Federated Learning

Min Du

Postdoc, UC Berkeley
Outline

- Preliminary: deep learning and SGD
- Federated learning: FedSGD and FedAvg
- Related research in federated learning
- Open problems
Outline

- Preliminary: deep learning and SGD
- Federated learning: FedSGD and FedAvg
- Related research in federated learning
- Open problems
The goal of deep learning

• Find a function, which produces a desired output given a particular input.

<table>
<thead>
<tr>
<th>Example task</th>
<th>Given input</th>
<th>Desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image classification</td>
<td><img src="image" alt="Image classification" /></td>
<td>8</td>
</tr>
<tr>
<td>Next-word-prediction</td>
<td><em>Looking forward to your ?</em></td>
<td><em>reply</em></td>
</tr>
<tr>
<td>Playing GO</td>
<td><img src="image" alt="Playing GO" /></td>
<td>Next move</td>
</tr>
</tbody>
</table>

$w$ is the set of parameters contained by the function
Finding the function: model training

- Given one input sample pair \((x_0, y_0)\), the goal of deep learning model training is to find a set of parameters \(w\), to maximize the probability of outputting \(y_0\) given \(x_0\).

Given input: \(x_0\)  
Maximize: \(p(5|x_0, w)\)
Finding the function: model training

- Given a training dataset containing $n$ input-output pairs $(x_i, y_i), i \in [1, n]$, the goal of deep learning model training is to find a set of parameters $w$, such that the average of $p(y_i)$ is maximized given $x_i$. 

Given input:

<table>
<thead>
<tr>
<th>Label</th>
<th>Label</th>
<th>Label</th>
<th>Label</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Output:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
Finding the function: model training

• Given a training dataset containing $n$ input-output pairs $(x_i, y_i), i \in [1, n]$, the goal of deep learning model training is to find a set of parameters $w$, such that the average of $p(y_i)$ is maximized given $x_i$.

• That is,

$$\text{maximize} \quad 1 \sum_{i=1}^{n} \frac{1}{n} p(y_i | x_i, w)$$

Which is equivalent to

$$\text{minimize} \quad 1 \sum_{i=1}^{n} -\log(p(y_i | x_i, w))$$

A basic component for loss function $l(x_i, y_i, w)$ given sample $(x_i, y_i)$:

Let $f_i(w) = l(x_i, y_i, w)$ denote the loss function.
Deep learning model training

For a training dataset containing \( n \) samples \((x_i, y_i), 1 \leq i \leq n\), the training objective is:

\[
\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)
\]

\( f_i(w) = l(x_i, y_i, w) \) is the loss of the prediction on example \((x_i, y_i)\)

**No closed-form solution**: in a typical deep learning model, \( w \) may contain millions of parameters.

**Non-convex**: multiple local minima exist.
Solution: Gradient Descent

Randomly initialized weight $w$

Compute gradient $\nabla f(w)$

At the local minimum, $\nabla f(w)$ is close to 0.

Learning rate $\eta$ controls the step size

$w_{t+1} = w_t - \eta \nabla f(w)$ (Gradient Descent)

How to stop? – when the update is small enough – converge.

$$\| w_{t+1} - w_t \| \leq \epsilon$$

or

$$\| \nabla f(w_t) \| \leq \epsilon$$

**Problem:** Usually the number of training samples $n$ is large – slow convergence
Solution: Stochastic Gradient Descent (SGD)

- At each step of gradient descent, instead of compute for all training samples, randomly pick a small subset (mini-batch) of training samples $(x_k, y_k)$.

$$w_{t+1} \leftarrow w_t - \eta \nabla f (w_t; x_k, y_k)$$

- Compared to gradient descent, SGD takes more steps to converge, but each step is much faster.
Outline

- Preliminary: deep learning and SGD
- Federated learning: FedSGD and FedAvg
- Related research in federated learning
- Open problems
The importance of data for ML

“...The biggest obstacle to using advanced data analysis isn’t skill base or technology; it’s plain old access to the data.”

-Edd Wilder-James, Harvard Business Review
“Data is the New Oil”
**Private data:** all the photos a user takes and everything they type on their mobile keyboard, including **passwords, URLs, messages, etc.**
Instead of uploading the raw data, **train a model locally and upload the model.**

**Addressing privacy:** Model parameters will never contain more information than the raw training data.

**Addressing network overhead:** The size of the model is generally smaller than the size of the raw training data.
Federated optimization

- Characteristics (Major challenges)
  - Non-IID
    - The data generated by each user are quite different
  - Unbalanced
    - Some users produce significantly more data than others
  - Massively distributed
    - \# mobile device owners >> avg \# training samples on each device
  - Limited communication
    - Unstable mobile network connections
A new paradigm – Federated Learning

*a synchronous update scheme that proceeds in rounds of communication*

Federated learning – overview

Deployed by Google, Apple, etc.
Federated learning – overview

In round number $i...$
Federated learning – overview

Round number $i+1$ and continue...
For efficiency, at the beginning of each round, a random fraction $C$ of clients is selected, and the server sends the current model parameters to each of these clients.
Federated learning – detail

- Recall in traditional deep learning model training
  - For a training dataset containing \( n \) samples \((x_i, y_i)\), \( 1 \leq i \leq n \), the training objective is:

\[
\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)
\]

\( f_i(w) = l(x_i, y_i, w) \) is the loss of the prediction on example \((x_i, y_i)\)

- Deep learning optimization relies on SGD and its variants, through **mini-batches**

\[
w_{t+1} \leftarrow w_t - \eta \nabla f(w_t; x_k, y_k)
\]
Federated learning – detail

- In federated learning
  - Suppose $n$ training samples are distributed to $K$ clients, where $P_k$ is the set of indices of data points on client $k$, and $n_k = |P_k|$.
  - For training objective: $\min_{w \in \mathbb{R}^d} f(w)$

\[
 f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) \overset{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i(w)
\]
A baseline – **FederatedSGD (FedSGD)**

- A randomly selected client that has \( n_k \) training data samples in federated learning \( \approx \) *A randomly selected sample in traditional deep learning*

- Federated SGD (FedSGD): a single step of gradient descent is done per round

- Recall in federated learning, a \( C \)-fraction of clients are selected at each round.
  - \( C=1 \): full-batch (non-stochastic) gradient descent
  - \( C<1 \): stochastic gradient descent (SGD)
A baseline – *FederatedSGD (FedSGD)*

Learning rate: $\eta$; total #samples: $n$; total #clients: $K$; #samples on a client $k$: $n_k$; clients fraction $C = 1$

- In a round $t$:
  - The central server broadcasts current model $w_t$ to each client; each client $k$ computes gradient: $g_k = \nabla F_k(w_t)$, on its local data.
    - Approach 1: Each client $k$ submits $g_k$; the central server aggregates the gradients to generate a new model:
      - $w_{t+1} \leftarrow w_t - \eta \nabla f(w_t) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$.
      - Recall $f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$
    - Approach 2: Each client $k$ computes: $w_{t+1}^{k} \leftarrow w_t - \eta g_k$; the central server performs aggregation:
      - $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k}$

*For multiple times $\Rightarrow$ FederatedAveraging (FedAvg)*
Federated learning – deal with limited communication

- Increase computation
  - Select more clients for training between each communication round
  - Increase computation on each client
Federated learning – *FederatedAveraging* (FedAvg)

- **Learning rate**: $\eta$; **total #samples**: $n$; **total #clients**: $K$; **#samples on a client $k$**: $n_k$; **clients fraction $C$**

- In a round $t$:
  - The central server broadcasts current model $w_t$ to each client; each client $k$ computes gradient: $g_k = \nabla F_k(w_t)$, on its local data.
    - **Approach 2**:
      - Each client $k$ computes for $E$ epochs: $w_{t+1}^k \leftarrow w_t - \eta g_k$
      - The central server performs aggregation: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
      - Suppose $B$ is the local mini-batch size, #updates on client $k$ in each round: $u_k = E \frac{n_k}{B}$.

*The amount of computation in each round is determined by:*
Federated learning – *FederatedAveraging (FedAvg)*

**Model initialization**

- Two choices:
  - On the central server
  - On each client

*Shared initialization works better in practice.*

The loss on the full MNIST training set for models generated by $\theta w + (1 - \theta)w'$.
Federated learning – *FederatedAveraging* (FedAvg)

Model averaging

- As shown in the right figure:

In practice, naïve parameter averaging works surprisingly well.

The loss on the full MNIST training set for models generated by

\[ \theta w + (1 - \theta)w' \]
Federated learning – *FederatedAveraging (FedAvg)*

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

**Server executes:**
- initialize $w_0$
- for each round $t = 1, 2, \ldots$ 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 $k$
- $B \leftarrow$ (split $P_k$ into batches of size $B$)
- for each local epoch $i$ from 1 to $E$ do
  - for batch $b \in B$ do
    - $w \leftarrow w - \eta \nabla \ell(w; b)$
  - return $w$ to server

1. At first, a model is randomly initialized on the central server.
2. For each round $t$:
   - i. A random set of clients are chosen;
   - ii. Each client performs local gradient descent steps;
   - iii. The server aggregates model parameters submitted by the clients.
**Federated learning – Evaluation**

<table>
<thead>
<tr>
<th>C</th>
<th>2NN</th>
<th>IID</th>
<th>Non-IID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>1455</td>
<td>316</td>
<td>4278</td>
</tr>
<tr>
<td>0.1</td>
<td>1474 (1.0×)</td>
<td>87 (3.6×)</td>
<td>1796 (2.4×)</td>
</tr>
<tr>
<td>0.2</td>
<td>1658 (0.9×)</td>
<td>77 (4.1×)</td>
<td>1528 (2.8×)</td>
</tr>
<tr>
<td>0.5</td>
<td>75 (4.2×)</td>
<td>16 (3.1×)</td>
<td>97 (9.9×)</td>
</tr>
<tr>
<td>1.0</td>
<td>70 (4.5×)</td>
<td>— (—)</td>
<td>380 (8.6×)</td>
</tr>
</tbody>
</table>

**FedSGD**  | **FedAvg**  | **FedSGD**  | **FedAvg**  |

**Impact of varying C**

In general, the higher C, the smaller #rounds to reach target accuracy.

**Image classification**

- **#clients**: 100
- **Dataset**: MNIST
  - IID: Random partition
  - Non-IID: each client only contains two digits
  - Balanced

![Images of digits](image-url)
Federated learning – Evaluation

- Dataset from: *The Complete Works of Shakespeare*
  - #clients: 1146, each corresponding to a speaking role
  - Unbalanced: different #lines for each role
  - Train-test split ratio: 80% - 20%
  - A balanced and IID dataset with 1146 clients is also constructed

- Task: next character prediction

- Model: character-level LSTM language model
Federated learning – *Evaluation*

In general, the more computation in each round, the faster the model trains. *FedAvg also converges to a higher test accuracy (B=10, E=20).*

- The effect of increasing computation in each round (decrease B / increase E)
- Fix C=0.1
Federated learning – *Evaluation*

- The effect of increasing computation in each round (decrease B / increase E)
- Fix $C=0.1$

In general, the more computation in each round, the faster the model trains. *FedAvg also converges to a higher test accuracy ($B=10$, $E=5$).*
Federated learning – *Evaluation*

- What if we maximize the computation on each client? $E \to \infty$

Best performance may achieve at earlier rounds; increasing #rounds do not improve.

*Best practice: decay the amount of local computation when the model is close to converge.*
Federated learning – **Evaluation**

<table>
<thead>
<tr>
<th>Acc.</th>
<th>80%</th>
<th>82%</th>
<th>85%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>18000 (——)</td>
<td>31000 (——)</td>
<td>99000 (——)</td>
</tr>
<tr>
<td>FedSGD</td>
<td>3750 (4.8×)</td>
<td>6600 (4.7×)</td>
<td>N/A (——)</td>
</tr>
<tr>
<td>FedAVG</td>
<td>280 (64.3×)</td>
<td>630 (49.2×)</td>
<td>2000 (49.5×)</td>
</tr>
</tbody>
</table>

**Image classification**

- **#clients:** 100
- **Dataset:** CIFAR-10
  - IID: Random partition
  - Non-IID: each client only contains two digits
  - Balanced
Federated learning – *Evaluation*

- Dataset from: *10 million public posts from a large social network*
  - #clients: 500,000, each corresponding to an author
- Task: next word prediction
- Model: word-level LSTM language model

200 clients per round; $B=8$, $E=1$
Outline

- Preliminary: deep learning and SGD
- Federated learning: FedSGD and FedAvg
- Related research in federated learning
- Open problems
Federated learning – related research

Google FL Workshop: https://sites.google.com/view/federated-learning-2019/home


Secure aggregation. Practical Secure Aggregation for Privacy-Preserving Machine Learning, CCS’17


Decentralize the central server via blockchain.
HiveMind: Decentralized Federated Learning

Local data

Local data

Local data

Local data

HiveMind Smart Contract

Differentially Private Global Model

aggregated noise

Oasis Blockchain Platform

Differential privacy

Secure aggregation

Model encryption
HiveMind: Decentralized Federated Learning

Local data

Model M(i)

Model M(i)

Model M(i)

Model M(i)

Global model M(i)

private global model

aggregated noise

Oasis Blockchain Platform

In round number i...
HiveMind: Decentralized Federated Learning

In round number $i$...

Differential privacy

Secure aggregation

Model encryption

Local data

Gradient updates for $M(i)$

$\text{Differential privacy}$

Local data

Gradient updates for $M(i)$

$\text{Secure aggregation}$

Model encryption

Local data

Gradient updates for $M(i)$

Oasis Blockchain Platform
HiveMind: Decentralized Federated Learning

- **Local data**
- **Gradient updates for M(i)**
- **Oasis Blockchain Platform**
- **HiveMind Smart Contract**
- **Local data**
- **Gradient updates for M(i)**

**In round number i...**

- **Secure aggregation**
- **Model encryption**
- **Differential privacy**

\[ \text{Gradient updates for M(i)} + \text{DP noise} \]

\[ \text{Gradient updates for M(i)} + \text{DP noise} \]

\[ \text{Gradient updates for M(i)} + \text{DP noise} \]
HiveMind: Decentralized Federated Learning

In round number $i$...

- Differential privacy
- Secure aggregation
- Model encryption

Local data

Oasis Blockchain Platform

Gradient updates for $M(i)$

Secure Aggregation

$DP$ noise

Local data
HiveMind: Decentralized Federated Learning

In round number $i$...

- **Differential privacy**
- **Secure aggregation**
- **Model encryption**

Local data

HiveMind Smart Contract

Global model $M(i+1)$

Gradient updates for $M(i)$

Oasis Blockchain Platform

Local data

Gradient updates for $M(i)$

Gradient updates for $M(i)$

Gradient updates for $M(i)$
HiveMind: Decentralized Federated Learning

- **Local data**
- **Local data**
- **Local data**
- **Local data**

**HiveMind** Smart Contract

- **Global model $M(i+1)$**
- **Differentially private**
- **Secure aggregation**
- **Model encryption**

**Round number $i+1$ and continue...**

**Oasis Blockchain Platform**
HiveMind: Decentralized Federated Learning

- **Local data**
- **Local data**
- **Local data**
- **Local data**

HiveMind Smart Contract

- Differentially Private Global Model
- Aggregated noise
- DP noise

Oasis Blockchain Platform

- Secure aggregation
- Model encryption
Outline

- Preliminary: deep learning and SGD
- Federated learning: FedSGD and FedAvg
- Related research in federated learning
- Open problems
Federated learning – open problems

- Detect data poisoning attacks, while secure aggregation is being used.
- Asynchronous model update in federated learning and its co-existence with secure aggregation.
- Further reduce communication overhead through quantization etc.
- The usage of differential privacy in each of the above settings.
- ......
Thank you!

Min Du
min.du@berkeley.edu