In this document, I start by giving a very brief summary of the FedProx framework as introduced in [1]. This is followed by results from simulation experiments. Lastly, I shed some light on the implementation aspects of simulation experiments performed in [1], followed by some of my own suggestions on implementing simulations for the federated learning model with cloudnets, and possible implementations.

Please feel free to skip any part that may feel irrelevant.

As for the other real-data sets, I could not replicate results related to them due to limited computational resources. I felt that most other data sets represented the same results, as long as they had the same category (iid/non-iid). However, I can find a work around and replicate those (using colab run-time, possibly) if required.

FedProx – Summary

FedProx – a federated learning framework introduced in [1] – is an extension of FedAvg – another federated learning framework introduced in [2]. FedProx builds on top of FedAvg and focuses on two main characteristics: system and statistical heterogeneity.

The basic idea behind FedProx is to include updates from straggler devices – devices unable to complete the set number of epochs due to system constraints and the like – instead of discarding these devices. In contrast, FedAvg discards all devices that fail to have completed a fixed E number of epochs in the round. FedProx accomplishes this by the use of proximal scaling term, μ**,** which is a penalty that makes model stay closer to their initial state, and hence trying to make diverging models to converge. This term is derived using a set of mathematical equations in [1].

Simulation Experiments Results

In the “diff_drop_rates” file with this document, you would find results replicated from the paper. (figure 7) These results are a comparison between FedProx and FedAvg. Unfortunately, I could only replicate results from the synthetic and MNIST data sets, due to limited computational resources. These results, however, show very important information observable in other real data sets (FEMNIST, Shakespeare, Sent140) as well. For each data set, it shows three graphs. These graphs represent experiments with different drop-percentages. From top-to-bottom, these are: 0%, 50%, and 90% respectively. It is apparent from these graphs that for data sets with high system heterogeneity FedProx outperforms FedAvg. For FedProx, two experiments were run per graph. There is a simulation with μ= 0, and another with $\mu > 0$ (μ= 1). A higher value of μ improves performance in case drop-percentage is high.

Aside from observations made on frameworks with different drop-rates, some observations were made with synthetic data sets (iid and non-iid), as in the file “]synthetic_stat_hetro”. These show how FedProx reacts to statistical heterogeneity. Statistical heterogeneity in these synthetic data sets is increasing from left to right. Comparing $\mu = 0$ with $\mu > 1$ shows that the model with μ= 0 (which is actually FedAvg since μ is constant and 0) performs poorly as data becomes more heterogeneous. Similarly, if we look at dissimilarity graphs (variance of gradient), we observe that the model with μ= 0 performs poorly. This is because the more dissimilar the data, the higher the heterogeneity. In such cases, a higher value of μ provides better convergence for the model by penalizing the model when it diverges.

Implementation

These simulations were performed on various different data sets – synthetic and real. This was to take in account data variation and the fact that data could be distributed within a network and hence related.

The implementation has a server, which selects clients, and then sends them local models. After each round there updates are aggregated to form global updates. These are then sent back to the clients in the next round. Clients, in this case, refers to end users with data.

System heterogeneity was simulated by choosing a set of clients (based on drop percentage) from the set of clients chosen for the round. In the simulation, these are chosen using a uniform distribution. These clients are then forced to run less than E epochs for the round. In case FedAvg is being used as the framework, all clients having run less than E epochs till the completion of aggregation cycle (slower devices) are dropped. In the case of FedProx, however, these partial updates are taken into account using the algorithm proposed in [1].

Furthermore, μ in FedProx could be changed/tuned to account for divergence introduced due to heterogeneity. High loss rate could mean an increase of μ, and vice versa.

Suggestions/take away with respect to cloudnets

Perhaps the code used to run simulations could be modified and used for our own simulation experiments. One possible way is to use the client architecture being used in FedProx simulations to form clients with more abundant compute resources, compared to end users. Then, maybe multiple client's models could be sent over to this cloudnet, and performance metrics could be generated to make comparisons between this new framework and other frameworks.

One possible challenge, however, would be to incorporate the effects of additional communication between the clients and the cloudnets. For FedProx, accounting for network constraints was not relevant since there is no additional communication cost – compared to other models. Hence, this needs to be taken into account when simulating results for this new framework.

Another possible use of cloudnets could be to make them act as mini-servers. Allowing them to perform aggregation on weights of a selected number of clients. Updates made by these cloudnets could perhaps be further aggregated after a specific number of cloudnet-level aggregations. The rate of both aggregations could be adaptive. This model however, does not take into account systems heterogeneity and instead focuses on reducing the effect caused by network constraints by allowing updates to be made closer to the end devices.

References

[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, H. Brendan McMahan. (https://arxiv.org/pdf/1602.05629.pdf)

[2] Federated Optimization in Heterogeneous Networks, Tiam Li. (https://arxiv.org/pdf/1812.06127.pdf)