# An Effective and Efficient Algorithm for K-Means Clustering With New Formulation

Feiping Nie, Ziheng Li, Rong Wang, and Xuelong Li, *Fellow, IEEE*

**Abstract**—K-means is one of the most simple and popular clustering algorithms, which implemented as a standard clustering method in most of machine learning researches. The goal of K-means clustering is finding a set of cluster centers and minimizing the sum of squared distances between each sample and its nearest clustering center. In this paper, we proposed a novel K-means clustering algorithm, which reformulate the classical K-Means objective function as a trace maximization problem and then replace it with a new formulation. The proposed algorithm does not need to calculate the cluster centers in each iteration and requires fewer additional intermediate variables during the optimization process. In addition, we proposed an efficient iterative re-weighted algorithm to solve the involved optimization problem and provided the corresponding convergence analysis. The proposed algorithm keeps a consistent computational complexity as Lloyd's algorithm, $\mathcal{O}(ndk)$, but shows a faster convergence rate in experiments. Extensive experimental results on real world benchmark datasets show the effectiveness and efficiency of the proposed algorithm.

**Index Terms**—Clustering, K-means, optimization, re-weighted

## 1 Introduction

Clustering is a fundamental machine learning problem, and has a long and rich history in a variety of scientific fields such as pattern recognition,machine learning, bioinformatics and image processing [1], [2], [3], [4]. In spite of the fact that there is a large number of clustering algorithms have been proposed, K-means clustering, as one of the most simple and well-studied clustering algorithms [5], is still been widely used [6], [7]. The goal of K-means clustering is assigning the data set into $k$ clusters such that the sum of the squared distances between each point and its nearest clustering center is minimized. Because of the advantages of simplicity, efficiency and stable performance, K-means clustering always be implemented as a standard clustering method in lot of machine learning researches [8], [9].

K-means clustering aims at finding $k$ cluster centers, such that the sum of squared distances between all points and the corresponding closest centers is minimized. Unfortunately, it is an NP-hard problem even with just two clusters [10], [11], [12]. Lloyd [13] proposed a local search algorithm for K-means clustering, which becomes one of the most popular clustering algorithms and has been used in extensive scientific and industrial applications. Lloyd's algorithm begins with $k$ arbitrary centers, such as $k$ randomly selected samples from the data set. Then, assigns each sample to the nearest center, and updates the $k$ centers by calculating the centers of the new clusters. Repeat the steps of assigning samples and calculating centers until the algorithm converges. Lloyd's algorithm is simple and effective, but it still suffers from several drawbacks. First, Lloyd's algorithm is a heuristic method. Second, Lloyd's algorithm is highly sensitive to the initialization of the $k$ centers, and it always converges to a local minimum and easily gets trapped in poor local solution [14], [15] with a bad initialization. Third, it needs to calculate the distance between each sample and all the centers in each iteration, which will poses new challenges to computation and storage cost.

A variety of algorithms have been developed for these issues. One line of researches focuses on the sensitivity to the initialization of K-means clustering algorithms. K-means clustering algorithm always finitely converges to a local minimum, and a poor initialization may leads to a bad clustering performance and an exponential running time in the worst cases. Lozano *et al.* [16] confirms that different initialization will affect the performance of K-means clustering algorithm. Therefore, how to choose a better initialization becomes a important issue to a lot of the existing researches [17], [18], [19], [20], [21], [22]. These researches believe that a good initialization can help K-means clustering algorithms to find a better solution with less iterations. The simplest strategy of initialization is randomly select a set of samples as cluster centers. However, the performance of the K-means clustering can not be guaranteed due to the uncertainty of random selection. Except for random initialization, as widely reported in the literature, the most representation method is the K-means++ method [23]. K-mean++ method uses a seeding strategy that preserves the diversity of seeds while being robust to outliers. Besides, Balanced K-means

• *The authors are with the School of Computer Science, School of Artificial Intelligence, Optics and Electronics (iOPEN), and the Key Laboratory of Intelligent Interaction and Applications (Ministry of Industry and Information Technology), Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China. E-mail: {feipingnie, liziheng09}@gmail.com, wangrong07@tsinghua.org.cn, li@nwpu.edu.cn.*

*Manuscript received 30 Oct. 2020; revised 21 Apr. 2021; accepted 29 Oct. 2021. Date of publication 1 Mar. 2022; date of current version 7 Mar. 2023.*
*This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101902, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2021JM-071, in part by the National Natural Science Foundation of China under Grants 61936014 and 61772427, and in part by the Fundamental Research Funds for the Central Universities under Grant G2019KY0501.*
*(Corresponding author: Feiping Nie.)*
*Recommended for acceptance by M. Wang.*
*Digital Object Identifier no. 10.1109/TKDE.2022.3155450*

1041-4347 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

based Hierarchical K-means(BKHK) [24] segments the samples into two balanced clusters iteratively, and adopts balanced binary tree structure to generates representative cluster centers. It has been widely used in the selection of anchor points in spectral clustering. Based on the effectiveness and efficiency of BKHK, it also can be used to select the initial cluster centers for K-means clustering algorithms with large-scale data sets.

Another line of researches aims to reduce the computational and storage consumption of K-means clustering algorithms. Some of them try to reduce the computational consumption by avoiding unnecessary distance calculations or accelerate this process [25], [26], [27]. Reducing unnecessary distance calculation can effectively reduce the calculational consumption caused by samples assignment in each iteration. For instance, Newling *et al.* [28] proposed a tight bounds technique, to eliminates further redundant distance calculations. Besides, dimensionality reduction is also commonly used to accelerating the process of K-means clustering [29], [30]. Dimensionality reduction based methods try to accelerate K-means clustering algorithms by reducing the dimension of the data points. In general, dimensionality reduction based methods can be divided into two groups: feature extraction based method and feature selection based method. For feature extraction based method, both singular value decomposition and principal component analysis have been used to preprocess the data for K-means clustering algorithms [29], [31]. Feature selection based methods try to construct the lower dimensional space of the datasets with a subset of representation features [32], [33], [34]. In addition, some algorithms use binary coding [35] or data structure [36], [37] to accelerate the k-means clustering algorithms. For example, Kaushik [38] proposed an effective sparsification method to speed up the algorithm by sparsify the original data matrix.

However, most of the existing works are devoted to accelerating the K-means algorithm and reducing the calculation cost by seeking better initialization conditions or changing the k-means distance calculation strategy. Different with these works, we focus on the optimization process of K-means clustering algorithm. In this paper, we develop a novel algorithm to solve the K-means clustering problem. Unlike Lloyd's algorithm, which is a heuristic method, our algorithm has definite theoretical guarantees. We reformulated the classical K-means objective function as a trace maximization problem and then replaced it with a new formulation. Compared with the original K-means clustering method, the new objective function does not need to calculate the cluster centers and requires fewer additional intermediate variables caused by the calculation of cluster centers. In addition, it can be easily optimized with an efficient iterative re-weighted method. For summarization, we present the main contributions of this work as follows:

- A novel algorithm is proposed for K-means clustering. We reformulated the objective function of K-means as an equivalent counterpart. The new objective function does not need to calculate the cluster centers and requires fewer additional intermediate variables.
- We develop an effective algorithm to solve the involved optimization problem. It has the same computational complexity as Lloyd's algorithm, $\mathcal{O}(ndk)$. More importantly, the convergence of our algorithm has a verifiable theoretical guarantee and shows a faster convergence rate in experiments.
- Our algorithm retains the advantage as Lloyd's algorithm which can be accelerated by parallel computing. In addition, our algorithm can also be combined with some existing methods for improving performance or speeding up the algorithm, such as initialization methods and dimensionality reduction based methods.
- Extensive experimental results on a series of real world data sets demonstrate the effectiveness and efficiency of the proposed method. Compared with Lloyd's algorithm, our algorithm performs better and more stable.

The reminder of this work is organized as follows. In Section 2, we will briefly describe K-means clustering, and then review the related work over it. We provide our approach and corresponding optimization method in Section 3, while in Section 4 we will verify its performance. Finally, a short conclusion and discussion is given in Section 5.

## 2 Related Work

In this section, we briefly review the prior K-means works. We denote the data matrix as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n] \in \mathcal{R}^{d\times n}$, where $d$ is the dimension and $n$ is the number of the samples. $\boldsymbol{x}_i$ denotes the $i$-th column of matrix $X$, and $x_{ij}$ is the $(i, j)$-element. $Tr(\boldsymbol{X})$ is the trace of $X$, $\|\boldsymbol{X}\|_F^2 = Tr(\boldsymbol{X}\boldsymbol{X}^T)$ denotes the squared Frobenius norm of $\boldsymbol{X}$. We define $\boldsymbol{F} \in \mathcal{R}^{n\times k}$ as indicator matrix, $\boldsymbol{F}$ is a binary matrix with $\boldsymbol{f}_{ij} = 1$ if $\boldsymbol{x}_i$ has label $y_i = j$, and the rest elements in the same row of $\boldsymbol{F}$ are 0. Matrices and vectors are represented with bold letters in uppercase and lowercase, respectively

### 2.1 K-Means Clustering

Given a data matrix $\boldsymbol{X} \in \mathcal{R}^{d\times n}$ that consists of $n$ samples $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$. K-means clustering problem aims at finding a partition $\Pi = \{\pi_1, \pi_2, ..., \pi_k\}$ that makes the sum of the square distances between each point and its closest center is minimized. We can formulate the problem as

$$\min_{\boldsymbol{C}} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in \pi_j} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|_2^2, \tag{1}$$

where $\boldsymbol{c}_j$ denotes the centroid of cluster $\pi_j$. Recalling the definition of squared Frobenius norm and indicator matrix $\boldsymbol{F}$. Then, K-means clustering problem can be rewritten as

$$\min_{\boldsymbol{F}\in Ind, \boldsymbol{C}} \|\boldsymbol{X} - \boldsymbol{C}\boldsymbol{F}^T\|_F^2, \tag{2}$$

where $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k] \in \mathcal{R}^{d\times k}$ is the centers of the $k$ clusters, $\boldsymbol{F}$ is the indicator matrix and $\boldsymbol{F} \in \mathcal{R}^{n\times k}$.

### 2.2 Lloyd's K-Means Algorithm

Solving problem (2) exactly is NP-hard. Lloyd [13] proposed a local search algorithm for K-means clustering. It is one of the most popular algorithms and implemented as a standard clustering method in most machine learning researches [17]. Lloyd's algorithm begins with $k$ randomly centers, the

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

centers can be obtained by randomly select $k$ samples from the data sets or randomly assign all the simples to $k$ clusters. Then assign all the samples to the nearest centers and moving the centers to the centroid of the new clusters. Repeat these two steps(assignment and center recomputed) until convergence. Essentially, Lloyd's algorithm can be regard as using alternate optimization method to solve problem (2). The entire procedure has been outlined in Algorithm 1. The objective function value of Eq. (2) decreases monotonically until convergence with the iteration of Algorithm 1.

**Algorithm 1.** Lloyd's K-Means Algorithm

**Require:** data matrix $\boldsymbol{X} \in \mathcal{R}^{d\times n}$, cluster number $k$.
**Initialization:** $k$ centers selected randomly.
**while** no converge **do**
Step 1. Assign each point to the nearest center and update the indicator matrix $\boldsymbol{F}$ in Eq. (2);
Step 2. Update matrix $\boldsymbol{C}$ in Eq. (2) by moving the $k$ centers to the centroid of the new clusters.
**end while**

Lloyd's algorithm is a heuristic method and sensitive to initialization. There are a lot of existing researches have been proposed to accelerated K-means clustering algorithm, such as initialization methods and dimensionality reduction methods. Both the proposed algorithm and Lloyd's algorithm can be accelerated by these method. Therefore, we will use the two most common initialization method: random initialization and K-means++ as instance to demonstrate the characteristics of the proposed algorithm.

### 2.3 K-Means++ Initialization

K-means++ [23] algorithm is one of the most representation initialization method. K-means++ method initializes K-means by choosing random starting centers with specific probabilities. Define $D(x)$ as the shortest distance from a data point $x$ to the closet center. K-means++ initialization can be summarized as following step:

- Randomly choose an initial center $\boldsymbol{c}_i$ from the dataset $\boldsymbol{X}$
- Choose another point $\boldsymbol{x}_i \in \boldsymbol{X}$ as the next center $\boldsymbol{c}_i$ with probability $\frac{D(\boldsymbol{x}_i)^2}{\sum_{\boldsymbol{x}\in\boldsymbol{X}} D(\boldsymbol{x})^2}$.
- Repeat step 2 until we obtain $k$ centers.

K-means++ method tries to avoid choosing two centers that are too close to each other and generates initial centers near the final cluster positions.

### 2.4 Iterative Re-Weighted Method

In this subsection, we introduce an iterative re-weighted method [39], [40] to solve the involved optimization problem. The re-weighted method can be used to solve the following general optimization problem

$$\max_{x\in\mathcal{C}} \sum_i h_i(g_i(x)) + f(x), \tag{3}$$

where $h_i(\cdot)$ is an arbitrary convex function with respect to $g_i(x)$, and define $h_i'(\cdot)$ as the sub-gradient of the convex function $h_i(\cdot)$. The overall process of re-weighted method is described in Algorithm 2.

**Algorithm 2.** Iterative Re-Weighted Method

**while** no converge **do**
Step 1. Calculate the sub-gradient of the function: $D_i = h_i'(g_i(x)) = \frac{\partial h_i(g_i(x))}{\partial g_i(x)}$;
Step 2. Update $x$ by the optimal solution to the problem:$\max_{x\in\mathcal{C}} \sum_i Tr(D_i^T g_i(x)) + f(x)$.
**end while**

Problem (3) can be converged to a locally optimal solution with Algorithm 2. Iterative re-weighted method can be wide used to solve these non-convex optimization problem like [41], [42], [43].

## 3 Proposed Method

In this section, we proposed a novel formulation to solve the K-means clustering problem. We reformulate K-means clustering as a trace maximization problem, and rewrite it as an equivalent formulation. Then, we develop an efficient algorithm to solve the involved optimization problem. Finally, complexity analysis as well as convergence analysis are given.

### 3.1 Formulation

Since $\|\cdot\|_F^2$ is the squared Frobenius norm and $\|\boldsymbol{X}\|_F^2 = Tr(\boldsymbol{X}\boldsymbol{X}^T)$. Then, the K-means clustering problem with Eq. (2) can be reformulated as:

$$\min_{\boldsymbol{F}\in Ind,\boldsymbol{C}} Tr((\boldsymbol{X}-\boldsymbol{C}\boldsymbol{F}^T)(\boldsymbol{X}-\boldsymbol{C}\boldsymbol{F}^T)^T). \tag{4}$$

Taking the derivative $w.r.t.\boldsymbol{C}$ and setting it to zero, we have:

$$\boldsymbol{C} = \boldsymbol{X}\boldsymbol{F}(\boldsymbol{F}^T\boldsymbol{F})^{-1}. \tag{5}$$

Substituting it into Eq. (4) gives:

$$\begin{aligned} &\min_{\boldsymbol{F}\in Ind} Tr(\boldsymbol{X}\boldsymbol{X}^T) - 2Tr(\boldsymbol{X}\boldsymbol{F}\boldsymbol{C}^T) + Tr(\boldsymbol{C}\boldsymbol{F}^T\boldsymbol{F}\boldsymbol{C}^T) \\ \Leftrightarrow &\min_{\boldsymbol{F}\in Ind} Tr(\boldsymbol{X}\boldsymbol{X}^T) - Tr(\boldsymbol{X}\boldsymbol{F}(\boldsymbol{F}^T\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{X}^T). \end{aligned} \tag{6}$$

Since $Tr(\boldsymbol{X}\boldsymbol{X}^T)$ is a constant, Eq. (6) is equivalent to the following problem:

$$\max_{\boldsymbol{F}\in Ind} Tr(\boldsymbol{F}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{F}(\boldsymbol{F}^T\boldsymbol{F})^{-1}). \tag{7}$$

Then, the K-means clustering problem can be reformulated as a trace maximization problem. Here, $\boldsymbol{F}$ is an indicator matrix, and there is only one non-zero element in each row of $\boldsymbol{F}$. Therefore, $\boldsymbol{F}^T\boldsymbol{F}$ is a diagonal matrix with the $i$-th diagonal element equal to $\boldsymbol{f}_i^T\boldsymbol{f}_i$, here $\boldsymbol{f}_i$ is the $i$-th column of matrix $\boldsymbol{F}$. Problem (7) can be rewritten as:

$$\max_{\boldsymbol{F}\in Ind} \sum_{i=1}^{k} \frac{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\boldsymbol{f}_i^T\boldsymbol{f}_i}. \tag{8}$$

It is difficult to optimize problem (8) directly. In order to solve the optimization problem, we introduce a variable $\boldsymbol{s}$, $\boldsymbol{s} \in \mathcal{R}^{1\times k}$. Then, problem (8) can be reformulated as following equivalent counterpart.

$$\max_{\boldsymbol{F}\in Ind,\boldsymbol{s}} \sum_{i=1}^{k} \left(2s_i\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i} - s_i^2\boldsymbol{f}_i^T\boldsymbol{f}_i\right). \tag{9}$$

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

**Lemma 1.** *Problem (9) is equivalent to problem (8):*

**Proof.** Problem (9) can be solved by updating variables $\boldsymbol{s}$ and $\boldsymbol{F}$ alternately. It is easily to find that variable $\boldsymbol{s}$ has close form solution. For each $s_i$, taking the derivative w.r.t.$s_1$ and set it to zero, we have:

$$s_i = \frac{\sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}}{\boldsymbol{f}_i^T \boldsymbol{f}_i}. \tag{10}$$

Substituting Eq. (10) to Eq. (9), then we can easily find that problem (9) is equivalent to problem (8). □

From the new objective function Eq. (9), we can find that the new formulation proposed in this paper does not need to calculate the cluster centers, it calculates vector $\boldsymbol{s}$ instead of the cluster centers $\boldsymbol{C}$. In other words, in each iteration, it only needs to update the vector $\boldsymbol{s}$ with $k$ variables instead of calculating the cluster centers with $d \times k$ variables. Therefore, the proposed objective function requires fewer additional intermediate variables.

### 3.2 Optimization

The involved optimization problem can be solved by alternate optimization method.

*Fixing indicator matrix $\boldsymbol{F}$, update $\boldsymbol{s}$:* Problem (9) can be rewritten as

$$\max_{\boldsymbol{s}} \sum_{i=1}^{k} \left( 2 s_i \sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i} - s_i^2 \boldsymbol{f}_i^T \boldsymbol{f}_i \right). \tag{11}$$

From Eq. (11), we can find that the optimization of $\{s_1, s_2, ..., s_k\}$ is independent to each other and can be calculated separately. For $s_i$, taking the derivative *w.r.t.*$s_i$ and setting it to zero, we have

$$s_i = \frac{\sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}}{\boldsymbol{f}_i^T \boldsymbol{f}_i}. \tag{12}$$

*Fixing $\boldsymbol{s}$, update indicator matrix $\boldsymbol{F}$:* Problem (9) can be rewritten as

$$\max_{\boldsymbol{F} \in Ind} \sum_{i=1}^{k} \left( 2 s_i \sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i} - s_i^2 \boldsymbol{f}_i^T \boldsymbol{f}_i \right), \tag{13}$$

here $\boldsymbol{X}^T \boldsymbol{X}$ is a positive semidefinite matrix, and $\sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}$ is a convex function w.r.t.$\boldsymbol{f}_i$. Problem (13) can be effectively solved by using the iterative re-weighted method introduced in section 2.4.

First, we calculate the sub-gradient of the convex function according to step 1 in Algorithm 2. Here, we define $\boldsymbol{m}_i$ as the sub-gradient of the convex function $\sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}$. And the $i$-th column of matrix $\boldsymbol{M}$ is $\boldsymbol{m}_i$. Then, we have:

$$\boldsymbol{m}_i = \frac{\partial \sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}}{\partial \boldsymbol{f}_i} = \frac{\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{f}_i}}. \tag{14}$$

According to the step 2 in Algorithm 2, the indicator matrix $\boldsymbol{F}$ can be obtained by solving the following problem:

$$\max_{\boldsymbol{F} \in Ind} \sum_{i=1}^{k} (2 s_i \boldsymbol{f}_i^T \boldsymbol{m}_i - s_i^2 \boldsymbol{f}_i^T \boldsymbol{f}_i). \tag{15}$$

Since $\boldsymbol{F}$ is an indicator matrix, we can easily find that $\boldsymbol{f}_i^T \boldsymbol{f}_i = \boldsymbol{f}_i^T \boldsymbol{1}$, (here $\boldsymbol{1} \in \mathcal{R}^{n \times 1}$ is a vector with all the elements are 1). Then Eq. (15) can be reformulated as the following trace maximization problem:

$$\max_{\boldsymbol{F} \in Ind} Tr(\boldsymbol{F}^T (\boldsymbol{A} - \boldsymbol{B})), \tag{16}$$

here, we define matrix $\boldsymbol{A} = 2\boldsymbol{M}\boldsymbol{S}$ and matrix $\boldsymbol{B} = \boldsymbol{1}\widetilde{\boldsymbol{S}}$, $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{R}^{n \times k}$. Matrix $\boldsymbol{S}$ and $\widetilde{\boldsymbol{S}} \in \mathcal{R}^{k \times k}$ are diagonal matrices, and the $i$-th diagonal element of matrix $\boldsymbol{S}$ is $s_i$, while the $i$-th diagonal element of matrix $\widetilde{\boldsymbol{S}}$ is $s_i^2$. Besides, $\boldsymbol{1} \in \mathcal{R}^{n \times k}$ is a matrix which all the elements are 1. The indicator matrix $\boldsymbol{F}$ has only one nonzero element in each row, and the nonzero element is 1. Then problem (16) can be solved by find the index of the largest element in each row of $(\boldsymbol{A} - \boldsymbol{B})$. For example, if the $l$-th element is the largest element in the $i$-th row of $(\boldsymbol{A} - \boldsymbol{B})$, then $\boldsymbol{F}_{il} = 1$.

Empirical evidences show Algorithm 2 converges very fast and usually converges in 20 iterations [40]. In the subsequent experiments, we found that the process of updating F with re-weighted method usually converges in a few iterations(most of them convergence in $3 \sim 10$ iterations).

To sum up, the algorithm we proposed can be summarized in Algorithm 3. If the indicator matrix $\boldsymbol{F}$ becomes stable, in other words, if $\boldsymbol{F}_{new} = \boldsymbol{F}_{old}$ or $\text{label}_{new} = \text{label}_{old}$, the algorithm converges. It is worth noting that Algorithm 3, like Lloyd's algorithm, retains the advantage of adopting parallel computation. For example, similar as Lloyd's algorithm, indicator matrix $\boldsymbol{F}$ can be updated row by row independently.

**Algorithm 3.** Algorithm to Solve the Problem (9)

**Require:** Data matrix $\boldsymbol{X} \in \mathcal{R}^{d \times n}$, number of clusters $k$.
  **Initialization:** Initial $\boldsymbol{F} \in \mathcal{R}^{n \times k}$.
  **while** no converge **do**
    Step 1. Update $\boldsymbol{s}$ by Eq. (12);
    **while** no converge **do**
      Step 2. Update $\boldsymbol{M}$ by Eq. (14);
      Step 3. Update $\boldsymbol{F}$ by find the index of the maximum value in each row of $(\boldsymbol{A} - \boldsymbol{B})$, which defined in Eq. (16);
  **end while**
  **end while**

### 3.3 Empty Clusters Problem

Empty cluster is a common problem to K-means clustering algorithms [44]. Some literatures consider the empty cluster problem as an insignificant problem and can be solved by executing the algorithm for multiple times. In our algorithm, the empty cluster problem may also occur. Our algorithm updates the vector $\boldsymbol{s}$ and matrix $\boldsymbol{M}$ by column independently, that is to say, the corresponding $\boldsymbol{f}_i$ to the empty cluster $i$ will not affect the variable $s_j$ and $\boldsymbol{m}_j$, which related to another cluster. Therefore, as long as the meaningless values generated by the empty cluster are removed, the whole algorithm will not be affected.

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

Our algorithm, like other K-means algorithm, can also solve the empty cluster problem through executing the algorithm multiple times. However, we prefer to avoid the generation of empty clusters in the optimization process. Therefore, we give the following two strategies as the consideration of the empty cluster problem. When updating indicator matrix $\boldsymbol{F}$ in step 3 of algorithm 3, if empty cluster occurs, we can adopt the following processing methods. One of them is we can keep the value of the last iteration for the $\boldsymbol{f}_i$ corresponding to the empty cluster $i$(or choose to keep one non-zero value in $\boldsymbol{f}_i$ to avoid the empty cluster problem). Another option is that we can randomly divide the cluster with the largest number of samples into two parts after the occurrence of the empty cluster, and assign one part to the empty cluster $\boldsymbol{f}_i$. These strategies can also be find in the other K-means algorithms.

Therefore, according to the specific situation, we can both choose executing the algorithm multiple times or use the above two strategies to avoid the empty cluster problem.

### 3.4 Complexity Analysis

Observing the Algorithm 3, we can find that the main computation cost occurs in Step 1 to Step 3. In Step 1, for the size of $\boldsymbol{f}_i$ is $n \times 1$, $\boldsymbol{X}$ is $d \times n$, and $\boldsymbol{F}$ is $n \times k$, the computation complexity of this step is $\mathcal{O}(ndk)$. Similarly, we can find the computation complexity of step 2 and 3 is $\mathcal{O}(ndk)$ and $\mathcal{O}(nk)$, respectively. Thus the computation complexity of Algorithm 3 is $\mathcal{O}(ndk)$. In addition, the experimental results in section 4 show that the proposed algorithm has a faster convergence rate.

### 3.5 Convergence Analysis

In this subsection, we will provide the convergence analysis over Algorithm 3. Since step 1 is the optimal solution of problem (11), we can easily prove its convergence. Therefore, the key problem of the convergence analysis of algorithm 3 is step2 and step3. And then, we will give the convergence analysis of step2 and step3 in Algorithm 3.

**Lemma 2.** *The step 2 and 3 in Algorithm 3 will decrease the objective value of the problem (13) in each iteration until it converge:*

**Proof.** Suppose $\tilde{F}$ and $\tilde{\boldsymbol{f}}_i$ is the solution updated by step 3 in Algorithm 3, respectively. According to step 3, we know:

$$\sum_{i=1}^{k}(2s_i\tilde{\boldsymbol{f}}_i^T\boldsymbol{m}_i - s_i^2\tilde{\boldsymbol{f}}_i^T\tilde{\boldsymbol{f}}_i) \geq \sum_{i=1}^{k}(2s_i\boldsymbol{f}_i^T\boldsymbol{m}_i - s_i^2\boldsymbol{f}_i^T\boldsymbol{f}_i), \quad (17)$$

where $\boldsymbol{m}_i = \frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}}$, Eq (17) can be rewritten as

$$\sum_{i=1}^{k}\left(2s_i\tilde{\boldsymbol{f}}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}} - s_i^2\tilde{\boldsymbol{f}}_i^T\tilde{\boldsymbol{f}}_i\right) \geq \sum_{i=1}^{k}\left(2s_i\boldsymbol{f}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}} - s_i^2\boldsymbol{f}_i^T\boldsymbol{f}_i\right), \quad (18)$$

Since $\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}$ is a convex function w.r.t.$\boldsymbol{f}_i$, according to the properties of convex function and subgradient, we have

$$\sqrt{\tilde{\boldsymbol{f}}_i^T\boldsymbol{X}^T\boldsymbol{X}\tilde{\boldsymbol{f}}_i} - \sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i} \geq \tilde{\boldsymbol{f}}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}} - \boldsymbol{f}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}}. \quad (19)$$

Therefore, we have

$$\sum_{i=1}^{k}\left(2s_i\sqrt{\tilde{\boldsymbol{f}}_i^T\boldsymbol{X}^T\boldsymbol{X}\tilde{\boldsymbol{f}}_i} - 2s_i\tilde{\boldsymbol{f}}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}}\right) \geq \sum_{i=1}^{k}\left(2s_i\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i} - 2s_i\boldsymbol{f}_i^T\frac{\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}{\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i}}\right). \quad (20)$$

Summing Eqs. (18) and (20) on both side, we arrive at:

$$\sum_{i=1}^{k}\left(2s_i\sqrt{\tilde{\boldsymbol{f}}_i^T\boldsymbol{X}^T\boldsymbol{X}\tilde{\boldsymbol{f}}_i} - s_i^2\tilde{\boldsymbol{f}}_i^T\tilde{\boldsymbol{f}}_i\right) \geq \sum_{i=1}^{k}\left(2s_i\sqrt{\boldsymbol{f}_i^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{f}_i} - s_i^2\boldsymbol{f}_i^T\boldsymbol{f}_i\right). \quad (21)$$

It is worth noting that equality in Eq. (21) holds only when the algorithm converges. Thus the step 2 and 3 in Algorithm 3 will decrease the objective value of the problem (13) in each iteration until it converge. □

## 4 Experiments

In this section, we will compare our algorithm with Llyod's algorithm. The performance of the proposed algorithm is validated via numerous experiments on real world benchmark datasets. In order to demonstrate the performance of the proposed algorithm, the experiments can be consist of four parts: the optimal objective function value, clustering performance, convergence analysis and the running time. Considering K-means clustering is sensitive to initialization, random initialization and k-means++ are used in the experiments. Both the two algorithms have the same initialization conditions in each experiment, the details of the experiments will be described later. All experiments are implemented on a DELL Optiplex 7050 with Intel i7-7700 Processor, 32G RAM, and the experimental environment is Matlab 2017a. The code of our algorithm and the datasets used in this paper are provided in https://github.com/ZihengLi6321/K-Means-with-new-formulation.

### 4.1 Data Sets Description

In this paper, a series of real world benchmark datasets are used to evaluate the performance of the algorithms, includes data sets with different dimensions, number of samples, and classes. The property of them is summarized in Table 1, and the data sets are sorted by the dimension of the features.

### 4.2 Evaluation Metrics

The objective function of our proposed algorithm is an equivalent counterpart of the classical K-means problem.

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Description of Datasets

| Datasets | Features | # of Samples | Classes |
|---|---|---|---|
| Iris | 4 | 150 | 3 |
| Balance | 4 | 625 | 3 |
| Dermatology | 34 | 366 | 6 |
| uspst | 256 | 2007 | 10 |
| USPSdata_20 | 256 | 1854 | 10 |
| USPSdata | 256 | 9298 | 10 |
| MSRA25 | 256 | 1799 | 12 |
| PalmData25 | 256 | 2000 | 100 |
| Binalpha | 320 | 1404 | 36 |
| Ecoli | 343 | 336 | 8 |
| Corel_5k | 423 | 5000 | 50 |
| MnistData_05 | 784 | 3495 | 10 |
| MnistData_10 | 784 | 6996 | 10 |
| Coil20Data_25 | 1024 | 1440 | 20 |
| Mpeg7 | 6000 | 1400 | 70 |
| TDT2_10 | 36771 | 653 | 10 |

Both the algorithm we proposed and the Lloyd's algorithm are optimizing problem (2). Therefore, the optimal objective function value[1] is used to compare the performance of the two algorithms. *In order to ensure the fairness of the experiment, the objective function values of the two algorithms are calculated with Eq. (2)*[2]. In addition, in order to make the experimental results more reliable, we repeat the experiments for multiple times and use *Min_obj*, *Max_obj*, *Mean_obj* and *Std_obj* to present the performance of the two algorithms. A short description over them is as follow:

- *Min_obj* is the minimum value of the optimal objective function value obtained from multiple experiments.
- *Max_obj* is the maximum value of the optimal objective function value obtained from multiple experiments.
- *Mean_obj* is the mean value of the optimal objective function obtained from multiple experiments. Since K-means is sensitive to initialization conditions, the mean value of multiple experimental results can be used to reduce the effects of initialization on the experiment, making the results more credible.
- *Std_obj* is the standard deviation of the optimal objective function obtained from multiple experiments. It reflects the degree of dispersion between individuals in the group. Thus, Std_obj can be used to compare the stability of algorithms under different initialization conditions.

We also present the clustering performance of the proposed algorithm. We compared the performance of our algorithm and Lloyd's algorithm with four widely used evaluation metrics, including *ACC*(Clustering Accuracy), *NMI*(Normalized Mutual Information), *F-score* and *ARI* (Adjusted Rand Index). In addition, we present the convergence curves of the two algorithms, while the number of iterations and the running time are also be considered. Similarly, we use *Mean_iter*, *Std_iter*, *Mean_time* and *Std_time* to denote the mean value, standard deviation of the number of iterations and the running time, respectively.

### 4.3 K-Means Clustering with Different Initialization

In this section, we present the performance of the algorithms by comparing the obtained optimal objective function value. *All the optimal objective function values of the two algorithms are calculated with Eq. (2).* Since K-means is sensitive to initialization conditions, in the following experiment, both random initialization and K-means++ method are used to demonstrate the performance of the two k-means algorithms.

#### 4.3.1 K-Means Clustering with Random Initialization

Random initialization is the simplest and most common initialization method in k-means clustering algorithms. In this experiment, we randomly selected $k$ samples as the initial clustering center, and assign all the samples to the neatest cluster as the initial label. To make the results more reliable, we repeated all the experiments 50 times, the initial clustering center was randomly reselected for each time. In addition, both Lloyd's algorithm and the proposed algorithm have the same initialization condition in each experiment.

We compare the Min_obj, Max_obj, Mean_obj and Std_obj on 17 benchmark datasets in Table 2. From Table 2, we can observe that:

a) According to the experimental results, we can find that K-means clustering is sensitive to initialization, especially for some data sets such as iris and dermatology. It can be seen from the min_obj and Max_obj that the optimal objective function values obtained from multiple experiments are quite different. This is because the K-means algorithms converges finitely to a local minimum solution, and it easily gets trapped in poor local solution with a poor initialization conditions. However, for some data sets, such as isolet and MnistData_10, initialization conditions have little effect on clustering results, this depends on the structure of the data sets.
b) In terms of the experimental results of min_obj, max_obj and Mean_obj, the performance of our algorithm is comparable to that of the Lloyd's algorithm, or slightly better than that of the Lloyd's algorithm. Especially for max_obj and Mean_obj, the experimental results obtained by the proposed algorithm are better than Lloyd's algorithm.
c) From the view of std_obj, the proposed algorithm performs better than the Lloyd's algorithm. The experimental results show that the proposed algorithm performs more stable than the Lloyd's algorithm with different initialization.

#### 4.3.2 K-Means Clustering with K-Means++

K-means++ is also one of the most popular initialization methods of K-means clustering. K-means++ choose random starting centers with very specific probabilities as the initial centers, aims to avoid choosing two centers that close to each other. Similarly, all the experiments are repeated for 50 times, and for each experiment, the initialization for our

1. The optimal objective function value is the objective function value obtained after the convergence of the algorithm.

2. In the subsequent experiments, our algorithm use Eq. (6) to calculate the objective function value. Both theory and experiment can verify that Eqs. (6) and (2) are equivalent.

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
K-Means Clustering with Random Initialization

| Data sets | Llyod's Algorithm | | | | Our Algorithm | | | |
|---|---|---|---|---|---|---|---|---|
| | Min_obj | Max_obj | Mean_obj | Std_obj | Min_obj | Max_obj | Mean_obj | Std_obj |
| Iris | **7.8941E+01** | 1.4660E+02 | 8.9281E+01 | 2.3931E+01 | **7.8941E+01** | **1.4345E+02** | **8.7915E+01** | **2.2463E+01** |
| Balance | **3.4723E+03** | 3.5968E+03 | 3.4989E+03 | 3.1017E+01 | **3.4723E+03** | **3.5456E+03** | **3.4953E+03** | **2.6274E+01** |
| Dermatology | **5.5809E+03** | 8.1906E+03 | 5.9691E+03 | 5.1806E+02 | **5.5809E+03** | **6.9262E+03** | **5.9312E+03** | **4.0417E+02** |
| uspst | 6.4249E+04 | 6.7019E+04 | 6.4928E+04 | 6.5203E+02 | **6.4246E+04** | **6.6398E+04** | **6.4863E+04** | **5.9359E+02** |
| USPSdata_20 | 6.7330E+04 | 7.0144E+04 | 6.8070E+04 | 7.1188E+02 | **6.7328E+04** | **6.9317E+04** | **6.7864E+04** | **5.0629E+02** |
| USPSdata | **3.3884E+05** | 3.4478E+05 | 3.3971E+05 | 1.7613E+03 | **3.3884E+05** | **3.4254E+05** | **3.3940E+05** | **1.3155E+03** |
| MSRA25 | 1.5515E+08 | 1.7214E+08 | 1.6132E+08 | 4.2495E+06 | **1.5514E+08** | **1.7154E+08** | **1.6125E+08** | **4.0781E+06** |
| PalmData25 | 5.3679E+08 | 5.7811E+08 | 5.6061E+08 | 1.0850E+07 | **5.3613E+08** | **5.7808E+08** | **5.6018E+08** | **1.0698E+07** |
| Binalpha | 6.8557E+04 | 6.9813E+04 | 6.9127E+04 | 3.6068E+02 | **6.8390E+04** | **6.9780E+04** | **6.9060E+04** | **3.2198E+02** |
| Ecoli | **3.4395E+02** | 3.4913E+02 | 3.4589E+02 | 9.9741E-01 | **3.4395E+02** | **3.4906E+02** | **3.4588E+02** | **9.9456E-01** |
| Corel_5k | **4.4700E+06** | 4.5999E+06 | 4.5459E+06 | 3.6060E+04 | 4.4834E+06 | **4.5949E+06** | **4.5420E+06** | **3.4675E+04** |
| MnistData_05 | **8.8648E+09** | 8.9884E+09 | 8.9012E+09 | 3.4075E+07 | **8.8648E+09** | **8.9529E+09** | **8.8944E+09** | **2.7555E+07** |
| MnistData_10 | 1.7826E+10 | 1.7994E+10 | 1.7890E+10 | 4.2997E+07 | **1.7825E+10** | **1.7944E+10** | **1.7868E+10** | **3.1689E+07** |
| Coil20Data_25 | **2.4549E+09** | 2.7534E+09 | 2.5789E+09 | 6.6713E+07 | 2.4910E+09 | **2.7341E+09** | **2.5764E+09** | **6.0914E+07** |
| Mpeg7 | **5.9253E+03** | 6.1652E+03 | 6.0115E+03 | 6.8039E+01 | 5.9266E+03 | **6.1475E+03** | **6.0053E+03** | **6.3751E+01** |
| TDT2_10 | 2.0124E+05 | **2.2267E+05** | 2.0864E+05 | 6.0453E+03 | **2.0085E+05** | **2.2267E+05** | **2.0793E+05** | **5.9844E+03** |

*The experiments were repeated for 50 times, the initial clustering center was randomly reselected for each time. The best results are highlighted in bold.*

algorithm and the Lloyd's algorithm is the same. The comparison of the algorithms across the various data sets is shown in Table 3. From Table 3, we can find that:

a) Compared with the experiments with random initialization, the value of min_obj almost unchanged. This is because multiple experiments can effectively alleviate the problem of poor initial conditions caused by random initialization. Thus, random initialization still has the possibility of giving a good initialization, but it is not stable. In contrast, for max_obj and mean_obj, the experiments with K-means++ initialization perform better than that that with random initialization, which is more obviously in some datasets like dermatology and iris. What's more, the value of std_obj is significantly for smaller on most of the data sets. This shows that K-means++ can give better initialization conditions, so that k-means algorithm has better and more stable performance.
b) Under different initialization conditions, our algorithm still performs better than Lloyd's algorithm. In particular, the upper bound and the mean value of the objective function value obtained in the multiple experiments are smaller than Lloyd's algorithm.
c) From the view of std_obj, our algorithm is still more stable under different initialization conditions.

From the above K-means clustering experiment under different initialization methods, it can be concluded that our algorithm performs better than Lloyd's algorithm in a variety of real world data sets. Especially in the upper bound and mean value of the optimal objective function given by our algorithm. At the same time, our algorithm has more stable performance than Lloyd's algorithm.

TABLE 3
K-Means Clustering with K-Means++ Initialization

| Data sets | Llyod's Algorithm | | | | Our Algorithm | | | |
|---|---|---|---|---|---|---|---|---|
| | Min_obj | Max_obj | Mean_obj | Std_obj | Min_obj | Max_obj | Mean_obj | Std_obj |
| Iris | **7.8941E+01** | **1.4345E+02** | 8.5382E+01 | 1.9514E+01 | **7.8941E+01** | **1.4345E+02** | **8.5370E+01** | **1.9478E+01** |
| Balance | **3.4723E+03** | 3.5480E+03 | 3.4886E+03 | 2.0856E+01 | **3.4723E+03** | **3.5472E+03** | **3.4824E+03** | **1.8245E+01** |
| Dermatology | 5.5849E+03 | 7.4933E+03 | 6.0176E+03 | 4.8562E+02 | **5.5847E+03** | **7.0782E+03** | **5.9998E+03** | **4.4545E+02** |
| uspst | 6.4248E+04 | **6.5989E+04** | 6.4842E+04 | 6.0216E+02 | **6.4245E+04** | 6.6430E+04 | **6.4746E+04** | **5.8843E+02** |
| USPSdata_20 | **6.7329E+04** | 6.9668E+04 | **6.7861E+04** | 4.9611E+02 | 6.7334E+04 | **6.9397E+04** | 6.7914E+04 | **4.4843E+02** |
| USPSdata | **3.3884E+05** | 3.4510E+05 | 3.3955E+05 | 1.6975E+03 | **3.3884E+05** | **3.4251E+05** | **3.3903E+05** | **8.2107E+02** |
| MSRA25 | 1.5493E+08 | 1.6992E+08 | 1.5945E+08 | **3.2622E+06** | **1.5409E+08** | **1.6817E+08** | **1.5918E+08** | 3.3812E+06 |
| PalmData25 | 5.3091E+08 | **5.6830E+08** | 5.5112E+08 | **8.8730E+06** | **5.2844E+08** | **5.6830E+08** | **5.5061E+08** | 9.1038E+06 |
| Binalpha | 6.8523E+04 | 6.9758E+04 | 6.9096E+04 | 2.7852E+02 | **6.8319E+04** | **6.9505E+04** | **6.9042E+04** | **2.6517E+02** |
| Ecoli | 3.4302E+02 | **3.4804E+02** | 3.4583E+02 | 9.3771E-01 | **3.4298E+02** | **3.4804E+02** | **3.4582E+02** | **9.3441E-01** |
| Corel_5k | **4.4820E+06** | **4.6506E+06** | 4.5276E+06 | 4.2590E+04 | 4.4827E+06 | 4.6558E+06 | **4.5252E+06** | **3.8850E+04** |
| MnistData_05 | 8.8647E+09 | 8.9871E+09 | 8.9023E+09 | 3.3475E+07 | **8.8644E+09** | **8.9772E+09** | **8.8966E+09** | **3.0252E+07** |
| MnistData_10 | **1.7825E+10** | 1.7996E+10 | 1.7887E+10 | 4.8705E+07 | **1.7825E+10** | **1.7955E+10** | **1.7859E+10** | **4.0644E+07** |
| Coil20Data_25 | **2.4416E+09** | 2.6499E+09 | 2.5498E+09 | 5.7354E+07 | 2.4458E+09 | **2.6423E+09** | **2.5496E+09** | **5.5846E+07** |
| Mpeg7 | 5.8751E+03 | 6.1314E+03 | 5.9962E+03 | **5.9893E+01** | **5.8728E+03** | **6.1246E+03** | **5.9876E+03** | 6.1401E+01 |
| TDT2_10 | **1.9938E+05** | 2.2932E+05 | 2.1123E+05 | 6.3459E+03 | 1.9956E+05 | **2.2172E+05** | **2.1105E+05** | **5.6797E+03** |

*All the experiments were repeated for 50 times. The best results are highlighted in bold.*

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
Clustering Results with Random Initialization

| Data sets | Llyods Algorithm | | | | Our Algorithm | | | |
|---|---|---|---|---|---|---|---|---|
| | ACC | NMI | F-score | ARI | ACC | NMI | F-score | ARI |
| Iris | 0.8407±0.1182 | 0.7103±0.0771 | 0.7896±0.0610 | 0.6769±0.1085 | **0.8570±0.0994** | **0.7213±0.0650** | **0.7982±0.0514** | **0.6921±0.0914** |
| Balance | 0.5147±0.0315 | 0.1078±0.0465 | 0.4616±0.0283 | 0.1346±0.0450 | **0.5246±0.0115** | **0.1078±0.0131** | **0.4635±0.0091** | **0.1369±0.0140** |
| Dermatology | 0.7592±0.1217 | 0.8114±0.0639 | 0.7838±0.0970 | 0.7259±0.1260 | **0.7675±0.1209** | **0.8144±0.0614** | **0.7860±0.0950** | **0.7288±0.1234** |
| uspst | 0.6431±0.0360 | 0.6074±0.0146 | 0.5656±0.0241 | 0.5136±0.0272 | **0.6510±0.0310** | **0.6104±0.0111** | **0.5703±0.0185** | **0.5188±0.0209** |
| USPSdata 20 | 0.6250±0.0312 | 0.6181±0.0155 | **0.5731±0.0210** | 0.5215±0.0237 | **0.6324±0.0334** | **0.6196±0.0164** | **0.5731±0.0224** | **0.5217±0.0254** |
| USPSdata | 0.6513±0.0292 | 0.6075±0.0117 | 0.5766±0.0184 | 0.5255±0.0208 | **0.6590±0.0249** | **0.6101±0.0086** | **0.5812±0.0126** | **0.5308±0.0146** |
| MSRA25 | 0.4903±0.0459 | 0.5739±0.0456 | 0.4047±0.0477 | 0.3411±0.0568 | **0.4943±0.0467** | **0.5752±0.0449** | **0.4062±0.0479** | **0.3429±0.0570** |
| PalmData25 | 0.6884±0.0211 | 0.8906±0.0077 | 0.6480±0.0237 | 0.6442±0.0240 | **0.6888±0.0214** | **0.8908±0.0076** | **0.6484±0.0234** | **0.6446±0.0237** |
| Binalpha | 0.4001±0.0199 | 0.5654±0.0131 | 0.2833±0.0148 | 0.2621±0.0153 | **0.4043±0.0179** | **0.5683±0.0114** | **0.2874±0.0139** | **0.2663±0.0143** |
| Ecoli | 0.5472±0.0724 | 0.4837±0.0421 | 0.5147±0.0695 | 0.3922±0.0761 | **0.5497±0.0739** | **0.4843±0.0403** | **0.5166±0.0690** | **0.3944±0.0754** |
| Corel_5k | 0.1642±0.0051 | 0.2639±0.0035 | 0.0814±0.0024 | 0.0603±0.0024 | **0.1656±0.0044** | **0.2647±0.0026** | **0.0821±0.0020** | **0.0609±0.0020** |
| MnistData_05 | 0.5346±0.0363 | 0.4904±0.0183 | 0.4295±0.0249 | 0.3636±0.0275 | **0.5399±0.0290** | **0.4939±0.0160** | **0.4335±0.0209** | **0.3680±0.0300** |
| MnistData_10 | 0.5346±0.0349 | 0.4897±0.0172 | 0.4307±0.0209 | 0.3648±0.0229 | **0.5412±0.0189** | **0.4915±0.0115** | **0.4341±0.0142** | **0.3685±0.0151** |
| Coil20Data 25 | 0.5768±0.0616 | 0.7360±0.0269 | 0.5566±0.0489 | 0.5299±0.0531 | **0.5800±0.0619** | **0.7364±0.0283** | **0.5570±0.0486** | **0.5304±0.0527** |
| Mpeg7 | 0.4627±0.0176 | 0.6615±0.0113 | 0.2931±0.0219 | 0.2798±0.0228 | **0.4641±0.0162** | **0.6624±0.0109** | **0.2953±0.0218** | **0.2821±0.0227** |
| TDT2_10 | 0.4003±0.0766 | 0.3431±0.0878 | 0.2625±0.0413 | 0.1174±0.0554 | **0.4064±0.0753** | **0.3526±0.0894** | **0.2661±0.0428** | **0.1218±0.0570** |

*All the experiments were repeated for 50 times. The best results are highlighted in bold.*

### 4.4 Clustering Performance

The clustering results of the proposed algorithm and Lloyd's algorithm are presented in Table 4. In this part, we use randomly initialization to obtain the $k$ initial clustering centers. Similar with the previous experiments, we repeated all the experiments 50 times, the initial clustering centers were randomly reselected for each time. In each experiment, the initialization is the same for the two algorithms. The performance of the algorithms is measured by ACC, NMI, F-score and ARI in Table 4. From Table 4, we can find that the proposed algorithm performs better than Lloyd's algorithm. However, both algorithms are optimizing problem 1, the performance of the two algorithms are comparable on some of the datasets. Therefore the difference between the two algorithms on these datasets is not significant.

### 4.5 Convergence Analysis

In this subsection, we will analysis the convergence of the two algorithms. In order to demonstrate the convergence of the two algorithms based on different datasets, we selected 4 data sets Binalpha, uspst, Mpeg7 and TDT2_10 from 16 data sets and tested their convergence. In the four data sets, the dimensions of Binalpha and uspst are low, while the dimensions of Mpeg7 and TDT2_10 are relatively high. In contrast, the number of clusters of uspst and TDT2_10 are relatively small, while the cluster number of Binalpha and Mpeg7 are relatively large. At the same time, we use random and K-means++ initialization methods to initialize the K-means clustering experiments, to present the convergence of the two algorithms. The experimental results are shown in Fig. 1, where (a)-(d) are the cases of random initialization, and (e)-(h) are the cases of initialization with K-means++.

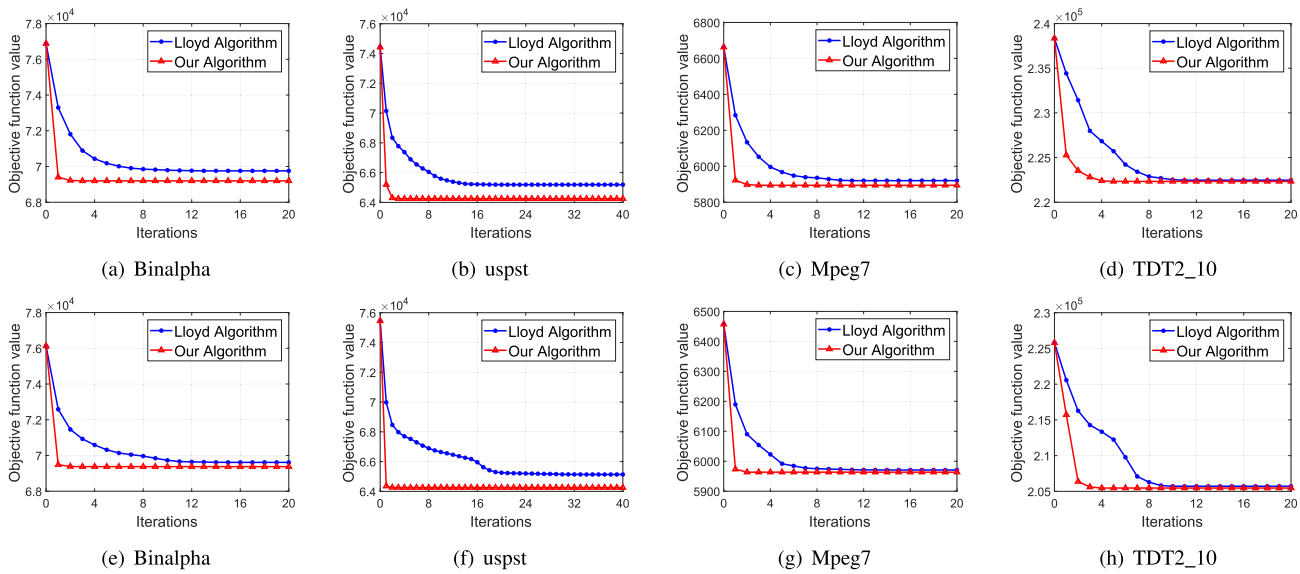
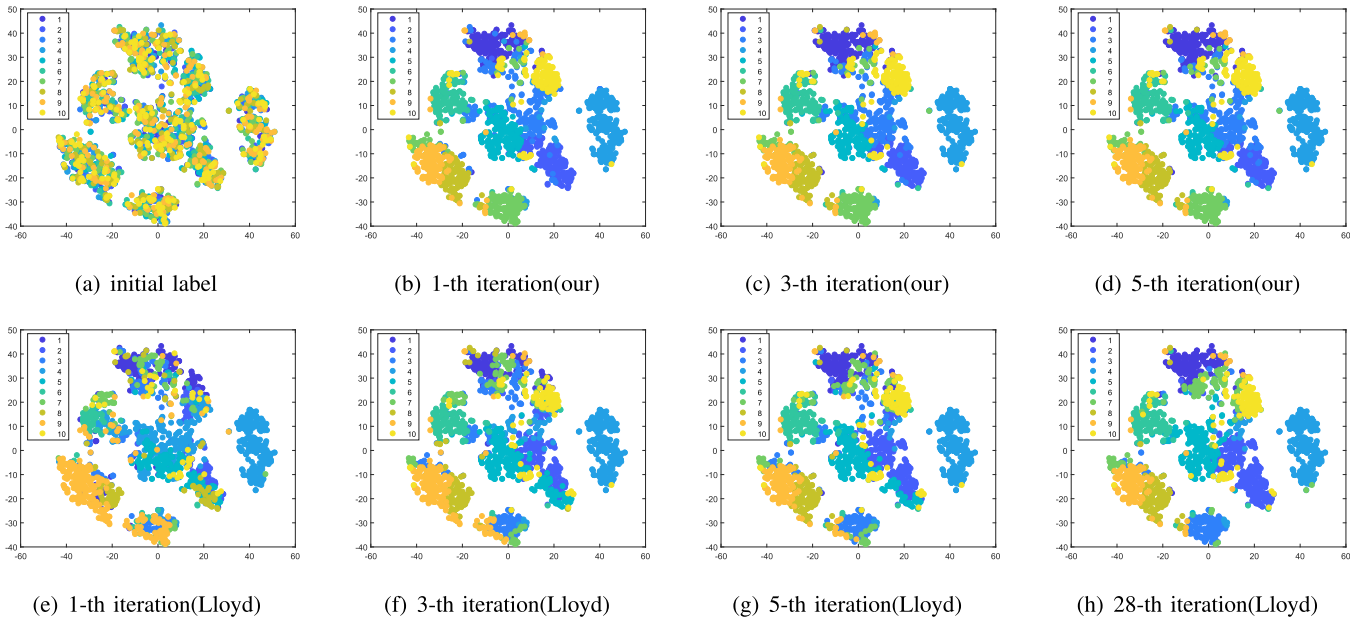



Fig. 1. Convergence curves of the two algorithms on different datasets, where (a)-(d) are the cases of random initialization, while (e)-(h) are the cases of initialization with K-means++.

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.



(a) initial label (b) 1-th iteration(our) (c) 3-th iteration(our) (d) 5-th iteration(our)

(e) 1-th iteration(Lloyd) (f) 3-th iteration(Lloyd) (g) 5-th iteration(Lloyd) (h) 28-th iteration(Lloyd)

Fig. 2. T-SNE results on uspst dataset. (a) is the result with initial label obtained by random assignment. (b)-(d) is the results obtained by our algorithms in different iterations, and our algorithm converges after five iterations. (e)-(f) is the results obtained by Lloyd's algorithms in different iterations, and it converges after twenty-eight iterations.

From Fig. 1, we can see that the objective function values of both algorithms decrease monotonously and converge well to a stable local solution. Besides, from Fig. 1, we can see that K-means++ is more likely to provide better initialization conditions than random initialization(Provides a smaller initial objective function value), but there are exceptions, such as Figs. 1b and 1c. However, regardless of the choice of initialization method, our algorithm shows a faster convergence rate than Lloyd's algorithm. In addtion, our algorithm can converge to a comparable or even better local solution for different data sets.

In order to demonstrate the performance of our algorithm visually, we visualize the clustering results of uspst in different iteration. The clustering results with different iterations are presented in Fig. 2. We randomly assigned all samples into $k$ clusters, and Fig. 2a shows the situation after the assignment. Using both algorithms for the clustering task, our algorithm converges after the fifth iteration, while Lloyd's algorithm converges after the 28th iteration. Figs. 2b, 2c, and 2d present the clustering results obtained by our algorithm in 1-th, 3-th and 5-th iteration respectively, while Figs. 2e, 2f, 2g, and 2h are the clustering results obtained by Lloyd's algorithm in 1-th, 3-th, 5-th and 28-th iteration. The experimental results show that our algorithm has a faster convergence rate.

### 4.6 Iterative Times and Running Time

In this subsection, we will present the iterative times and running time of the two algorithms. The performance of the two algorithms are measured by metrics including the number of iteration to convergence and the running time. Both two algorithms can be accelerated by parallel computing. In order to ensure the fairness of the experiments, neither two algorithms use parallel computing in the experiment of comparing the running time, and both algorithms iterated at the same level. Specifically, for Lloyd's algorithm, we loop through $k$ classes to implement the two steps of samples assignment and center calculation. Besides, we use matrix calculation to calculate the distance between the samples and k clustering centers respectively, assign the samples to the nearest clustering centers, and then updated k clustering centers respectively.

To make the experiment results more reliable, we repeated all the experiments 50 times. The average number of iterations, average time cost and standard deviation were used to measure the performance of the algorithm. The relevant experimental results are shown in Tables 5 and 6, where Table 5 is the number of iterations of the two algorithms for different data sets, while Table 6 is the running time.

TABLE 5
The Number of Iterations of the Two Algorithms

| Data sets | Lloyd's Algorithm | | Our Algorithm | |
|---|---|---|---|---|
| | Mean_iter | Std_iter | Mean_iter | Std_iter |
| Iris | 4.78 | 1.4748 | **3.30** | **0.8631** |
| Balance | 9.92 | 2.3460 | **4.74** | **0.9649** |
| Dermatology | 7.90 | 3.1053 | **3.66** | **1.0224** |
| uspst | 29.38 | 15.0141 | **4.96** | **1.4841** |
| USPSdata_20 | 27.94 | 10.4636 | **4.92** | **1.2428** |
| USPSdata | 59.62 | 19.7700 | **5.96** | **1.3547** |
| MSRA25 | 20.36 | 7.3866 | **5.74** | **1.9981** |
| PalmData25 | 12.68 | 2.5828 | **2.38** | **0.4903** |
| Binalpha | 17.56 | 3.9443 | **5.00** | **1.0498** |
| Ecoli | 4.18 | 1.5345 | **2.76** | **0.8935** |
| Corel_5k | 9.90 | 6.6555 | **6.52** | **4.2197** |
| MnistData_05 | 36.50 | 13.7280 | **9.48** | **2.8374** |
| MnistData_10 | 43.94 | 11.1692 | **11.28** | **3.3869** |
| Coil20Data_25 | 20.92 | 7.8943 | **5.36** | **1.5086** |
| Mpeg7 | 14.10 | 4.1400 | **4.14** | **0.8300** |
| TDT2_10 | 11.68 | 3.4608 | **6.46** | **1.5147** |

*The experiments was repeated for 50 times, in which mean_iter was the average number of iterations and std_iter was the standard deviation.*

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 6
The Running Time of the Two Algorithms

| Data sets | Lloyd's Algorithm | | Our Algorithm | |
|---|---|---|---|---|
| | Mean_time | Std_time | Mean_time | Std_time |
| Iris | **0.0029s** | **0.0020** | 0.0045s | 0.0021 |
| Balance | 0.0065s | 0.0017 | **0.0493s** | **0.0099** |
| Dermatology | **0.0028s** | **0.0011** | 0.0040s | 0.0017 |
| uspst | 0.6633s | 0.3381 | **0.2786s** | **0.1026** |
| USPSdata_20 | 0.5640s | 0.2064 | **0.2263s** | **0.0720** |
| USPSdata | 30.6772s | 10.2156 | **21.9030s** | **4.8933** |
| MSRA25 | 0.4868s | 0.1752 | **0.2281s** | **0.0741** |
| PalmData25 | 2.5850s | 0.5265 | **0.6007s** | **0.1103** |
| Binalpha | 1.1489s | 0.2429 | **0.2469s** | **0.0495** |
| Ecoli | 0.0061s | 0.0020 | **0.0035s** | **0.0021** |
| Corel_5k | 8.1967s | 5.5001 | **1.4539s** | **0.7816** |
| MnistData_05 | 4.7159s | 1.7549 | **1.5411s** | **0.4042** |
| MnistData_10 | 50.6742s | 12.8733 | **20.3061s** | **5.5228** |
| Coil20Data_25 | 2.6110s | 0.9797 | **0.1975s** | **0.0589** |
| Mpeg7 | 5.8161s | 1.6460 | **0.3501s** | **0.0533** |
| TDT2_10 | 8.8393s | 2.5683 | **0.0731s** | **0.0150** |

*The experiments was repeated for 50 times, in which mean_time was the average time and std_time was the standard deviation.*

From Table 5, we can observe that our algorithm has a fast convergence rate. Our algorithm always converges after several iterations. More importantly, it can be seen from the std_iter that the convergence rate of our algorithm is very stable, the standard deviation of the number of iterations of multiple experiments is small. Besides, from Table 6 we can easily find that, for most data sets, the running time of the proposed algorithm is shorter than that of Lloyd algorithm, while the standard deviation of the running time is still smaller than Lloyd algorithm.

From the above two experiments, it can be seen that our algorithm has a faster convergence rate and requires the same or even shorter running time compared with Lloyd algorithm. In addition, our algorithm is more stable than Lloyd's algorithm with a smaller standard deviation of multiple experiments on iterative times and running times.

## 5 Conclusion

In this paper, we proposed a novel algorithm for K-means clustering. We reformulated the original objective function of K-means clustering as a trace maximization problem. Then, we replaced the trace maximization problem with an equivalent counterpart. Besides, we utilized the alternate optimization method and iterative re-weighted method to solve the involved optimization problem efficiently. Our algorithm has a definite theoretical guarantee for convergence. Compared with Lloyd's algorithm, our algorithm does not need to calculate the cluster centers in each iteration, and requires fewer additional variables caused by centers calculation. A series of experiments on real world benchmark data sets have demonstrated its effectiveness and efficiency. It performs effectively and stably in K-means clustering experiments. In addition, the proposed algorithm shows a faster convergence rate in the experiments. However, since the proposed objective function is substantially equivalent to the original objective function of K-means clustering. The performance of clustering with the two algorithms on the benchmark datasets is comparable, and the difference of the experimental results on some of the datasets is not significant. In the future, we prefer to do more work on how to get a better local solution. The relevant details will be discussed in our future work.

## References


[1] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[2] J. Wen, Z. Zhang, Z. Zhang, L. Fei, and M. Wang, "Generalized incomplete multiview clustering with flexible locality structure diffusion," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 101–114, Jan. 2021.

[3] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, "A survey on large-scale machine learning," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 11, 2020, doi: 10.1109/TKDE.2020.3015777.

[4] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1101–1114, May 2017.

[5] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[6] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*, Berlin, Germany: Springer, 2012, pp. 561–580.

[7] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++ a clustering algorithm for data streams," *J. Exp. Algorithmics (JEA)*, vol. 17, pp. 2.1–2.30, 2012.

[8] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Informat. Syst.*, vol. 14, no. 1, pp. 1–37, 2007.

[9] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, Jul. 2016.

[10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering large graphs via the singular value decomposition," *Mach. Learn.*, vol. 56, no. 1, pp. 9–33, 2004.

[11] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *Proc. Int. Workshop Algorithms Comput.*, 2009, pp. 274–285.

[12] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Mach. Learn.*, vol. 75, no. 2, pp. 245–248, 2009.

[13] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[14] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognit. Lett.*, vol. 20, no. 10, pp. 1027–1040, 1999.

[15] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Syst. Appl.*, vol. 40, no. 1, pp. 200–210, 2013.

[16] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognit. Lett.*, vol. 20, no. 10, pp. 1027–1040, 1999.

[17] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause, "Approximate k-means++ in sublinear time," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1459–1467.

[18] A. Agarwal and A. Saxena, "An accountability of various clustering techniques for improvement of page searching process and analysis," in *Proc. Int. Conf. Sustain. Comput. Sci.*, 2019, pp. 645–650.

[19] Y. Xu, W. Qu, Z. Li, G. Min, K. Li, and Z. Liu, "Efficient *k*-means++ approximation with MapReduce," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3135–3144, Dec. 2014.

[20] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on MapReduce," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2009, pp. 674–679.

[21] G. Tzortzis and A. Likas, "The minmax k-means clustering algorithm," *Pattern Recognit.*, vol. 47, no. 7, pp. 2505–2516, 2014.

[22] A. Aggarwal, A. Deshpande, and R. Kannan, "Adaptive sampling for k-means clustering," in *Approximation, Randomization, Combinatorial Optimization. Algorithms Techniques*, Berlin, Germany: Springer, 2009, pp. 15–28.

[23] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2006, pp. 1027–1035.

[24] W. Zhu, F. Nie, and X. Li, "Fast spectral clustering with efficient large graph construction," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 2492–2496.


Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.

[25] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in *Proc. Int. Conf. on Mach. Learn.*, 2015, pp. 579–587.
[26] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 147–153.
[27] J. Drake and G. Hamerly, "Accelerated k-means with adaptive distance bounds," *Proc. 5th NIPS Workshop Optim. Mach. Learn.*, vol. 8, 2012.
[28] J. Newling and F. Fleuret, "Fast k-means with accurate bounds," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 936–944.
[29] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality reduction for k-means clustering and low rank approximation," in *Proc. 47th Annu. ACM Symp. Theory Comput.*, 2015, pp. 163–172.
[30] G. Hamerly, "Making k-means even faster," in *Proc. SIAM Int. Conf. Data Mining*, 2010, pp. 130–140.
[31] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed PCA and k-means clustering," *Proc. Big Learn. Workshop NIPS*, vol. 2013, 2013.
[32] C. Boutsidis, P. Drineas, and M. W. Mahoney, "Unsupervised feature selection for the $k$-means clustering problem," in *Proc. Adv. Neural Informat. Process. Syst.*, 2009, pp. 153–161.
[33] M. Capó, A. Pérez, and J. A. Lozano, "A cheap feature selection approach for the k-means algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2195–2208, May 2021.
[34] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 333–342.
[35] X. Shen, W. Liu, I. W. Tsang, F. Shen, and Q.-S. Sun, "Compressed k -means for large-scale clustering," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2527–2533.
[36] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," in *Proc. 5th ACM SIGKDD Int. Conf. on Knowl. Discov. Data Mining*, 1999, pp. 277–281.
[37] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.
[38] K. Sinha, "K-means clustering using random matrix sparsification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4684–4692.
[39] F. Nie, X. Wang, and H. Huang, "Multiclass capped P-norm SVM for robust classifications," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2415–2421.
[40] F. Nie, J. Yuan, and H. Huang, "Optimal mean robust principal component analysis," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1062–1070.
[41] R. He, W.-S. Zheng, and B.-G. Hu, "Maximum correntropy criterion for robust face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1561–1576, Aug. 2011.
[42] R. He, W.-S. Zheng, T. Tan, and Z. Sun, "Half-quadratic-based iterative minimization for robust sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 261–275, Feb. 2014.
[43] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *J. Mach. Learn. Res.*, vol. 14, no. 4, pp. 899–925, 2013.
[44] M. K. Pakhira, "A modified k-means algorithm to avoid empty clusters," *Int. J. Recent Trends Eng.*, vol. 1, no. 1, 2009, Art. no. 220.

**Feiping Nie** received the PhD degree in computer science from Tsinghua University, China in 2009, and currently is full professor with Northwestern Polytechnical University, China. His research interests are machine learning and its applications, such as pattern recognition, data mining, computer vision, image processing and information retrieval. He has published more than 100 papers in the following journals and conferences: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IJCV, TIP, TNNLS, *IEEE Transactions on Knowledge and Data Engineering*, ICML, NIPS, KDD, IJCAI, AAAI, ICCV, CVPR, and ACM MM. His papers have been cited more than 20000 times and the H-index is 84. He is now serving as associate editor or PC member for several prestigious journals and conferences in the related fields.

**Ziheng Li** is currently working toward the PhD degree with Northwestern Polytechnical University, Xian, China, under the supervision of Prof. F. Nie. His research interests are machine learning and its applications, including clustering, matrix completion, and information retrieval.

**Rong Wang** received the PhD degree in computer science from Tsinghua University, China in 2009, and currently is full professor with Northwestern Polytechnical University, China. His research interests are machine learning and its applications, such as pattern recognition, data mining, computer vision, image processing and information retrieval. He has published more than 100 papers in the following journals and conferences: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IJCV, *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Knowledge and Data Engineering*, ICML, NIPS, KDD, IJCAI, AAAI, ICCV, CVPR, and ACM MM. His papers have been cited more than 20000 times and the H-index is 84. He is now serving as associate editor or PC member for several prestigious journals and conferences in the related fields.

**Xuelong Li** (Fellow, IEEE) is a full professor with the School of Artificial Intelligence, Optics and Electronics (iOPEN), and the Key Laboratory of Intelligent Interaction and Applications (Ministry of Industry and Information Technology), Northwestern Polytechnical University, Xian 710119, Shaanxi, China.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Harbin Institute of Technology. Downloaded on February 28,2025 at 07:01:55 UTC from IEEE Xplore. Restrictions apply.