

# Boost K-Means

Wan-Lei Zhao, Cheng-Hao Deng, and Chong-Wah Ngo, *Senior Member, IEEE*

**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.

**Index Terms**—clustering, k-means, incremental optimization



## 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], [6], [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 [14], [15], [16], [13], [17], [18]. K-means has been also tailored to perform web-scale image clustering [2], [19].

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  $C_{1\dots k} \in R^d$  are cluster centroids,  $\{x_i \in R^d\}_{i=1\dots n}$  are samples to be clustered, above procedure essentially minimizes the following objective function:

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

duration essentially minimizes the following objective function:

In Eqn. 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 [20]. 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 [21] and Mean shift [22], 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 ( $m$ ), 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 initialization procedure. However, k-means is slow 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 Eqn. 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], [23]. These modifications lead to several advantages.

- Fujian Key Laboratory of Sensing and Computing for Smart City, and the School of Information Science and Engineering, Xiamen University, Xiamen, 361005, P. R. China.  
E-mail: wlzhao@xmu.edu.cn
- Department of Computer Science, City University of Hong Kong.  
E-mail: cscwngo@gapps.cityu.edu.hk

- K-means clustering without initial assignment results in better quality as well as higher speed effi-

ciency.

- 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 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 Eqn. 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 [24]. 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], [17], [18] 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 Ostrovsky 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. Noticed that this is a nearest neighbor search problem, Kanungo et al. [14] proposed to index dataset in a KD Tree [25] 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 Dan

et al. [26]. 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. Attributing 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], [27] 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 samples and centroids, namely performing clustering in a top-down hierarchical manner [1], [28], [29]. 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], [29] is that cluster centroids are updated incrementally [1], [23]. Moreover, the update process is explicitly driven by an objective function (called as criterion function in [1], [29]). Unfortunately, objective functions proposed in [1], [28], [29] 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 Eqn. 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  $\{S_1, \dots, S_r, \dots, S_k\}$ . Accordingly, the sizes of clusters are given as  $n_1, \dots, n_r, \dots, n_k$ . The composite vector of a cluster is defined as  $D_r = \sum_{x_i \in S_r} x_i$ . The cluster centroid  $C_r$  is defined by its members,

$$C_r = \frac{\sum_{i=1}^{n_r} x_i}{n_r} = \frac{D_r}{n_r} \quad (2)$$

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.

$$\begin{aligned} C_r' C_r &= \frac{1}{n_r^2} [(x_1' x_1 + \dots + x_1' x_i + \dots + x_1' x_{n_r}) + \\ &\quad (x_2' x_1 + \dots + x_2' x_i + \dots + x_2' x_{n_r}) + \\ &\quad \dots \\ &\quad (x_i' x_1 + \dots + x_i' x_i + \dots + x_i' x_{n_r}) + \\ &\quad \dots \\ &\quad (x_{n_r}' x_1 + \dots + x_{n_r}' x_i + \dots + x_{n_r}' x_{n_r})] \\ &= \frac{1}{n_r^2} (\sum_{i=1}^{n_r} x_i^2 + 2 \sum_{i,j=1 \& i < j}^{n_r} \langle x_i, x_j \rangle) \end{aligned}$$

Re-arrange the above equation, we have

$$\sum_{i,j=1 \& i < j}^{n_r} \langle x_i, x_j \rangle = \frac{1}{2} (n_r^2 \cdot C_r' C_r - \sum_{i=1}^{n_r} x_i^2). \quad (3)$$

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

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

Plug Eqn. 3 into Eqn. 4, we have

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

Eqn. 5 is rewritten as

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

### 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

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

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 considers the tightness of clusters, while another focuses on alienating different clusters. In this paper, we focus 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.

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

The above equation is simplified as

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

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

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

Although objective function in Eqn. 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.

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

Plug Eqn. 6 in, we have

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

In Eqn. 11,  $\frac{n_r}{n_r-1}$  is close to 1, the above criterion function can be approximated as

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

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

$$\text{Max. } \mathcal{I}_2^* \approx \sum_{r=1}^k \frac{D'_r D_r}{n_r}. \quad (13)$$

Noticed that similar optimization objectives have been discussed under *Cosine* similarity measure in [1]. As it is shown that above two objective functions are the same in the paper. This is different from the result obtained in our case (general  $l_2$ -space). As shown above, in  $l_2$ -space, the objective functions for  $\mathcal{I}_1^*$  and  $\mathcal{I}_2^*$  are only 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 meantime. 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 Eqn. 9 is derived from Eqn. 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 Eqn. 9 attains a higher score (implies lower distortion in terms of Eqn. 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], [23] and driven by objective function  $\mathcal{I}_1^*$  (Eqn. 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.

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

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 Alg. 1.

As seen from **Step 3** of Alg. 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 highest increase of function score. From the optimization point of view, the algorithm reduces the clustering distortion greedily. From another point of 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.

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) [30], in which the cluster centroids are updated incrementally. In the above iteration procedure, no cluster centroids are explicitly produced. As a result, there is no need to update cluster centroid. The clustering iteration is explicitly driven by an objective function rather than driven by the discrepancy between cluster centroids and their cluster members. As revealed later in the experiment, compared to LVQ, Alg 1 is more efficient and leads to considerably lower distortion.

**Algorithm 1.** Direct  $k$ -way Boost k-means

- 1: **Input:** matrix  $X_{n \times d}$
- 2: **Output:**  $S_1, \dots, S_r, \dots, S_k$
- 3: Assign  $x_i \in X$  with a random cluster label;
- 4: Calculate  $D_1, \dots, D_r, \dots, D_k$  and  $\mathcal{I}_1^*$ ;
- 5: **while** not convergence **do**
- 6:   **for** each  $x_i \in X$  (in random order) **do**
- 7:      $x'_i x_i$  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**

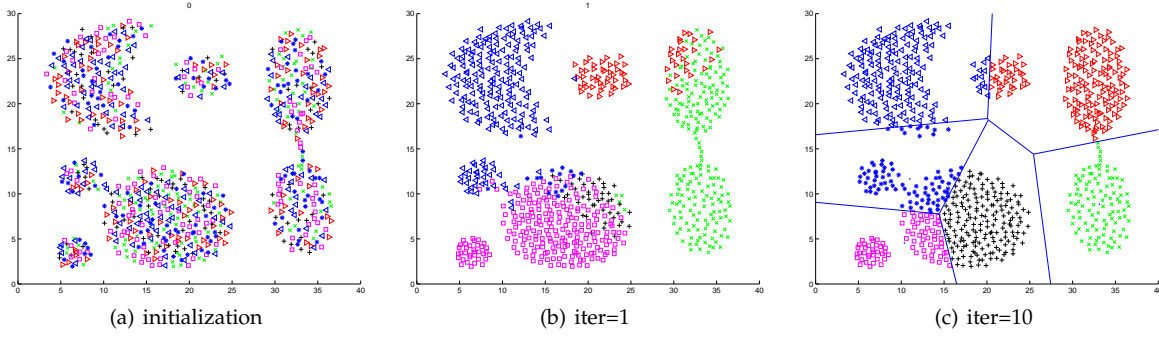


Fig. 1. Illustration of direct k-way k-means clustering with Alg. 1. The clustering process starts from the state that samples are all assigned with random label. The final cluster centroids in (c) form a convex partition over the 2D space, which are called as *Voronoi* diagram. According to *Lloyd's*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

12: **end while**  
**end**

Figure 1 illustrates three iterations of Alg. 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 10th round, where *Lloyd's* condition is reached. The optimality of this procedure is analyzed in Appendix A and its convergence is proved in Appendix B.

Overall, method presented in Alg. 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.

The method presented in Alg. 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.

The method is revised into a top-down hierarchical clustering version for large-scale clustering. Specifically, at each time, one intermediate cluster is selected and bisected into two smaller clusters by calling Alg. 1. The details of this method are given in Alg. 2.

As shown in Alg. 2, priority queue  $Q$  pops out one cluster for bisecting each time. As discussed in [29], there are basically two ways to organize the priority queue. One can prioritize the cluster with biggest size or the one with highest average intra-cluster distance to split. Similar as [29], 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.

**Algorithm 2.** Bisecting Boost k-means

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

- 9: Push  $S_i, S_{i+1}$  into queue  $Q$ ;
- 10:  $i = i + 1$ ;
- 11: **end while**

**end**

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 Alg. 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 Alg. 2 is partitioned evenly. In addition, we generalize Alg. 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.

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

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

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

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

$$n * (s - 1) * \log_s k. \quad (16)$$

Given  $n$  and  $k$  are fixed, Eqn. 16 increases monotonically with respect to  $s$ . As a result, the number of comparisons

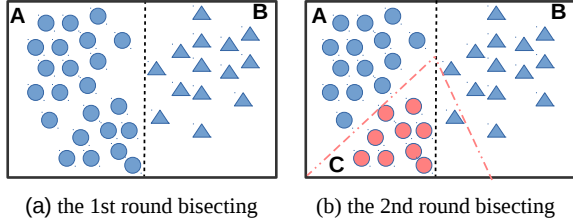


Fig. 2. Illustration of two consecutive bisecting in the bisecting clustering where *Lloyd's* condition breaks.

reaches to the minimum (i.e.,  $n \log_2 k$ ) when  $s = 2$ . To this end, it is clear that bisecting is the most efficient setting scheme.

Compared with Alg. 1, the complexity of Alg. 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 Alg. 1 has been largely reduced since term  $n \cdot d$  has been multiplied by a much smaller factor  $\bar{t} \cdot \log(k)$ .

Although Alg. 2 is efficient, the clustering result produced by Alg. 2 unfortunately does not satisfy with *Lloyd's* condition. This problem is illustrated in Figure 2. As one of the clusters is further partitioned into two (from Figure 2(a) to Figure 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 Alg. 1 as a refinement procedure after Alg. 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 boost k-means. After a few iterations, both k-means and boost k-means will be trapped in a local minima. Only samples that bordering between different clusters should be 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, sample only needs to compare with top- $k_0$  ( $k_0 \ll k$ ) centroids to search the suitable movement. We find that, this simple modification typically leads to 7~8 times speed-up while without significant performance degradation.

## 5 EXPERIMENTS

In this section, the effectiveness of proposed clustering method, namely boost k-means (BKM) is studied under different scenarios. In the first experiment, dataset SIFT1M [5] is adopted to evaluate the clustering quality. In the second experiment, BKM is tested on the nearest neighbor search task based on product quantizer (PQ) [5] in which this method is adopted for quantizer training. In the third experiment, BKM has been applied to traditional document clustering. Following the practice of [1], [29], 15 document datasets<sup>2</sup> have been adopted. In the last experiment, the

scalability of BKM 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) [29], online Learning Vector Quantization (LVQ) [30] 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  $4 \times 10^{-4}$  in one iteration.

As shown in Table 1, there are variants of k-means depending on cluster initialization and data partitioning strategies (e.g., direct k-way or bisecting). This is also true for the proposed BKM. In the table, 'initial assignment' refers to the operation of assigning each sample to its closest initial centroid. When the initial assignment is based on random seeding like traditional k-means, it is denoted as 'rnd'. When it is based on probability distribution seeding as k-means++, it is denoted as 'kpp'. Initialization without initial assignment is denoted as 'non'. In the experiments, all the variants out of these different configurations on k-means as well as BKM are considered. Performance evaluation is separately conducted for k-way and bisecting clustering method. Noted that BsBKM(rnd) is the same as RBK if the input data is  $l_2$ -normalized. The experiment in this section is conducted using 1 million SIFT features [31]. The features are clustered into 10,000 partitions and the average distortion error is calculated for performance evaluation.

In addition, we also study the performance trend of BKM when Steps 7-10 in Alg. 1 are modified to moving the sample as soon as  $\Delta I_1(x_i) > 0$ . The variants under this modification are denoted as BKM(XXX)+Fast.<sup>3</sup> All the methods considered in the paper are implemented in C++ and the simulations are conducted on a PC with 2.4GHz Xeon CPU and 32G memory setup.

### 5.1 Evaluation of Clustering Distortion

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

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

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

The first experiment mainly studies the behavior of the proposed BKM under different initializations. The average distortion curves produced by variants direct k-way BKM

2. Available at <http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/datasets.html> that this is not applicable for bisecting BKM.

TABLE 1  
Configurations of k-means and its variants and their corresponding Abbreviations

Initial assignment	k-means		boost k-means	
	k-way	bisecting	k-way	bisecting
Random	k-means [8]	BsKM	BKM(rnd)	BsBKM(rnd)
Probability based [12]	k-means++ [12]	BsBKM++	BKM(kpp)	BsBKM(kpp)
None	-	-	BKM(non)	BsBKM(non)

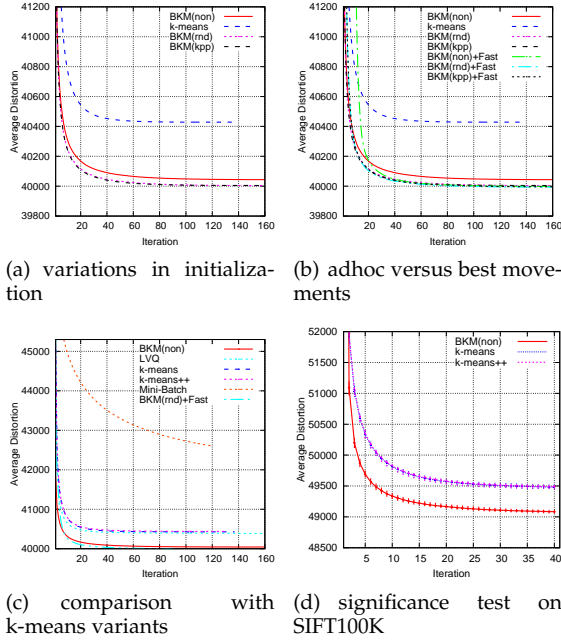


Fig. 3. The experiments are conducted on SIFT1M for figures (a)-(c) and on SIFT100K for figure (d). The results show different performance: (a) impact of initialization in different ways; (b) fast version of BKM by not seeking optimal movement in the steps 7-10 of Alg 1; (c) Comparison of BKM to variants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

are given in Figure 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 BKM drops faster than traditional k-means. The average distortion from traditional k-means is around 40,450 after 130 iterations. In contrast, BKM without initial assignment (BKM(non)) is able to reach to the same distortion level after only 7 iterations. Moreover, we find that initializing BKM as traditional k-means way (BKM(rnd)) or as k-means++ (BKM(kpp)) allows the iteration to start from a low distortion level. Nevertheless the advantage over BKM(non) fades away after 15 iterations. In comparison to BKM(non), the extra cost of adopting initial assignment in BKM is relatively high.

The second experiment studies the performance trend of Alg. 1 in case when **Steps 7-10** do not seek the best movement (BKM(xxx)+Fast). As shown in Figure 3(b), the distortion drops slower than BKM(non) which seeks the best movement. However, lower distortion is achievable by BKM(rnd)+Fast when reaching to sufficiently number of iterations (e.g., 20 iterations). This indicates that when the optimization scheme is more greedy, it is likely to get trapped in a worse local optima. This observation applies

to BKM under different kinds of initialization. Noted that the time cost for BKM(xxx)+Fast is lower than that of BKM that seeks the best movement in each iteration. Whereas, BKM(xxx)+Fast usually needs a few more number of iterations to reach to the similar distortion level. Overall, as investigated in Section 5.4, BKM(xxx)+Fast is %5 faster than BKM(xxx).

Figure 3(c) studies the trend of average distortion among the proposed BKM (specifically BKM(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. BKM(non) outperforms k-means and k-means++ by requiring 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 possibility of terminating the iteration at 20. Although similar as BKM, 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 Figure 3(d) further confirms the significance of the improvement achieved by BKM. 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 BKM 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  $4.5 \times 10^4$ . Methods built upon Alg. 1 always perform 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 Alg. 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 BKM slightly.

TABLE 2  
Average Distortion from K-means Variants under Bisecting Strategy

Method	k-means	RBK	BsKM	BsKM++	BsBKM(non)	BsBKM(rnd)	BsBKM(kpp)
$\mathcal{E}$	40,450.0	45,713.5	45,835.2	45,823.8	<b>45,650.7</b>	45,661.2	45,658.4
$\mathcal{E}$ after Rfn.	-	43,364.4	4,3323.9	43,366.2	43,293.3	43,285.5	<b>43,285.4</b>

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 BKM, namely BKM(non) and BKM(rnd)+Fast are presented. We leave the other possible configurations to the readers.

## 5.2 Nearest Neighbor Search by Product Quantizer (PQ)

In this section, BKM is applied for visual vocabulary training using product quantization [5]. Following the practice of [5], 100K 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 1,000 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 Figure 4(a)-(d) under two different settings ( $m = 8$  and  $m = 16$ ). As seen from the figures, the performances from k-means, k-means++ and BKM(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 tolerant to 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 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 has been represented with *TF/IDF* model and normalized to unit length. Similar to [1], entropy is adopted for the evaluation as following

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

where  $c$  is the number of classes. Eqn. 18 evaluates to what degree that elements from the same class are put in one cluster. The lower the value, the better the result is. 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.

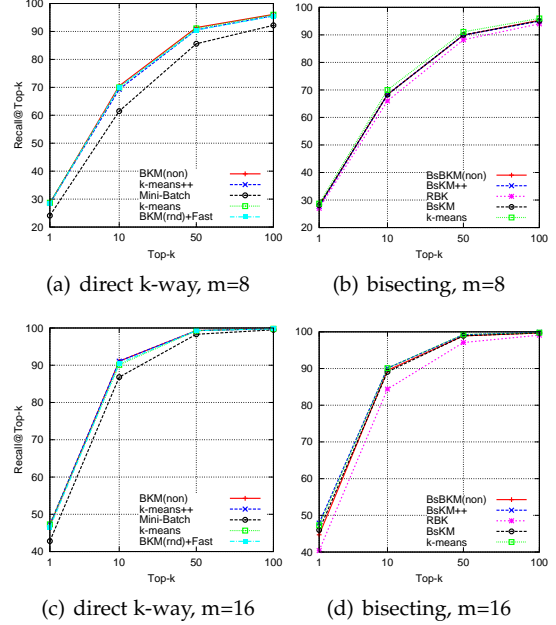


Fig. 4. Performance of Nearest Neighbor Search by PQ on SIFT1M when adopting different clustering methods for quantizer training. The size of each product quantizer is fixed to 256 across all the experiments. The assymmetric distance calculation (ADC) [5] is adopted for nearest neighbor search.

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
BKM(non)	0.552	0.442	0.388	0.368
BKM(rnd)+Fast	<b>0.506</b>	<b>0.419</b>	<b>0.380</b>	<b>0.353</b>
BsKM	0.532	0.438	0.410	0.373
BsKM++	0.507	0.422	0.400	0.379
BsBKM(non)	0.514	<b>0.388</b>	<b>0.353</b>	<b>0.329</b>
RBK	<b>0.486</b>	0.402	0.366	0.339

In general, methods based on BKM perform considerably better. Furthermore, methods with bisecting strategy demonstrate slightly better performance than that of direct k-way in the document clustering task, which shares similar observation as [29]. Overall, BsBKM(non) shows the best performance. While the performance of RBK is close to BsBKM(non). These two methods are quite similar except that no initial assignment is involved in BsBKM(non). This indicates the advantage of no initial assignment in this scenario, which allows clustering to converge to a better local optima.



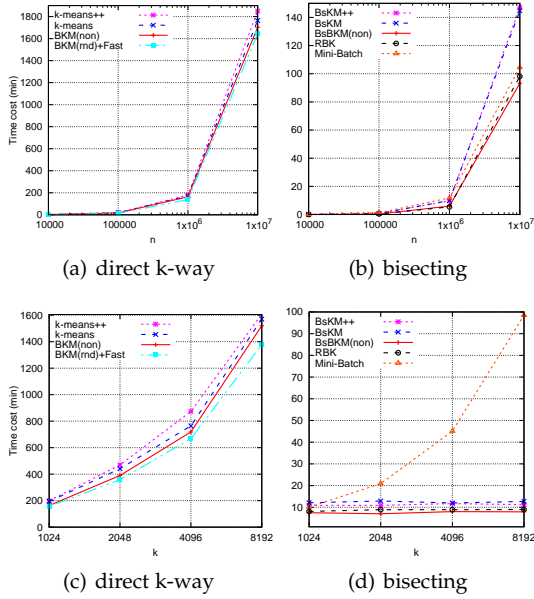


Fig. 5. Scalability test by varying the scale of input data: (a)-(b) and by varying the number of clusters: (c)-(d).

#### 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* [32]. *Hessian-Affine* [33] keypoints are extracted from each image and are described by RootSIFT feature [34]. Finally, the RootSIFT features from each image are pooled by VLAD [35] with a small visual vocabulary of size 64. The resulting 8,192-dimensional feature is further mapped to 512 dimensions by PCA. Following [35], 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 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. A fixed number of clusters, i.e., 1,024 is used regardless of the size of dataset. The time costs for direct k-way and bisecting methods are presented in Figure 5(a)-(b). Accordingly, the average distortion of all the methods are presented in Figure 6(a).

As shown in the figures, BKM 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 no requirement of initial assignment. Compared with BKM(non), BKM(rnd)+Fast takes extra time. However, the cost of initial assignment is compensated later for not seeking the best movement. Compared with direct k-way clustering, methods with bisecting strategy achieve much higher scalability. In particular, BsBKM(non) shows the highest scalability. It only takes less than 94 minutes to cluster 10 million vectors (in 512 dimensions) into 1,024 clus-

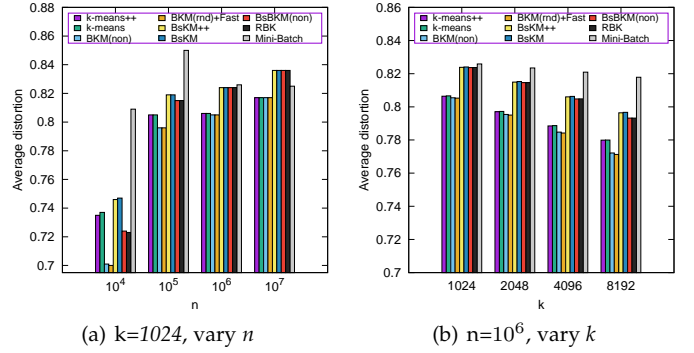


Fig. 6. Average distortion from all 9 methods under two different scalability testings on Flickr10M (best viewed in color).

ters. The efficiency of Mini-Batch is close to BsBKM(non). However, as shown in Figure 6(a), the clustering quality is poor in most of the cases. Overall, BKM(rnd)+Fast achieves the highest speed efficiency and lowest distortion among all direct k-way clustering methods. While in the bisecting case, BsBKM(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 1,024 to 8,192, while the scale of input data is fixed to 1 million. Figure 5(c)-(d) show the time cost of all 9 methods. Accordingly, the average distortion from all these 9 methods are presented in Figure 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. 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 Figure 6(b), in both direct k-way and bisecting cases, clustering driven by the proposed optimization procedure (Alg. 1) performs considerably better. A clear trend is observed from Figure 6(b), methods based on Alg. 1 shows increasingly higher performance than the rest as  $k$  grows. Overall, clustering driven by proposed optimization process shows higher speed and better quality. The highest speed is achieved by BsBKM(non), for which only 8 minutes are required to cluster 1 million high dimensional data into 8,192 clusters. Due to extra cost in initial assignment, bisecting with traditional k-means and k-means++ still shows around 35% slower speed than BsBKM(non).

As a summary, clustering based on Alg. 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 Alg. 1 is performed under bisecting manner (i.e., BsBKM(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.

## ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China under grants 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: OPTIMALITY OF INCREMENTAL K-MEANS

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

6.0.0.1 Proof: : For contradiction, let  $A^\circ = \{S_1, S_2, \dots, S_k\}$  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  $A^* = \{S_1, S_2, \dots, \{S_i - d\}, \dots, \{S_j + d\}, \dots, S_k\}$ . 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  $e = \mathcal{I}_1(A^\circ) - \mathcal{I}_1(A^*)$ , then

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

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,  $\delta_i$  and  $\delta_j$  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

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

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  $\mathcal{I}_1(A^\circ) < \mathcal{I}_1(A^*)$ , which is contradicting. ■

## APPENDIX B: CONVERGENCE OF INCREMENTAL 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

$$\frac{(D_i + d)'(D_i + d)}{n_i + 1} + \frac{D_j' D_j}{n_j} < \frac{D_i' D_i}{n_i} + \frac{(D_j + d)'(D_j + d)}{n_j + 1} \quad (20)$$

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.  $\delta_i$  and  $\delta_j$  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.

$$2\delta_i - 2\delta_j + \frac{d' d}{n_i + 1} < \mu_i - \mu_j + \frac{d' d}{n_j + 1}. \quad (21)$$

## REFERENCES

- [1] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Machine Learning*, vol. 55, pp. 311–331, 2004.
- [2] Y. Avrithis, Y. Kalantidis, E. Anagnostopoulos, and I. Z. Emiris, "Web-scale image clustering revisited," in *ICCV*, pp. 1502–1510, 2015.
- [3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Trans. PAMI*, vol. 22, pp. 888–905, Aug. 2000.
- [4] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, pp. 1470–1477, Oct. 2003.
- [5] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *Trans. PAMI*, vol. 33, pp. 117–128, Jan. 2011.
- [6] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, pp. 2227–2240, 2014.
- [7] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *CVPR*, pp. 931–938, 2014.
- [8] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, pp. 129–137, 1982.
- [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, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information System*, vol. 14, pp. 1–37, Dec. 2007.
- [10] N. Ailon, R. Jaiswal, and C. Monteleoni, "Streaming k-means approximation," *NIPS*, pp. 10–18, 2009.
- [11] A. Vattani, "k-means requires exponentially many iterations even in the plane," *Discrete and Computational Geometry*, vol. 45, no. 4, pp. 596–616, 2011.

- [12] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.
- [13] M. Shindler, A. Wong, and A. W. Meyerson, "Fast and accurate k-means for large datasets," in *NIPS*, pp. 2375–2383, 2011.
- [14] T. Kanungo, D. M. Mount, N. S. Neenyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *Trans. PAMI*, vol. 24, pp. 881–892, Jul. 2002.
- [15] C. Elkan, "Using the triangle inequality to accelerate," in *ICML*, 2003.
- [16] D. Sculley, "Web-scale k-means clustering," in *In Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, 2010.
- [17] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *In Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.
- [18] A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii, and S. Venkatesan, "Scalable k-means by ranked retrieval," in *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 233–242, 2014.
- [19] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Boundev, and R. Fergus, "Web scale photo hash clustering on a single machine," in *CVPR*, pp. 19–27, 2015.
- [20] L. Bottou and Y. Bengio, "Convergence properties of the kmeans algorithm," *Advances in Neural Information Processing Systems*, pp. 585–592, 1995.
- [21] M. Ester, H. Peter Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *In Proceedings of Knowledge Discovery and Data Mining*, pp. 226–231, 1996.
- [22] D. Comaniciu and P. Mee, "Mean shift: A robust approach toward feature space analysis," *Trans. PAMI*, vol. 24, pp. 603–619, May 2002.
- [23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2 ed., 2001.
- [24] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computer Survey*, vol. 31, pp. 264–323, Sep. 1999.
- [25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of ACM*, vol. 18, pp. 509–517, Sep. 1975.
- [26] D. Pelleg and 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*, pp. 277–281, ACM, 1999.
- [27] A. Goswami, R. Jin, and G. Agrawal, "Fast and exact out-of-core k-means clustering," in *Proceedings of the Fourth IEEE International Conference on Data Mining*, pp. 83–90, 2004.
- [28] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.
- [29] Y. Zhao and G. Karypis, "Hierarchical clustering algorithms for document datasets," *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, 2005.
- [30] T. Kohonen, M. R. Schroeder, and T. S. Huang, eds., *Self-Organizing Maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 3rd ed., 2001.
- [31] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, Nov. 2004.
- [32] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "YFCC100M: The new data in multimedia research," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016.
- [33] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *IJCV*, vol. 60, pp. 63–86, Oct. 2004.
- [34] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in *CVPR*, pp. 2911–2918, Jun. 2012.
- [35] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local descriptors into compact codes," *Trans. PAMI*, vol. 34, pp. 1704–1716, Sep. 2012.