Article Type: Data-centric Artificial Intelligence

Integrating Curriculum Learning with *k*-Means: A Data-centric Approach to Faster Clustering

Abdul Majeed and Seong Oun Hwang, *Department of Computer Engineering, Gachon University, Korea*

Abstract—k-Means clustering is a very popular method that groups n observations into k clusters based on the nearest mean, whereas each observation serves as the prototype of a cluster. Although k-means yields desirable results in most cases, the computing overhead is very high even for modest-size datasets, which makes it unsuitable for larger datasets. The main reasons for high overheads are extensive proximity analysis and excessive center updates in the clustering process. To lower computing overhead without degrading performance, in this paper, we propose and implement a curriculum learning (CL) integrated k-means clustering method for efficiently clustering observations. In our method, the data to be clustered via k-means is evaluated beforehand and sorted based on complexity using the CL approach. By applying CL, the data portion that is naturally clustered is identified and bypassed by k-means so only some complex portions of the data undergo processing, leading to a significant reduction in computing overhead. Experiments on benchmark datasets prove the efficacy of the proposed concept, and results are significantly better than those from the conventional k-means algorithm.

Index Terms– *k-means clustering, curriculum learning, data complexity, CL-integrated k-means clustering, observations, data-centric approach*

The demand for drawing actionable insights from data is increasing, and companies are striving to explore the latest data mining tools that can extract knowledge from data with the least possible cost. To this end, machine learning (ML) techniques are handy for getting useful knowledge from data for downstream tasks [1]. These ML techniques capture patterns and correlations in the raw data, which can assist in the prediction, classification, and grouping of tasks [2]. The knowledge produced by these ML methods can widely contribute to the medical domain for recommendations, decision-making, event prediction, hyperspectral band selection [3], etc. The COVID-19 pandemic also showed the effectiveness of data mining and ML techniques to effectively combat infections.

Clustering is a widely used approach to enhancing the analysis of big data and enabling knowledge discovery. How to efficiently divide n observations (or records) into k clusters without losing statistical information is a crucial task. The main task of clustering is to group data based on similarity, and it comes under the umbrella of unsupervised ML. The k-means algorithm is a widely used method in the ML community to cluster raw data for deriving meaningful information/knowledge from big data [4]. The k-means algorithm has various applications in diverse domains, such as community clustering, customer segmentation, and medical diagnosis. Although it is a state-of-the-art (SOTA) algorithm, there are three critical problems with the conventional k-means.

- The possibility that some data are naturally clustered is ignored, which can lead to extensive and unnecessary operations during the clustering process. In many cases, some parts of the data can be readily clustered, and extracting them beforehand can save a lot of computing time.
- The entire dataset is given as input without prior classification based on complexity, and all observations are treated as equally complex. Not sorting data based on complexity, and giving each observation equal treatment, may increase the number of iterations and could leave unclustered observations (a.k.a. leftover records/

XXXX-XXX © 2023 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

observations).

- Conventional k-means cannot handle data with uneven distribution, and some complex parts of the data are deleted before the clustering process begins, which may hamper knowledge discovery and data mining results [5].

In the past, some methods improved the technical deficiencies of k-means clustering by using algorithms such as regular-grid [6], p-value-based k-means [7], HCSA-based k-means [8], imbalanced clustering [9], outlier-removal-aided k-means [10], and tri-layer k-means [11]. We affirm the contributions of these methods, however, none have explored data classification based on complexity to reduce the computing overheads of k-means algorithms. Our major contributions are as follows.

- We explore the technical drawbacks of the k-means algorithm in terms of unnecessary computation and poor convergence when dealing with mixed data, and we identify opportunities to amalgamate curriculum learning with k-means to resolve the above-cited problems.
- To the best of our knowledge, we are the first to propose and implement a CL-based k-means algorithm to sort data based on complexity and to keep some parts with high cohesion out of the clustering process to reduce computing overhead.
- By classifying data based on complexity, the performance deficiencies of the k-means algorithm are effectively addressed, and the resulting clustered data are better in most cases than those from the conventional k-means algorithm.
- Our work is expected to open various new research tracks in this line of work (e.g., amalgamating the latest AI techniques with conventional methods) to optimize the existing ones.

We performed exhaustive experiments on five benchmark datasets to prove the efficacy of the newly proposed method and attained satisfactory results. This work has five technical contributions to the category of partitional clustering techniques: (i) it reduces the # of calculations in terms of distance between cluster center and data points, (ii) It reduces the count of cluster centers updates compared to conventional k-means, (iii) It contributes to achieving faster convergence of clustering process by reducing # of iterations, (iv) it processes some parts of data while giving clustering results that are mostly similar (or better) to traditional k-means, and (v) It reduces the leftover records by exploiting the global composition of meta-features in data. Also, the proposed method can be the ideal choice in scenarios when the majority of data is collected from the same individuals or community. The novelty of this work lies in a lightweight clustering approach by exploiting data characteristics/composition rather than model-related adjustments, which is the latest trend in AI developments. Lastly, CL is used in conjunction with conventional k-means for the first time to prevent unnecessary calculations and computations. In the next sections, we describe the traditional and CL-integrated k-means algorithms in detail.

Related work

Many research studies have proposed improvements to the original k-means clustering from the perspective of new distance metrics, convergence speed, proximity computation, clustering effect, etc. In [6], the authors employed a regular grid to cluster high dimensional data by choosing some cells as the cluster center. Later, the grid-weighted iteration strategy was employed to find K clusters in a short time. In [7], the authors investigated the impact of p, where p conditions on all intermediate clustering assignments in k-means. The proposal helps reduce Type I error in k-means, and the value of p can be efficiently computed. In [8], the authors used a meta-heuristic-based method, named Hybrid Capuchin Search Algorithm (HCSA) to address noise sensitivity, initialization sensitivity, and vulnerability to imbalanced sample distributions. In [9], the authors proposed an adaptive version of k-means, namely k-means with Adaptive Cluster Weight (MACW) for imbalanced clustering problem. Later, they developed a re-weighting strategy that assigns smaller weights to major clusters to retain balanced samples in most clusters while being efficient. In [10], the authors employed a modified Tukey rule to remove outliers from the data before the clustering process and developed new distance metrics for assigning each data point to the nearest cluster. The proposed method has improved centroid convergence and clustering accuracy.

A notable development to enhance the computational efficiency of the k-means is the mini-batch k-means[1] that uses random data batches of fixed size rather than loading entire data to memory at once. This method uses data from mini-batches to update the cluster centers and enable faster convergence. However, this approach still processes the entire data without pre-partitioning it which can lead to higher

[1] https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/

distance calculations and computations when data is large. Zhu et al. [12] improved the *k*-means algorithm by replacing the Euclidean distance with the evidence distance to improve the clustering effect and convergence speed. Wei et al. [13] employed the concept of density peaks in *k*-means to improve the # and location of initial cluster centers. However, this algorithm has higher computing costs and time overheads were not analyzed. Kong et al. [14] extended conventional *k*-means to multi-view clustering scenarios. Specifically, the authors improved the clustering results by removing irrelevant information. Zhang et al. [15] improved the clustering process by pruning some redundant distance computations. The three new concepts non-linear embedding, block vector, and segment mean are employed to fasten the clustering process. Recently, some researchers proposed parallel computing to overcome the convergence issues of the clustering process [16]. More detailed insights about *k*-means and its variants can be learned from a recent survey [17]. We affirm the contributions of all prior methods, however, these methods have not explored ways to pre-partition datasets to reduce hefty computations.

Conventional *k*-means algorithm

The *k*-means algorithm has been widely used in many domains owing to its conceptual simplicity as well as its algorithmic soundness. Despite the few shortcomings discussed above, it is still a SOTA algorithm in the ML community. In *k*-means, it is very challenging to find the optimal value of *k* (# of clusters), and therefore, elbow and silhouette methods are widely used to pick a suitable *k*. The workflow of the conventional *k*-means algorithm is outlined below.

1) Acquire data (*D*) and the number of clusters (*k*) as input.
2) Randomly initialize *k* centroids.
3) Assign each observation from *D* to the nearest centroid (i.e., dividing all of *D* into *k* clusters).
4) Re-estimate means of the clusters.
5) Repeat steps 2 and 3 until convergence (no changes in observation membership).
6) Return clusters (intra-cluster similarity $\rightarrow$ high and inter-cluster similarity $\rightarrow$ low).

Aided by this process, any *D* can be converted to *k* clusters, where the sizes of clusters can be either the same or different. The workflow of the *k*-means clustering algorithm is illustrated in Figure 1.

Under conventional *k*-means, the # of iterations can be large, and the resulting clusters can overlap in some cases. To this end, improving *k*-means is imperative in

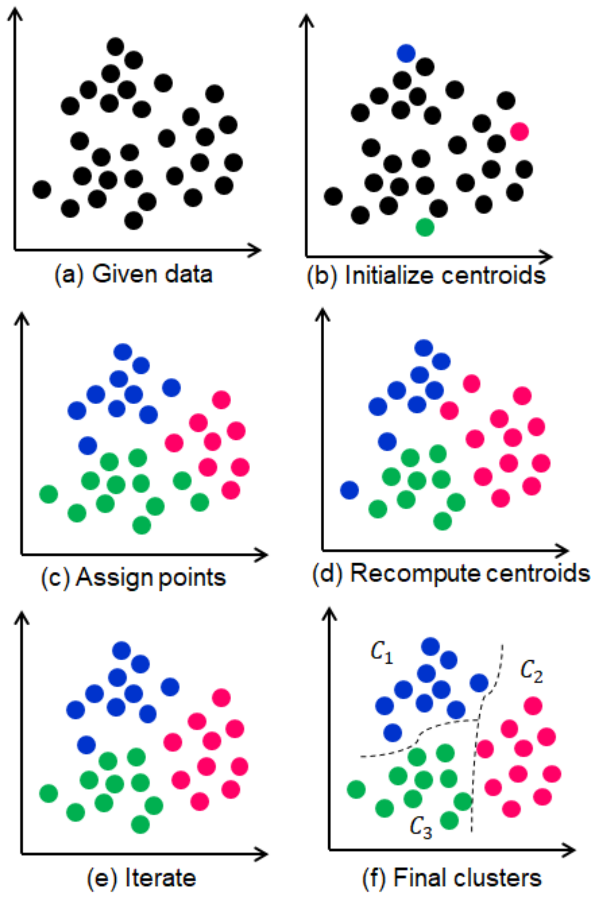

FIGURE 1. Workflow of the conventional *k*-means clustering.

order to yield desirable results in real-life scenarios. In the next section, we present the workflow of the proposed CL-integrated *k*-means algorithm.

CL-integrated *k*-means algorithm

In this section, we provide the workflow and related concepts of the proposed CL-integrated *k*-means algorithm. To prevent unnecessary operations during clustering, we integrate *k*-means with a CL approach [18] because CL can sort data based on complexity, which can be efficiently perceived by the respective model. The CL approach resembles the human learning process[2] (examples should be organized in a meaningful order not randomly presented) and is advantageous in terms of performance and convergence speed in ML. The CL assists in selecting data of easy difficulty first and then gradually increases the complexity/difficulty of

[2]https://hwiyong.tistory.com/327

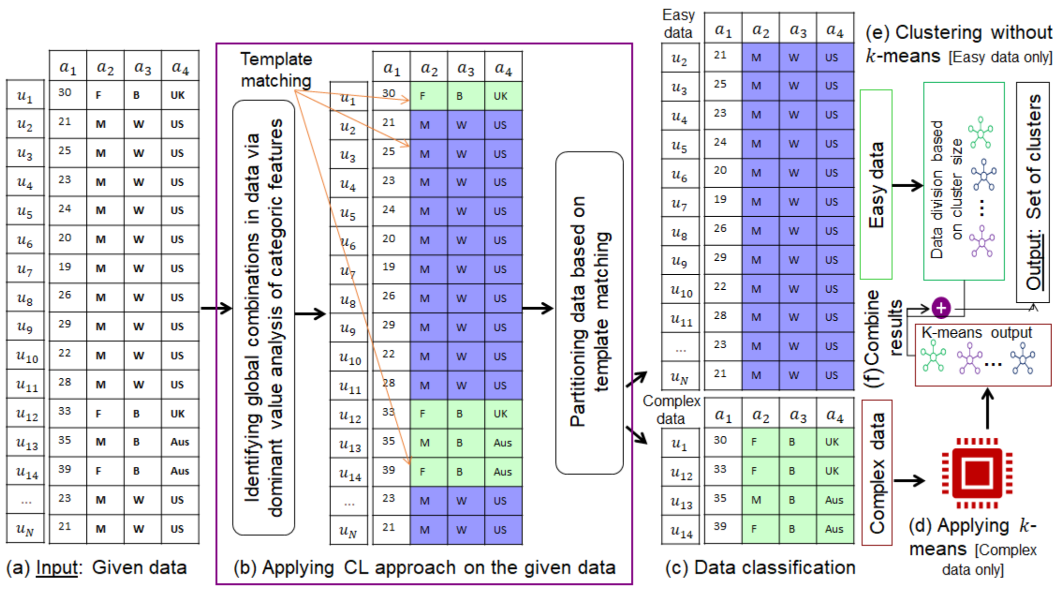
Template matching
Identifying global combinations in data via dominant value analysis of categoric features
Partitioning data based on template matching
Easy data
Complex data
(e) Clustering without k-means [Easy data only]
Data division based on cluster size
Output: Set of clusters
(f) Combine results
K-means output
(d) Applying k-means [Complex data only]
(a) Input: Given data
(b) Applying CL approach on the given data
(c) Data classification

FIGURE 2. Workflow of the proposed CL-integrated k-means clustering method to efficiently cluster the data.

the data. It is worth noting that DL/ML models usually take data in the form of batches, however, the data allocation in each batch is random (e.g., no order or shape is given to data). In contrast, the CL approach gives some meaningful order to the data considering the underlying model to fasten the computation without losing performance. CL has three key conditions [18] concerning data complexity:

$$H(X_t) < H(X_{t+1}), W_t(z) < W_{t+1}(z) \, \forall z \in D, X_t(z) = P(z) \quad (1)$$

In Eq. 1, H, W, and P denote the entropy, weight, and training data distribution, respectively. The above three conditions gradually increase the complexity of data by increasing entropy (H), weights (W), and # of samples (P). Specifically, a few easy samples are chosen first which are further enriched as time passes by to improve the learning ability of the underlying AI model.

CL gives a meaningful representation and order to the data, and therefore, processing can be very fast compared to conventional approaches (i.e., those that do not give shape/order). In the AI discipline, CL is an emerging paradigm that offers promising solutions to train complex neural network models, reducing computations. In the clustering domain, a similar concept is required, because clustering approaches like k-means consume a lot of time extracting commonalities from among observations that could be captured beforehand with CL-like approaches. To the best of our knowledge, we are the first to apply CL in the clustering domain to speed up the process by identifying portions of data that are already clustered. The key idea is to divide the given dataset into two partitions—already clustered, and partially (or not) clustered—and to send only the latter to clustering to speed up the process by reducing similarity computing as well as center updates. In this work, we employ CL as a pre-processing step to k-means clustering to extract data portions that are readily clustered. Specifically, we explore similarities between observations by using the information of dominant values of categorical features to pre-partition the portion of datasets with high cohesion. By doing so the size of the data to be clustered with k-means is reduced, which can subsequently reduce the computation time because proximity analysis (distance computation) and center updates are restricted to some complex parts of data only. Since CL assists in sorting data based on complexity (e.g., from easy to hard), we exploit this property of CL in the context of clustering and obtain promising results. We found that the portion regarded as easy by CL is readily clustered and can be skipped from clustering algorithms to save computing time without losing the clustering performance. On the other hand, the data portion regarded as complex by CL can only be processed with clustering algorithms as they require extensive proximity evaluation and center updates for accurate clustering.

Figure 2 demonstrates the conceptual overview of the CL-integrated k-means clustering method. The pseudocode of the CL-integrated k-means clustering is given in algorithm 1. In algorithm 1, *NF* and *CF* denote the numerical and categorical features, respectively. m denotes the feature size, and m_i refers to one feature (e.g., age/sex) in D. $\mathcal{E}$ and $\mathcal{C}$ refer to easy and complex data, respectively. Algorithm 1 has four key modules: CL application as a pre-processing step to classify D into $\mathcal{E}$ and $\mathcal{C}$, clustering of easy data ($\mathcal{E}$) by respecting the cluster size criteria, k-means clustering application to complex data ($\mathcal{C}$), and combining results of the second and third module to produce final clusters. Lines 2-22 implement the first module by analyzing the values of *CF*, creating templates, and classifying D into two categories (e.g., $\mathcal{E}$ and $\mathcal{C}$). The second module is implemented in Lines 23-24 by determining the # of clusters from $\mathcal{E}$ and dividing $\mathcal{E}$ into clusters of size n. The third module is implemented in Lines 25-26 by applying k-means to $\mathcal{C}$ by finding optimal # of clusters and dividing $\mathcal{C}$ into clusters of size n. The last module combines the results of both the second and third modules to create final clusters (Line # 27).

In algorithm 1, data are analyzed before being processed via k-means. Specifically, some global combinations or templates are determined by using dominant values of categorical features, and records are subsequently matched to them. By identifying and using such templates, a significant amount of data that is naturally clustered is extracted. For example, in the Adult dataset, 89% of the records have the USA in the Country field, 85.42% records have White in the Race field, and 84% records have Male for Gender. By creating a template like $<$ *USA*, *White*, *Male* $>$, a significant portion (e.g., ∼84.01%) of the data that are naturally clustered is extracted beforehand. The global combination of categorical features in a dataset allows us to determine the appropriate template as discussed above. In most datasets, there are dominant values under categorical features, and their correlation with other neighbor categorical features can assist in constructing the appropriate template. In the above example, three categorical features (e.g., country, race, and gender) and the correlation between their values allow the creation of an appropriate template that can be subsequently used to create pre-partition clusters. These findings are reliable in the sense that clustering is built around certain critical features in most cases. For example, in the privacy domain, only some features serve as quasi-identifiers that require clustering, and all other attributes are either removed or not collected. The data classified as easy by CL can simply be divided into clusters based on cluster size requirements. In experiments, we noticed that some records were left after CL-integrated k-means clustering, but the number of records is small, and therefore, the assertions in this paper are still valid.

Algorithm 1 CL-integrated k-means clustering.

```
Require: $D$ (dataset) of $N \times m$ size, $n$ (cluster size)
Ensure: $C$, where $C = \{C_1, C_2, ..., C_u\}$ ▷ Clusters set
 1: $C \leftarrow \emptyset$                ▷ initializing $C$ as empty sets.
 2: for $i = 1$ to length$(m - 1)$ do
 3:     if (typeof$(m_i)$ == NF) then
 4:         Skip additional processing
 5:     else if (typeof$(m_i)$ == CF) then
 6:         $\mathcal{X} \leftarrow \emptyset$        ▷ $\mathcal{X}$ to store dominant values.
 7:         $S \leftarrow$ unique(index$[m_i]$) ▷ $m_i$ unique values.
 8:         $F \leftarrow$ freq$(m_i)$, $\forall v \in S$     ▷ Freq. of values.
 9:         Sort $F$ in ascending order.
10:         Pick highly dominant values $v_1^{m_i}$ from $F$
11:         $\mathcal{X} \leftarrow \mathcal{X} \cup \{v_1^{m_i}\}$         ▷ Add value into $\mathcal{X}$.
12:     end if
13: end for
14: While ($F$ of each CF has been found) do
15: Create templates $\{\sigma_1\}$ from the information in $\mathcal{X}$
16: $\sigma_1 = v_1^{m_1} \cap v_1^{m_2} \cap, \ldots, \cap v_1^{m_{|CF|}}$   ▷ Template creation.
17: End While
18: for $j = 1$ to length$(D)$ do
19:     Map records based on $\sigma_1$ (Template)
20:     $\mathcal{E} \leftarrow \mathcal{E} \cup r_j \in \sigma_1$      ▷ Naturally clustered data.
21:     $\mathcal{C} \leftarrow \mathcal{C} \cup r_j \notin \sigma_1$   ▷ Partially/not clustered data.
22: end for
23: $H \leftarrow \frac{|\mathcal{E}|}{n}$           ▷ Find # of clusters from $\mathcal{E}$ via $n$
24: $P_1 \leftarrow P_1 \cup \{C_1, \ldots, C_{|H|}\}$ -Partition $\mathcal{E}$ into $H$ cluster.
25: Apply $k$-means to $\mathcal{C}$ with optimal # of clusters($I$).
26: $P_2 \leftarrow P_2 \cup \{C_1, \ldots, C_{|I|}\}$ ▷ Partition $\mathcal{C}$ into $I$ cluster.
27: $C \leftarrow C \cup \{P_1, P_2\}$    ▷ Combine clustering results.
28: return $C$, where $C$ has $u$ clusters of size at least $n$
```

The proposed approach utilizes some meta-features and correlations between their values to pre-partition the dataset before the clustering. The resulting pre-partition clusters created with the proposed approach are naturally meaningful because they are determined through the highly dominant values of the meta-features. The remaining complex data is still processed with the k-means algorithm. As stated above, three features are common for 84.01% of the rows in the adult dataset, and pre-partition created by utilizing these three features can accurately capture the patterns w.r.t. income in this dataset. Furthermore, there are many realistic scenarios like customer segmentation based on demographics, bias analysis based on demographics (e.g., gender, race, nationality, etc.), product/POI recommendations, and fairness analysis, where some meta-features are more relevant than others depending upon the context. Our work considers highly relevant meta features information only that encompass many dominant values for creating pre-partition clusters, and therefore, most pre-partitions are naturally meaningful for the intended purposes.

CL-integrated *k*-means clustering is expected to significantly lower computing overhead when clustering large-scale datasets. This approach is handy when most of the datasets belong to the same community/region. Also, this approach is simple because extensive computing operations (e.g., distance, similarity) are executed on only some parts of the data. Lastly, these types of hybrid approaches are vital as avenues for data generation rapidly expand, and getting knowledge from big data in a short time is demanded in many sectors nowadays.

Performance evaluation

To prove the validity of the proposed concept, we conducted a reasonable number of experiments with five real-life benchmark datasets: Adult[3], Census income[4], German Credit[5], Diabetes [6], and Careplans [19]. All datasets are publicly available and have been widely used in previous studies. The Adult dataset encompasses fourteen features, and the class size is two income levels (<=50K, >50K). Similarly, the Census income dataset encompasses fifteen features and the class size is two income levels (-50000, +50000). The German credit dataset encompasses twenty features, and the class size is two credit ratings (Good, Bad). Diabetes is a medical dataset having demographic information and clinical features of individuals. There are forty-seven different features in this data, and the class size is two diabetes diagnosis results (Yes, No). Careplans is also a medical dataset that contains various demographics and information on medical expenses. There are twenty-five different features in this dataset, and the class size is four categories of healthcare expenses. In all datasets, the feature type is mixed (e.g., categorical (string) and numerical (integer)). The minimum cardinality is for the gender feature which is 2, and the maximum cardinality is for the age which is 91. Most datasets are dominated by categorical features. The chosen datasets closely resemble real-world settings in social science and medical domains. Implementation of the concept was performed in Python with the help of multiple packages such as *pandas*, *scikit-learn*, *matplotlib*, *numpy*, and *time*. The implementation procedure is in Figure 3. The key steps are data pre-processing, data classification, data clustering, and generating the final results.

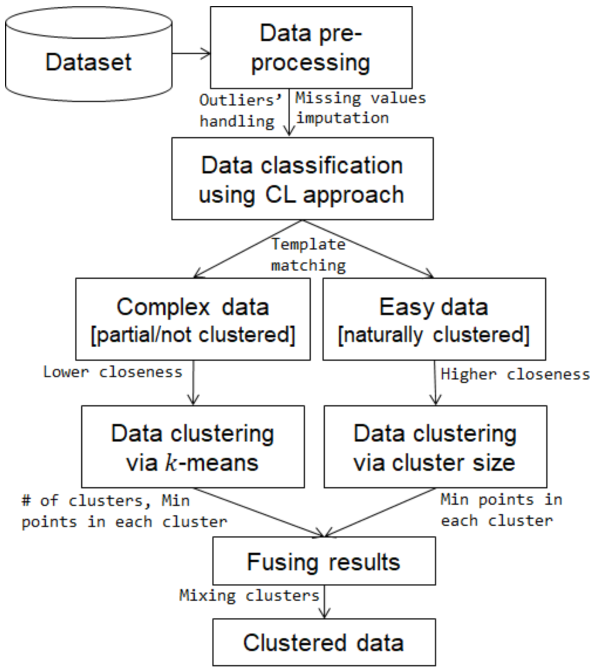
Dataset
Data pre-processing
Outliers' handling
Missing values imputation
Data classification using CL approach
Template matching
Complex data [partial/not clustered]
Easy data [naturally clustered]
Lower closeness
Higher closeness
Data clustering via k-means
Data clustering via cluster size
# of clusters, Min points in each cluster
Min points in each cluster
Fusing results
Mixing clusters
Clustered data

FIGURE 3. Implementation of CL-integrated *k*-means.

In the pre-processing stage, the tuples with null/missing values are imputed, and outliers are identified and replaced with other values. The proposed approach is suitable for tabular data, in which the outliers are removed beforehand by exploiting the information of feature type as well as their legitimate values. For instance, in the age feature, all values need to be numeric and within the desirable range [min-max]. In categorical features like race, there should not be any illegitimate/fake race value. This work assumes that ground truth values and feasible range values are acquired from the data owners and are explicitly used to figure out outliers. This work does not delete any record having outliers instead they are imputed to maintain the data size. The outliers in the numerical features are replaced with the mean of that feature. In contrast, the outliers in categorical features are imputed with less occurring values in that feature. In some cases, the outliers can also be removed once the clustering process is finished by utilizing the distance information of each cluster. Specifically, the distance between the cluster center and data points can be (re)-analyzed and compared with the respective threshold to either eliminate outliers or map them to another

[3]https://archive.ics.uci.edu/dataset/2/adult

[4]https://archive.ics.uci.edu/dataset/117/census+income+kdd

[5]https://archive.ics.uci.edu/dataset/573/south+german+credit+update

[6]https://archive.ics.uci.edu/dataset/296/diabetes+130-us+hospitals+for+years+1999-2008

cluster.

In the next step, data is classified with the help of the CL approach. Specifically, we divide datasets into two parts: easy and hard. The process of dividing the dataset using the template created with dominant CF values is expressed in Lines 2-22 of algorithm 1. The easy parts of the data are readily clustered, and therefore, they are simply divided into different clusters by respecting the cluster size, as expressed in Lines 23-24 of algorithm 1. The complex data is fed into k-means for clustering as per the cluster size, as shown in Lines 25-26 of algorithm 1. Later, the results from both processes are fused, as shown in Line 27 of algorithm 1. In the end, clustered data is achieved, where each cluster is compact and encompasses at least n data points, as given in output (Line 28) of algorithm 1. The value of n can be adjusted as per the desired # of data points in each cluster. The issue of uneven cluster distribution occurs when the optimal value of k is not used. In this work, we determined the optimal values of k using the elbow method for each dataset, and therefore, most clusters were balanced. We also introduced a small modification (e.g., *min_data_points_per_cluster* = 5) to the code to keep the minimum # of data points in each cluster. We believe the maximum bound can also be imposed in a similar way to curate balanced clusters.

To compare the performance of the proposed method, five different evaluation metrics were used: inertia, silhouette score (SS), Davies-Bouldin Index (DBI), computing time, and number of iterations. The first three metrics are used to evaluate clustering performance and the last two are used to evaluate computing overheads. The formulization for calculating inertia (Inr) is Eq. 2.

$$Inr = \sum_{i=1}^{N} (x_i - C_k)^2 \tag{2}$$

where N denotes number of samples, x denotes samples' values, and C is the center of the cluster. The formulization for calculating SS is Eq. 3.

$$SS = \frac{b - a}{max(a, b)} \tag{3}$$

where b is the distance between a data point and nearby clusters that a data point doesn't belong to, and a denotes the intra-cluster distance. The value for all data points is computed and the mean value is used in calculations. The formulization for calculating DBI is Eq. 4.

$$DBI = \frac{1}{n'} \sum_{i=1}^{n'} max_{i \neq j} \left(\frac{\alpha_i + \alpha_j}{d(c_i, c_j)} \right) \tag{4}$$

where n' shows the # of clusters, and α_i is the average distance of all data points in a cluster i from centroids c_j. The formulization for iteration is in Eq. 5.

$$iterations = count(Itr) = Total \ \# \ of \ iterations \tag{5}$$

The formalization for calculating time is in Eq. 4.

$$time = T_{end} - T_{start} \tag{6}$$

where T_{end} is the end time of clustering, and T_{start} is the begining time.

The open-source Python library *scikit-learn* provides an implementation of most metrics under the *sklearn.metrics* module. The time was measured using *time* library in Python.

In experiments, we chose various salient attributes and data points from these datasets for evaluation. The numbers of records in these datasets are 32561, 199523, 1000, 101766, and 12352, in Adult, Census income, German credit, Diabetes, and Careplans, respectively. The CL application determined that the following percentages of data are already clustered: (1) Adults [easy: 84.01% (partially)hard:15.99%], (2) Diabetes [easy: 42.83% (partially)hard:57.17%], (3) Careplans [easy: 74.18% (partially)hard:25.82%], (4) Census [easy: 76.03% (partially)hard:23.97%], and (5) German [easy: 51.70% (partially)hard:48.30%].

Based on the above results, we conclude that CL is a handy approach to figuring out naturally clustered parts of datasets, and their identification can ease the clustering process. However, the traditional k-means does not identify such valuable information concerning datasets, and therefore, the computing overhead is high. Figure 4 presents a comparison of computing time incurred by conventional k-means, mini-batch k-means, distributed ensemble clustering (DEC) framework [16], and the CL-integrated k-means algorithms while clustering the five real-life datasets. The x-axis shows the min. cluster size, where 1=5, 2=10, 3=30, 4=50, and 5=100, respectively. From the results, we can see that computing time rises with cluster size owing to an increase in computing operations. However, computing time for the CL-integrated k-means

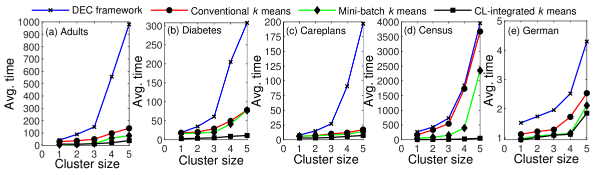

FIGURE 4. Comparisons of computing time on each dataset.

algorithm is significantly lower (e.g., an 83.66% improvement) compared to the conventional *k*-means. Our algorithm has yielded 53.05% improvements as a whole in reducing computing time than mini-batch *k*-means (a faster implementation of *k*-means). However, the only dataset upon which our algorithm yielded marginal improvements over mini-batch *k*-means is German, where the improvements are only 6.25%. These results fortify the assertions of this paper and prove that CL-integrated *k*-means can be a valuable solution in realistic cases.

In the next set of experiments, we compare the number of iterations required to cluster data using the three existing and proposed method. Figure 5 presents the experimental results on the number of iterations (by averaging five tests) that were achieved with the five benchmark datasets.

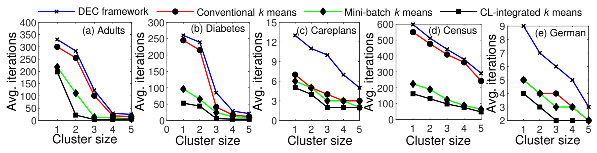

FIGURE 5. Comparisons of # of iterations on each dataset.

From the results in Figure 5, the number of iterations decreased along with cluster size due to an increase in mapping space (a.k.a. relaxed clustering). The proposed CL-integrated *k*-means has significantly less number of iterations compared to the conventional *k*-means and two other clustering methods. These results further clarify the significance of our method in terms of reducing overheads.

To further justify the efficacy of the proposed approach, we compared the results of the clustering effect with three state-of-the-art approaches by using three evaluation metrics (e.g., *Inr*, *SS*, *DBI*). The lower *Inr* indicates that clusters are compact, thereby better clustering performance[7]. The *SS* lies between -1 and 1, where a higher value (e.g., close to 1) indicates the better separation between clusters[8]. The lower value for *DBI* is desirable for well-separated clusters [9]. Figure 6 presents the results and comparisons of the clustering effect on each dataset. The *x*-axis shows the datasets, where A=Adults, B=Diabetes, C=Careplans, D=Census, and E=German, respectively. From the results, it can be seen that our algorithm has yielded lower values for the *Inr* and *DBI* on most datasets,

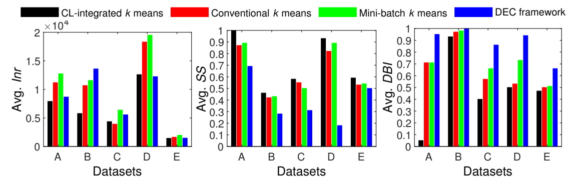

FIGURE 6. Comparisons of clustering effect on each dataset.

which indicates superior performance than previous methods. Also, the *SS* score is higher than the previous baselines. These results indicate comparable performance from our approach w.r.t. previous methods.

From the results given in Figures 4-6, it can be observed that the proposed CL-integrated *k*-means has lower computing overheads and better clustering effect than two baseline methods and conventional *k*-means clustering. The improvements in the first part are due to data reduction to be processed in clustering, and lowering the extensive computing operations such as proximity analysis and center updates. In contrast, both baseline methods do not explore ways to reduce data size before clustering and perform more computing operations than the proposed method. Hence, their computing overheads are relatively larger than our method. The clustering effect results are better from our method as data characteristics are exploited to accurately classify data based on information on meta-features and their values. In contrast, the existing methods do not identify such valuable information from *D*, leading to low clustering quality despite performing more computations/mathematical operations.

Also, the previous baselines require additional optimizations in terms of selecting the appropriate batch size, # of random samples, and subset size. In contrast, the proposed approach requires few optimizations (e.g., only appropriate # of clusters need to be determined beforehand). Lastly, the number of leftover records after clustering under the CL-integrated *k*-means was also lower than with the conventional *k*-means. These results verify the significance of the CL-integrated *k*-means algorithm in realistic scenarios. In some cases, coarse clustering can only be required to learn the characteristics of data, and therefore, the CL-integrated *k*-means clustering can be very helpful in this context. In addition, optimization of both time and the number of iterations extends the clustering applications to very large datasets, and the results can be obtained quickly.

[7]https://victorleungtw.com/2023/12/16/inertia/
[8]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
[9]https://www.geeksforgeeks.org/davies-bouldin-index/

Novelty, significance, limitations: To the best of our knowledge, this is pioneering work that has explored ways to reduce the computing overhead from conventional k-means clustering by exploiting data characteristics and compositions. The excessive, but often unnecessary, computation of distances/similarities is reduced by separating data that are readily clustered, which makes the clustering process an order of magnitude faster than conventional k-means. The proposed concept mingles conventional AI approaches with the latest ones to address performance bottlenecks. This work aligns with recent trends toward exploiting data to attain useful results for AI applications [20]. The proposed method can be highly useful in domains where computing budgets are small or data belong to the same population group or origin. Lastly, this work can reduce memory runout issues and long latency when the amount of data to be clustered is large. This work is a timely contribution to the AI field because low-cost solutions are always in demand, especially to deploy AI solutions on resource-constrained devices. Although our approach has merits, some data characteristics can degrade its performance. For instance, CL may not capture some important patterns when either all features in the data are numeric or dominant values do not exist for categoric features in a dataset.

CONCLUSION

This paper proposed and implemented a novel CL-integrated k-means clustering method that is significantly faster than conventional k-means clustering. In our proposed method, the data to be clustered are analyzed and sorted in terms of complexity by using the CL approach, and some readily clustered parts of the data bypass the clustering process. By doing so, unnecessary computing operations and the number of iterations are restrained, leading to significantly reduced computing overhead. The experimental results achieved from five benchmark datasets proved the significance of the proposed method over the conventional k-means and two other baseline methods. Our findings give new insights to AI and database communities in terms of leveraging the latest data-centric AI techniques to improve critical aspects of conventional ML approaches. In the future, we intend to apply the proposed approach to optimizing other clustering algorithms.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (2020R1A2B5B01002145).

REFERENCES

1. K. Arumugam, M. Naved, P. P. Shinde, O. Leiva-Chauca, A. Huaman-Osorio, and T. Gonzales-Yanac, “Multiple disease prediction using machine learning algorithms,” *Materials Today: Proceedings*, vol. 80, pp. 3682–3685, 2023.
2. X. Shu and Y. Ye, “Knowledge discovery: Methods from data mining and machine learning,” *Social Science Research*, vol. 110, p. 102817, 2023.
3. W. Sun, J. Peng, G. Yang, and Q. Du, “Fast and latent low-rank subspace clustering for hyperspectral band selection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 3906–3915, 2020.
4. H. Hu, J. Liu, X. Zhang, and M. Fang, “An effective and adaptable k-means algorithm for big data cluster analysis,” *Pattern Recognition*, vol. 139, p. 109404, 2023.
5. A. Abbasi and B. Mohammadi, “A clustering-based anonymization approach for privacy-preserving in the healthcare cloud,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 1, p. e6487, 2022.
6. Z. Huang, Y. Xiang, B. Zhang, D. Wang, and X. Liu, “An efficient method for k-means clustering,” *Deakin University*, 2023.
7. Y. T. Chen and D. M. Witten, “Selective inference for k-means clustering,” *Journal of Machine Learning Research*, vol. 24, no. 152, pp. 1–41, 2023.
8. A. Qtaish, M. Braik, D. Albashish, M. T. Alshammari, A. Alreshidi, and E. J. Alreshidi, “Optimization of k-means clustering method using hybrid capuchin search algorithm,” *The Journal of Supercomputing*, pp. 1–60, 2023.
9. J. Zhang, H. Tao, and C. Hou, “Imbalanced clustering with theoretical learning bounds,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 9, pp. 9598–9612, 2023.
10. N. H. Shrifan, M. F. Akbar, and N. A. M. Isa, “An adaptive outlier removal aided k-means clustering algorithm,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6365–6376, 2022.
11. S.-S. Yu, S.-W. Chu, C.-M. Wang, Y.-K. Chan, and T.-C. Chang, “Two improved k-means algorithms,” *Applied Soft Computing*, vol. 68, pp. 747–755, 2018.
12. A. Zhu, Z. Hua, Y. Shi, Y. Tang, and L. Miao, “An improved k-means algorithm based on evidence distance,” *Entropy*, vol. 23, no. 11, p. 1550, 2021.
13. D. Wei and Z. Zhang, “K-means clustering algorithm based on improved density peak,” in *Proceedings of*

the 2023 3rd International Conference on Bioinformatics and Intelligent Computing, 2023, pp. 105–109.

14. G. Kong, Y. Ma, Z. Xing, and X. Xin, “Multi-view k-means clustering algorithm based on redundant and sparse feature learning,” *Physica A: Statistical Mechanics and its Applications*, vol. 633, p. 129405, 2024.
15. H. Zhang, J. Li, J. Zhang, and Y. Dong, “Speeding up k-means clustering in high dimensions by pruning unnecessary distance computations,” *Knowledge-Based Systems*, vol. 284, p. 111262, 2024.
16. M. S. Mahmud, J. Z. Huang, and S. García, “Clustering approximation via a fusion of multiple random samples,” *Information Fusion*, vol. 101, p. 101986, 2024.
17. A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data,” *Information Sciences*, vol. 622, pp. 178–210, 2023.
18. X. Wang, Y. Chen, and W. Zhu, “A survey on curriculum learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, 2021.
19. Z. El Ouazzani, A. Braeken, and H. El Bakkali, “Proximity measurement for hierarchical categorical attributes in big data,” *Security and Communication Networks*, vol. 2021, 2021.
20. A. Langer and A. Mukherjee, “Building data-centric products,” in *Developing a Path to Data Dominance: Strategies for Digital Data-Centric Enterprises*. Springer, 2023, pp. 143–161. [Online]. Available: https://doi.org/10.1007/978-3-031-26401-6_6

Abdul Majeed is an Assistant Professor in the Department of Computer Engineering, Gachon University, Korea. He received his Ph.D. in Computer Information Systems & Networks from the Korea Aerospace University, Korea, in 2021. His research interests include data-centric artificial intelligence, machine learning, information privacy, and secure personal data publishing. Contact him at ab09@gachon.ac.kr

Seong Oun Hwang is a Professor in the Department of Computer Engineering, Gachon University, Korea. He received his Ph.D. in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2004. He is a senior member of IEEE. He is the corresponding author of this paper. His research interests include cryptography, data-centric artificial intelligence, cybersecurity, and machine learning. Contact him at sohwang@gachon.ac.kr