Federated Learning

Idea, Applications, and

Daniel Schalk

November 6, 2018

Table of Contents

Terminology

Distributed Learning

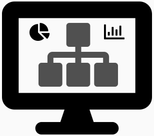
Classical parallelization, advantages:

- Speed up fitting process
- Train model on much more data
- Idea behind Spark, Hadoop, ...
- Assumption that we already have a database which we want to distribute, hence data of the splits should follow the same distribution

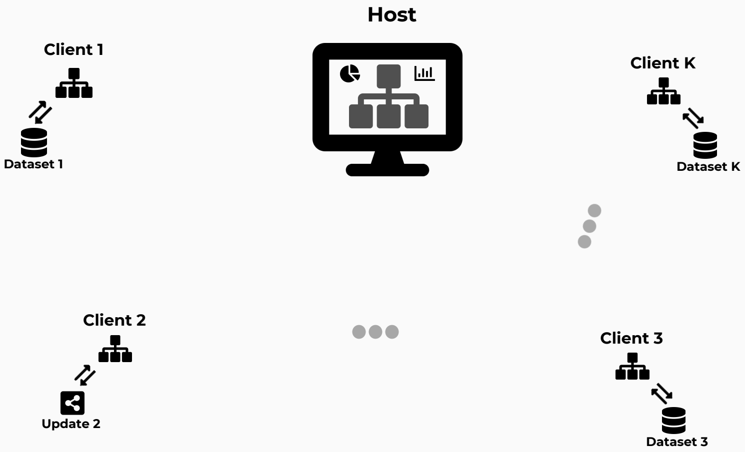
Federated/Decentralized Learning

What is it About?

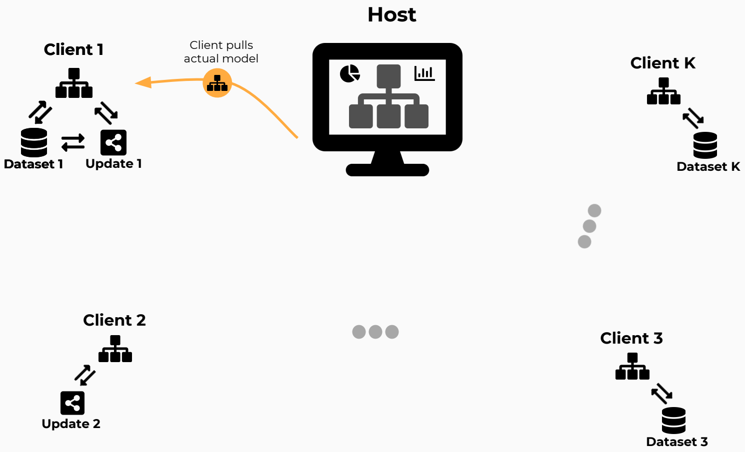
Host



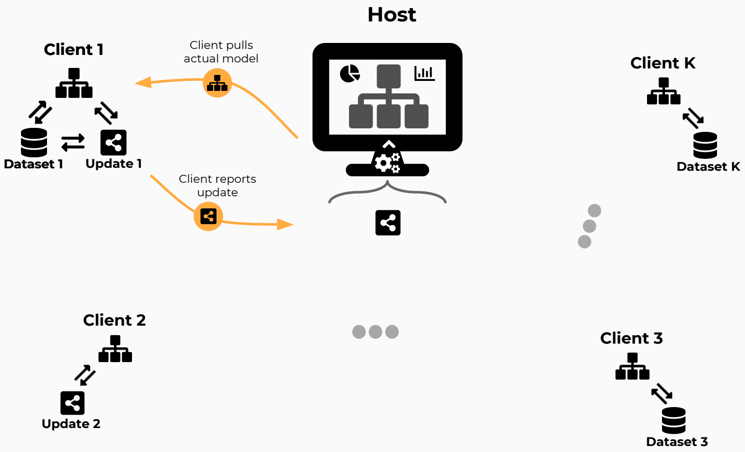
What is it About?



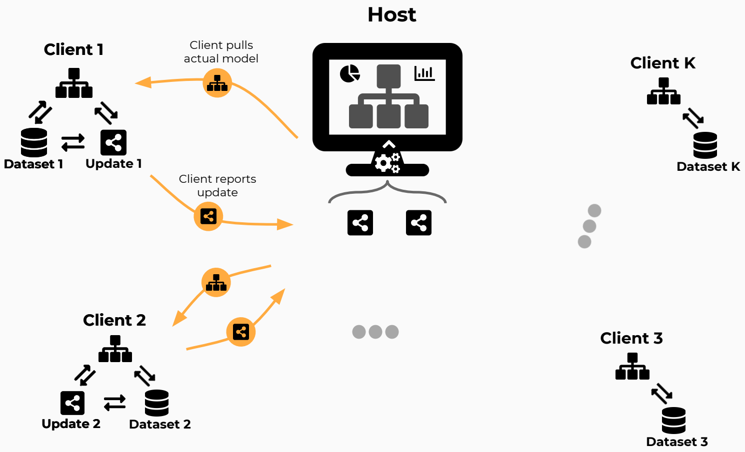
What is it About?



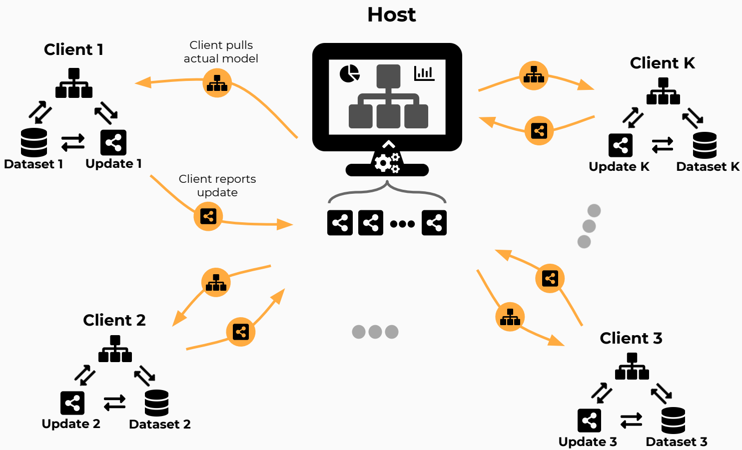
What is it About?



What is it About?



What is it About?



Federated Learning/Decentralized Learning

- Model comes to the data, not data to the model
- Privacy concerning method
- ...

Common Problems in Decentralized Learning

Federated Learning as learning on decentralized data with the following properties:

- **Non-IID** The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution.
- **Unbalanced** Similarly, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data.
- **Massively distributed** We expect the number of clients participating in an optimization to be much larger than the average number of examples per client.
- **Limited communication** Mobile devices are frequently offline or on slow or expensive connections.

Federated Learning

Terminology

Feature Vector	$x \in \mathcal{X}$
Target Variable	$y \in \mathcal{Y}$
Parameter Vector	$\theta \in \Theta$
Prediction	$\hat{y} = f(x, \hat{\theta})$
Loss Function	$L(y, \hat{y})$
Dataset	$\mathcal{D} = (x^{(i)}, y^{(i)}),\ \forall i \in \{1, \dots n\}$
Empirical Risk	$\mathcal{R}_{\text{emp}}(\mathcal{D}, \theta) = \frac{1}{n} \sum_{(x,y) \in \mathcal{D}} L(y, f(x, \theta))$

Gradient Descent

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta \nabla_\theta \mathcal{R}_{\textsf{emp}}(\hat{\theta}_t)$$

- With Gradient:

$$\frac{\delta}{\delta\theta} L(y, f(x, \theta)) = \nabla_\theta \mathcal{R}_{\textsf{emp}}(\theta)$$

- And learning rate $\eta > 0$

Federated Averaging 1

- We now got K different clients
- Each client holds a non-distributable dataset $\mathcal{D}_k$, $k \in \{1, \dots, K\}$ with n_k observations
- Each dataset yields an empirical risk $\mathcal{R}_{\text{emp}}(\mathcal{D}_k, \theta)$

$\Rightarrow$ How to find a good model (represented by $\hat{\theta}$) trained on all datasets?

Federated Averaging 2

Data: $\mathcal{D}_1, \dots, \mathcal{D}_K$
Result: Parameter vector $\hat{\theta}$
Initialization: $\hat{\theta}_0$ e.g. randomly and set $t = 1$;
while *Stop criteria is not reached* **do**
    Send $\hat{\theta}_{t-1}$ to all K clients;
    for $k = 1$ **to** K **do**
        Calculate and report $\nabla_\theta \mathcal{R}_{\text{emp}}(\mathcal{D}_k, \hat{\theta}_{t-1})$ to host;
    end
    `/* Host conduct Federated Averaging step: */`
    $$\hat{\theta}_t = \hat{\theta}_{t-1} - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla_\theta \mathcal{R}_{\text{emp}}(\mathcal{D}_k, \hat{\theta}_{t-1});$$
    Check if stop criteria is reached, e.g. $\|\hat{\theta}_t - \hat{\theta}_{t-1}\|_2 < \varepsilon$;
    Increment $t \leftarrow t + 1$;
end

This algorithm requires communication between host and clients after each iteration. $\rightarrow$ **Very expensive!**

Federated Averaging 3

In order to make one huge update on the host side we can conduct the update on the client side and average the updates:

$$\gamma_k = \hat{\theta}_{t-1} - \eta \mathcal{R}_{\text{emp}}(\mathcal{D}_k, \hat{\theta}_{t-1})$$

$$\Rightarrow \ \hat{\theta}_t = \sum_{k=1}^{K} \frac{n_k}{n} \gamma_k = \hat{\theta}_{t-1} - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla_\theta \mathcal{R}_{\text{emp}}(\mathcal{D}_k, \hat{\theta}_{t-1})$$

Additionally, we can think of different methods to average the reported updates:

$$\hat{\theta}_t = \text{avg}(\gamma_1, \dots, \gamma_K, \alpha) = \sum_{k=1}^{K} \frac{n_k}{n} \gamma_k$$

Federated Averaging 4

Host side: Distribute and collect data;

Data: $\mathcal{D}_1, \dots, \mathcal{D}_K$

Result: Parameter vector $\hat{\theta}$

Initialization: $\hat{\theta}_0$ e.g. randomly and set $t = 1$;

while *Stop criteria is not reached* **do**

  Send $\hat{\theta}_{t-1}$ to all K clients;

  for $k = 1$ **to** K **do**

    $\gamma_k = \text{clientUpdate}(k, \hat{\theta}_{t-1})$;

    Report γ_k to host;

  end

  `/* Host conduct Federated Averaging step: */`

  $\hat{\theta}_t = \text{avg}(\gamma_1, \dots, \gamma_K, \alpha)$;

  Check if stop criteria is reached, e.g. $\|\hat{\theta}_t - \hat{\theta}_{t-1}\|_2 < \varepsilon$;

  Increment $t \leftarrow t + 1$;

end

Algorithm 1: Federated Averaging Algorithm

Federated Averaging 5

With algorithm 1 we can also think about reducing communication costs. Therefore, we conduct E updates in clientUpdate:

Client side: Conduct update on local dataset;
Data: $\mathcal{D}_k$ and $\hat{\theta}_{t-1}$
Result: k-th client update γ_k
Initialization: $\gamma_{k,0} = \hat{\theta}_{t-1}$;
for $i = 1$ **to** E **do**
$\quad \gamma_{k,i} = \gamma_{k,i-1} - \eta \nabla_\theta \mathcal{R}_{\text{emp}}(\mathcal{D}_k, \gamma_{k,i-1})$;
end
Report $\gamma_k = \gamma_{k,E}$ to host;

Algorithm 2: Communication reduction in clientUpdate

Example with Logistic Regression

Setup

- **Loss Function/Negative Log-Likelihood**

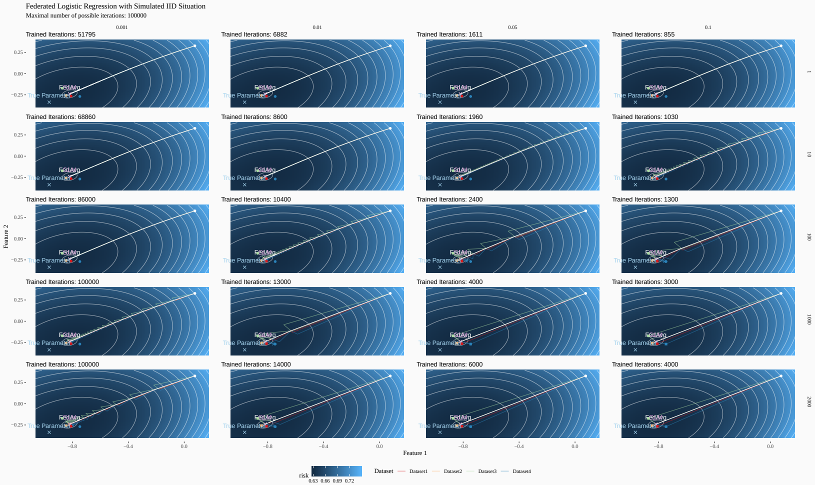
$$L\left(y, f(x,\theta)\right) = -y \log\left(f(x,\theta)\right) - (1-y)\log\left(1 - f(x,\theta)\right)$$

- **Response Function**

$$f(x) = \left(1 + \exp\left(-x^T\theta\right)\right)^{-1}$$

- **Score Function**

$$\frac{\delta}{\delta\theta} L(y, f\,(x,\theta)) = x\,(y - f(x,\theta))$$


Federated Logistic Regression with Simulated IID Situation
Maximal number of possible iterations: 100000
0.001
0.01
0.05
0.1
1
10
100
1000
2000
Trained Iterations: 51795
Trained Iterations: 6882
Trained Iterations: 1611
Trained Iterations: 855
Trained Iterations: 68860
Trained Iterations: 8600
Trained Iterations: 1960
Trained Iterations: 1030
Trained Iterations: 86000
Trained Iterations: 10400
Trained Iterations: 2400
Trained Iterations: 1300
Trained Iterations: 100000
Trained Iterations: 13000
Trained Iterations: 4000
Trained Iterations: 3000
Trained Iterations: 100000
Trained Iterations: 14000
Trained Iterations: 6000
Trained Iterations: 4000
FedAvg
Feature 2
Feature 1
0.25
0.00
−0.25
−0.8
−0.4
0.0
risk
0.63 0.66 0.69 0.72
Dataset
Dataset1
Dataset2
Dataset3
Dataset4

Boosting and Federated Learning