Project Outline

Mahad Ali

May 2020

1 The problem

With machine learning and deep learning on the rise, and the number of advantages of applying these algorithms to actual user data being immense, it is only natural to find a framework that trains on user data while preserving privacy of the users. One such way to do so is Federated Learning. Which is a framework where instead of bringing user data to the model, we bring model to the user device (we'll refer to them as clients). While a great idea, the original Federated Learning algorithm does not take into account low-end devices with low compute and network resources. This is particularly problematic since data might be distributed in nature (non-IID data sets) and not including low-end devices can introduce a bias in our trained model. While there have been several approaches to tackle device's low-end network (bandwidth issues, etc), not enough light has been device's that are computationally low-end. The goal of this project is to tackle systems heterogeneity while addressing the challenges associated with the proposed approach.

2 Proposal - the use of cloudlets

This project proposes the use of cloudlets (mini clouds) to tackle systems heterogeneity. Each device would then send it's data to the cloudlet, along with it's model that is to be trained. The cloudlet would then use that data to train the local model, and similar to the standard Federated Averaging (FedAvg) algorithm proposed in [1], the cloudlet would send updates back to the server for global aggregation after a specified number of rounds. Instead of the device itself sending the update, it will be an instance (running on the cloudlet as a VM) sending the update. The server would then communicate the new weights back to all cloudlets running instances of clients. This way, the everything works the same as it did before at the server level.

3 Challenges with this approach

Network latency - from the device to the cloudlet The main challenge with this approach is to tackle the added latency in sending data from the client

to the cloudlet. Especially because this data could be massive. One possible approach could be to train the model in chunks. Another could be to place a upper bound/time limit on the time taken to send the data set from the client to the cloudlet. This is the approach that this project takes to tackle this challenge.

Privacy The main purpose of FL is to ensure privacy of user data. Since data is leaving the device, there may be privacy concerns. We handle these concerns using the privacy policy enforcement as described in [3]

4 Approach used to tackle the challenges

Network latency - placing an upper bound on time Data from the client will be sent sequentially in batches. To tackle the latency introduced by network (in sending data set from client to cloudlet), it is assumed that there is an underlying distribution of the time it takes for a batch to transfer from a client to a cloudlet. Since this underlying distribution is unknown, and we aim to place a bound such that most of our batches take time less than this bound that we place, we decided to use the empirical rule stating that 95 percent of our observations lie within two standard deviations of our mean. However, since we do not know the underlying distribution, we instead use sampling to estimate the population mean and standard deviation. While we can not estimate them exactly, we can use the upper bound of their 95 percent confidence intervals, calculated using the t-distribution and chi-square distributions respectively, to find an upper bound. We can then use the empirical rule to provide an upper bound for the time it takes for a batch to transfer from this client to its respective cloudlet. After having profiled the network and having calculated this bound, each device will send it's upper bound value to the main server, which will try to find the max time it will take for a device (among all devices) to transfer all it's data to a cloudlet (upper bound of time taken to send a batch multiplied by the number of steps). This max value (+ the time it takes to run a model on the cloudlet) would then be the time limit for all data to be transferred from the client to the cloudlet. Since we are sending data in batches, training can proceed as batches are received. At the end, devices having exceeeded their time limit would be dropped (their updates will not be used). In practise, however, partial updates from these devices could perhaps still be incorporated using FedProx (introduced in [2]), for example. Notice that we do not worry about systems heterogeneity. This is because it is assumed that all these devices are running on VMs having the same specifications, and hence should not differ in the time it takes for a device to complete its training.

Privacy As mentioned in [3], since cloudlets would serve as the first point of contact for these client devices when sending data, privacy mediators could ensure privacy policy enforcement by allowing clients to decide if they want to denature or delete their data after a round. This allows execution to proceed

as it would for FedAvg without client worrying about their privacy. They do, however, need to trust the cloudlet.

5 Simulations and evaluations

For the sake of this experiment, we aren't profiling the network and taking actual samples. Instead we are using a normal distribution, taking samples out of that distribution and using those to calculate the upper bound of time it takes to transfer a batch - as mentioned above. In practise, however, actual samples will be taken (greater number of samples is encouraged) as soon as the device receives it's FL task. It will then communicate the estimated maximum time it will take. Most of the code being used is the same as that used in [1]. We are comparing our accuracy and loss plots with that of FedAvg at different device drop rates. We are trying to show how drop rates stay consistent even with high systems heterogeneity, in our case. We also experiment with different parameters (intervals, etc) for evaluation purposes.

Datasets used For our simulations, we are using the MNIST and synthetic datasets, along with MCLR model respectively. The synthetic datasets are the same as the ones used in [1], for comparison. They include both, IID and non-IID data sets. The MNIST data set comprises of 10 hand-written digits. The model is trained to accurately predict the digit (labels).

6 Limitations

1 Some devices may dropout in between the process, especially if their connection is not stable and gets disconnected in between.

2 Using this approach for high end devices with constrained network resources may not be a good idea. This is largely because it defeats the purpose of training on the cloudlet and unnecessarily uses already constrained network resources.

3 Using this approach means devices with a bandwidth limit due to their data packages and connection plans may not be able to participate due to the additional cost they might incur.

7 References

1 https://arxiv.org/pdf/1602.05629.pdf

2 https://arxiv.org/pdf/1812.06127.pdf

3 http://elijah.cs.cmu.edu/DOCS/satya-edge2016.pdf