UPPSALA
UNIVERSITET

Studying Data Distribution Dependencies In Federated Learning

Meenal Pathak, Mohamed Hussein

Project in Computational Science: Report

February 2020

PROJECT REPORT

Institutionen för informationsteknologi

Abstract

Federated Learning is a relatively new and promising machine learning approach, emerging as an alternative to conventional centralized learning. It is especially relevant in cases where data are confidential, large in quantity, or both. Unlike centralized machine learning, Federated Learning may involve data that are distributed among a large number of participants/clients in a non-IID (Independent and Identically Distributed) fashion.

We investigate the impact of data heterogeneity on the performance of a simulated Federated Learning environment, compared to a centralized model, and methods to address that. The algorithm implemented in our investigation is Federated Averaging, which is evaluated on both unbalanced and non-IID partitioning of two different types of datasets. Results indicate that Federated Averaging has a similar performance on both Federated Learning and a centralized model, when the data are IID-partitioned. The accuracy drop varies depending on the severity of the non-IID partitioning and can be mitigated by introducing certain modifications to the Federated Averaging Algorithm.

Acknowledgements

We would like to thank Dr. Prashant Singh and Dr. Salman Toor, the project supervisors for their constant help and guidance throughout every stage of the project, and their constructive feedback. The completion of this project would not have been possible without their support.

Mohamed Hussein
Meenal Pathak

Contents

1 Introduction

1.1 Background

With the rise of the Internet of Things (IoT) and smart phones, the number of end point devices hosting data has increased exponentially. This data, if assembled together, would be of significant importance in machine learning. A typical setting of machine learning involves gathering all the data from the edge devices to a central server, where all the learning takes place. This approach is often termed as centralized machine learning.

Centralized machine learning is often characterized by powerful computing resources and large volumes of data, which can lead to highly accurate models. However, centralized learning usually requires a huge storage capacity at one location, which also implies that a substantial amount of resources is needed to transfer data to the centralized server. Furthermore, the approach, by default, poses major privacy concerns when data are of a confidential nature. Federated Learning is an alternative approach that offers solutions to all of the aforementioned challenges.

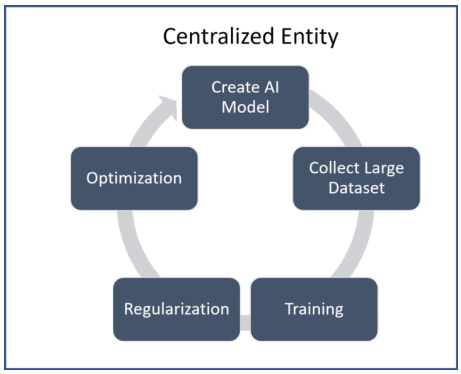


Figure 1: A typical centralised machine learning setting [7]

1.1.1 Federated Learning

Federated Learning is a decentralized machine learning approach that trains an algorithm at multiple sites holding local data sets, without exchanging data points between participants. Since all the training takes place locally, each client/edge device computes an update which is then uploaded to the main server. The main server's task is to aggregate and perform parameter averaging of these local updates. Federated learning is not to be confused with conventional distributed machine learning. While both

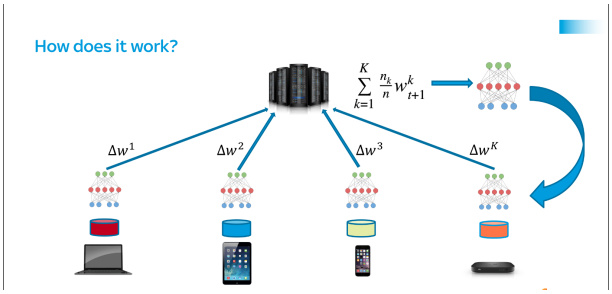


Figure 2: High-level scheme of Federated Learning [1]

approaches may share some similarities, Federated Learning differs from distributed machine learning in four key properties [4]:

- **Expensive Communication** There could potentially be millions of devices participating in Federated Learning. As it is an iterative learning approach, communication among the local sites and the

main server could be a challenge. Thus, an efficient communication system and robust algorithms need to be developed to speed up and minimize the magnitude of the uploaded updates.

- **System Heterogeneity** The sites participating in the learning process could have varying processing powers, storage capacities and availability of network bandwidth. Furthermore, some devices could drop out of the learning process mid-way and the updates from those devices would become unavailable.

- **Statistical Heterogeneity** The devices contributing to the training process may generate/collect non-independent and identically distributed data. Additionally, some of the sites may have large amounts of data points while others may have very few. Therefore, it is safe to assume that there would be sites with data that violate the assumption of independent and identical distribution (a.k.a. I.I.D. Data). This poses various challenges in optimization of the model, introducing errors and reducing accuracy of predictions.

- **Privacy Concerns** Federated Learning offers further protection (compared to a centralized model) by communicating only model's parameters (updates) to the main server instead of raw data. However, these updates carry sensitive information, i.e., data distribution, that can be revealed to a third-party. In the case where a large number of clients participate in training, classical privacy protocols like differential privacy need to be enhanced to meet the challenges without compromising model performance or communication efficiency.

1.2 Objectives

The main goal of this project is to explore the implications of statistical heterogeneity of the data in a Federated Learning setting. We particularly focus on the following three questions:

1. How do unbalanced datasets affect overall model training in a Federated Learning environment?

2. How do non-IID datasets affect model training in a Federated Learning environment?

3. How can the performance of models trained on unbalanced and non-IID datasets in a Federated Learning environment be improved?

2 Artificial Neural Networks

The machine learning method used in this project is Artificial Neural Networks (ANNs).

ANNs are processing/computing systems that are vaguely inspired by the architecture of the human brain, consisting of multiple layers of artificial neurons. The neurons in a layer are connected to the neurons in the preceding layer and the neurons in the next layer through trainable parameters (weights). The output of a neuron in a certain layer serves as an input to all the neurons in the next layer, as shown in Figure 3. Based on that input and its internal state, the neuron produces an output. This output depends upon a threshold level, known as the activation function. This process of connecting neurons is called ***Feedforward***. The layers between the input layer and the output layers are called ***hidden layers***.

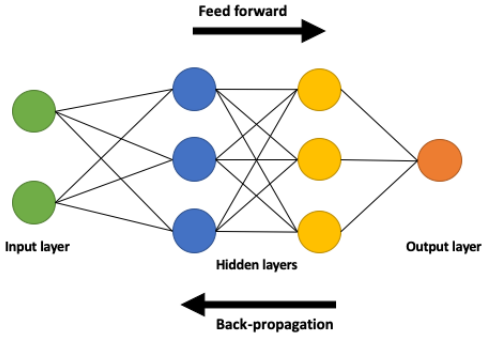


Figure 3: Architecture of a simple ANN with two hidden layers.

The quality of the neural network or how well it performs is measured in terms of a cost function of the network using labelled training data, i.e. data for which we already know the expected output. A cost function computes the error between predicted and expected values and quantifies it in one number. The training process is done through ***Backpropagation***, an algorithm that computes the gradient of the cost function with respect to the network weights for a single input-output pair.

There are some parameters associated with the training process whose values are set before initiating the learning process, often called hyperparameters. Examples of hyperparameters are the learning rate, number of epochs, batch size and number of hidden layers in a ANN.

An optimizer, associated with the neural network, is responsible for implementing Backpropagation for a specified number of iterations to get the best possible results from the network.

Finally, a model produced by the training process of the neural network is an equation where each of the inputs have a value associated with its own weight. The equation has the addition of the product of these inputs and the weights equating the output of the network. Thus, a model can be thought of as the pattern found out by the neural network in the training data. This model/equation can be applied on a new set of inputs to predict their possible outputs.

3 Theory

Here, we explain in more detail the concept of Federated Learning and provide a brief mathematical background for it. We also present a high-level description of the learning algorithm used in our investigation, Federated Averaging (FedAvg).

3.1 Federated Optimization

For a typical non-convex neural network, the aim is to optimize a finite-sum objective function

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w). \tag{1}$$

In a supervised learning context, $f_i(w) = \ell(x_i, y_i; w)$ is the loss of the prediction on input-output pair (x_i, y_i), one of n labeled samples, using model parameters w.

When computation and data are distributed over K clients/nodes, the objective $f(w)$ in (1) becomes

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w), \tag{2}$$

where $\mathcal{P}_k$ is the subset of data assigned to client k with $n_k = |\mathcal{P}_k|$ and $F_k(w)$ is the average loss on that subset. It is worth noting that in conventional distributed optimization, data are assumed to be distributed uniformly and randomly amongst all clients, i.e. all subsets are equal in size and IID. When these assumptions are fulfilled, $\mathbb{E}_{\mathcal{P}k}[F_k(w)] = f(w)$ holds true for any client k, where $\mathbb{E}_{\mathcal{P}k}[F_k(w)]$ is the expected value of the average loss for client k [5]. However, for reasons mentioned in section 1.1.3, none of these assumptions are satisfied in a federated optimization setting.

Another major difference between federated optimization and distributed optimization lies in optimization priorities. Communication efficiency are the main concern in federated optimization, where the number of clients is potentially very large and upload bandwidth is limited. In contrast, the main priority in a conventional distributed optimization setting, i.e. data-center setting is computational efficiency where communication is usually cheap.

3.2 Federated Averaging

Federated Averaging is an algorithm that builds on Stochastic Gradient Descent (SGD) optimization algorithm used in the majority of machine learning applications [3]. FedAvg contains three hyperparameters that control the total amount of computation: C, the fraction of clients participating in computation on each round; E, the number of epochs that a single client performs over its local dataset on each round and B, the minibatch size used for training by clients.

The main server shares global model parameters with the fraction of each clients C, where these parameters are trained on the datasets locally. Each client k takes a step in gradient descent direction on the currently shared global model:

$$w_{t+1}^k = w_t - \eta g_k, \tag{3}$$

where η is the learning rate and $g_k = \nabla F_k(w_t)$ is the gradient using local data n_k. After the training process is complete, the clients upload their local updates to the main server. The collected updates are then aggregated and a weighted average of all the local models is computed:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k. \tag{4}$$

This process is repeated for each communication round between the main server and the clients until a desired accuracy is achieved. A detailed pseudo-code is presented in Algorithm 1.

Algorithm 1: Federated Averaging

ClinetUpdate(k, w)**:**
  for each epoch i from 1 to E **do**
    batches $\leftarrow \mathcal{P}_k$ split into batches of size B
    for batch b in batches **do**
      $w \leftarrow w - \eta \ell(w; b)$

Main server excutes:
  intialise w_0
  for each round t = 1, 2, ... **do**
    S_t = (random set of $\max(C \cdot K, 1)$ clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

For the purposes of this project, we assume that all clients are always available to participate in the training process, meaning that $C = 1$.

4 Data

To evaluate the performance of a simulated Federated Learning environment, we use two datasets with different features. We present the characteristics of both datasets and how these characteristics may influence the training process.

4.1 Statlog (Shuttle)

The Shuttle dataset is a multivariate dataset obtained from the UCI machine learning repository. The dataset consists of 58,000 data points split into 43,500 points for training, and 14,500 for testing. It contains nine numerical attributes as input x_i, and an output y_i of seven classes. Since around 80% of data points belong to one class (class 1), the default accuracy is also expected to be around the same percentage. The distribution of all classes is shown in Figure 4.

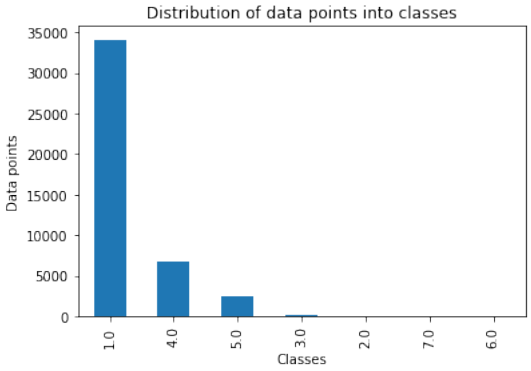


Figure 4: Distribution of Data Points into Classes

4.2 Power Modelling

The dataset is obtained from the Finnish CSC datacenter and consists of 4,418 data points split into 3,538 points for training, and 880 for testing. Network and CPU serve as the input attributes while 'Power Usage' is the target output. Unlike the Shuttle dataset where the output is discrete, the output for this dataset is continuous. Since the output is continuous due to the nature of collected data, we divide the dataset into 10 ranges. The distribution of the data is given in Figure 4.

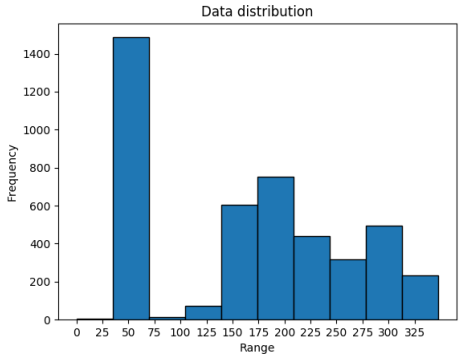


Figure 5: Distribution of Data Points into Ranges of Values

5 Architecture and Hyper-parameters

This section covers our ANN design and specifications. The first part covers the ANN architectures and hyper-parameter values set for our training models. Next, we present our choices of hyper-parameter values.

5.1 Architecture

As mentioned in Section 3, the datasets differ in the nature of their output. Hence, the tasks of built models differ from one another. For the Shuttle dataset, the assigned task is classification, while for Power-Modelling dataset the task is function modelling, i.e. regression. Both neural networks are built using Keras with TensorFlow backend. We use Adam, a powerful optimizer with an adaptive learning rate, that usually offers higher convergence rates compared to other optimizers [2].

In a Federated Learning environment, the size of a subset hosted by each local site is smaller than than that of a dataset hosted on a server where optimization is centralised. Accordingly, each local model should have a number of parameters (weights) that are in proportion to the size of its respective subset. A model with thick layers, i.e. layers that have a large number of neurons, is prone to overfitting, causing an overall degradation of network accuracy. Thus, after a few trials we use between 8 to 32 neurons in a layer in the neural networks used in our investigation.

For the classification task, three fully connected hidden layers are used. The first two of them consist of 32 neurons and the third has 16 neurons. All neurons are activated by a Hyperbolic tangent (tanh) activation function. The output layer has seven neurons, one for each class. Output neurons are activated via a softmax function.

For the linear regression task, we use two fully connected hidden layers consisting of 32 and 8 neurons, respectively. A rectified Linear Unit (ReLU) activation function is used. The output layer consists of one neuron, representing the predicted output without activation.

5.2 Hyper-parameters

We determined the number of epochs and the batch size for each task by observing the behaviour of the models. The number of epochs for the classification task is five and the batch size 1000 for each of the local models whereas the regression models are trained with 10 epochs in a batch size of 50.

In the classification task, as only one category is applicable to each of the data points i.e. a single label categorization is implemented, we use ***categorical_crossentropy*** as our loss function. In categorical crossentropy, the true class a data point belongs to, is encoded as one hot vector. The closer the value of the network's output for this data point to the one hot encoded vector of the true class, the smaller the loss value. The most suitable activation function for the output layer in such context is Softmax.

$$\mathrm{L}(y, \hat{y}) = -\sum_{j=1}^{M} \sum_{i=1}^{N} (y_{ij} * log(\hat{y}_{ij})).$$

Mean squared error is the most commonly used loss function for the regression tasks. This loss function is sensitive to outliers. Since the data set, after input conditioning, is approximately normally distributed around the mean and it is important to penalise the outliers, mean squared error validates to be a good loss function for this task, $\mathrm{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$.

6 Federated Learning on Balanced and IID Datasets

In this section, we present our results for datasets that are uniformly and randomly distributed on clients.

The local models are trained on the local datasets for 200 rounds of communication. After each round of training, the weights are aggregated at the main server. The weighted average of all weights is used

as the starting point for the training process in the next training round.

Since all the datasets have equal number of data points and all datasets are IID, the local models are expected to get updated to equivalent weights. This can be verified by evaluating the performance of local models on a global test dataset. All the local models as well as the global model are expected to converge to a maximum accuracy value simultaneously.

This expected behaviour is verified on both the Shuttle and the Power Modelling datasets.

6.1 Shuttle Dataset

Three local models are trained, each on an IID subset at each site. By doing so, we ensure that all of the three local subsets share a similar distribution of the seven classes. The metric used for evaluating the performance of the models is accuracy. A global test data set (shuttle.tst) is used to evaluate the accuracy of the models. By implementing FedAvg, it is shown that all the models converge to $\sim$ 99.9% accuracy.

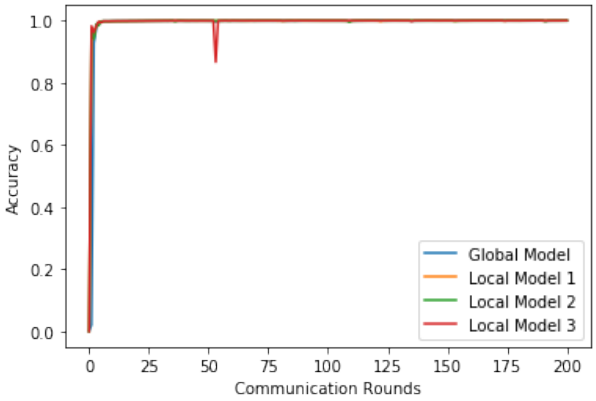


Figure 6: Shuttle dataset: The accuracy of a global model and three local models on balanced and IID local subsets

6.2 Power Modelling Dataset

Four local models are trained, each on an IID subset at each site, ensuring that the four local subsets have similar distributions of ranges. The metric used for comparison of the models is the Mean Absolute Error (MSE). A global test data set consisting of 885 data points is used to compute the MSE of all the models. The global and the local models reach the lowest value of MSE ($\sim$18%) after 60 communication rounds.

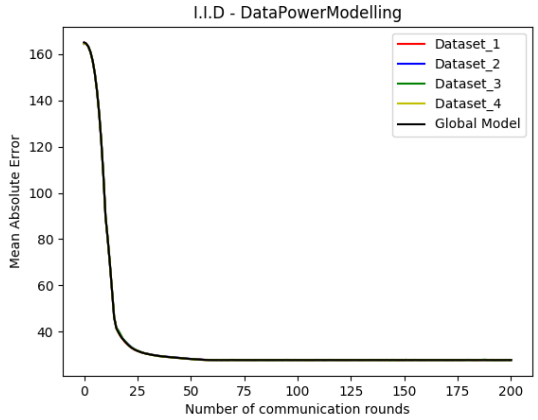


Figure 7: Power Modelling dataset: The MSE of a global model and three local models on balanced and IID local subsets

As shown in the plots, Federated Learning works well when the FedAvg algorithm is implemented in a controlled and a simulated equal data distribution at the local sites. But in the real world applications, the local sites may have unbalanced data. Thus, its important to understand how such data skew influences the convergence of the models.

7 Federated Learning on Unbalanced and non-IID Datasets

This section contains the results of our investigation. We demonstrate the results obtained based on the architecture and hyper-parameters discussed in Section 4 for unbalanced and Non-IID partitioning of data. First, we show the influence of non-IID partitioning of data on accuracy of predictions, and introduce a No-Insight modification to FedAvg to tackle the accuracy drop. Next, the impact of size imbalance in local datasets is presented.

7.1 Non-IID

Non-IID data partitioning in a Federated Learning context, usually means that specific classes or ranges are assigned to clients based on their values. Accordingly, the distribution of data at one local site does not reflect the overall distribution of the whole dataset.

7.1.1 Shuttle Dataset

The dataset is equally distributed among three clients, each holding 14,500 samples. Figure 8 shows a class distribution for the three local sites, where local site no. 3 has four classes missing from its distribution.

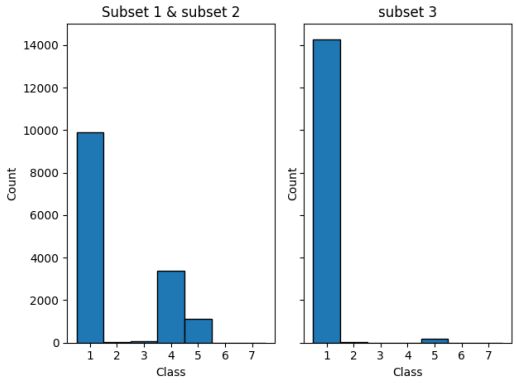


Figure 8: Shuttle Dataset: Local distribution of samples in three local sites

7.1.1.1 Naive Training

Figure 9 shows the results obtained by naive training on the three local datasets for 200 rounds of communication. This approach is called naive, because the global model is being updated through simple averaging (section 2.2) without any modifications on the learning algorithm. The global model manages to eventually converge to a high accuracy $\sim 97\%$ but the convergence process is slow in terms of required communication rounds. Local model 3, which is trained on Subset 3, performs relatively poorly ($\sim$84% accuracy), an expected outcome since the representation of its classes in the test dataset is of the same percentage.

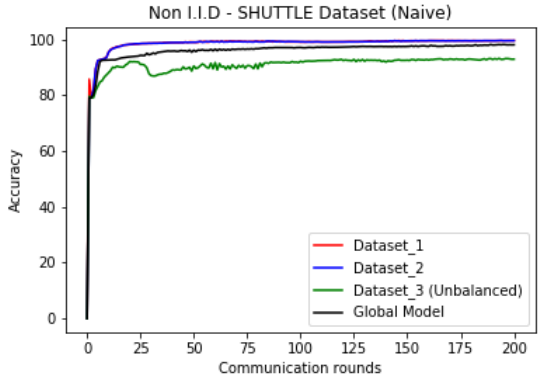


Figure 9: Shuttle Dataset: Naive Federated Learning on local models 1, 2, and 3.

7.1.1.2 No-Insight Training

In this approach, a simple correcting mechanism is implemented, without any further communication between clients and the main server, i.e. the main server does not have access to local data or its class distribution.

One round of communication consists of the following steps:

- Global weights are sent to all clients, where each model is locally trained on the current global weights.
- The main server receives all local updates, aggregates them and computes a new global model.
- The server evaluates both local and global accuracies on a global test dataset.

- Finally, the server sends the weights with the highest accuracy to all the clients for the next round of training.

Results produced by implementing the no-insight approach are shown in figure 10. Both the global model and local model 3 achieve a much faster convergence rate compared with naive training. This gain in performance is achieved through the last two steps mentioned above.

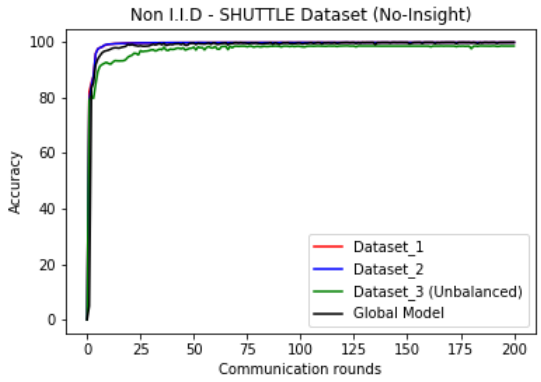


Figure 10: Shuttle dataset: No-insight training achieves a faster convergence rate on the global model and local model 3.

7.1.2 Power Modelling Dataset

Data points were equally distributed but varying in range distribution among four clients, each holding around 885 samples. Figure 11 shows the data distribution for the four local sites, where local sites 3 and 4 have eight and three missing ranges of values, respectively.

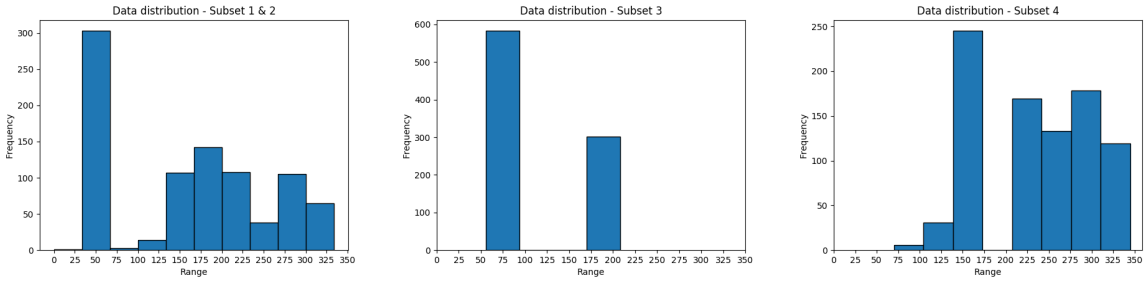
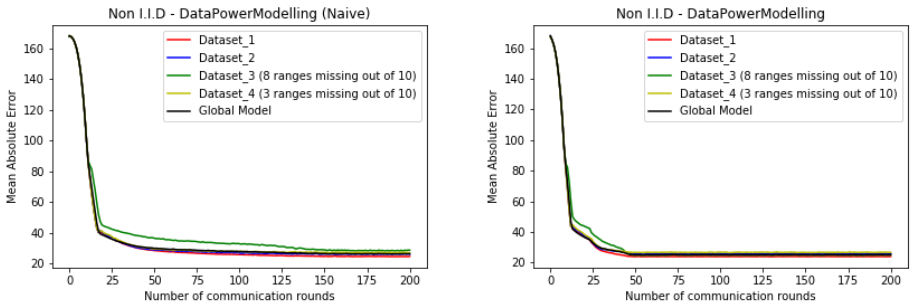


Figure 11: Power Modelling Dataset: Local distribution of samples in four sites

7.1.2.1 Results

Both the naive and no-insight approaches are implemented on the Power Modelling dataset. Figure 12 shows a comparison between the performance of the two approaches. As indicated in Figure 12b, a no-insight improvement leads to a faster convergence rate compared to naive training shown in 12a. The global model, local model 3 and local model 4 achieve MSE value of $\sim$ 27 after 50 rounds of communication.



(a) Naive Training (b) No-insight Training

Figure 12: Power Modelling dataset: Training using naive and no-insight approaches on models 1, 2, 3, and 4

7.2 Size Imbalance

7.2.1 Classification Data:

In this set of experiments, we trained the NNs on the Statlog (Shuttle) data set.

7.2.2 Data Property Explored: Size of Data Set at Each Site

We distribute next the data in an unbalanced way. The first site gets 5500 data points, the second one gets 23500 and the third site 14500 data points. The models are compared based on their accuracy when tested on the shuttle test data set as in the case of balanced data set.

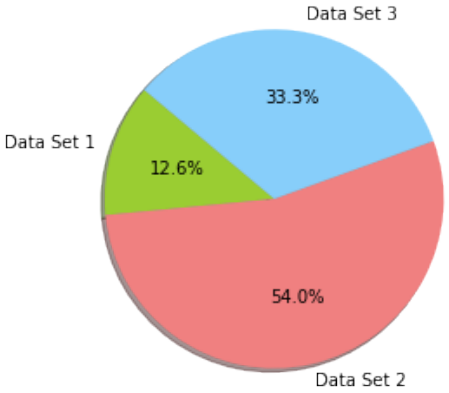


Figure 13: Unbalanced Local Data Sets - Data Distribution

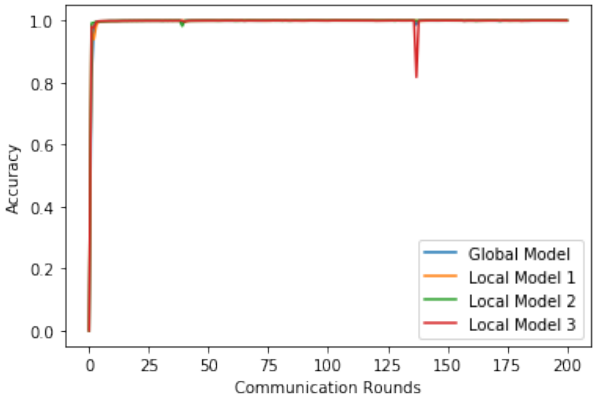


Figure 14: Unbalanced Local Data Sets

As we can see, since the federated averaging algorithm accounts for the size of the data set at a site, the balanced and the unbalanced distributions of the data show similar pattern of the accuracy.

7.2.3 Regression Data:

In this set of experiments, we train the NNs on the Power Modelling data set.

7.2.4 Data Property Explored: Size of Data Set at Each Site

We distribute the data in an unbalanced way. The first site gets 1000 data points, the second one gets 500, the third site 1150 data points and the fourth site 883 data points. The models are compared based on their mean absolute values when tested on a global test data set as in the case of balanced data set.

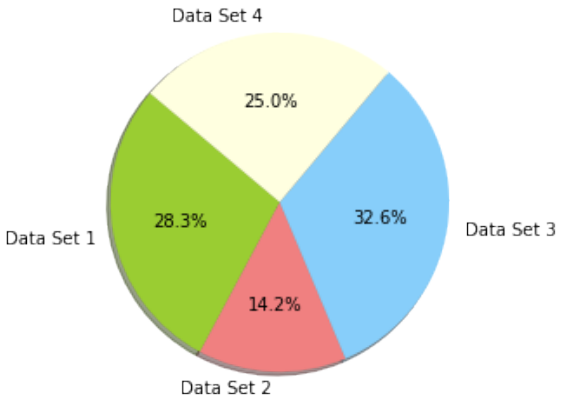


Figure 15: Unbalanced Local Data Sets - Data Distribution

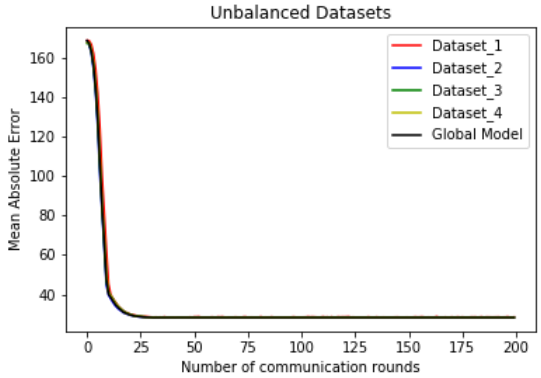


Figure 16: Unbalanced Local Data Sets

As expected, the federated averaging algorithm accounts for the difference in number of data points at the sites. Thus, we see that the balanced and unbalanced data distributions give similar plots for the mean absolute error as the communication among the sites progresses.

8 Discussion

Our investigation is limited in computation and setting. Further investigation in a more realistic environments is needed for a deeper insight into the influence of skewed data in a Federated Learning environment. Training in our investigation is done over all local data (fixed size) at each communication round, which is not the case in actuality. A more realistic situation is when data are added iteratively depending on client usage/behavior and communication efficiency, introducing more statistical heterogeneity at each iteration. Likewise, a global test dataset might not be available to the main server, in

which case the evaluation of accuracy is done locally without a global reference.

The results of our study show that in the case where data points are uniformly distributed among the clients at random, i.e. balanced and IID, FedAvg algorithm has a similar performance to centralized optimization in terms of accuracy. Since all statistical assumptions are met, weighted averaging produces an accurate approximation of the ground truth.

When data are non-IID, and at least one client has sufficient representation of all the classes/ranges, as the case in our investigation, this model's weights eventually dominate over the training process. Such influence speeds up the convergence rate of the global model and reduces the number of required communication rounds.

There are multiple alternatives to FedAvg that can be used in federated optimization like Federated Stochastic Variance Reduced Gradient (FSVRG) and CO-OP. While all of the three algorithms build on SGD, they differ in execution and features. FedAvg is shown to have the highest accuracy among them and performs particularly well with non-IID datasets[6].

9 Conclusion And Future Work

Federated Learning is a promising machine learning paradigm that offers novel solutions to a number of challenges associated with centralized learning. Our investigation shows that Federated Learning is capable of generating highly accurate and robust models in privacy-preserving environments.

Statistical heterogeneity of data, also called data skew, is one of the major issues in Federated Learning that can create biased models at local sites and negatively impact global accuracy. Federated Averaging is an algorithm which handles difference in size of datasets at the clients/sites by taking a weighted sum of local models' parameters. No-insight improvement helps in faster convergence of models to their highest possible accuracy values in case of non-IID datasets.

There are more techniques for handling data skew that have been theorised and are being currently explored in the machine learning community. Generative Adversarial Networks (GAN) is one of such techniques [8]. This type of NNs, after getting an insight into the data present at a site with IID data, generates dummy data at the sites which have non-IID data. This helps in training the local models with better accuracy.

Bounds Aware Fusion is a technique for aggregating local weights at the main server, where aggregation is done while taking the bounds of data ranges present at each local site into consideration. The local models are aggregated at the global level only if they are trained on subsets that share the same feature space. This further increases the accuracy of the global model, but requires that the main server gains an insight into the data distribution at each local site.

Bounds expanding data exchange is another approach where some data exchange is allowed among the sites so that the extreme values of data at each site are comparable. Thus, a selected number of data points could be shared among the sites to fill in the missing values, extrapolating rest of the points. This helps in getting the aggregated values closer to the ground truth.

In a future work, these approaches can be explored on the classification and the regression models to explore the extent of their effectiveness.

References

[1] Corbacho, Jose: Federated Learning - Bringing Machine Learning to the edge with Kotlin and Android. (2018). – URL https://proandroiddev.com/federated-learning-e79e054c33ef

[2] Dozat, Timothy: Incorporating Nesterov Momentum into Adam, 2016

[3] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[4] Li, Tian ; Sahu, Anit K. ; Talwalkar, Ameet ; Smith, Virginia: Federated Learning: Challenges, Methods, and Future Directions. In: *ArXiv* abs/1908.07873 (2019)

[5] McMahan, H. B. ; Moore, Eider ; Ramage, Daniel ; Arcas, Blaise A. y: Federated Learning of Deep Networks using Model Averaging. In: *CoRR* abs/1602.05629 (2016). – URL http://arxiv.org/abs/1602.05629

[6] Nilsson, Adrian ; Smith, Simon: Evaluating the Performance of Federated Learning: A Case Study of Distributed Machine Learning with Erlang, 2018

[7] Rodriguez, Jesus: The Challenges of Centralized AI. (2019). – URL https://towardsdatascience.com/the-challenges-of-decentralized-ai-78bb44b7b69

[8] Verma, Dinesh C. ; White, Graham ; Julier, Simon ; Pasteris, Stepehen ; Chakraborty, Supriyo ; Cirincione, Greg: Approaches to address the data skew problem in federated learning. In: Pham, Tien (Hrsg.): *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications* Bd. 11006 International Society for Optics and Photonics (Veranst.), SPIE, 2019, S. 542 – 557. – URL https://doi.org/10.1117/12.2519621