Guarding user Privacy with Federated Learning and Differential Privacy

Brendan McMahan
mcmahan@google.com

DIMACS/Northeast Big Data Hub Workshop on Overcoming Barriers to Data Sharing including Privacy and Fairness
2017.10.24
Federated Learning

Our Goal

Imbue mobile devices with state of the art machine learning systems without centralizing data and with privacy by default.
Federated Learning

Imbue mobile devices with state of the art machine learning systems without centralizing data and with privacy by default.

A very personal computer
2015: 79% away from phone ≤2 hours/day
63% away from phone ≤1 hour/day
25% can't remember being away at all

2013: 72% of users within 5 feet of phone most of the time.

Plethora of sensors

Innumerable digital interactions

1. 2015 Always Connected Research Report, IDC and Facebook
2. 2013 Mobile Consumer Habits Study, Jumio and Harris Interactive.
Federated Learning

Deep Learning

non-convex

millions of parameters

complex structure (e.g., LSTMs)

Our Goal

Imbue mobile devices with state of the art machine learning systems without centralizing data and with privacy by default.
Federated Learning

Our Goal

Imbue mobile devices with state of the art machine learning systems without centralizing data and with privacy by default.

Distributed learning problem

Horizontally partitioned
Nodes: millions to billions
Dimensions: thousands to millions
Examples: millions to billions
Federated Learning

Our Goal

Imbue mobile devices with state of the art machine learning systems without centralizing data and with privacy by default.

Federated decentralization

facilitator
Deep Learning, the short short version

\[ f(\text{input, parameters}) = \text{output} \]
Deep Learning, the short short version

\[ f(\text{input}, \text{parameters}) = \text{output} \]

\[ \text{loss}(\text{parameters}) = \frac{1}{n} \sum_i \text{difference}(f(\text{input}_i, \text{parameters}), \text{desired}_i) \]
Deep Learning, the short short version

\[ f(\text{input}, \text{parameters}) = \text{output} \]

\[ \text{loss}(\text{parameters}) = \frac{1}{n} \sum_i \text{difference}(f(\text{input}_i, \text{parameters}), \text{desired}_i) \]

Adjust these to minimize this.
Deep Learning, the short short version

\[ f(\text{input}, \text{parameters}) = \text{output} \]

\[ \text{loss}(\text{parameters}) = \frac{1}{n} \sum_i \text{difference}(f(\text{input}_i, \text{parameters}), \text{desired}_i) \]

**Stochastic** Choose a random subset of training data

**Gradient** Compute the "down" direction on the loss function

**Descent** Take a step in that direction

(Rinse & repeat)
Cloud-centric ML for Mobile
The model lives in the cloud.
We train models in the cloud.
Mobile Device

Current Model Parameters
Make predictions in the cloud.
Gather training data in the cloud.
And make the models better.
On-Device Predictions
(Inference)
Instead of making predictions in the cloud
Distribute the model, make predictions on device.
On-device inference

User Advantages
- Low latency
- Longer battery life
- Less wireless data transfer
- Better offline experience
- Less data sent to the cloud

Developer Advantages
- Data is already localized
- New product opportunities

World Advantages
- Raise privacy expectations for the industry
On-device training

User Advantages
- Low latency
- Longer battery life
- Less wireless data transfer
- Better offline experience
- Less data sent to the cloud (training data stays on device)

Developer Advantages
- Data is already localized
- New product opportunities
- Straightforward personalization
- Simple access to rich user context

World Advantages
- Raise privacy expectations for the industry

Bringing **model training** onto mobile devices.
On-device training

User Advantages
- Low latency
- Longer battery life
- Less wireless data transfer
- Better offline experience
- Less data sent to the cloud (training data stays on device)

Developer Advantages
- Data is already localized
- New product opportunities
- Straightforward personalization
- Simple access to rich user context

World Advantages
- Raise privacy expectations for the industry

Bringing model training onto mobile devices.
Federated Learning
Federated Learning

Federated Learning is the problem of training a shared global model under the coordination of a central server, from a federation of participating devices which maintain control of their own data.
Federated Learning

Many devices will be offline.
Federated Learning

1. Server selects a sample of e.g. 100 online devices.
Federated Learning

1. Server selects a sample of e.g. 100 online devices.
2. Selected devices download the current model parameters.
3. Users compute an update using local training data
Federated Learning

4. Server aggregates users’ updates into a new model.

Repeat until convergence.
Applications of federating learning

What makes a good application?

- On-device data is more relevant than server-side proxy data
- On-device data is privacy sensitive or large
- Labels can be inferred naturally from user interaction

Example applications

- Language modeling (e.g., next word prediction) for mobile keyboards
- Image classification for predicting which photos people will share
- ...
Challenges of Federated Learning

Massively Distributed
   Training data is stored across a very large number of devices

Limited Communication
   Only a handful of rounds of unreliable communication with each device

Unbalanced Data
   Some devices have few examples, some have orders of magnitude more

Highly Non-IID Data
   Data on each device reflects one individual's usage pattern

Unreliable Compute Nodes
   Devices go offline unexpectedly; expect faults and adversaries

Dynamic Data Availability
   The subset of data available is non-constant, e.g. time-of-day vs. country

… or, why this isn't just "standard" distributed optimization
Server

Until Converged:
1. Select a random subset (e.g. 100) of the (online) clients

2. In parallel, send current parameters $\theta_t$ to those clients

3. $\theta_{t+1} = \theta_t + \text{data-weighted average of client updates}$

---

Selected Client $k$

1. Receive $\theta_t$ from server.

2. Run some number of minibatch SGD steps, producing $\theta'$

3. Return $\theta' - \theta_t$ to server.

---

Large-scale LSTM for next-word prediction

Rounds to reach 10.5% Accuracy
FedSGD  820
FedAvg  35

23x decrease in communication rounds

Model Details
1.35M parameters
10K word dictionary
embeddings $\in \mathbb{R}^{96}$, state $\in \mathbb{R}^{256}$
corpus: Reddit posts, by author
CIFAR-10 convolutional model

Updates to reach 82%
- SGD: 31,000
- FedSGD: 6,600
- FedAvg: 630

49x decrease in communication (updates) vs SGD
(IID and balanced data)
Federated Learning & Privacy
4. Server aggregates users’ updates into a new model.

Repeat until convergence.
Might these updates contain privacy-sensitive data?
Might these updates contain privacy-sensitive data?
Might these updates contain privacy-sensitive data?

1. Ephemeral
Might these updates contain privacy-sensitive data?

1. Ephemeral

2. Focussed

Improve privacy & security by minimizing the "attack surface"
Might these updates contain privacy-sensitive data?

1. Ephemeral
2. Focussed
3. Only in aggregate
Wouldn't it be even better if...

Google aggregates users' updates, but cannot inspect the individual updates.
Google aggregates users' updates, but cannot inspect the individual updates.

A novel, practical protocol

Might the final model memorize a user's data?

1. Ephemeral
2. Focussed
3. Only in aggregate
4. Differentially private
Differential Privacy
Differential Privacy (trusted aggregator)
Federated Averaging

Server

Until Converged:
1. Select a random subset (e.g. C=100) of the (online) clients
2. In parallel, send current parameters $\theta_t$ to those clients

Selected Client $k$

1. Receive $\theta_t$ from server.
2. Run some number of minibatch SGD steps, producing $\theta'$
3. Return $\theta' - \theta_t$ to server.

3. $\theta_{t+1} = \theta_t + \text{data-weighted average of client updates}$
**Server**

**Until Converged:**
1. Select each user **independently** with probability $q$, for say $E[C]=1000$ clients

2. In parallel, send current parameters $\theta_t$ to those clients

**Selected Client $k$**

1. Receive $\theta_t$ from server.

2. Run some number of minibatch SGD steps, producing $\theta'$

3. Return $\theta' - \theta_t$ to server.

3. $\theta_{t+1} = \theta_t + \text{data-weighted average of client updates}$
Until Converged:
1. Select each user **independently** with probability $q$, for say $E[C]=1000$ clients
2. In parallel, send current parameters $\theta_t$ to those clients

**Selected Client $k$**

1. Receive $\theta_t$ from server.
2. Run some number of minibatch SGD steps, producing $\theta'$
3. Return $\text{Clip}(\theta' - \theta_t)$ to server.

3. $\theta_{t+1} = \theta_t + \text{data-weighted average of client updates}$
**Server**

Until Converged:
1. Select each user independently with probability $q$, for say $E[C]=1000$ clients
2. In parallel, send current parameters $\theta_t$ to those clients

**Selected Client $k$**

1. Receive $\theta_t$ from server.
2. Run some number of minibatch SGD steps, producing $\theta'$
3. Return $\text{Clip}(\theta' - \theta_t)$ to server.

$\theta_{t+1} = \theta_t + \text{bounded sensitivity}$ data-weighted average of client updates
Differentially-Private Federated Averaging

Server

Until Converged:
1. Select each user independently with probability $q$, for say $E[C]=1000$ clients
2. In parallel, send current parameters $\theta_t$ to those clients

Selected Client $k$

1. Receive $\theta_t$ from server.
2. Run some number of minibatch SGD steps, producing $\theta'$
3. Return $\text{Clip}(\theta' - \theta_t)$ to server.

$\theta_{t+1} = \theta_t + \text{bounded sensitivity} \text{ data-weighted average of client updates} + \text{Gaussian noise } N(0, I\sigma^2)$
Privacy Accounting for Noisy SGD: Moments Accountant

Large-scale LSTM for next-word prediction

Rounds to reach 10.5% Accuracy
FedSGD: 820
FedAvg: 35

23x decrease in communication rounds
Large-scale LSTM for next-word prediction

Rounds to reach 10.5% Accuracy
FedSGD 820
FedAvg 35

23x decrease in database queries
The effect of clipping updates

No Clipping

Sampling $E[C] = 100$ users per round.

Aggressive Clipping
The effect of clipping updates

Sampling $E[C] = 100$ users per round.
The effect of noising updates

Clipping at $S=20$

Sampling $E[C] = 100$
users per round.
Differential Privacy for Language Models

Differential Privacy for Language Models

**Baseline Training**
- users per round = 100
- tokens per round = 160k
- 17.5% accuracy in 4120 rounds

(1.152, 1e-9) DP Training
- $\mathbb{E}$[users per round] = 5k
- $\mathbb{E}$[tokens per round] = 8000k
- 17.5% estimated accuracy in 5000 rounds
Differential Privacy for Language Models

Baseline Training
users per round = 100
tokens per round = 160k
17.5% accuracy in 4120 rounds

(1.152, 1e-9) DP Training
E[users per round] = 5k
E[tokens per round] = 8000k
17.5% estimated accuracy in 5000 rounds

Private training achieves equal accuracy, but using 60x more computation.
Differential Privacy

Differential Privacy (trusted aggregator)
Differential Privacy

Local Differential Privacy

∑

[Image of brain diagrams and statistical distributions]
Differential Privacy

Differential Privacy with Secure Aggregation

Google
Differential Privacy is complementary to Federated Learning

- FL algorithms touch data one user (one device) at time — natural algorithms for user-level privacy.
- Communication constraints mean we want to touch the data as few times as possible — also good for privacy.
- The DP guarantee is complementary to FL's focused collection & ephemeral updates.
Federated Learning in Gboard

Federated Learning: Collaborative Machine Learning without Centralized Training Data
Thursday, April 06, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

Standard machine learning approaches require centralizing the training data on one machine or in a datacenter. And Google has built one of the most secure and robust cloud infrastructures for processing this data to make our services better. Now for models trained from user interaction with mobile devices, we’re introducing an additional approach: Federated Learning.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and OnDevice Smart Reply) by bringing model training to the device as well.

It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.
Open Questions and Challenges

Showing privacy is possible
Many open research questions:
- Further lower computational and/or utility cost of differential privacy
- More communication-efficient algorithms for FL

Making privacy easy
Possible is not enough. Research to enable "privacy by default" in machine learning.
- Can federated learning be as easy as centralized learning?
- Differential privacy for deep learning without parameter tuning?
- How do we handle privacy budgets across time and across domains?
Questions