Approximate *k* values using Repulsive Force without Domain Knowledge in *k*-means

Jung-Jae Kim[1], Minwoo Ryu[2], and Si-Ho Cha[3*]
[1] Process and Engineering Research Lab. Control and Instrumentation Research Group, POSCO, Pohang, Korea
[e-mail: jungjae.kim@posco.com]
[2] Service Laboratoire Institute of Convergence Technology, KT R&D Center, Seoul, Korea
[e-mail: mw.ryu@kt.com]
[3] Departement of Multimedia Science, Chungwoon University, Incheon, Korea
[e-mail: shcha@chungwoon.ac.kr]
*Corresponding author: Si-Ho Cha

Received July 30, 2019; revised December 15, 2019; accepted January 14, 2020; published March 31, 2020

Abstract

The *k*-means algorithm is widely used in academia and industry due to easy and simple implementation, enabling fast learning for complex datasets. However, *k*-means struggles to classify datasets without prior knowledge of specific domains. We proposed the repulsive *k*-means (RK-means) algorithm in a previous study to improve the *k*-means algorithm, using the repulsive force concept, which allows deleting unnecessary cluster centroids. Accordingly, the RK-means enables to classifying of a dataset without domain knowledge. However, three main problems remain. The RK-means algorithm includes a cluster repulsive force offset, for clusters confined in other clusters, which can cause cluster locking; we were unable to prove RK-means provided optimal convergence in the previous study; and RK-means shown better performance only normalize term and weight. Therefore, this paper proposes the advanced RK-means (ARK-means) algorithm to resolve the RK-means problems. We establish an initialization strategy for deploying cluster centroids and define a metric for the ARK-means algorithm. Finally, we redefine the mass and normalize terms to close to the general dataset. We show ARK-means feasibility experimentally using blob and iris datasets. Experiment results verify the proposed ARK-means algorithm provides better performance than *k*-means, *k'*-means, and RK-means.

Keywords: Clustering, *k*-means algorithm, machine learning, repulsive force

This study was carried out by supporting the research fund for academic research at Chungwoon University in 2019.

http://doi.org/10.3837/tiis.2020.03.004 **ISSN : 1976-7277**

1. Introduction

The *k*-means algorithm is widely used in academia and industry due to easy and simple implementation, enabling fast learning for complex datasets [1-3]. *K*-means have been employed in many research areas, including big data, machine learning, and data mining [4-6]. However, *k*-means converges to numerous local minima under iterative clustering and has a high dependency upon the initial *k* cluster centers. The initial cluster centroids are usually selected randomly, but to ensure results quality, they should be selected using domain knowledge [7-9]. Unfortunately, practical datasets are often created from diverse domain areas, including smart cities, healthcare, and IoT, etc., forming very large datasets. Thus, it is difficult to select appropriate cluster values that reflect domain knowledge. Therefore, we propose a method to drive approximate *k* value without domain knowledge.

In a previous study, we proposed the repulsive *k*-means (RK-means) algorithm [10] to resolve this problem. We assumed that the cluster centroids had repulsive forces between them, where each cluster had a mass proportion to the sum of errors between data and centroids. The key concept for RK-means was that clusters with larger mass represented a single data group better. Consequently, cluster centroids with larger mass would tend to maintain their current position, whereas cluster centroid with smaller mass would tend to search for positions to acquire larger mass. Thus, smaller mass cluster centroids were "pushed" by larger centroids, and hence the concept of a repulsive force, until these smaller clusters became amalgamated within larger mass clusters or empty. Empty cluster centroids were then deleted. The RK-means algorithm iterated this process and could accurately classify a dataset without requiring specific domain knowledge.

However, although the RK-means algorithm resolved the *k*-means algorithm problem, three problems remained.

- The RK-means algorithm includes an offset to the cluster repulsive force where a cluster is confined within other clusters, which can cause cluster locking.
- We were unable to prove RK-means generated optimal convergence as optimal in the previous study.
- RK-means showed better performance only normalize terms and weight.

Therefore, we propose an advanced RK-means (ARK-means) algorithm to resolve these RK-means problems, establishing an initial strategy for deploying cluster centroids to resolve cluster locking. We also prove ARK-means convergence to be locally optimal, and redefine cluster mass and normalize terms to close to a general dataset.

We show ARK-means algorithm feasibility experimentally compared with RK-means and *k'*-means algorithms using blob and iris datasets. Experimental results verify that the proposed ARK-means algorithm provides more accurate performance than the other algorithms.

The rest of this paper is organized as follows. Section 2 discusses related work, including the methods to find *k* without domain knowledge, and briefly introduces RK-means. Section 3 details the proposed ARK-means algorithm, and Section 4 presents the experimental results and subsequent performance evaluation. Section 5 summarizes and concludes the paper with remarks on future study directions.

2. Related Work

2.1 Selecting *k* without domain knowledge

The simplest approach to select appropriate *k* without domain knowledge is to use heuristic methods, i.e., gradually increase the initial cluster number. However, this approach rapidly becomes prohibitively computationally and tie expensive as dataset size increases [11,12]. Consequently, many previous studies have investigated how clustering could be accomplished with high accuracy without domain knowledge based on the finite mixture model [13-16].

Cheung et al. proposed rival penalized competitive learning (RPCL) based on competitive learning [17] to perform data clustering without knowing the exact number of clusters. They considered two centroids, winner and rival, the winner was the closest centroid and the rival was the second closest centroid for each datapoint. The rival was then reduced using the delearning rate factor to select the appropriate centroid. Although RPCL could perform data clustering without knowing the initial cluster number, the performance was not only sensitive to the preselected rival delearning rate but also could not guarantee convergence as optimal.

Ma et al. [18] and Xie et al. [19] proposed rival penalized controlled competitive learning (RPCCL) and distance sensitive rival penalized competitive learning (DSRPCL), respectively, to improve the RPCL algorithm. RPCCL used a weight calculated from the distance between clusters to resolve the RPCL rival delearning rate, and DSRPCL was able to guarantee convergence as optimal using a cost function generalized from RPCL. These proposed approaches achieved better performance than previous algorithms, and hence have been used in diverse research areas [20-23].

Žalik et al. proposed the *k'*-means algorithm to improve the *k*-means algorithm [24]. Unlike existing the *k*-means algorithm, the *k*'-means algorithm composed of two main phases. In the first phase, the *k*'-means algorithm performs initial clustering and then, it executes preprocessing for assigning seed points into each cluster. Finally, the cost function assigned seed point is adjusted as a minimum. In this phase, the cluster with more data was selected as the winner, and clusters with fewer data became a centroid for an empty cluster, which were subsequently excluded from being winner candidates. Thus, *k'*-means selected the initial number of centroids without domain knowledge.

Arthur *et.al* provided the k-means++ algorithm to resolve the learning results of unstable clustering in exiting the k-means algorithm [25]. To this end, the proposed k-means++ adjust sampling probability distribution. In other words, a point that has a larger distance than pre-selected points is selected as a higher probability in the phase of selecting initial points. Accordingly, initial points are selected as a point that has a larger distance than pre-selected points. However, the k-means++ algorithm cannot resolve the cluster locking problem as mention in Section 2.2. Consequently, we need a strategy of selecting the initial point to resolve clustering locking problem.

2.2 RK-means Algorithm

This section introduces our previously proposed repulsive *k*-means (RK-means) algorithm, which basically follows the classic *k*-means algorithm: (1) *k* initial centroids are randomly chosen from the dataset; (2) data points closest to each centroid create a cluster; (3) each centroid is moved to mean point of cluster; and (4) repeat steps (2) and (3) until the centroids stop moving. Unlike k'-means algorithm, the RK-means algorithm automatically creates empty clusters and then deletes the points using distance function which is consists of three phases.

In these steps, RK-means changes step (3) to divide the dataset into N clusters, $\{\mathbf{x}_i\}_{i=1}^{N}$, where each cluster consists of a d-dimensional vector $\mathbf{x} = \{x_1, x_2, \ldots, x_d\}$. The algorithm uses the same membership function as k-means,

$$\gamma_{nk} = \begin{cases} 1 & where\ k = \underset{k'}{\operatorname{argmin}}(\mathbf{x}_n - \boldsymbol{\mu}_{k'})^2 \\ 0 & others \end{cases}, \tag{1}$$

where $\boldsymbol{\mu}$ is a d-dimensional vector that represents the current centroid position.

However, the centroid position is defined by the sum of mean positions of data points included in a cluster,

$$\boldsymbol{\mu}_{\mathrm{k}} = \frac{\sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n}{\sum_{n=1}^{N} \gamma_{nk}} + D_k, \tag{2}$$

where the D_k vector includes distance and direction of the kth repulsive force from another centroid,

$$D_k = \sum_{k' \neq k} D_{kk'} = \sum_{k' \neq k} C \frac{1}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k'}\|^2} \cdot \frac{1}{M_k} \cdot \frac{\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k'}}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k'}\|}, \tag{3}$$

where C is a normalizing term; the rightmost term is the direction vector of repulsive force from other centroids; and M_k is mass of k^{th} cluster,

$$M_k = \frac{J_k}{J_{k'}} = \frac{\sum_{n=1}^{N} \gamma_{nk} \|\boldsymbol{\mu}_k - \mathbf{x}_n\|}{\sum_{n=1}^{N} \gamma_{nk'} \|\boldsymbol{\mu}_{k'} - \mathbf{x}_n\|}, \tag{4}$$

where J is the sum of errors for the cluster.

Since RK-means cluster mass is a ratio relative to other clusters, we need to normalize the physical quantity. The sum of errors overall clusters, $\sum_k J_k$, is an appropriate normalization, but we use the reciprocal of $\sum_k J_k$ because distance moved by an object is inversely proportional to the mass, i.e.,

$$C = \frac{1}{\sum_k J_k}. \tag{5}$$

Eq. (2) allows a centroid with the largest mass to completely capture a data group and Eq. (5) ensures the repulsive force is normalized as it changes in every step.

Although the RK-means algorithm improves k-means using repulsive force, three issues remain, as discussed in Section 1. Therefore, we proposed ARK-means, based on the existing RK-means algorithm to address cluster locking, proving convergence, and redefine cluster mass and normalization.

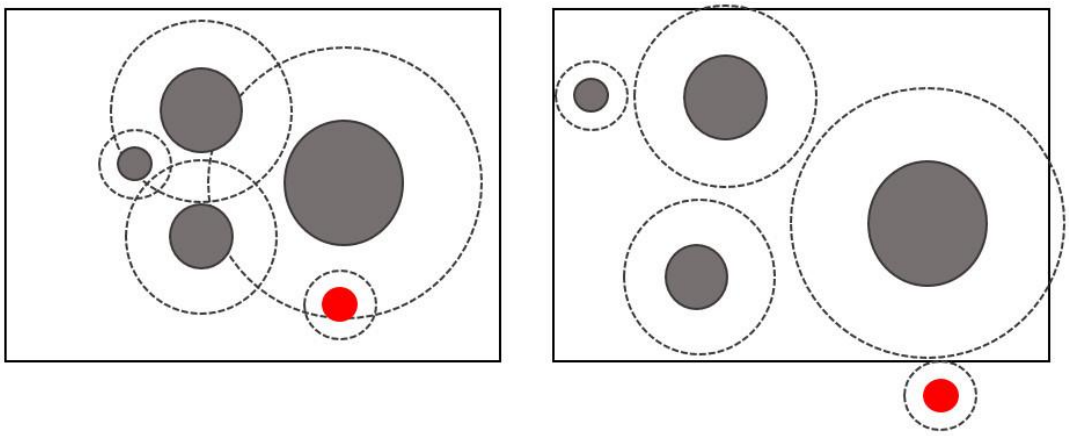
3. Advanced Repulsive Force *k*-means Algorithm

3.1 Motivation

This section discusses the motivation to apply the repulsive force concept to clustering problems.

Suppose we place several magnets on a table, as shown in **Fig. 1**, where the outer rectangle represents the table, circles represent magnet position, central diameter represents their mass, and the dotted circles represent repulsive force. We assume all magnets have the same characteristics aside from their mass, i.e., density and magnetic flux density, and the magnetic field strength is proportional to magnet size (mass).

Fig. 1-(b) shows the magnets moved to non-overlapping field positions after some time, pushed by their repulsive force (we do not consider inertia, gravitation or friction). In particular, the smallest mass magnet is pushed furthest from its initial position.



(a) Initial magnet deployment on a table

(b) Magnet position changes due to their repulsive forces after some time

Fig. 1. Magnet position changes due to mutual repulsive forces

The RK-means algorithm applies this of the repulsive force concept to clustering. Each cluster is considered as a magnet in vector space, where datasets are divided into several clusters. Consequently, cluster centroids are pushed between clusters, and each centroid creates a new cluster in the new position or is pushed out another position. Finally, centroids with smaller mass amalgamate with a data group of clusters with larger mass or become empty, and empty clusters are deleted. Thus, we can classify a dataset without an initial k derived from domain knowledge.

3.2 Initialization

Fig. 2 demonstrates the RK-means cluster locking problem. Since the repulsive force is represented as a vector the number of centroids depends on their initial position. RK-means selects initial centroid positions randomly. Hence, a cluster with small mass could be surrounded by larger mass clusters (**Fig. 2**), with the repulsive force from any larger cluster offset by force from other clusters. Consequently, after several time steps, the smaller cluster remains in the vector space rather than becoming empty or converging another data group. This creates a poor cluster and excludes the cluster members from the opportunity to explore.

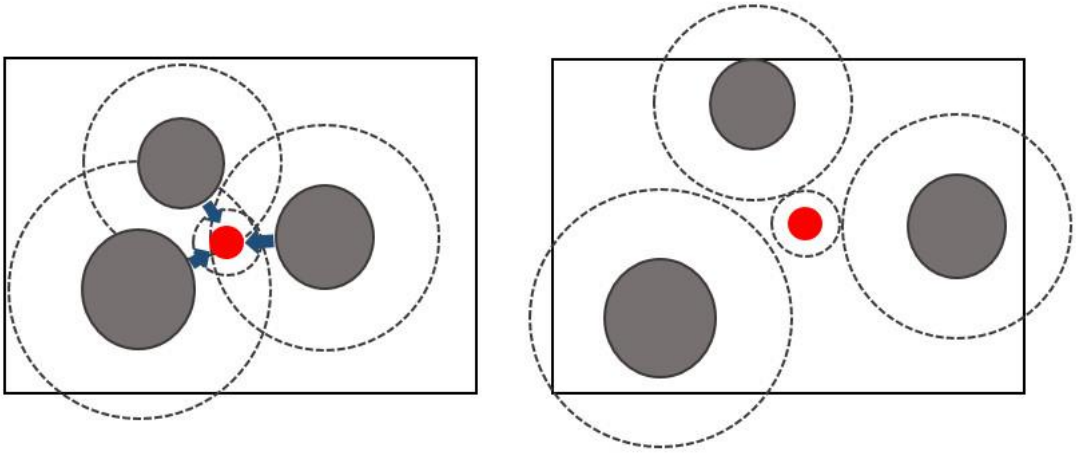


Fig. 2. RK-means cluster locking problem

Therefore, the proposed ARK-means algorithm randomly selects initial cluster values within a sufficiently small space (initial space), defined from the overall mean data position. Thus, no poor clusters can be created except for the special case where the data is perfectly symmetrically distributed from the center.

To obtain the initial space, we first obtain the overall mean data position,

$$\mathbf{x}_{means} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \,, \tag{6}$$

where $\mathbf{x}_i = \{x_1, \dots, x_d\}$ for N d-dimensional data points. Then the vector $\{v_i\}_{i=1}^{d}$, which moves a point on $\mathbf{x}_{means}$ to a random position within the initial space boundary, is

$$\{v_i\}_{i=1}^{d} = \begin{cases} v_i = -\varepsilon \; or \; \varepsilon & i = I \\ v_i = \mathbb{R} \in [-\varepsilon, \varepsilon] & otherwise \end{cases}, \tag{7}$$

where $I \in \{1, 2, \dots, d\}$ is randomly selected, and ε is a sufficiently small positive real number ($\varepsilon > 0$). Thus, $\{v_i\}_{i=1}^{d}$ has value $-\varepsilon$ or ε for the I^{th} component, with remaining components being random real numbers between $-\varepsilon$ and ε.

Consequently, the ARK-means initial centroid position is

$$\boldsymbol{\mu}_{init} = \mathbf{x}_{means} + \{v_i\}_{i=1}^{d}. \tag{8}$$

3.3 Advanced Repulsive Force *k*-means Algorithm

we modify the RK-means algorithm structure to prove ARK-means convergence. RK-means modified k-means centroid update (Section 2.2), whereas for ARK-means we modify the metric between clusters and data to

$$\mathrm{d}(\mathbf{x}, \boldsymbol{\mu}_k) = \{\mathbf{x} - (\boldsymbol{\mu}_k + D_k)\}^2 \,, \tag{9}$$

where **x** is a data vector and $\boldsymbol{\mu}$ is a centroid vector. Thus, $\mathrm{d}(\mathbf{x}, \boldsymbol{\mu}_k)$ is the square of the distance between the data **x** and the position $\boldsymbol{\mu}_k$ is pushed into due to the repulsive force from other centroids.

To prove ARK-means convergence is optimal, we substitute Eq. (9) into the cluster membership (Eq. (1)) and cost function, and use the k-means centroid update rule (Section 2.2) rather than Eq. (2). Consequently, centroids are moved to positions that minimize the changed cost function, and hence ARK-means converges to the local optimum metric (Eq. (9)) at every step. However, this tuning does not affect algorithm performance.

We also modify D_k elements (Eq. (3)), which could affect algorithm performance. We redefine M_k as the ratio of the error sum for the current and neighboring cluster divided by the root mean error,

$$M_k = \frac{J_k \sqrt{\frac{J_{k'}}{N_{k'}}}}{J_{k'} \sqrt{\frac{J_k}{N_k}}}, \tag{10}$$

where N_k is the number of data points in the k^{th} cluster.

However, mean cluster error has become larger and mean neighboring cluster error smaller since the mass has become smaller. Therefore, cluster centroids can be moved large distances by the repulsive force via reducing the mass of cluster which is likely to include multiple data groups. And also, a cluster which is likely to include multiple data group can be dissembled by the way. Centroids also prefer to move to densely populated points. Therefore, we redefined *C* as

$$C = \sqrt[\alpha]{\frac{1}{\sum_k J_k}}, \tag{11}$$

where $1 \leq \alpha \in \mathbb{z}$, but generally $\alpha = [1, 2]$; to resolve C converging to 0 when cluster error sum is too large, by increasing α.

3.4 Proposed ARK-means Algorithm

Table 1 shows the ARK-means algorithm.

Table 1. ARK-means algorithm

Input:
Initial number of Cluster which is more than K' , K
Data set, $\mathbf{x}_{n=1}^{N}$, parameter, α
Output:
Clustering results, $s_{n=1}^{N}$ number of Cluster, cur_k

$cur\_k = K, s_{n=1}^{N} = 0, E_{total} = 0, D_k = 0, \varepsilon' = small\ positive\ number$
$\mathbf{x}_{means} = means\ of\ all\ Data$
Initialize centroid of clusters $\boldsymbol{\mu}_{k=1}^{K} = \mathbf{x}_{means} + \{v_i\}_{i=1}^{d}$
While true:
$data_{k=1}^{cur\_k} = [], E_{k=1}^{cur\_k} = 0,$

For n=1 to N:

14. $k = \underset{k}{\operatorname{argmin}}(\|\mathbf{x}_n - \boldsymbol{\mu}_k - D_k\|^2)$
15. $data_k.append(\mathbf{x}_n), s_n = k$
16. $E_k \mathrel{+}= \|\mathbf{x}_n - \boldsymbol{\mu}_k\|$
17. **For** k=1 to cur_k
18. **If** $\|data_k\| = 0$:
19. remove $\boldsymbol{\mu}_k$ and component of $\boldsymbol{\mu}_k$
20. $cur_k \mathrel{-}= number\ of\ removed\ \boldsymbol{\mu}\ in\ this\ step$
21. **For** k=1 to cur_k
22. $\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{\|data_k\|} data_{k,n}}{\|data_k\|}$
23. $C = \sqrt[\alpha]{1/\sum_{k=1}^{cur_k} E_k}$
24. $D_k = \sum_{k' \neq k} C \frac{E_{k'}}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k'}\|^3 E_k} \cdot \sqrt{\frac{E_k \|data_{k'}\|}{E_{k'} \|data_k\|}} \cdot \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k'}$
25. **If** $\left|E_{total} - \left\{\sum_{k=1}^{cur_k} E_k\right\}^2\right| < \varepsilon'$:
26. **break**
27. $E_{total} = \left\{\sum_{k=1}^{cur_k} E_k\right\}^2$
28.
29. **end While**
30. **return s**, cur_k

In the ARK-means algorithm, lines 2 and 3 describe input values, and line 5 describes the output value. Line 7 initializes return values and all of the variables used in the algorithm. Lines 8 and 9 apply the initial centroid cluster deployment strategy. Lines 11 to 14 describe the process to assign data to centroids that have the smallest metric (Eq. (9)). Line 15 updates $data_k$ and E_k on the cluster whenever data belongs to a centroid of the cluster. Lines 16 to 19 show the deletion process for empty clusters. The k^{th} centroid component includes a space to store data for k^{th} cluster, such as the number of data points and error sum (line 16). Lines 20 and 21 show the process to obtain the distance each cluster is pushed by repulsive force from all of the neighboring clusters. Lines 22 and 23 show the process to move centroids from their current location to a point with the smallest cost function. Lines 24 to 26 describe the termination condition, used in line 6; when the difference between total error sum of squares for the previous and current iteration is less than some pre-selected moderately small real number.

4. Experimental Clustering Results and Analysis

This section compares the proposed ARK-means algorithm experimentally with RK-means and *k'*-means algorithms using blob and iris datasets provided by scikit-learn [26]. We focus on the known problems discussed in Section: cluster locking, proof of optimal convergence, and redefine mass and normalization to close to the general dataset. Therefore, we performed 2 experiments as follows.

1. We used the blobs dataset to fix the 2-dimensional data positions and then compared ARK-means with RK-means algorithm performance to investigate poor cluster possibility concerning different the initial *k*.
2. We compare ARK-means, RK-means, *k*-means, and *k'*-means algorithms for approximated k without applying domain knowledge using blobs and iris datasets for different termination conditions:

a. iteration maximum number 100,
b. square error difference < 0.01%.

We also used different parameters to select the initial centroids for each algorithm:

- *k*-means uses the *k*-means++ method to stochastically select the centroid,
- *k'*-means uses random value, and
- ARK-means uses fixed value = 0.0025.

The experiments were performed on a PC with Intel Core i-5-35550, 3.30GHz CPU and 4GB DDR3 RAM. The ARK-means algorithm was developed in the Python 3.5.2, and random values were generated by NumPy 1.11.1 with seed = 1.

4.1 Initialization Strategy Experiment

If the blobs dataset receives the number of data *N* and class *S*, and the seed value as input values, it is possible to deploy data generated by *N* 2-dimensional data which comply with normal distribution, divided into *S* circular data group.

Fig. 3 shows the likelihood for clusters using the blob dataset with class $S = 4$, $N = 350$ and initial $k = 5$ to 100. The result shown is the average over 1000 repeats, and poor clusters were defined as clusters with ≤ 20 data points. RK-means poor cluster possibility is strongly sensitive to increasing *k*, frequently defining smaller mass clusters encircled by larger mass neighbors. In contrast, the proposed ARK-means achieved significantly lower poor cluster possibilities, particularly for larger *k*. Thus, the proposed initial dispersion mechanism for ARK-means successfully avoids poor clustering. This is why the ARK-means algorithm fundamentally disperses the initial point for centroids due to the initial strategy proposed in this paper.

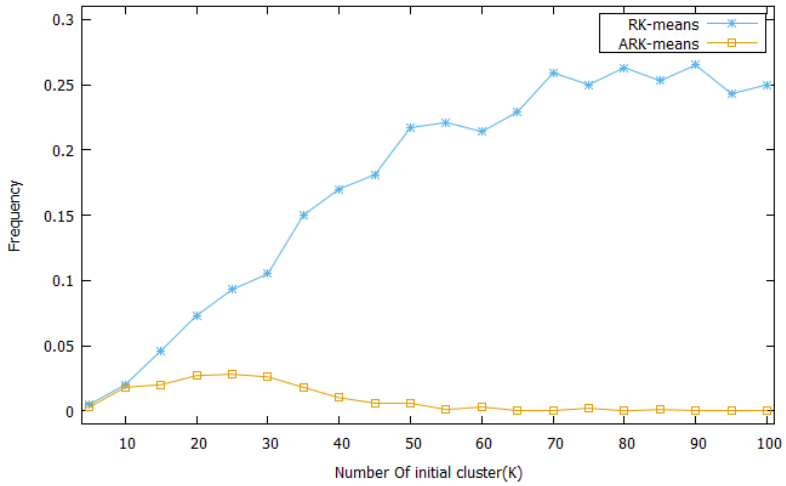


Fig. 3. Possibility of poor clusters concerning the number of initial clusters (*k*) for the indicated algorithms

4.2 Evaluation of the ARK-means Algorithm

We compared ARK-means performance as follows:

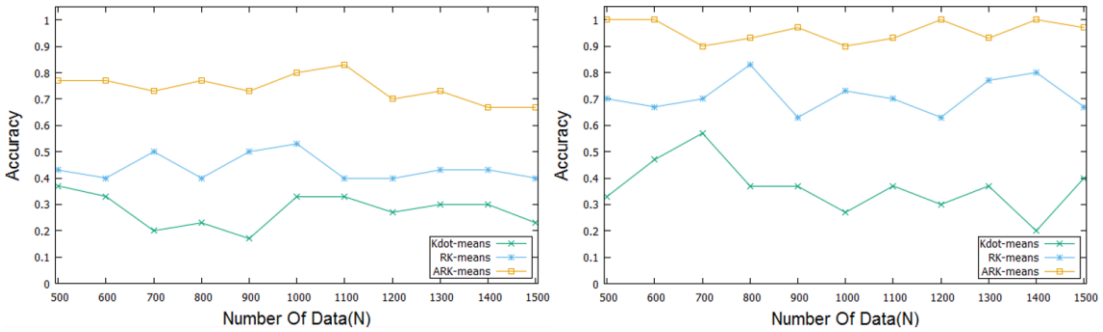
1. accuracy compared with RK-means and k'-means using the blobs dataset with initial k = 5 and 10;
2. accuracy for various combinations of initial k and seed points, S, using the blobs dataset;
3. accuracy and time to complete to initial k compared with RK-means, k-means, and k'-means using the iris dataset.

4.2.1 Performance Comparison 1

We used the blobs dataset with S = 3, initial k = 5 and 10, and N = 500, 600, …, 1500. Experiments were repeated 30 times, and the presented data are the mean overall 30 repeats.

Fig. 4 shows average accuracy concerning N. All algorithms provided broadly constant average performance irrespective of N, which confirms the algorithms all provided good performance. However, k'-means exhibited the lowest performance due to the upper bound problem for k, as discussed in Section 2, and also exhibited stronger sensitivity to N. In contrast, RK-means and ARK-means changed significantly depending on initial k, with RK-means exhibiting significantly lower performance than ARK-means because RK-means strongly depends on initial centroid positions.

The ARK-means algorithm classified all data into three clusters regardless of the N, even when initial k > 3. Thus, the proposed ARK-means resolved both the upper bound and large dataset classification problems without domain knowledge.



(a) Initial k = 5 (b) Initial k = 10

Fig. 4. Algorithm accuracy concerning dataset size for different initial k

4.2.2 Performance Comparison 2

This experiment particularly focused on the proposed ARK-means feasibility. Therefore, we used the blobs dataset with N = 100, S = 3 to 6, and initial k = 10, 12, 14, and 16. The experiments were repeated 30 times, and presented numbers are averaged over the repeats. Table 2 shows mean accuracies for three cases: K' = S, $|K'–S|$ = 1, and $|K'–S|$ > 1, where K' is the final k from ARK-means.

Table 2. Experimental results of ARK-means accuracy using the blobs dataset

Number of initial centroids, k	Number of seed points, S					
	S=3			S=4		
	$K' = S$	$\|K' - S\| = 1$	$\|K' - S\| > 1$	$K' = S$	$\|K' - S\| = 1$	$\|K' - S\| > 1$
10	1.0	0.0	0.0	0.8	0.2	0.0
12	0.97	0.03	0.0	0.77	0.23	0.0

14	0.97	0.03	0.0	0.85	0.12	0.03
16	1.0	0.0	0.0	0.9	0.1	0.0
	***S*=5**			***S*=6**		
	$K' = S$	$\|K' - S\| = 1$	$\|K' - S\| > 1$	$K' = S$	$\|K' - S\| = 1$	$\|K' - S\| > 1$
10	0.53	0.47	0.0	0.07	0.77	0.17
12	0.47	0.53	0.0	0.23	0.73	0.03
14	0.23	0.77	0.0	0.3	0.63	0.07
16	0.37	0.63	0.0	0.17	0.83	0.0

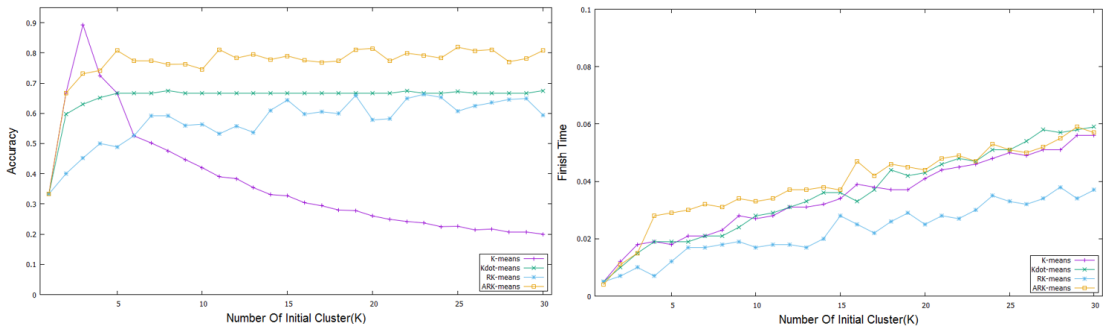
K' = final ARK-means cluster count

4.2.2 Performance Comparison 3

We compared ARK-means, RK-means, *k*-means, and *k'*-means using the iris dataset, comprising 50×3 4-dimensional feature vectors with 3 classes, setting initial k = 1 to 30, as shown in **Fig. 5**.

Fig. 5-(a) shows that *k*-means achieved the lowest accuracy, which also rapidly decreased for $k > 3$, since poor cluster occurrence increased for larger initial $k > S$. RK-means achieved approximately 50% accuracy, confirming performance degradation of the initialization strategy. *k'*-means achieved approximately 60% accuracy, but this algorithm only found 2 classes, rather than the actual 3. Thus, *k'*-means did classify the data well. ARK-means achieved approximately 75% accuracy due to correcting the cluster locking problem even for large initial *k*. Thus, the proposed initialization strategy for ARK-means resolved cluster locking.

Fig. 5-(b) shows the algorithm time to complete for initial *k*. All algorithms show have similar performance to k-means, although ARK-means is slightly improved since convergence to local optimum is guaranteed and convergence speed is improved by the term obtained the formula of the ARK-means algorithm.



(a) Algorithm accuracy for the number of initial clusters (*k*)

(b) Algorithm tie to completion for the number of initial clusters (*k*)

Fig. 5. Algorithm accuracy and speed using the iris dataset

5. Discussion

This paper explained several current classification problems and showed that the proposed ARK-means algorithm provided appropriate solutions. However, we must consider the remaining issues with the ARK-means algorithm.

Section 4.2 showed that ARK-means accuracy decreased when the number of classes in the dataset increased, and the final number of clusters was smaller than the actual number of classes in most cases. These problems due to overlapping in the limited space, a known characteristic of the blobs dataset. A common way to resolve this problem is to distribute the data in infinite space. However, the blobs dataset circular distribution has high center cluster density, which makes it difficult to sufficiently distinguish the clusters. Therefore, we need to consider how weights and normalization terms could be more precisely defined in the algorithm and add mechanisms to distinguish clusters with overlapping edges in different densities. These problems are not unique to the proposed ARK-means algorithm and remain generally challenging tasks to improve classification performance.

Another problem is to define the exact dataset class from a data group linearly distributed in space. For ARK-means, more data groups linearly distributed in space make it more difficult to find the correct dataset class, i.e., ARK-means struggles to fill the data group by reducing the total cluster mass due to data on the elongated side being a significantly larger distance from the cluster centroid. This leads to misclassifications since ARK-means define repulsive force based on distance. Therefore, we need to consider another criterion to define repulsive forces and hence resolve this problem, e.g. similarity, which will be somewhat challenging.

ARK-means struggles to identify approximate classes for outliers in the dataset, although outliers do not significantly affect selecting the approximate *k*. Outliers may occur in different data groups or completely separate from all data groups. For the former case, ARK-means could deal with outliers as an integrated cluster, and hence this has little effect on the selecting approximate *k*; whereas for the latter case, ARK-means recognizes the outlier as a separate cluster, and we could find a suitable approximate *k* by removing the outlier using filtering based on the number of data points in a cluster. Hence, outliers in the dataset should not affect dataset clustering without domain knowledge, which is the primary ARK-means purpose.

Finally, Although the proposed ARK-means algorithm provides approximate *k* value without domain knowledge, it has weak points in practice. As shown in Fig. 4 and Fig. 5, jitter is generated by random seed when blob data is created by changing all of the distance between data every time. And also, the ARK-means algorithm has performance deviation according to *k* value given as initial value. Consequently, we can know the ARK-means algorithm has a different performance deviation by location of initial points. To resolve this phenomenon, a common way is to define initial *k* value based on several simulations of inputted data in a specific service domain. However, this way has a higher cost and time. Furthermore, we cannot know changing characteristics of data including distance and average. Accordingly, it is also a challenging task.

6. Conclusion

This paper proposed the ARK-means algorithm to resolve three specific significant problems with the RK-means algorithm. We performed three sets of experiments to verify ARK-means feasibility, with ARK-means achieving the highest performance compared with current *k*-means, *k'*-means, and RK-means algorithms. We also discussed remaining practical ARK-means issues and challenging tasks to resolve them. Therefore, we can conclude with confidence that the proposed ARK-means algorithm will improve classification performance for large datasets, selecting initial *k* without domain knowledge. Although we improve classification performance based on proposed our algorithm, there are remain challenge tasks

to apply the method in practice. To this end, future research will include statistics based on selecting an initial point to reduce performance deviation in practice.

References

[1] Y.-M. Cheung, "On rival penalization controlled competitive learning for clustering with automatic cluster number selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1583-1588, 2005. Article (CrossRef Link).

[2] M. E. Celebi, H. A. Kingravi, Patricio A. vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Systems with Applications*, vol 40, no. 1, pp. 200-210, 2013. Article (CrossRef Link).

[3] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881-892, 2002. Article (CrossRef Link).

[4] D. Arthur, S. Vassilvitskii. "k-means++: the advantages of careful seeding," *Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, SODA, SIAM*, pp.1027-1035, 2007.

[5] C. Fang, W. Jin, J. Ma, "k′-Means algorithms for clustering analysis with frequency sensitive discrepancy metics," *Pattern Recognition Letters*, vol. 34, no. 5, pp. 580-586, Apr. 2013. Article (CrossRef Link).

[6] H. Khatter, V. Aggarwal, A. K. Ahlawat, "Performance Analysis of the Competitive Learning Algorithms on Gaussian Data in Automatic Cluster Selection," in *Proc. of IEEE International Conference on Computational Intelligence & Communication Technology (CICT), U.P., India*, 12-13 Feb. 2016. Article (CrossRef Link).

[7] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, "Constrained K-means Clustering with Background Knowledge," in *Proc. of the Eighteenth International Conference on Machine Learning*, pp. 577-584, 2001.

[8] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651-666, June 2010. Article (CrossRef Link).

[9] J. Baek, Y. Kim, B. Chung, C. Yim, "Linear Spectral Clustering with Contrast-limited Adaptive Histogram Equalization for Superpixel Segmentation," *IEIE Transactions on Smart Processing and Computing*, vol. 8, no. 4, pp. 255-264, August, 2019. Article (CrossRef Link).

[10] J. Lei, T. Jiang, K. Wu, H. Du, G. Zhu, Z. Wang, "Robust K-means algorithm with automatically splitting and merging clusters and its applications for surveillance data," *Multimedia Tools and Applications*, vol. 75, no. 19, pp. 12043-12059, Oct. 2016. Article (CrossRef Link).

[11] D. T. Pham, S. S. Dimov, C. D. Nguyen, "Selection of K in K-means clustering," in *Proc. of ImechE, vol. 2019 Part C: Journal of Mechanical Engineering Science*, vol. 219, pp. 103-119, 2005. Article (CrossRef Link).

[12] T. M. Kodinariya, P. R. Makwana, "Review on determining number of Cluster in K-Means Clustering," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, no. 6, pp. 90-95, 2013.

[13] P. S. Bradley, U. M. Fayyad, "Refining Initial Points for K-Means Clustering," in *Proc. of the 15th International Conference on Machine Learning (ICML98)*, San Francisco, USA, pp. 91-99, 1998.

[14] D. Pelleg, A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *Proc. of the 17th International Conference on Machine Learning* (*ICML2000*), pp. 727-734, 2000.

[15] M. K. Ng, J. Z. Huang, L. Jing, "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 19, pp. 1026-1041, 2007. Article (CrossRef Link).

[16] J. Luo, J. Ma, "Image segmentation with the competitive learning-based MS model," in *Proc. of the 2015 IEEE International Conference on Image Processing (ICIP), Quebec, Canada*, 27-30 Sep. 2015. Article (CrossRef Link).

[17] Y.-M. Cheung, "Rival penalization controlled competitive learning for data clustering with unknown cluster number," in *Proc. of the 9th International Conference on Neural Information Process* (*ICONIP '02), Singapore*, 18-22 Nov. 2002. Article (CrossRef Link).
[18] J. Ma, T. Wang, "A cost-function approach to rival penalized competitive learning (RPCL)," *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*), vol. 36, no. 4, pp. 722-737, 2006. Article (CrossRef Link).
[19] H. Xie, X. Luo, C. Wang, S. Liu, X. Xu, X. Tong, "Multispectral remote sensing image segmentation using rival penalized controlled competitive learning and fuzzy entropy," *Soft Computing*, vol. 20, no. 12, pp. 4709-4722, Dec. 2016. Article (CrossRef Link).
[20] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, USA*, 16 Jan. 2008.
[21] J. Qin, W. Fu, H. Gao, W. X. Zheng, "Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 772-783, Mar. 2016. Article (CrossRef Link).
[22] L. Xu, A, Krzyzak, E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Transactions on Neural networks*, vol. 4, no. 4, pp. 636-649, 1993. Article (CrossRef Link).
[23] G.-J. Yu, K.-Y. Yeh, "A K-Means Based Small Cell Deployment Algorithm for Wireless Access Networks," in *Proc. of 2016 International Conference on Networking and Network Applications (NaNA), Hokkaido, Japan*, 23-25 July 2016.
[24] K. R. Žalik, "An efficient k′-means clustering algorithm," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1385-1391, 2008. Article (CrossRef Link).
[25] D. Arthur, S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proc. of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA*, pp. 1027-1035, 2007.
[26] Scikit-learn, "Machine Learning in Python," *Pedregosa et al., JMLR 12*, pp. 2825-2830, 2011.

Jung-Jae Kim received his B.S., M.S., and Ph.D. degrees in Computer Science from Kwangwoon University, Seoul, Korea in 2013, 2015, 2019, respectively. He is now a senior researcher in the Process and Engineering Research Lab., Control and Instrumentation Research Group, POSCO, Pohang, Korea. His research interests include Intelligent Network System, Machine Learning and Deep Learning.

Minwoo Ryu received his B.S. degree in Internet Information Processing from Yeoju Institute of Technology, Korea in 2007, and his M.S. and Ph.D. degrees in Computer Science from Kwangwoon University, Seoul, Korea in 2009 and 2012, respectively. From 2011 to 2016, he worked as a research scientist at Korea Electronics Technology Institute (KETI), Korea. He is now a research scientist in the KT R&D center, Korea Telecom (KT), Korea. His research interests include Internet of Things, Semantics, Cognitive Computing, Intelligence Service and Vehicular Ad Hoc Networks.

Si-Ho Cha received his B.S. degree in Computer Science from Suncheon National University, Suncheon, South Korea in 1995, and his M.S. and Ph.D. degrees in Computer Science from Kwangwoon University, Seoul, South Korea in 1997 and 2004, respectively. From 1997 to 2000, he worked as a senior researcher at Daewoo Telecom R&D Center, South Korea. He is a professor in the Department of Multimedia Science, Chungwoon University, Incheon, South Korea. His research interests include Network Management, Vehicular Ad Hoc Networks, and Machine Learning and Deep Learning.