Reuse-Centric K-Means Configuration

Hui Guan, Yufei Ding, Xipeng Shen, Hamid Krim
North Carolina State University, Raleigh, NC, USA 27519
{hguan2, yding8, xshen5, ahk}@ncsu.edu

***Abstract*—K-means configuration is a time-consuming process due to the iterative nature of k-means. This paper proposes *reuse-centric k-means configuration* to accelerate k-means configuration. It is based on the observation that the explorations of different configurations share lots of common or similar computations. Effectively reusing the computations from prior trials of different configurations could largely shorten the configuration time. The paper presents a set of novel techniques to materialize the idea, including *reuse-based filtering*, *center reuse*, and a two-phase design to capitalize on the reuse opportunities on three levels: validation, k, and feature sets. Experiments show that our approach can accelerate some common configuration tuning methods by 5-9X.**

I. Introduction

The effectiveness of k-means in applications depends on many factors, such as the features used for clustering and the resulting number of clusters. As a result, algorithm configuration is essential for k-means–based data mining [1]. However, k-means configuration is a time-consuming process due to the iterative nature of k-means and the requirement of many runs of k-means in various settings.

There are some general methods proposed for speeding up the configuration process of algorithms [2]. They have mostly focused on how to reduce the number of trial configurations. How to accelerate the examination of the remaining configurations through historical information reuse is a complementary direction that has not received sufficient explorations. And how to effectively accomplish it for k-means is yet a largely unexplored problem.

This paper presents a systematic exploration in that direction. It introduces the concept of *reuse-centric k-means configuration*, which promotes information reuse across the explorations of different configurations of k-means. The motivating observation is that the explorations of different configurations of k-means share lots of common and similar computations. Effectively reusing the computations could largely cut the configuration time.

Specifically, we have designed two techniques, called *reuse-based filtering* and *center reuse*, to promote computation reuse across trials of different configurations. *Reuse-based filtering* takes advantage of the clusters and the distance between a point and its nearest center unveiled in a previous trial of k-means. Through the reuse, it is able to use computationally efficient lower bounds of the distances between a point and potential centers to filter out some centers that are unlikely to be the nearest to a point, and avoid calculating the distances to those centers. *Center reuse* is to use the clustering results of some earlier trials to initialize cluster centers for some later trials on different configurations. The reuse helps make the later trials converge faster and hence saves configuration time.

For both types of reuse, we have explored the opportunities in multiple levels: across validations, across k, and across feature sets. Besides their effectiveness in drastically cutting configuration time, an appealing property of these techniques is their simplicity. They are designed to be simple to implement and deploy to ensure their applicability in general data mining applications.

In addition to the two techniques, we have also explored the use of a *two-phase design* to speed up the configuration process when a full error surface is needed for meeting various desired trade-offs among multiple quality metrics.

We evaluate the efficiency and effectiveness of these techniques in both sequential and parallel settings. Our results show that these techniques can work together in synergy, speeding up a heuristic search-based configuration process by up to 5.8X and the attainment of the error surface of k-means–based classifiers by a factor of 9.1. We also confirm only little disparity the proposed techniques may cause to the quality of k-means results.

II. Proposed Techniques

We describe, in this section, our proposed techniques. In k-means–based applications, data are first projected onto some feature space. K-means clustering subsequently runs on the projected data to form some clusters, which are then used by the application for some follow-up purposes (e.g., classification). We consider two important factors of k-means to configure: S, the set of features to extract or select from the raw data, and k, the number of clusters to form. Although the configuration involves only two factors, even with the fast Yinyang k-means algorithm [3], on a dataset of modest size, the configuration still takes a long time (days) to explore all combinations of k and feature sets.

Our acceleration techniques consist of three stages: *reuse-based filtering*, *center reuse*, and a *two-phase design*. They work at different aspects of the problem and can function in synergy. The dash-lined boxes in Figure 1 illustrate the scopes they each work on. We next explain each of the three techniques in detail.

A. Reuse-Based Filtering

In the standard k-means, each iteration needs to compute $n \times k$ distances (n is the number of points, k is the number of cluster centers), from every data point to every cluster center in order to identify which cluster center is the closest to the

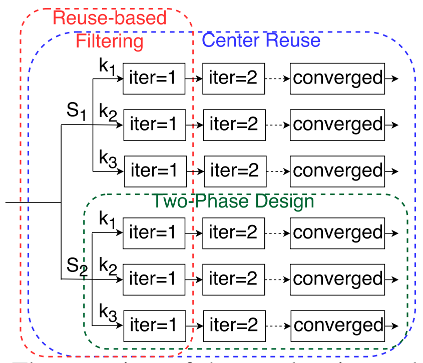
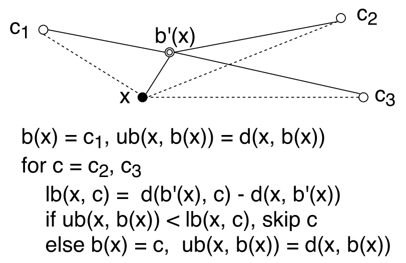


Fig. 1: The overview of the acceleration techniques.



```
b(x) = c1, ub(x, b(x)) = d(x, b(x))
for c = c2, c3
    lb(x, c) =  d(b'(x), c) - d(x, b'(x))
    if ub(x, b(x)) < lb(x, c), skip c
    else b(x) = c,  ub(x, b(x)) = d(x, b(x))
```

Fig. 2: Illustration of how the configuration with $k = k_1$ can help save distance computation in the first iteration of another configuration with $k = k_2$. $b'(x)$ is the closest center of point x when $k = k_1$; c_1, c_2, c_3 are the initial centers and $b(x)$ is the so-far nearest center of point x when $k = k_2$.

data point. Modern k-means algorithms (e.g., Yingyang k-means [3]) successfully avoid many distance calculations in later iterations of a k-means, but they all still need the $n \times k$ distance calculations in the first iteration of k-means. In our experiments, we observe that the first iteration weights up to 40% of the entire k-means time. We call it the *first iteration problem*. K-means configuration needs many runs of k-means; every one of them suffers from the *first iteration problem*.

To alleviate the issue, we propose *reuse-based filtering*, which is based on the well-known geometric property of *Triangle Inequality* (TI). We explain the design of reuse-based filtering in two levels: across k and across feature sets.

Reuse across k happens among the configurations that share the same set of features, with different k values. Figure 2 provides the illustration. Note that the distance from x to $b'(x)$ can be directly reused from the trial with $k = k_1$, the only extra distance computations we need to carry out are those from the centers in $k = k_1$ to the initial centers in $k = k_2$. In total, there are $k_1 \times k_2$ distance computations, which is negligible in comparison to the cost of distance computations from every point to every center (i.e. $n \times k_2$), where n is the total number of points. Similarly to the previous usage of TI [3], [4], this optimization does not change clustering results as distance computations to some center c will be eliminated only when c can not be the closest center to the point x.

Reuse across feature sets happens among the configurations that have the same number of clusters k, but use different sets of features. A distance computed in dimension space F_1 with a feature set S_1 can be used for bound computations in dimension space F_2 with a feature set S_2 without affecting clustering results as long as S_2 is a subset of S_1. In particular, for each center c'^{F_1} obtained in dimension space F_1, we remove those dimensions that are not used in F_2 to build a corresponding center c'^{F_2} in dimension space F_2. The upper bound $ub(x, b(x))$ is the same as Figure 2 while the lower bound becomes $lb(x^{F_2}, c^{F_2}) = d(b'(x)^{F_2}, c^{F_2}) - d(x^{F_1}, b'(x)^{F_1})$

B. Center Reuse

The second technique we propose for accelerating k-means configuration is called *center reuse*. The basic idea is to use the cluster centers attained in the exploration of some earlier configuration as the initial centers for the explorations of later configurations. Specifically, we consider *center reuse* in three scenarios, corresponding to the different levels of the explorations of k-means configurations shown in Figure 1.

Reuse across validations is among the different folds in cross validations in the exploration of a certain configuration. *Reuse across* k happens among the configurations that share the same set of features, but different k values. Suppose that the reuse is from one configuration with $k = k_1$ to another with $k = k_2$. If $k_2 > k_1$, in addition to using the centers attained in exploring the earlier configuration, we add $k_2 - k_1$ randomly generated centers as needed. If $k_2 < k_1$, we cluster the k_1 centers into k_2 groups and then take the group centers as the initial centers for the exploration of the latter configuration. *Reuse across feature sets* happens among the configurations that use different sets of features. Suppose that the reuse is from configuration C_1 with feature set S_1 to configuration C_2 with feature set S_2. Our design is to reuse the values of overlapped features between S_1 and S_2, and generate the values of the other features of S_2 (if there are any). The generated value can be the mean value of each feature.

When two configurations differ in both k and feature sets, *center reuse* first applies the *reuse across feature sets* to handle the dimension differences in the features, and then applies the *reuse across* k to set the initial cluster centers.

C. Two-Phase Design

The *two-phase design* is based on the observation that some points on an error surface more critically affect the accuracy of the error surface than some other points do. The idea of the *two-phase design* is to first quickly obtain an estimated shape of the error surface, based on which, it then conducts a focused exploration of the configurations (e.g., those fall into the elbow area) that are most important for the accuracy of the final error surface.

The first phase employs both the *reuse-based filtering* and *center reuse* for speed. It also adopts two approximation methods. The first is to use only the first fold of cross-validation; the second is to replace k-means clustering with only a one-step clustering. The one-step clustering assigns points to clusters based on their distances to the centers produced from our *center reuse* scheme; no center updates or point reassignments are done. Based on that shape of curve, the second phase identifies the critical sections (i.e., the elbow section [5]) of

TABLE I: Data Statistics.

Dataset	size(B)	n	#attr	#c	#d	*dstep*	*kstep*
gamma	1.2e6	1.9e4	10	2	8	1	10
sensorless	4.4e6	5.8e4	49	11	10	1	10
credit [6]	3.0e6	3.0e4	24	2	13	1	10
gassensor [7]	1.7e6	1.4e4	11	2	16	1	10
miniboone	3.4e7	1.3e5	50	2	34	2	20
adult	2.0e7	4.5e4	14	2	59	2	10
connect	3.1e7	6.8e4	42	2	61	2	10
activity [8]	1.1e7	1.0e5	561	6	157	8	10
census	2.0e8	1.4e5	68	2	186	8	20

TABLE II: Speedup on Stochastic Hill Climbing.

Dataset	time(s)*	reuse-based	center reuse	
		filtering	across validations	across k and feature sets
gamma	2808.1	3.09	3.58	2.05
sensorless	10730.6	3.22	3.40	1.73
credit	5713.5	3.30	4.05	2.04
gassensor	1901.9	4.37	4.39	2.33
miniboone	56636.8	1.56	3.73	1.83
adult	9904.4	4.30	5.80	2.18
connect	23621.7	1.54	4.42	1.63
activity	6569.6	1.11	2.76	1.91
census	79872.7	2.30	4.14	1.68

* time(s) refers to k-means clustering time in seconds for all 200 configurations without our optimizations.

the curve and selects some important configurations to conduct more focused and detailed explorations to subsequently get the precise accuracy at those points. Through interpolation across those points, it finally obtains the error curve.

III. Evaluations

We conduct a series of experiments to evaluate the proposed techniques on both speedups and clustering quality. Our experiments use k-means–based classification as a concrete usage scenario of k-means configurations. It is worth noting that the techniques are general, applicable to other usage of k-means.

All the experiments run on an HPE Apollo 2000 server with two Intel Haswell CPUs (14 cores/CPU, 2.3-3.3GHz) and 128GB RAM. We use nine large, real-world data sets taken from the UCI machine learning repository [9]. The statistics of the datasets including data size (size), the number of instances (n), the number of attributes (#attr) and the number of classes (#c) are listed in Table I. We used principal component analysis (PCA) as the feature projection method to extract features from attributes and adopted the step-wise feature selection method to select feature sets. We retained a maximum number of components that cumulatively explain 99% of variation. The minimum number of dimensions is two and the smallest value of k is 20. The overall range of feature dimensions to consider in the configuration explorations is shown in the "#d" column. "$dstep$" and "$kstep$" columns show the default step size to increase or decrease the dimension and the number of clusters k respectively.

The state-of-the-art k-means algorithm Yinyang k-means [3] is used in the baseline implementations of the algorithm configuration. Euclidean distance is used in all the experiments.

A. Speedups on Heuristic Search

This part reports the speedups brought to the k-means configuration process by our *reuse-based filtering* and *center reuse*. Algorithm configurations typically employ some heuristic search algorithms to explore the configuration space. Our optimizations are largely orthogonal to what search algorithms are used. Our experiments use stochastic hill climbing [10].

The tuning objective considers both the classification accuracy and the response time of the built classifier. Specifically, it is to find the smallest k (hence giving the fastest classification response) that can achieve a classification accuracy over a given threshold (e.g. 90%). The stopping criterion is that the maximum number of configurations (200) is tested.

The speedups from each technique are listed in Table II. For reuse-based filtering, we report the speedup of reuse-based filtering for the first iteration. The speedup comes from the distance computation saved by TI-based filtering. Center reuse affects the entire k-means clustering and thus has the dominant influence on the overall speedup. It confirms that our simple design for center reuse across k and features sets works well.

B. Speedup on the Attainment of Error Surfaces

Our second experiment studies the benefits of our techniques on the attainment of classification error surfaces. As Section II-C has mentioned, error surfaces capture the relations between configurations and the errors of the corresponding classifiers. They could be helpful when the criterion for the best configuration varies. In this experiment, we apply all three techniques that Section II proposes to accelerate the attainment of the error surfaces. Uniform search of the configuration space is used in the baseline implementation. It evaluates every combination of the uniformly sampled d and k values.

Our proposed method is to use two-phase design to reduce the number of k to be evaluated for building a classification error curve. Computation reuse techniques are applied to save the clustering time for each sampled configuration. Our method starts with the largest d and the largest k and apply reuse to smaller d values and k values. We apply all the three techniques to speed up sequential uniform search, and then parallel uniform search on the 28-core parallel machine.

1) Sequential: In this setting, only one thread is used for the configuration process. The number of sampled k is $8, 16, 32$. The speedups from all three techniques are listed in Table III. With our two-phase design, we could reduce the number of k to be evaluated for recovering the error curve without affecting the benefits from the computation reuse. As a consequence, the overall speedup is up to 9.17X.

The overall speedup on the dataset *activity* is not as high as on the other datasets. Specifically, the dominated acceleration factor, center reuse across validations, produces a smaller speedup compared with that on the other datasets. In contrast, the results from the dataset *census*, which has the same large dimensions but about 14 times larger data size, shows much larger speedups for reuse-based filtering and center reuse across all three levels. This is because when a dataset is relatively small but has a large size of feature sets, training sets are likely to follow different distributions and thus the centers resulting from one fold of training set is not as good for another fold of training set.

TABLE III: Speedup on the Attainment of Error Surfaces.

Dataset	Speedup by Computation Reuse (#k=32)					Percentage of k Saving			Overall Speedup		
	reuse-based filtering		center reuse			two-phase design			#k=8	#k=16	#k=32
	across k	across feature sets	across validations	across k	across feature sets	#k=8	#k=16	#k=32			
gamma	3.42	3.12	4.81	2.78	1.44	11.51%	29.52%	16.81%	4.57	6.30	5.70
sensorless	3.61	3.80	4.50	2.37	1.59	21.41%	31.00%	24.21%	5.34	6.73	6.94
credit	3.65	3.70	5.31	2.81	1.82	6.69%	23.82%	15.09%	4.98	6.64	6.77
gassensor	4.31	4.34	5.75	3.32	3.19	26.54%	38.77%	11.95%	6.54	8.59	6.74
miniboone	1.91	1.97	4.56	2.57	2.02	3.85%	49.24%	48.98%	4.50	8.24	9.17
adult	4.65	5.05	6.43	3.86	3.88	12.99%	24.64%	22.71%	6.76	8.27	9.07
connect	1.59	1.74	4.13	2.91	2.28	11.19%	13.88%	5.89%	4.58	5.07	4.98
activity	1.21	1.23	2.77	1.84	2.07	15.25%	17.36%	13.79%	3.02	3.49	3.34
census	2.71	2.78	4.89	2.63	2.61	16.57%	33.12%	28.91%	5.34	7.32	7.64

TABLE IV: Speedup in Parallel Settings.

Dataset	#threads	Speedup			
		#k=5	#k=10	#k=20	#k=30
credit	2	3.73	4.01	4.77	5.12
	4	3.96	4.49	4.89	5.08
	8	3.71	4.40	4.60	4.88
	16	3.69	4.02	4.30	4.75
adult	2	5.65	6.01	6.45	6.68
	4	5.48	5.32	5.98	6.67
	8	4.87	5.64	6.06	6.31
	16	2.41	3.06	5.77	5.94

2) Parallel: The parallel results are interesting to examine because our techniques, especially the computation reuses, bring data dependences to the exploration of different configurations: for a configuration to reuse results from another, it has to wait for the results to be produced. They hence could hamper the parallel search.

Table IV presents results of our algorithms in parallel settings when the two computation reuse techniques are applied. Results on two datasets are shown due to the space constraint. In order to run our algorithms in parallel, some dependencies incurred by the computation reuse have to be removed. When scheduling the task to each thread, we use the following strategy to break dependencies: if the number of threads supported is no larger than the number of feature sets to be evaluated, then only remove dependencies caused by reuse across feature sets. Each thread examines a subset of feature sets and the entire sampled k values. To balance the workload among the threads, we assign each thread feature sets in an alternating manner. For example, if the dimensions are from two to five and the number of threads is two, then the first thread runs dimensions two and four while the second thread runs dimensions three and five. When the thread supported is larger than the number of feature sets, some dependencies caused by reuse across k are also removed in the same way.

As shown in Table IV, we have good speedups with various numbers of threads and numbers of sampled k. The larger the number of sampled k is, the larger the speedup is. This is because we have a larger ratio of reusable distance computation for a larger set of sampled k.

C. Quality Influence of Center Reuse

Among all the three optimizations we introduce, only center reuse might affect the clustering quality due to the sensitivity of k-means on initial centers. To evaluate the quality influence of center reuse, we perform 100 runs of k-means–based classification with different random seeds for each configuration and take the average. We use Mean Percent Error (MPE) to measure the discrepancy between values of an external metric (classification error) and internal metrics (Davies-Bouldin index, Dunn index, Silhouette coefficient and within-cluster sum of squares (WCSS)) with our center reuse technique and those without the optimization. Experiment results show that the MPEs are lower than 3% for external metrics and 5% for internal metrics, indicating the little influence of center reuse on classification and clustering quality.

IV. Conclusion

In this work, we introduced the concept of *reuse-centric k-means configurations* to promote information reuse across the explorations of different configurations of k-means. Our computation-reuse promotion techniques, *reuse-based filtering* and *center reuse*, could largely cut the configuration time of k-means–based data classification. We also introduced a two-phase design, which when working in synergy with the other two techniques, reduces the uniform-search–based attainment of classification error surfaces by a factor of 9.

References

[1] A. Kalogeratos and A. Likas, "Dip-means: an incremental clustering method for estimating the number of clusters," in *Advances in neural information processing systems*.

[2] Holger H. Hoos, *Automated Algorithm Configuration and Parameter Tuning*. (Edited by Y. Hamadi etc.) Springer Berlin Heidelberg, 2012.

[3] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*.

[4] C. Elkan, "Using the triangle inequality to accelerate k-means," in *ICML*.

[5] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a" kneedle" in a haystack: Detecting knee points in system behavior," in *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*. IEEE, 2011, pp. 166–171.

[6] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*.

[7] R. Huerta, T. Mosqueiro, J. Fonollosa, N. F. Rulkov, and I. Rodriguez-Lujan, "Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring," *Chemometrics and Intelligent Laboratory Systems*.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *ESANN*, 2013.

[9] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[10] S. Russel and P. Norvig, "Artificial intelligence: A modern approach, 2010," *EUA: Prentice Hall*.