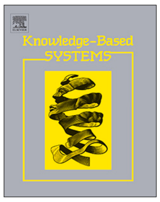
Contents lists available at ScienceDirect

Knowledge-Based Systems

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



CAPKM++2.0: An upgraded version of the collaborative annealing power k-means++ clustering algorithm[☆]

Hongzong Li [a], Jun Wang [a,b,*]

[a] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
[b] School of Data Science, City University of Hong Kong, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 1 November 2022
Received in revised form 14 December 2022
Accepted 25 December 2022
Available online xxxx

Keywords:
k-means clustering
k-means++
Power *k*-means
Collaborative annealing power *k*-means++

ABSTRACT

The collaborative annealing power k-means++ (CAPKM++) clustering algorithm has been recently proposed based on multiple modules by minimizing annealed power-mean functions. This paper presents an upgraded version of CAPKM++ called CAPKM++2.0. Different from CAPKM++ where the anchor points of surrogate functions for majorizing the power-mean functions are re-initialized and minimized repeatedly after annealing, CAPKM++2.0 re-initializes the weights of the majorization function during annealing. In addition, unlike CAPKM++ that minimizes the majorization function of the power-mean sum, CAPKM++2.0 adds an inner loop to minimize the power-mean sum iteratively and locally at every annealing step. Ablation study results are discussed to justify the adoption of the power-mean and the collaboration of multiple modules. Experimental results on sixteen benchmark datasets are elaborated to demonstrate the superior clustering performance of the upgraded algorithm compared with its predecessor and six other mainstream algorithms in terms of cluster validity indices and algorithmic complexities.

1. Introduction

As an important procedure of data processing, clustering is to group similar data into homogeneous clusters [1], with widespread applications, such as image segmentation [2,3], text mining [4], bioinformatic analysis [5]. Numerous clustering algorithms have been proposed and they may be classified into several categories from different perspectives, as shown in Fig. 1; e.g., divisive hierarchical clustering [6], agglomerative hierarchical clustering [7], *k*-means [8–10], *k*-medoids [11], *k*-harmonic means [12], spectral clustering [3,13,14], distribution clustering [15,16], density clustering [17,18], subspace clustering [19–21], feature-weighted clustering [22–24], probabilistic clustering [25], fuzzy clustering [26–28], cardinality-constrained clustering [29–31], capacitated clustering [32,33], must-link and cannot-link constrained clustering [34], and rank-constrained clustering [35].

Lloyd's algorithm [8] is a classic *k*-means algorithm that minimizes the sum of squared distances between each point and its assigned cluster center in a greedy way. The *k*-means and *k*-means-type algorithms are very popular and widely used in data clustering and analysis, owing to their efficiency and simplicity. However, their clustering qualities vary depending on the initialization of centers. To overcome this limitation, many alternative methods have been proposed, such as initialization improvements [11,36–38] and objective function improvements [9,12, 39–41]. Of particular interest, the power *k*-means (PKM) algorithm [9] clusters data by minimizing the majorization function of an annealed power-mean sum. Nevertheless, its clustering performance still depends on initialization. To eliminate the initialization dependence, CAPKM++ employs multiple PKM modules initialized by using *k*-mean++ and re-initialized repeatedly and collaboratively using a particle swarm optimization rule after each annealing process [10]. Although CAPKM++ outperforms many baselines, the clustering efficiency could be further improved by carrying out cluster re-initialization during the annealing process.

In this paper, we propose an upgraded version of CAPKM++ called CAPKM++2.0. Instead of re-initializing cluster centers after annealing, CAPKM++2.0 re-initializes the weights in the majorization function during annealing. Additionally, CAPKM++2.0 minimizes the power-mean functions directly rather than their

☆ This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region of China under Grants 11202318, 11202019, and 11203721; and in part by the InnoHK initiative, the Government of the Hong Kong Special Administrative Region, and Laboratory for AI-Powered Financial Technologies.

* Corresponding author at: Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong.
E-mail addresses: hongzli2-c@my.cityu.edu.hk (H. Li), jwang.cs@cityu.edu.hk (J. Wang).

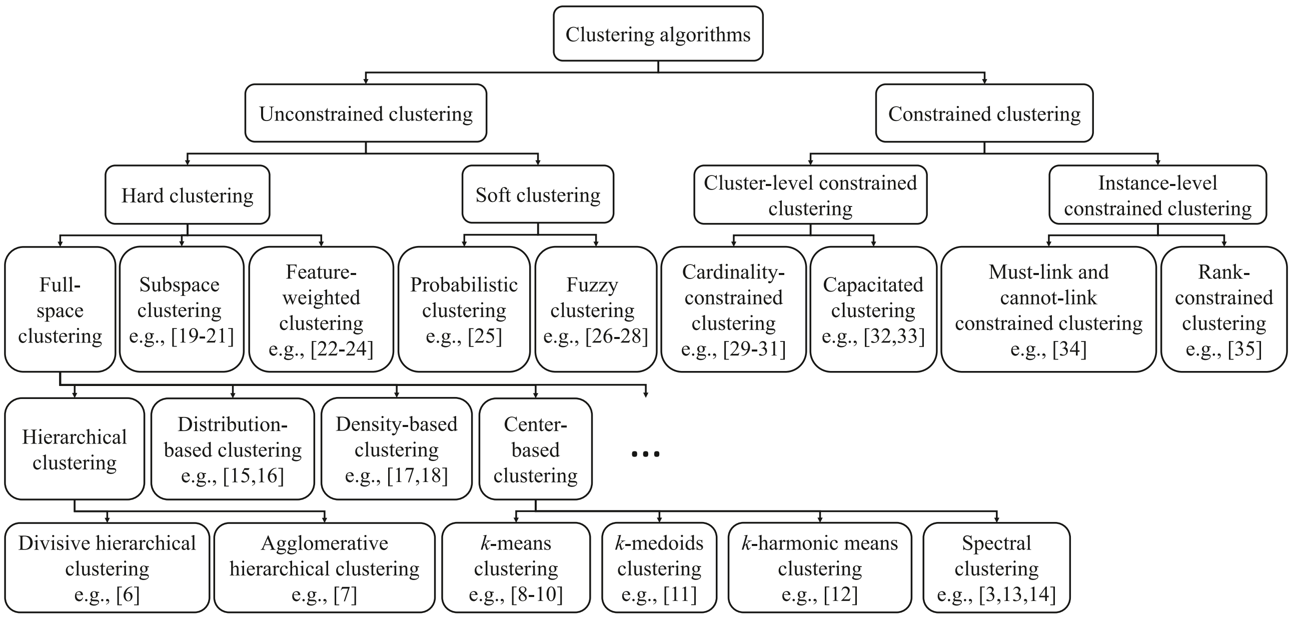

Clustering algorithms
Unconstrained clustering
Constrained clustering
Hard clustering
Soft clustering
Cluster-level constrained clustering
Instance-level constrained clustering
Full-space clustering
Subspace clustering e.g., [19-21]
Feature-weighted clustering e.g., [22-24]
Probabilistic clustering e.g., [25]
Fuzzy clustering e.g., [26-28]
Cardinality-constrained clustering e.g., [29-31]
Capacitated clustering e.g., [32,33]
Must-link and cannot-link constrained clustering e.g., [34]
Rank-constrained clustering e.g., [35]
Hierarchical clustering
Distribution-based clustering e.g., [15,16]
Density-based clustering e.g., [17,18]
Center-based clustering
...
Divisive hierarchical clustering e.g., [6]
Agglomerative hierarchical clustering e.g., [7]
k-means clustering e.g., [8-10]
k-medoids clustering e.g., [11]
k-harmonic means clustering e.g., [12]
Spectral clustering e.g., [3,13,14]

Fig. 1. A taxonomical classification of clustering algorithms.

majorization function as in PKM and CAPKM++. The novelties of this work are summarized as follows.

i. Carrying out re-initialization during annealing rather than after it enables more efficient clustering with reduced spatial and temporal complexities.
ii. Minimizing the power-mean sum instead of its majorization function enables to further improve cluster performance in terms of algorithmic efficiency.

The remainder of this paper is organized as follows. Section 2 provides necessary preliminary information on problem formulation, PKM, and CAPKM++. Section 3 describes the proposed CAPKM++2.0 algorithm. Section 4 reports experimental results on sixteen datasets. Section 5 concludes the paper.

2. Preliminaries

2.1. Problem statement

Given a data set $X = \{x_1, \dots, x_n\}$, a k-means clustering algorithm partitions the data into k clusters by minimizing the within-cluster distance with the following objective function [8]:

$$f(\Theta) = \sum_{i=1}^{n} \min_{1\le j\le k} \|x_i - \theta_j\|_2^2, \tag{1}$$

where $\theta_j \in \Re^m$ is the center (centroid) of cluster j $(j = 1, 2, \dots, k)$, $\Theta = [\theta_1, \dots, \theta_k]$ is the cluster centers matrix.

2.2. Fundamentals of the PKM algorithm

Since it is hard to minimize (1) directly, the PKM algorithm utilizes the following smoother function as a surrogate function of (1) [9]:

$$f_s(\Theta) := \sum_{i=1}^{n}\left(\frac{1}{k}\sum_{j=1}^{k}\|x_i-\theta_j\|_2^{2s}\right)^{\frac{1}{s}}, \tag{2}$$

where $s < 0$ denotes a power parameter. Let us denote $M_s(y) = \left(\frac{1}{k}\sum_{i=1}^{k} y_i^s\right)^{\frac{1}{s}}$ as the power mean function. Since $\lim_{s\to-\infty} M_s(y) = \min_{1\le j\le k}\{y_1, \dots, y_k\}$ [42], $\lim_{s\to-\infty} f_s(\Theta) = \sum_{i=1}^{n}\min_{1\le j\le k}\|x_i-\theta_j\|_2^2$; i.e., $\lim_{s\to-\infty} f_s(\Theta) = f(\Theta)$.

If $s < 1$, the power-mean functions are concave and satisfy the following inequality [9]:

$$\begin{aligned} &f_s(\Theta(t+1)) \\ \le\ & f_s(\Theta(t)) - \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}(t)\left\|x_i-\theta_j(t)\right\|_2^2 \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}(t)\|x_i-\theta_j(t+1)\|_2^2. \end{aligned} \tag{3}$$

where $w_{ij}(t)$ is the weight defined by the partial derivative of $M_s(\|x_i-\theta_j(t)\|^2)$:

$$w_{ij}(t) = \frac{\frac{1}{k}\|x_i-\theta_j(t)\|_2^{2(s-1)}}{(\frac{1}{k}\sum_{l=1}^{k}\|x_i-\theta_l(t)\|_2^{2s})^{1-\frac{1}{s}}}. \tag{4}$$

Note that the first two terms on the right-hand side of (3) are independent of $\Theta(t+1)$ and the last term of (3) is a convex majorization function of $\Theta(t+1)$.

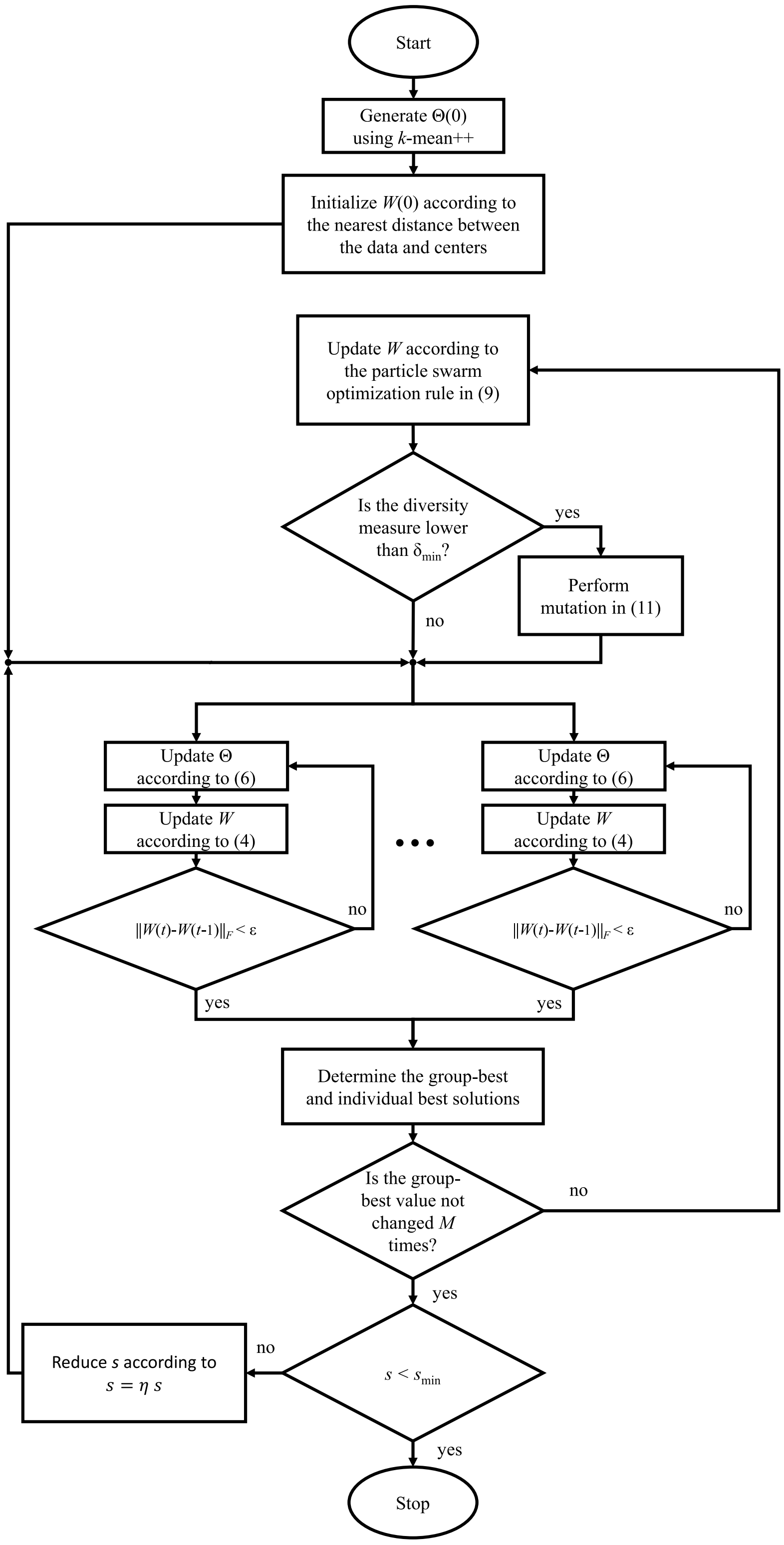

Start
Generate Θ(0) using k-mean++
Initialize W(0) according to the nearest distance between the data and centers
Update W according to the particle swarm optimization rule in (9)
Is the diversity measure lower than δmin?
yes
Perform mutation in (11)
no
Update Θ according to (6)
Update W according to (4)
‖W(t)-W(t-1)‖F < ε
no
yes
Determine the group-best and individual best solutions
Is the group-best value not changed M times?
no
yes
s < smin
no
Reduce s according to s = η s
yes
Stop

Fig. 2. Flowchart of CAPKM++2.0.

Rather than minimizing the concave power-mean functions, PKM minimizes the convex majorization function [9]:

$$\hat{f}_s(\Theta) = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}(t)\|x_i - \theta_j(t+1)\|_2^2. \tag{5}$$

The cluster center updating rule is defined as follows:

$$\theta_j(t+1) = \frac{1}{\sum_{i=1}^{n} w_{ij}(t)} \sum_{i=1}^{n} w_{ij}(t)x_i. \tag{6}$$

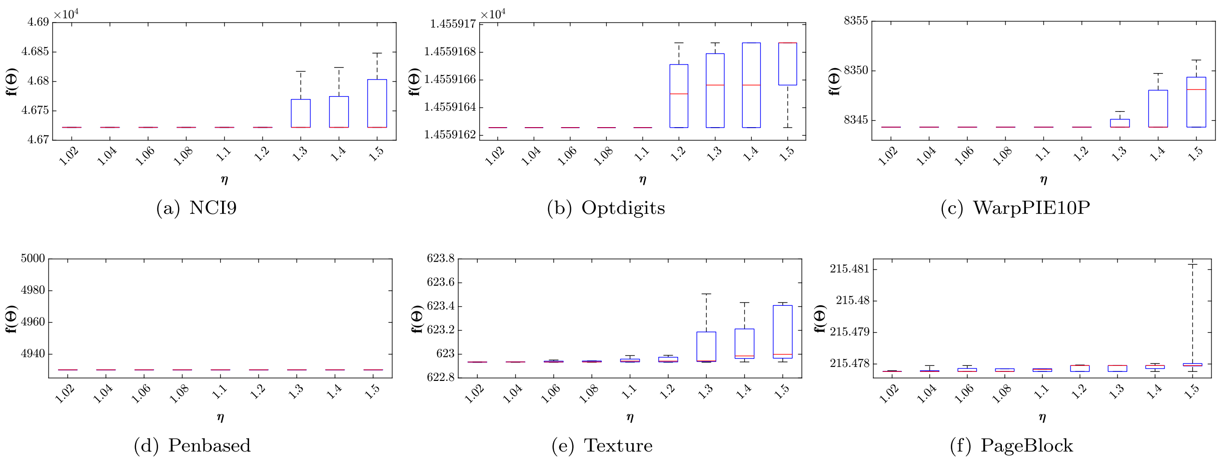


Fig. 3. Monte Carlo test results over 50 runs using CAPKM++2.0 ($N = 2$ and $M = 10$) with several values of η on six datasets.

Table 1
Information about the sixteen benchmark datasets and the corresponding hyper-parameter values used in CAPKM++ and CAPKM++2.0, where the first twelve datasets are used in [10].

Datasets	n	m	k	Algorithms	N	M
NCI9	60	9712	9	CAPKM++	40	30
				CAPKM++2.0	4	10
Lymphoma	96	4026	9	CAPKM++	10	30
				CAPKM++2.0	2	5
ORL10P	100	10304	10	CAPKM++	4	30
				CAPKM++2.0	2	5
WarpPIE10P	210	2420	10	CAPKM++	10	30
				CAPKM++2.0	2	10
Segment	2310	19	7	CAPKM++	3	30
				CAPKM++2.0	2	5
SpamBase	4597	57	2	CAPKM++	5	5
				CAPKM++2.0	2	5
PageBlocks	5472	10	5	CAPKM++	12	30
				CAPKM++2.0	2	15
Texture	5500	40	11	CAPKM++	40	15
				CAPKM++2.0	4	15
Optdigits	5620	65	10	CAPKM++	40	30
				CAPKM++2.0	2	10
Satimage	6435	36	7	CAPKM++	5	30
				CAPKM++2.0	2	5
COIL2000	9822	85	2	CAPKM++	2	5
				CAPKM++2.0	2	5
Penbased	10992	16	10	CAPKM++	2	15
				CAPKM++2.0	2	5
WQ-Red	1599	11	6	CAPKM++	10	10
				CAPKM++2.0	2	5
WQ-White	4898	11	11	CAPKM++	200	50
				CAPKM++2.0	4	30
Banana	5300	2	2	CAPKM++	12	30
				CAPKM++2.0	2	5
Phoneme	5404	5	2	CAPKM++	12	30
				CAPKM++2.0	2	5

At each step, the power parameter s is devalued according to the following cooling schedule:

$$s(t+1) = \eta s(t), \tag{7}$$

where $s(0) < 0$ and $\eta > 1$.

It is demonstrated in [9] that PKM performs better than Lloyd's algorithm [8] and k-harmonic means [12].

2.3. Fundamentals of the CAPKM++ algorithm

To overcome the limitation of the PKM performance dependency on initialization, the CAPKM++ algorithm [10] is developed by integrating the k-mean++ and multiple PKM modules in a collaborative way. For better seeding, k-means++ is used to provide initial centers according to a probability defined as follows [37]:

$$P(x_i) = \frac{d(x_i)^2}{\sum_{x\in\mathcal{X}} d(x_i)^2}, \tag{8}$$

where x_i is sample i, $\mathcal{X}$ is a set of samples other than selected centers, $d(x_i)$ is the distance between x_i and its nearest center. To generate multiple centers as alternative clusters, multiple PKM modules are leveraged and repeatedly re-initialized using a particle swarm optimization rule in [43]:

$$v_i(t+1) = c_0 v_i(t) + c_1 r_1 (p_i^* - p_i(t)) + c_2 r_2 (p^* - p_i(t)), \tag{9a}$$

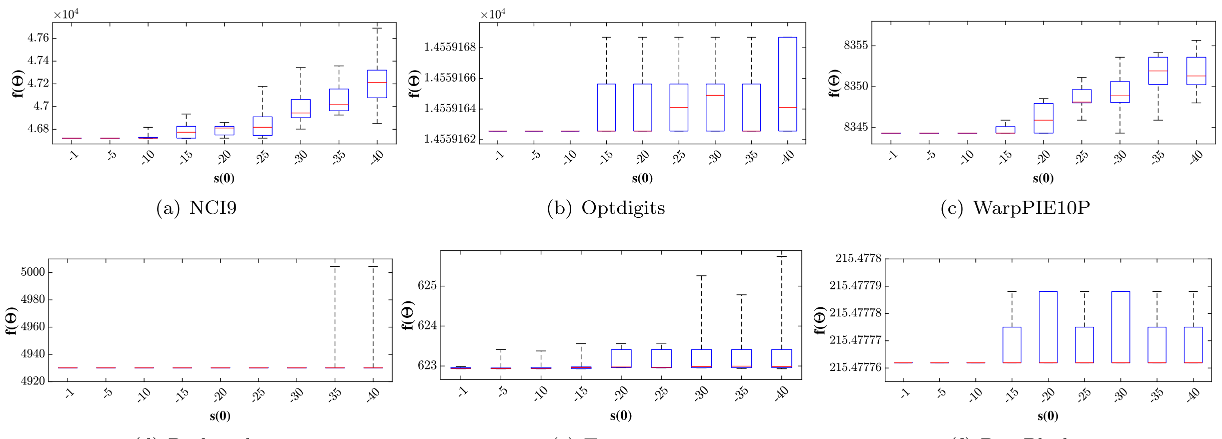
$$p_i(t+1) = p_i(t) + v_i(t+1), \tag{9b}$$

where $v_i(t)$ is a velocity determining the searching direction of the ith particle at the tth iteration, p_i^* is the current best position of the ith particle, $p_i(t)$ is the current position of the ith particle at the tth iteration, p^* is the current best position of a group (solution set), $c_0 \in [0, 1]$ is an inertia weight determining the weight of the previous velocity, $c_1 \in [0, 1]$ is a cognitive learning factor, $c_2 \in [0, 1]$ is a social learning factor, and $r_1, r_2 \in [0, 1]$ are two random numbers. A diversity measure is defined by:

$$\delta(\Theta) = \frac{1}{Nn}\sum_{j=1}^{N} \|\Theta^{(j)} - \Theta^*\|_2, \tag{10}$$

where N is the population size (i.e., the number of clustering configurations), and Θ^* is the best cluster centers matrix among a population of modules. CAPKM++ uses a wavelet mutation operator to diversify cluster centers. If the diversity is lower than a threshold (i.e., $\delta(\Theta) < \delta_{\min}$), the following wavelet mutation operation is performed for $i = 1, \ldots, k$, and $j = 1, \ldots, m$:

$$\theta_{ij}(t+1) = \begin{cases} \theta_{ij}(t) + \tau(\overline{\theta}_{ij} - \theta_{ij}(t)) & \tau > 0, \\ \theta_{ij}(t) + \tau(\theta_{ij}(t) - \underline{\theta}_{ij}) & \tau < 0, \end{cases} \tag{11}$$



(a) NCI9 (b) Optdigits (c) WarpPIE10P

(d) Penbased (e) Texture (f) PageBlock

Fig. 4. Monte Carlo test results over 50 runs using CAPKM++2.0 ($N = 2$ and $M = 10$) with several values of $s(0)$ on six datasets.

where $\overline{\theta}_{ij}$ and $\underline{\theta}_{ij}$ are the upper bound and lower bound of the searching space, and τ is defined by a wavelet function:

$$\tau = \frac{1}{\sqrt{a}} \exp -\frac{1}{2}(\frac{\psi}{a})^2 \cos(\frac{5\psi}{a}),$$

where ψ is randomly generated from interval $[-2.5a, 2.5a]$, $a = \exp(10(\ell/\ell_{\max}))$, and $\ell_{\max}$ is the maximum iteration.

3. CAPKM++2.0

In the PKM and CAPKM++ algorithms, the majorization function of power-mean sum in (5) is minimized at every step of annealing. Minimizing the power-mean function in (2) may attain better clustering results than minimizing its majorization function. To do so, in CAPKM++2.0, an inner loop is added to update the weights and the centers alternately until convergence at every step of annealing.

In CAPKM++, the re-initialization of cluster centers is performed after each annealing process until convergence. To improve the efficiency of global repositioning, CAPKM++2.0 re-initializes $W(0)$ of the majorization function in every module collaboratively using particle swarm optimization rule (9) at every annealing stage instead of re-initializing $\Theta(0)$ after annealing as in CAPKM++.

Fig. 2 shows a flowchart of the CAPKM++2.0 algorithm and Algorithm 1 details its pseudo-codes. In Step 1, N initial centers are generated by using the k-means++ algorithm. In Steps 2–7, the initial weights $W(0)$ are assigned according to the generated initial centers. In Steps 10–16, centers Θ and weights W are updated alternately until convergence. In Step 16, the Frobenius norm of the difference between the weight matrices in two consecutive inner-loop iterations is checked for convergence status. In Steps 17–19, the individual best weights $\tilde{W}^{(i)}$ are determined. In Steps 21–28, the best weights W^* are determined. In Steps 29–32, the weights are re-initialized using the particle swarm optimization rule (9). In Step 33, the diversity of the weights is measured. In Steps 34–36, a wavelet mutation operation is carried out if the diversity is lower than a threshold $\delta_{\min}$. In Step 38, the power parameter s is reduced with a preset discount factor η.

4. Experimental results

4.1. Setups

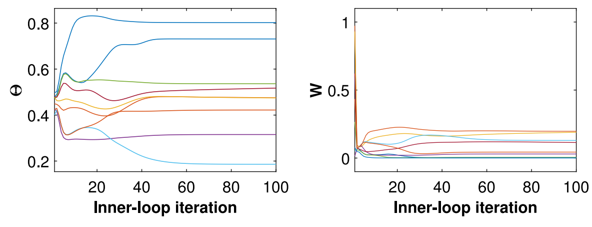
The experimental results are based on sixteen commonly used datasets as listed in Table 1: NCI9 [44], Lymphoma [44], ORL10P [44], WarpPIE10P [44], Segment [45], SpamBase [45], PageBlocks [45], Texture [45], Optdigits [45], Satimage [45], COIL2000 [45], Penbased [45], WineQuality-Red [45] (WQ-Red), WineQuality-White [45] (WQ-White), Banana [45], Phoneme [45].

The clustering performance of CAPKM++2.0 is compared with those of the following seven baselines: k-means (KM),[1] k-mean++ (KM++),[2] PKM [9], entropy weighted power k-means (EWPKM),[3] spectral clustering (SC),[4] hierarchical clustering (HC),[5] CAPKM++ [10]. The code of PKM is shared by the first author of [9]. The code of CAPKM++ is obtained from a link in [10]. The hierarchical clustering algorithm used is agglomerative, and clustering results are attained by cutting hierarchical cluster tree into k clusters.[6] The code of CAPKM++2.0 are available from https://github.com/HongzongLI-CS/CAPKM2.0-Github. The Euclidean distance is used as the dissimilarity metric in all algorithms.

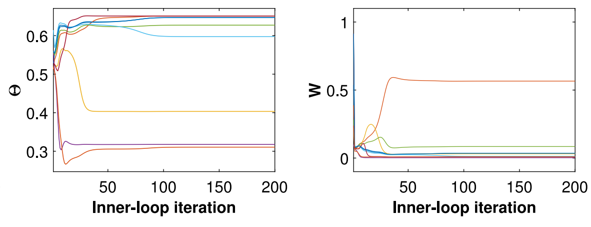
The performance evaluation criteria for the experimental results are based on nineteen internal cluster validity indices listed in Table 3 in [10] and three external indices described in subsection 4.1 in [10]. Since WGSS, CHI, XBI and TWI range widely, their values are normalized around 1 by dividing them with m, $(n-k)/(k-1)$, n, and mk, respectively, to facilitate the tabular presentation later.

Figs. 3 and 4 depict the box-plots of 50-run Monte Carlo test results using CAPKM++2.0 with different values of η and $s(0)$, respectively. As shown in Figs. 3 and 4, the objective function values reach their minima in most runs with $\eta = 1.2$ and $s(0) = -5$ on NCI9, $\eta = 1.1$ and $s(0) = -10$ on Optdigits, $\eta = 1.2$ and $s(0) = -10$ on WarpPIE10P, $\eta = 1.5$ and $s(0) = -30$ on Penbased, $\eta = 1.2$ and $s(0) = -1$ on Texture, $\eta = 1.1$ and $s(0) = -10$ on PageBlock. As the objective function values reach their elbow points $\eta \geq 1.1$ and $s(0) \geq -5$, $\eta = 1.1$ and $s(0) = -5$ in all the other experiments. $\epsilon = 0.001$ and $\delta_{\min} = 0.001$. In the particle swarm optimization rule in (9), $c_0 = c_1 = c_2 = 1$, as in CAPKM+++.

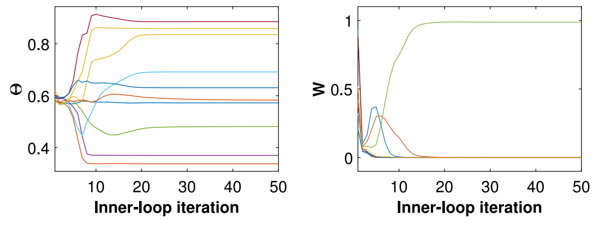
[1] https://www.mathworks.com/help/stats/kmeans.html?s_tid=srchtitle_kmean_1
[2] https://github.com/xuyxu/Clustering
[3] https://github.com/DebolinaPaul/EWP
[4] https://www.mathworks.com/help/stats/spectralcluster.html
[5] https://www.mathworks.com/help/stats/hierarchical-clustering.html?s_tid=srchtitle_hierarchical%20clustering_1
[6] https://www.mathworks.com/help/stats/cluster.html



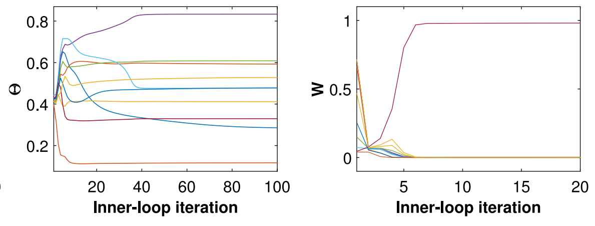
(a) NCI9



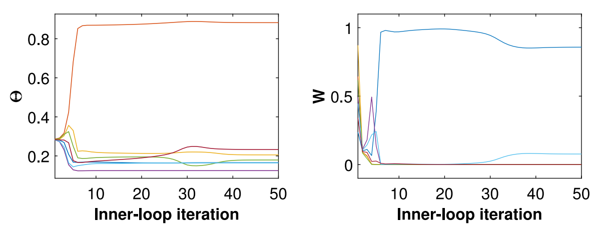
(b) Lymphoma



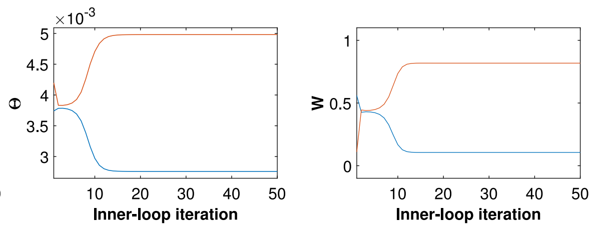
(c) ORL10P



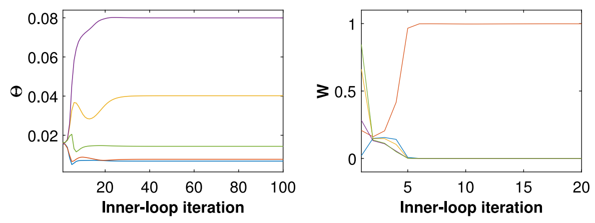
(d) WarpPIE10P



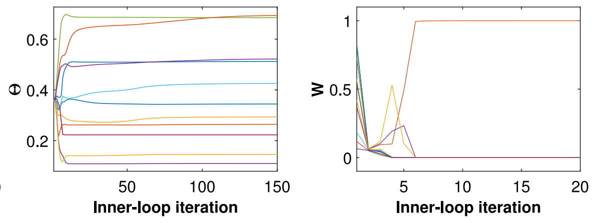
(e) Segment



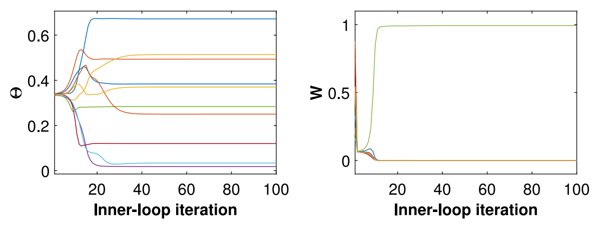
(f) Spambase



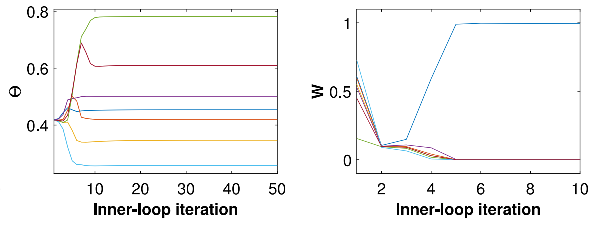
(g) PageBlock



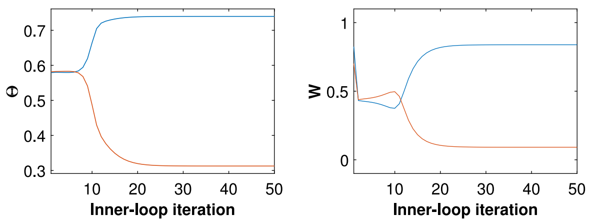
(h) Texture



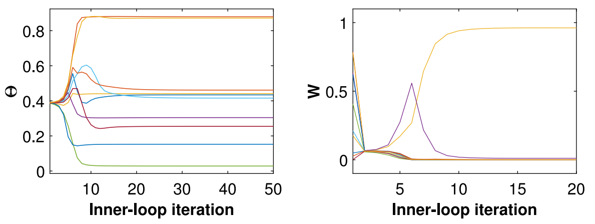
(i) Optdigits



(j) Satimage



(k) COIL2000



(l) Penbased

Fig. 5. Snapshots of the convergent centers Θ and the convergent weights W values in the inner loop (Steps 12–16) of CAPKM++2.0 on the first twelve datasets, where the lines in the left subplots portray the first feature values of k centers, and the lines in the right subplots portray the values of k weights.

4.2. Convergent behaviors

Figs. 5–6 depict the snapshots of the convergent Θ and the convergent W in the inner loop of CAPKM++2.0 (Steps 12–16) on the sixteen datasets, where the lines in the left subplot correspond to the first feature values of k centers, and the lines in the right subplot correspond to the value of k weights of a datum.

Fig. 7 depicts the monotonically decreasing values of $f_s(\Theta)$ in Eq. (2) corresponding to Θ and W in Figs. 5–6. Fig. 8 depicts the monotonically decreasing values of $f(\Theta)$ running CAPKM++2.0 on the sixteen datasets, where each line indicts the change in objective function value of a PKM module. It is shown that CAPKM++2.0 converges within 300 iterations.

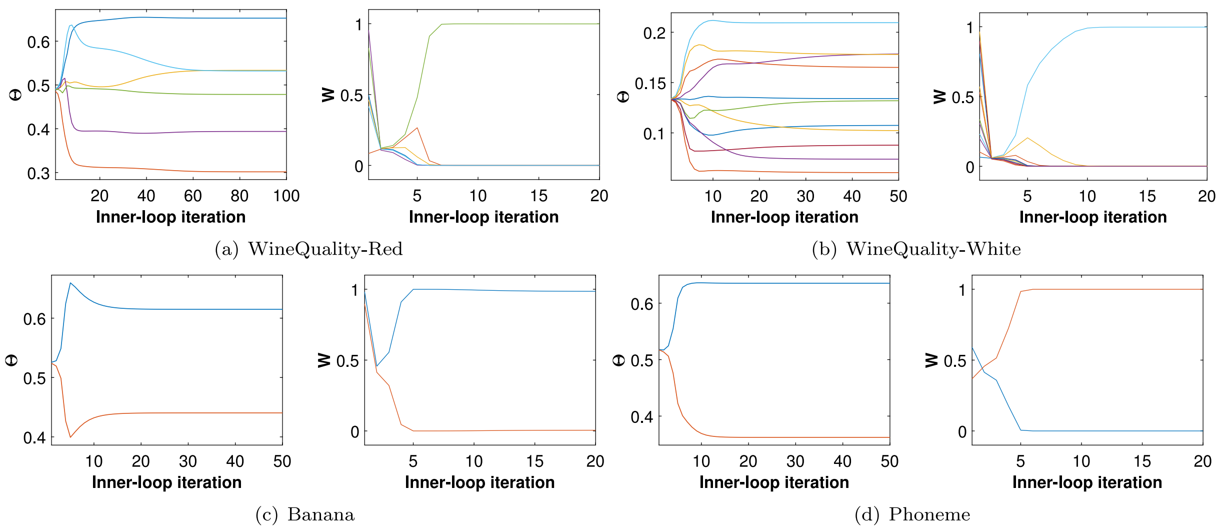


Fig. 6. Snapshots of the convergent centers Θ and the convergent weights W values in the inner loop (Steps 12–16) of CAPKM++2.0 on the last four datasets, where the lines in the left subplots portray the first feature values of k centers, and the lines in the right subplots portray the values of k weights.

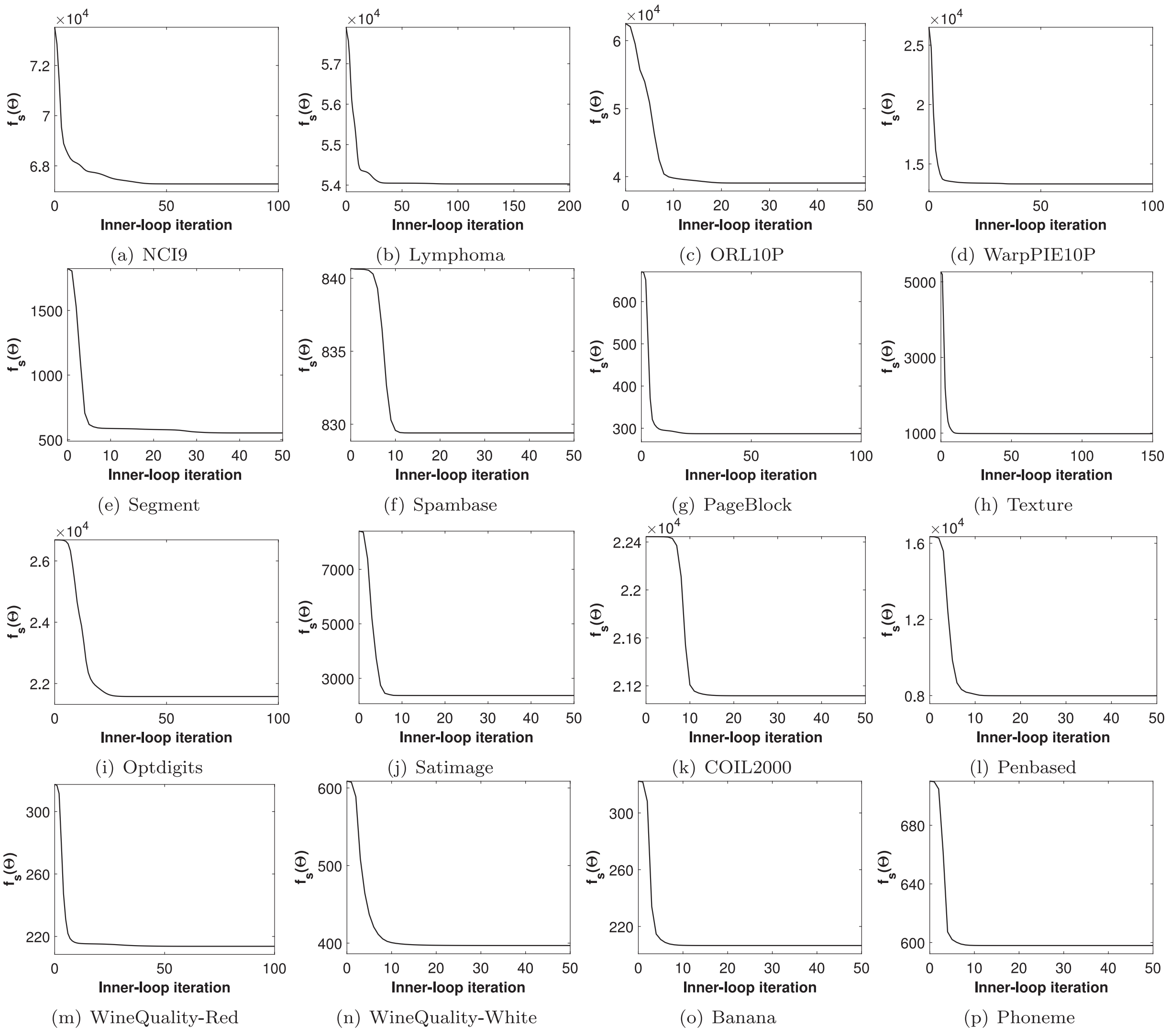
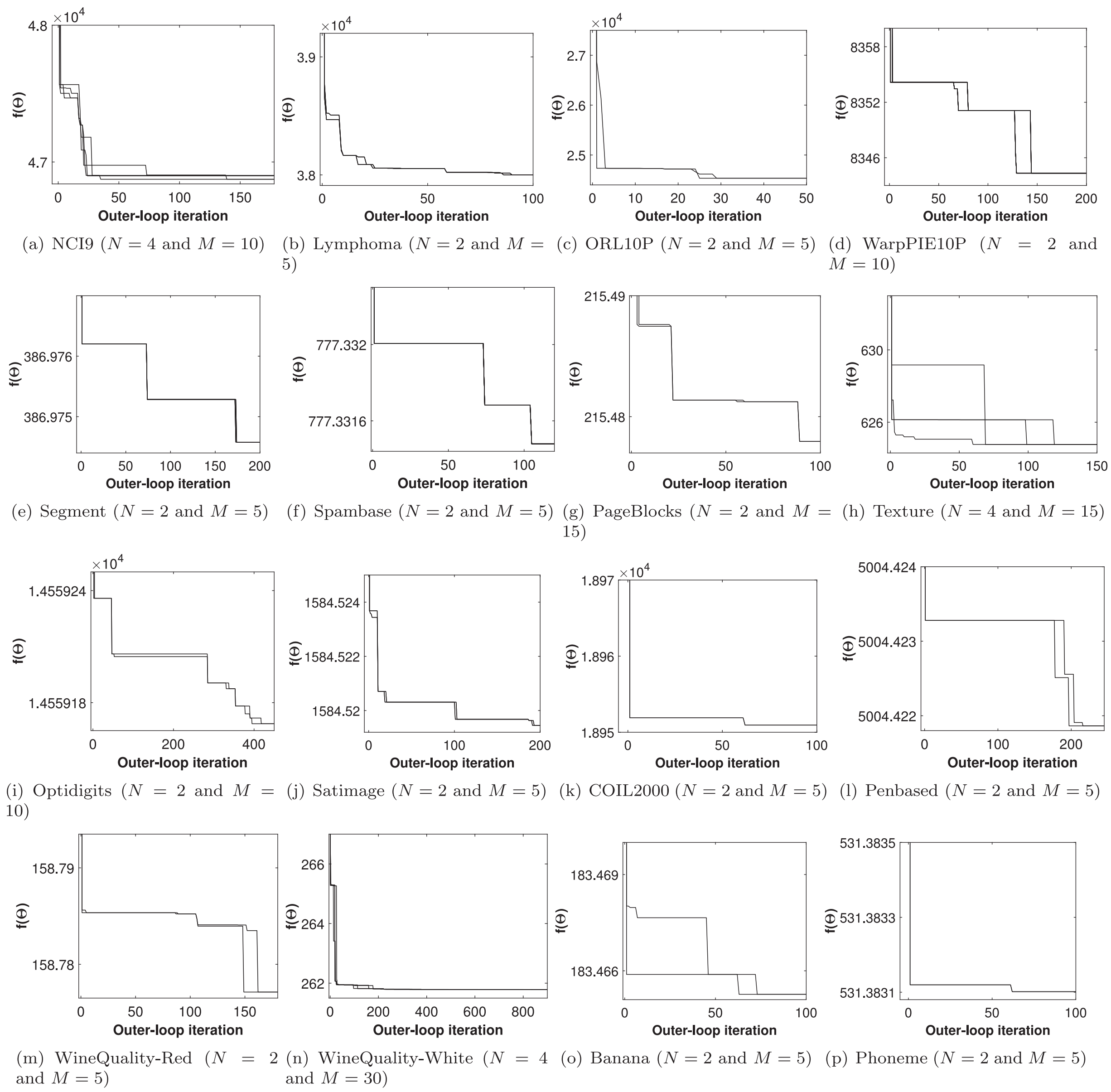


Fig. 7. Snapshots of the power-mean function in (2) values with $s = -5$ in the inner loop (Steps 12–16) of CAPKM++2.0 on the sixteen datasets.



(a) NCI9 ($N = 4$ and $M = 10$) (b) Lymphoma ($N = 2$ and $M = 5$) (c) ORL10P ($N = 2$ and $M = 5$) (d) WarpPIE10P ($N = 2$ and $M = 10$)

(e) Segment ($N = 2$ and $M = 5$) (f) Spambase ($N = 2$ and $M = 5$) (g) PageBlocks ($N = 2$ and $M = 15$) (h) Texture ($N = 4$ and $M = 15$)

(i) Optidigits ($N = 2$ and $M = 10$) (j) Satimage ($N = 2$ and $M = 5$) (k) COIL2000 ($N = 2$ and $M = 5$) (l) Penbased ($N = 2$ and $M = 5$)

(m) WineQuality-Red ($N = 2$ and $M = 5$) (n) WineQuality-White ($N = 4$ and $M = 30$) (o) Banana ($N = 2$ and $M = 5$) (p) Phoneme ($N = 2$ and $M = 5$)

Fig. 8. The convergent behavior of the CAPKM++2.0 algorithm on the sixteen datasets, where each line indicts the change in objective function value of a PKM module.

4.3. CAPKM++2.0 with or without inner-loop iterations

Fig. 9 depicts the 200-run Monte Carlo test results of CAPKM++2.0 with and without any inner-loop iteration, where N and M take several values. The ablation study shows that CAPKM++2.0 outperforms it in the absence of inner-loop iteration. It demonstrates that inner-loop iteration makes CAPKM++2.0 more efficient.

4.4. CAPKM++2.0 vs. KM, PKM, and CKM++

Fig. 10 shows 100-run Monte Carlo test results of KM in multiple times, PKM in multiple times, collaborative k-means++ (CKM++), CAPKM++, and CAPKM++2.0, where $M = 5$, CKM++ denotes an algorithm based on multiple KM modules re-initialized repeatedly similar to CAPKM++. CKM++, CAPKM++, and CAPKM++2.0 are collaborative methods. CKM++ can be viewed as a degenerated version of CAPKM++ and CAPKM++2.0 that sets $s(0) = -\infty$. Fig. 10 shows that the objective function values decrease or reach their minima in all algorithms as N increases, whereas CKM++ and CAPKM++ outperform their counterparts KM++ and PKM, and CAPKM++2.0 achieves the best results among the five algorithms. It demonstrates the high efficiency of collaborative re-initialization.

Fig. 10 also shows that CAPKM++ performs much better than CKM++, implying that the annealing is important as well as the collaboration. Fig. 11 depicts the average objective function values using CKM++ and CAPKM++2.0 with several values of N and M over 100 runs on seven datasets. As shown in Fig. 11, CAPKM++2.0 outperforms CKM++ consistently and significantly, demonstrating the effectiveness of annealing in CAPKM++2.0.

Algorithm 1: CAPKM++2.0

```
Input: Population size $N$, velocity vector $v^{(i)} \in [-1, 1]^{mk}$,
       termination criterion $M$, termination counter $l$,
       particle/group best weights $\tilde{W}^{(p)}/\ W^*$,
       $f(\tilde{W}^{(p)}) = f(W^*) = \infty$, initial power parameter $s(0)$,
       discount factor $\eta$, particle swarm optimization
       parameters $c_0$, $c_1$ and $c_2$, input data $X \in \mathbb{R}^{n\times m}$ .
Output: $W^*$.
Select the initial centers $[\Theta^{(1)}(0), ..., \Theta^{(N)}(0)] \in \{0, 1\}^{mk\times N}$
  according to Eq. (8);
for $p = 1$ to $N$ do
    for $i = 1$ to $n$ do
        $j \leftarrow \underset{1\le j\le k}{\mathrm{argmin}} ||x_i - \theta_j^{(p)}(0)||_2^2$;
        $w_{ij}^{(p)}(0) \leftarrow 1$;
    end
end
repeat
    while $l \le M$ do
        for $p = 1$ to $N$ do
            $t \leftarrow 1$;
            repeat
                Update centers $\Theta^{(p)}(t)$ according to Eq. (6);
                Update weights $W^{(p)}(t)$ according to Eq. (4);
                $t \leftarrow t + 1$;
            until $||W^{(p)}(t) - W^{(p)}(t-1)||_F < \epsilon$;
            if $f(W^{(p)}) < f(\tilde{W}^{(p)})$ then
                $\tilde{W}^{(p)} \leftarrow W^{(p)}$;
            end
        end
        $i^* = \arg\min_i\{f(W^{(1)}), ..., f(W^{(i)}), ..., f(W^{(N)}\}$;
        if $f(W^{(i^*)}) < f(W^*)$ then
            $W^* \leftarrow W^{(i^*)}$;
            $l \leftarrow 0$;
        else
            $l \leftarrow l + 1$;
        end
        for $i = 1$ to $N$ do
            Update velocity according to Eq. (9a);
            Update weights according to Eq. (9b);
        end
        Compute diversity $\delta(W)$ according to Eq. (10);
        if $\delta(W) < \delta_{\min}$ then
            Perform mutation according to Eq. (11);
        end
    end
    $s(t+1) \leftarrow \eta s(t)$;
until $s < s_{\min}$;
return $W^*$.
```

4.5. CAPKM++2.0 vs. CAPKM++

Figs. 12–13 depict 50-run Monte Carlo test results using CAPKM++ and CAPKM++2.0 with different N and M on the sixteen benchmark datasets, where the results in Fig. 12 are based on datasets used in [10]. Table 1 tabulates the values of the two hyper-parameters in CAPKM++ and CAPKM++2.0 (i.e., N and M), where CAPKM++ and CAPKM++2.0 reach the highest consistent results with zero standard deviation. It shows that CAPKM++2.0 needs much smaller values of N and M than CAPKM++ (i.e., 2 or 4 vs. 2 to 40 for N, 5 or 10 vs. 5 to 50 for M), indicating the significant reduction of algorithmic complexities on many datasets.

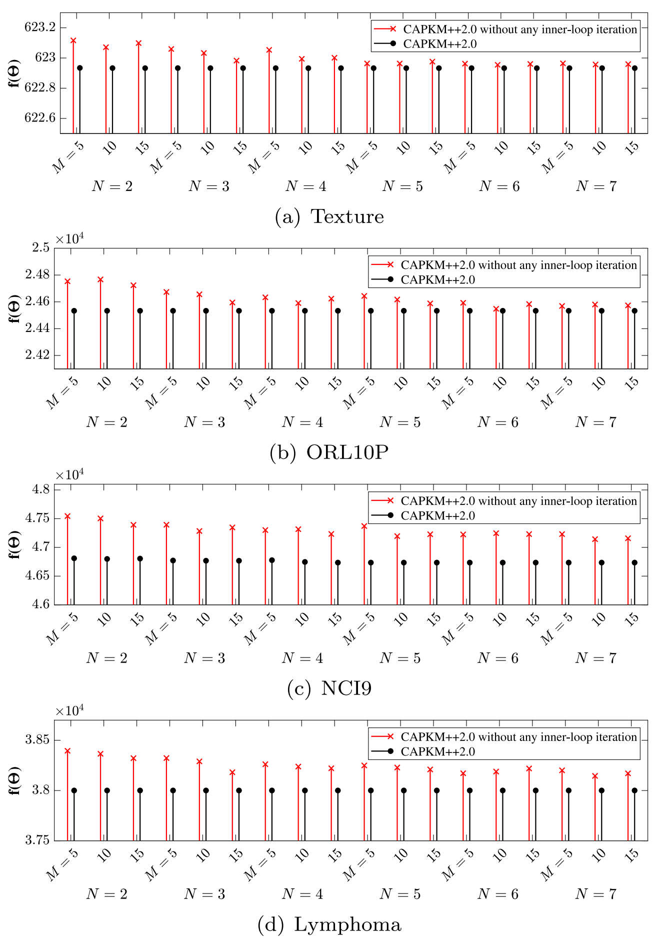


Fig. 9. Average objective function values over 100 runs using CAPKM++2.0 with and without any inner-loop iteration.

4.6. CAPKM++2.0 vs. Seven baselines

Tables 2–3 tabulate the means and standard deviations of the clustering results in terms of nineteen internal indices and three external indices over 50 runs by using CAPKM++2.0, CAPKM++, and six competing algorithms on the last four datasets in Table 1, where the best and second-best results are bold and underlined, respectively. Specifically, CAPKM++2.0 achieves 46 best and 11 second-best means out of 84 entries (i.e., 54.76% and 67.86% for the best and the best plus the second-best), while the CAPKM++ ranks in second place, achieving 24 best and 28 second-best means (i.e., 28.57% and 61.90%). In addition, the standard deviations of the results using CAPKM++2.0 are zero, indicating the highest consistency of the algorithm.

5. Concluding remarks

This paper presents an upgraded version of CAPKM++ for k-means clustering. With reduced algorithmic complexities,

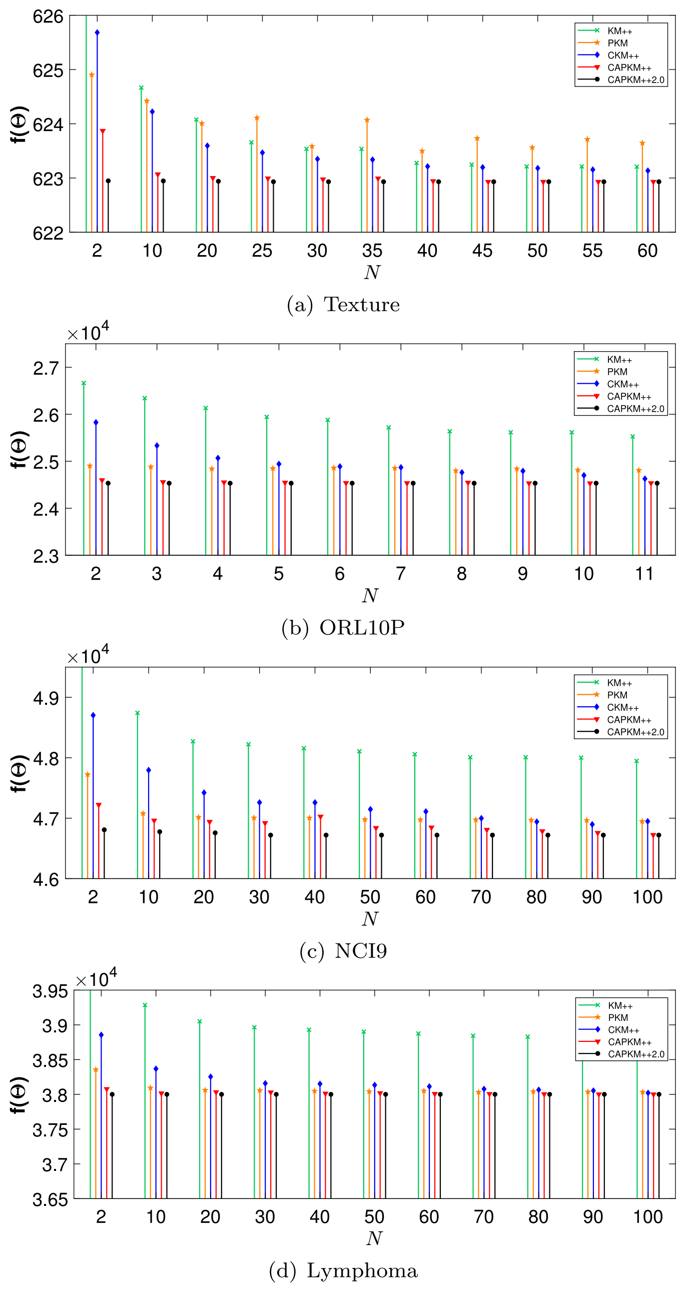


Fig. 10. Average objective function values over 100 runs using KM++ over N times, PKM over N times, CKM++, CAPKM++, and CAPKM++2.0 with several values of N.

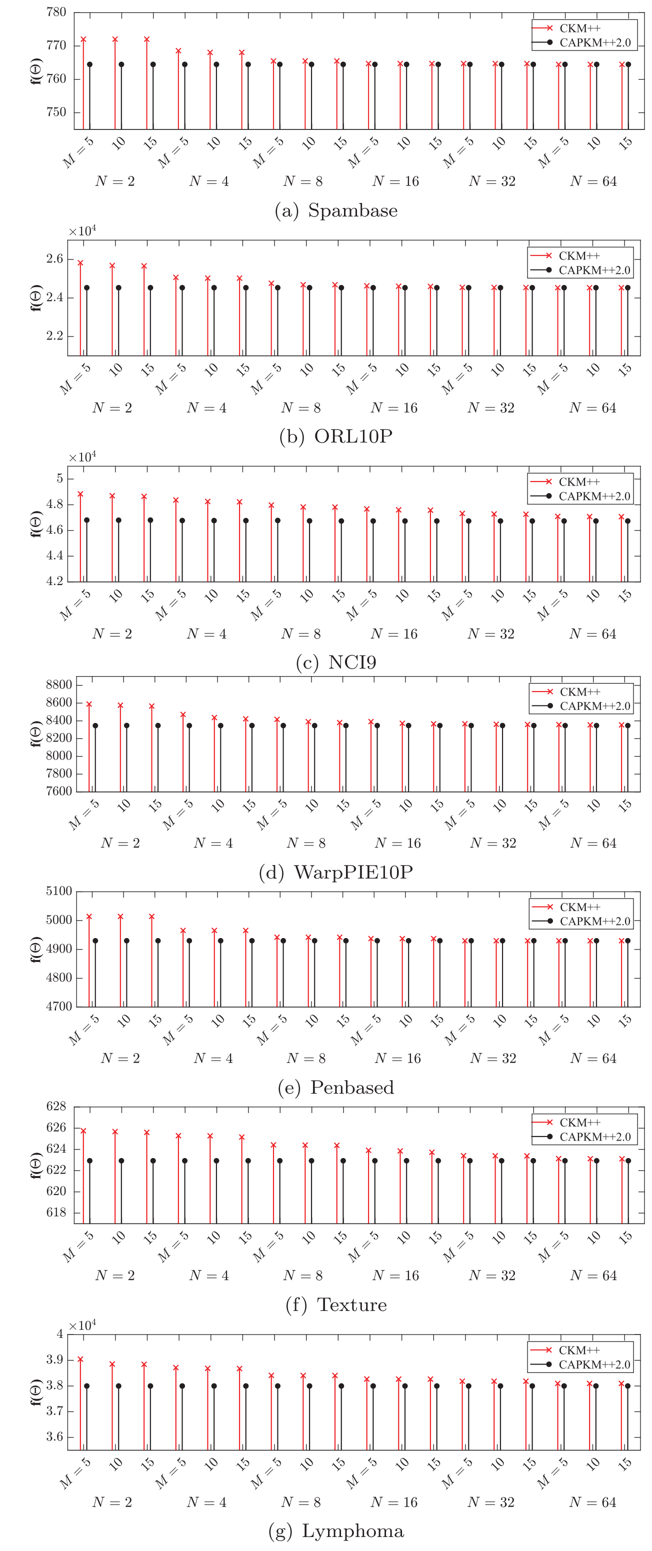
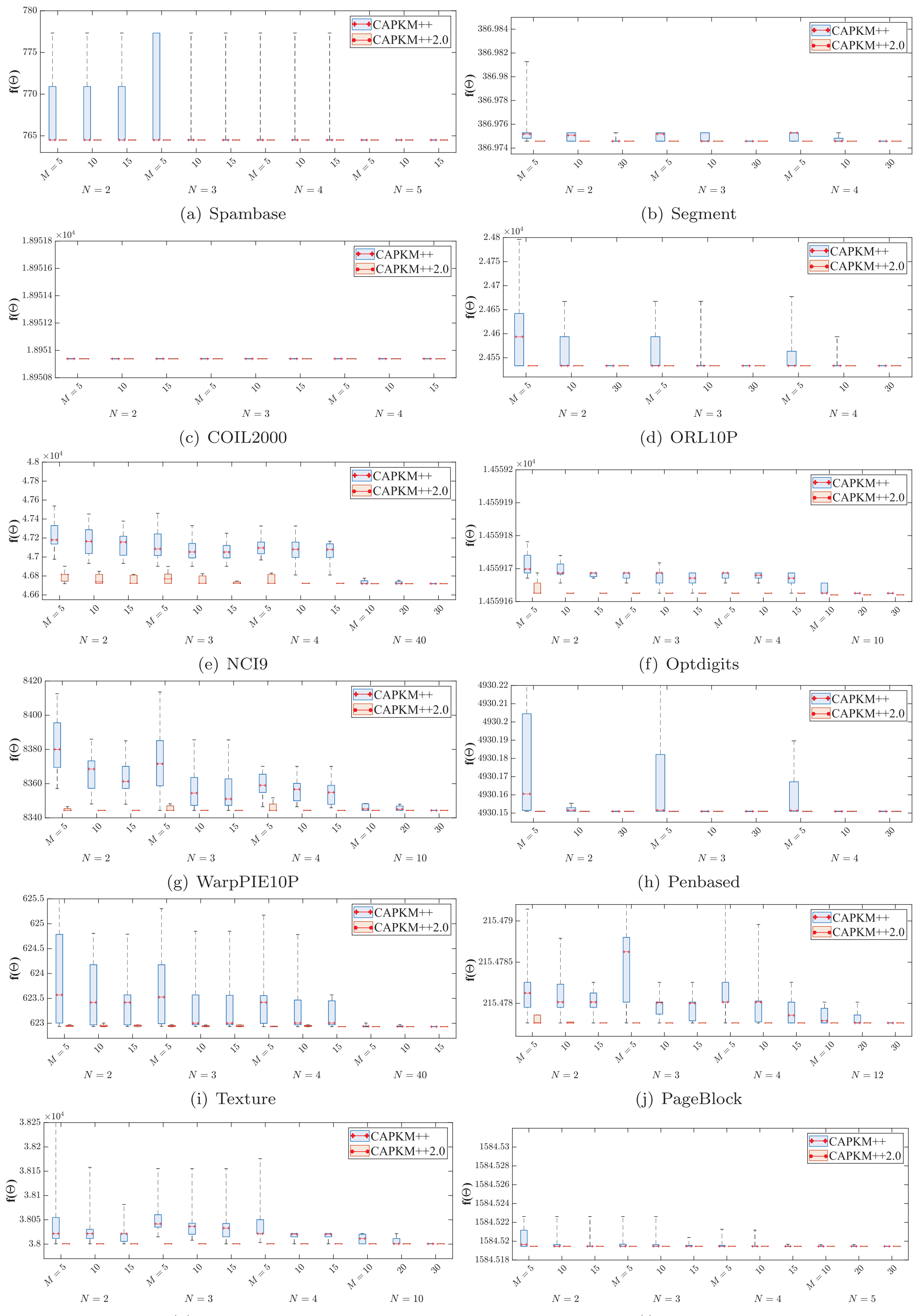


Fig. 11. Average objective function values over 100 runs using CKM++ and CAPKM++2.0 with several values of N and M on seven datasets.

CAPKM++2.0 is more efficient than CAPKM++, as it minimizes the power-mean functions directly rather than minimizing their majorization function and re-initializes the weights of the majorization function during annealing instead of re-initializing the anchor points after annealing. The experimental results on sixteen datasets demonstrate that the upgraded algorithm statistically outperforms many competing algorithms in terms of cluster validity indices and algorithmic complexities. Further research may aim at the improvements of its scalability, the extensions for fuzzy clustering, and the applications in engineering and finance.



(a) Spambase (b) Segment (c) COIL2000 (d) ORL10P (e) NCI9 (f) Optdigits (g) WarpPIE10P (h) Penbased (i) Texture (j) PageBlock (k) Lymphoma (l) Satimage

Fig. 12. Monte Carlo test results using CAPKM++ and CAPKM++2.0 with several values of N and M on the first twelve datasets.

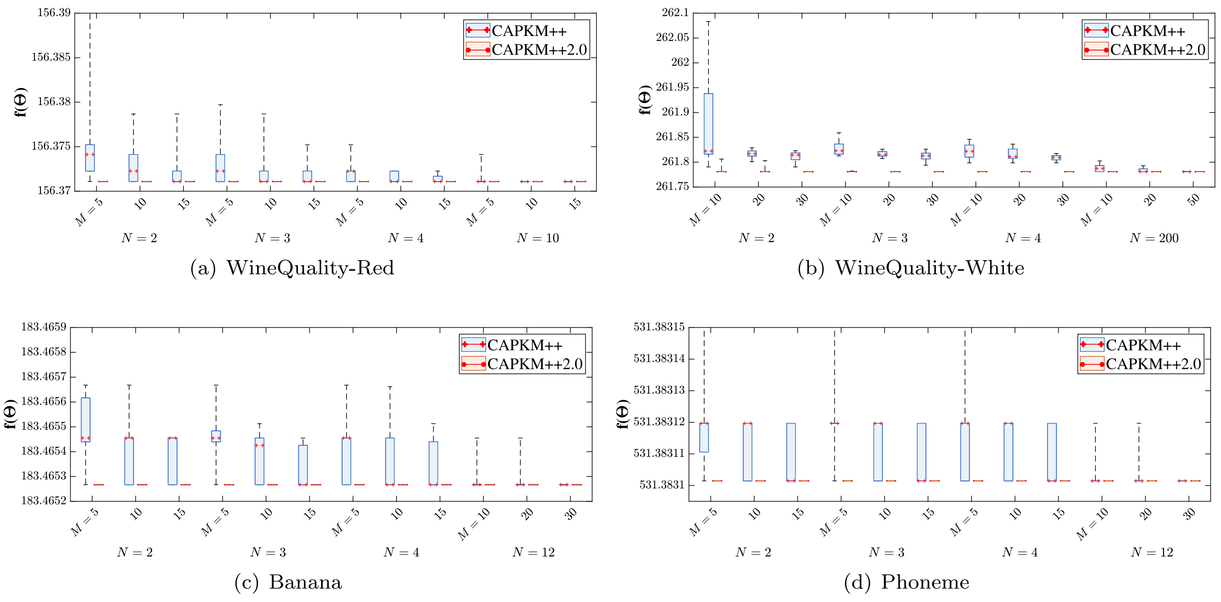


Fig. 13. Monte Carlo test results using CAPKM++ and CAPKM++2.0 with several values of N and M on the last four datasets.

Table 2
The mean values and standard deviations of internal and external cluster validity indices resulting from CAPKM++2.0, CAPKM++, and six baselines on WQ-White and WQ-Red, where $N = 2$ and $M = 5$ on WQ-Red, and $N = 4$ and $M = 30$ on WQ-White in CAPKM++ and CAPKM++2.0.

WQ-White	KM	KM++	PKM	EWPKM	SC	HC	CAPKM++	CAPKM++2.0
WGSS↓	24.0673 ± 0.2049	24.5069 ± 2.1869	24.1219 ± 0.0312	36.4648 ± 0.7859	28.6233 ± 0.7560	27.0748 ± 0	23.801 ± 0.0008	**23.7981** ± **0**
MRI↓	0.6218 ± 0.0042	0.6211 ± 0.0038	0.6210 ± 0.0011	0.7848 ± 0.0120	0.6974 ± 0.0120	0.6672 ± 0	0.6142 ± 0.0010	**0.6130** ± **0**
GPI↓	0.0265 ± 0.0006	0.0266 ± 0.0011	0.0260 ± 0.0002	0.0562 ± 0.0015	0.0676 ± 0.0096	0.0409 ± 0	0.0260 ± 0.0002	**0.0258** ± **0**
BHGI↑	0.7119 ± 0.0100	0.7138 ± 0.0081	0.7109 ± 0.0017	0.3846 ± 0.0226	0.5809 ± 0.0216	0.6272 ± 0	0.7294 ± 0.0026	**0.7324** ± **0**
CI↓	0.1305 ± 0.0044	0.1296 ± 0.0036	0.1307 ± 0.0008	0.2646 ± 0.0097	0.1811 ± 0.0104	0.1638 ± 0	0.1226 ± 0.0010	**0.1215** ± **0**
TI↑	0.3057 ± 0.0090	0.3078 ± 0.0082	0.3013 ± 0.0007	0.1644 ± 0.0105	**0.3287** ± **0.0044**	0.2939 ± 0	0.3196 ± 0.0018	0.3215 ± 0
DGI↑	0.0953 ± 0.0395	0.1047 ± 0.0335	0.0869 ± 0.0258	0.0675 ± 0.0145	0.0696 ± 0.0206	**0.3741** ± **0**	0.0938 ± 0.0232	0.0844 ± 0
RLI↑	0.1993 ± 0.0045	0.1992 ± 0.0044	0.1938 ± 0.0005	0.1919 ± 0.0060	0.1881 ± 0.0024	0.1921 ± 0	**0.2022** ± **0.0002**	0.2019 ± 0
CHI↑	1.2999 ± 0.0194	1.3019 ± 0.0184	1.2946 ± 0.0030	0.5185 ± 0.0324	0.9349 ± 0.0478	1.0443 ± 0	1.3255 ± 0.0001	**1.3258** ± **0**
RTI↓	1.2681 ± 0.1337	1.2561 ± 0.0907	1.2762 ± 0.0375	4.3189 ± 0.9265	1.6101 ± 0.6273	1.4968 ± 0	1.1516 ± 0.1166	**1.0309** ± **0**
WGI↑	0.2342 ± 0.0054	0.2358 ± 0.0048	0.2318 ± 0.0001	0.0357 ± 0.0027	0.2125 ± 0.0062	0.1669 ± 0	0.2427 ± 0.0009	**0.2430** ± **0**
DI↑	0.0107 ± 0.0044	0.0117 ± 0.0042	0.0106 ± 0.0032	0.0090 ± 0.0020	0.0076 ± 0.0022	**0.0389** ± **0**	0.0113 ± 0.0028	0.0102 ± 0
BHI↑	0.0580 ± 0.0009	0.0578 ± 0.0012	0.0578 ± 0.0002	**0.0822** ± **0.0012**	0.0501 ± 0.0009	0.0642 ± 0	0.0593 ± 0.0002	0.0593 ± 0
PBMI↑	0.0090 ± 0.0004	0.0107 ± 0.0081	0.0089 ± 0.0003	0.0043 ± 0.0002	**0.0188** ± **0.0005**	0.0077 ± 0	0.0090 ± 0.0001	0.0089 ± 0
XBI↓	0.1239 ± 0.3366	0.0392 ± 0.0421	0.0488 ± 0.0379	0.1039 ± 0.0808	0.2498 ± 0.5735	**0.0022** ± **0**	0.0365 ± 0.0106	0.0409 ± 0
DBI↓	1.7218 ± 0.0499	1.7080 ± 0.0641	1.7378 ± 0.0143	3.3264 ± 0.3030	**1.4843** ± **0.0653**	1.8248 ± 0	1.6479 ± 0.0182	1.6317 ± 0
LSSRI↑	0.2622 ± 0.0151	0.2637 ± 0.0142	0.2582 ± 0.0023	−0.6586 ± 0.0635	−0.0687 ± 0.055	0.0433 ± 0	0.2818 ± 0.0001	**0.2820** ± **0**
TWI↓	2.1879 ± 0.0186	2.1860 ± 0.0176	2.1929 ± 0.0028	3.3150 ± 0.0714	2.6021 ± 0.0687	2.4613 ± 0	2.1637 ± 0.0001	**2.1635** ± **0**
ACC↑	0.1934 ± 0.0107	0.1941 ± 0.0095	0.1899 ± 0.0027	0.1749 ± 0.0031	**0.3106** ± **0.0203**	0.2201 ± 0	0.1990 ± 0.0040	0.2029 ± 0
NMI↑	0.0831 ± 0.0019	0.0838 ± 0.0019	**0.0840** ± **0.0005**	0.0512 ± 0.0039	0.0758 ± 0.0018	0.0695 ± 0	0.0823 ± 0.0009	0.0807 ± 0
ARI↑	0.0316 ± 0.0010	0.0318 ± 0.0011	0.0318 ± 0.0009	0.0169 ± 0.0009	**0.0403** ± **0.0014**	0.0315 ± 0	0.0316 ± 0.0004	0.0313 ± 0
WQ-Red	KM	KM++	PKM	EWPKM	SC	HC	CAPKM++	CAPKM++2.0
WGSS↓	14.6529 ± 0.3805	14.5249 ± 0.2385	14.4337 ± 0	15.1581 ± 0.7027	16.335 ± 0.0273	15.4005 ± 0	14.2157 ± 0.0003	**14.2156** ± **0**
MRI↓	0.6348 ± 0.0083	0.6327 ± 0.0093	0.6387 ± 0	0.6502 ± 0.0286	0.6824 ± 0.0056	0.6776 ± 0	0.6236 ± 0.0002	**0.6235** ± **0**
GPI↓	0.0523 ± 0.0042	0.0519 ± 0.0029	**0.0503** ± **0**	0.0576 ± 0.0053	0.0825 ± 0.0025	0.0666 ± 0	0.0513 ± 0.0001	0.0513 ± 0
BHGI↑	0.6718 ± 0.0187	0.6755 ± 0.0219	0.6595 ± 0	0.6408 ± 0.0648	0.5879 ± 0.0109	0.5812 ± 0	**0.7007** ± **0.0003**	**0.7007** ± **0**
CI↓	0.1411 ± 0.0081	0.1389 ± 0.0098	0.1471 ± 0	0.1536 ± 0.0279	0.1793 ± 0.0054	0.1798 ± 0	**0.1279** ± **0.0001**	**0.1279** ± **0**
TI↑	0.3792 ± 0.0180	0.3827 ± 0.0210	0.3585 ± 0	0.3665 ± 0.0509	0.3719 ± 0.0062	0.3277 ± 0	**0.4101** ± **0.0002**	**0.4101** ± **0**
DGI↑	0.1728 ± 0.0471	0.1636 ± 0.0365	0.1799 ± 0	0.1913 ± 0.0448	0.0791 ± 0.0606	**0.4332** ± **0**	0.1842 ± 0.0005	0.1844 ± 0
RLI↑	0.2611 ± 0.0058	0.2651 ± 0.0069	0.2593 ± 0	0.2669 ± 0.0125	0.2567 ± 0.0004	0.2635 ± 0	**0.2763** ± **0.0001**	**0.2763** ± **0**
CHI↑	0.9703 ± 0.0492	0.9869 ± 0.0320	0.9989 ± 0	0.9072 ± 0.0860	0.7663 ± 0.0029	0.8734 ± 0	1.0296 ± 0	**1.0296** ± **0**
RTI↓	1.0282 ± 0.2261	0.9577 ± 0.2353	1.2852 ± 0	0.9814 ± 0.3675	1.4410 ± 0.0228	1.0334 ± 0	0.6869 ± 0.0023	**0.6858** ± **0**
WGI↑	0.3014 ± 0.0160	0.3082 ± 0.0186	0.2906 ± 0	0.2708 ± 0.0544	0.2687 ± 0.0025	0.2601 ± 0	**0.3369** ± **0.0001**	**0.3369** ± **0**
DI↑	0.0286 ± 0.0098	0.0290 ± 0.0057	0.0269 ± 0	0.0331 ± 0.0091	0.0129 ± 0.0099	**0.0688** ± **0**	0.0344 ± 0.0012	0.0339 ± 0
BHI↑	0.1110 ± 0.0113	0.1140 ± 0.0113	0.1028 ± 0	**0.1153** ± **0.001**	0.0948 ± 0.0011	0.1068 ± 0	0.1137 ± 0.0001	0.1137 ± 0
PBMI↑	0.0390 ± 0.0057	0.0406 ± 0.0053	0.0400 ± 0	0.0459 ± 0.0076	0.0451 ± 0.0004	0.0479 ± 0	0.0509 ± 0.0005	**0.0511** ± **0**
XBI↓	0.0356 ± 0.0176	0.0334 ± 0.0147	0.0300 ± 0	0.0319 ± 0.0203	4.3277 ± 5.3283	**0.0061** ± **0**	0.0229 ± 0	0.0229 ± 0
DBI↓	1.6499 ± 0.1152	1.5712 ± 0.1175	1.6149 ± 0	1.6738 ± 0.2895	1.4134 ± 0.0061	1.5440 ± 0	1.3924 ± 0.0010	**1.3920** ± **0**
LSSRI↑	−0.0315 ± 0.0529	−0.0137 ± 0.0331	−0.0011 ± 0	−0.1019 ± 0.0979	−0.2662 ± 0.0038	−0.1353 ± 0	**0.0292** ± **0**	**0.0292** ± **0**
TWI↓	2.4421 ± 0.0634	2.4208 ± 0.0398	2.4056 ± 0	2.5264 ± 0.1171	2.7225 ± 0.0045	2.5668 ± 0	2.3693 ± 0.0001	**2.3693** ± **0**
ACC↑	0.3019 ± 0.0221	0.3048 ± 0.0199	0.2714 ± 0	0.2977 ± 0.0298	**0.3923** ± **0.0034**	0.3383 ± 0	0.3279 ± 0.0006	0.3277 ± 0
NMI↑	0.0998 ± 0.0050	0.1012 ± 0.0047	0.0972 ± 0	0.0823 ± 0.0177	0.0866 ± 0.0045	**0.1207** ± **0**	0.1051 ± 0.0002	0.1052 ± 0
ARI↑	0.0659 ± 0.0116	0.0669 ± 0.0109	0.0553 ± 0	0.0538 ± 0.0170	0.0637 ± 0.0022	**0.0813** ± **0**	0.0763 ± 0.0002	0.0764 ± 0

Table 3
The mean values and standard deviations of internal and external cluster validity indices resulting from CAPKM++2.0, CAPKM++, and six baselines on Banana and Phoneme, where $N = 2$ and $M = 5$ in CAPKM++ and CAPKM++2.0.

Banana	KM	KM++	PKM	EWPKM	SC	HC	CAPKM++	CAPKM++2.0
WGSS↓	91.7332 ± 0.0006	91.7334 ± 0.0006	91.7346 ± 0	100.3771 ± 4.9593	91.9065 ± 0.0010	102.5170 ± 0	91.7327 ± 0.0001	**91.7326** ± **0**
MRI↓	**0.5864** ± **0**	**0.5864** ± **0**	0.5865 ± 0	0.6252 ± 0.0259	0.5880 ± 0	0.6519 ± 0	**0.5864** ± **0**	**0.5864** ± **0**
GPI↓	**0.0936** ± **0**	**0.0936** ± **0**	**0.0936** ± **0**	0.1114 ± 0.0127	0.0945 ± 0	0.1236 ± 0	**0.0936** ± **0**	**0.0936** ± **0**
BHGI↑	**0.6257** ± **0**	**0.6257** ± **0**	**0.6257** ± **0**	0.5542 ± 0.0506	0.6221 ± 0	0.5049 ± 0	**0.6257** ± **0**	**0.6257** ± **0**
CI↓	**0.1756** ± **0**	**0.1756** ± **0**	0.1757 ± 0	0.2128 ± 0.0243	0.1773 ± 0	0.2370 ± 0	**0.1756** ± **0**	**0.1756** ± **0**
TI↑	**0.4425** ± **0**	**0.4425** ± **0**	0.4424 ± 0	0.3919 ± 0.0358	0.4399 ± 0	0.3568 ± 0	0.4424 ± 0	0.4424 ± 0
DGI↑	0.0145 ± 0.0046	0.0141 ± 0.0048	0.0083 ± 0	0.0207 ± 0.0149	0.0167 ± 0	**0.0465** ± **0**	0.0177 ± 0	0.0178 ± 0
RLI↑	0.4597 ± 0.0002	0.4597 ± 0.0002	0.4594 ± 0	0.4327 ± 0.0065	0.4592 ± 0	0.4205 ± 0	**0.4598** ± **0**	**0.4598** ± **0**
CHI↑	0.7572 ± 0	0.7572 ± 0	0.7572 ± 0	0.6096 ± 0.0788	0.7539 ± 0	0.5724 ± 0	0.7572 ± 0	**0.7572** ± **0**
RTI↓	0.3292 ± 0.0001	0.3292 ± 0.0001	0.3294 ± 0	0.4164 ± 0.0554	0.3316 ± 0	0.4213 ± 0	0.3292 ± 0	**0.3291** ± **0**
WGI↑	0.5188 ± 0	0.5188 ± 0	0.5188 ± 0	0.4770 ± 0.0256	0.5174 ± 0	0.4242 ± 0	**0.5189** ± **0**	**0.5189** ± **0**
DI↑	0.0018 ± 0.0006	0.0018 ± 0.0006	0.0010 ± 0	0.0026 ± 0.0019	0.0021 ± 0	**0.0064** ± **0**	0.0022 ± 0	0.0022 ± 0
BHI↑	0.0347 ± 0	0.0347 ± 0	0.0347 ± 0	**0.0379** ± **0.0018**	0.0347 ± 0	0.037 ± 0	0.0347 ± 0	0.0347 ± 0
PBMI↑	**0.0496** ± **0**	**0.0496** ± **0**	**0.0496** ± **0**	0.0397 ± 0.0055	0.0493 ± 0	0.0387 ± 0	**0.0496** ± **0**	**0.0496** ± **0**
XBI↓	3.3170 ± 2.5539	3.5630 ± 2.6364	6.7092 ± 0	10.4206 ± 11.3646	1.6401 ± 0	**0.1901** ± **0**	1.4904 ± 0	1.4904 ± 0
DBI↓	1.0355 ± 0.0001	1.0355 ± 0.0001	1.0356 ± 0	1.1647 ± 0.0816	1.0378 ± 0	1.1462 ± 0	1.0355 ± 0	**1.0354** ± **0**
LSSRI↑	−**0.2781** ± **0**	−**0.2781** ± **0**	−**0.2781** ± **0**	−0.5031 ± 0.1316	−0.2825 ± 0	−0.558 ± 0	−**0.2781** ± **0**	−**0.2781** ± **0**
TWI↓	45.8666 ± 0.0003	45.8667 ± 0.0003	45.8673 ± 0	50.1885 ± 2.4797	45.9532 ± 0.0005	51.2585 ± 0	45.8664 ± 0	**45.8663** ± **0**
ACC↑	0.5829 ± 0.0015	0.5832 ± 0.0015	**0.5843** ± **0**	0.5434 ± 0.0338	0.5707 ± 0.0001	0.5006 ± 0	0.5821 ± 0.0002	0.5825 ± 0
NMI↑	0.0217 ± 0.0007	0.0219 ± 0.0007	**0.0223** ± **0**	0.0093 ± 0.0084	0.0150 ± 0	0.0003 ± 0	0.0213 ± 0.0001	0.0215 ± 0
ARI↑	0.0273 ± 0.0010	0.0275 ± 0.0010	**0.0282** ± **0**	0.0117 ± 0.0108	0.0198 ± 0	−0.0006 ± 0	0.0267 ± 0.0001	0.0270 ± 0
Phoneme	KM	KM++	PKM	EWPKM	SC	HC	CAPKM++	CAPKM++2.0
WGSS↓	107.0775 ± 1.0059	106.5772 ± 0.7331	106.2767 ± 0	120.5846 ± 2.1418	111.6192 ± 0	107.4892 ± 0	**106.2766** ± **0**	**106.2766** ± **0**
MRI↓	0.7504 ± 0.0088	0.7460 ± 0.0065	0.7434 ± 0	0.8544 ± 0.0161	0.7788 ± 0	0.7522 ± 0	**0.7433** ± **0**	0.7434 ± 0
GPI↓	0.1330 ± 0.0048	0.1307 ± 0.0035	0.1293 ± 0	0.1719 ± 0.0217	0.1519 ± 0	0.1330 ± 0	**0.1292** ± **0**	**0.1292** ± **0**
BHGI↑	0.4659 ± 0.0209	0.4764 ± 0.0153	0.4826 ± 0	0.2553 ± 0.0353	0.3925 ± 0	0.4656 ± 0	0.4827 ± 0.0001	**0.4827** ± **0**
CI↓	0.2523 ± 0.0085	0.2481 ± 0.0062	**0.2456** ± **0**	0.3584 ± 0.0198	0.2859 ± 0	0.2536 ± 0	**0.2456** ± **0**	**0.2456** ± **0**
TI↑	0.3289 ± 0.0153	0.3366 ± 0.0112	0.3411 ± 0	0.1722 ± 0.0174	0.2775 ± 0	0.3285 ± 0	**0.3412** ± **0**	**0.3412** ± **0**
DGI↑	0.0588 ± 0.0131	0.0517 ± 0.0091	0.0508 ± 0	0.0450 ± 0.0152	0.0658 ± 0	**0.1454** ± **0**	0.0508 ± 0	0.0508 ± 0
RLI↑	0.3337 ± 0.0073	0.3373 ± 0.0053	0.3394 ± 0	0.2919 ± 0.0085	0.3040 ± 0	0.3326 ± 0	**0.3395** ± **0**	**0.3395** ± **0**
CHI↑	0.3264 ± 0.0124	0.3326 ± 0.0090	0.3363 ± 0	0.1781 ± 0.0205	0.2723 ± 0	0.3212 ± 0	0.3363 ± 0	**0.3363** ± **0**
RTI↓	0.7267 ± 0.0054	0.7242 ± 0.0041	**0.7225** ± **0**	1.0678 ± 0.1711	0.9070 ± 0	0.7272 ± 0	0.7232 ± 0.0003	0.7227 ± 0
WGI↑	0.3774 ± 0.0028	0.3788 ± 0.0021	**0.3797** ± **0**	0.3025 ± 0.0379	0.3089 ± 0	0.3775 ± 0	0.3794 ± 0.0001	0.3796 ± 0
DI↑	0.0112 ± 0.0032	0.0095 ± 0.0022	0.0091 ± 0	0.0086 ± 0.0027	0.0137 ± 0	**0.0262** ± **0**	0.0091 ± 0	0.0091 ± 0
BHI↑	0.0959 ± 0.0005	0.0961 ± 0.0003	0.0962 ± 0	0.0986 ± 0.0081	**0.1012** ± **0**	0.0958 ± 0	0.0963 ± 0	0.0962 ± 0
PBMI↑	0.0467 ± 0.0006	0.047 ± 0.0005	**0.0472** ± **0**	0.0326 ± 0.0065	0.0368 ± 0	0.047 ± 0	**0.0472** ± **0**	**0.0472** ± **0**
XBI↓	0.1279 ± 0.0525	0.1548 ± 0.0434	0.1494 ± 0	0.2285 ± 0.1015	0.0868 ± 0	**0.0182** ± **0**	0.1494 ± 0	0.1494 ± 0
DBI↓	1.5734 ± 0.0014	1.5733 ± 0.0018	1.5729 ± 0	1.8153 ± 0.2299	1.7868 ± 0	**1.5616** ± **0**	1.5741 ± 0.0005	1.5734 ± 0
LSSRI↑	−1.1202 ± 0.0383	−1.1012 ± 0.0279	−**1.0897** ± **0**	−1.7324 ± 0.124	−1.3007 ± 0	−1.1356 ± 0	−**1.0897** ± **0**	−**1.0897** ± **0**
TWI↓	53.5387 ± 0.5029	53.2886 ± 0.3666	53.1384 ± 0	60.2923 ± 1.0709	55.8096 ± 0	53.7446 ± 0	**53.1383** ± **0**	**53.1383** ± **0**
ACC↑	0.7044 ± 0.0490	0.6802 ± 0.0356	0.6656 ± 0	0.5901 ± 0.0841	**0.7063** ± **0**	0.6417 ± 0	0.6661 ± 0.0004	0.6652 ± 0
NMI↑	0.1761 ± 0.0055	0.1789 ± 0.0041	0.1807 ± 0	0.0725 ± 0.0194	0.1311 ± 0	**0.1911** ± **0**	0.1798 ± 0.0001	0.1799 ± 0
ARI↑	**0.1692** ± **0.0806**	0.1293 ± 0.0586	0.1053 ± 0	0.0197 ± 0.0872	0.1685 ± 0	0.0698 ± 0	0.1061 ± 0.0006	0.1048 ± 0

CRediT authorship contribution statement

Hongzong Li: Data curation, Software, Investigation, Validation, Writing – original draft. **Jun Wang:** Conceptualization, Methodology, Writing – review & editing, Funding acquisition, Resources, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data are publicly available.

References

[1] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., USA, 1988.
[2] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 15 (11) (1993) 1101–1113.
[3] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[4] M.W. Berry, M. Castellanos, Survey of Text Mining II: Clustering, Classification, and Retrieval, second ed., Springer, 2007.
[5] B. Abu-Jamous, R. Fa, A.K. Nandi, Integrative Cluster Analysis in Bioinformatics, John Wiley & Sons, 2015.
[6] P. Macnaughton-Smith, W. Williams, M. Dale, L. Mockett, Dissimilarity analysis: a new technique of hierarchical sub-division, Nature 202 (4936) (1964) 1034–1035.
[7] S.C. Johnson, Hierarchical clustering schemes, Psychometrika 32 (3) (1967) 241–254.
[8] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (2) (1982) 129–137.
[9] J. Xu, K. Lange, Power k-means clustering, in: International Conference on Machine Learning, PMLR, 2019, pp. 6921–6931.
[10] H. Li, J. Wang, Collaborative annealing power k-means++ clustering, Knowl.-Based Syst. 255 (2022) 109593.
[11] J. Wang, A linear assignment clustering algorithm based on the least similar cluster representatives, IEEE Trans. Syst. Man Cybern. A 29 (1) (1999) 100–104.
[12] B. Zhang, Generalized k-harmonic means–dynamic weighting of data in unsupervised learning, in: Proceedings of the 2001 SIAM International Conference on Data Mining, SIAM, 2001, pp. 1–13.
[13] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: T. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, 2001, URL https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.
[14] Q. Wang, Z. Qin, F. Nie, X. Li, Spectral embedded adaptive neighbors clustering, IEEE Trans. Neural Netw. Learn. Syst. 30 (4) (2018) 1265–1271.
[15] J. Bilmes, A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, Tech. Rep. TR-97-021, International Computer Science Institute, 1997.

[16] W. Chen, X. Wang, Z. Cai, C. Liu, Y. Zhu, W. Lin, DP-GMM clustering-based ensemble learning prediction methodology for dam deformation considering spatiotemporal differentiation, Knowl.-Based Syst. 222 (2021) 106964, http://dx.doi.org/10.1016/j.knosys.2021.106964, URL https://www.sciencedirect.com/science/article/pii/S0950705121002276.

[17] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619.

[18] W. Guo, W. Wang, S. Zhao, Y. Niu, Z. Zhang, X. Liu, Density Peak Clustering with connectivity estimation, Knowl.-Based Syst. 243 (2022) 108501, http://dx.doi.org/10.1016/j.knosys.2022.108501, URL https://www.sciencedirect.com/science/article/pii/S0950705122002155.

[19] T. Zhang, P. Ji, M. Harandi, W. Huang, H. Li, Neural collaborative subspace clustering, in: International Conference on Machine Learning, PMLR, 2019, pp. 7384–7393.

[20] Q. Zheng, J. Zhu, Z. Tian, Z. Li, S. Pang, X. Jia, Constrained bilinear factorization multi-view subspace clustering, Knowl.-Based Syst. 194 (2020) 105514.

[21] L. Wei, F. Zhang, Z. Chen, R. Zhou, C. Zhu, Subspace clustering via adaptive least square regression with smooth affinities, Knowl.-Based Syst. 239 (2022) 107950, http://dx.doi.org/10.1016/j.knosys.2021.107950, URL https://www.sciencedirect.com/science/article/pii/S0950705121010856.

[22] L. Jing, M.K. Ng, J.Z. Huang, An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data, IEEE Trans. Knowl. Data Eng. 19 (8) (2007) 1026–1041.

[23] S. Chakraborty, D. Paul, S. Das, J. Xu, Entropy weighted power k-means clustering, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 691–701.

[24] S. Chakraborty, S. Das, Detecting meaningful clusters from high-dimensional data: A strongly consistent sparse center-based clustering approach, IEEE Trans. Pattern Anal. Mach. Intell. (2022) in press.

[25] R. Krishnapuram, J.M. Keller, A possibilistic approach to clustering, IEEE Trans. Fuzzy Syst. 1 (2) (1993) 98–110.

[26] M.J. Li, M.K. Ng, Y. Cheung, J.Z. Huang, Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters, IEEE Trans. Knowl. Data Eng. 20 (11) (2008) 1519–1534.

[27] J. Fan, J. Wang, A two-phase fuzzy clustering algorithm based on neurodynamic optimization with its application for PolSAR image segmentation, IEEE Trans. Fuzzy Syst. 26 (1) (2018) 72–83, http://dx.doi.org/10.1109/TFUZZ.2016.2637373.

[28] Y. Gao, Z. Wang, J. Xie, J. Pan, A new robust fuzzy c-means clustering method based on adaptive elastic distance, Knowl.-Based Syst. 237 (2022) 107769, http://dx.doi.org/10.1016/j.knosys.2021.107769, URL https://www.sciencedirect.com/science/article/pii/S0950705121009898.

[29] P. Zhou, J. Chen, M. Fan, L. Du, Y.-D. Shen, X. Li, Unsupervised feature selection for balanced clustering, Knowl.-Based Syst. 193 (2020) 105417.

[30] L.E. Caraballo, J.-M. Díaz-Báñez, N. Kroher, A polynomial algorithm for balanced clustering via graph partitioning, European J. Oper. Res. 289 (2) (2021) 456–469.

[31] X. Dai, J. Wang, W. Zhang, Balanced clustering based on collaborative neurodynamic optimization, Knowl.-Based Syst. 250 (2022) 109026, http://dx.doi.org/10.1016/j.knosys.2022.109026.

[32] F. Mai, M.J. Fry, J.W. Ohlmann, Model-based capacitated clustering with posterior regularization, European J. Oper. Res. 271 (2) (2018) 594–605.

[33] H. Li, J. Wang, Capacitated clustering via majorization-minimization and collaborative neurodynamic optimization, IEEE Trans. Neural Netw. Learn. Syst. 34 (2023) in press.

[34] Y. Jia, J. Hou, S. Kwong, Constrained clustering with dissimilarity propagation-guided graph-Laplacian PCA, IEEE Trans. Neural Netw. Learn. Syst. 32 (9) (2020) 3985–3997.

[35] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang, Y. Yang, Rank-constrained spectral clustering with flexible embedding, IEEE Trans. Neural Netw. Learn. Syst. 29 (12) (2018) 6073–6082.

[36] P.S. Bradley, U.M. Fayyad, Refining initial points for k-means clustering, in: International Conference on Machine Learning, Vol. 98, PMLR, 1998, pp. 91–99.

[37] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Tech. Rep., Stanford University, 2006.

[38] O. Bachem, M. Lucic, H. Hassani, A. Krause, Fast and provably good seedings for k-means, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016, URL https://proceedings.neurips.cc/paper/2016/file/d67d8ab4f4c10bf22aa353e27879133c-Paper.pdf.

[39] H. Zha, X. He, C. Ding, M. Gu, H. Simon, Spectral relaxation for k-means clustering, in: T. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, 2001, URL https://proceedings.neurips.cc/paper/2001/file/d5c186983b52c4551ee00f72316c6eaa-Paper.pdf.

[40] Z. Güngör, A. Ünler, K-harmonic means data clustering with simulated annealing heuristic, Appl. Math. Comput. 184 (2) (2007) 199–209.

[41] F. Yang, T. Sun, C. Zhang, An efficient hybrid data clustering method based on K-harmonic means and Particle Swarm Optimization, Expert Syst. Appl. 36 (6) (2009) 9847–9852.

[42] D.W. Cantrell, E.W. Weisstein, Power Mean, MathWorld–A Wolfram Web Resource, 2015, URL https://mathworld.wolfram.com/PowerMean.html.

[43] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[44] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Comput. Surv. 50 (6) (2018) 94.

[45] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2011).