Optimization algorithms for heterogeneous clients in Federated Learning

Praneeth Karimireddy, Martin Jaggi, **Satyen Kale**, Mehryar Mohri,
Sashank Reddi, Sebastian Stich, Ananda Theertha Suresh

Tight analysis of FedAvg when clients are heterogeneous (non iid data)

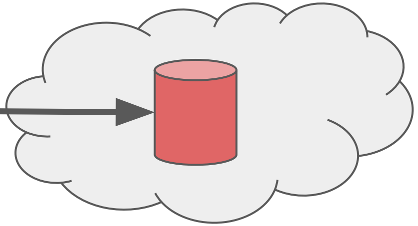
Explain degradation of FedAvg via the '*drift*' in the client updates

Prove SCAFFOLD is resilient to heterogeneity and client sampling

Federated Learning: Setting

[McMahan et al. 2016]

model



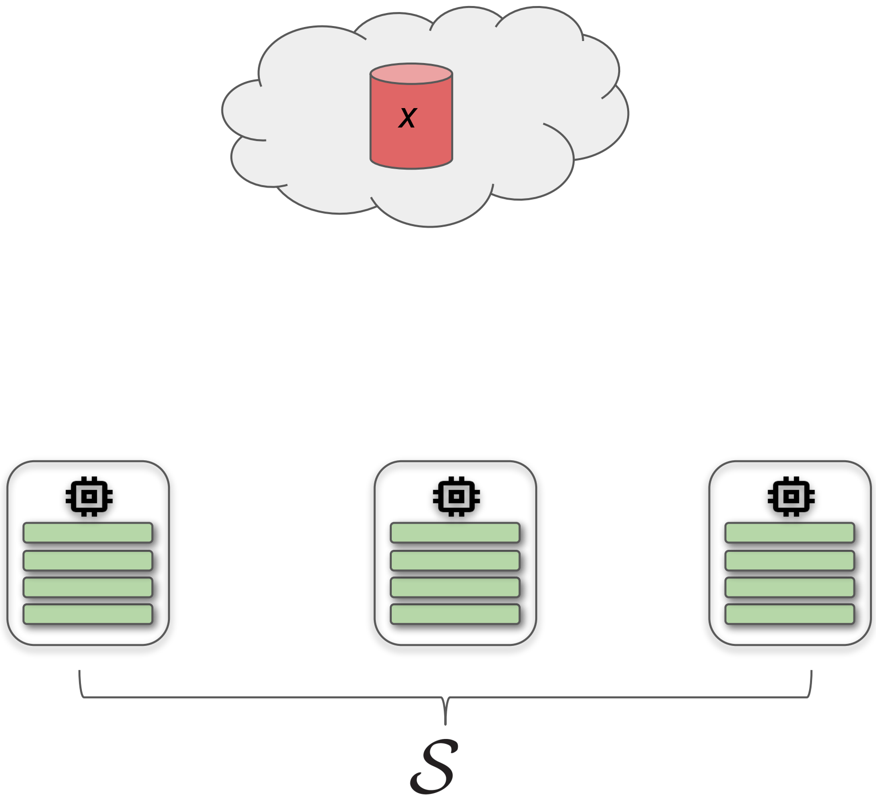
Server
(e.g. Google)

data

Clients
(e.g. hospitals, phones)

Federated Learning: Setting

[McMahan et al. 2016]

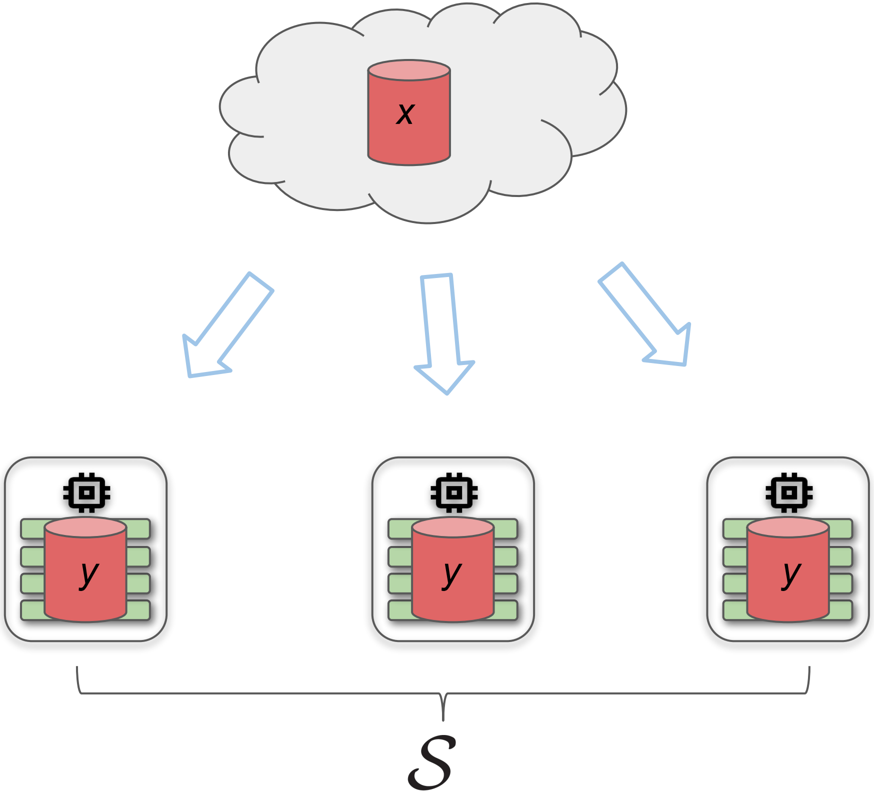


In each round,

- Some subset of clients are chosen

Federated Learning: Setting

[McMahan et al. 2016]

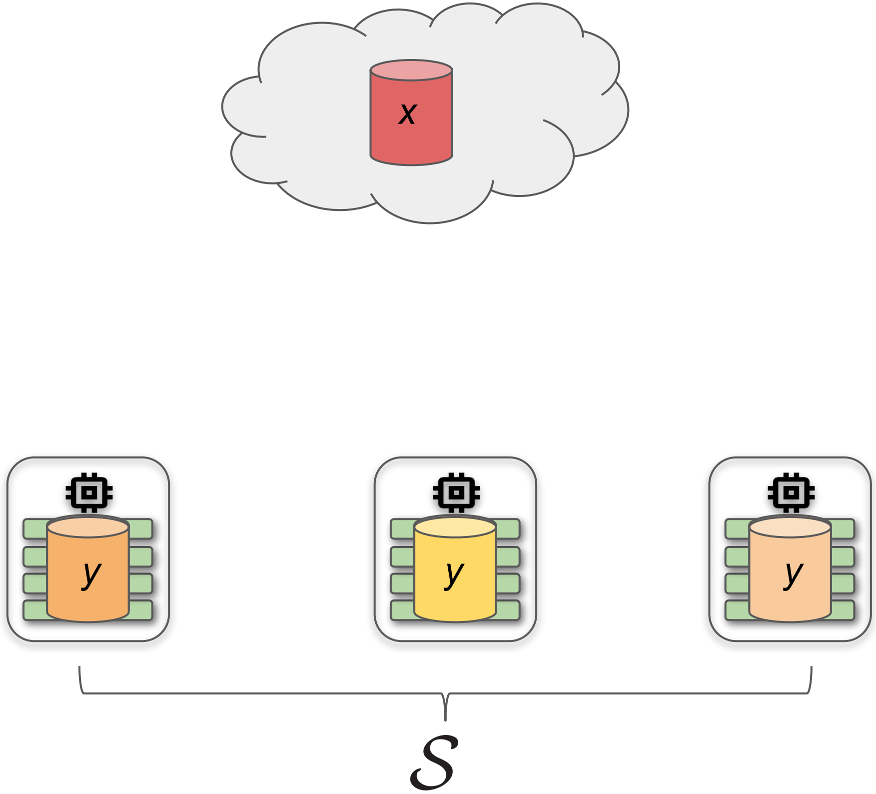


In each round,

- Some subset of clients are chosen
- copy of server model is sent to clients

Federated Learning: Setting

[McMahan et al. 2016]

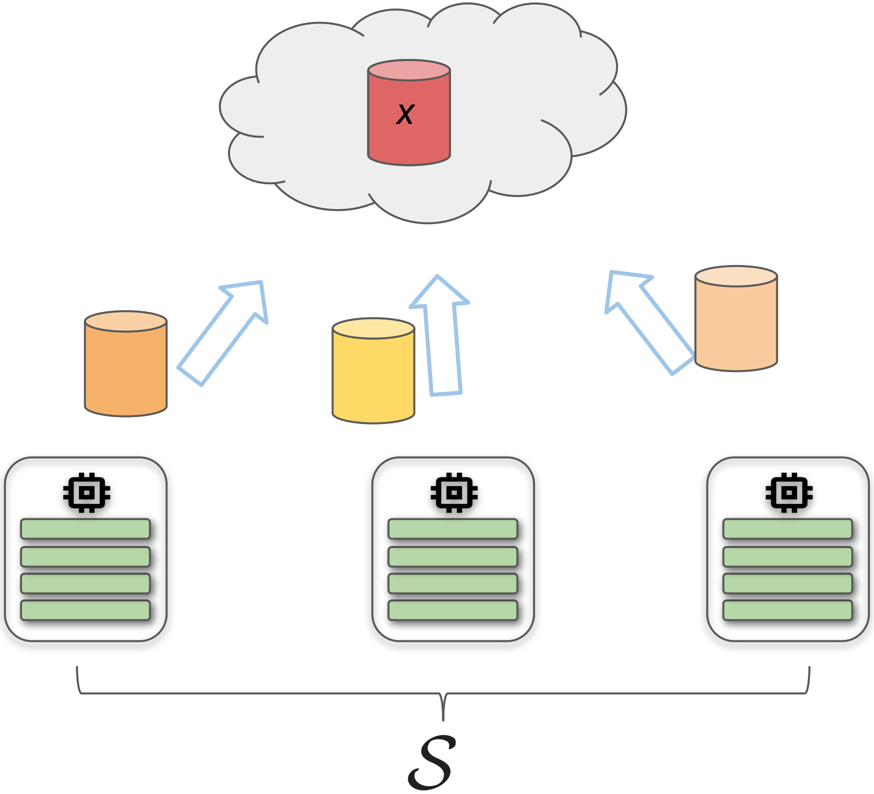


In each round,

- Some subset of clients are chosen
- copy of server model is sent to clients
- model is updated using client data

Federated Learning: Setting

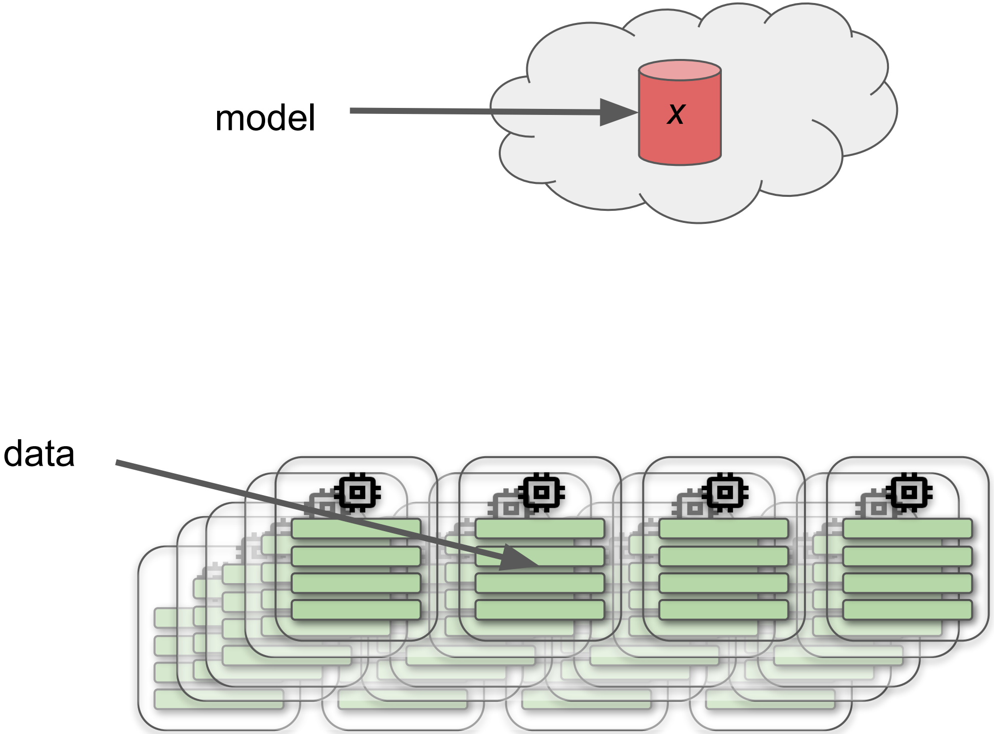
[McMahan et al. 2016]



In each round,

- Some subset of clients are chosen
- copy of server model is sent to clients
- model is updated using client data
- Client updates are aggregated
- server model is updated

Federated Learning: Characteristics



- High overhead per round
- Only a few clients participate in each round
- The data of the clients is ***heterogeneous*****:** i.e. data drawn from different distributions for different clients

Cross-Silo vs Cross-Device Federated Learning

Cross-Silo FL

- **Small/medium** number (2-100, typically) of total clients
- E.g. hospitals, financial organizations
- **Large amount** of data per client
- **Persistent** clients: almost always available
- **Stateful** clients: clients can carry state from round to round

Cross-Device FL

- **Very large** number (e.g. 10^{10}) of total clients
- E.g. mobile or IoT devices
- **Small amount** of data per client
- **Transient** clients: only a fraction of clients available at any time
- **Stateless** clients: clients generally participate only once in each task

Cross-Silo Federated Learning: Formalism

Expectation over client data

$$\min_{x \in R^n} \frac{1}{N} \sum_{i=1}^{N} E_{\zeta}\left[f_i(x;\zeta)\right]$$

Model parameters

average over clients

Loss function wrt parameters and client data

Algorithms for Cross-Silo Federated Learning

Solving FL: SGD

- on each client i in $\mathcal{S}$, compute a large batch stochastic gradient and average them

$$x = x - \boxed{\frac{1}{S}\sum_{i\in\mathcal{S}}}\boxed{\frac{1}{K}\sum_{k=1}^{K} g_i(x)}$$

Average over all sampled clients

Mini-batch gradient with Batch size K per client

\+ Equivalent to synchronous centralized large-batch training

\- very slow (only 1 update)

Solving FL: Federated Averaging (FedAvg) [McMahan et al. 2016]

- on each client i in $\mathcal{S}$, perform K steps of SGD

$$\underbrace{y_i = y_i - \eta g_i(y_i)}_{\text{Repeat K times}}$$

- Server model is an average of client models

$$x = \frac{1}{S} \sum_{i \in \mathcal{S}} y_i$$

Average over all sampled clients

\+ Potentially faster (performs K updates)

\- different from centralized updates

\- may not converge

Solving FL: SCAFFOLD

New!

- on each client i in $\mathcal{S}$, perform K steps of SGD

$$\underbrace{y_i = y_i - \eta(g_i(y_i) + \boxed{c - c_i})}_{\text{Repeat K times}}$$

Correction term!

- Average as before $$x = \frac{1}{S}\sum_{i \in \mathcal{S}} y_i$$

\+ Potentially faster (performs K updates)

\+ mimics centralized updates!

When does FedAvg fail?

FedAvg degrades with heterogeneous clients

Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification

Tzu-Ming Harry Hsu*
MIT CSAIL
stmharry@mit.edu

Hang Qi
Google Research
hangqi@google.com

Matthew Brown
Google Research
mtbr@google.com

Federated Optimization in Heterogeneous Networks

Tian Li
CMU
tianli@cmu.edu

Anit Kumar Sahu
Bosch Center for AI
anit.sahu@gmail.com

Manzil Zaheer
Google Research
manzilz@google.com

Maziar Sanjabi
USC
maziar.sanjabi@gmail.com

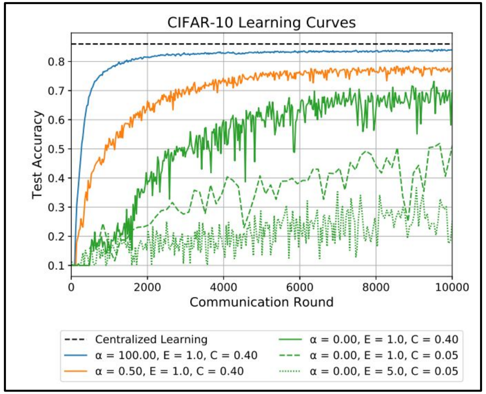
Ameet Talwalkar
CMU & Determined AI
talwalkar@cmu.edu

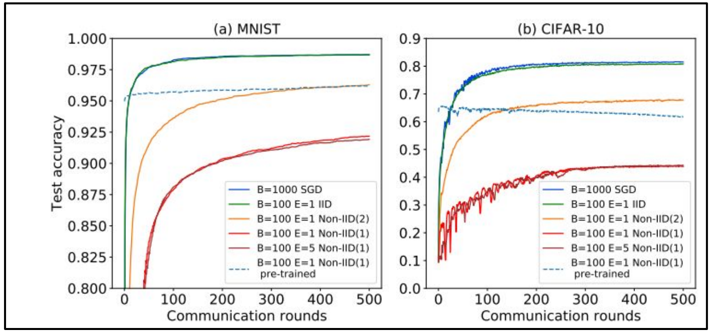
Virginia Smith
CMU
smithv@cmu.edu

Federated Learning with Non-IID Data

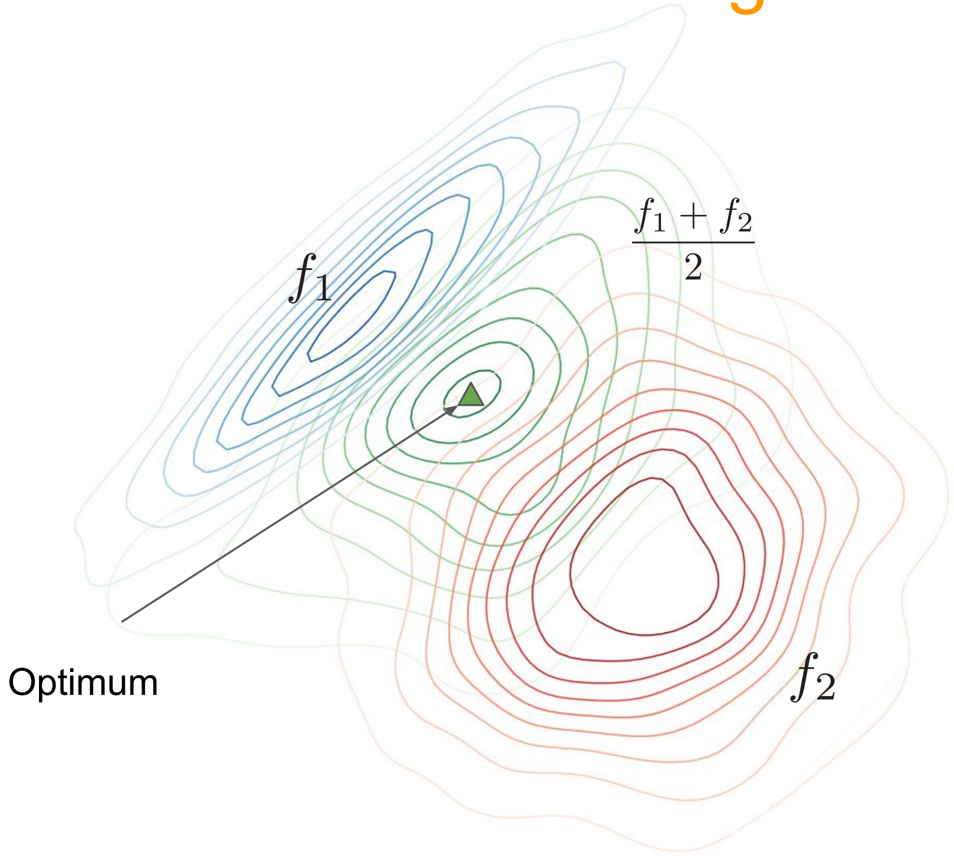
Yue Zhao*, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, Vikas Chandra

Arm, San Jose, CA
{yue.zhao, meng.li, liangzhen.lai, naveen.suda, damon.civin, vikas.chandra}@arm.com



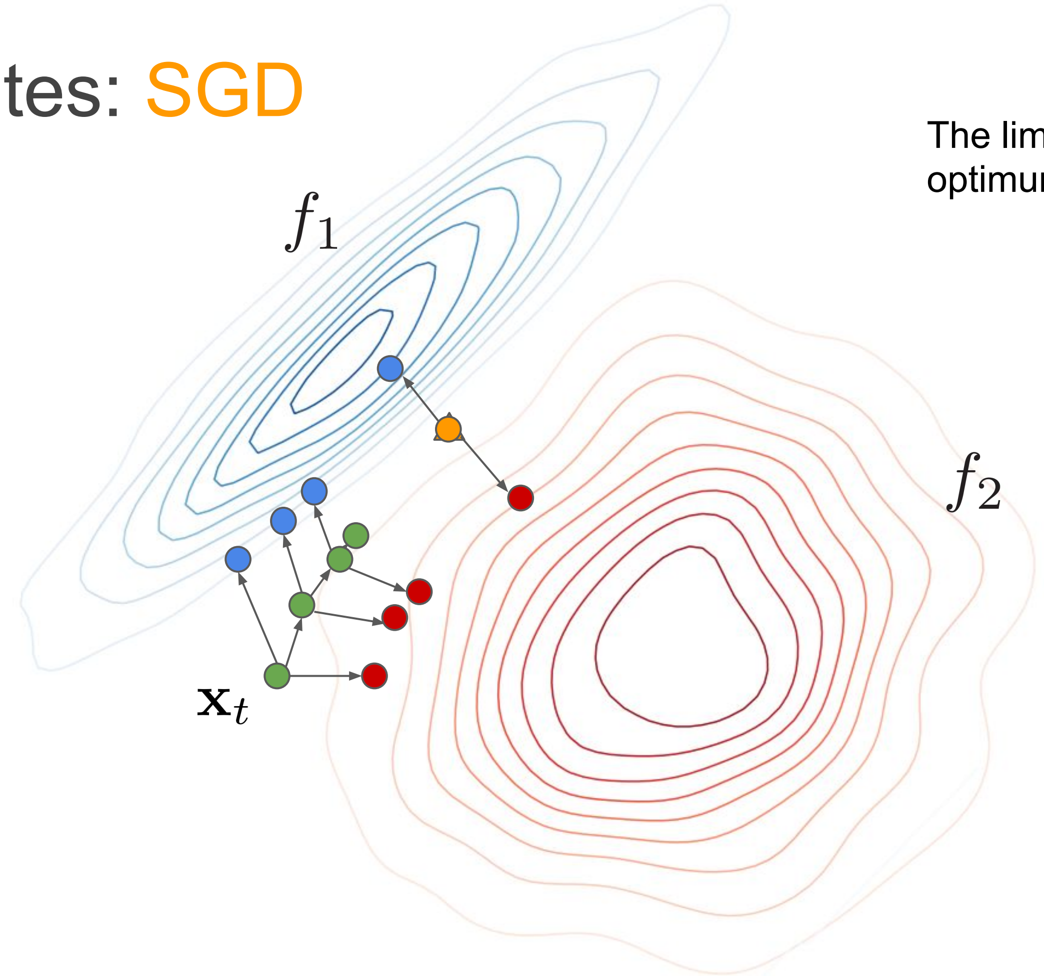


Client updates: SGD vs. FedAvg



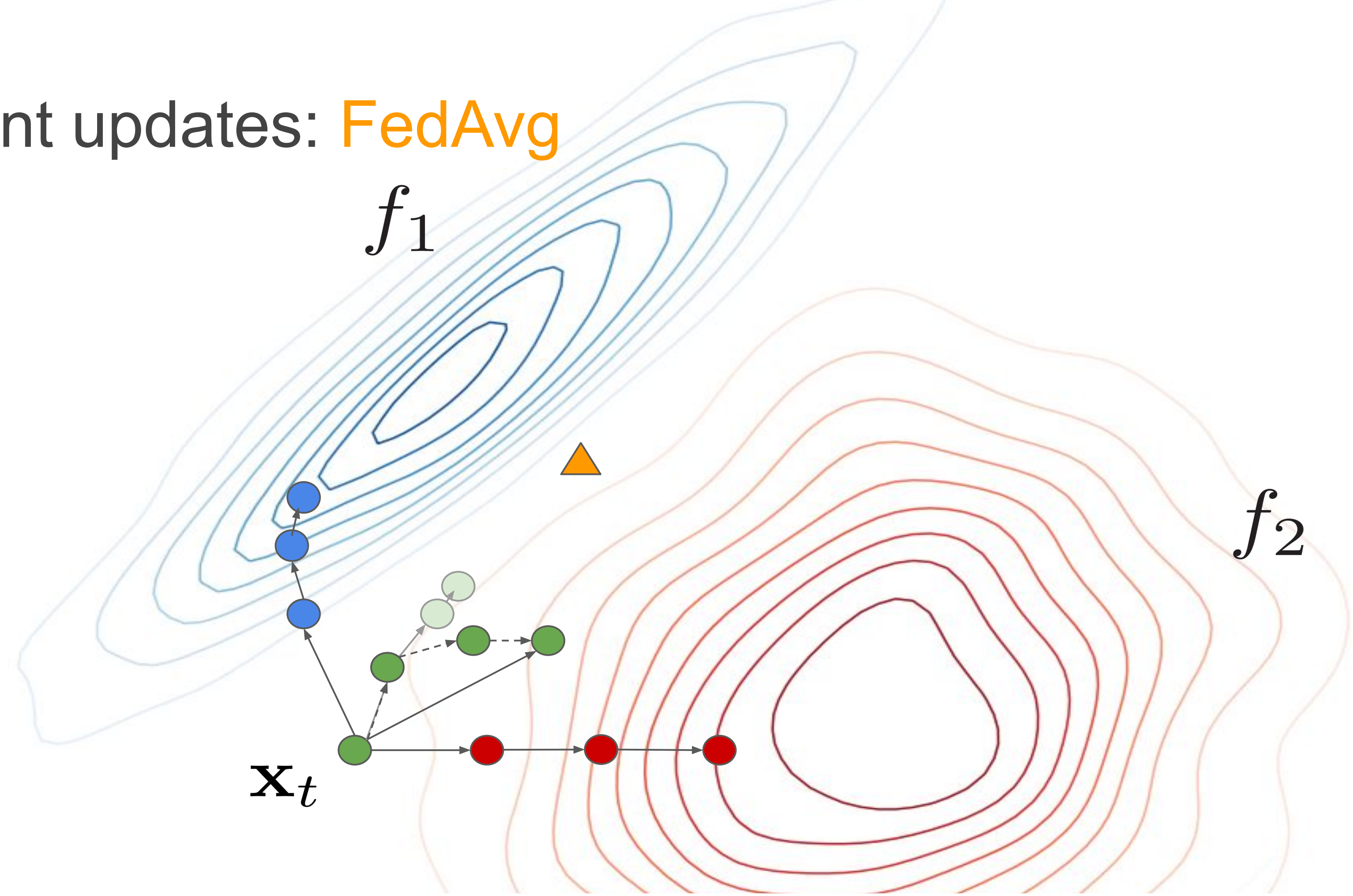
Surface of two client loss functions, and the combined function

Client updates: SGD

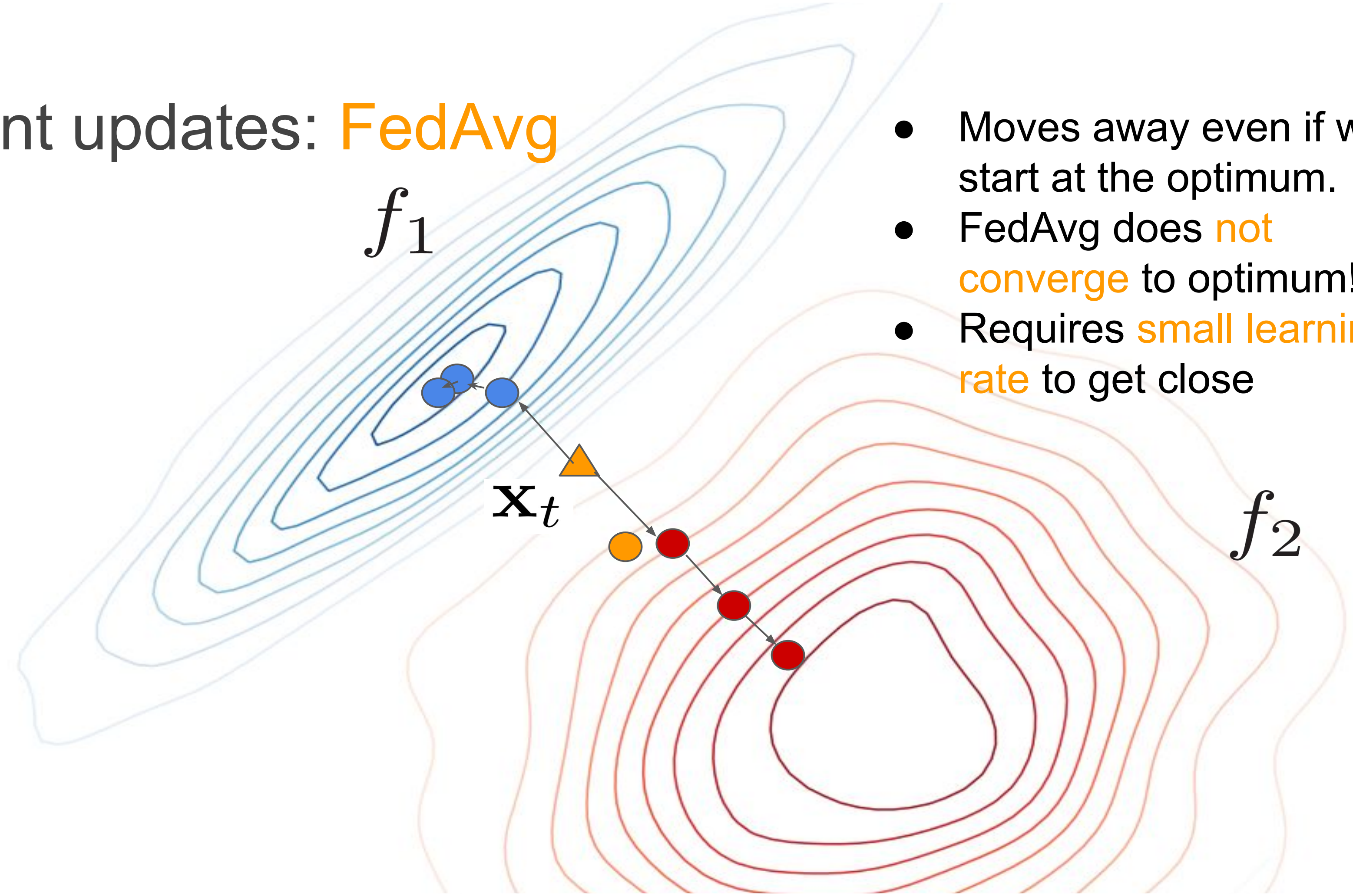


The limit point of SGD is the optimum.

Client updates: FedAvg

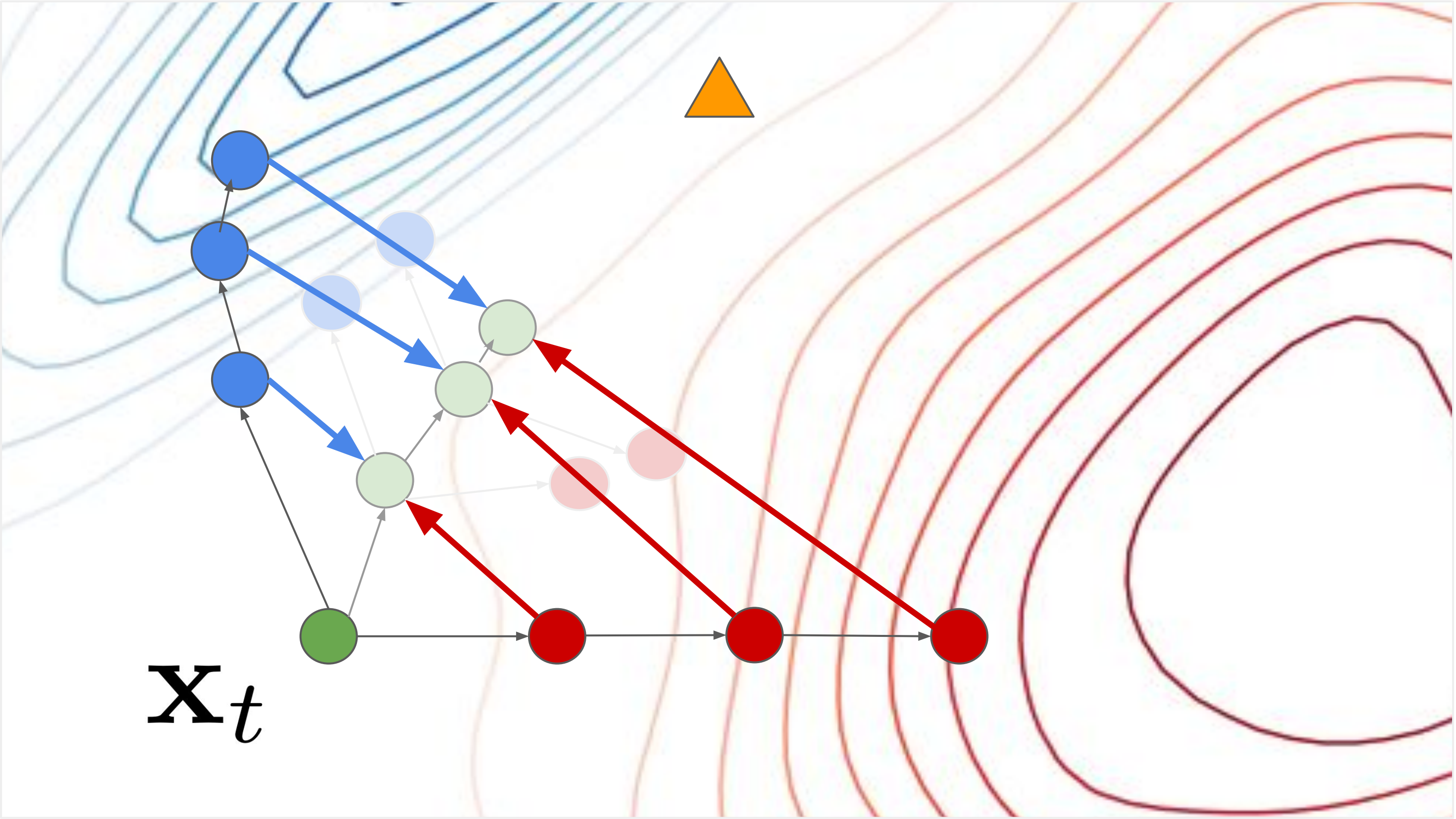


Client updates: FedAvg



- Moves away even if we start at the optimum.
- FedAvg does not converge to optimum!
- Requires small learning rate to get close

Drift in client updates: SGD vs. FedAvg



Convergence Rates: SGD

- For strongly convex

$$\frac{\sigma^2}{\mu RKN} + \exp\left(\frac{-\mu R}{L}\right)$$

- For non-convex functions

$$\frac{\sigma}{\sqrt{RKN}} + \frac{L}{R}$$

Notation:

- R communication rounds
- K local steps
- Total N clients

- L - smooth, μ - strongly convex
- σ - variance within a client

Convergence Rates: SGD vs. FedAvg

Generalizes [Li et al. 2019] and [Khaled et al. 2019]

Assume: (B,G)-similar gradients

$$\mathbb{E}_i \|\nabla f_i(x)\|^2 \leq G^2 + B^2 \|\nabla f(x)\|^2$$

- For strongly convex

SGD

$$\boxed{\mathcal{N} + \exp\left(\frac{-\mu R}{L}\right)}$$

FedAvg

$$\boxed{\mathcal{N} + \frac{LG^2}{\mu^2 R^2} + \exp\left(\frac{-\mu R}{L}\right)}$$

Convergence Rates: SGD vs. FedAvg

SGD

- For strongly convex

$$\boxed{\mathcal{N} + \exp\left(\frac{-\mu R}{L}\right)}$$

- For non-convex functions

$$\boxed{\mathcal{N} + \frac{L}{R}}$$

FedAvg

Assume: (B,G)-similar gradients

$$\boxed{\mathcal{N} + \frac{LG^2}{\mu^2 R^2} + \exp\left(\frac{-\mu R}{L}\right)}$$

Tightest rates, uses server and client step-sizes

$$\boxed{\mathcal{N} + \frac{L^{\frac{1}{3}} G^{\frac{2}{3}}}{R^{\frac{2}{3}}} + \frac{L}{R}}$$

Lower bound: FedAvg

Theorem: For any G, we can find functions with (2, G)-similar gradients such that FedAvg for K>1 *with arbitrary step-sizes* always has error

$$\geq \frac{G^2}{\mu R^2}$$

Assume: (B,G)-similar gradients

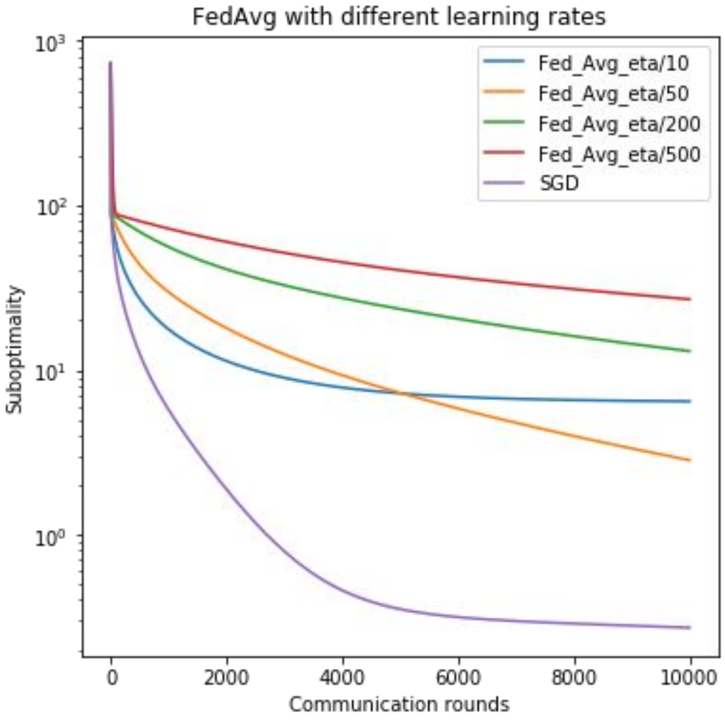
$$\boxed{\mathcal{N} + \boxed{\frac{LG^2}{\mu^2 R^2}} + \exp\left(\frac{-\mu R}{L}\right)}$$

Necessary!

$$\boxed{\mathcal{N} + \frac{L^{\frac{1}{3}} G^{\frac{2}{3}}}{R^{\frac{2}{3}}} + \frac{L}{R}}$$

Quick demo: SGD vs. FedAvg

Lower is better



- Linear regression
- concrete dataset (UCI)
- 10 clients (no sampling)
- K = 10 local steps

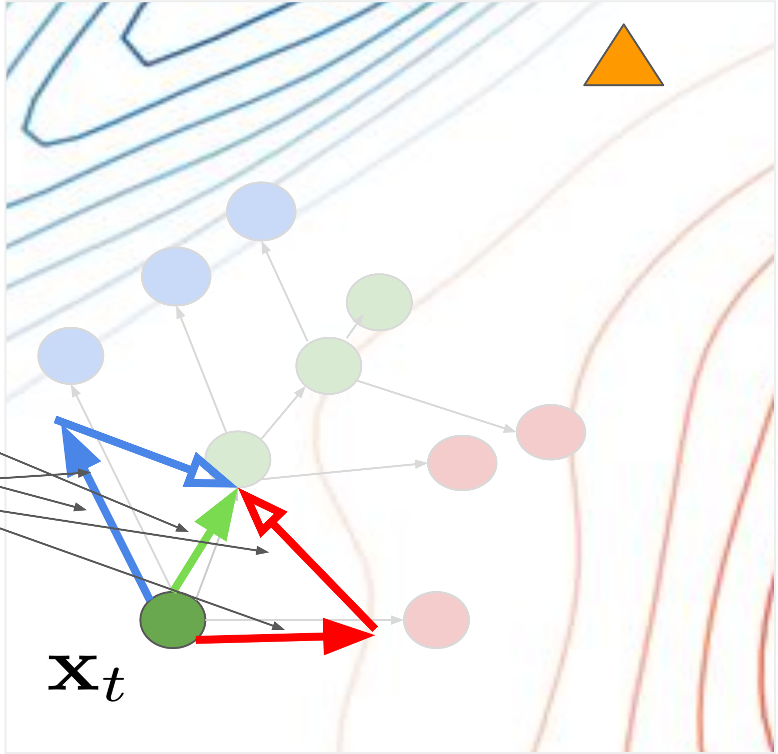
> FedAvg needs smaller learning rate

> Slower than SGD

SCAFFOLD:
stochastic *controlled* averaging

Main Idea: Use control variates

- Guess direction of server update c
- Guess direction of client update c_i
- Use the correction $(c - c_i)$

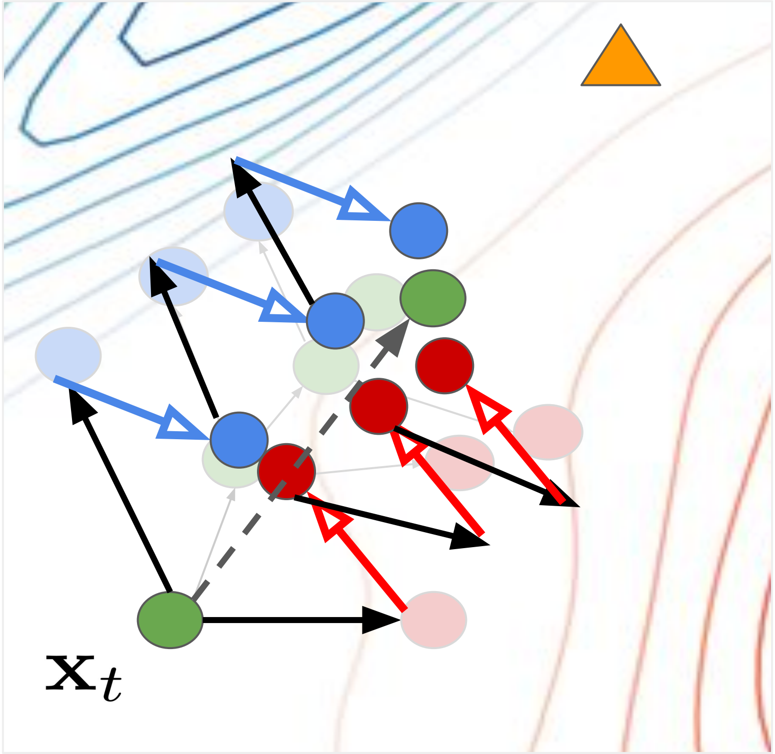


Main Idea: Corrected updates

$$y_i = y_i - \eta(g_i(y_i) + \boxed{c - c_i})$$

Correction terms

Mimics centralized updates!

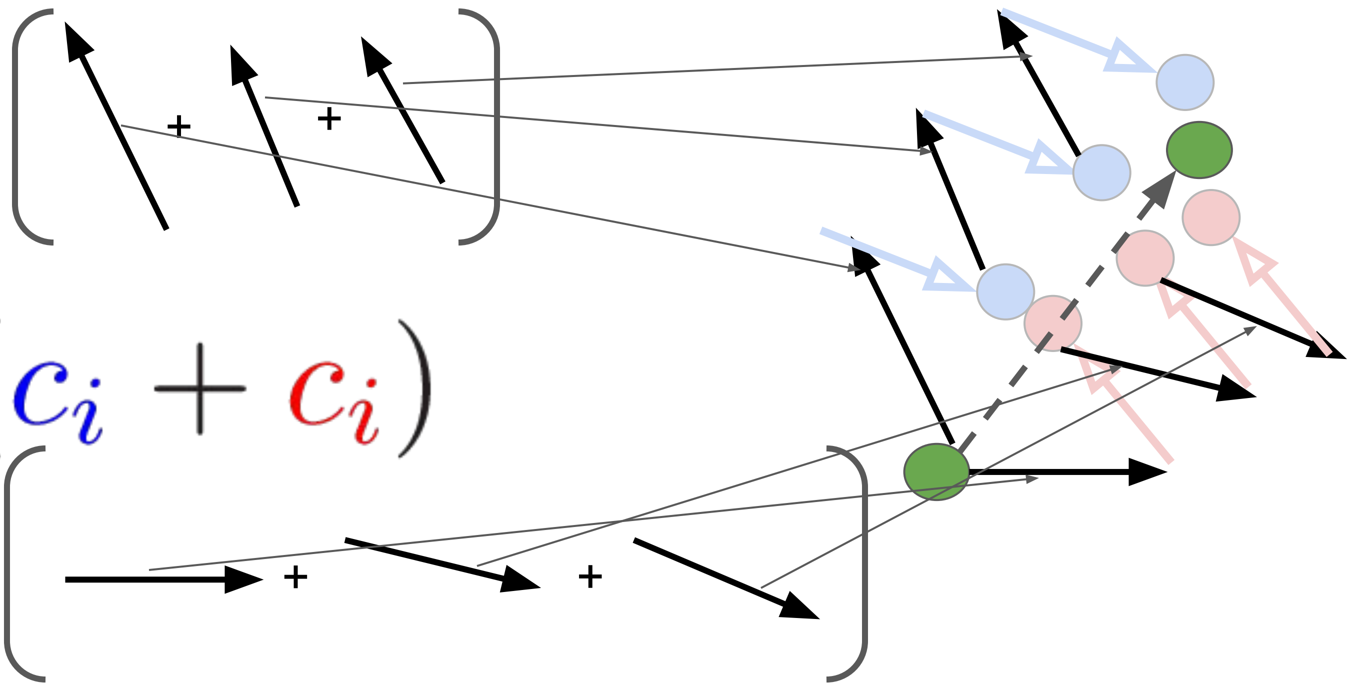


Main Idea: Updating control variates

$$c_i = \frac{1}{K}\left(\quad + \quad + \quad \right)$$

$$c = \frac{1}{2}(c_i + c_i)$$

$$c_i = \frac{1}{K}\left(\quad + \quad + \quad \right)$$



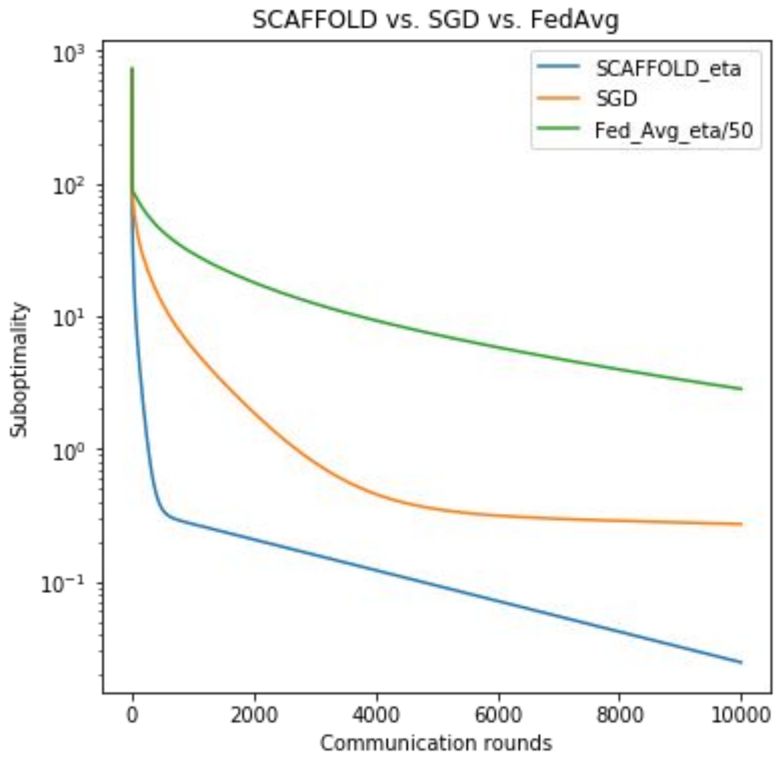
SCAFFOLD: Algorithm

Algorithm 1 SCAFFOLD: Stochastic Controlled Averaging for federated learning

server input initial parameters $\boldsymbol{x}$, control variate $\boldsymbol{c}$, and global step-size η_g
client input for each i local control variate $\boldsymbol{c}_i$, and local step-size η_l
for each communication round $r = 1, \dots, R$ **do**
  select a subset of clients $\mathcal{S} \subseteq \{1, \dots, N\}$
  communicate $(\boldsymbol{x}, \boldsymbol{c})$ to all clients $i \in \mathcal{S}$
  on each client $i \in \mathcal{S}$ **do**
    initialize local parameters $\boldsymbol{y}_i \leftarrow \boldsymbol{x}$
    for each local step $k = 1, \dots, K$ **do**
      compute a stochastic gradient $g_i(\boldsymbol{y}_i)$ of f_i
      $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta_l \left(g_i(\boldsymbol{y}_i) - \boldsymbol{c}_i + \boldsymbol{c}\right)$ ▷ local updates with correction
    end for
    $\boldsymbol{c}_i^+ \leftarrow$ (i) $g_i(\boldsymbol{x})$, or (ii) $\boldsymbol{c}_i - \boldsymbol{c} + \frac{1}{K\eta_l}(\boldsymbol{x} - \boldsymbol{y}_i)$ ▷ compute new control variate
    communicate $(\Delta \boldsymbol{y}_i, \Delta \boldsymbol{c}_i) \leftarrow (\boldsymbol{y}_i - \boldsymbol{x}, \boldsymbol{c}_i^+ - \boldsymbol{c}_i)$
    $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i^+$ ▷ update control variate
  end client
  $\Delta \boldsymbol{x} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta \boldsymbol{y}_i$ and $\Delta \boldsymbol{c} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta \boldsymbol{c}_i$ ▷ aggregate client outputs
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \eta_g \Delta \boldsymbol{x}$ and $\boldsymbol{c} \leftarrow \boldsymbol{c} + \frac{|\mathcal{S}|}{N} \Delta \boldsymbol{c}$ ▷ update parameters and control
end for

SCAFFOLD: Quick demo

Lower is better



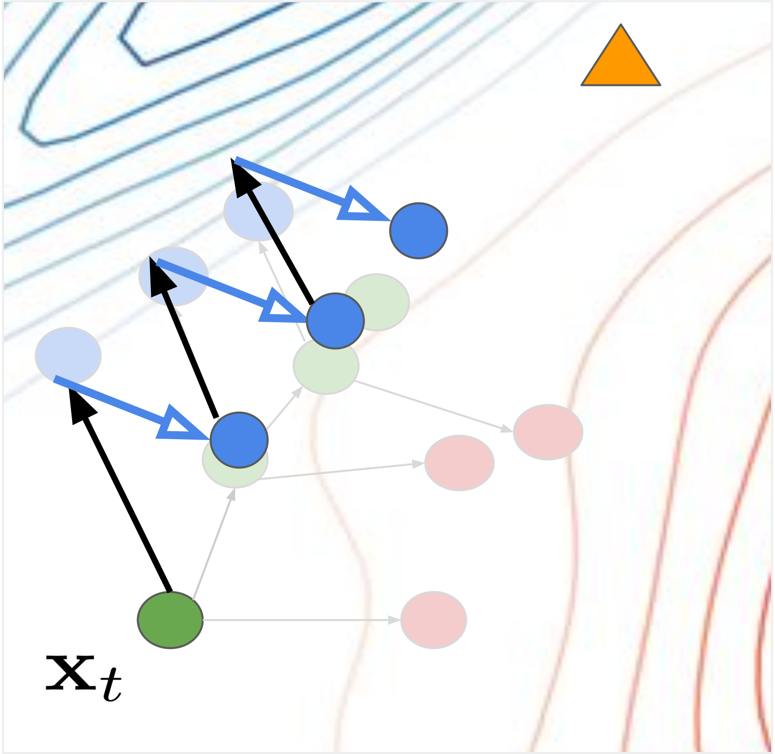
- Linear regression
- concrete dataset (UCI)
- 10 clients (no sampling)
- K = 10 local steps

> SCAFFOLD works with same learning rate as SGD

> Faster than SGD!

SCAFFOLD: Client sampling

- Updates of every client mimics centralized updates.
- Few #clients works, as long as control variates are accurate.
- Hence, very robust to client sampling.



Different view: SAGA is a special case of SCAFFOLD with client sampling

SCAFFOLD: Variance reduced convergence Rates

- For strongly convex functions

$$\boxed{\mathcal{N} + \exp\left(-\min\left\{\frac{\mu}{L}, \frac{S}{N}\right\} R\right)}$$

- For non-convex functions

$$\boxed{\mathcal{N} + \left(\frac{N}{S}\right)^{\frac{2}{3}} \frac{L}{R}}$$

Notation:

- R communication rounds
- S out of N clients sampled
- L - smooth, μ - strongly convex

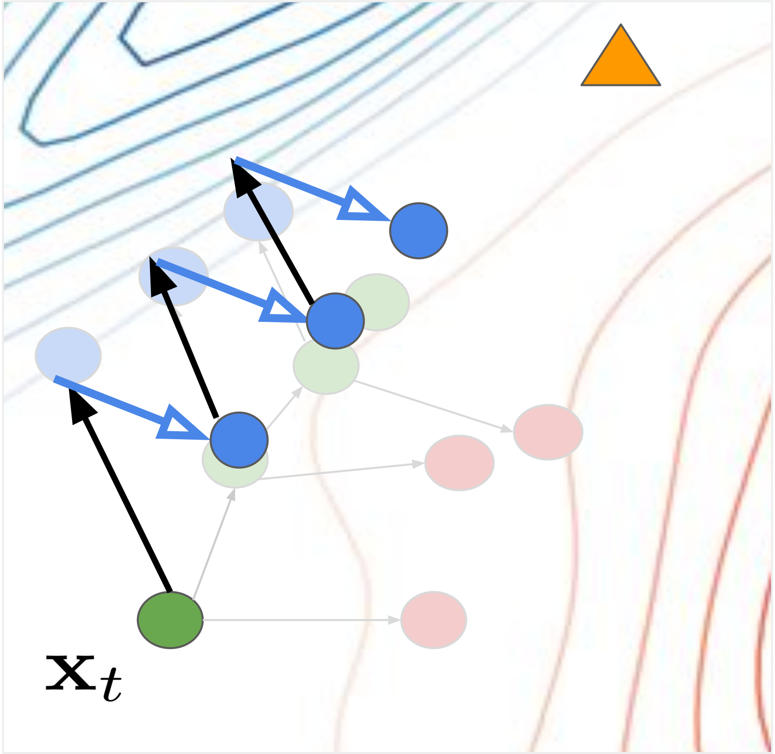
> Better than FedAvg!

SCAFFOLD: Why take more than 1 step?

- **Each update** mimics a centralized update => local steps should help.

- In worst case not true [Arjevani & Shamir, 2015] :(

- Possible if similar Hessians!



SCAFFOLD: Why take more than 1 step?

δ - BHD (Bounded Hessian Dissimilarity)

$$\boxed{\|\nabla^2 f_i(x) - \nabla^2 f(x)\| \leq \delta}$$

And is δ-weakly convex.

Note that

$$\delta \leq L\,, \ \text{and typically } \delta \ll L$$

SCAFFOLD: Why take more than 1 step?

Assume: δ - BHD, quadratics

- For strongly convex functions

$$\boxed{\mathcal{N} + \exp\left(\frac{-\mu R K}{L + (\mu + \delta) K}\right) \approx \mathcal{N} + \exp\left(\frac{-\mu R}{\mu + \delta}\right)}$$

- For non-convex functions

$$\boxed{\mathcal{N} + \frac{L + \delta K}{R K} \approx \mathcal{N} + \frac{\delta}{R}}$$

Notation:

- R communication rounds
- All N clients participate
- K local steps

- L - smooth, μ - strongly convex

SCAFFOLD: Why take more than 1 step?

Assume: δ - BHD, quadratics

> Best to take $K \approx \frac{L}{\delta}$

- For strongly convex functions

$$\boxed{\mathcal{N} + \exp\left(\frac{-\mu R K}{L + (\mu + \delta) K}\right) \approx \mathcal{N} + \exp\left(\frac{-\mu R}{\mu + \delta}\right)}$$

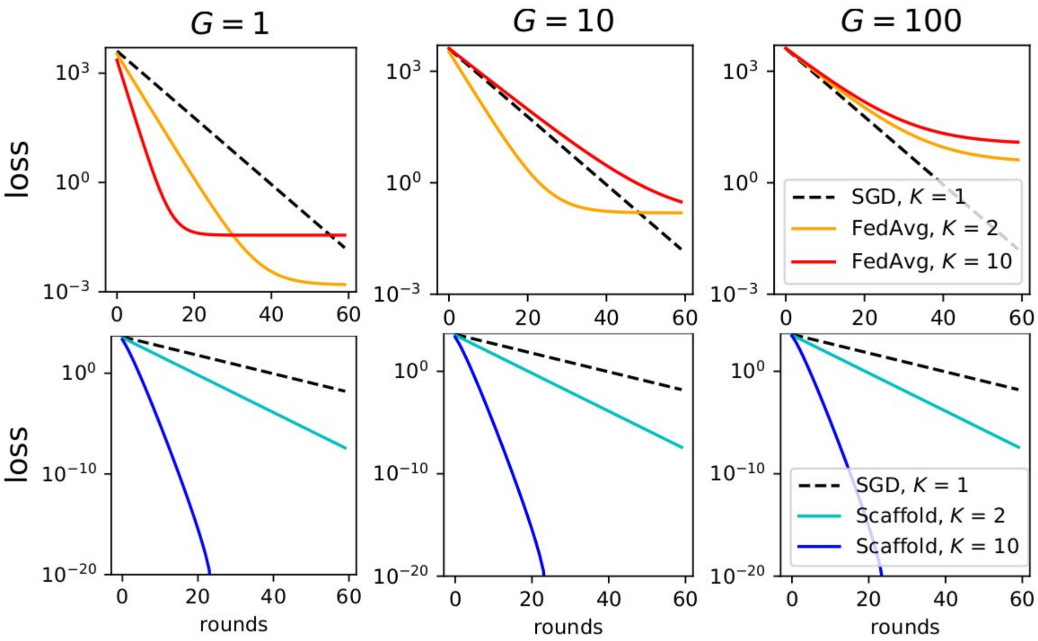
> We replaced L with δ in the rates (typically δ << L)

- For non-convex functions

$$\boxed{\mathcal{N} + \frac{L + \delta K}{R K} \approx \mathcal{N} + \frac{\delta}{R}}$$

> First rate to characterize improvement due to local steps!

SCAFFOLD: Why take more than 1 step?



Quick demo on scalar quadratics

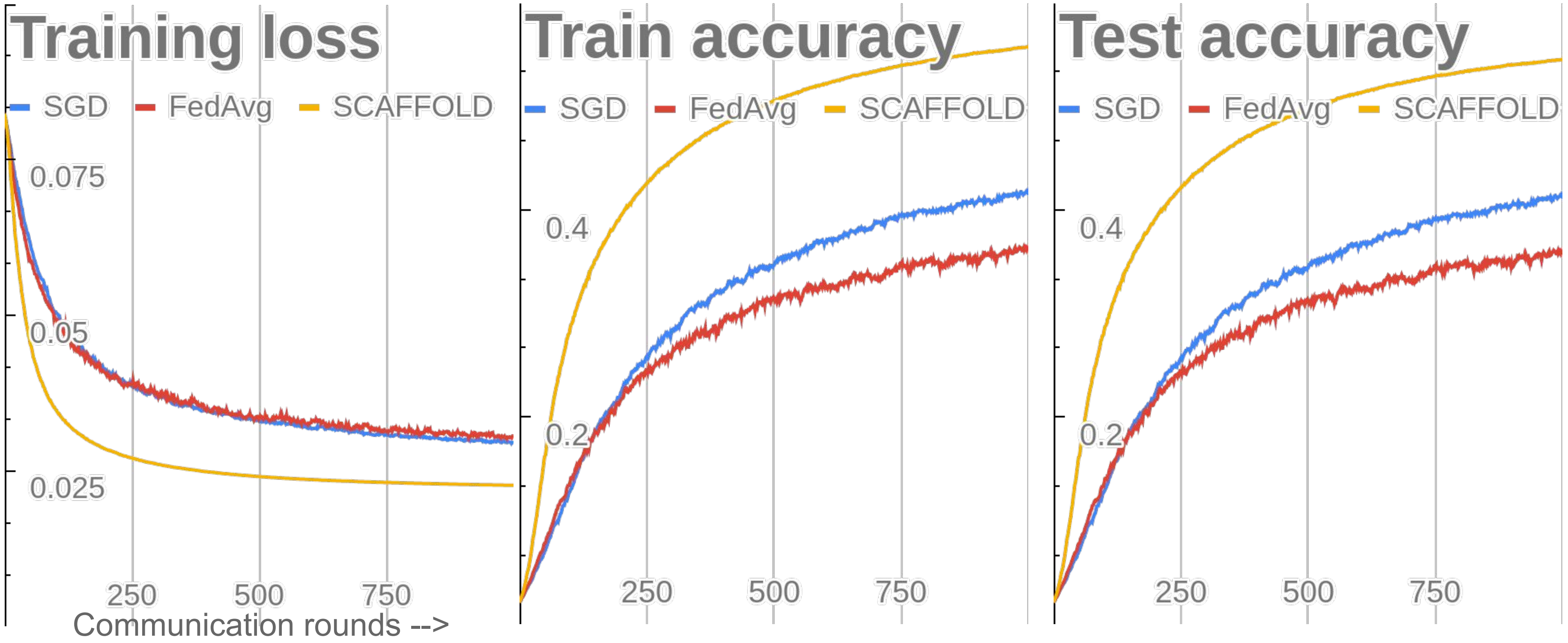
- Scaffold is unaffected by G
- Larger K is better
- K=2 is 2 times faster
- K=10 is only 4 times better

Experiments

Experimental Setup

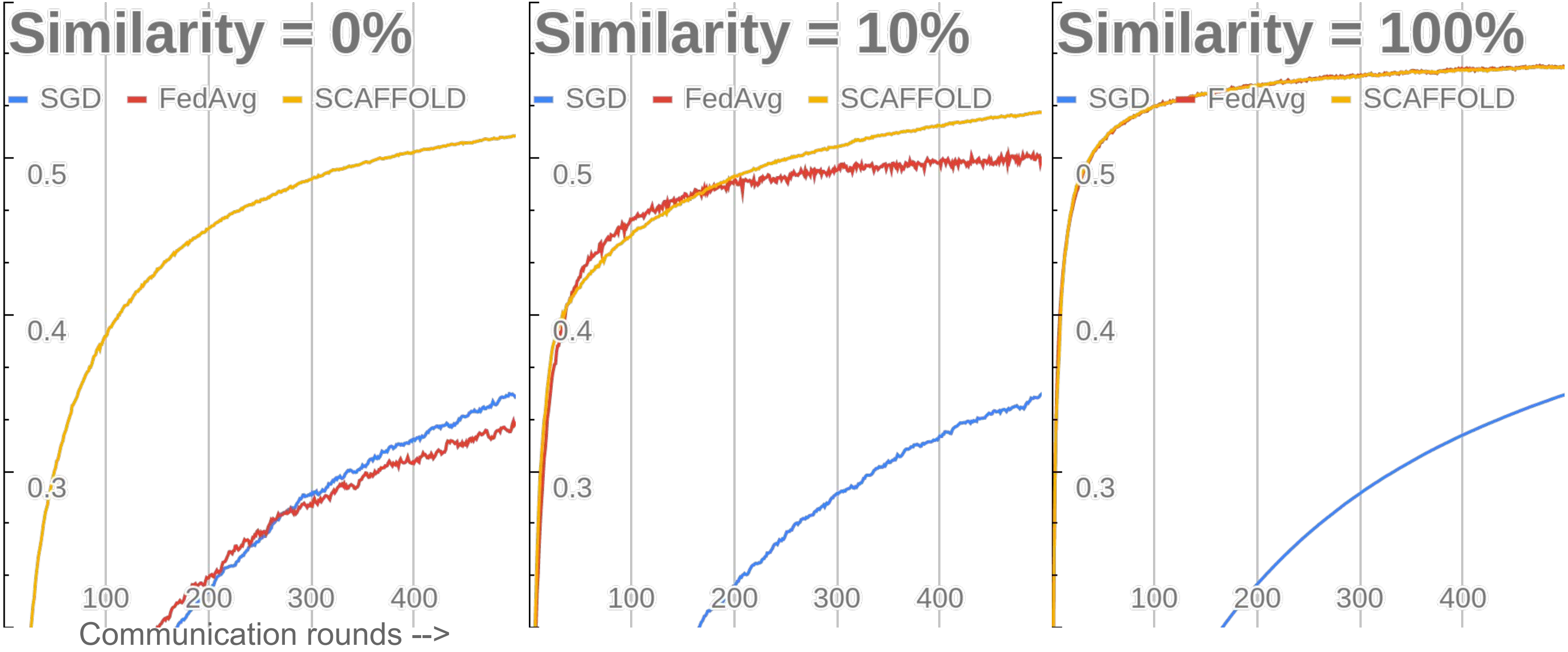
- Extended MNIST (balanced) dataset
- Multi-class logistic regression (47 classes)
- Partitioned into N clients
- Sorted by labels and then ‘slightly shuffled’ before splitting

Performance of SCAFFOLD



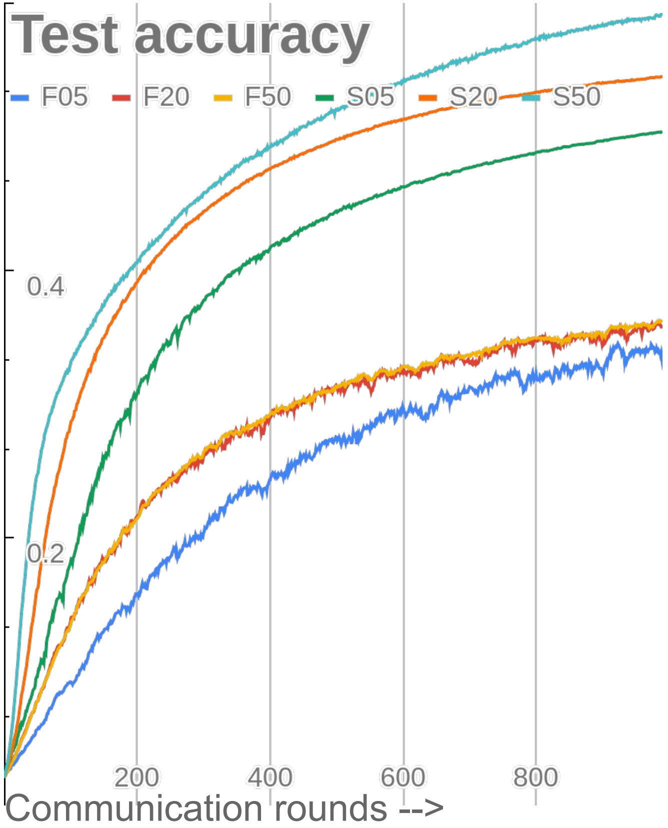
Similarity = 0, 1 Epoch, #sampled clients = 20, total clients = 400

Effect of similarity



Test accuracy, 10 Epochs, #sampled clients = 20, total clients = 100

Effect of number of clients



SCAFFOLD with 5 clients is better than FedAvg with 50!

> Total #clients = 400
> Total #categories = 47
> 1 Epoch per round
> Similarity = 0

Take aways

- Degradation of FedAvg is due to the client drift. If you use FedAvg, use separate server and client step-sizes.
- Why you should use SCAFFOLD:
  - Provably converges faster than SGD and FedAvg
  - Resilient to heterogeneity and client sampling
- **Main limitation:** requires maintaining client state, so applicable only to cross-silo FL

Cross-Silo vs Cross-Device Federated Learning

Cross-Silo FL

- **Small/medium** number (2-100, typically) of total clients
- E.g. hospitals, financial organizations
- **Large amount** of data per client
- **Persistent** clients: almost always available
- **Stateful** clients: clients can carry state from round to round

Cross-Device FL

- **Very large** number (e.g. 10^{10}) of total clients
- E.g. mobile or IoT devices
- **Small amount** of data per client
- **Transient** clients: only a fraction of clients available at any time
- **Stateless** clients: clients generally participate only once in each task

Cross-device Federated Learning: Formalism

Sum over client data

$$\min_x \mathbb{E}_{i \sim \mathcal{D}} \left[f_i(x) := \frac{1}{n_i} \sum_{\nu=1}^{n_i} f_i(x; \xi_\nu) \right]$$

Model parameters

Expectation over (possibly infinitely many) clients

Client loss function over the parameters and data

Algorithms for Cross-Device Federated Learning

Solving FL: SGD with (server) momentum

[Assume only 1 client per round]

$$x = x - \eta * (\nabla f_i(x) + \beta * m)$$

Update server parameters

$$m = \nabla f_i(x) + \beta m$$

Update server momentum

+ Convergence guaranteed

- Communicates every update round

Solving FL: SGD with server momentum

[Assume only 1 client per round]

$$y_i = y_i - \eta \nabla f_i(y_i)$$

$$x \leftarrow x - \eta(\nabla f_i(y_i) + \beta m)$$

Update server parameters

$$m = \nabla f_i(x) + \beta m$$

Update server momentum

+ Convergence guaranteed

- Communicates every update round

Solving FL: FedAvg with momentum

[McMahan et al. 2016,
Hsu et al. 2019,
Reddi et al. 2020]

- Starting from x, run K local updates

$$\underbrace{y_i = y_i - \eta \nabla f_i(y_i)}_{\text{Repeat K times}}$$

+ Communicates only every K updates

- Use (x - y_i) as a pseudo-gradient.

$$x = x - \eta * (\{x - y_i\} + \beta * m)$$

Update server parameters

- bad convergence due to client drift (though momentum helps!)

Solving FL: Mime with momentum

- Apply server momentum locally in the clients

$$\underbrace{y_i = y_i - \eta * (\nabla f_i(y_i) + \underbrace{\boxed{\beta * m}}_{\text{Fixed server momentum}})}_{\text{Repeat K times}}$$

- Momentum is computed globally (at server) and applied locally (at clients)

$$m = \nabla f_i(x) + \beta m$$

Update server momentum

+ Communicates only every K updates
+ Reduce client drift using (fixed) server momentum!
+ Extends to Adam, etc.

Solving FL: Mime with momentum

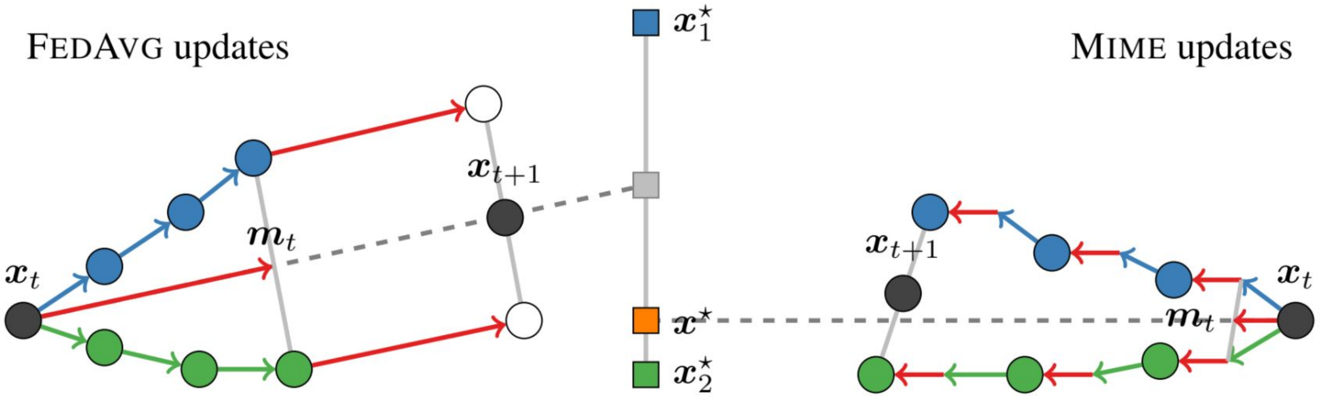


Figure 1: Client-drift in FEDAVG (left) and MIME (right) is illustrated for 2 clients with 3 local steps and momentum parameter $\beta = 0.5$. The local SGD updates of FEDAVG (shown using arrows for client 1 and client2) move towards the average of client optima $\frac{x_1^\star + x_2^\star}{2}$ which can be quite different from the true global optimum $x^\star$. Server momentum only speeds up the convergence to the wrong point in this case. In contrast, MIME uses unbiased momentum and applies it locally at every update. This keeps the updates of MIME closer to the true optimum $x^\star$.

Mime framework: adapting optimizers to FL setting

Base optimizer updates: given gradient ***g*** and current internal state ***s*** (e.g. momentum, Adagrad/Adam accumulators, etc.)

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\,\mathcal{U}(\boldsymbol{g}, \boldsymbol{s}),$$
$$\boldsymbol{s} \leftarrow \mathcal{V}(\boldsymbol{g}, \boldsymbol{s}).$$

E.g. SGD with momentum:

$$x \leftarrow x - \eta(g + \beta m)$$
$$m \leftarrow g + \beta m$$

Mime server update: update state ***s*** using gradients from clients:

$$\boldsymbol{s} \leftarrow \mathcal{V}\Big(\tfrac{1}{|\mathcal{S}|}\textstyle\sum_{i\in\mathcal{S}} \nabla f_i(\boldsymbol{x}),\ \boldsymbol{s}\Big)$$

Mime*lite* client update: obtain ***x*** and state ***s*** from server and repeat K times starting from ***y*** = ***x***:

$$y \leftarrow y - \eta\,\mathcal{U}\Big(\nabla f_i(y, \xi), s\Big)$$

Mime framework: adapting optimizers to FL setting

Base optimizer updates: given gradient ***g*** and current internal state ***s*** (e.g. momentum, Adagrad/Adam accumulators, etc.)

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\,\mathcal{U}(\boldsymbol{g}, \boldsymbol{s})\,,$$
$$\boldsymbol{s} \leftarrow \mathcal{V}(\boldsymbol{g}, \boldsymbol{s})\,.$$

Mime server update: update state ***s*** using gradients from clients:

$$\boldsymbol{s} \leftarrow \mathcal{V}\Big(\tfrac{1}{|\mathcal{S}|}\textstyle\sum_{i\in\mathcal{S}} \nabla f_i(\boldsymbol{x}),\ \boldsymbol{s}\Big)$$

Mime client update: obtain ***x*** and state ***s*** from server and repeat K times starting from ***y*** = ***x***:

$$y \leftarrow y - \eta\,\mathcal{U}\Big(\nabla f_i(y,\xi) - \nabla f_i(x,\xi) + \tfrac{1}{S}\textstyle\sum_{j\in S}\nabla f_j(x), s\Big)$$

Algorithm 1 **Mime** and **MimeLite**

input: initial $\boldsymbol{x}$ and $\boldsymbol{s}$, learning rate η and base algorithm $\mathcal{B} = (\mathcal{U}, \mathcal{V})$
for each round $t = 1, \cdots, T$ **do**
  sample subset $\mathcal{S}$ of clients
  communicate $(\boldsymbol{x}, \boldsymbol{s})$ to all clients $i \in \mathcal{S}$
  communicate $\boldsymbol{c} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(\boldsymbol{x})$ (only for Mime)
  on client $i \in \mathcal{S}$ **in parallel do**
    initialize local model $\boldsymbol{y}_i \leftarrow \boldsymbol{x}$
    for $k = 1, \cdots, K$ **do**
      sample mini-batch ζ from local data
      $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta \mathcal{U}(\nabla f_i(\boldsymbol{y}_i; \zeta) - \nabla f_i(\boldsymbol{x}; \zeta) + \boldsymbol{c}, \boldsymbol{s})$ **(Mime)**
      $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta \mathcal{U}(\nabla f_i(\boldsymbol{y}_i; \zeta), \boldsymbol{s})$ **(MimeLite)**
    end for
    compute full local-batch gradient $\nabla f_i(\boldsymbol{x})$
    communicate $(\boldsymbol{y}_i, \nabla f_i(\boldsymbol{x}))$
  end on client
  $\boldsymbol{x} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \boldsymbol{y}_i$, and $\boldsymbol{s} \leftarrow \mathcal{V}\left(\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(\boldsymbol{x}),\ \boldsymbol{s}\right)$
end for

Analysis

Client data

$$\min_{x} \mathbb{E}_{i\sim\mathcal{D}} \left[f_i(x) := \frac{1}{n_i} \sum_{\nu=1}^{n_i} f_i(x;\xi_\nu) \right]$$

parameters clients loss function

- **G²** - Bounded Gradient dissimilarity:

$$\mathbb{E}_{i\sim\mathcal{D}} \|\nabla f_i(x) - \nabla f(x)\|^2 \leq G^2$$

- □- Bounded Hessian dissimilarity:

$$\|\nabla^2 f_i(x;\xi_i) - \nabla^2 f(x)\| \leq \delta$$

Convergence rates

Algorithm	Non-convex	μ-Strongly convex	Assumptions
SERVER-ONLY			
SGD [8]	$\frac{G^2}{S\epsilon^2} + \frac{L}{\epsilon}$	$\frac{G^2}{\mu S\epsilon} + \frac{L}{\mu}$	G^2-BGD
MVR [4]	$(\frac{G}{\sqrt{S\epsilon}})^{\frac{3}{2}} + \frac{L}{\epsilon}$	–	G^2-BGD
FEDAVG [1]			
FedSGD [15]	$\frac{G^2}{S\epsilon^2} + \frac{G}{\epsilon^{3/2}} + \frac{L}{\epsilon}$	$\frac{G^2}{\mu S\epsilon} + \frac{G}{\mu\sqrt{\epsilon}} + \frac{L}{\mu}$	G^2-BGD
MIME [2]			
MimeSGD	$\frac{G^2}{S\epsilon^2} + \frac{\delta}{\epsilon}$	$\frac{G^2}{\mu S\epsilon} + \frac{\delta}{\mu}$	G^2-BGD, δ-BHD
MimeMVR	$(\frac{G}{\sqrt{S\epsilon}})^{\frac{3}{2}} + \frac{\delta}{\epsilon}$	–	G^2-BGD, δ-BHD
Lower bound [1]	$\Omega(\frac{G}{\sqrt{S\epsilon}})^{\frac{3}{2}}$	$\Omega(\frac{G^2}{S\epsilon})$	G^2-BGD

Experiments

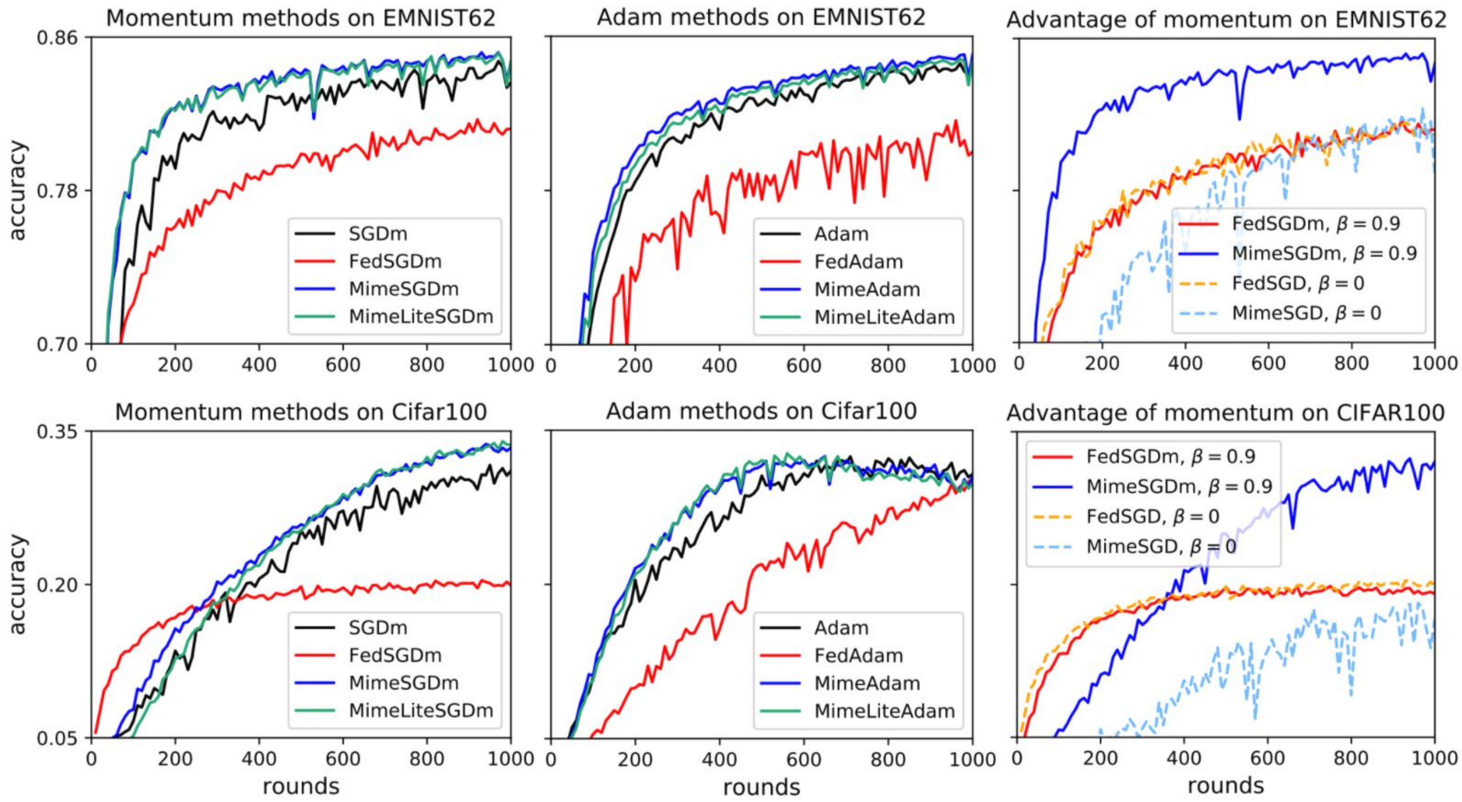


Figure 3: **Server-only**, FedAvg, Mime, and MimeLite with SGDm (left) and Adam (middle) run on EMNIST62 (top) and CIFAR100 (bottom). Mime and MimeLite have very similar performance and are consistently the best. FedAvg is often even worse than the server-only baselines. Also, Mime makes better use of momentum than FedAvg, with a large increase in performance (right).

Takeaways

- Momentum injects global information and helps reduce client drift.

- Compute momentum globally at server, apply it during each client update.

- Usefulness of local steps depends on Hessian variance.

Thank You.

Questions?