Federated K-Means Clustering

Richard Qiu, Daniel Rodrigues, Catherine Zeng, Bryan Lee
Harvard University
{rqiu, rodriguesmd, catherinezeng, bryanlee}@college.harvard.edu

***Abstract*—The federated learning setting, where models are trained from online distributed data, is a growing area of research due to its widespread applicability. Past works focused on federated supervised learning, and did not provide algorithms for federated unsupervised learning. In our work, we adapt the k-means clustering algorithm specifically for the federated setting, and thoroughly evaluate its performance compared to traditional k-means clustering as well as distributed versions of k-means. Our novel Federated K-means Clustering algorithm shows robustness against the key challenges of the federated setting: non-uniformly distributed data, dynamic data across rounds, and unreliable device participation.**

***Index Terms*—Federated learning, distributed clustering, k-means**

I. Introduction

Federated learning refers to the training of a centralized model from data distributed across a large number of clients ("devices"). Unlike previous settings, the federated setting entails unreliable communication between the central server and devices as well as dynamic online data. One standard example of federated learning is training an image classifier using the local images on cell phones, but federated learning is also applicable to the medical context, where data is gathered from different hospitals.

Thus far, the work on federated learning is limited to supervised learning. We introduce an algorithm for federated unsupervised learning by adapting the k-means algorithm into a federated k-means algorithm. An important possible use case is the clustering of patients across different hospitals based on their medical data. We have two main contributions:

Federated K-means Clustering: We introduce two versions of a novel Federated K-means Clustering algorithm specifically adapted to handle unreliable communication and dynamic data. Because the data distribution could change over time, our algorithm features dynamic k-finding, which changes the number of clusters k across time.

Thorough Experimental Analysis: We thoroughly investigate Federated K-means Clustering by evaluating its performance in response to factors specific to the federated setting, such as data bias, non-homogeneous cluster partitioning, and various forms of data change over time. Using two datasets, we compare our algorithm against the tradition k-means clustering and two distributed versions of k-means clustering.

II. Background and Related Work

A. Federated Context

Federated learning was introduced in 2016 by by McMahan et al. [8]. Specifically, federated learning trains models through repeated rounds; in each round, successfully connected devices perform local training and report back to the central server, which then updates a global model that seeds the next round of local device training. Its applications are explored in training models from data distributed on different clients, such as cell phones and hospitals [23] [9]. There has been much work analyzing federated learning for supervised learning algorithms [8] [10] [11] [12] [13] [14], but to our knowledge, our work is the first to analyze federated unsupervised learning.

B. k-means Clustering

The k-means algorithm is among the most widely used centroid-based clustering algorithms [21]. Briefly, the basic k-means algorithm works by iteratively optimizing a set of centroids from random initialization. It does this in two steps: (1) assign each data point in the dataset to the nearest centroid, and (2), move each centroid to the euclidean mean of all points assigned to that centroid.

C. Distributed k-means

Numerous efforts have been made to distribute the computation of k-means for large distributed datasets. Below, we broadly summarize their efforts into two groups: centralized MapReduce k-means and decentralized gossip k-means.

MapReduce k-means: The largest class of centralized distributed extensions to the k-means algorithm follow the MapReduce paradigm. Broadly, they work by uniformly distributing the dataset to many workers. Each worker then locally runs basic k-means to find locally optimal centroids, and then these centroids are communicated to a central process and perform an aggregation step across local centroids to determine the final clustering.

Gossip k-means: In many large distributed systems, the global synchronization required of MapReduce k-means is not feasible. To this end, the k-means algorithm has been augmented with gossip protocols in order to decentralize the computation and preserve the privacy of localized datasets. Briefly, these algorithms work by having each node in the network computing basic k-means centroids on local data and then communicating and aggregating local results with a limited set of "neighbors", dissipating the information through the community of nodes over time by averaging local centroids with neighbors' centroids.

However, this both the centralized MapReduce and decentralized gossip paradigms are incompatible with the federated context. Both paradigms assume that the data are distributed uniformly over all nodes, the data is static with time, and that

all nodes are online at the time the algorithm is run. Further, MapReduce k-means makes no privacy guarantees for local data, while gossip k-means cannot independently determine the number of clusters in the combined dataset.

Nevertheless, the MapReduce k-means paradigm serves as an important launching point for federated k-means, providing a general method of aggregating cluster information from distributed data and shifting the burden of finding the optimal k from each worker to the aggregation step, where a more complete picture of the dataset has been formulated. Conversely, the gossip k-means protocol places explicit bounds on peer-to-peer communication to maintain privacy standards.

In our work, we adapt the k-means algorithm explicitly for the federated context and show robustness against non-uniformly distributed data, dynamic data across rounds, and unreliable device participation, while maintaining stronger privacy standards than gossip k-means.

III. The Federated k-means Algorithm

We adapt the k-means algorithm for the federated context. We describe the procedure for a given device D_j which is selected to participate in a given federated round. The device holds a (possibly time-dependent) subset $X_j(t)$ of the collective data $X(t) = \bigcup_i X_i(t)$.

A. Local Clustering

At the beginning of round t, the global cluster centers $M'(t-1)$ are communicated with each device in the round. Each device D_j then runs the standard k-means clustering algorithm on the local data subset X_j to finding local cluster centers $M_j(t)$, but initiating k' candidate cluster centers using the k-means-parallel [3] seeding algorithm, with seeding provided by the global cluster centers $M'(t-1)$, and where k' is an arbitrarily large number.

B. Centralized Meta-Clustering

These local cluster centers are aggregated on the central server to form a meta-dataset of communicated local clusters. We propose two paradigms for creating the meta-paradigm:

Server One-Shot: where $M(t) = \bigcup_i M_i(t)$ (i.e. the meta-clustering is performed over only the current round of communication).

Server Keep: where $M(t) = \bigcup_t \bigcup_i M_i(t)$ (i.e. the meta-clustering is performed over all past communications). *Server Keep* may be preferred over *Server One-Shot* in contexts where each federated round may not be representative or contain disappearing or seasonal data, since this allows for propagation of information distilled in prior rounds into the present round. This propagation comes at the cost of increased computation, storage, and potential privacy concerns.

In both paradigms, meta-clustering is performed on $M(t)$ using a clustering algorithm of choice with k-finding (in our experiments, we used k-means with standard silhouette scoring to find k). The resulting meta-cluster centers $M'(t)$ are stored on the central server for communication to devices in the next federated round. We note that the choice of k' is not tied to the value of the number of clusters we wish to infer, because we leave the task of finding k (the number of clusters across the collective dataset X) to the meta-clustering algorithm.

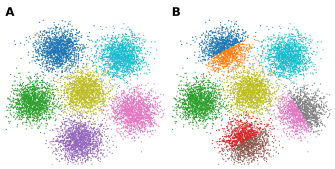

Fig. 1: Synthetic experimental datasets. (A) Dataset and labels used in all experiments. (B) Data partitions used for non-homogeneous partition experiment.

C. On-Device Cluster Inference

Now we propose a straightforward method of performing local cluster inference on a given device. After any given federated round, $M'(t)$ represents the best collective estimate of the true cluster centers over the collective dataset $X(t)$. As such, clusters present on any given device can be identified using the same global cluster center communication at the beginning of each federated round, with communication of the existence and/or prevalence of each cluster on each device.

IV. Experiments

We compare the proposed federated algorithms, Server One-Shot and Server Keep, against multiple baseline algorithms: a traditional (i.e non-federated) k-means algorithm, a centralized distributed k-means algorithm (from [7]), and a decentralized distributed k-means algorithm (Gossip k-means clustering algorithm from [21]).[1]

The proposed federated algorithms were tested in a simulated federated environment constructed with one server that communicated with a group of 12 devices that were randomly sampled from a population of 120 devices every round for 5-36 rounds depending on the experiment.

A. Baseline Experiment

In this experiment, we establish baselines for each of the 5 algorithms and provide a comparison in a simple scenario where the data between devices is static (does not change between rounds) and uniformly independent identically distributed (IID).

B. Bias of Devices

Second, we evaluate the algorithms with different bias levels. We establish four tests of varying levels of bias. Both the traditional k-means algorithm and the distributed centralized algorithm do not have experience bias, because they operate on the entire dataset. The two proposed federated algorithms and Gossip k-means do experience this bias as both depend on on-device calculations. Bias manifests as the non-IIDness of the data distribution between devices. A set of devices defined with $n\%$ bias is constructed by assigning each device a collection of data points where $n\%$ is sampled form a specified

[1] https://github.com/Danieltech99/CS-242-Clustering

group for that device and $100\% - n\%$ (the rest of the data) is sampled uniformly from the entire dataset (any duplicates data points when merging the two samples are removed).

C. Non-Homogeneous Cluster Partitioning

Third, we review each algorithm's ability to detect clusters across devices. Each device is assigned a disjoint partition, where a partition is one cluster (e.g. green points in the dataset) or a contiguous subset of a cluster (e.g. orange points in the dataset); importantly, no device has data from a partition other than their own, meaning there is no overlap in data between devices that have different partitions. This experiment examines the ability for each algorithm to recognize when multiple partitions belong to same cluster and identify common clusters across devices. We also include the traditional k-means and distributed centralized algorithms as a comparative baseline; since these two have access to the entire dataset, this metric is used to compare with non-partitioned performance.

D. Dynamic Distribution of Data

Lastly, we explore the unique opportunities and challenges of federated environments. Specifically, the ability for data on the devices to change over time leading to a substantial change in the distribution of each device as well as the collective dataset, $X(t)$ (data velocity/time dependence).

Crowd Discover: We use this test to evaluate each algorithm's ability to adapt to new data and detect when new clusters arise universally across all devices. We define "crowd" to be all devices. To simulate the introduction of new data in the device datasets and thus the collective dataset, $X(t)$, we begin with each device, in the crowd, having data only from the same cluster, such that the collective dataset, $X(t)$ contains data points from only this one cluster. Over the span of 3 rounds, every device gains data points from the same new cluster and no data points are removed. As a result, between rounds 2-4, every data point in each device and every point in the collective dataset, $X(t)$ is from two clusters. This continues such that by round 5-7, every data point is from 3 clusters; by round 16-18 all clusters are represented by data points in the collective dataset, $X(t)$.

Crowd Replacement: This test evaluates each algorithm's ability to adapt and detect new clusters as data is added, changed, or deleted. Importantly, this test isolates an algorithm's ability to correctly classify clusters even after the contributing data or represented data points have either been removed from the devices or is not being reported. To simulate this, each data cluster is numbered randomly from 1-6, then each device starts with data from cluster 1. Then over the span of three rounds, all devices transition linearly to having data only from cluster 2. This results in every data point in each device and every point in the collective dataset, $X(t)$ being from a second cluster at round 4. This continues such that at round 7, every data point is from a third cluster; by round 16 all data points are from a the last cluster. Because of the linear transition, each device and the collective dataset, $X(t)$ have data points from at most 2 clusters at any point in time. At the end of each 3 round phase, every device and the collective dataset, $X(t)$ have points only from one cluster. This scenario also ensures that once a device removes all the points from a cluster, those points and that cluster are never represented again in that device; once all points are for a cluster are removed from the collective dataset, $X(t)$, that cluster is never represented again in any device.

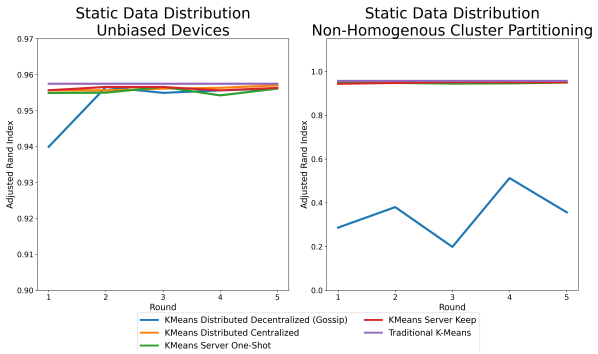

Fig. 2: Baseline benchmarks of all clustering algorithms using a uniform i.i.d. static distribution of data on each federated device.

Fig. 3: ARI of all algorithms with a static dataset with circular clusters, a subset are split in half into two sub-clusters (i.e. non-homogenous). This tests the central server in recombining sub-clusters into one primary cluster.

Subset Discover: This test investigates each algorithm's ability to detect new clusters when only a subset of devices report the new cluster. This test importantly reflects when a new cluster arises but is not present in the majority of devices. To design this scenario, we used a setup and progression similar to **Crowd Discover**. In this scenario, a fixed subset of devices are allowed to progress through the timeline established in **Crowd Discover**, however the rest of the devices (those not in the subset) are not allowed to progress and only have data from the same one cluster for all rounds. Specifically, each algorithm is tested in its ability to detect 5 additional clusters when only a subset are reporting the emergence of these 5 new clusters over time and while the rest of the devices are only aware of and report the one original cluster.

V. Results and Discussion

In all results, we compare the clustering algorithms using the widely used adjusted Rand index (ARI) [24] because our synthetic dataset provides ground truth classes and the ARI makes no assumptions on cluster structure.

A. Baseline Experiment

Figure 2 shows that in a simple IID federated environment, the two proposed federated algorithms perform as well as the traditional k-means algorithm and as the distributed centralized k-means and gossip (distributed decentralized). We use these results to validate our direction.

B. Bias of Devices

Figure 4 shows that the algorithms achieve near-ideal ARIs within a single federated round (or equivalent) with the excep-

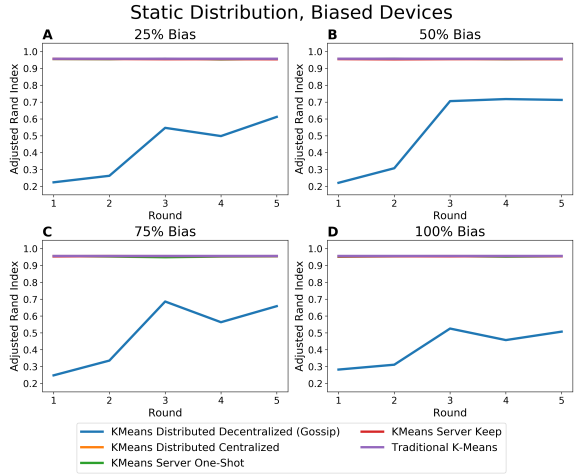

Fig. 4: Adjusted Rand Index (ARI) obtained over several federated rounds using our algorithm versus other distributed and non-distributed k-means algorithms on varying levels of device bias.

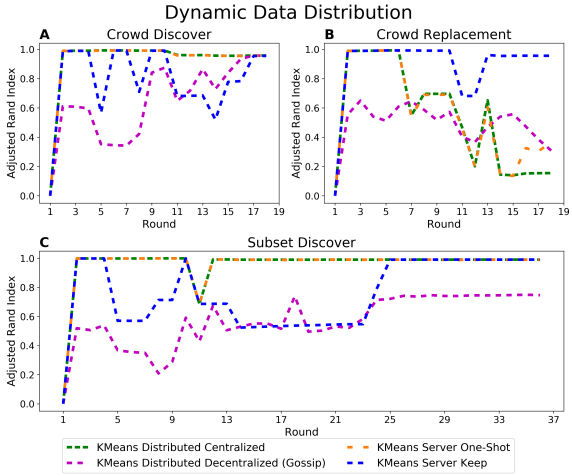

Fig. 5: ARI of the federated and distributed clustering algorithms over dynamic underlying datasets, reflective of online long-term federated environments.

tion of gossip k-means, which achieves a comparatively poor maximum ARI of below 0.8 across all bias levels.

C. Non-Homogeneous Cluster Partitioning

Figure 3 shows that in a federated setting where devices have disjoint subsets of data, even disjoint subsets of clusters, the proposed algorithms are able to detect when multiple partitions belong to the same cluster, even when compared to algorithms with the full dataset. The proposed algorithms can detect that these 9 reported partitions make up 6 clusters and can assign each partition to the correct cluster. Gossip reacts poorly in these circumstances and does comparatively worse in classifying the clusters.

D. Dynamic Distribution of Data

Crowd Discover: Figure 5(a) shows that the proposed federated *Server One-Shot* k-means algorithm performs as well as the distributed centralized k-means algorithm. The results also show that the proposed Server Keep federated algorithm is able to match this accuracy but requires a several round learning period, as demonstrated by the brief drops in ARI following with the introduction of new clusters. The gossip algorithm does poorly at any point when the collective dataset distribution does not match the pre-assumed gossip distribution. In this scenario, the distribution is changing and only reaches the assumed distribution at round 16.

Crowd Replacement: Figure 5(b) demonstrates that the proposed federated *Server Keep* algorithm, is the only algorithm of the 4 that performs well in this scenario, and it performs almost perfectly. All other algorithms perform worse as more data is deleted.

Subset Discover: Figure 5(c) shows that even at the end when the true distribution approaches the assumed distribution for the Gossip algorithm, the Gossip algorithm struggles to learn the new clusters. The results show that both proposed algorithms are able to detect new clusters, even when only a subset of devices discover a new cluster (which is reflective of real-world federated scenarios), and they perform as well as the distributed centralized k-means algorithm. Again we see a learning period in the proposed federated *Server Keep* algorithm.

VI. Conclusion

In this work, we proposed a novel federated clustering algorithm for k-means using centralized meta-clustering and on-device cluster inference. We benchmarked the federated algorithm against traditional k-means as well as standard distributed centralized and gossip learning extensions.

The results of this paper show that the proposed algorithms are effective tools in clustering in both a non-federated and federated environment. These algorithms provide storage and computation benefits in efficiency while maintaining accuracy on the level of traditional k-means. Our algorithm can also serve as a distributed centralized algorithm and provides the benefits just mentioned. Notably, our algorithm provides an effective and accurate algorithm to infer k (the number of clusters in a dataset) and correctly generate a model to query cluster membership when the clustering points are scattered across devices over time. Our algorithm can adapt to new data even when it only appears in a subset of devices. The *Server Keep* algorithm can is robust to data point removal and inconsistent device round participation. Both algorithms achieve this while operating under the same privacy restrictions of gossip k-means. This opens new applications in the realm of federated learning as clustering is now able to be done on a data level rather than a device level.

Extensions include investigating analogs of other clustering algorithms, such as SOM and DBSCAN. In the future, this work can be applied to cases in which clustering privacy-sensitive data sets with specialized models are desired, such as patient-specific models for hospitals.

REFERENCES

[1] Vrahatis, et al., "The New k-Windows Algorithm for Improving the k-means Clustering Algorithm" (2002) Journal of Complexity

[2] Lu, et al., "Differentially Private k-means Clustering with Guaranteed Convergence" (2020) arXiv preprint.

[3] Bahmani, et al., "Scalable k-means++". (2012) Proceedings of the VLDB Endowment.

[4] Wittek, et al., "Somuclu: An Efficient Parallel Library for Self-Organizing Maps". (2017) Journal of Statistical Software.

[5] Li Huang, et al., "Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records". (2019) Journal of Biomedical Informatics.

[6] Comiter, M., Cha, M., Kung, H. T., Teerapittayanon, S. (2016, December). Lambda means clustering: automatic parameter search and distributed computing implementation. In 2016 23rd international conference on pattern recognition (ICPR) (pp. 2331-2337). IEEE.

[7] Hai, M., Zhang, S., Zhu, L., Wang, Y. (2012, August). A survey of distributed clustering algorithms. In 2012 International Conference on Industrial Control and Electronics Engineering (pp. 1142-1145). IEEE.

[8] McMahan, H. B., Moore, E., Ramage, D., Hampson, S. (2016). Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629.

[9] Liu, D., Dligach, D., Miller, T. (2019). Two-stage federated phenotyping and patient representation learning. arXiv preprint arXiv:1908.05596.

[10] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... Van Overveldt, T. (2019). Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046.

[11] Yang, Q., Liu, Y., Chen, T., Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1-19.

[12] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492.

[13] Smith, V., Chiang, C. K., Sanjabi, M., Talwalkar, A. S. (2017). Federated multi-task learning. In Advances in Neural Information Processing Systems (pp. 4424-4434).

[14] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V. (2018). Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.

[15] Kargupta, H., Huang, W., Sivakumar, K., Johnson, E. (2001). Distributed clustering using collective principal component analysis. Knowledge and Information Systems, 3(4), 422-448.

[16] Younis, O., Fahmy, S. (2004). HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Transactions on mobile computing, 3(4), 366-379.

[17] Merugu, S., Ghosh, J. (2003, November). Privacy-preserving distributed clustering using generative models. In Third IEEE International Conference on Data Mining (pp. 211-218). IEEE.

[18] Taheri, H., Neamatollahi, P., Younis, O. M., Naghibzadeh, S., Yaghmaee, M. H. (2012). An energy-aware distributed clustering protocol in wireless sensor networks using fuzzy logic. Ad Hoc Networks, 10(7), 1469-1481.

[19] Bendechache, M., Kechadi, M. T. (2015, July). Distributed clustering algorithm for spatial data mining. In 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM) (pp. 60-65). IEEE.

[20] Hamerly, G., Elkan, C. (2004). Learning the k in k-means. In Advances in neural information processing systems (pp. 281-288).

[21] Di Fatta, G., Blasa, F., Cafiero, S., Fortino, G. (2011, December). Epidemic k-means clustering. In 2011 IEEE 11th international conference on data mining workshops (pp. 151-158). IEEE.

[22] Fellus, J., Picard, D., Gosselin, P. H. (2013, December). Decentralized k-means using randomized Gossip protocols for clustering large datasets. In 2013 IEEE 13th International Conference on Data Mining Workshops (pp. 599-606). IEEE.

[23] McMahan, B., Ramage, D. (2017). Federated learning: Collaborative machine learning without centralized training data. Google Research Blog, 3.

[24] Rand, W. M. (1971). Objective Criteria for the Evaluation of Clustering Methods. Journal of the American statistical Association. 846-850