Reproducibility Challenge:
Making AI Forget You: Data Deletion in Machine Learning

Xiaohui Wang, Zijin Nie, and Zijun Yu

Abstract - This report examines the reproducibility of the paper *Making AI Forget You: Data Deletion In Machine Learning*[1]. The original paper initiated a framework studying what to do when specific data is no longer accessible for deploying models. The paper also proposed two efficient deletion algorithms for k-means clustering model called Q-k-means and DC-k-means. We studied the replicability of the two algorithms, examining if they could achieve similar performance as they stated in the report. By re-implementing Q-k-means and DC-k-means, we reproduced the experiments using MNIST, Gaussian and Covtype datasets. We also performed the experiments of the effect of parameter width and labda to the efficiency of algorithms. The experiments we conducted yields similar results as shown in the original paper.

I. INTRODUCTION

RECENTLY, intense discussions have focused on individual's right to remove their personal data from internet search or other directories. Many efforts have been put towards this direction including the EU's Right To Be Forgotten. The Right allows people to remove personal information from the Internet or other directories by request. If these policies come into effective, it could bring a challenge to data science field and any related fields since research and outcome based on the affected data would be redone or re-evaluated. As a recent mentioned topic, there are not many research done about efficient data deletion problems. Thus, a framework studying how to efficiently remove specific data applied in machine learning models needs to be formalized.

In the original paper, Ginart *et al.* developed a notion of deletion efficiency for large-scale learning systems. They gave formal definition of data deletion, approached data deletion as an online problem and rose the notion of deletion efficiency. In specific, they proposed two deletion efficient unsupervised clustering algorithms, Q-k-means, a quantized variant of Lloyd's algorithm and DC-k-means (Divide-and-Conquer k-means), which works by partitioning the dataset into sub-problems and recursively merging the result of each subproblem. Also, the authors provided detailed mathematical proof of the runtime of algorithms.

In this report, we reproduced what we think is the most important content. We implemented the Q-k-means and DC-k-means using three of the datasets which Ginart *et al.* used, MNIST, Gaussian and Covtype. Following the algorithms and experiments setup in the original paper, we took out experiments testing the data deletion efficiency and clustering performance. The results we got accord with what the original paper produced, which proved its reproducibility.

II. RELATED WORK

Aside from Q-k-means and DC-k-means, there are multiple efficient deletion operations as introduced by the authors of the original paper, which are known for some canonical learning algorithms. The non-parametric Nadaraya-Watson kernel regressions[2] and nearest-neighbors methods[3] are examples of lazy learning techniques for such efficient deletion operations.

The authors addressed the issue of data privacy with regards to data deletion. Cryptography and differential privacy[4], for example, aim to make data non-identifiable and secure while the major concern is not how to delete data efficiently.

III. DATASETS

The datasets that we applied in our experiments are the covtype, MNIST and Gaussian datasets. The exclusion of celltype, posture and botnet datasets is due to inaccessibility of datasets or limited computational resource. The original paper states that the specified celltype data was retrieved from the Mouse Cell Atlas[5], where the described dataset is not currently available. The botnet dataset is available online[6], however, we decided not to conduct the experiments with it because the training duration is lengthy for botnet dataset using limited computational power accessible.

A. Proposed Datasets for Experiments

In the original paper, 6 datasets were used for evaluating the algorithms.

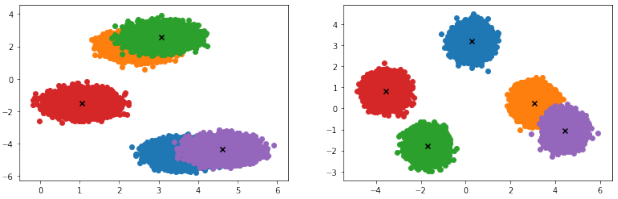
1) Celltype consists of single cell RNA sequences for 4 cell types: microglial cells, endothelial cells, fibroblasts, and mesenchymal stem cells.
2) Covtype consists of cartographic features at various times of day for 7 forest cover types. The original paper states that it has applied 15,120 samples for the implementation.
3) MNIST consists of images of handwritten digits and the task is to classify each image into one of the ten classes. The MNIST dataset is available in Keras library.
4) Posture consists of signals of 5 different hand postures with unlabeled markers attached to a glove.
5) Botnet consists of malicious and non-malicious traffic data between different IP addresses. The task is to classify benign and 11 classes of malicious traffic data.
6) Gaussian dataset synthesized by ourselves.

B. Synthesis and Pre-processing of Datasets

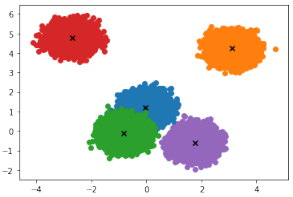
1) Gaussian

The Gaussian dataset we created consists of 5 clusters that were generated from 25-variate Gaussian distributions

centered at randomly chosen locations. For each cluster, 20,000 samples are taken and there are in total 100,000 samples in the Gaussian dataset.



(a) using dimension 2 and 10 (b) using dimension 16 and 21



(c) using dimension

Fig. 1: The Gaussian dataset using K-means, plot using two dimensions out of 25 by random

2) Covtype

Preprocessing of Covtype dataset was conducted for simulating the experiments described in the original paper. The Covtype dataset we used in the experiments is available in the Sklearn library[7] with originally 581,012 data samples. For each forest cover type in the original dataset, we randomly selected 2160 samples so that the resulting dataset contains 15,120 total number of samples for experiment.

3) MNIST

MNIST data set can be reached using Keras[8] dataset. The dataset received is of dimension (60000,28,28). To fit the dataset to our model, we flattened the 2^{nd} and 3^{rd} dimension and converted it to size (60000,784).

4) Posture

The dataset for MoCap Hand Posture can be found in UCI Machine Learning Repository[9]. The dataset provided has 78,095 instances and 38 attributes for each posture. The raw data contains several question marks for each data, we replaced the question marks using number out of the range -100 to 100 so that our algorithms can measure the Euclidean distance between each data points.

IV. Methods

To evaluate the replicability of the original paper, we attempted to reproduce the deletion efficiency(amortized runtime) and clustering performance experiments of Q-k-means and DC-k-means algorithm introduced in the paper. To compose the experiments, similar setups from original papers were used. We referenced github code for the two algorithms[1], and implement the test benchmark ourselves using datasets found from sites, since they are not provided along. We also tested the effect of quantization granularity and tree width on deletion efficiency.

We simulated a stream of 1,000 deletion requests for each algorithm and each dataset. For each individual group, we took the average of 3 identical tests. Deletion of data points were carried out at uniformly random and without replacement. To satisfy each deletion request, we produced an intermediate model at each request. The next deletion would happen on the previous intermediate model. Deletion requests for the k-means++ baseline model requires re-training from scratch.

To measure deletion efficiency, we used wall clock time for completing the deletion stream above. For evaluating clustering performances, we adopted the error metric of optimization loss used in the original paper. Optimization loss demonstrates the sum of squared Euclidean distances from each data point to its nearest centroid.

A. Baseline

As a baseline, we used the canonical Lloyd's algorithm initialized by k-means++ seeding.

The Lloyd's algorithm aims to find evenly spaced sets of points. It repeatedly finds the centroid for each set in the partition and re-partition based on Euclidean spaces. K-means++ is an algorithm for choosing the initial values for the k-means clustering algorithm. It chooses the first cluster center at random and chooses the subsequent cluster centers with probability proportional to its squared distance from the point's closest existing cluster center.

B. Quantized k-means

Q-k-means is proposed as a quantized variant of Lloyd's algorithm. By quantizing the centroids, it is shown that deletion of data points causes centroids remain constant with high probability. Thus, an efficient deletion algorithm is proposed without re-computing centroids from scratch at each iteration. In scenario when the quantized centers change, which happens relatively infrequent, centroids are re-computed from scratch. With Q-k-means, we expect the amortized runtime to behave stronger than the baseline.

C. Divide-and-Conquer k-Means

Divide-and-Conquer k-means (DC-k-means) algorithm partitions the dataset into sub-problems and merges the results solved by sub-problems. DC-k-means requires initialization of a tree structure with width w and height h, where the original dataset is partitioned into each leaf in the tree. Smaller k-means problems are solved at each leaf and results are recursively merged till root. On deletion of a data point, only

[1]https://github.com/tginart/deletion-efficient-kmeans

the path from one leaf up to the root needs to be re-computed. With DC-k-means, we expect a faster deletion operation with decreased computation need.

V. Results

In this section we present the results yielded through conducting subsets of experiments described in the original paper. Detailed comparisons and key observations are narrated in the next section.

A. Statistical Performance Metrics

We summarize the key findings of experiments with Q-k-means, DC k-means algorithms in the following tables. K-means was used as baseline for measurement. Table 1 shows the statistical clustering performance of the proposed methods and the baseline, with the loss ratio as error metric. In Table 2, we reported the amortized runtime under deletion of 1000 data points in for our proposed methods and the baseline. We observed that both proposed algorithms demonstrate competitive performance comparing to the baseline in statistical clustering. The table also shows that both proposed algorithms yield levels of runtime speedup.

TABLE I: Loss Ratio

Dataset	k-means	Q-k-means	DC-k-means
Gaussian	1	0.992	1.001
MNIST	1	1.005	1.095
Covtype	1	1.062	1.029

TABLE II: Amortized Runtime in Online Deletion Benchmark (Train once + 1,000 Deletions)

Dataset	k-means	Q-k-means		DC-k-means	
	Runtime	Runtime	SpeedUp	Runtime	SpeedUp
Gaussian	34.811	0.104	334x	2.156	16.14x
MNIST	46.014	20.482	2.25x	2.0364	22.59x
Covtype	4.351	0.314	13.85x	0.307	14.17x

B. Effect of Quantization Granularity on Optimization Loss

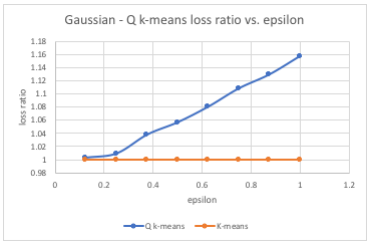
In this experiment, we compared the k-means optimization loss by changing epsilon and tree width, with the loss normalized with respect to the baseline. In Figure 2, we plotted the results derived from performing Q-k-means method with different epsilon values. We observed that loss ratio increased rapidly as epsilon value approaching 1.

C. Effect of Tree Width on Optimization Loss and Amortized Runtime

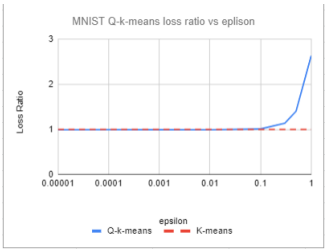
We experimented with the effect of varied tree width for DC-k-means.
We plotted the results in loss ratios derived from performing DC-k-means method with different tree width values.

We observed that the amortized runtime first decreased and then increased as tree width increased, which met the result of $Figure6$, $Appendix\ D.3.3$ in the original paper.

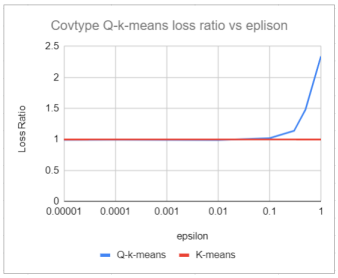
We found that the DC-k-means algorithm tended to yield better loss ratio as height increased.



(a) Gaussian



(b) MNIST



(c) Covtype

Fig. 2: Q-k-means Loss ratio v.s. epsilon

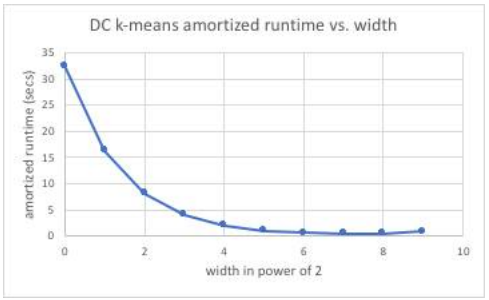
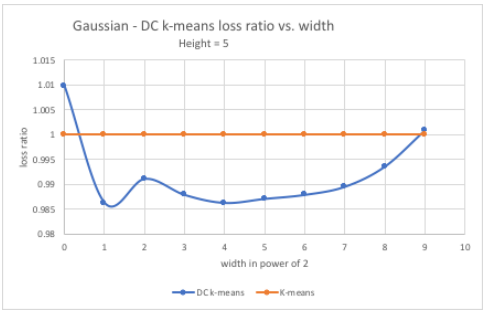


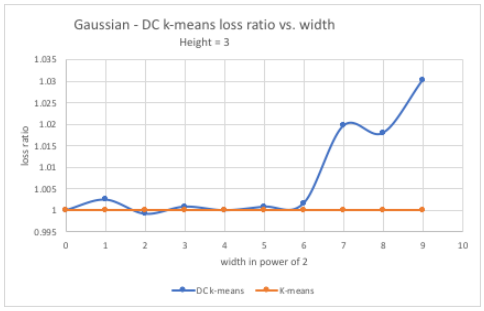
Fig. 3: DC-k-means Amortized Runtime vs width

D. Effect of Tree Height on DC-k-means Training Time

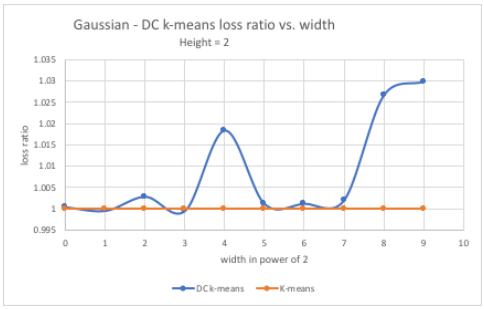
The original paper does not discuss the optimization of the DC tree respect to the tree height. We conducted this experiment out of our interest. Comparing training time for DC-k-means method of various tree heights, we found the training time increased with the tree depth when tree width value went above 16. However, no significant difference was



(a) Height = 5



(b) Height = 3



(c) Height = 2

Fig. 4: DC-k-means Loss ratio v.s. width

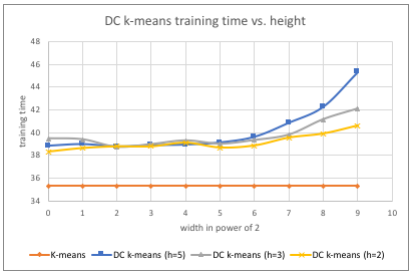
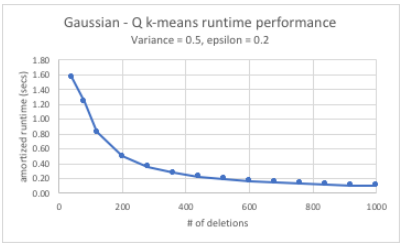


Fig. 5: DC-k-means Training Runtime vs height

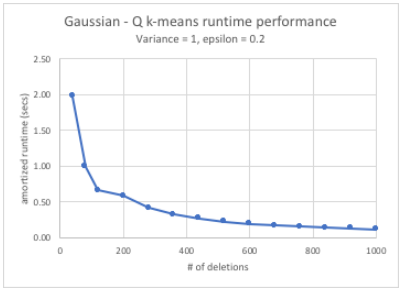
observed when tree width was sufficiently small.

E. Effect of Variance on Q-k-means Deletion Performance

We experimented with the effect of varied variance for Q-k-means. We plotted the results derived from performing Q-k-means method on the Gaussian dataset with different variances. We observed a smoother transition in the results derived with smaller variance.



(a) Variance = 0.5



(b) Variance = 1

Fig. 6: Q-k-means Delete Performance v.s. variance

VI. Conclusion and Discussion

Conclusion

In this report, we attempted to prove the reproducibility of Q-k-means and DC-k-means algorithms proposed by Ginart *et al.* by means of performing experiments on runtime efficiency and clustering performance.

In our experiment for loss ratio versus epsilon and width, we observed that the trends we derived were consistent with those demonstrated by $Figure\ 3$ and $Figure\ 4$ in the original paper. In the aspect of amortized runtime versus the two parameters, we observed that the time complexity reduced with lower epsilons and the point of highest efficiency lied around tree width of 30. Moreover, we tested how heights of DC-trees affected the time and loss. We concluded that having the height equal to 2 led to shortest runtime overall, corresponded to what was proposed in the original paper.

Thus we demonstrate that, as stated in the original paper, the proposed algorithms have competitive statistical clustering performance compared to the k-means baseline. While the results vary for different datasets, the proposed methods generally reduce deletion runtime to a high degree.

Discussion

In terms of reproducibility, the experiments presented in the original paper is easy to follow and implement. Although not affecting understanding, there is a minor error labeling the horizontal axis of $figure\ 4$. It should be "width" instead of epsilon.

Overall, we focused on the main parts of original results, and we left the mathematical proof of theoretical runtime of algorithms for further exploration.

For the databases used in the original report, there are some points that could be justified more clearly for the ease of reproduction. First, although the Celltype dataset is referenced, we were not able to find that data resource from the original paper and thus we did not use Celltype dataset as part of experiment. Second, Gaussian dataset misses variance value for each cluster of data. This value has been added in the latest version of the paper. However, Gaussian dataset with different variance would have various distribution, that could potentially change the retrain frequency of the proposed algorithms, resulting in incompatible deletion efficiency.

REFERENCES

[1] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making ai forget you: Data deletion in machine learning," 2019.
[2] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.
[3] D. Coomans and D. L. Massart, "Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-nearest neighbour classification by using alternative voting rules," 1982.
[4] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, Aug. 2014.
[5] G. Guo, "MCA DGE Data," 10 2018.
[6] Y. Meidan *et al.*, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
[7] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[8] F. Chollet *et al.*, "Keras." https://keras.io, 2015.
[9] D. Dua and C. Graff, "UCI machine learning repository," 2017.