Federated Learning through Distance-Based Clustering

Phani Rohith M, Rajitha Muthukrishnan, Mostafa El Katerji, Ibrahim Helmy

December 16, 2020

Abstract

Federated Learning (FL) is a system architecture that leverages distributed networks such as mobile phones to enhance AI model personalization and performance while keeping personal data confidential to its owner. In this paper, we aim to propose a novel addition to the traditional FL design that leverages clustering to group similar devices together and enhance their collaboration with one another during training.

1. Introduction

Federated learning is a paradigm where a distributed system of devices is set up in a way where its devices can collaborate with each other to train a model. Traditional federated learning involves having a centralized server that contains model weights with devices that contribute to the training of that model by occasionally sending their weights back to the server. When the weights get sent back to the server, all devices are given an equal chance to update the main model using a process called Federated Averaging (FedAvg). Once FedAvg is applied, it is used to update the centralized model, and then the updated weights are broadcasted to all the devices in the network.

The problem with the traditional approach is that it relies heavily on the assumption all devices will learn in a similar manner. However, realistically speaking, devices would be operating with distinct sets of data each representing a subset of the overall model representing by FL. Therefore, averaging device weights is unlikely to be the most optimal approach as it disregards biases that arise from unbalanced data across devices.

Therefore, in our approach, we look at the problem differently. We see that instead of having the central model only, we can introduce the concept of clustering that represents groups of different devices that exhibit similar learning behavior. By doing this kind of clustering, we claim that algorithms such as FedAvg applied on a cluster-level should emphasise the weight updates that matter the most for the devices that are within them.

In this paper, we carry out a three-phase experiment to explore our clustering method. The following is a summary of what we carried out:

- Phase 1
  - Split a target dataset across multiple devices
  - Train all devices using a traditional FL learning method
- Phase 2
  - Based on the weights of the devices after Phase 2, we run two clustering algorithms. The clustering algorithms output a list of clusters, with each cluster having a list of device ids belonging to it. Each cluster gets its own weights computed for it by averaging the weights of all the devices within it.
  - Train the devices further but instead of reporting their weights back to the central server, they report them to their cluster only. Therefore, this essentially creates multiple traditional FL networks where each network represents one cluster and the subset of devices belonging to that cluster
- Phase 3
  - We calculate multiple permutations of the weights generated after Phase 2 is complete and then use it to evaluate the improvement of the model after clustering is applied.

2. Federated Learning

Machine learning algorithms, especially deep learning algorithms, are always in need of a large dataset to train effectively. In real-world scenarios, the data is spread across multiple organizations under privacy restrictions (Hjerppe et al., 2019). Furthermore, there are many sectors where the data cannot be shared internally such as government, finance, hospitals, and research labs, where internal collaboration is impossible due to data regulations and policies. However, if these entities are given the ability to cooperate while keeping their data anonymous, we can expose our model to a rich and diverse dataset.

To accommodate for such restrictions, a solution was needed for collaborative learning without the exchange of the user's data. That solution was proposed by combining federation with machine learning, and it was called federated learning (FL). In general machine learning, the data is present on a single device or at a data center and these algorithms use this data and build a model upon it. Whereas, in FL, the server sends the model to the individual devices in its network and trains on data present on each device. After the training is completed, only model parameters are returned while the data stays on the device that owns it. As a consequence, the server aggregates all the clients' results and performs aggregation using FedAvg (Segal et al., 2017). There are chances that the model behaviour can be determined using the model parameters. To overcome this problem, a secure aggregation (Elkordy & Avestimehr, 2020) can be computed where the model parameters obtained from the clients are encrypted, and the server model can decrypt the aggregation. Here is an overview of the FL training process.

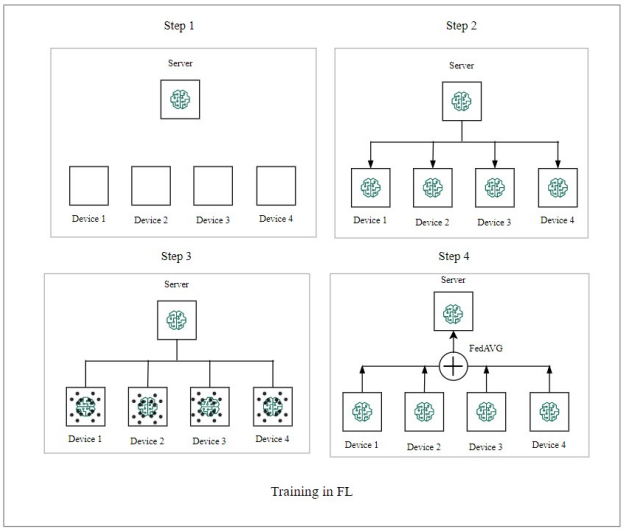

Figure 1. Traditional FL

An FL model typically carries out four steps. In the first step, a machine learning model is selected on the centralized server. In the second step, this model is propagated to all the devices in the network. In the third step, each device starts training the model and starts having their own unique weights that reflect their own data. Finally, in step 4, they devices send their weights back to the server where they go through federated averaging before the result is broadcasted to all devices. These 4 steps count as one round of FL training.

For our project, we have used TensorFlow Federated (Bonawitz et al., 2019), which is a machine learning framework that allows computation on decentralized data. This open-source framework allows users to research and build the FL architecture-based model.

3. Previous Work

3.1. Paper 1

Research paper "Federated Learning: Strategies for Improving Communication Efficiency" (Konečný et al., 2017) investigates methods that can reduce the uplink communication costs. This paper targets to reduce the uplink communication cost during federated learning and proposes two different solutions. Structured and Sketched Updates.

Structured Updates is where learning happens directly on an update from a restricted space parametrized using a smaller number of variables. For example, a low-rank or a random mask.

Sketched Updates is where learning is done through a full model update and then compressed using a combination of quantization, random rotations, and subsampling before sending it to the server.

3.2. Paper 2

The paper "Federated learning with hierarchical clustering of local updates to improve training on non-IID data" (Briggs et al., 2020) focuses on improving training on non-IID data while reducing the communication cost.

A new technique of federated learning is introduced where a hierarchical clustering step is introduced after r communication rounds during the traditional Federated training process. Clustering of clients is based on the similarity between the clients. The local updated weights are used to compute the similarity. This paper uses an Agglomerative hierarchical clustering technique with a distance threshold to determine the end of the clustering process. Once the clients are clustered, they are initialized with the global model at the current state. Federated Learning then proceeds for each cluster independently for a total of 50 communication rounds. This approach has helped the clients to attain benchmark test accuracy with fewer communication rounds.

This paper is related to ours in the authors' use of clustering. A more detailed overview of the comparison between their approach and ours will be provided in a later section.

4. Our Framework

Our framework has been constructed with the comparison of our proposed multi-phased model and a vanilla FL model in mind. Consequently, these two models are the two components that form our framework in our study. Both models are configured to run over E epochs, and contain 100 devices within their network. Each device in the network contains a

random distribution of the dataset.

4.1. Datasets

For the dataset, 10-fold cross validation was used, with the metrics being recall and accuracy.

4.1.1. EMNIST

TensorFlow EMNIST dataset is used. It contains 60,000 training examples and 10,000 testing examples. Tensorflow federated provides ways to distribute the dataset to multiple clients, which is used to split the dataset among clients in this implementation.

5. Clustering

An essential part of our proposed model is the clustering of the devices. It allows for devices to benefit from an added layer of collaboration from devices with similar learning traits to them. For example, assume that the EMNIST dataset was being used for training, two devices can likely have a great deal of experience learning how to identify the number 5. By sharing their weights they can ideally help each other learn at a faster rate. Clustering takes place during the second phase of our model, and plays a large role in the training conducted in both the second and third phases. In our study, we tested two methods for clustering.

5.1. K-Means Clustering

The K-means algorithm (scikit learn, 2020) is a well established method for creating clusters of a certain set of input points. The algorithm takes alongside these input points, an input integer K. K represents the number of clusters to be formed.

5.1.1. Algorithm

In the given algorithm 1, let X represent the set of input points.

Algorithm 1 kMeansClustering

Input: data x, number of clusters k
Randomly place centroids c_1,...,c_k across x
repeat
  Initialize $m = len(x)$
  for $i = 0$ **to** m **do**
    Find the nearest centroid C_j: arg min D(x_i, C_j) {Euclidean Distance}
    Assign the point x_i to cluster j
  end for
  for $j = 1$ **to** k **do**
    Centroid C_j = mean of all points x in cluster j
  end for
until None of the cluster assignments change

5.1.2. Key Characteristics

There are a few important characteristics of this method of clustering. The first pertains to the number of clusters a device can be included in. With K-means clustering, a device can only be a part of one cluster at a time. Another important characteristic is the number of the clusters, the K variable, which is a predefined integer. This means that the number of clusters cannot grow or shrink dynamically, and that this integer will have to be a fine-tuned hyper-parameter that ideally will be a different value depending on the dataset used and the number of devices.

5.1.3. Updating Device Weights

Every Device in a cluster receives an update at certain points during model training. For this method of clustering, each device's weights are set to be the average of all the weights of every device sharing its cluster.

5.2. Dynamic Distance-based Clustering

The second method for clustering is the Dynamic Distance-based clustering method. For this method, we perform multiple operations in order to achieve the final list of clusters. Firstly, the Euclidean distance between all devices is computed based on their weights. Secondly, a threshold value is determined. This threshold value serves to be the maximum distance between two different devices contained within a cluster. Once this value is selected the algorithm begins the clustering process. For our study, the threshold value was set to be the average of all the weights in the cluster. The algorithm 2 describes this in further detail.

5.2.1. Algorithm

Algorithm 2 DynamicDistance-BasedClustering

Input: data x
Initialize $m = len(x)$
Initialize an empty $DeviceDistanceMap$
for i **in** x **do**
  Compute Distance between x_i and all other devices
  Add computed distance to $DeviceDistanceMap$
end for
Initialize $threshold$ to the average of the client distances
Initialize an empty $ClusterMap$
for i **in** x **do**
  if x_i in $ClusterMap$ **then**
    Add x_i to $ClusterMap$
  else
    for all other devices j **do**
      if distance between x_i and $x_j <= threshold$ **then**
        Add x_j to $ClusterMap[x_i]$
      end if
    end for
  end if
end for

5.2.2. Example

For example, let's take this Python dictionary as the map of a device and all of the other devices that fall within the threshold distance previously defined. The Keys in a python dictionary are the values preceding the ':' and the values are the elements after it. With reference to the example below, '1' is a key and '[9,10]' is its value.

{0: [], 1: [9, 10], 2: [9], 3: [], 4: [], 5: [9], 6: [], 7: [], 8: [], 9: [1, 2, 5], 10: [1]}

In the example here, the keys of the dictionary are the device Ids, and the value is a list of device IDs for devices that are at most threshold distance away from the device. Using algorithm 2, the resulting list of clusters will be as follows:

[[5, 9], [9, 1], [10, 1], [2, 9], [0], [3], [4], [6], [7], [8]]

5.2.3. Key Characteristics

As can be seen in example given previously, unlike K-Means clustering, the results show that a single device can be a part of multiple clusters. This is a unique property that will allow a device to collaborate with other devices that share multiple learning traits to it, without the restriction of them being non-similar to each other. It is also important to note that the algorithm for this method will not allow for subsets as well, despite allowing a device to be a part of multiple clusters. Devices that also fail to be a part of a group cluster, form a cluster of their own.

5.2.4. Updating Device Weights

Since a single device can be a part of multiple clusters using this method, the device weights are computed individually rather than in clusters. Each device's weights are updated to be the average of all weight averages, for each cluster it is a part of. Using the cluster list from the example 5.2.2 the update for device 9 would be as such:

$Device9\_newWeights$ = Avg(Avg([5, 9]), Avg([9, 1]), Avg([2, 9]))

6. Phases

6.1. Phase 1 - Initialization

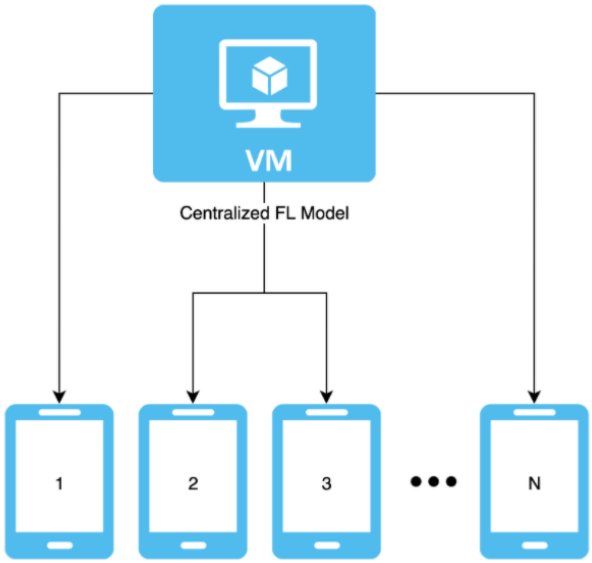

Figure 2. Phase 1

This phase serves the purpose of initializing our experiment. It is composed of a traditional FL model where N devices train over e epochs and collaborate with each other through FedAvg on the centralized FL model. The Centralized FL model would broadcast weight updates to all devices as well during the training.

After the e epochs are completed, each device would have unique weights that are both as a result of the collaboration with other devices and a good fit for the device's own dataset.

We keep track of the individual device weights for Phase 2 and we evaluate the general performance of this model.

6.2. Phase 2 - Clustered Training

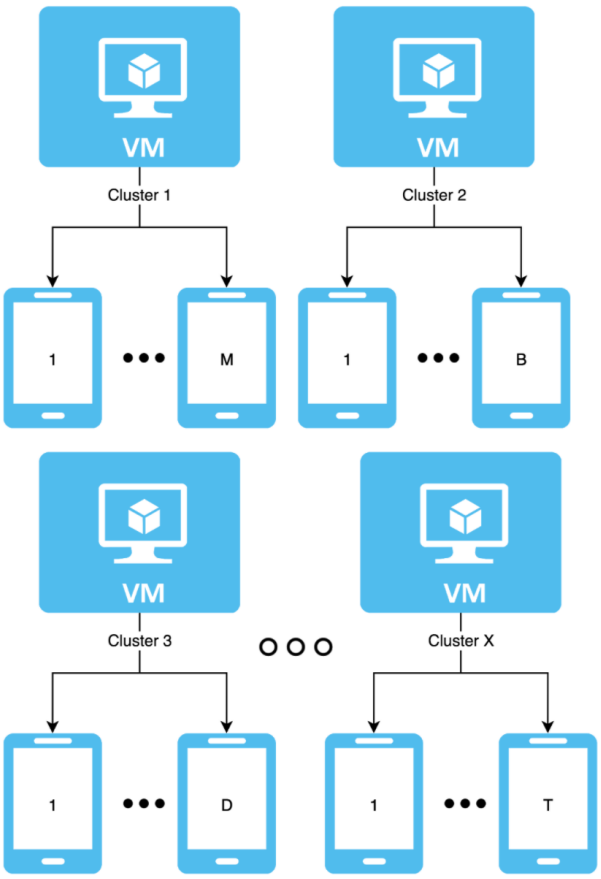

Figure 3. Phase 2

In this phase, we introduce our contribution in Clustered Training.

Using the individual device weights from Phase 2, we cluster them using K-means and distance-based clustering. For each clustering algorithm, we get a different set of clusters of devices.

We then treat every cluster of devices as its own FL model where the cluster's central model is initialized with the cluster's average device weights. After that, for every cluster, we train the devices in it for e epochs which eventually results in updated central cluster weights.

As a result, clustering encourages devices that train similarly in Phase 1 to strictly collaborate with one another. This is advantageous as compared to the traditional FL training where irrelevant devices contribute weights to each other which may hurt their performance on their respective datasets.

6.3. Phase 3 - Evaluation

In phase 3, we evaluate the effectiveness of our clustering methods explored in Phase 2 by evaluating the final weights that are derived from it.

- We evaluate the new weights in three different ways:
  - How well does the average of all weights perform on a generalized test set?
  - How well does the weighted average of all weights perform on a generalized test set?
    - Each cluster weights are weighted using their highest attained accuracy
  - If we were to update each cluster weights by taking the average between its weights and the average of all clusters, would each cluster still retain its specialization but also attain better generalizability by benefiting from other clusters?
    - This evaluation uses each of the cluster's devices' test sets

7. Results & Comparison

7.1. Evaluation Metrics

The metrics used for the evaluation are accuracy and precision. The TFF's Federated learning API layer provides high-level interfaces that can be directly used in TensorFlow models. However, it only returns an accuracy score and obtaining precision value is not possible with this API layer. To obtain the recall score, TFF's Federated Core API layer can be used as a base on which FL was built. It provides a set of lower-level interfaces as it allows users to perform computations by altering the functional programming environment.

A test set is split for the global model and unique test sets are distributed across every client. Evaluation of the global model and the local updated models are performed with their unique test sets. The following evaluations are performed on test sets and results are recorded:

- Accuracy of the clients and the global model (Phase 1)
- Accuracy of the global model after cluster averages update (Phase 3)
- Accuracy of the global model after the weighted clusters average update (Phase 3)
- Accuracy of the clients after clusters average update (Phase 3)

Results of the model with both K-means and Dynamic distance-based clustering are also reported.

7.2. Contrast to Previous Work

The hierarchical clustering technique used in (Briggs et al., 2020) experiments with different factors like the number of communication rounds before clustering, different non-iid settings, distance metrics like L1, L2, cosine distances.

The number of clients participating, communication rounds, test accuracy of the clients, percentage of the clients that attain the target test accuracy are recorded in (Briggs et al., 2020). It is to be noted that the global model's performance after the clustering step is not reported.

Following are the novelties presented in our paper:

- Clustering techniques - K-means and Dynamic distance clustering are used.
- All clients participate in the learning process.
- The local models are not initialized with the current global state after clustering. Instead, the weights that were used for clustering are used as such for further cluster-based training.
- Clusters are trained independently without any feedback to the global model.
- Phase 3 is introduced in this paper, where the global model is updated in two different ways - cluster average and weighted cluster average. Clients are updated with the average of the cluster weights it belongs to and all other clusters.

A convolutional neural network is used for the central server and the clients to train on the EMNIST dataset. Data is distributed to 100 clients with non-iid settings. Phase 1 is executed for 20 epochs, followed by Phase 2 clustered training for 20 epochs. SGD optimizer is used with a learning rate of 0.1.

In the proposed methodology, Euclidean distance is used to compute similarity, and device to device distance is computed. Hence, the results of L2 and single linkage from (Briggs et al., 2020) are considered for a fair comparison. Since the highest test accuracy achieved by the clients and the percentage of clients achieving it are reported in (Briggs et al., 2020) the same is reported for the proposed methodology. Table 1 depicts the results of (Briggs et al., 2020) with L2 distance, single linkage settings. The following are the results observed from the implementation of the proposed methodology.

7.2.1. K-means Clustering

- Phase 1:
  - The accuracy of the global model on the test set is 92.5%
- Phase 3 (Post cluster):
  - Test set accuracy of the global model
    - Updated with cluster average is 93.4%
    - Updated with a weighted cluster average is 93.4%
  - 26% of the clients have reached 99.6% accuracy when updated with average weights of all the clusters

7.2.2. Dynamic Clustering

- Phase 1:
  - The accuracy of the global model on the test set is 91%
- Phase 3 (Post cluster):
  - Test set accuracy of the global model
    - Updated with cluster average is 93%
    - Updated with a weighted cluster average is 93%
  - 28% of the clients have reached 99.7% accuracy when updated with average weights of all the clusters

K-means and Dynamic clustering were executed in two different runs. Table 2 depicts the highest accuracy achieved by the clients with the proposed method and the percentage of clients that have achieved the highest accuracy. Results of the global model with the proposed methodology are reported in Table 3.

From the results in Table 1 and Table 2, it is evident that the proposed methodology has managed to attain higher client test accuracy than the accuracy achieved in (Briggs et al., 2020). However, the percentage of clients that have achieved the highest test accuracy is pretty low.

Table 1. From (Briggs et al., 2020) results of clients using L2 Distance, Single linkage for clustering

	Post cluster + 50 communication rounds
Highest Test accuracy achieved	98.1%
% of clients with the highest test accuracy	67%

Table 2. Results of clients with the proposed methodology

Clustering algorithm		Post-Clustering
K-means clustering	Highest Test accuracy achieved % of clients with the highest test accuracy	99.6% 26%
Dynamic clustering	Highest Test accuracy achieved % of clients with the highest test accuracy	99.7% 28%

Table 3. Results of the global model with the proposed methodology using a global test dataset

Clustering Algorithm	Phase 1	Phase 3 (clusters average)	Phase 3 (weighted cluster average)
K-means clustering	92.5%	93.4%	93.4%
Dynamic clustering	91%	93%	93%

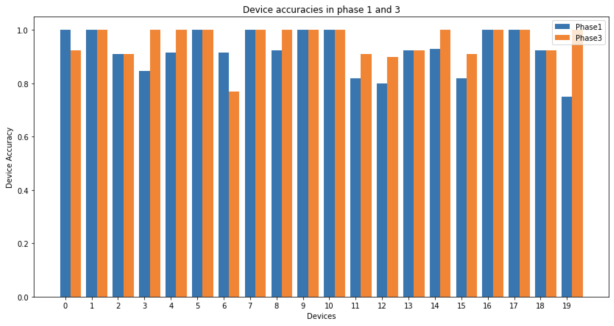

Figure 4. Test accuracies of 20 clients in Phase 1 and Phase 3(clusters average) with K-means clustering

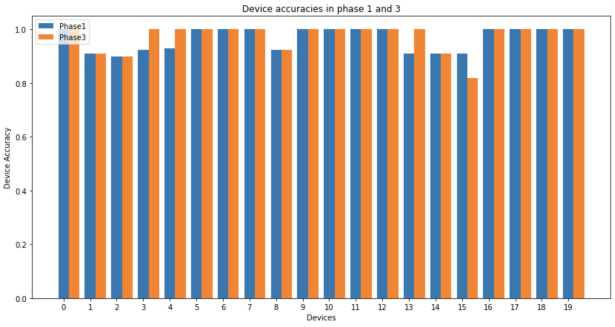

Figure 5. Test accuracies of 20 clients in Phase 1 and Phase 3(clusters average) with Dynamic clustering

It is evident from the results in Table 3 that dynamic clustering has given slightly better results on the global model. There is a difference of 0.9% between phase 1 and phase 3 with K-means clustering. With dynamic clustering, the difference between phase 1 and phase 3 is 2%. It is because, with dynamic clustering, the devices get to be a part of multiple clusters and can generalize better with similar devices' information.

Figure 4 and Figure 5 (K-means clustering and Dynamic clustering respectively) depicts the test accuracies of only 20 clients in phase 1 and phase 3(where client weights are updated with clusters average) for better visualization.

8. Conclusion and Future scope

Observed an overview of the training process in Federated Learning and provided how Baseline FL works and the comparison of it with the multi-phase approach with the two datasets. The later ones achieved better results than the baseline FL as clustering allows similar devices to train mutually. By doing this, similar devices are formed as clusters and irrelevant devices that affect their performance are separated. Since a single device can be part of multiple clusters, it yielded better results than the traditional clustering method, where a device can be part of a single cluster. Training clusters independently without any feedback to the global model yielded better results. FL is a hot research topic and extending the clustering methodology with confidence intervals or other derivatives from it will be a powerful research direction.

References

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... Roselander, J. (2019). *Towards federated learning at scale: System design.*

Briggs, C., Fan, Z., & Andras, P. (2020). *Federated learning with hierarchical clustering of local updates to improve training on non-iid data.*

Elkordy, A. R., & Avestimehr, A. S. (2020). *Secure aggregation with heterogeneous quantization in federated learning.*

Hjerppe, K., Ruohonen, J., & Leppanen, V. (2019, Sep). The general data protection regulation: Requirements, architectures, and constraints. *2019 IEEE 27th International Requirements Engineering Conference (RE)*. Retrieved from http://dx.doi.org/10.1109/RE.2019.00036 doi: 10.1109/re.2019.00036

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2017). *Federated learning: Strategies for improving communication efficiency.*

scikit learn. (2020). *2.3.2. k-means.* Retrieved from https://scikit-learn.org/stable/modules/clustering.html#k-means

Segal, A., Marcedone, A., Kreuter, B., Ramage, D., McMahan, H. B., Seth, K., ... Ivanov, V. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Ccs.* Retrieved from https://eprint.iacr.org/2017/281.pdf