NeurIPS 2019 Reproducibility Challenge Report on Ginart et al's "Making AI Forget You: Data Deletion in Machine Learning"

Jenisha Patel **Edward Son** **Li Zhang**

Abstract

In the context of the NeurIPS 2019 Reproducibility Challenge's Baselines Track, the baseline algorithm of Ginart et al.'s "Making AI Forget You: Data Deletion in Machine Learning" was re-implemented. Ginart et al. proposed two algorithms to support efficient data deletion for k-means clustering. Because the baseline was Lloyd's algorithm initialized by k-means++ seeding (k-means), we proceeded to re-implement the baseline. We also implemented three additional baselines, including bisecting k-means , weighted k-means and Gaussian mixture model. Among our implemented baselines, our Lloyd's algorithm initialized by k-means++ performed the best. It achieved similar clustering quality as the baseline implemented by Ginart et al. However, our implementation of the baseline lead to lower amortized runtimes, which can be attributed to a more computationally optimal implementation.

1 Background

Numerous machine learning models, particularly those pertaining to healthcare applications, are trained using data from individuals. One of the main ethical and legal issues that arise is the right of an individual to control the use of their data. In fact, an individual may request for their data to not be used anymore for a machine learning model. That request may be legally supported by laws such as the European Union's General Data Protection Regulation (GDPR), also known as the "Right to be forgotten". If a machine model is already trained, to respect the deletion request, the model would have to be retrained on the remaining data. Consequently, additional computational resources would be required to do so. However, retraining from scratch is not feasible or practical for many models requiring nonstop weeks worth of energy and computational expenditure [7]. Thus, efficient approaches to data deletion are required.

In "Making AI Forget You: Data Deletion in Machine Learning", Ginart et al. proposed two algorithms to support efficient data deletion for k-means clustering[7]. The baseline was Lloyd's algorithm initialized by k-means++ seeding (k-means). They proposed a quantized version of Lloyd's algorithm called Quantized k-means (Q-k-means): the centroids are quantized at each iteration such that each point is mapped to the nearest vertex of a uniform ϵ-lattice and the optimization state is memoized into the metadata of the model [7]. The metadata indicates whether the deletion of a specific datum would result in a different quantized centroid, which would require retraining from scratch [7]. They also proposed a another variant of Lloyd's algorithm called "Divide-and-Conquer k-Means" (DC-k-means) [7]. The algorithm consists of partitioning a dataset into subsets where each subset is trained as an independent k-means instance [7]. The resulting instances are merged [7]. Thus, a deletion of a datum would only require a sub-k-means instance to be retrained from scratch. In their paper, compared to the baseline , they empirically claimed that Q-k-means and DC-k-means produced clusters of similar statistical quality while leading to improvements of amortized times of over an 100-fold for five publicly available datasets and 1 simulated dataset.

Preprint. Under review.

1.1 Related Work

Sebastian Schelter addressed the data deletion problem by proposing ways to detrimentally update trained models [17]. The approach, which was implemented on collaborative filtering, ridge regression and k-nearest neighbors with locality sensitive hashing, lead to an average of one-fold improvement for ridge regression and an average of two-fold improvement for the other two aforementioned algorithms [17]. Similarly, Masayuki Karasuyama and Ichiro Takeuchi proposed a multiple decremental implementation of support vector machines (SVM) to add or delete multiple data points[10]. However, the aforementioned works specifically refer to deterministic learning algorithms. Ginart et. al specifically tried to address deletion efficiency for general learning algorithms, including stochastic models.

1.2 Task

The main goal of this report was to asses of the reproductibility of the experiments described in "Making AI Forget You: Data Deletion in Machine Learning", which was selected from a list of accepted 2019 Conference on Neural Information Processing Systems (NeurIPS) papers. Track 1 (Baselines Track) was chosen. The canonical baseline of the paper, Lloyd's algorithm initialized by k-means++ seeding, was re-implemented and fine-tuned. In addition, the following variants of k-means clustering were implemented and rigorously analyzed:

1. Bisecting K-Means
2. Weighed K-Means
3. Gaussian Mixture Models with Expectation Maximization

2 Methodology

2.1 Proposed Baselines

We re-implemented the baseline of the paper, which was Lloyd's algorithm initialized with k-means++. We also re-implemented three additional baselines, including bisecting k-means, weighed k-means and Gaussian mixture models with expectation maximization.

2.1.1 Lloyd's algorithm initialized with k-means++

k-means clustering is canonically done using Lloyd's algorithm, which was proposed by Stuart Lloyd in 1957 [12]. It consists of representing each cluster by a centroid and assigning each data point to the geometrically closest (smallest Euclidean distance) centroid. The first step of the algorithm consist of randomly assigning the centroids. The algorithm proceeds to iteratively update the cluster membership of each datum. During each iteration, the centroids are recomputed to be the center of the data in the corresponding cluster. The last two steps are repeated until convergence. While the algorithm is relatively straightforward to understand and implement, it converges to a local minima solution and it can therefore lead to poor clustering quality [9]. It also leads to suboptimal computation speeds [9]. Thus, in 2007, David Arthur and Sergei Vassilvitskii proposed a way to initialize the centroids in order to improve clustering quality while minimizing computation times [1]. The k-means++ initialization consists of first uniformly picking a first centroid from data [1]. The next centroids are picked from the remaining non-centroid data: the probability of each datum being picked is proportional to the squared distance to the closest centroid that has already been picked [1]. The k-means++ initialization guarantees an O(log k) approximation ratio [1]. The major hyperparameter to explore is the convergence criteria. Examples include a maximum number of iterations and setting the threshold for the minimum difference between subsequent iterations to continue.

2.1.2 Bisecting K-means

Bisecting K-means uses principles of hierarchical clustering to improve the clustering quality and the runtime of the canonical clustering algoritm [19]. The bisecting step of the algorithm consists of splitting a cluster into 2 [19] The bisecting step is repeated for a certain number of iterations and the

best split (best clustering quality) is returned. The two previous steps are repeated until the desired number of clusters is reached [19].

2.1.3 Weighed K-means

Weighted K-means takes into account the weights of each of the data points in the dataset[21]. That is, a datapoint with a bigger weight will contribute more heavily to the computation of each new centroid. At each centroid computation step, a weighted mean is used to calculated the new location of the centroid[21].

2.1.4 Gaussian Mixture Models with Expectation Maximization

The aforementioned K-means algorithms are hard clustering algorithms, meaning that each datum is associated to a single cluster without considering probabilities [20]. In contrast, Gaussian Mixture Model (GMM) is a soft clustering algorithm which considers different probabilities. GMM consist of representing the set of clusters with a weighed multivariate Gaussian distribution. Each cluster is represented by a Gaussian distribution and each Gaussian distribution has an associated weight. The parameters representing the Gaussian distribution of each cluster include the mean/mean matrix and the variance/covariance matrix. The maximum likelihood parameters are found using the expectation maximization algorithm, which was proposed in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin [4]. The algorithm can be initialized by randomly picking the mean/mean matrix and picking the variance/covariance matrix in a way to reflect the entire dataset. The algorithm consists of a expectation step and a maximization step. The expectation step is to compute the probability of each point being assigned to a cluster. The maximization step consists of changing the mean/mean matrix and variance/covariance matrix of the distribution representing each cluster in a way to maximize the probabilities of each datum being assigned to its corresponding cluster.

2.2 Datasets

Ginart and Zou ran their experiments on the following five publicly available datasets:

1. Celltype (N = 12,009, D = 10, K = 4), a dataset of mouse cell types [8]
2. Covtype (N = 15,120, D = 52, K = 7), a dataset of forest covertypes [2]
3. MNIST (N = 60,000, D = 784, K = 10), a dataset of handwritten digits [11]
4. Postures (N = 74,975, D = 15, K = 5), a dataset of static hand postures [5]
5. Botnet (N = 1,018,298, D = 115, K = 11), a dataset of raw network traffic data [14]

Each dataset was associated with a certain number of clusters, denoted by K. Each dataset had ground-truth labels, which were used to evaluate the statistical quality of each model. The datasets were normalized, such that each element was between 0 and 1. We evaluated our proposed baselines on those datasets.

2.3 Performance Metric

2.3.1 Evaluation of Amortized Runtime

Ginart et al. used a stream of 1000 deletion uniformly random requests to calculate the amortized runtime for each algorithm [7]. The baseline required re-training the model from scratch for each deletion operation. However, due to time constraints, we only used only a stream of 10 deletion operations.

2.4 Evaluation of Clustering Quality

Ginart et al. evaluated the quality of clustering via three criteria [7]:

1. Optimization loss of the k-means objective
2. Silhouette Coefficient
3. Normalized Mutual Information (NMI)

The optimization loss of the k-means objective correspond to the sum of the squared Euclidean distance from each data point to its nearest centroid [7]. The Silhouette Coefficient is a number between -1 and 1 that represents how dense and well-separated the clusters are: denser and better separated clusters give way to higher scores [7]. NMI is a number indicating how well each cluster matches the ground-truth labels while considering permutations: higher scores indicate higher correspondences between the assigned clusters and the ground truth labels [7].

3 Implementation Details

The aforementioned baselines were implemented on Python3, particularly using the Numpy library [15, 16]. Visualization was done using the matplotlib library [13]. The silhouette coefficient and NMI was evaluated using the sklearn library [18]. In addition, in order to accurately compare Ginart's implementations of k-means, Q-k-means and DC-means were compared to our implementations of various baselines, we also re-ran his codes[6]. A Google Compute Engine backend (12.72 GB of RAM), also known as "Google Collaboratory", was used to train and test the models. In contrast, Ginart et al. ran their experiments on a computer with an Intel Xeon E5-2640v4 processor [6]. For Q-k-means, the epsilon parameter was chosen for each dataset to be $2^{-log_{10}(n/(k*d^{1.5}))-3}$ while for DC-k-means, the depth parameter was chosen for each dataset to be $w = round(n**0.3))$ where n corresponds to the number of examples in the dataset, k corresponds to the number of clusters and d corresponds to the dimension of each feature in the dataset [3] .

4 Results

4.1 Determination of Optimal Hyperparameters

4.1.1 Maximum number of iterations

One of the most important hyperparameters in the k-means algorithm is the maximum number of iterations, which dictates how many times it will compute the centroids before converging. This hyperparameter was tuned on the baseline k-means with each dataset, and evaluated using the loss metric. Table 1 shows a decrease in loss in most datasets as the maximum iterations is increased from 5 to 20. Namely the MNIST dataset improved the most with a reduction in loss of 0.15. In order to get the best metrics, a max iteration hyperparameter of 20 was used in the subsequent experiments.

Table 1: Loss of k-means using different numbers of maximum iterations

Dataset	**5 iterations**	**10 iterations**	**20 iterations**
Botnet	0.18 ± 0.01	0.17 ± 0.00	0.18 ± 0.01
Celltype	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
Covtype	1.01 ± 0.03	1.01 ± 0.04	0.992 ± 0.03
MNIST	39.65 ± 0.26	39.52 ± 0.19	39.40 ± 0.17
Postures	0.20 ± 0.00	0.20 ± 0.00	0.21 ± 0.00

4.2 Comparison of Performance

4.2.1 Amortized Runtime

Table 2: The amortized runtimes were computed for Ginart's implementation of k-means, Q-k-means and DC-k-means

Dataset	**k-means [7]**	**Q-k-means [7]**	**DC-k-means [7]**
Botnet	61.9 ± 0.444	62.99 ± 33.56	78.7 ± 0.146
Celltype	5.08 ± 0.0201	0.512 ± 0.0226	0.892 ± 0.0204
Covtype	7.70 ± 0.0212	2.74 ± 0.384	1.26 ± 0.0824
MNIST	70.3 ± 0.109	22.89 ± 10.5	9.792 ± 0.0768
Postures	33.2 ± 0.192	14.603 ± 1.72	4.47 ± 0.393

Table 3: The amortized runtimes were computed our implementations of k-mean, bisecting k-means, weighed k-means and Gaussian mixture model

Dataset	k-means	b. k-means	w. k-means	GMM
Botnet	79.87 ± 5.39	484.57 ± 28.86	NA	NA
Celltype	1.34 ± 0.26	7.28 ± 0.30	5.15 ± 0.05	3.42 ± 0.00696
Covtype	0.93 ± 0.04	4.63 ± 0.14	6.23 ± 0.11	133.70 ± 0.0599
MNIST	17.49 ± 1.46	103.82 ± 1.01	NA	NA
Postures	16.40 ± 5.21	50.20 ± 1.39	33.57 ± 0.05	NA

4.2.2 Clustering Quality

The following experiments were run in triplicates

Optimization loss of the k-means objective

Table 4: The optimization losses of the k-means objective were computed for Ginart's implementation of k-means, Q-k-means and DC-k-means

Dataset	**k-means [7]**	**Q-k-means [7]**	**DC-k-means [7]**
Botnet	0.216 ± 0.0275	0.214 ± 0.0118	0.219 ± 0.0354
Celltype	0.0159 ± 000282	0.0182 ± 0.00168	0.0254 ± 0.000563
Covtype	0.994 ± 0.0180	1.05 ± 0.0340	1.00 ± 0.0276
MNIST	39.8 ± 0.0981	43.7 ± 0.252	39.9 ± 0.0572
Postures	0.208 ± 0.00173	0.211 ± 0.00382	0.210 ± 0.000567

Table 5: The optimization losses of the k-means objective were computed for our implementations of k-mean, bisecting k-means, weighed k-means and Gaussian mixture model

Dataset	k-means	b. k-means	w. k-means	GMM
Botnet	0.175 ± 0.00544	NA	0.233 ± 0.0703	NA
Celltype	0.0157 ± 0.000111	0.01 ± 0.00	0.016 ± 0.0005	0.0247 ± 0.000653
Covtype	0.992 ± 0.0306	1.12 ± 0.03	1.005 ± 0.0153	NA
MNIST	39.4 ± 0.169	NA	39.338 ± 0.0940	NA
Postures	0.206 ± 0.0066	NA	0.207 ± 0.0011	0.258 ± 0.00769

Silhouette Coefficient

Table 6: The silhouette coefficients were computed for Ginart's implementation of k-means, Q-k-means and DC-k-means

Dataset	**k-means [7]**	**Q-k-means [7]**	**DC-k-means [7]**
Botnet	0.608 ± 0.00940	0.585 ± 0.0315	0.647 ± 0.0151
Celltype	0.391 ± 0.00966	0.411 ± 0.0194	0.484 ± 0.0267
Covtype	0.210 ± 0.004914	0.239 ± 0.0308	0.183 ± 0.0122
MNIST	0.0665 ± 0.00196	0.0512 ± 0.00722	0.0695 ± 0.00314
Postures	0.0990 ± 0.00246	0.101 ± 0.00367	0.106 ± 0.000730

Table 7: The silhouette coefficients were computed for our implementations of k-mean, bisecting k-means, weighed k-means and Gaussian mixture mode

Dataset	k-means	b. k-means	w. k-means	GMM
Botnet	0.641 ± 0.0316	NA	0.591 ± 0.0298	NA
Celltype	0.390 ± 0.00761	-0.01 ± 0.00	0.375 ± 0.0258	0.151 ± 0.0361
Covtype	0.203 ± 0.00306	-0.05 ± 0.01	0.205 ± 0.0144	0.0126 ± 0.0129
MNIST	0.0611 ± 0.010	NA	0.061 ± 0.0034	NA
Postures	0.105 ± 0.00498	NA	0.111 ± 0.0016	0.0320 ± 0.0200

Normalized Mutual Information (NMI)

Table 8: The NMIs were computed for Ginart's implementation of k-means, Q-k-means and DC-k-meansl

Dataset	**k-means [7]**	**Q-k-means [7]**	**DC-k-means [7]**
Botnet	0.704 ± 0.0297	0.692 ± 0.00992	0.704 ± 0.0297
Celltype	0.356 ± 0.00645	$0.307 \pm .0704$	0.356 ± 0.00646
Covtype	0.307 ± 0.0174	0.330 ± 0.0267	0.307 ± 0.0174
MNIST	0.473 ± 0.0122	0.454 ± 0.0203	$0.473 \pm .0122$
Postures	0.163 ± 0.0138	0.158 ± 0.0111	0.163 ± 0.0138

Table 9: The NMIs were compared for our implementations of k-mean, bisecting k-means, weighed k-means and Gaussian mixture model

Dataset	k-means	b. k-means	w. k-means	GMM
Botnet	0.734 ± 0.0204	NA	0.691 ± 0.0278	NA
Celltype	0.351 ± 0.0132	0.00 ± 0.00	0.357 ± 0.0348	0.407 ± 0.0625
Covtype	0.350 ± 0.00887	0.02 ± 0.01	0.312 ± 0.0158	0.0884 ± 0.0385
MNIST	0.487 ± 0.0161	NA	0.493 ± 0.0032	NA
Postures	0.167 ± 0.0114	NA	0.166 ± 0.0115	0.243 ± 0.00690

5 Discussion

The code provided by Ginart et al. was first explored in order to determine whether his code would produce similar results and in order to obtain values obtained from the same source of computation as our implemented baselines. The speedups were not as drastic as those reported by Ginart et al. However, the specific parameters used to generate those performance values were not reported. In addition, Ginart et al. used 1000 deletion operations whereas we only used 10 deletion operations because of time constraints. The reported clustering qualities are similar for all the metrics (losses, NMIs and silhouette coefficients).

Comparing Tables 2, with the exception of the Botnet dataset, our implementation of the baseline achieved lower amortized run times for the deletion operation. This can be mainly attributed to optimized code implementation. With respect to hyperparameter tuning, we explored some of it for a single dataset, but as stated in Ginart et al's paper[6], hyperparameter tuning may often be dependent on the dataset, as the parameters may be optimal for some datasets but not others. For this reason, the reproducibility challenge was performed with fixed hyperparameters to keep the variance to a minimum.

Our implementation of the Gaussian mixed model lead to singular matrices. In addition, it involved unstable numerical computation values, which were handled by forcing values. Despite extensive (several days-worth) of debugging, the numerical instability errors remained. Thus, it was unable to cluster the MNIST and the Botnet dataset. Similarly, our bisecting k-means was unable to compute the label association of each point in a computationally tractable way.

Comparing the performance of all of our baselines, our implementation of Lloyd's algorithm with k-means ++ initialization generally lead to best clustering quality and the lowest runtime.

6 Conclusion

The baseline set by Ginart et al.[6] for the k-means algorithm with k-means++ initialization was reproducible when considering the loss, the silhouette coefficient and the NMI. In fact, we were able to achieve similar metrics for the various datasets. The original implementation of the baseline k-means algorithm with k-means++ initialization was very barebone and transparent, which allowed us to reproduce it with confidence. Using various variation of the barebone k-means algorithm with k-means++ initialization, we explored the possible variations of the algorithm and its effect on the metrics used to evaluate the model. Future directions include optimizing our Gaussian mixed model to be more numerically stable as well as doing an extensive ablation study on the hyperparameters of DC-k-means and Q-k-means.

7 References

[1] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.

[2] Jock A Blackard and Denis J Dean. "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables". In: *Computers and electronics in agriculture* 24.3 (1999), pp. 131–151.

[3] DebanujNayak. *DebanujNayak/Reproducibility-Neurips-Code*. URL: `https://github.com/DebanujNayak/Reproducibility-Neurips-Code/blob/master/efficient%20deletion.ipynb`.

[4] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[5] Andrew Gardner et al. "3d hand posture recognition from small unlabeled point sets". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2014, pp. 164–169.

[6] Antonio Ginart. *Deletion-efficient-kmeans*. Nov. 2019. URL: `https://github.com/tginart/deletion-efficient-kmeans`.

[7] Antonio Ginart et al. “Making AI forget you: Data deletion in machine learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 3513–3526.

[8] Xiaoping Han et al. “Mapping the mouse cell atlas by microwell-seq”. In: *Cell* 172.5 (2018), pp. 1091–1107.

[9] Tapas Kanungo et al. “An efficient k-means clustering algorithm: Analysis and implementation”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2002), pp. 881–892.

[10] Masayuki Karasuyama and Ichiro Takeuchi. “Multiple incremental decremental learning of support vector machines”. In: *IEEE Transactions on Neural Networks* 21.7 (2010), pp. 1048–1059.

[11] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[12] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[13] *Matplotlib*. URL: `https://matplotlib.org/`.

[14] Yair Meidan et al. “N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders”. In: *IEEE Pervasive Computing* 17.3 (2018), pp. 12–22.

[15] *NumPy¶*. URL: `https://numpy.org/`.

[16] *Python*. URL: `https://www.python.org/`.

[17] Sebastian Schelter. *“Amnesia”–Towards Machine Learning Models That Can Forget User Data Very Fast*.

[18] *scitkit-learn*. URL: `https://scikit-learn.org/stable/.`.

[19] Michael Steinbach, George Karypis, Vipin Kumar, et al. “A comparison of document clustering techniques”. In: *KDD workshop on text mining*. Vol. 400. 1. Boston. 2000, pp. 525–526.

[20] Domenico Talia, Paolo Trunfio, and Fabrizio Marozzo. “Introduction to Data Mining”. In: *Data Analysis in the Cloud, 1st ed, Elsevier*. 2016, pp. 1–25.

[21] *Unsupervised learning with weighted k-means*. URL: `https://medium.com/@dey.mallika/unsupervised-learning-with-weighted-k-means-3828b708d75d`.