Federated averaging

Alex Bie

February 12, 2022
Updated February 18, 2024

This describes my implementation of federated averaging (FedAvg) from McMahan et al. [2017]. I reproduce some of their MNIST experiments. Not all the runs completed; I think because Colab timed out.

1 Sources used

- Looked at this PyTorch tutorial for the structure of my code: `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`;
- Also this example: `https://github.com/pytorch/examples/tree/master/mnist`;
- And this forum post for averaging parameters in PyTorch: `https://discuss.pytorch.org/t/average-each-weight-of-two-models/77008`

2 Overview

I implemented FedAvg on MNIST. Following their experiment details, I tried two methods of partitioning the dataset for clients (iid and non-iid), and two neural net architectures (*CNN* and a 2 hidden-layer net, referred to in the paper as *2NN*).

My experiments were conducted with 100 clients, and I tried sampling $m = 10$ and $m = 50$ random clients each round. To save memory, I ran the client training sequentially, and kept a running sum of client parameters.

I evaluated accuracy on the test set after every round, and recorded the first round where validation reported test accuracy above 97% (2NN) or above 99% (CNN). Following the paper, clients train for either 1 local epoch (MLP) or 5 local epochs (CNN).

Parameter	Value	Explanation
N	100	# of clients
m	10, 50	# of clients sampled per round
E	1(MLP), 5(CNN)	# of local epochs
B	10	Local batch size
Learning rate	0.05(MLP), 0.01(CNN)	–
Optimizer	`optim.SGD`	–

Table 1: Table of hyperparameters/experiment settings.

The only hyperparameter I tuned was the learning rate. To do this, I did normal training with SGD on MNIST for either 1 epoch (2NN) or 5 epochs (CNN) using the same batch size $B = 10$

as I planned to use in FedAvg experiments. I found that 0.05 worked well for the MLP (95.9% accuracy), and 0.01 worked well for the CNN (98.9% accuracy).

3 Model architectures

I followed the text descriptions in *Section 3: Experimental Results* of the paper to define my models (which can be found in the code).

The 2NN is an MLP with 2 hidden layers with 200 units each and ReLU activations. My implementation has the same number of parameters (199,210) as described in the text.

The CNN has 2 5×5 convolution layers, the first with 32 and the second with 64 channels, each followed by 2×2 max pooling. They are followed by a fully connected layer with 512 units and ReLu activation, and then a linear output layer. My implementation has 582,026 parameters compared to what they report (1,663,370). They did not release code so I couldn't check their exact model definition and figure out why there is a discrepancy.

4 Results

	My results		McMahan et al.	
m	iid	non-iid	iid	non-iid
10	46	199	87	664
50	42	168	75	443

Table 2: Rounds until convergence (97%) to train 2NN with FedAvg (note $E = 1$).

	My results (98.5%)		My results (99%)		McMahan et al. (99%)	
m	iid	non-iid	iid	non-iid	iid	non-iid
10	33	144	>100	>200	18	206
50	28	>100	>100	>100	18	261

Table 3: Rounds until convergence (either 98.5% or 99%) to train CNN with FedAvg (note $E = 5$).

The main results are pictured above in Tables 2 and 3, along with the numbers from their paper. I was able to get thee 2NN working better than what was reported in the paper, but I wasn't able to get the CNN working. My CNNs weren't reaching 99% accuracy quickly enough (although some of them got close) so I had to kill the experiments. If after training for T rounds my CNN still did not hit 99% accuracy, I reported $> T$. I also reported the number of rounds it took get to 98.5% (left-most table), which is not directly comparable to their results. Probably I need to do more hyperparameter search to get the CNNs working well.

The total time to run all experiments on Google Colab was roughly 6 hours.

Figure 1 and 2 plot test accuracy as a function of rounds for all training runs.

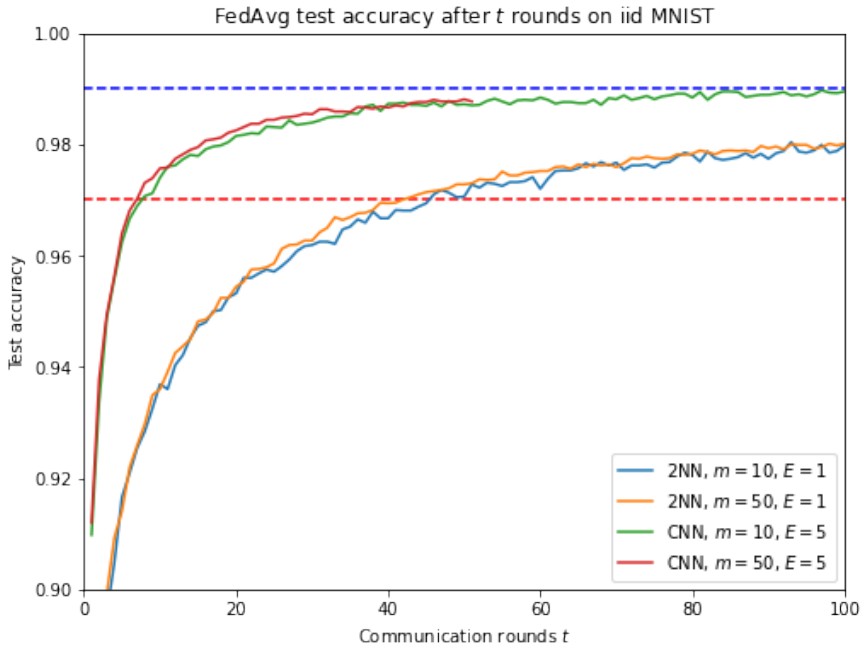

Figure 1: Test accuracy after round t, iid case. We take more steps per round for larger m, larger E.

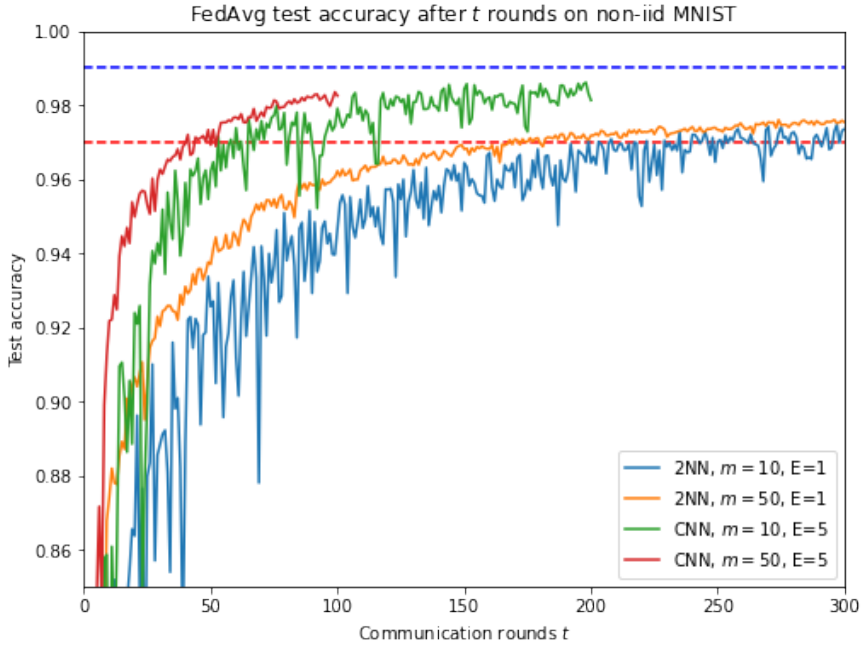

Figure 2: Test accuracy after round t, non-iid case.

References

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS'17)*, 2017.