Communication-Efficient Learning of Deep Networks from Decentralized Data

Umar Ahmed

Agenda

- Introduction
- Federated Learning & Optimization
- Federated Algorithms
- Experiments

Introduction

- Phones, tablets, and other electronic devices generate a significant amount of data
    - How can we use this for building more intelligent models?
- What issues would we face?
    - Privacy/Storing data

Introduction: Federated Learning

- Rather than using a central server for both storage and computation of data, use ‘federation’ of clients (devices) controlled by central server
- Clients compute updates to the model and send to server - this is the only thing communicated, and doesn’t need to be stored

Advantages:

- No need for direct access to raw training data, it is decoupled from model training
- Reduction of privacy/security risks
- Real world data

Use cases: Image classification, Language models

Federated Learning: Key Properties

- Non-IID
- Unbalanced
- Massively distributed
- Limited communication

Federated Learning: Optimization

- Computation cost is reduced/free, Communication cost is high
  - Goal: add computation so that communication is lessened
- How do we add computation?
  - Increased Parallelism
  - Increased computation on each client

Federated Learning: Outline

- Fixed set of K clients
- Each round, random fraction of C clients selected
- The server sends the current global model to each client
- Each client performs a local computation on the given global state and produces an update
- Each update from each client is aggregated to the global model/updated to central server
- All steps are repeated

Federated Learning: Optimization

$$\min_{w \in \mathbb{R}^d} f(w)$$

w: model
fi = loss(xi, yi;w)
K: clients
n: total data
nk: data for client k
Pk: indexes of data points for client k

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

FederatedAveraging: SGD

The baseline algorithm: FedSGD

- C is set to 1 (C controls global batch size, so gradient descent on whole batch)
- Fixed learning rate
- Each client computes gradient on its local data on current model (one step)
- Central server aggregates these gradients and applies update

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

$$g_k = \nabla F_k(w_t)$$

$$\forall k,\ w_{t+1}^k \leftarrow w_t - \eta g_k$$

$$\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$$

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

FederatedAveraging (FedAvg)

- Iterate local update multiple times before moving it to averaging step

$$w^k \leftarrow w^k - \eta \nabla F_k(w^k)$$

- Manipulating computation:
  - B: local minibatch size
  - C: fraction of clients per round
  - E: number of training passes on local dataset

FederatedAveraging: Updates

$$u_k = E\frac{n_k}{B}$$

Number of local updates per round

- Manipulating computation:
  - B: local minibatch size
  - C: fraction of clients per round
  - E: number of training passes on local dataset

FederatedAveraging: Algorithm

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

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

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

Experimentation: Overview

- MNIST digit recognition
  - 2NN
  - CNN
- 'The Complete Works of William Shakespeare'
  - LSTM
- CIFAR-10 Images
  - LSTM
- Large-Scale LSTM - Social Media Posts

MNIST: Digit Recognition Task

- Multi-layer perceptron, 2 hidden layers, 200 units each using ReLu activation functions
- CNN, 5x5 convolution layers, fully connected layer with 512 units and ReLu activation, softmax output layer

- IID: data shuffled, 100 clients with 600 examples each
- Non-IID: data sorted first by digit, 200 shards of size 300, give 2 shards to each of 100 clients

2NN	IID		NON-IID	
C	$B=\infty$	$B=10$	$B=\infty$	$B=10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E=5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

Shakespeare Dataset

- Non-IID: Client dataset for each speaking role
  - 1146 Clients
  - Train-test split of 80/20
  - Highly unbalanced/temporally separated
- IID
  - 1146 Clients
  - Balanced dataset

- LSTM language model
  - Reads a character, predicts the next character
  - 2 LSTM layers, 256 nodes each, softmax output layer - one node per character
  - Unroll length of 80 characters

Shakespeare & Digit: Fixed C Size

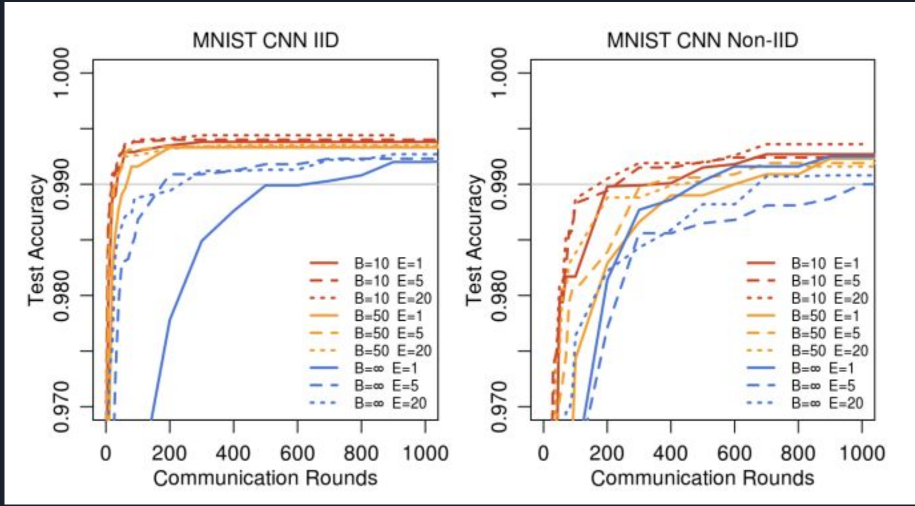
- C = 0.1
- Little to no cost for computation

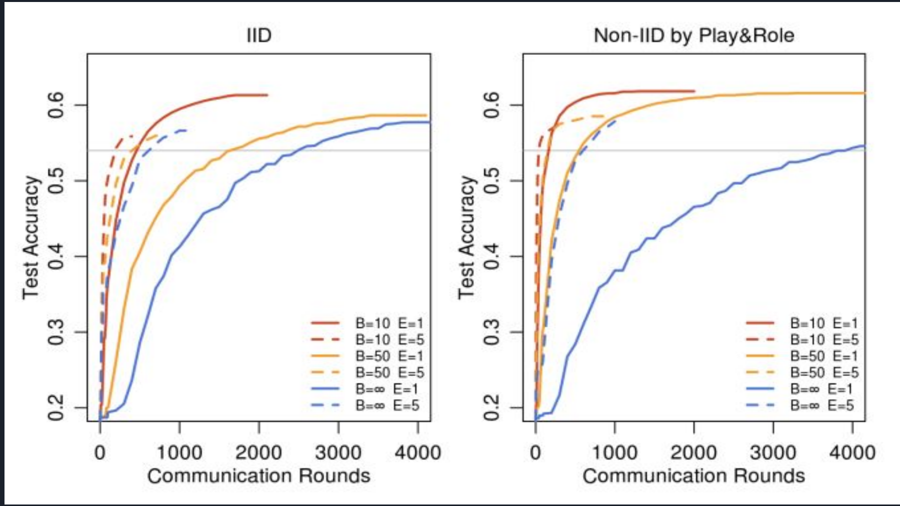
$$u = (\mathbb{E}[n_k]/B)E$$
$$nE/(KB)$$

Table 2: Number of communication rounds to reach a target accuracy for FedAvg, versus FedSGD (first row, $E = 1$ and $B = \infty$). The u column gives $u = En/(KB)$, the expected number of updates per round.

	MNIST CNN, 99% ACCURACY				
CNN	E	B	u	IID	NON-IID
FEDSGD	1	∞	1	626	483
FEDAVG	5	∞	5	179 (3.5×)	1000 (0.5×)
FEDAVG	1	50	12	65 (9.6×)	600 (0.8×)
FEDAVG	20	∞	20	234 (2.7×)	672 (0.7×)
FEDAVG	1	10	60	34 (18.4×)	350 (1.4×)
FEDAVG	5	50	60	29 (21.6×)	334 (1.4×)
FEDAVG	20	50	240	32 (19.6×)	426 (1.1×)
FEDAVG	5	10	300	20 (31.3×)	229 (2.1×)
FEDAVG	20	10	1200	18 (34.8×)	173 (2.8×)
	SHAKESPEARE LSTM, 54% ACCURACY				
LSTM	E	B	u	IID	NON-IID
FEDSGD	1	∞	1.0	2488	3906
FEDAVG	1	50	1.5	1635 (1.5×)	549 (7.1×)
FEDAVG	5	∞	5.0	613 (4.1×)	597 (6.5×)
FEDAVG	1	10	7.4	460 (5.4×)	164 (23.8×)
FEDAVG	5	50	7.4	401 (6.2×)	152 (25.7×)
FEDAVG	5	10	37.1	192 (13.0×)	41 (95.3×)

Shakespeare & Digit: Fixed C Size





Gray lines show target accuracies used

Observations

- Evidence for robustness for Federated approach:
    - Using more computation per client (FedAvg) -> less number of rounds
    - Significant speedup in non-IID data
- Shakespeare data
    - Representative of real-world applications
    - Unbalanced, still converges relatively fast
- FedAvg converges faster than FedSGD
    - Manipulation of B and E

Shakespeare dataset, changing E value

Can we over-optimize on the client datasets?

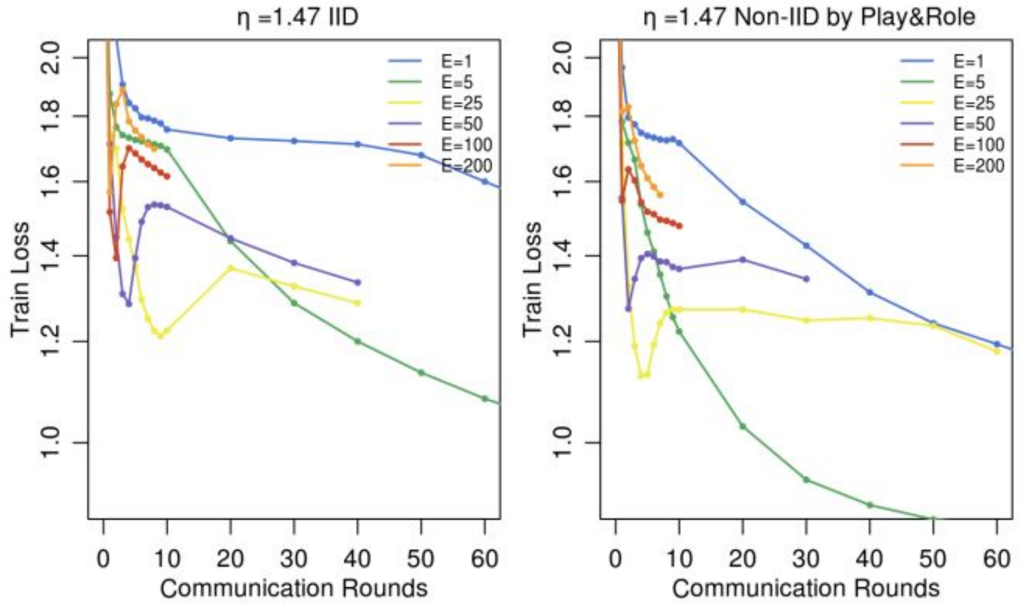


Figure 3: The effect of training for many local epochs (large E) between averaging steps, fixing $B = 10$ and $C = 0.1$ for the Shakespeare LSTM with a fixed learning rate $\eta = 1.47$.

CIFAR-10 Experiment

- 10 classes of 32x32 images
- 100 clients
  - 500 training
  - 100 testing
- Balanced/IID data

- TensorFlow model
  - Two CNNs, two fully connected layers,
  - linear transformation layer

Table 3: Number of rounds and speedup relative to baseline SGD to reach a target test-set accuracy on CIFAR10. SGD used a minibatch size of 100. `FedSGD` and `FedAvg` used $C = 0.1$, with `FedAvg` using $E = 5$ and $B = 50$.

Acc.	80%	82%	85%
SGD	18000 (—)	31000 (—)	99000 (—)
FedSGD	3750 (4.8×)	6600 (4.7×)	N/A (—)
FedAvg	280 (64.3×)	630 (49.2×)	2000 (49.5×)

CIFAR-10 Experiment

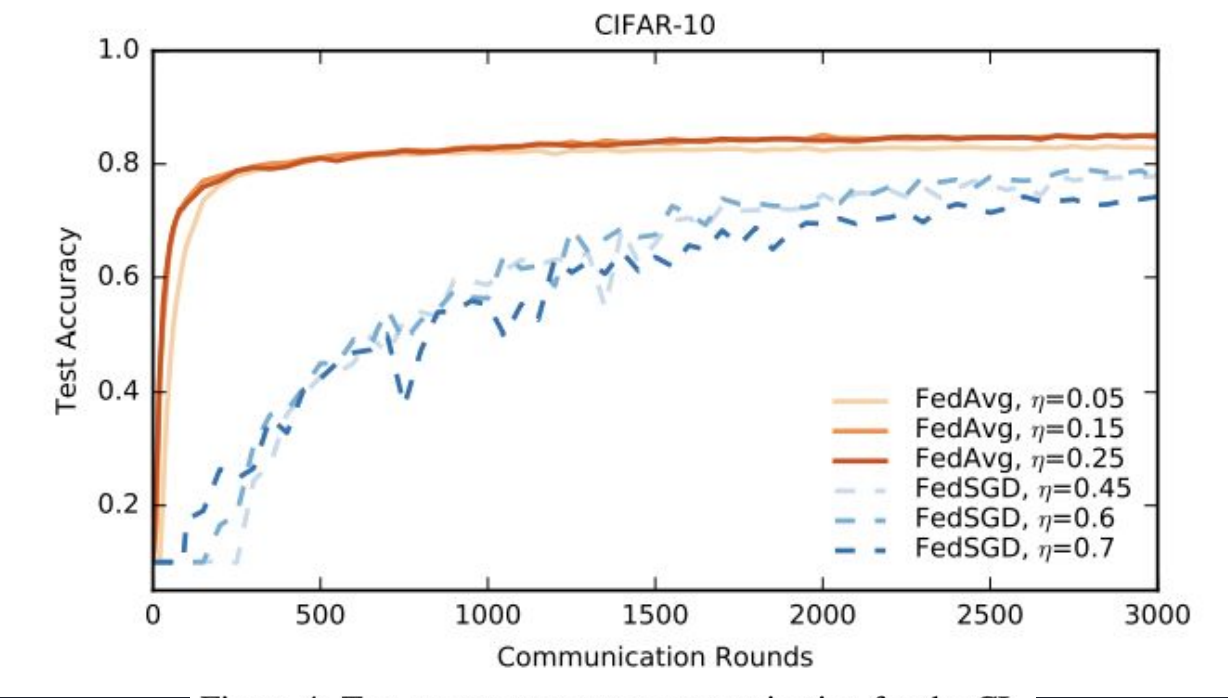
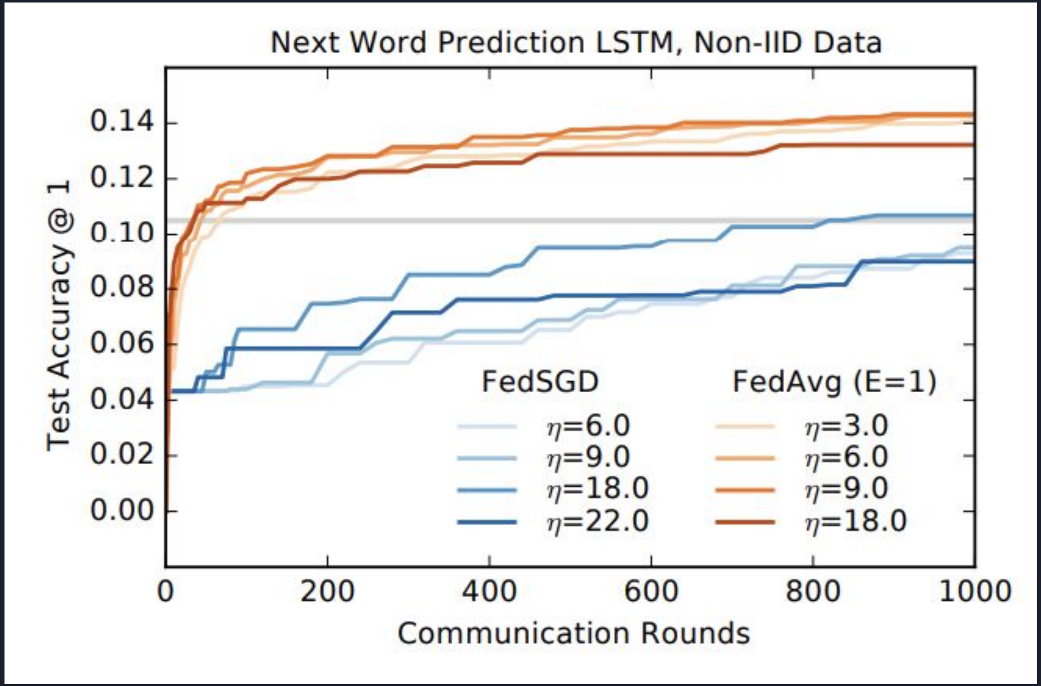


Figure 4: Test accuracy versus communication for the CIFAR10 experiments. `FedSGD` uses a learning-rate decay of 0.9934 per round; `FedAvg` uses $B = 50$, learning-rate decay of 0.99 per round, and $E = 5$.

Large Scale LSTM Experiment

- Large-scale next word prediction

- 10 million public posts
  - 500,000 clients
  - 5000 words per client
  - 100,000 posts test set
- Model:
  - LSTM, 256 nodes
  - 10,000 word vocabulary
  - Unroll of 10 words
- FedAvg: B = 8, E = 1



Conclusions/Takeaway

- FedAvg able to reduce communication rounds significantly
  - Tested on a variety of different model architectures
- Positive results even in Non-IID and Unbalanced cases
- Practical privacy benefits, more methods may be interesting to explore later

Questions?