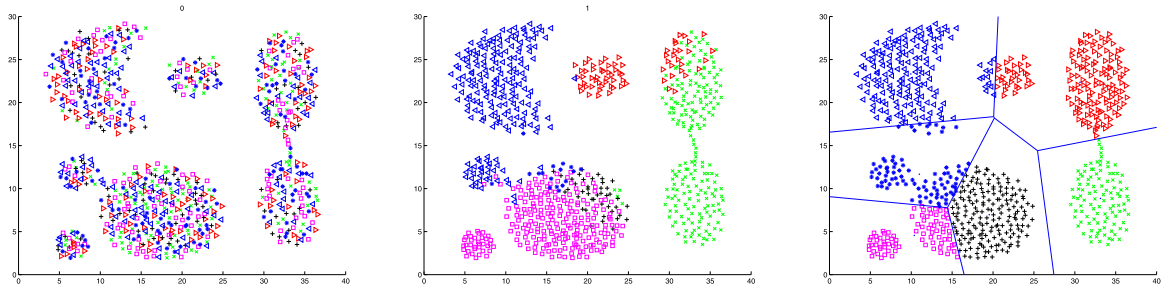
Note that this incremental updating scheme is essentially different from online learning vector quantization (LVQ) [33], in which the cluster centroids are updated incrementally. In the above iteration procedure, no cluster centroids are explicitly produced. As a result, it is no need to update cluster centroid. The clustering iteration is explicitly driven by an objective function rather than by the discrepancy between cluster centroids and their cluster members. As revealed later in the experiment, compared to LVQ, Algorithm 1 is more efficient and leads to considerably lower distortion.

Fig. 1 illustrates three iterations of Algorithm 1 in 2D case. As shown in the figure, the initial clustering result is random and messy. Samples belonging to different clusters are totally mixed up. However, only after one round of iteration, the clustering result becomes much more compact. The clustering terminates at the *10*th round, where *Lloyd*'s condition is reached. The optimum of this procedure is analyzed in Appendix A and its convergence is proved in Appendix B.

Overall, method presented in Algorithm 1 is different from traditional k-means in three major aspects. Firstly, no initial assignment is required. Moreover, the egg-chicken loop in the traditional k-means has been replaced by a simpler stochastic optimization procedure. Furthermore, unlike traditional k-means, it is not necessary to seek the best movement for each sample in the iteration. Due to the essential upgrade of our method makes over traditional k-means, it is named as k-means$^{\#}$.

The method presented in Algorithm 1 is on the same complexity level as traditional k-means (i.e., $O(t \cdot n \cdot d \cdot k)$), which is unbearably slow when dealing with large-scale data. In order to adapt it to large-scale task, the method is revised into a top-down hierarchical clustering. Specifically, at each time, one intermediate cluster is selected and bisected into two smaller clusters by calling Algorithm 1. The details of this method are given in Algorithm 2.

As shown in Algorithm 2, priority queue Q pops out one cluster for bisecting each time. As discussed in [32], there are basically two ways to organize the priority queue. One can priori-



(a) initialization (b) iter=1 (c) iter=10

Fig. 1. Illustration of direct k-way k-means clustering with Algorithm 1. The clustering process starts from the state that samples are all assigned with random label. The final cluster centroids in (c) form a convex partition over the 2D space, which are called as *Voronoi* diagram. According to *Lloyd*'s condition, all the samples belonging to one cluster fall into the same *Voronoi* cell.

Algorithm 2 Bisecting k-means$^{\#}$.

Input: matrix $X_{n\times d}$
Output: $S_1, \ldots, S_r, \ldots S_k$
Treat X as one cluster S_1;
Push S_1 into a priority queue Q;
i = 1;
while $i < k$ **do**
  Pop cluster S_i from queue Q
  Call Alg. 1 to bisect S_i into $\{S_i, S_{i+1}\}$;
  Push S_i, S_{i+1} into queue Q;
  i = i + 1;
end while

tize the cluster with biggest size or the one with highest average intra-cluster distance to split. Similar as [32], we find splitting the biggest cluster usually demonstrates more stable performance. As a result, the queue is sorted in descending order by the cluster sizes in our practice.

It is possible to partition the intermediate cluster into more than two clusters each time. In the following, we are going to show that this bisecting scheme achieves highest scalability among all alternative top-down secting schemes.

4.2. Scalability analysis

In this section, the computation complexity of Algorithm 2 is studied by considering the total number of comparisons required in the series of bisecting clustering. The number of iterations in each bisecting is assumed to be a constant by taking the average number of iterations.

In order to facilitate the analysis while without loss of generality, we assume that each intermediate cluster in Algorithm 2 is partitioned evenly. In addition, we generalize Algorithm 2 to an s-secting algorithm. Namely, an intermediate cluster is partitioned to s ($s \geq 2$) clusters. Now we consider the size of series of intermediate clusters that are produced when performing sequential secting. Given q is the depth of splitting, it is easy to see $\lceil \log_s k \rceil = q + 1$. The sizes of all intermediate clusters are given as following.

$$n, \underbrace{\frac{n}{s}, \frac{n}{s}, \ldots,}_{s} \underbrace{\frac{n}{s^2}, \frac{n}{s^2}, \ldots,}_{s^2}, \ldots, \underbrace{\frac{n}{s^q}, \frac{n}{s^q}, \ldots}_{s^q}$$

As a result, the number of samples to be visited during the clustering procedure is

$$\begin{aligned} & n + \frac{n}{s} * s^1 + \frac{n}{s^2} * s^2 + \frac{n}{s^3} * s^3 \cdots + \frac{n}{s^q} * s^q \\ & = n + \underbrace{n + n + n + \cdots + n}_{q} \\ & = n * (1 + q) \\ & \approx n * \log_s k. \end{aligned} \tag{15}$$

Considering that one sample has to compare with $s - 1$ centroids each time, the total number of comparisons is

$$n * (s - 1) * \log_s k. \tag{16}$$

Given n and k are fixed, Eq. (16) increases monotonically with respect to s. As a result, the number of comparisons reaches to the minimum when $s = 2$ i.e., $n\log_2 k$. To this end, it is clear that bisection is the most efficient secting scheme.

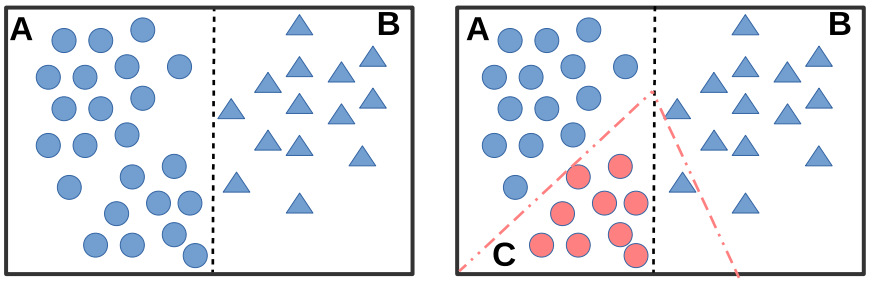
Compared with Algorithm 1, the complexity of Algorithm 2 is reduced to $O(\bar{t} \cdot n \cdot d \cdot log(k))$, where $\bar{t}$ is the average number of iterations in each bisecting. Compared with t in traditional k-means, $\bar{t}$ is much smaller given the scale of clustering problem is much smaller in terms of both the size of input data and the number of clusters to be produced. As a result, the complexity of Algorithm 1 has been largely reduced since term $n \cdot d$ has been multiplied by a much smaller factor $\bar{t} \cdot log(k)$.

Although Algorithm 2 is efficient, the clustering result produced by Algorithm 2 unfortunately does not satisfy with *Lloyd*'s condition. This problem is illustrated in Fig. 2. As one of the clusters is further partitioned into two (from Fig. 2(a) to Fig. 2(b)), the partition over 2D space is formed by centroids changes. Cluster C claims bordering points from cluster B. However, points from cluster B cannot be reassigned to cluster C if no further intervention is involved. This is actually an underfitting issue and exists for any hierarchical clustering method. Fortunately, this issue can be alleviated by adopting Algorithm 1 as a refinement procedure after Algorithm 2 outputs k clusters. To do so, extra time is required. It therefore becomes a problem of balancing between efficiency and quality.

According to our observation, it is possible to further speed-up the proposed k-means$^{\#}$. After a few iterations, both k-means and k-means$^{\#}$ will be trapped in a local minima. Only samples that bordering between different clusters are shuffled from one cluster to another. As a result, given a sample, it is no need to search for the best movement among k clusters. Instead, the sample only needs to compare to the closest k_0 ($k_0 \ll k$) centroids (or clusters) to search for the suitable movement. We find that, this simple modification leads to typically $7 \sim 8$ times speed-up while without significant performance degradation.

5. Experiments

In this section, the effectiveness of proposed clustering method, namely k-means$^{\#}$ is studied under different scenarios. In the first experiment, dataset SIFT1M [5] is adopted to evaluate the clustering quality. In the second experiment, k-means$^{\#}$ is tested on



(a) the 1st round bisecting (b) the 2nd round bisecting

Fig. 2. Illustration of two consecutive bisections in the bisecting clustering where *Lloyd*'s condition breaks.

Table 1
Configurations of *k*-means and its variants and their corresponding abbreviations.

	k-means		*k*-means$^{\#}$	
Initial assignment	*k*-way	bisecting	*k*-way	bisecting
Random	*k*-means [8]	BsKM	*k*-means$^{\#}$(rnd)	BsKM$^{\#}$(rnd)
Probability based [12]	*k*-means+ [12]	BsKM++	*k*-means$^{\#}$(kpp)	BsKM$^{\#}$(kpp)
Non	–	–	*k*-means$^{\#}$(non)	BsKM$^{\#}$(non)

the nearest neighbor search task based on product quantizer (PQ) [5] in which this method is adopted for quantizer training. In the third experiment, *k*-means$^{\#}$ has been applied to traditional document clustering. Following the practice of [1,32], *15* document datasets[2] have been adopted. In the last experiment, the scalability of *k*-means$^{\#}$ has been tested on large-scale image clustering task, for which the number of images we use is as large as *10* million.

In our study, the performance from traditional *k*-means is treated as comparison baseline. In addition, representative *k*-means variants, such as Mini-Batch [16], Repeated Bisecting *k*-means (RBK) [32], online Learning Vector Quantization (LVQ) [33] and *k*-means++ [12] are considered in the comparison. For Mini-Batch, our configuration makes sure that the iteration covers *10%* of the input data. The configuration is fixed across all the experiments. For RBK, we select the objective function that maximizes the average *Cosine* similarity between samples within one cluster, which is the special case of ours given the input data is l_2-normalized. LVQ is similar to *k*-means except that in each round, a cluster centroid is updated as soon as a sample is assigned. The updating rate starts from *0.01* and decreases at a pace of 4×10^{-4} in one iteration.

As shown in Table 1, there are several variants of *k*-means and *k*-means$^{\#}$ due to the differences in initial assignment schemes and variations in partitioning strategies (i.e., direct *k*-way or bisecting). In the table, 'initial assignment' refers to the operation of selecting samples as initial centroids and assigning each sample to its closest initial centroid. When the initial assignment is operated by selecting seeds randomly as traditional *k*-means, it is denoted as 'rnd'. When the initial centroids are selected based on probability as *k*-means++, it is denoted as 'kpp'. While for the initialization without initial assignment (proposed by us) is denoted as 'non'. In this initialization, a random label is assigned to each sample. In the experiments, all the variants out of these different configurations on *k*-means as well as *k*-means$^{\#}$ are considered. Their configurations and corresponding abbreviations are shown in Table 1. Noted that BsKM$^{\#}$(rnd) is the same as RBK [32] if the input data is l_2-normalized. The experiment in this section is conducted on 1 million SIFT features [34]. The features are clustered into *10,000* partitions.

In addition, we also study the performance trend of *k*-means$^{\#}$ when **Steps 7-10** in Algorithm 1 are modified to moving the sample as soon as $\Delta I_1(x_i) > 0$. The variants under this modification are denoted as *k*-means$^{\#}(\cdot)$+Fast [3]. All the methods considered in the paper are implemented in C++ and the simulations are conducted on a PC with *2.4*GHz Xeon CPU and *32*G memory setup.

5.1. Evaluation of clustering distortion

Since *k*-means and most of its variants share the same objective function (Eq. (1)), it is straightforward to evaluate the clustering performance by checking to what degree the objective is reached. The average distortion (given in Eq. (17)) is adopted for evaluation [2], which takes average over Eq. (1),

$$\mathcal{E} = \frac{\sum_{q(x_i)=r} \| C_r - x_i \|^2}{n}. \tag{17}$$

For above equation, the lower the distortion value, the better is the clustering quality.

The first experiment mainly studies the behavior of the proposed *k*-means$^{\#}$ under different initializations. The average distortion curves produced by variants direct *k*-way *k*-means$^{\#}$ are given in Fig. 3(a) as a function of numbers of iteration. Traditional *k*-means is treated as baseline for performance comparison. The result shows that clustering distortion of *k*-means$^{\#}$ drops faster than traditional *k*-means. The average distortion from traditional *k*-means is around *40,450* after *130* iterations. In contrast, *k*-means$^{\#}$ without initial assignment (*k*-means$^{\#}$(non)) is able to reach to the same distortion level after only 7 iterations. Moreover, we find that initializing *k*-means$^{\#}$ as traditional *k*-means way (*k*-means$^{\#}$(rnd)) or as *k*-means++ (*k*-means$^{\#}$(kpp)) allows the iteration to start from a low distortion level. Nevertheless the advantage over *k*-means$^{\#}$(non) fades away after *15* iterations. In comparison to *k*-means$^{\#}$(non), the extra cost is required for clustering that adopts initial assignment, which is close to ('rnd' case) or higher than ('kpp' case) the cost of one round iteration.

The second experiment studies the performance trend of Algorithm 1 when **Steps 7-10** do not seek the best movement (*k*-means$^{\#}(\cdot)$+Fast). As shown in Fig. 3(b), the distortion drops slower than *k*-means$^{\#}$(non) which seeks the best movement. However, lower distortion is achievable by *k*-means$^{\#}$(rnd)+Fast as the number of iterations is sufficiently large e.g., *20*. This indicates that

[2] Available at http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/datasets.tar.gz

[3] Note that this is not applicable for bisecting *k*-means$^{\#}$.

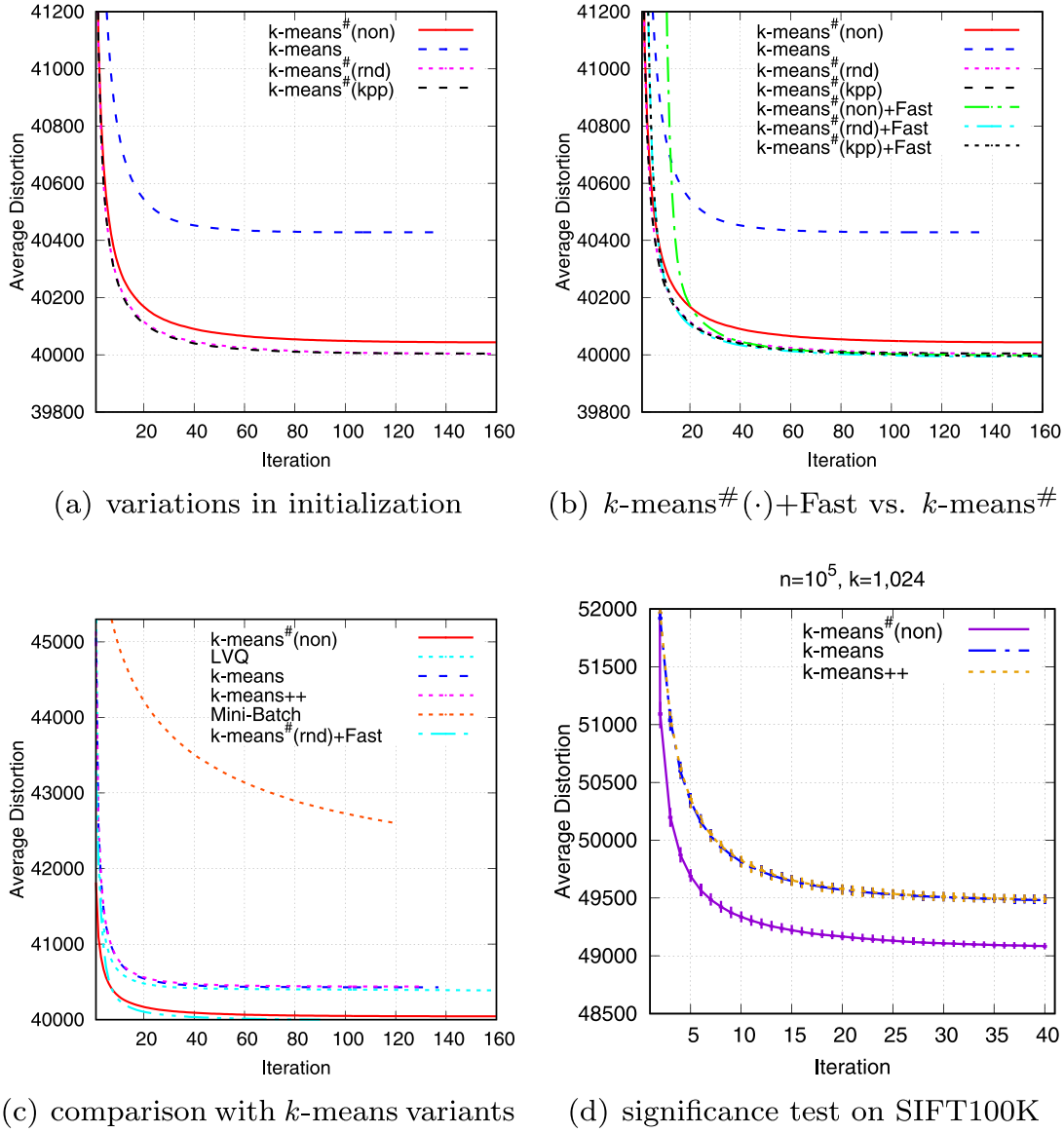


Fig. 3. The experiments are conducted on SIFT1M for figures (a)–(c) and on SIFT100K for figure (d). The results show different performance: (a) impact of initialization in different ways; (b) fast version of k-means$^{\#}$ by not seeking optimal movement in the steps 7–10 of Algorithm 1; (c) Comparison of k-means$^{\#}$ to variants of k-means; (d) significance of improvement over other k-means variants achieved by k-means$^{\#}$ by repeating the experiments by 128 runs.

when the optimization scheme is less greedy, it is less likely to be trapped in a worse local optima. This observation applies to k-means$^{\#}$ under different types of initialization. Noted that the time cost for k-means$^{\#}(\cdot)$+Fast is lower than that of k-means$^{\#}$ that seeks the best movement in each iteration. Whereas, k-means$^{\#}(\cdot)$+Fast usually needs a few more number of iterations to reach to the similar distortion level. Overall, as investigated in Section 5.4, k-means$^{\#}(\cdot)$+Fast is 5% faster than k-means$^{\#}(\cdot)$.

Fig. 3(c) studies the trend of average distortion among the proposed k-means$^{\#}$ (specifically k-means$^{\#}$(non)), traditional k-means, k-means++, Mini-Batch and LVQ. For all the methods presented, their distortion decreases steadily as the iteration continues. A big performance gap is observed between Mini-Batch and other k-means variants. In addition k-means and k-means++ share similar distortion curve. k-means$^{\#}$(non) outperforms k-means and k-means++ after only *7* iterations. Most of the methods including k-means and k-means++ take more than *120* iterations to finally converge. On the other hand, little distortion is observed after *20* iterations, which implies the validity of early termination at i.e., *20*. Although similar as k-means$^{\#}$, LVQ updates the intermediate clusters incrementally, updating cluster centroid directly turns out to be inefficient, which leads to considerably poor performance.

Since k-means and its variants are all sensitive to initialization, the performance fluctuates from one run to another. The candlestick chart shown in Fig. 3(d) further confirms the significance of the improvement achieved by k-means$^{\#}$. This chart is plotted with *128* clustering runs ($k = 1,024$) on SIFT100K [5] for each method. As shown in the figure, although the performance fluctuates for all the methods, the variations are minor. Similar as previous observation, there is no significant difference between traditional k-means and k-means++. In contrast, the performance gap between k-means$^{\#}$ and traditional k-means is much more significant than the performance variations across different runs.

Table 2 shows the average distortion of different k-means variants under bisecting strategy. The result from k-means (after *130* iterations) is presented for the comparison. As shown from the table, the average distortion from all bisecting methods are on the level of 4.5×10^4. Methods built upon Algorithm 1 always perform

Table 2
Average distortion from k-means variants under bisecting strategy.

Method	k-means	RBK	BsKM	BsKM++	BsKM$^{\#}$(non)	BsKM$^{\#}$(rnd)	BsKM$^{\#}$(kpp)
$\mathcal{E}$	40450.0	45713.5	45835.2	45823.8	**45650.7**	45661.2	45658.4
$\mathcal{E}$ after Rfn.	–	43364.4	43323.9	43366.2	43293.3	43285.5	**43285.4**

better. The average distortion from all bisecting clustering methods are much higher than that of k-means. They are actually only close to the distortion level of k-means after one iteration. However, the merit of clustering with bisecting strategy is that it is more than *20* times faster than k-means of a single iteration. The relatively poor clustering quality produced by bisecting strategy is mainly due to the issue of underfitting (as discussed in Section 4.2). The clustering results can be further refined by Algorithm 1 as shown on the *3rd* row of Table 2.

As learned from above experiments, on one hand initial assignment under k-means manner or under k-means++ manner is able to improve the performance of k-means$^{\#}$ slightly. On the other hand, the initial assignment slows down the method considerably. A trade-off has to be made. In the following experiments, only the results from two representative configurations of k-means$^{\#}$, namely k-means$^{\#}$(non) and k-means$^{\#}$(rnd)+Fast are presented. k-means$^{\#}$(rnd)+Fast is written as k-means$^{\#}$+Fast for succinctness in the rest of the paper. We leave other possible configurations to the readers.

5.2. Nearest neighbor search by product quantizer (PQ)

In this section, k-means$^{\#}$ is applied for visual vocabulary training using product quantization [5]. Following the practice of [5], *100*K SIFT features are used for product quantizer training, while SIFT1M set [5] is encoded with the trained product quantizers as the reference set for nearest neighbor search (NNS). The obtained recall@top-k is averaged over *1000* queries for each method. In the experiment, two different settings are tested for product quantizer. Namely, the *128*-dimensional SIFT vector is encoded with *8* and *16* product quantizers, respectively. For clarity, the evaluations are separately conducted for direct k-way and bisecting k-means.

Recall@top-100 for direct k-way are presented in Fig. 4(a)–(d) under two different settings ($m = 8$ and $m = 16$), where m is the number of divisions that PQ applies on a vector [5]. As seen from the figures, the performances from k-means, k-means++ and k-means$^{\#}$(non) are all very close to each other under different settings. The product quantizer trained with bisecting clustering methods shows only *0.1–1.3%* lower performance than that of direct k-way methods. This basically indicates that product quantizer itself is insensitive to the clustering quality. The performance of Mini-Batch and RBK is around *2–6%* lower than the other methods. The poor performance of RBK basically indicates the optimization objective function defined under *Cosine* similarity is not directly feasible for general l_2-space.

5.3. Document clustering

In this section, the performance of proposed method is evaluated under the context of document clustering. Following in [1], *15* document datasets are used for evaluation. The documents have been represented with *TF/IDF* model and normalized to unit length. Similar to [1], entropy as follows is adopted for the evaluation

$$Entropy = \sum_{r=1}^{k} \frac{n_r}{n} \frac{1}{\log c} * \sum_{i=1}^{c} \frac{n_r^i}{n_r} * \log \frac{n_r^i}{n_r}, \tag{18}$$

where c is the number of classes. Eq. (18) evaluates to what degree that elements from the same class are put in one cluster. The lower of the value, the better is the performance. In the experiment, each method performs clustering for *10* runs, and the run with the lowest entropy is presented in Table 3. The presented entropy are averaged over *15* datasets.

Table 3
Clustering performance (average entropy) on 15 datasets.

	k = 5	k = 10	k = 15	k = 20
k-means	0.539	0.443	0.402	0.387
k-means++	0.550	0.441	0.403	0.389
Mini-Batch	0.585	0.488	0.469	0.475
LVQ	0.800	0.761	0.681	0.674
k-means$^{\#}$(non)	0.552	0.442	0.388	0.368
k-means$^{\#}$+Fast	**0.506**	**0.419**	**0.380**	**0.353**
BsKM	0.532	0.438	0.410	0.373
BsKM++	0.507	0.422	0.400	0.379
BsKM$^{\#}$(non)	0.514	**0.388**	**0.353**	**0.329**
RBK	**0.486**	0.402	0.366	0.339

In general, k-means$^{\#}$ under different configurations performs considerably better. Furthermore, methods with bisecting strategy demonstrate slightly better performance than that of direct k-way in the document clustering task. Similar observation is shared in [32]. As observed in [32], the tightness of different document classes are different. Moreover, there is a discrepancy between *TF/IDF* model and human perception about document classes.[4] Lower distortion does not necessarily mean better cluster quality (i.e., lower entropy). Compared to direct k-way clustering, bisecting strategy partitions the documents in a top-down manner. From the top view, it is easier for bisecting to partition tight cluster from loose one in its early stage. Due to the nature of bisecting strategy, the bisecting in later stages is restricted from moving documents from tight to loose. Because of that, the integrity of document classes with different tightness are largely preserved. In contrast, in direct k-way, documents in different classes are free to move to achieve lower distortion since the proposed k-means iteration is a greedy process. As k increases, documents have more wrong candidate clusters to move in to attain lower distortion. Therefore performance gap between bisecting and k-way grows as k increases in Table 3. Overall, BsKM$^{\#}$(non) shows the best performance. The performance of RBK (the same as BsKM$^{\#}$(rnd)) is close to BsKM$^{\#}$(non). The marginal performance gap between these two methods is due to the difference in their initialization.

5.4. Scalability test on image clustering

In this section, the scalability of the proposed k-means is tested on image clustering. The experiment is conducted on *10* million Flickr images (*Flickr10M*), which are a subset of *YFCC100M* [35]. *Hessian-Affine* [36] keypoints are extracted from each image and are described by RootSIFT feature [37]. Finally, the RootSIFT features from each image are pooled by VLAD [38] with a small visual vocabulary of size *64*. The resulting *8192*-dimensional feature is further mapped to *512* dimensions by PCA. Following [38], the final VLAD vector is normalized to unit length. In the direct k-way clustering case, we set the number of maximum iterations for all methods to *20*. While for the bisecting case, there is no threshold

[4] The class label of each document in the ground-truth is given by human being.

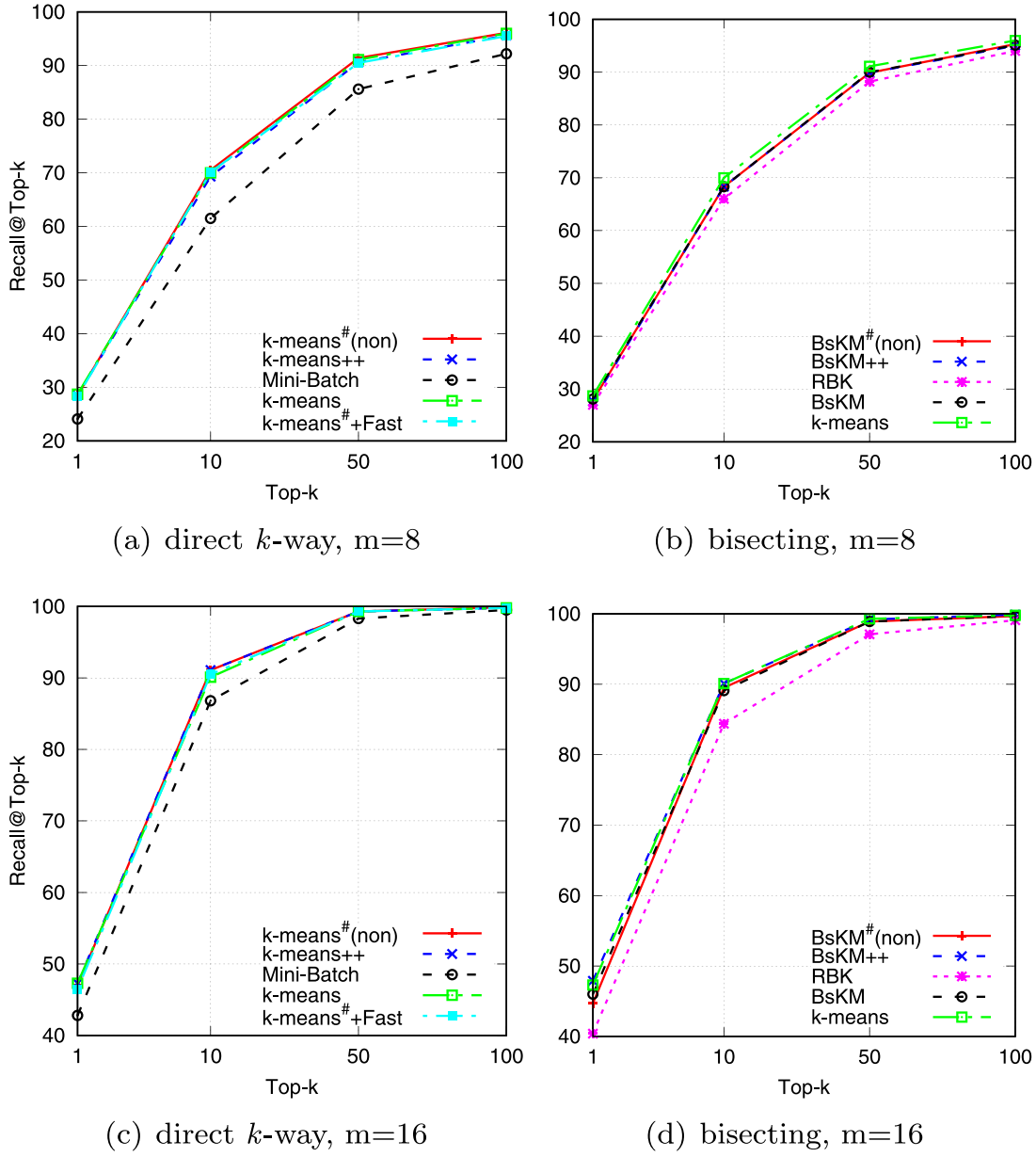


Fig. 4. Performance of nearest neighbor search by PQ on SIFT1M when adopting different clustering methods for quantizer training. The size of each product quantizer is fixed to *256* across all the experiments. The asymmetric distance calculation (ADC) [5] is adopted for nearest neighbor search.

on the number of iterations. The results reported in this section have been averaged over *10* runs for each method.

In the first experiment, clustering methods are tested in the way that the scale of input images varies from 10K to 10M. While the number of clusters to be produced is fixed to *1024* regardless of the size of dataset. The time costs for direct k-way and bisecting methods are presented in Fig. 5(a) and (b). Accordingly, the average distortion of all the methods are presented in Fig. 6(a).

As shown in the figures, k-means$^{\#}$ exhibits slightly faster speed over k-means and its variants across different scales of input data under both direct k-way and bisecting cases. The speed-up becomes more significant as the scale of input data increases. The higher efficiency of these methods is mainly attributed to the no involvement of initial assignment. Compared to k-means$^{\#}$(non), k-means$^{\#}$+Fast takes extra time. However, the cost of initial assignment is compensated later by no seeking of the best movement. Compared with direct k-way clustering, methods with bisecting strategy achieve much higher scalability. In particular, BsKM$^{\#}$(non) shows the highest scalability. It only takes less than *94* minutes to cluster *10* million vectors (in *512* dimensions) into *1024* clusters. The efficiency of Mini-Batch is close to BsKM$^{\#}$(non). However, as shown in Fig. 6(a), its quality is poor in most of the cases. Overall, k-means$^{\#}$+Fast achieves the highest speed efficiency and lowest distortion among all direct k-way clustering methods. While in the bisecting case, BsKM$^{\#}$(non) shows the best performance in terms of both speed efficiency and clustering quality. Similar to the experiments in Section 5.1, the average distortion introduced by bisecting clustering is much higher than direct k-way due to the problem of under-fitting.

In addition, the scalability of clustering methods is tested in the way that the number of clusters by varying from *1024* to *8192*, while the scale of input data is fixed to *1* million. Fig. 5(c) and (d) show the time cost of all *9* methods. Accordingly, the average distortion from all these *9* methods are presented in Fig. 6(b). As shown in the figures, for all direct k-way clustering methods, the time cost increases linearly as the number of clusters increases.

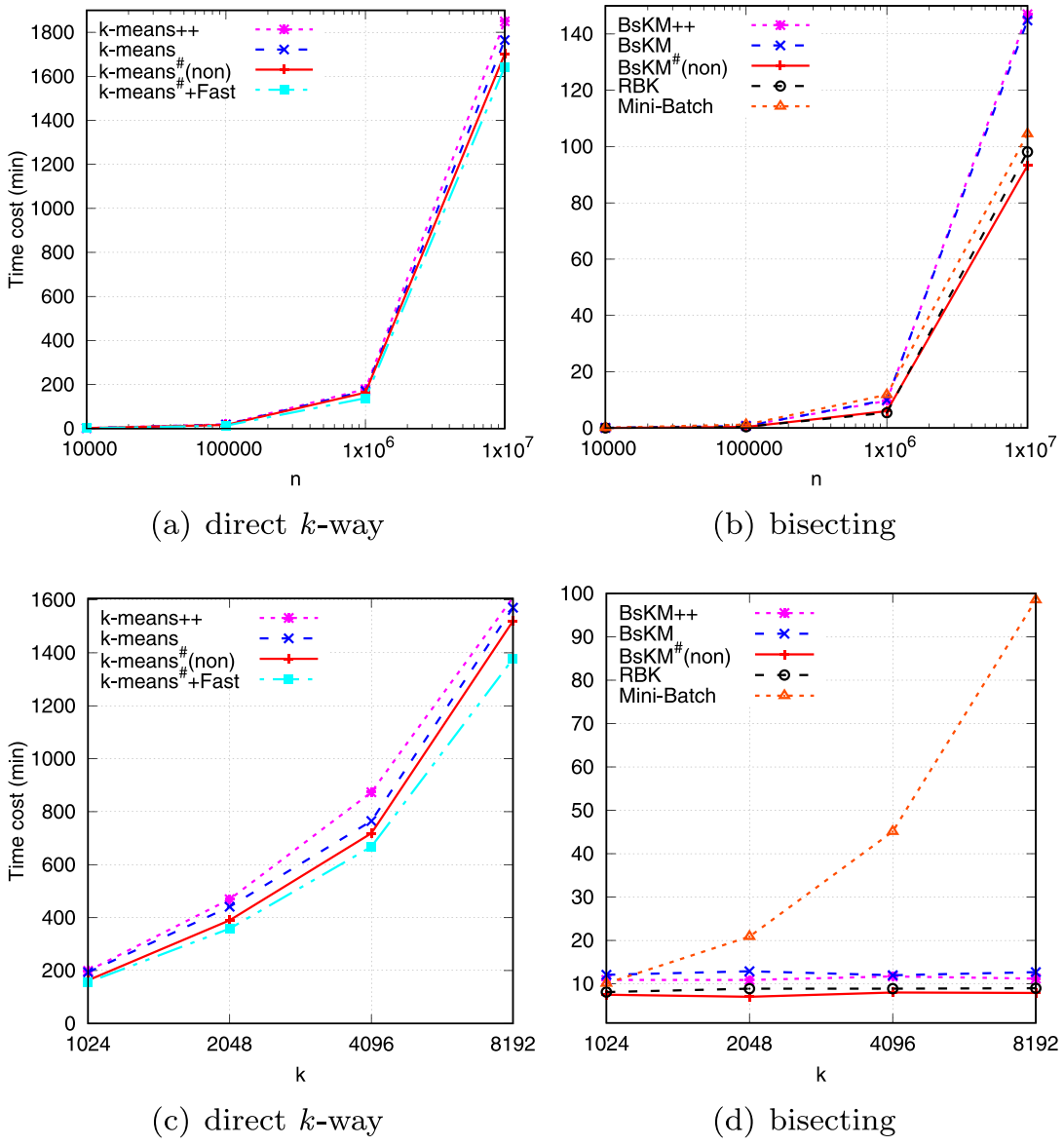


Fig. 5. Scalability test by varying the scale of input data: (a) and (b) and by varying the number of clusters: (c) and (d).

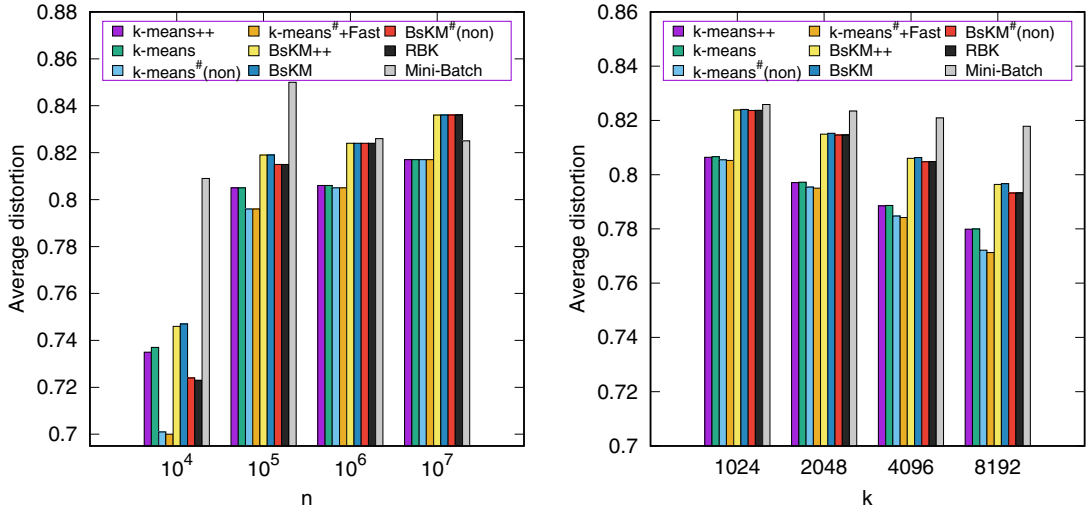


Fig. 6. Average distortion from all 9 methods under two different scalability testings on Flickr10M (best viewed in color).

Mini-Batch is no longer efficient as k increases. In contrast, the time cost of all bisecting methods remains steady across different cluster numbers. In terms of clustering quality, as seen from Fig. 6(b), in both direct k-way and bisecting cases, clustering driven by the proposed optimization procedure (Algorithm 1) performs considerably better. A clear trend is observed from Fig. 6(b), methods based on Algorithm 1 shows increasingly higher performance than the rest as k grows. Overall, clustering driven by the proposed optimization process shows higher speed and better quality. The highest speed is achieved by BsKM$^{\#}$(non), for which only *8* minutes are required to cluster *1* million high dimensional data into *8192* clusters. Due to extra cost in initial assignment, bisecting with traditional k-means and k-means++ still shows around *35%* slower speed than BsKM$^{\#}$(non).

As a summary, clustering based on Algorithm 1 shows superior performance in terms of both speed efficiency and quality under different scenarios. This is mainly due to the nature of incremental updating scheme, which allows the cluster structures to be fine-tuned in a more efficient way. When the proposed Algorithm 1 is performed under bisecting manner (i.e., BsKM$^{\#}$(non)), it shows two orders of magnitude faster than traditional k-means.

6. Conclusion

We have presented a novel k-means variant. Firstly, a clustering objective function that is feasible for the whole l_2-space is developed. Supported by the objective function, the traditional k-means clustering has been modified to simpler form. In this novel k-means variant, we interestingly find that neither the costly initial assignment nor the seeking of closest centroid for each sample in the iteration are necessary. This leads to higher speed and considerably lower clustering distortion. Furthermore when the proposed clustering method is undertaken in the ways of top-down bisecting, it achieves the highest scalability and best quality among all hierarchical k-means variants. Extensive experiments have been conducted in different contexts and on various datasets. Superior performance over most of the k-means variants is observed across different scenarios.

Acknowledgment

This work is supported by National Natural Science Foundation of China under grant no. 61572408. The authors would like to express their sincere thanks to Prof. George Karypis from University of Minnesota, USA for his detailed explanation about the implementation of repeated bisecting k-means.

Appendix A. Optimum of k-means$^{\#}$

As shown in Eqs. 13 and 9, two optimal objectives are quite similar. In this section, we show that optimal solution with respect to objective function (Eq. (9)) can be reached with incremental updating scheme presented in Algorithm 1.

Proof. For contradiction, let $A^o = \{S_1, S_2, \ldots, S_k\}$ be an optimal solution and assume that there exists one element d and clusters S_i and S_j such that $d \in S_i$. Now consider the clustering solution $A^* = \{S_1, S_2, \ldots, \{S_i - d\}, \ldots, \{S_j + d\}, \ldots, S_k\}$. Let D_i, C_i, and D_j, C_j be the composite and centroid vectors of cluster $S_i - d$ and S_j, respectively. Let $e = \mathcal{I}_1(A^o) - \mathcal{I}_1(A^*)$, then

$$
\begin{aligned}
e &= \frac{(D_i + d)'(D_i + d)}{n_i + 1} + \frac{D_j' D_j}{n_j} - \left(\frac{D_i' D_i}{n_i} + \frac{(D_j + d)'(D_j + d)}{n_j + 1} \right) \\
&= \left(\frac{(D_i + d)'(D_i + d)}{n_i + 1} - \frac{D_i' D_i}{n_i} \right) - \left(\frac{(D_j + d)'(D_j + d)}{n_j + 1} - \frac{D_j' D_j}{n_j} \right) \\
&= \frac{2n_i d' D_i + n_i d' d - D_i' D_i}{n_i(n_i + 1)} - \frac{2n_j d' D_j + n_j d' d - D_j' D_j}{n_j(n_j + 1)}
\end{aligned}
$$

Let's define $\mu_i = \frac{D_i' D_i}{n_i(n_i+1)}$, $\mu_j = \frac{D_j' D_j}{n_j(n_j+1)}$ are the average pairwise inner product in cluster S_i and S_j, respectively. In addition, δ_i and δ_j are given as the average inner-products between d and elements in S_i and S_j, respectively, viz $\delta_i = \frac{d' D_i}{n_i}$, and $\delta_j = \frac{d' D_j}{n_j}$. Above Equation is rewritten as

$$
\begin{aligned}
e &= \left(\frac{2n_i \delta_i}{n_i + 1} + \frac{d' d}{n_i + 1} - \frac{n_i \mu_i}{n_i + 1} \right) - \left(\frac{2n_j \delta_j}{n_j + 1} + \frac{d' d}{n_j + 1} - \frac{n_j \mu_j}{n_j + 1} \right) \\
&\approx \left(2\delta_i - 2\delta_j + \frac{d' d}{n_i + 1} \right) - \left(\mu_i - \mu_j + \frac{d' d}{n_j + 1} \right)
\end{aligned} \tag{19}
$$

Given the fact that $(2\delta_i - 2\delta_j + \frac{d'd}{n_i+1}) < (\mu_i - \mu_j + \frac{d'd}{n_j+1})$, we have $\mathcal{I}_1(A^o) < \mathcal{I}_1(A^*)$, which is contradicting. □

Appendix B. Convergence of k-means$^{\#}$

S_i and S_j are two clusters. d is initially part of S_i, and D_i is the composite of S_i exclude d, C_i is the centroid of S_i exclude d, D_j, C_j is the composite and centroid of cluster S_j, the move condition of d from S_i to S_j should satisfied

$$
\frac{(D_i + d)'(D_i + d)}{n_i + 1} + \frac{D_j' D_j}{n_j} < \frac{D_i' D_i}{n_i} + \frac{(D_j + d)'(D_j + d)}{n_j + 1} \tag{B.1}
$$

This equation can be rewritten as:

$$
\begin{aligned}
&\frac{(D_i + d)'(D_i + d)}{n_i + 1} - \frac{D_i' D_i}{n_i} < \frac{(D_j + d)'(D_j + d)}{n_j + 1} - \frac{D_j' D_j}{n_j} \\
&\frac{D_i' D_i + 2d' D_i + d^2}{n_i + 1} - \frac{D_i' D_i}{n_i} < \frac{D_j' D_j + 2d' D_j + d^2}{n_j + 1} - \frac{D_j' D_j}{n_j} \\
&\frac{2n_i d' D_i + n_i d^2 - D_i' D_i}{n_i(n_i + 1)} < \frac{2n_j d' D_j + n_j d^2 - D_j' D_j}{n_j(n_j + 1)} \\
&2\frac{n_i}{n_i + 1}\frac{d' D_i}{n_i} - \frac{D_i' D_i}{n_i(n_i + 1)} + \frac{d^2}{n_i + 1} < 2\frac{n_j}{n_j + 1}\frac{d' D_j}{n_j} \\
&- \frac{D_j' D_j}{n_j(n_j + 1)} + \frac{d^2}{n_j + 1}
\end{aligned}
$$

Now if we assume that both n_i and n_j are sufficiently large, then $\frac{n_i}{n_i+1}$ and $\frac{n_j}{n_j+1}$ will be close to *1*. Under these assumptions, we can get

$$
2\frac{d' D_i}{n_i} - \frac{D_i' D_i}{n_i(n_i + 1)} + \frac{d^2}{n_i + 1} < 2\frac{d' D_j}{n_j} - \frac{D_j' D_j}{n_j(n_j + 1)} + \frac{d^2}{n_j + 1}.
$$

Now $\mu_i = \frac{D_i' D_i}{n_i(n_i+1)}$, $\mu_j = \frac{D_j' D_j}{n_j(n_j+1)}$ are defined as the average pairwise inner product in cluster S_i and S_j respectively. δ_i and δ_j are given as the average inner-products between d and elements in S_i and S_j respectively, viz $\delta_i = \frac{d' D_i}{n_i}$, and $\delta_j = \frac{d' D_j}{n_j}$, the following inequation holds.

$$
2\delta_i - 2\delta_j + \frac{d' d}{n_i + 1} < \mu_i - \mu_j + \frac{d' d}{n_j + 1}. \tag{B.2}
$$

References

[1] Y. Zhao, G. Karypis, Empirical and theoretical comparisons of selected criterion functions for document clustering, Mach. Learn. 55 (2004) 311–331, doi:10.1023/B:MACH.0000027785.44527.d6.

[2] Y. Avrithis, Y. Kalantidis, E. Anagnostopoulos, I.Z. Emiris, Web-scale image clustering revisited, in: Proceedings of the ICCV, 2015, pp. 1502–1510.

[3] J. Shi, J. Malik, Normalized cuts and image segmentation, Trans. PAMI 22 (8) (2000) 888–905, doi:10.1109/34.868688.

[4] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: Proceedings of the ICCV, 2003, pp. 1470–1477, doi:10.1109/ICCV.2003.1238663.

[5] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, Trans. PAMI 33 (1) (2011) 117–128, doi:10.1109/TPAMI.2010.57.

[6] M. Muja, D.G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 2227–2240, doi:10.1109/TPAMI.2014.2321376.

[7] A. Babenko, V. Lempitsky, Additive quantization for extreme vector compression, in: Proceedings of the CVPR, 2014, pp. 931–938, doi:10.1109/CVPR.2014.124.

[8] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (1982) 129–137, doi:10.1109/TIT.1982.1056489.

[9] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2007) 1–37, doi:10.1007/s10115-007-0114-2.

[10] N. Ailon, R. Jaiswal, C. Monteleoni, Streaming *k*-means approximation, in: Proceedings of the NIPS, 2009, pp. 10–18.

[11] A. Vattani, *k*-means requires exponentially many iterations even in the plane, Discret. Comput. Geom. 45 (4) (2011) 596–616, doi:10.1007/s00454-011-9340-1.

[12] D. Arthur, S. Vassilvitskii, *k*-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.

[13] M. Shindler, A. Wong, A.W. Meyerson, Fast and accurate *k*-means for large datasets, in: Proceedings of the NIPS, 2011, pp. 2375–2383.

[14] T. Kanungo, D.M. Mount, N.S. Neanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient *k*-means clustering algorithm: analysis and implementation, Trans. PAMI 24 (7) (2002) 881–892, doi:10.1109/TPAMI.2002.1017616.

[15] C. Elkan, Using the triangle inequality to accelerate, in: Proceedings of the ICML, 2003.

[16] D. Sculley, Web-scale *k*-means clustering, in: Proceedings of the Nineteenth International Conference on World Wide Web, 2010, pp. 1177–1178, doi:10.1145/1772690.1772862.

[17] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable *k*-means++, in: Proceedings of the VLDB Endowment, 5, 2012, pp. 622–633, doi:10.14778/2180912.2180915.

[18] A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii, S. Venkatesan, Scalable *k*-means by ranked retrieval, in: Proceedings of the Seventh ACM International Conference on Web Search and Data Mining, 2014, pp. 233–242, doi:10.1145/2556195.2556260.

[19] J. Wang, J. Wang, Q. Ke, G. Zeng, S. Li, Fast approximate *k*-means via cluster closures, Multimed. Data Min. Anal. (2015) 373–395, doi:10.1007/978-3-319-14998-1_17.

[20] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Boundev, R. Fergus, Web scale photo hash clustering on a single machine, in: Proceedings of the CVPR, 2015, pp. 19–27, doi:10.1109/CVPR.2015.7298596.

[21] L. Bottou, Y. Bengio, Convergence properties of the *k*-means algorithm, Adv. Neural Inf. Process. Syst. (1995) 585–592.

[22] M. Ester, H. peter Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[23] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, Trans. PAMI 24 (5) (2002) 603–619, doi:10.1109/34.1000236.

[24] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[25] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, s, Wiley-Interscience, 2001.

[26] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323, doi:10.1145/331499.331504.

[27] X. Cui, P. Zhu, X. Yang, K. Li, C. Ji, Optimized big data *k*-means clustering using MapReduce, J. Supercomput. 70 (2014) 1249–1259.

[28] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517, doi:10.1145/361002.361007.

[29] D. Pelleg, A. Moore, Accelerating exact *k*-means algorithms with geometric reasoning, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 277–281, doi:10.1145/312129.312248.

[30] A. Goswami, R. Jin, G. Agrawal, Fast and exact out-of-core *k*-means clustering, in: Proceedings of the Fourth IEEE International Conference on Data Mining, 2004, pp. 83–90, doi:10.1109/ICDM.2004.10102.

[31] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., 1988.

[32] Y. Zhao, G. Karypis, Hierarchical clustering algorithms for document datasets, Data Min. Knowl. Discov. 10 (2) (2005) 141–168, doi:10.1007/s10618-005-0361-3.

[33] T. Kohonen, M.R. Schroeder, T.S. Huang (Eds.), Self-Organizing Maps, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[34] D. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110, doi:10.1023/B:VISI.0000029664.99615.94.

[35] B. Thomee, D.A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, L.-J. Li, YFCC100M: the new data in multimedia research, Commun. ACM 59 (2) (2016) 64–73, doi:10.1145/2812802.

[36] K. Mikolajczyk, C. Schmid, Scale and affine invariant interest point detectors, IJCV 60 (1) (2004) 63–86, doi:10.1023/B:VISI.0000027790.02288.f2.

[37] R. Arandjelovic, A. Zisserman, Three things everyone should know to improve object retrieval, in: Proceedings of the CVPR, 2012, pp. 2911–2918.

[38] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, C. Schmid, Aggregating local descriptors into compact codes, Trans. PAMI 34 (9) (2012) 1704–1716, doi:10.1109/TPAMI.2011.235.

Wan-Lei Zhao received his Ph.D degree from City University of Hong Kong in 2010. He received M.Eng. and B.Eng. degrees in Department of Computer Science and Engineering from Yunnan University in 2006 and 2002, respectively. He currently works with Xiamen University as an associate professor, China. Before joining Xiamen University, he was a Postdoctoral Scholar in INRIA, France. His research interests include multimedia information retrieval and video processing.

Cheng-Hao Deng received his Bachelor degree of Science from Nanchang University, China in 2014. He is currently a graduate student at Department of Computer Science, Xiamen University. His research interest is large-scale image clustering and linking.

Chong-Wah Ngo received the B.Sc. and M.Sc. degrees in computer engineering from Nanyang Technological University, Singapore, and the Ph.D. in computer science from the Hong Kong University of Science and Technology (HKUST), Hong Kong. He is currently a Professor with the Department of Computer Science, City University of Hong Kong. Before joining the City University, he was a Postdoctoral Scholar with the Beckman Institute, University of Illinois at Urbana Champaign (UIUC), Urbana, IL, USA. He was also a Visiting Researcher with Microsoft Research Asia, Beijing, China. His research interests include large-scale multimedia information retrieval, video computing, multimedia mining and visualization.