Federated Learning on Medical Applications

Sizhuang He, Mentored by Professor Maria Han Veiga

August 2022

Abstract

In this REU project, we focus on a machine learning paradigm called federated learning and its applications on medical data. Federated learning was proposed as a communication efficient way of learning from decentralized data that is believed to protect data privacy [2].

We explored federated learning on I.I.D. and non-I.I.D. data. A defensive mechanism against corrupted client is put forward. Finally, we experimented federated learning on the BRAZPD Dataset, a medical dataset [5].

1 Introduction

Traditional machine learning frameworks require a centralized data center where all training data are stored. However, this may not be feasible in many applications where data privacy is emphasized. Thus, federated learning, a machine learning framework where all data never leave client devices and different clients jointly train a global machine learning model, is introduced [2].

In federated learning, there are generally two parties of interest, the clients and the server. Each of the clients hold a local training dataset, which is never uploaded to the server. Training takes place in a distributive manner over the clients. The server is a communication center that aggregates the local updates to the machine learning model computed by the clients [2].

For a typical machine learning problem, assume we are given a dataset S of n data points, the goal is to find model parameters $\omega \in \mathbb{R}^d$ to minimize a chosen objective function, also called the empirical risk, defined as

$$L : \mathbb{R}^d \to \mathbb{R}, \qquad L(\omega) = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, \omega) \tag{1}$$

where $\ell(x_i, y_i, \omega)$ is the loss on one data point $(x_i, y_i) \in S$, with model parameter ω.

In the setting of federated learning, the goal is also to minimize $L(\omega)$ but the data points are partitioned into K clients. We denote $\mathcal{P}_k$ as the set of indices of data points owned by client k. Let n_k be the size of $\mathcal{P}_k$ and $\sum_{k=1}^{K} n_k = n$. Then we have

$$L(\omega) = \sum_{k=1}^{K} \frac{n_k}{n} L_k(\omega) \qquad \text{where} \qquad L_k(\omega) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} \ell(x_i, y_i, \omega) \tag{2}$$

And some simple algebra shows the definition of $L(\omega)$ in Eq.(2) matches that in Eq.(1). $L_k(\omega)$ can be interpreted as the local empirical risk of client k, which is analogous to Eq.(1).

The partition $\mathcal{P}_k$ is worth more focus in the federated setting. If we form $\mathcal{P}_k$ by randomly distribute the data points to the clients, we have for each client k, $\mathbb{E}_{\mathcal{P}_k}[L_k(\omega)] = L(\omega)$, indicating each $L_k(\omega)$ is a good approximation of $L(\omega)$ [2]. This is referred to as the **I.I.D.** assumption [2]. In reality, data distribution may vary greatly across different clients and when above setting does not hold, we refer to it as the **non-I.I.D.** setting.

A very important optimization algorithm widely used in machine learning is the Stochastic Gradient Descent algorithm (SGD). It is natural to consider the local dataset of each client as a batch of data in the SGD setting. This gives rise to the `FederatedSGD` algorithm [2]. In `FederatedSGD`, in a global round t, each client k computes $g_k = \nabla L_k(\omega_t)$, the local gradient of the model parametrized by ω_t. Then the server aggregates all gradients and the update rule [2] is

$$\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \tag{3}$$

Equivalently, we can formulate the update rule in Eq.(3) in the following way

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} (\omega_t - \eta g_k) \tag{4}$$

Let $\omega_{t+1}^k = \omega_t - \eta g_k = \omega_t - \eta \nabla L_k(\omega_t)$, then the update rule becomes

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t^k \qquad \text{where} \qquad \omega_{t+1}^k = \omega_t - \eta \nabla L_k(\omega_t) \tag{5}$$

For each client k, ω_{t+1}^k can be viewed as the local model by doing one local gradient update to the global model ω_t with the client's local data. The server then aggregates the local models and compute a weighted average of them and this is equivalent to aggregating local gradients. Inspired by this, we can do more local computation during each training round since communication between clients and the server is the bottleneck in most cases [2]. This is termed as the `FederatedAveraging` algorithm (`FedAvg`) [2] as shown in Algorithm 1

One scenario where federated learning may be particularly useful is the medical settings. Medical data are usually scattered in a large number of clinics and

Algorithm 1: `FedAvg`. K: number of clients indexed by k, B: size of local minibatches, C: fraction of active clients, E: number of local epochs, η: learning rate

```
Server executes:
    initialize $\omega_0$
    for each round $t$ = 1, 2, ... do
        $s \leftarrow \max(C \cdot K, 1)$
        $S_t \leftarrow$ a random set of $s$ clients
        for each client $k \in S_t$ in parallel do
            $\omega_{t+1}^k \leftarrow$ ClientUpdate(k, $\omega_t$)
            $\omega_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$
        end for
    end for
ClientUpdate(k,$\omega$), for each client $k$
    $\mathcal{B} \leftarrow$ split $\mathcal{P}_k$ into small batches of size $B$
    for epoch $i$ from 1 to $E$ do
        for one batch of data $b \in \mathcal{B}$ do
            $\omega \leftarrow \omega - \eta \nabla \ell(\omega; b)$
        end for
    end for
    return $\omega$
```

hospitals and are highly privacy sensitive. Thus, training machine learning models on medical data is always limited by the availability of adequate datasets [4]. Federated learning can provide a way to learn decentralized data from a large number of clients without breaking data privacy. One example of such medical datasets is the BRAZPD [5].

The BRAZPD dataset was collected in a nationwide cohort study on peritoneal dialysis (PD) in Brazil from December 2004 to January 2011 [5]. A total of 9005 patients, 5707 of which remained in the study after 90 days, from 122 medical centers took part in this study. The researchers recorded patients' biomedical measurements, such as Potassium, Glucose, Phosphate, etc. on a monthly basis. The duration that each patient stayed in the study was also recorded, indicating the final outcome of the petiant since the main cause of dropping out the study is death (54%) [3].

2 Methods

In this REU project, we first explored federated learning on the Iris dataset. Then we applied the idea of federated learning to the BRAZPD dataset [5].

2.1 Experiments on the Iris Dataset

The Iris dataset is a commonly used public machine learning dataset for classification tasks [1].It contains 3 classes, Iris Setosa, Iris Versicolour and Iris Virginica, one class of which is linearly separable from the rest while the other 2 are not. Each class contains 50 instances. Each instance has four attributes and the goal is to classify the instance given the attributes [1].

First, we constructed two different data partition methods of the Iris Dataset, I.I.D. and non-I.I.D. and compared the performance of `FedAvg` on classifying iris instances.

Then we explored the scenario where certain clients may contain corrupted data, that is its data may be very noisy. This may be the result of an adversarial attack or very poor measurements and training on the noisy local dataset will result in a very badly-performing local model, which will harm the global model during the averaging process. It is thus important to set up a defensive mechanism to detect the presence of such corrupted clients and try to minimize the effect of their influence.

We modify the original `FedAvg` by adding a validation stage to each client after a local model is trained. The local datasets of each client is split randomly into a training set and a validation set according some predetermined ratio r. Immediately after each client finished training on its training set (the validation set is left untouched during training), the client validates its local model on its validation set and report metrics such as loss and accuracy to indicate how well the newly trained local model is doing on the validation set. Models trained on very noisy training sets are expected to perform very badly on the validation set because noises are considered to be purely random and not predictable. Then the server will adjust the weights of each client model according to the metrics reported. The algorithm is shown in Algorithm 2.

Algorithm 2: `FedAvg` with weight adjustments. K: number of clients indexed by k, B:size of local minibatches, C: fraction of active clients, E: number of local epochs,η: learning rate, r ratio to split the local datasets

```
Server executes:
    initialize $\omega_0$
    for each round $t$ = 1, 2, ... do
        $s \leftarrow \max(C \cdot K, 1)$
        $S_t \leftarrow$ a random set of $s$ clients
        $m \leftarrow$ empty array
        for each client $k \in S_t$ in parallel do
            $\omega_{t+1}^k, m_k \leftarrow$ ClientUpdate(k, $\omega_t$)
            $w_k \leftarrow$CalculateWeights($n_k$, $m_k$)
        end for
        $w = \sum_{k=1}^K \omega_{t+1} \leftarrow \sum_{k=1}^K \frac{w_k}{w} \omega_{t+1}^k$
    end for
ClientUpdate(k,$\omega$), for each client $k$
    $\mathcal{T}_k, \mathcal{V}_k \leftarrow$ split $\mathcal{P}_k$ according to $r$
    $\mathcal{B} \leftarrow$ split $\mathcal{T}_k$ into small batches of size $B$
    for epoch $i$ from 1 to $E$ do
        for one batch of data $b \in \mathcal{B}$ do
            $\omega \leftarrow \omega - \eta \nabla \ell(\omega; b)$
        end for
    end for
    $m_k \leftarrow$ LocalValidation($\omega$, $\mathcal{V}_k$)
    return $\omega, l_k, a_k$
LocalValidation($\omega$, $\mathcal{V}_k$), for each client $k$
    Test model $\omega$ on $\mathcal{V}_k$
    $m \leftarrow$ chosen metric
    return $m$
CalculateWeights($n_k$, $m_k$)
    return a computed weight
```

We experimented on two different `CalculateWeights`, apart from the original `FedAvg`. One way is to adjust according to validation loss and the other is according to validation accuracy.

No adjustments (`FedAvg`) The `CalculateWeights` function is as shown in Algorithm 3. There is no mechanism in detecting corrupted clients in `FedAvg`.

Algorithm 3: No adjustments

```
CalculateWeights($n_k$, $m_k$)
    return $n_k$
```

Adjust weights according to validation loss The `CalculateWeights` function is as shown in Algorithm 4.The weight of a certain client is proportional to its dataset size and inversely proportional to the validation loss of its local model, since badly-performing models tend to report a larger validation loss.

Algorithm 4: Adjust weights according to validation loss. m_k is the validation loss of client k

```
CalculateWeights($n_k$, $m_k$)
    return $\frac{n_k}{m_k}$
```

Adjust weights according to validation accuracy The `CalculateWeights` function is as shown in Algorithm 5. The weight of a certain client is proportional to its dataset size and the validation accuracy of its local model, since badly-performing models tend to report a smaller validation accuracy.

Algorithm 5: Adjust weights according to validation accuracy. m_k is the validation accuracy of client k

```
CalculateWeights($n_k$, $m_k$)
    return $n_k \cdot m_k$
```

2.2 Experiments on the BRAZPD Dataset

We further explored federated learning on the BRAZPD dataset. As described above, a total number of 122 clinics or hospitals took part in the study. This distributed nature of the BRAZPD dataset makes it suitable for federated learning.

The BRAZPD dataset includes 1000 features, including non-time-series features and time-series features. Time-series features are biochemical substances that are measured on a timely basis, such as blood glucose every month. Non-time-series features are ones that are only measured at the start the study. The duration for which a certain patient stays in the study is also recorded [5].

We are generally interested in predicting whether a patient stays in the study after one year from the beginning, given the baseline feature set. The baseline feature set consists of several non-time-series features, including age, region, BMI, etc. The complete baseline set is included in the appendix.

3 Result

3.1 Performance of `FedAvg` on I.I.D. and non-I.I.D. data

I.I.D. setting In the I.I.D. setting, we randomly partition the data points to 3 clients. Then we used local minibatch sizes $B = 10$, number of local epochs $E = 30$, number of global communication rounds $t = 30$, fraction of active clients $C = 1$. The cross entropy loss function is usd as the loss function. For each client model, we used a fully connected neural network with 2 hidden layers, each containing 200 neurons, (2NN). The FedAvg gives a test accuracy of 98.33%.

Non-I.I.D. setting In the non-I.I.D. setting, data points are partitioned into 3 clients with respect to the label. In this way, all data points of the same client belong to almost the same class and those of different clients belong to different classes. With the same set of hyper-parameters and model architectures as in the I.I.D. setting, the FedAvg also gives a test accuracy of 98.33%.

Researchers found empirically that FedAvg can still reach the same test accuracy in the non-I.I.D . setting as in the I.I.D. setting if trained for more communication rounds [2]. For example, with $C = 0.1, K = 100, B = 10, E = 1$ and 2NN on MNIST, a handwritten digit classification dataset, FedAvg reached a test accuracy of 97% within 86 communication rounds in the I.I.D. setting and 664 rounds in the non-I.I.D. setting [2]. We may conclude that when given enough computation power, FedAvg is considered to be robust to data partition. This explains our experimental result. Since the Iris dataset is relatively small, with only 150 instances while MNIST, which the researchers used, has 70000, 50 communication rounds is more than the threshold to produce good outcome even in the non-I.I.D setting.

3.2 Client weight adjustment

We are more interested in non-I.I.D. partition in this case because in reality the non-I.I.D. setting is more common. We used the non-I.I.D. data partition and the same set of hyper-parameters as described in Section 3.1. To simulate the existence of a corrupted client, one client is randomly chosen and noise that follows a Gaussian distribution of mean $\mu = 0$ and standard deviation $\sigma = 300$ is added to the local dataset of that client. This is considered as a relatively large noise since the origin data points are pre-processed so that most of them are of order 10^0.

The experimental results are shown in Table 1.

	FedAvg	Adjust with loss	Adjust with accuracy
Test Accuracy	38.33%	63.33%	70.00%

Table 1: Test Accuracy for different weight adjustment methods

Since FedAvg has no defense mechanism against corrupted clients, it only successfully classified 38.33% of the test dataset. This is dramatic decline in performance since FedAvg achieved a 98.33% test accuracy with the same set of hyper-parameters without corrupted clients. Both weight adjustment mechanisms improved the test accuracy greatly. Adjusting with accuracy outperforms adjusting with loss. One possible explanation is that the loss function is usually sensitive to outliers. Loss functions essentially measure how far a predicted label deviates from the true label. Therefore, it is possible that a few mislabeled data points contribute to a large proportion of the loss function. However, since we are dealing with a classification task here, what's more representative of a model's performance is the proportion of data points it correctly classifies.

3.3 Federated Learning on the BRAZPD dataset

To explore how federated learning performs on the BRAZPD dataset, we trained and tested federated learning models on three scenarios, referred to as grouping, non-grouping and no-FL. Each patient in the BRAZPD dataset are assigned an integer representing the clinic or hospital he/she belongs to [5]. In the grouping scenario, we group clinics to data centers and form ten small data centers with similar sizes by sorting the table with respect to the clinic index and split into ten groups, simulating data centers, in a balanced manner. Patient data are stored in data centers, where local training takes place. There is a server that communicates model parameters with the data centers. In the non-grouping scenario, patient data are stored within the clinics or hospitals. Local training and testing also takes place in the clinics and hospitals. A global server aggregates model parameters and coordinate the global training. The no-FL scenario refers to the centralized scenario where no federated learning is applied.

The way the dataset is split into the training set and test set may influence the performance of the machine learning model trained on the training set. Chances are that if the training dataset does not represent the entire data distribution good enough, the consequent machine learning model may perform badly. To minimize this, a five-fold cross validation process is adapted in the experiments.

In the five-fold cross validation training process, the entire dataset is split into five subsets, with balanced sizes. Then, five separate training processes take place. In each process, one of the five subsets is chosen as the test set and the rest combine to be the training set.A federated learning model is trained on the training set and tested on the test set. This ensures all data have at some time of the training been used as the test set and the mean F1 score of five models is calculated and adopted as a criterion of the learning process.

In each experiment, we use the 2NN model architecture as described in previous experiments. The number of local training epochs is chosen to be $E = 30$. The number of global communication rounds is set to be $t = 500$ since predicting on the BRAZPD dataset is considered quite complicated. Other hyperparameters include batch size $B = 10$ and learning rate $\eta = 0.01$.

The mean F1 score and standard deviations are as shown in Table 2. Detailed results are included in the appendix.

	Grouping	Non-grouping	No-FL
F1 score	0.578 ± 0.120	0.614 ± 0.122	0.633 ± 0.106

Table 2: FL results on the BRAZPD

Since only non-time-series features are taken into consideration in these experiments, the F1 scores are all considered acceptable. The case where no federated learning is applied reports the highest F1 score among the three. Applying federated learning makes the performance very slightly worse but still acceptable. The non-grouping case produces a slightly better F1 score. A possible explanation is that the grouping algorithm depends on the specific indexing of the clinics which may not be random enough.

4 Conclusion

In this REU project, we first tested `FedAvg` with I.I.D. and non-I.I.D. data partition on the Iris Dataset. The results indicates that if trained for more global rounds, `FedAvg` on non-I.I.D. data can return a model that reaches similar test accuracy with that on I.I.D. data. Then we put forward a defensive mechanism against corrupted client by validating local models on local validation datasets and adjusting the weights of each local models in the global model update. Finally, we explored federated learning on the BRAZPD Dataset. We find federated learning generally performs as good compared to traditional centralized learning.

A Appendix

Baseline features: 'CenterSizenpatients', 'Age', 'IncidentinPD', 'PrevalentinPDNet', 'Primaryrenaldisease', 'PreviousHD', 'Previoustx', 'DaviesScore', 'Peripheralarterydisease', 'Cancer', 'Stroke','Hypertension', 'HIV', 'HCV', 'HBC', 'Gender', 'Race', 'Familyincome', 'predialysiscare', 'timeofpredialysiscare', 'Educationdic4y', 'Region', 'Centerexperiencepatientyear', 'BMI', 'Distancefromcenter'

GitHub Repository https://github.com/SizhuangHe/FL-REU

References

[1] R. A. Fisher. “The Use of Multiple Measurements in Taxonomic Problems”. In: *Annals of Eugenics* 7.7 (1936), pp. 179–188.

[2] H. Brendan McMahan et al. “Federated Learning of Deep Networks using Model Averaging”. In: *CoRR* abs/1602.05629 (2016). arXiv: `1602.05629`. URL: `http://arxiv.org/abs/1602.05629`.

[3] Thyago Proença de Moraes et al. “Characterization of the Brazpd ii Cohort and Description of Trends in Peritoneal Dialysis Outcome across Time Periods”. In: *Peritoneal Dialysis International* 34.7 (2014). PMID: 25185014, pp. 714–723. DOI: `10.3747/pdi.2013.00282`. eprint: `https://doi.org/10.3747/pdi.2013.00282`. URL: `https://doi.org/10.3747/pdi.2013.00282`.

[4] Nicola Rieke et al. “The Future of Digital Health with Federated Learning”. In: *CoRR* abs/2003.08119 (2020). arXiv: `2003.08119`. URL: `https://arxiv.org/abs/2003.08119`.

[5] Neimar da Silva Fernandes et al. “The Brazilian Peritoneal Dialysis Multicenter Study (BRAZPD) : characterization of the cohort.” In: *Kidney international. Supplement* 108 (2008), S145–51.