# On Personalized Asynchrony in Distributed Learning

**ELLIIT Seminar 2023**

Taha Toghani, Soomin Lee, César A. Uribe
mttoghani@gmail.com

RICE ENGINEERING
Electrical and Computer Engineering

yahoo!

# On Personalized Asynchrony in Distributed Learning

**CCDC Seminar 2023**

Taha Toghani, Soomin Lee, César A. Uribe
mttoghani@gmail.com

RICE ENGINEERING
Electrical and Computer Engineering

yahoo!

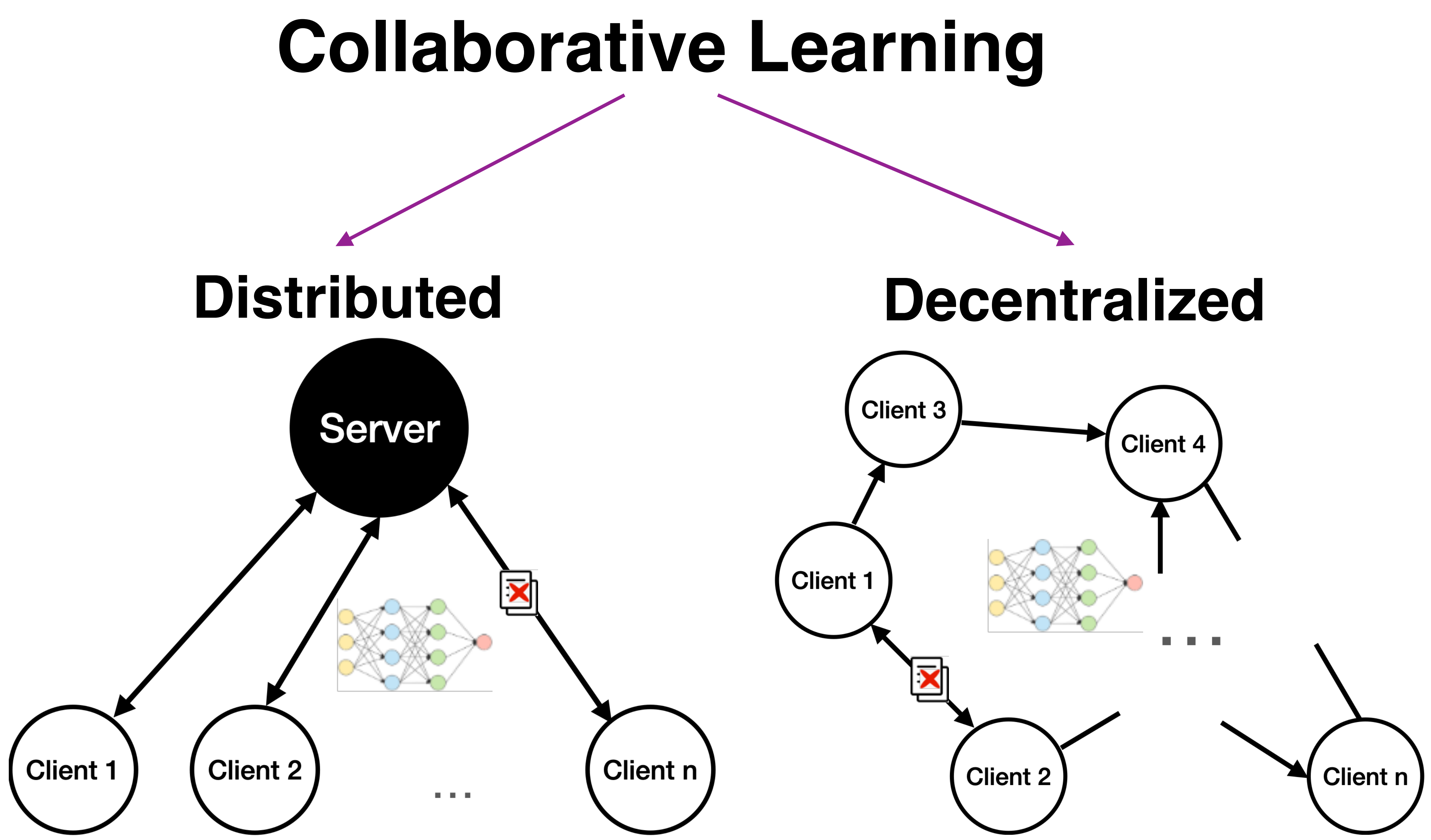
# Motivation

Advantages:

- distributed data
- parallel processing
- privacy preservation
- personalization
- physical constraints

Challenges:

- connection failures
- data heterogeneity
- adversarial attacks
- scalability
- server failure
- directed communications
- communication-efficiency




$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^d} \left[ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right]$$

# Plan for Today

- PART I: Federated Learning, Personalization & Asynchrony
- PART II: Decentralized Learning & Robustness, Model Agnostic Meta-Learning
- PART II: Reinforcement Learning & Moreau Envelopes

# PART I

## PersA-FL: Personalized Asynchronous ~~Federated~~ Distributed Learning

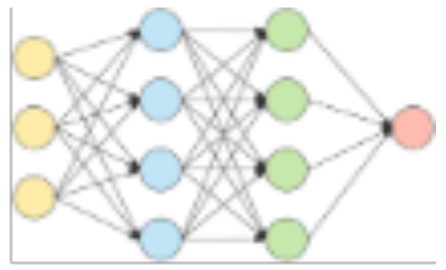
# ~~Federated~~ Distributed Learning

Challenges:
- data heterogeneity
- asynchronous communications

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^d} \left[ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right]$$

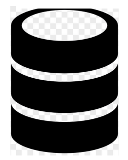
number of clients

local cost function



number of parameters

$$f_i(\mathbf{x}) = \mathbb{E}_{\xi_i \sim p_i}[\ell_i(w, \xi_i)]$$



local distribution

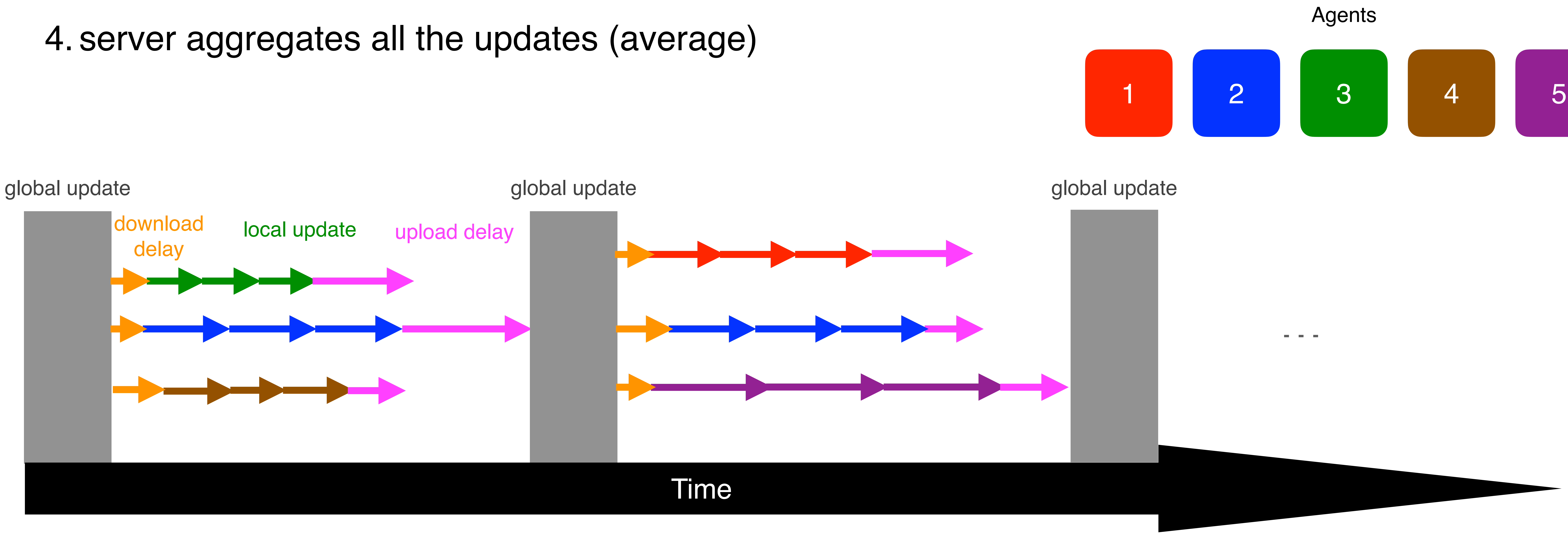
- central server
- parameters
- heterogeneous distributions

Stochastic cost over data batch $\mathcal{D}_i$:

$$\tilde{f}_i(w, \mathcal{D}_i) := \frac{1}{|\mathcal{D}_i|}\sum_{\xi_i \in \mathcal{D}_i} \ell_i(w, \xi_i)$$

# FedAvg Algorithm

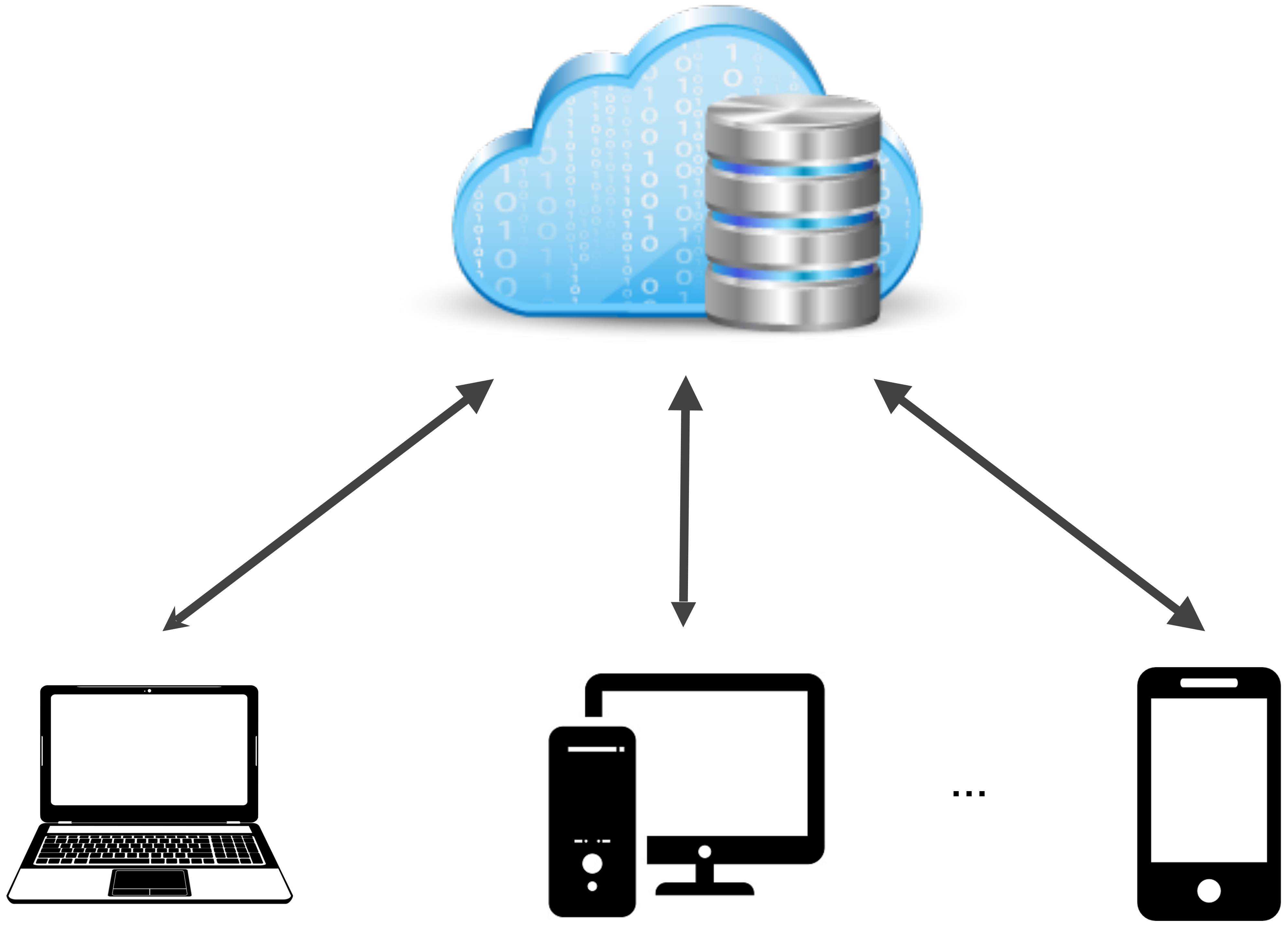
1. server sends its current set of parameters to a subset of clients
2. selected client $i$ perform $Q$ steps of local updates (stochastic gradient descent) on $f_i$
3. server waits to receive all local updates back
4. server aggregates all the updates (average)



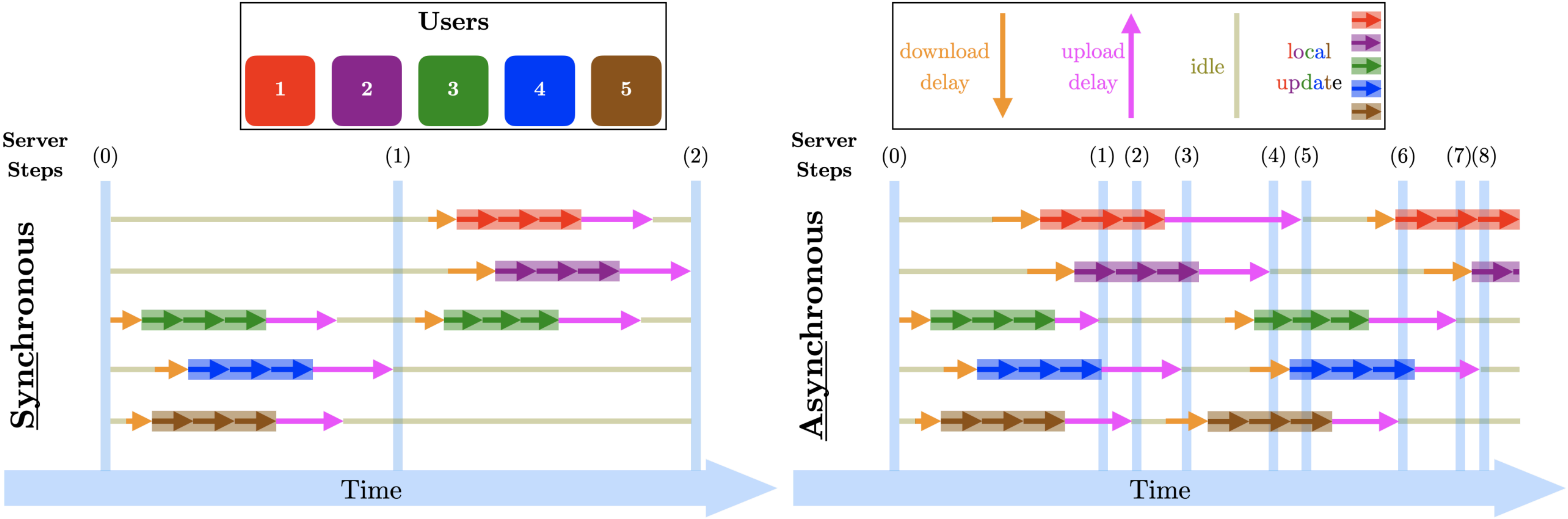
# Asynchronous Communications

Limitations of synchronous algorithms:

- limited bandwidth
- different delays
- parallel communication
- connection reliability
- unavailability



# Communication & Update Schedule



# Personalization

- Why do we need personalization?

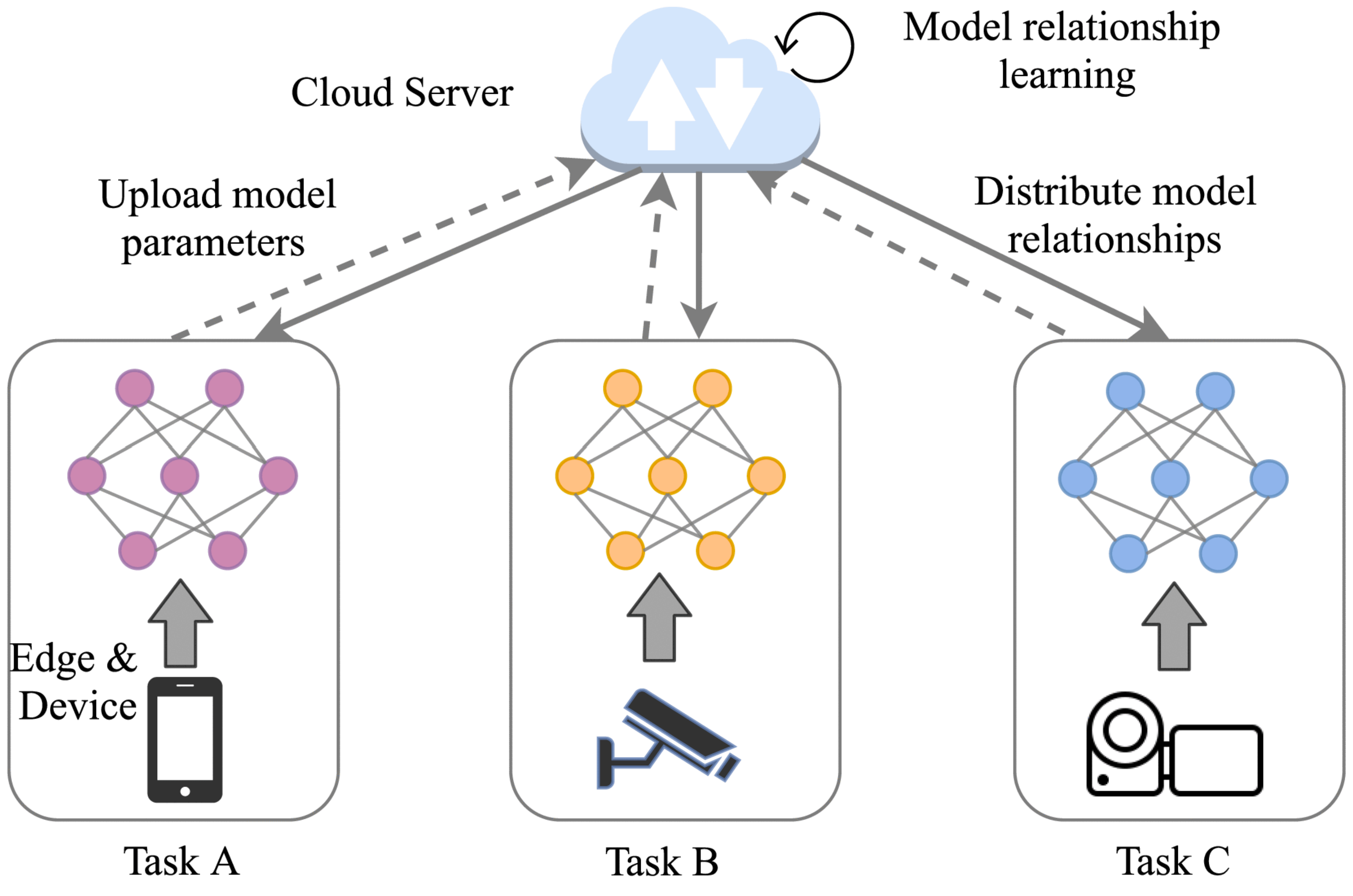
Personalized Federated Learning

Federated Learning — Local Learning

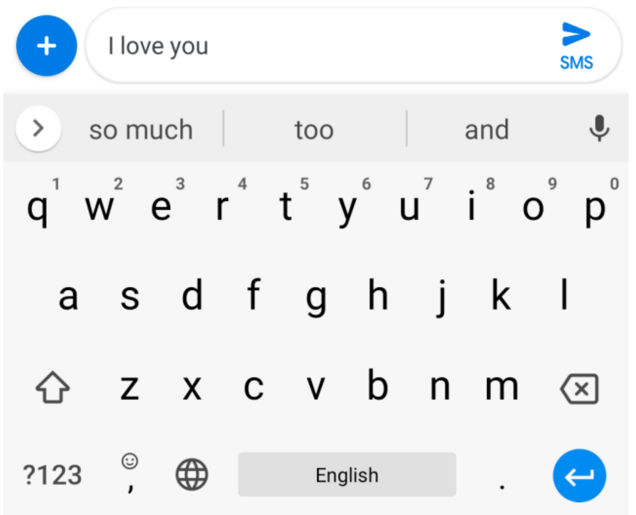
- identical tasks
- few data

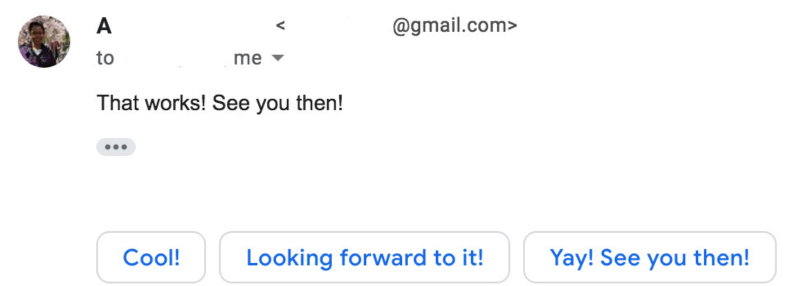
- similar tasks (same nature)
- few data

- very different tasks
- large data




**Examples:**
- Search Query Auto-Completion
- Smart Keyboard Prediction
- Email Quick Reply



# Personalized ~~Federated~~ Distributed Cost

Vanilla Federated Learning

$$\min_{w\in\mathbb{R}^d} f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Personalized Federated Learning

$$\min_{w\in\mathbb{R}^d}\left\{F(w) := \frac{1}{n}\sum_{i=1}^{n} F_i(w)\right\}$$

$$F_i(w) := f_i(w - \alpha\nabla f_i(w))$$

MAML, Fallah et al.

$$F_i(w) = \min_{\theta_i\in\mathbb{R}^d}\left\{f_i(\theta_i) + \frac{\lambda}{2}\|\theta_i - w\|^2\right\}$$

Moreau Envelopes, Dinh et al.

$$\nabla F_i(w) = \left(I - \alpha\nabla^2 f_i(w)\right)\nabla f_i(w - \alpha\nabla f_i(w))$$

- Hessian-vector product approximation

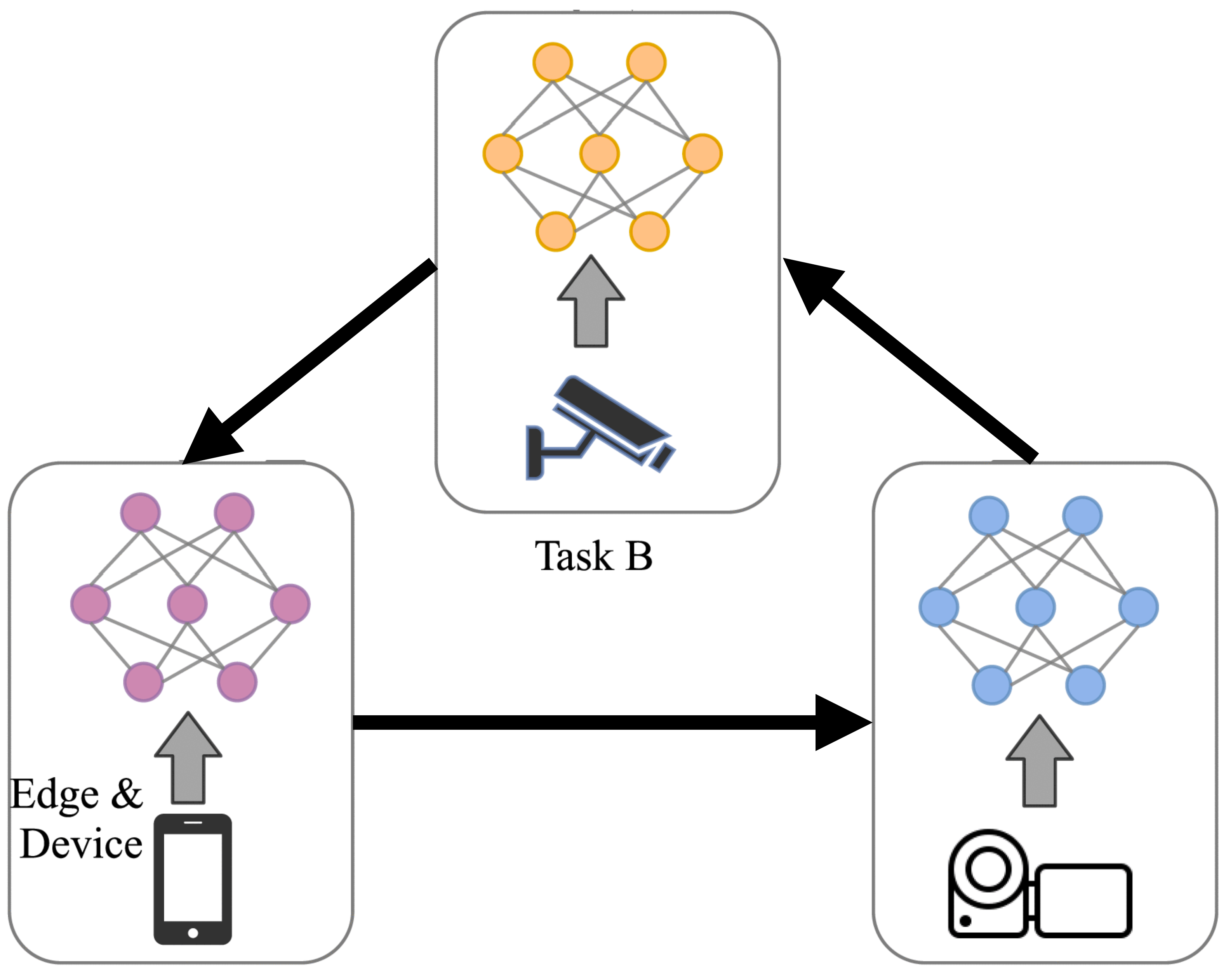
$$\nabla F_i(w) = \lambda(w - \hat{\theta}_i(w))$$

$$\hat{\theta}_i(w) := \arg\min_{\theta_i\in\mathbb{R}^d}\left[f_i(\theta_i) + \frac{\lambda}{2}\|\theta_i - w\|^2\right]$$

- exact solution approximation

# Personalization

- Why do we need personalization?




Personalized Distributed Optimization

Distributed Optimization

Local Optimization

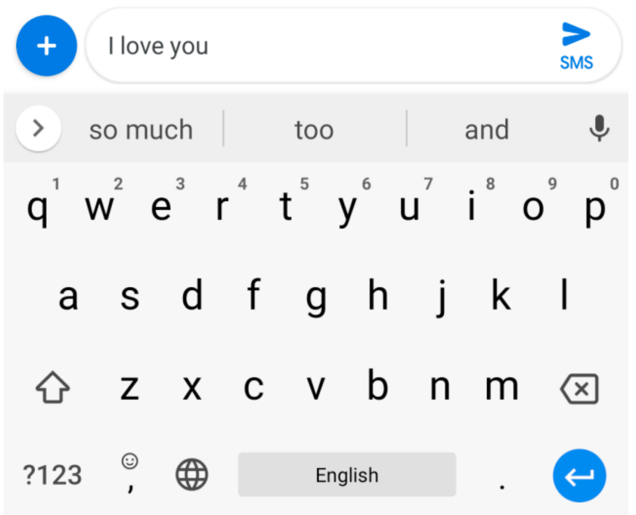
- identical tasks
- few data

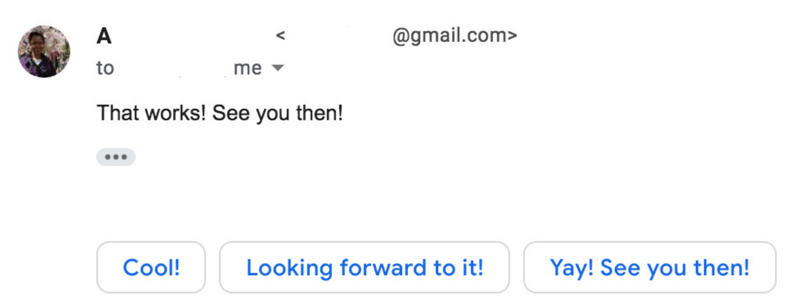
- similar tasks (same nature)
- few data

- very different tasks
- large data

**Examples:**

- Search Query Auto-Completion
- Smart Keyboard Prediction
- Email Quick Reply

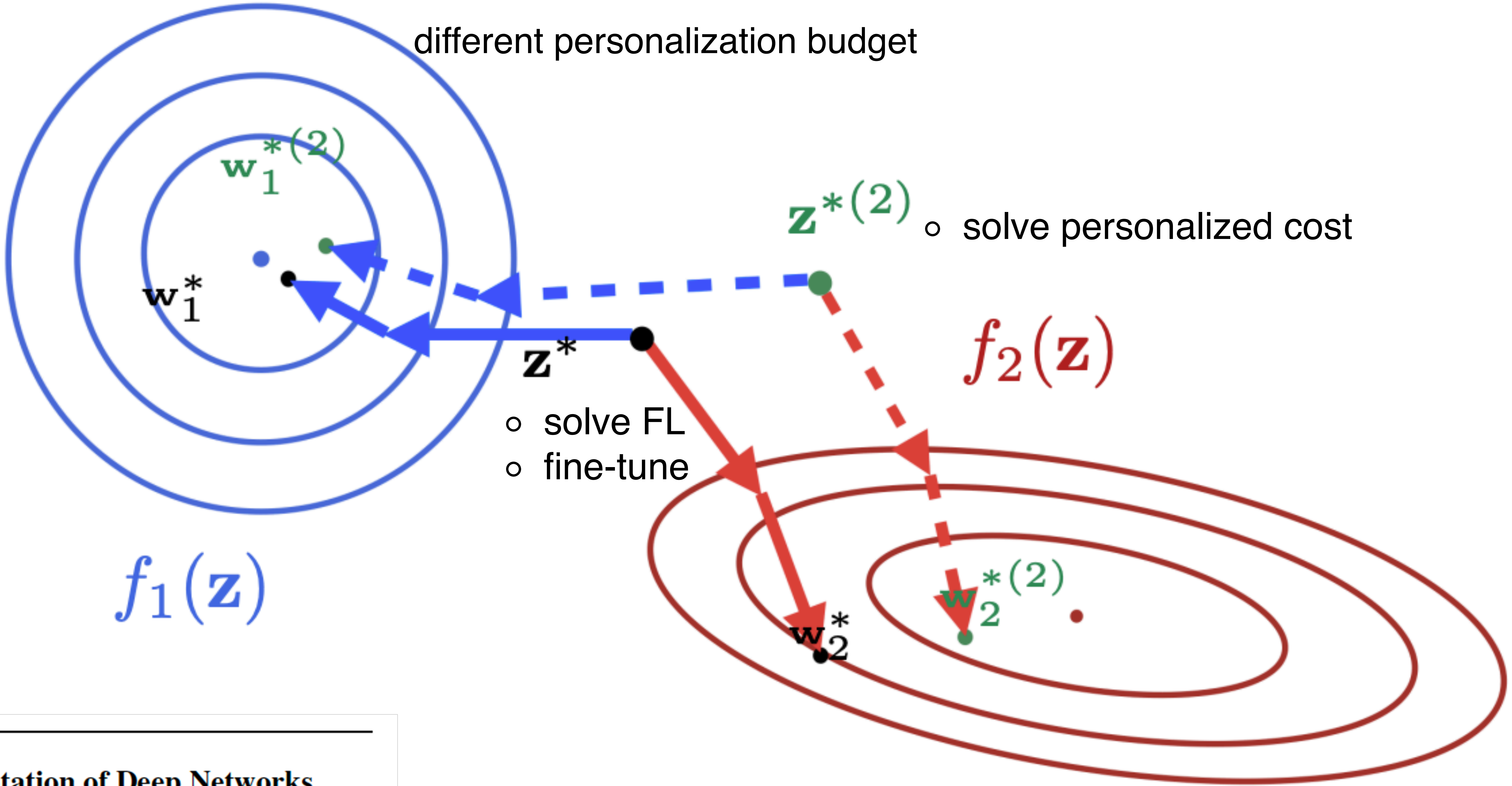


# Personalized (Distributed) Optimization

Distributed optimization

$$\min_{w \in \mathbb{R}^d} f(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x})$$




1. exploiting shared properties
2. use local properties
3. inspired by fine-tuning

**Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**

Chelsea Finn [1] Pieter Abbeel [1 2] Sergey Levine [1]

# Personalization Setup: Multi-step MAML

- $u$ steps of stochastic gradient descent (personalization budget)

$$\mathbf{z}^{*(u)} = \arg\min_{\mathbf{z}\in\mathbb{R}^d} F^{(u)}(\mathbf{z}) := \frac{1}{n}\sum_{i=1}^{n} F_i^{(u)}(\mathbf{z}),$$

$$F_i^{(u)}(\mathbf{z}) := \mathbb{E}_{p_i}\left[f_i\left(\Psi_i\left(\ldots\left(\Psi_i\left(\mathbf{z}, \mathcal{D}_{i,0}^{\text{test}}\right)\ldots\right), \mathcal{D}_{i,u-1}^{\text{test}}\right)\right)\right],$$

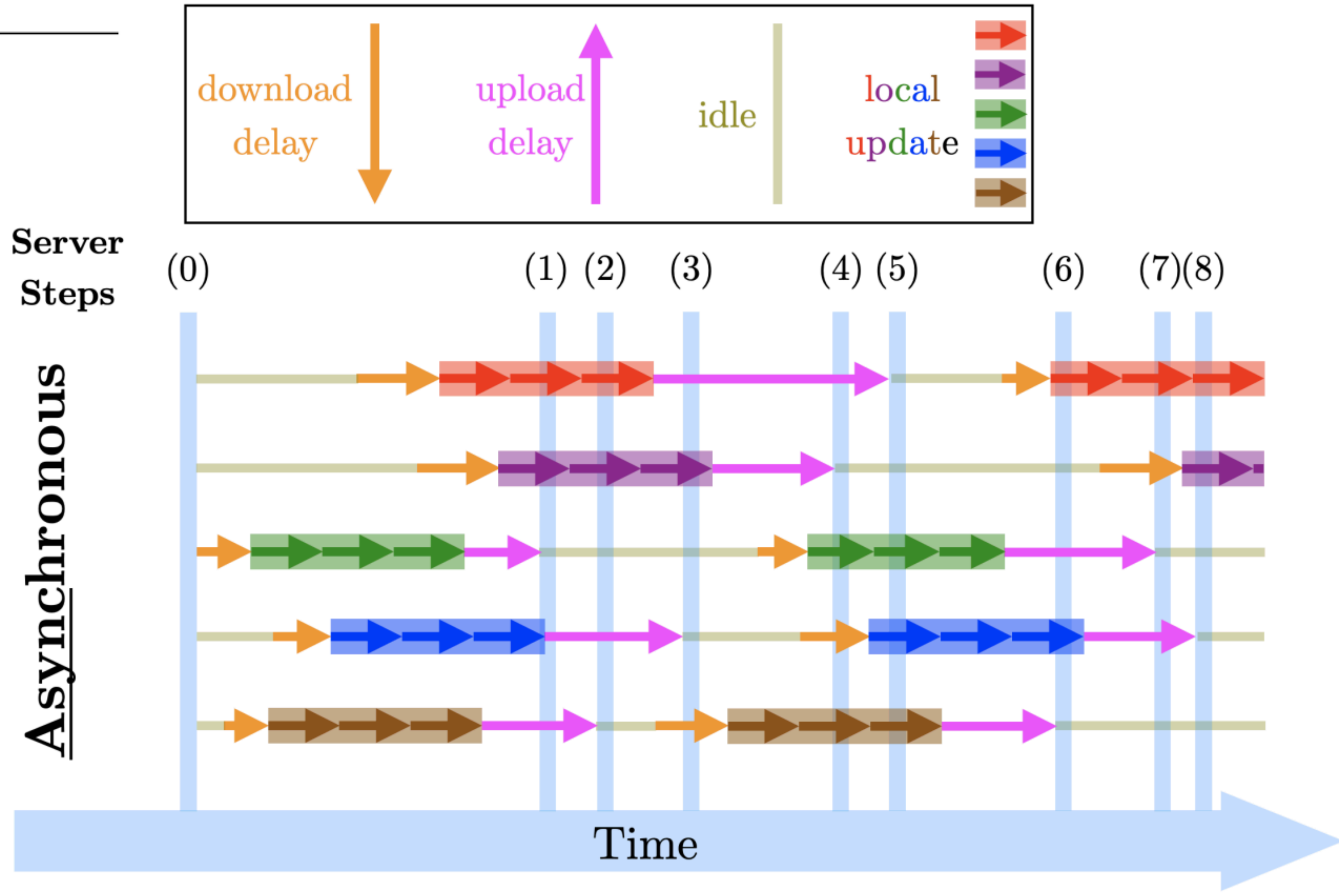
$$\Psi_i(\mathbf{z}, \mathcal{D}_i) := \mathbf{z} - \alpha\nabla\tilde{f}_i(\mathbf{z}, \mathcal{D}_i)$$

$$f_i(\mathbf{z}) := \mathbb{E}_{\xi_i\sim p_i}[\ell(\mathbf{z}, \xi_i)]$$

# PersA-FL (Server)

**Algorithm 1** [Personalized] Asynchronous Federated Learning (**Server**)

**input:** model $w^0$, $t = 0$, server stepsize $\beta$.
**repeat**
  **if** the server receives an update $\Delta_{i_t}$ from some client $i_t \in [n]$ **then**
    $w^{t+1} \leftarrow w^t - \beta \Delta_{i_t}$
    $t \leftarrow t + 1$
  **end if**
**until** not converge



# PersA-FL (Client)

$$\tilde{f}_i(w, \mathcal{D}_i) \coloneqq \frac{1}{|\mathcal{D}_i|} \sum_{\xi_i \in \mathcal{D}_i} \ell_i(w, \xi_i)$$

$$f_i(w)$$

$$F_i^{(b)}(w) \coloneqq f_i(w - \alpha \nabla f_i(w))$$

$$F_i^{(c)}(w) \coloneqq \min_{\theta_i \in \mathbb{R}^d} \left[ f_i(\theta_i) + \frac{\lambda}{2} \|\theta_i - w\|^2 \right]$$

**Algorithm 2** [Personalized] Asynchronous Federated Learning (**Client** $i$)

**input:** number of local steps $Q$, local stepsize $\eta$, `MAML` stepsize $\alpha$, Moreau Envelope (ME) regularization parameter $\lambda$, minimum batch size $b$, estimation error $\nu$.
**repeat**
  read $w$ from the server ▷ download phase
  $w_{i,0} \leftarrow w$
  **for** $q = 0$ to $Q-1$ **do** ▷ local updates
    sample a data batch $\mathcal{D}_{i,q}$ from distribution $p_i$ ▽ 3 options:
    ▷ **Option A** (AFL)
    $w_{i,q+1} \leftarrow w_{i,q} - \eta \nabla \tilde{f}_i(w_{i,q}, \mathcal{D}_{i,q})$
    ▷ **Option B** (`PersA-FL: MAML`)
    sample two data batches $\mathcal{D}'_{i,q}, \mathcal{D}''_{i,q}$ from distribution $p_i$
    $w_{i,q+1} \leftarrow w_{i,q} - \eta \left[ I - \alpha \nabla^2 \tilde{f}_i(w_{i,q}, \mathcal{D}''_{i,q}) \right] \nabla \tilde{f}_i \left( w_{i,q} - \alpha \nabla \tilde{f}_i(w_{i,q}, \mathcal{D}'_{i,q}), \mathcal{D}_{i,q} \right)$
    ▷ **Option C** (`PersA-FL: ME`)
    $\tilde{h}_i(\theta_i, w_{i,q}, \mathcal{D}_{i,q}) \coloneqq \tilde{f}_i(\theta_i, \mathcal{D}_{i,q}) + \frac{\lambda}{2} \|\theta_i - w_{i,q}\|^2$
    minimize $\tilde{h}_i(\theta_i, w_{i,q}, \mathcal{D}_{i,q})$ w.r.t. $\theta_i$ up to accuracy level $\nu$ to find $\tilde{\theta}_i(w_{i,q})$:

$$\left\| \nabla \tilde{h}_i \left( \tilde{\theta}_i(w_{i,q}), w_{i,q}, \mathcal{D}_{i,q} \right) \right\| \leq \nu$$

    $w_{i,q+1} \leftarrow w_{i,q} - \eta \lambda (w_{i,q} - \tilde{\theta}_i(w_{i,q}))$
  **end for**
  $\Delta_i \leftarrow w_{i,0} - w_{i,Q}$
  client $i$ broadcasts $\Delta_i$ to the server ▷ upload phase
**until** not interrupted by the server

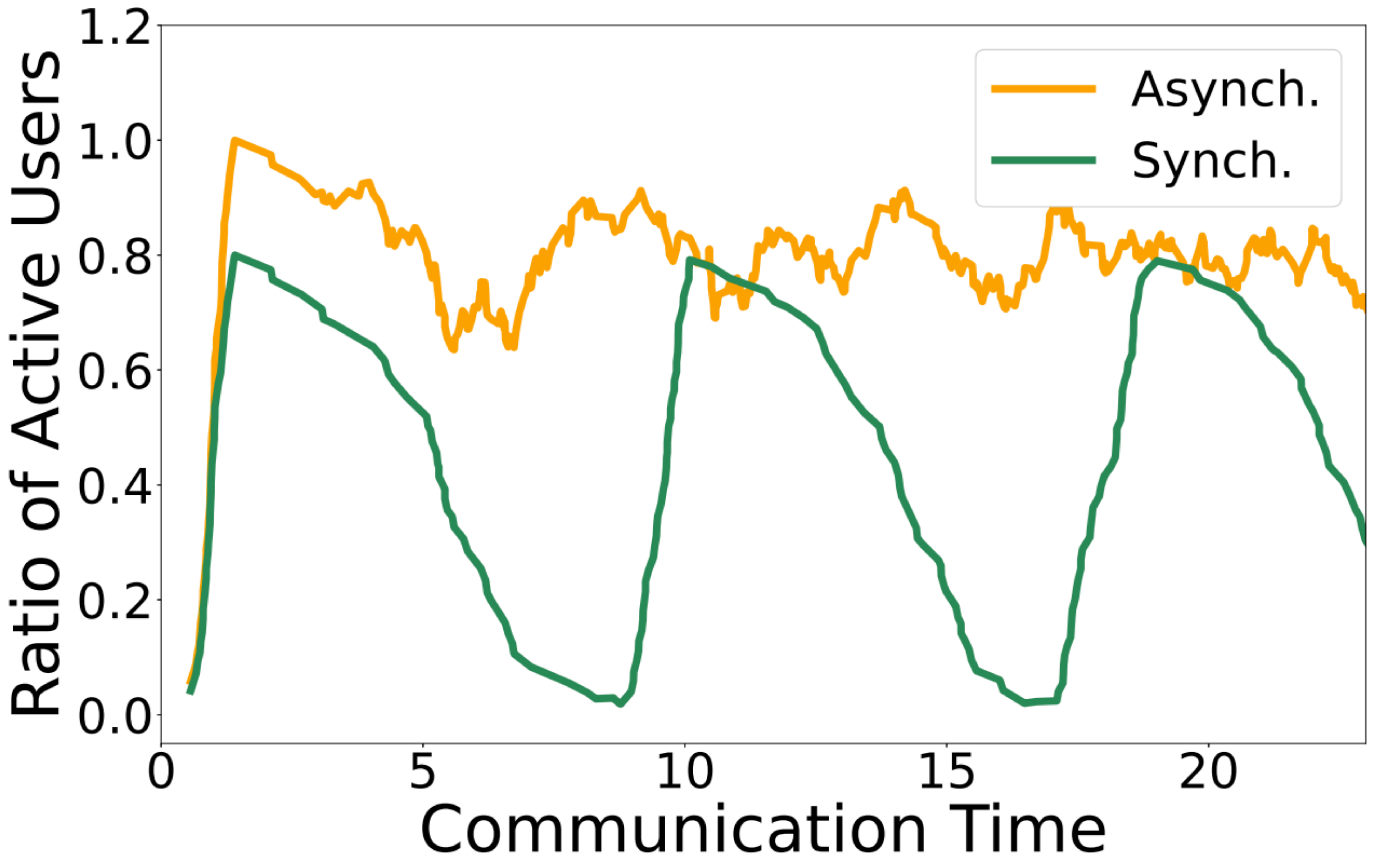
# Convergence Result

| Algorithm | & Reference | Personalized Cost | Asynchronous Updates | Unbounded Gradient | Convergence Rate |
|---|---|---|---|---|---|
| FedAvg | McMahan et al. [47] | ✗ | ✗ | - | No Analysis |
| | Yu et al. [71] | ✗ | ✗ | ✗ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ |
| | Wang et al. [67] | ✗ | ✗ | ✓ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ |
| FedAsync | Xie et al. [70] | ✗ | ✓ | ✗ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\tau^2}{T}\right)$ |
| FedBuff | Nguyen et al. [51] | ✗ | ✓ | ✗ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\tau^2}{T}\right)$ |
| Per-FedAvg | Fallah et al. [17] | ✓ | ✗ | ✗ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\alpha^2}{b}\right)$ |
| pFedMe | Dinh et al. [13] | ✓ | ✗ | ✓ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\lambda^2\left(\frac{1}{b}+\nu^2\right)}{(\lambda-L)^2}\right)$ |
| This Work | AFL | ✗ | ✓ | ✓ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\tau^2}{T}\right)$ |
| | PersA-FL: MAML | ✓ | ✓ | ✗ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\tau^2}{T}\right)+\mathcal{O}\left(\frac{\alpha^2}{b}\right)$ |
| | PersA-FL: ME | ✓ | ✓ | ✓ | $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\tau^2}{T}\right)+\mathcal{O}\left(\frac{\lambda^2}{(\lambda-L)^2}\nu^2\right)$ |

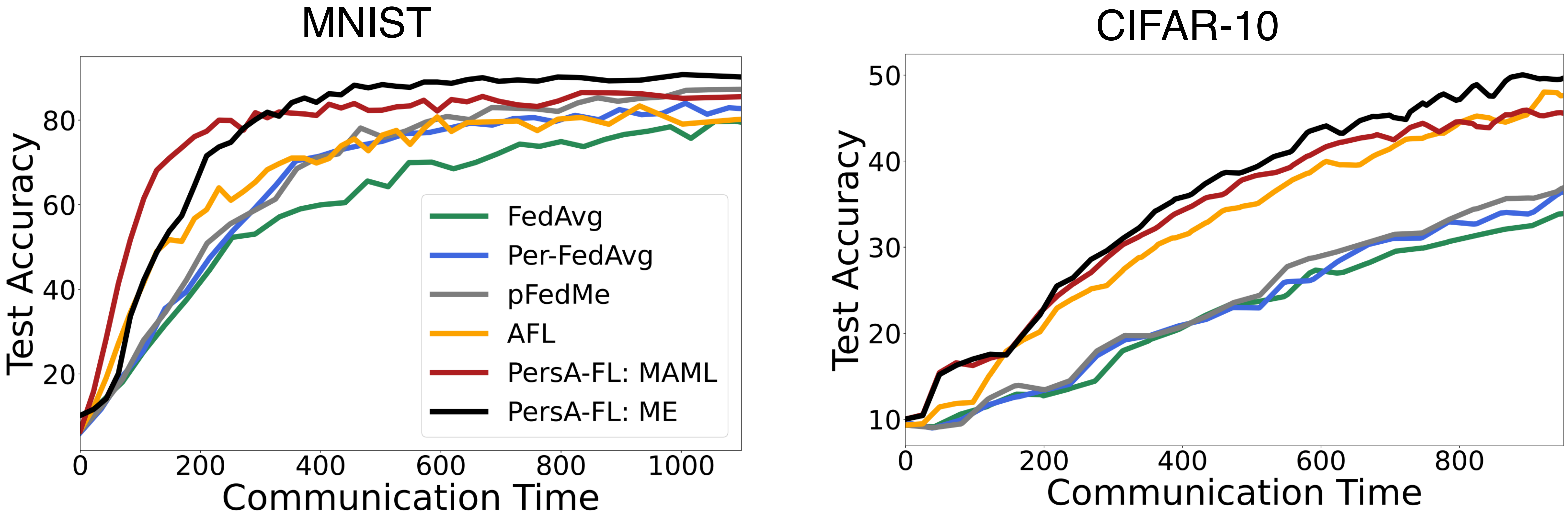
- $\tau$: maximum delay
- $\alpha$: MAML stepsize
- $\lambda$: ME regularization
- $\nu$: ME error
- $b$: batch size

# Asynchronous Setup: Concurrency

- upload/download $\approx 4.4$
- percentage of active users
- staleness

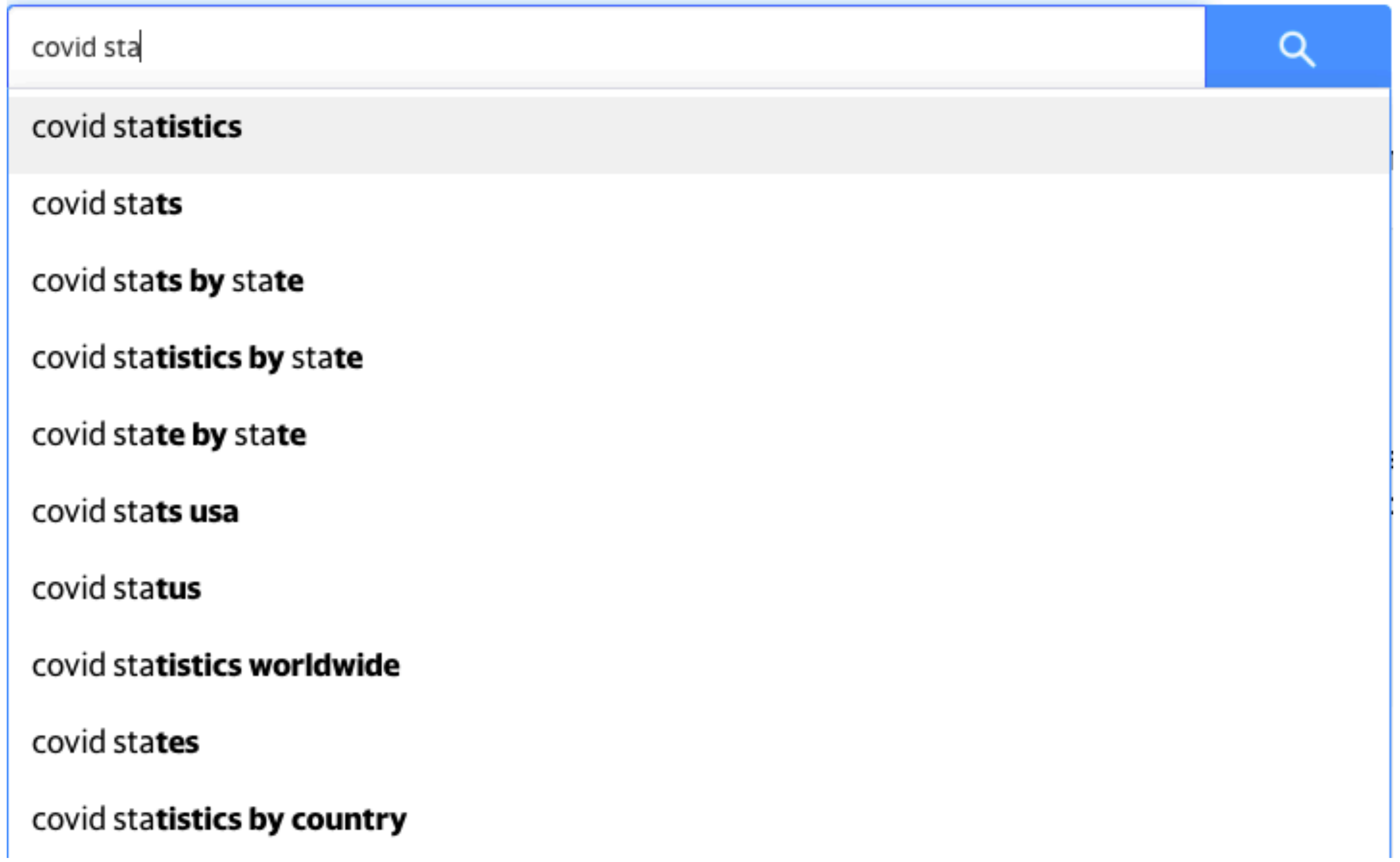


# Numerical Experiments: Heterogeneous Data



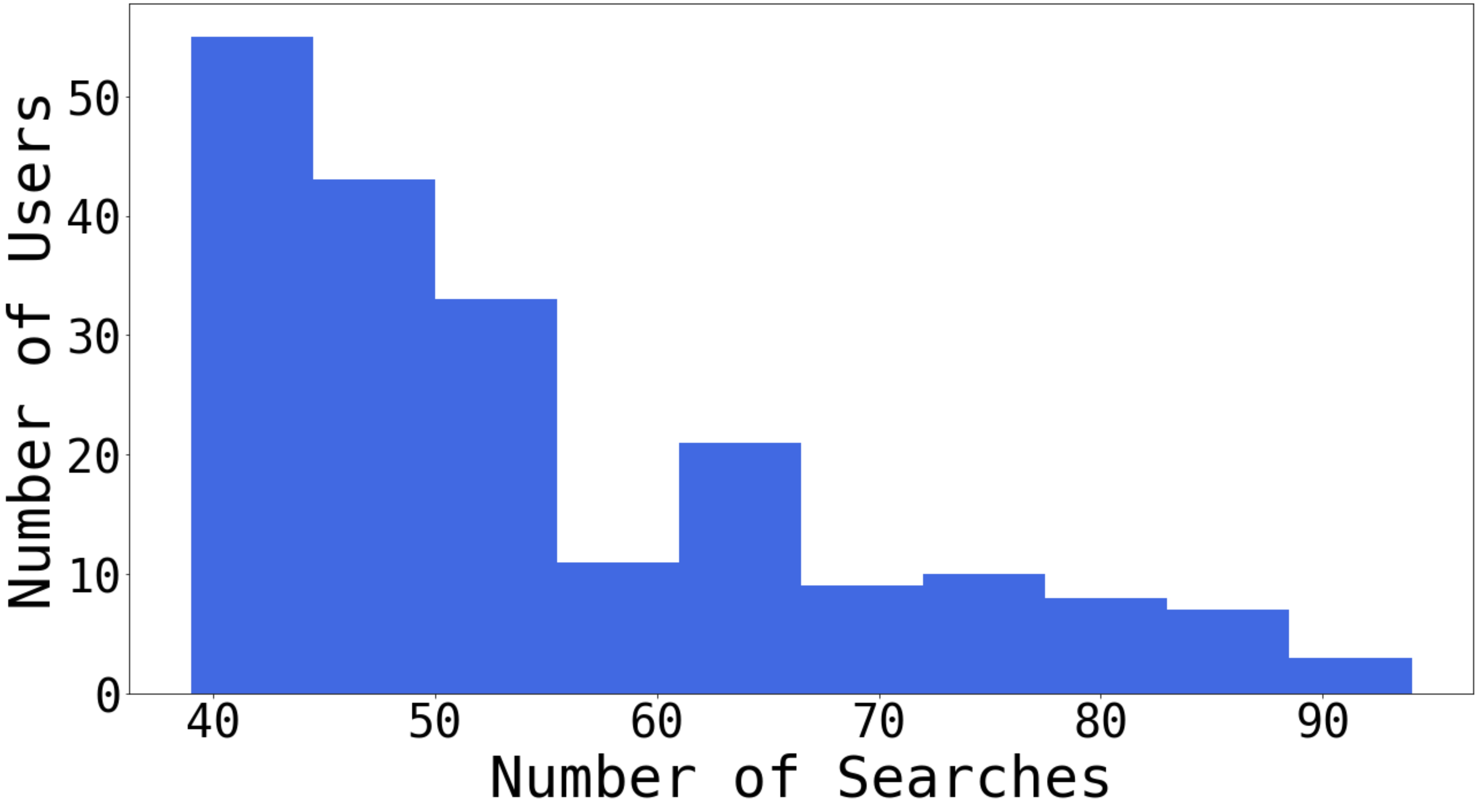
# Personalized Search Ranking

- Data:
  - input: partial query
  - output: a list of suggestions
  - action: top $k$ best suggestions
- Main Question:
  - identify top suggestions
  - ranking problem



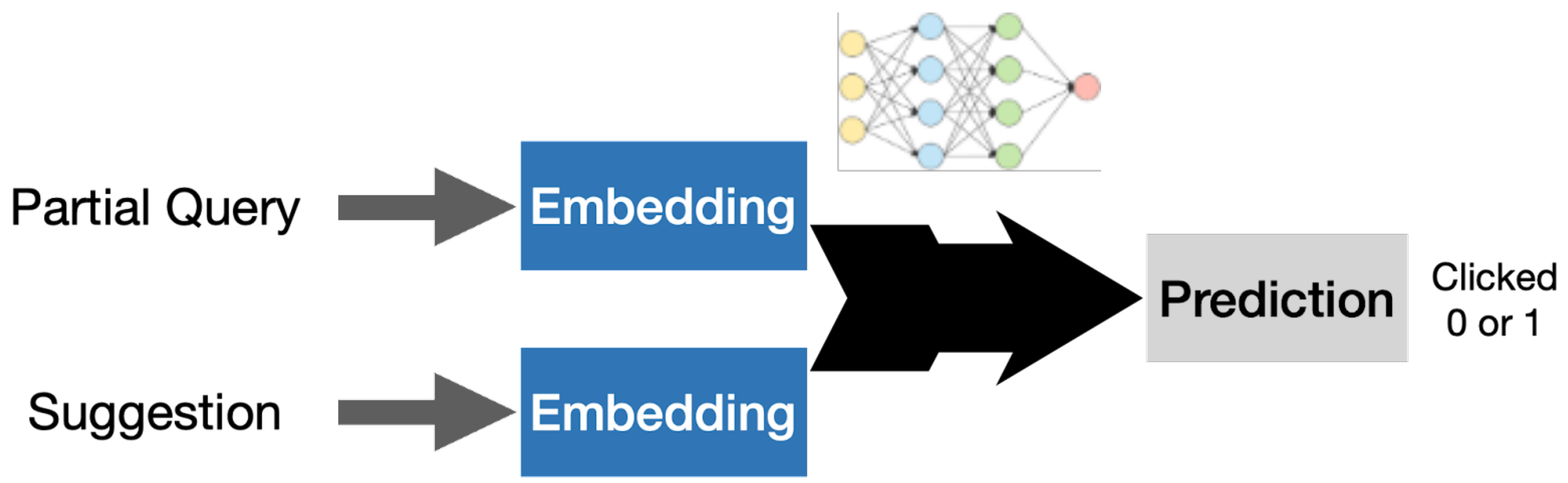
# Personalized Search Ranking

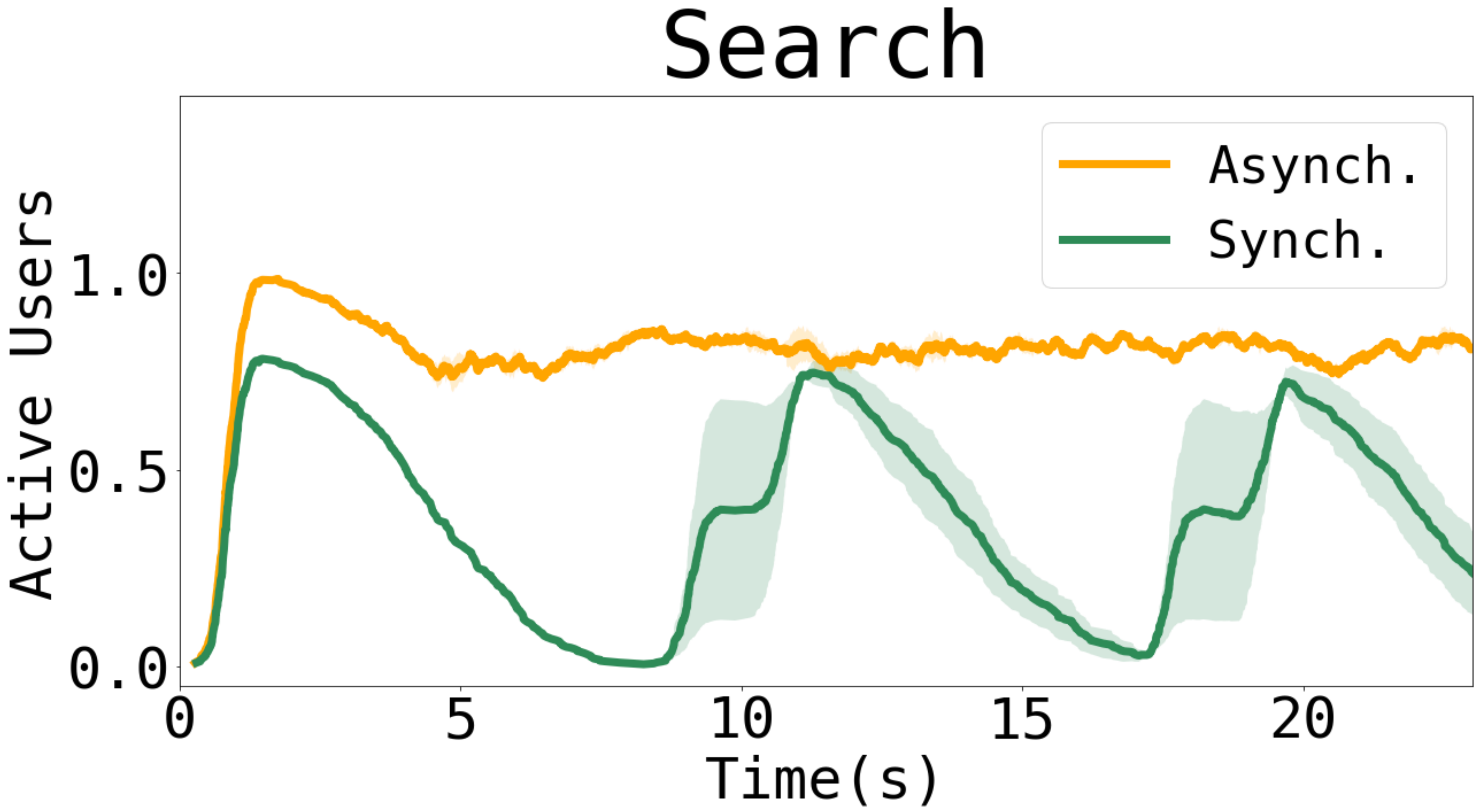
- 200 distinguished User IDs
- different personal preferences
- number of queries: [30, 100]
- list of suggestions: [2, 25]
- Potential suggestions: x3
- identify top suggestions among a small group of proposals



# Personalized Ranking of Suggestions

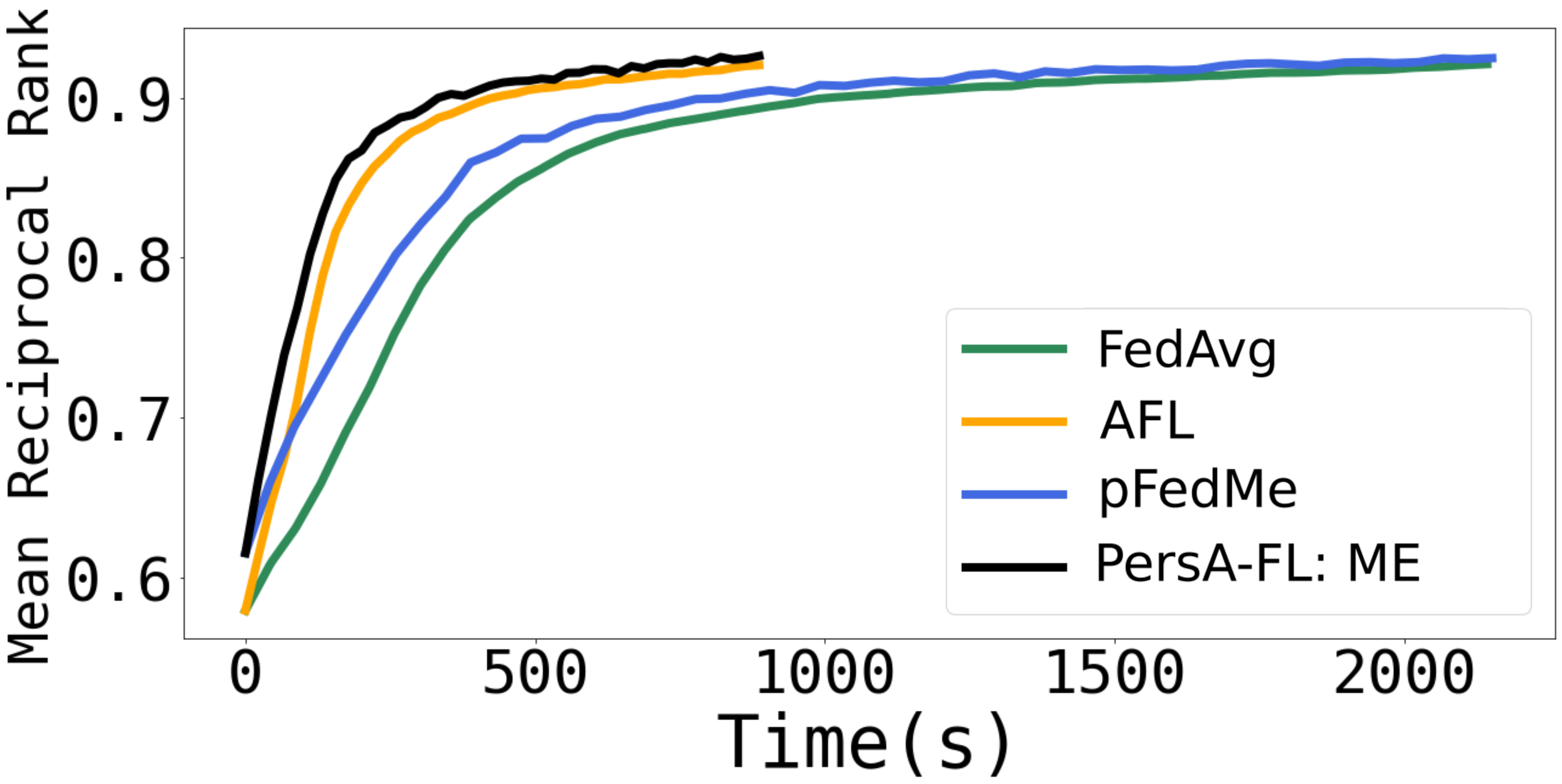
- Model:
  - Random Forest (Classic Model)
  - MLP
  - MLP + CNN
- Loss Function:
  - Binary Cross-Entropy
  - Mean Square Error
- Criterion:
  - accuracy
  - normalized mean reciprocal rank (MRR)



# Numerical Result: Personalized Search Ranking

- $n = 200$
- $\lambda = 15$
- $\eta \approx 0.05$
- $\beta = 1/n$




$$1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{k}$$

- Weighted accuracy based on the location in the suggestion list

# PART II

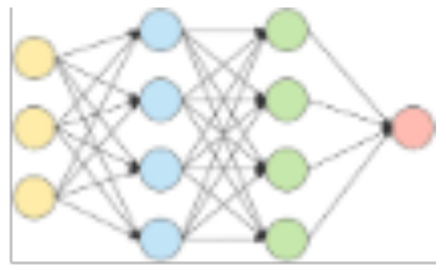
# PARS-Push: Personalized, Asynchronous and Robust Decentralized Optimization

# Distributed Optimization

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^d} \left[ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right]$$

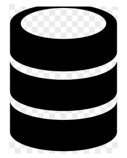
number of clients

local cost function



number of parameters

$$f_i(\mathbf{x}) = \mathbb{E}_{\xi_i \sim p_i}[\ell_i(\mathbf{x}, \xi_i)]$$



local distribution

- parameters
- heterogeneous distributions

Stochastic cost over data batch $\mathcal{D}_i$:

$$\tilde{f}_i(\mathbf{x}, \mathcal{D}_i) := \frac{1}{|\mathcal{D}_i|}\sum_{\xi_i \in \mathcal{D}_i} \ell_i(\mathbf{x}, \xi_i)$$

# Decentralization Challenge

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^d}\left[f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})\right]$$

number of clients

local cost function

number of parameters

$$f_i(\mathbf{x}) = \mathbb{E}_{\xi_i\sim p_i}[\ell_i(\mathbf{x},\xi_i)]$$

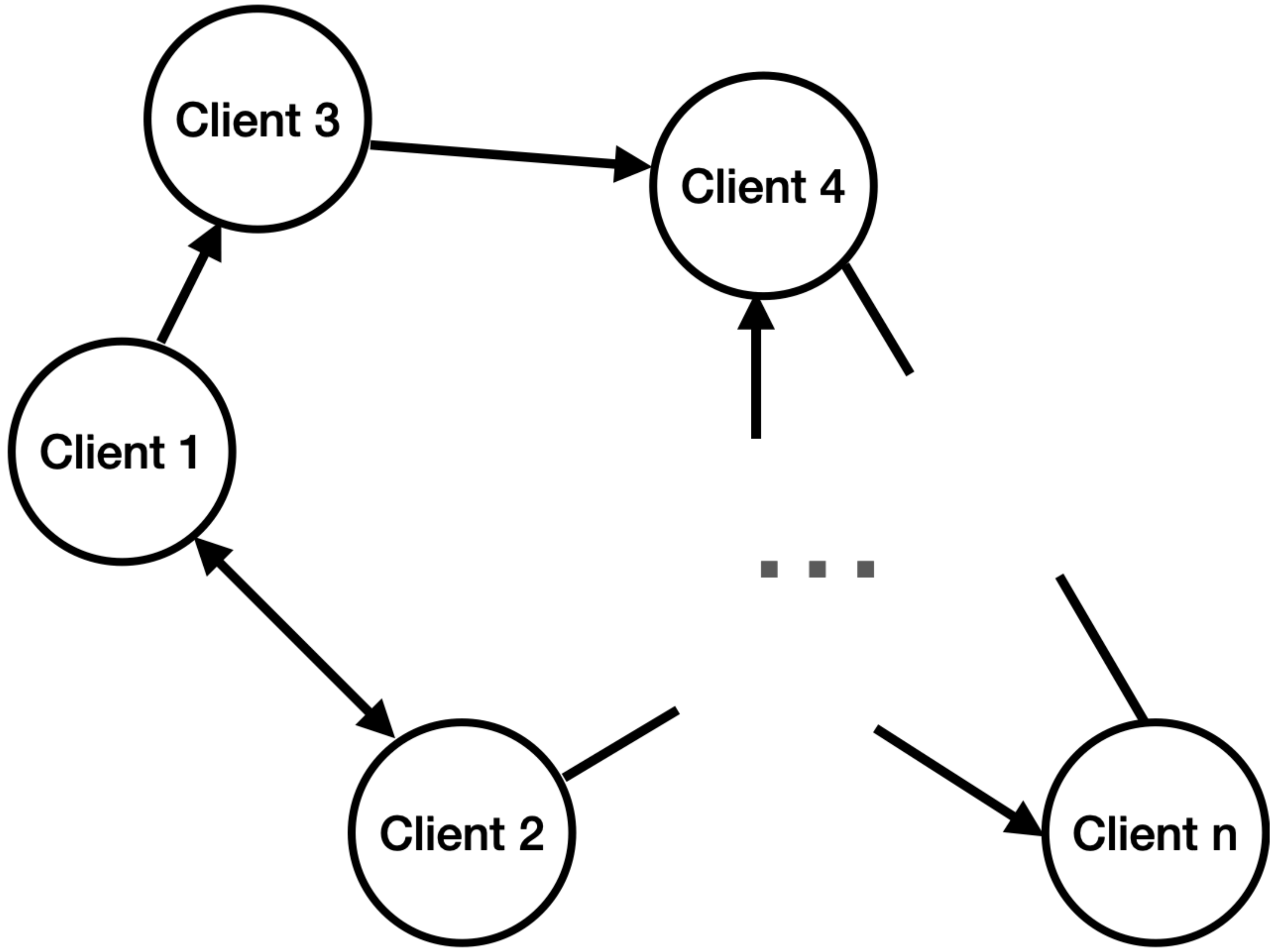
local distribution

$$\min_{\substack{(\mathbf{x_1},\dots,\mathbf{x_n})\in(\mathbb{R}^d)^n \\ \text{s.t.} \quad \mathbf{x_1}=\cdots=\mathbf{x_n}}}\left[f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x_i})\right]$$

serverless $\Rightarrow$ consensus

# Network Setup

- $\mathcal{G} = ([n], \mathcal{E})$ is a static, directed, and strongly-connected graph
- $(i, j) \in \mathcal{E}$ iff there exists a directed link from node $i$ to node $j$
- $\mathcal{N}_i^- = \{j | (j, i) \in \mathcal{E}\} \cup \{i\}$
- $\mathcal{N}_i^+ = \{j | (i, j) \in \mathcal{E}\} \cup \{i\}$



# Asynchronous Communications

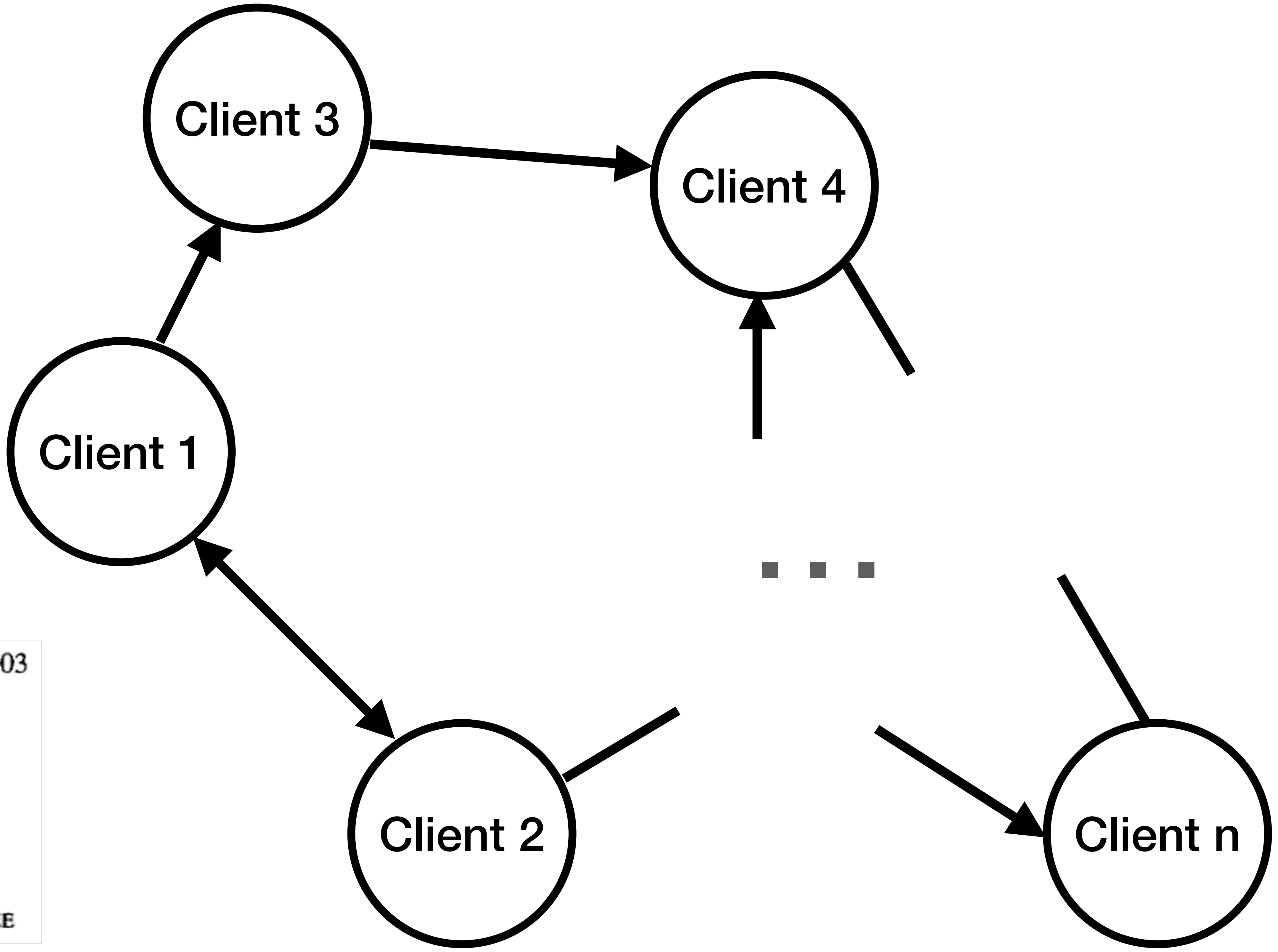
Limitations of synchronous algorithms:

- communication delays
- connection reliability
- agent unavailability

IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. AC-31, NO. 9, SEPTEMBER 1986 803

**Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms**

JOHN N. TSITSIKLIS, MEMBER, IEEE, DIMITRI P. BERTSEKAS, FELLOW, IEEE, AND MICHAEL ATHANS, FELLOW, IEEE



# Asynchronous Communication Setup

Journal of Machine Learning Research 21 (2020) 1-47 Submitted 12/18; Revised 2/20; Published 3/20

**Robust Asynchronous Stochastic Gradient-Push: Asymptotically Optimal and Network-Independent Performance for Strongly Convex Functions**

**Artin Spiridonoff** ARTIN@BU.EDU
**Alex Olshevsky** ALEXOLS@BU.EDU
**Ioannis Ch. Paschalidis** YANNISP@BU.EDU
*Division of Systems Engineering*
*Boston University*
*Boston, MA 02215, USA*

A. each client $i$ wakes up at least once in $\Gamma_w$ consequent rounds,

B. the delays on each communication link are bounded by $\Gamma_d \geq 1$,

C. each communication link fails at most $\Gamma_f \geq 0$ consecutive times.

- effective maximum delay $\Gamma_e = \Gamma_w + \Gamma_d - 1$,
- each agent receives a message from its in-neighbors at least once every $\Gamma_s = \Gamma_w(\Gamma_f + 1) + \Gamma_e$

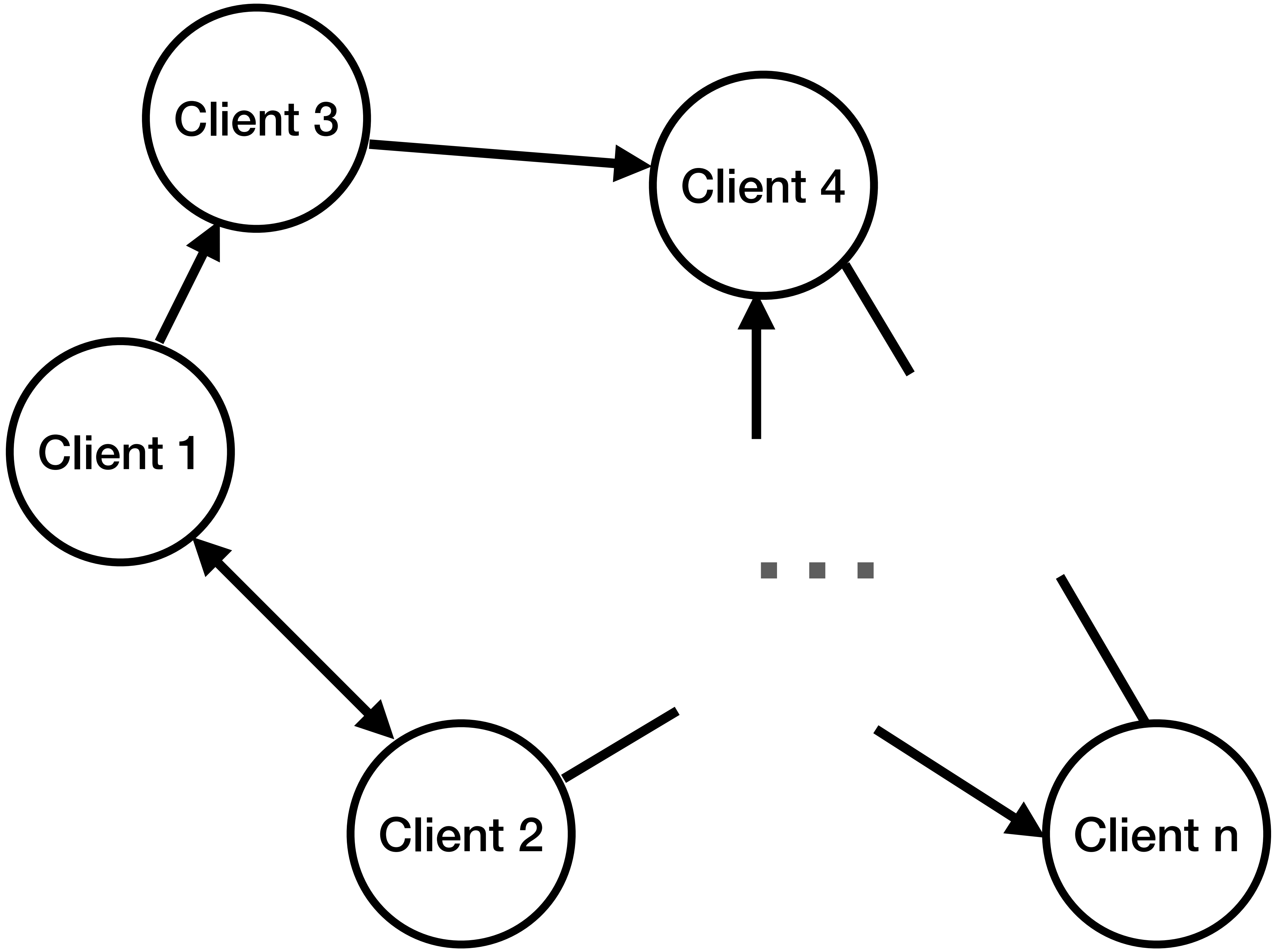
# “Running Sums” Technique for Decentralized Consensus

**Setup:**

- initialize: $\mathbf{x}_i \in \mathbb{R}^d$
- $d_i^- = \ \mathcal{N}_i^- \ , d_i^+ = \ \mathcal{N}_i^+$

**Sketch of the idea:**

- $\phi_i^{\mathbf{x}}$: sum of client $i$'s updates
  - $\phi_i^{\mathbf{x}} \leftarrow \phi_i^{\mathbf{x}} + \dfrac{\mathbf{x}_i}{d_i^+ + 1}$, when client $i$ is active
  - broadcasts $\phi_i^{\mathbf{x}}$ to out-neighbors , when link $(i, j)$ is active
  - receives $\phi_j^{\mathbf{x}}$ from in neighbors, when link $(j, i)$ is active
- $\rho_{ij}^{\mathbf{x}}$: copy of $\phi_j^{\mathbf{x}}$ from the most recent communication of node $j$ to the server
  - $\mathbf{x_i} \leftarrow \mathbf{x_i} + \sum_{j \in \mathcal{N}_i^-} (\phi_j^{\mathbf{x}} - \rho_{ij}^{\mathbf{x}})$
  - $\rho_{ij}^{\mathbf{x}} \leftarrow \phi_j^{\mathbf{x}}$
- $y_i$: slack scalar which is initialized with $1$, push-sum variable
- $\phi_i^y, \rho_{ij}^y$: defined and updated similarly




**Robust Distributed Average Consensus via Exchange of Running Sums**

C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García

# PARS-Push Algorithm

- Multi-step personalization budget ($u$)
- Asynchronous Communications
- Message Loss
- Communication Delay

Stochastic gradient calculation

Robust asynchronous aggregation

Gradient-Push on an Augmented Communication Graph

```
Initialize: y_i = 1, κ_i = −1, φ_i^x = 0, φ_i^y = 0, ∀i ∈ [n],
  and κ_ij = −1, ρ_ij^x = 0, ρ_ij^y = 0, ∀(j,i) ∈ E.
for t = 0, 1, 2, ..., in parallel for all i ∈ [n] do
  if node i wakes up then
    η_i(t) := Σ_{r=κ_i+1}^{t} θ(r)
    w_i^(0) := z_i
    for r = 0, 1, 2, ..., u − 1 do
      Sample a batch D_{i,r}^t with size b from p_i
      w_i^(r+1) := w_i^(r) − α∇f̃_i(w_i^(r), D_{i,r}^t)
    end for
    Sample a batch D_{i,u}^t with size b from p_i
    x_i := x_i − η_i(t) [ Π_{r=0}^{u−1} (I − α∇²f̃_i(w_i^(r), D_{i,r}^t)) ] × ∇f̃_i(w_i^(u), D_{i,u}^t)
    κ_i := t
    x_i := x_i / (d_i^+ + 1), y_i := y_i / (d_i^+ + 1)
    φ_i^x := φ_i^x + x_i, φ_i^y := φ_i^y + y_i
    Node i sends (φ_i^x, φ_i^y, κ_i) to N_i^+
    R_i := messages received from N_i^−
    for (φ_j^x, φ_j^y, κ_j) in R_i do
      if κ_j > κ_ij then
        ρ_ij^{*x} := φ_j^x, ρ_ij^{*y} := φ_j^y, κ_ij := κ_j
      end if
    end for
    x_i := x_i + Σ_{j∈N_i^−} (ρ_ij^{*x} − ρ_ij^x)
    y_i := y_i + Σ_{j∈N_i^−} (ρ_ij^{*y} − ρ_ij^y)
    ρ_ij^x := ρ_ij^{*x}, ρ_ij^y := ρ_ij^{*y}, z_i := x_i / y_i
  end if
end for
```

$$\eta_i(t) := \sum_{r=\kappa_i+1}^{t} \theta(r)$$

$$\mathbf{w}_i^{(r+1)} := \mathbf{w}_i^{(r)} - \alpha \nabla \tilde{f}_i\left(\mathbf{w}_i^{(r)}, \mathcal{D}_{i,r}^t\right)$$

$$\mathbf{x}_i := \mathbf{x}_i - \eta_i(t)\left[\prod_{r=0}^{u-1}\left(\mathbf{I} - \alpha\nabla^2 \tilde{f}_i\left(\mathbf{w}_i^{(r)}, \mathcal{D}_{i,r}^t\right)\right)\right] \times \nabla \tilde{f}_i(\mathbf{w}_i^{(u)}, \mathcal{D}_{i,u}^t)$$

$$\mathbf{x}_i := \frac{\mathbf{x}_i}{d_i^+ + 1},\quad y_i := \frac{y_i}{d_i^+ + 1}$$

$$\mathbf{x}_i := \mathbf{x}_i + \sum_{j\in\mathcal{N}_i^-}(\boldsymbol{\rho}_{ij}^{*\mathbf{x}} - \boldsymbol{\rho}_{ij}^{\mathbf{x}}),\quad y_i := y_i + \sum_{j\in\mathcal{N}_i^-}(\rho_{ij}^{*y} - \rho_{ij}^{y})$$

Stochastic gradient computation

Robust asynchronous aggregation

# PARS-Push Update Rule Analysis

$$\tau_i(t) = \begin{cases} 1, & \text{if node } i \text{ wakes up at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$\tau_{ji}^l(t) = \begin{cases} 1, & \text{if } \tau_i(t) = 1 \text{ and the message from } j \text{ to } i \text{ arrives after an effective delay } l \in [\Gamma_e] \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{x}_i(t+\tfrac{1}{2}) := \mathbf{x}_i(t) - \tau_i(t)\, \eta_i(t)\, \nabla \tilde{F}_i^{(u)}(\mathbf{z}_i(t), \vartheta_i^t),$$

$$\mathbf{x}_i(t+1) := \left(1 - \tau_i(t) + \frac{\tau_i(t)}{d_i^+ + 1}\right) \mathbf{x}_i(t+\tfrac{1}{2}) + \sum_{j \in \mathcal{N}_i^-} \mathbf{x}_{ji}^1(t),$$

$$\hat{\mathbf{x}}_{ji}^l(t+1) := \tau_{ji}^l(t) \left[\tilde{\mathbf{x}}_{ji}(t) + \frac{\mathbf{x}_j(t)}{d_j^+ + 1}\right] + \mathbb{1}_{\{l<\Gamma_e\}} \hat{\mathbf{x}}_{ji}^{l+1}(t+1),$$

$$\tilde{\mathbf{x}}_{ji}^l(t+1) := \left(1 - \sum_{l=1}^{\Gamma_d} \tau_{ji}^l(t)\right) \left[\tilde{\mathbf{x}}_{ji}(t) + \tau_i(t) \frac{\mathbf{x}_j(t)}{d_j^+ + 1}\right],$$

→

**Gradient-Push on an Augmented Communication Graph**

$$\mathbf{X}(t+1) := \mathbf{M}(t)\left(\mathbf{X}(t) - \boldsymbol{\Delta}(t)\right),$$
$$\mathbf{y}(t+1) := \mathbf{M}(t)\, \mathbf{y}(t),$$
$$\mathbf{z}_i(t+1) := \mathbf{x}_i(t)/y_i(t), \quad \forall i \in [n]$$

$$[\boldsymbol{\Delta}(t)]_i := \begin{cases} \tau_i(t)\, \eta_i(t)\, \nabla \tilde{F}_i^{(u)}(\mathbf{z}_i(t), \vartheta_i^t)^\top, & i \in [n], \\ \mathbf{0}^\top, & i \notin [n]. \end{cases}$$

$\{\mathbf{M}(t)\}_{t \in \mathcal{Z}_0^+}$ is a sequence of column stochastic mixing matrices

# Assumptions: Smooth & Strongly-Convex

- Smoothness:

$$\left\|\nabla \ell(\mathbf{z}, \xi) - \nabla \ell(\hat{\mathbf{z}}, \xi)\right\| \leq L\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|$$

- Lipschitz Hessian:

$$\left\|\nabla^2 \ell(\mathbf{z}, \xi) - \nabla^2 \ell(\hat{\mathbf{z}}, \xi)\right\| \leq H\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|$$

- Strong Convexity:

$$\left\|\nabla \ell(\mathbf{z}, \xi) - \nabla \ell(\hat{\mathbf{z}}, \xi)\right\| \geq \mu\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|$$

- Bounded Gradient:

$$\|\nabla \ell(\mathbf{z}, \xi)\| \leq G$$

# Stochastic Gradient-Push for Strongly Convex Functions on Time-Varying Directed Graphs

Angelia Nedić and Alex Olshevsky

*Lemma 3:* Let $q : \mathbb{R}^d \to \mathbb{R}$ be a $\mu$-strongly convex function with $\mu > 0$ and have Lipschitz continuous gradients with constant $M > 0$. Let $v \in \mathbb{R}^d$ and let $u \in \mathbb{R}^d$ be defined by

$$u = v - \alpha \left( \nabla q(v) + \phi(v) \right),$$

where $\alpha \in (0, \frac{\mu}{8M^2}]$ and $\phi : \mathbb{R}^d \to \mathbb{R}^d$ is a mapping such that

$$\|\phi(v)\| \leq c \qquad \text{for all } v \in \mathbb{R}^d.$$

Then, there exists a compact set $\mathcal{V} \subset \mathbb{R}^d$ (which depends on $c$ and the funtion $q(\cdot)$ but not on $\alpha$) such that

$$\|u\| \leq \begin{cases} \|v\| & \text{for all } v \notin \mathcal{V} \\ R & \text{for all } v \in \mathcal{V}, \end{cases}$$

where $R = \max_{z \in \mathcal{V}} \{\|z\| + (\mu/(8M^2))\|\nabla q(z)\|\} + (\mu c)/(8M^2)$.

# Convergence Guarantee: Smooth & Strongly-Convex

**Strongly-Convex** $\hat{\mu}(u) = \mu(1-\alpha L)^{2u} - \alpha u G H (1-\alpha\mu)^{u-1}$

**Smooth** $\hat{L}(u) = L(1-\alpha\mu)^{2u} + \alpha u G H (1-\alpha\mu)^{u-1}$

**Bounded Variance** $\mathbb{E}_{p_i}\left\|\nabla \tilde{F}_i^{(u)}(\mathbf{z}, \vartheta_i) - \nabla F_i^{(u)}(\mathbf{z})\right\|^2 \leq \hat{\sigma}(u)^2 := 4(1-\alpha\mu)^{2u} G^2$

$\vartheta_i = \{\mathcal{D}_{i,r}\}_{r=0}^{u}$

↓

$$\mathbb{E}\left[\left\|\mathbf{z}_i(T) - \mathbf{z}^{*(u)}\right\|^2\right] = \mathcal{O}\left(\frac{\Gamma_w \hat{\sigma}(u)^2}{\hat{\mu}(u)\, n\, T}\right) + \mathcal{O}\left(\frac{1}{T^{\frac{3}{2}}}\right)$$

# Assumptions: Smooth & Non-Convex

- Smoothness:

$$\left\|\nabla \ell(\mathbf{z}, \xi) - \nabla \ell(\hat{\mathbf{z}}, \xi)\right\| \leq L\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|$$

- Lipschitz Hessian:

$$\left\|\nabla^2 \ell(\mathbf{z}, \xi) - \nabla^2 \ell(\hat{\mathbf{z}}, \xi)\right\| \leq H\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|$$

- Bounded Gradient:

$$\|\nabla \ell(\mathbf{z}, \xi)\| \leq G$$

- Awake Nodes:

$$\Gamma_w := 1$$

# Convergence Guarantee: Smooth & Non-Convex

**Smooth** $\hat{L}(u) = (L + \alpha u G H)(1 + \alpha L)^{2u},$

**Bounded Variance** $\mathbb{E}_{p_i} \left\| \nabla \tilde{F}_i^{(u)}(\mathbf{z}, \vartheta_i) - \nabla F_i^{(u)}(\mathbf{z}) \right\|^2 \leq \hat{\sigma}(u)^2$

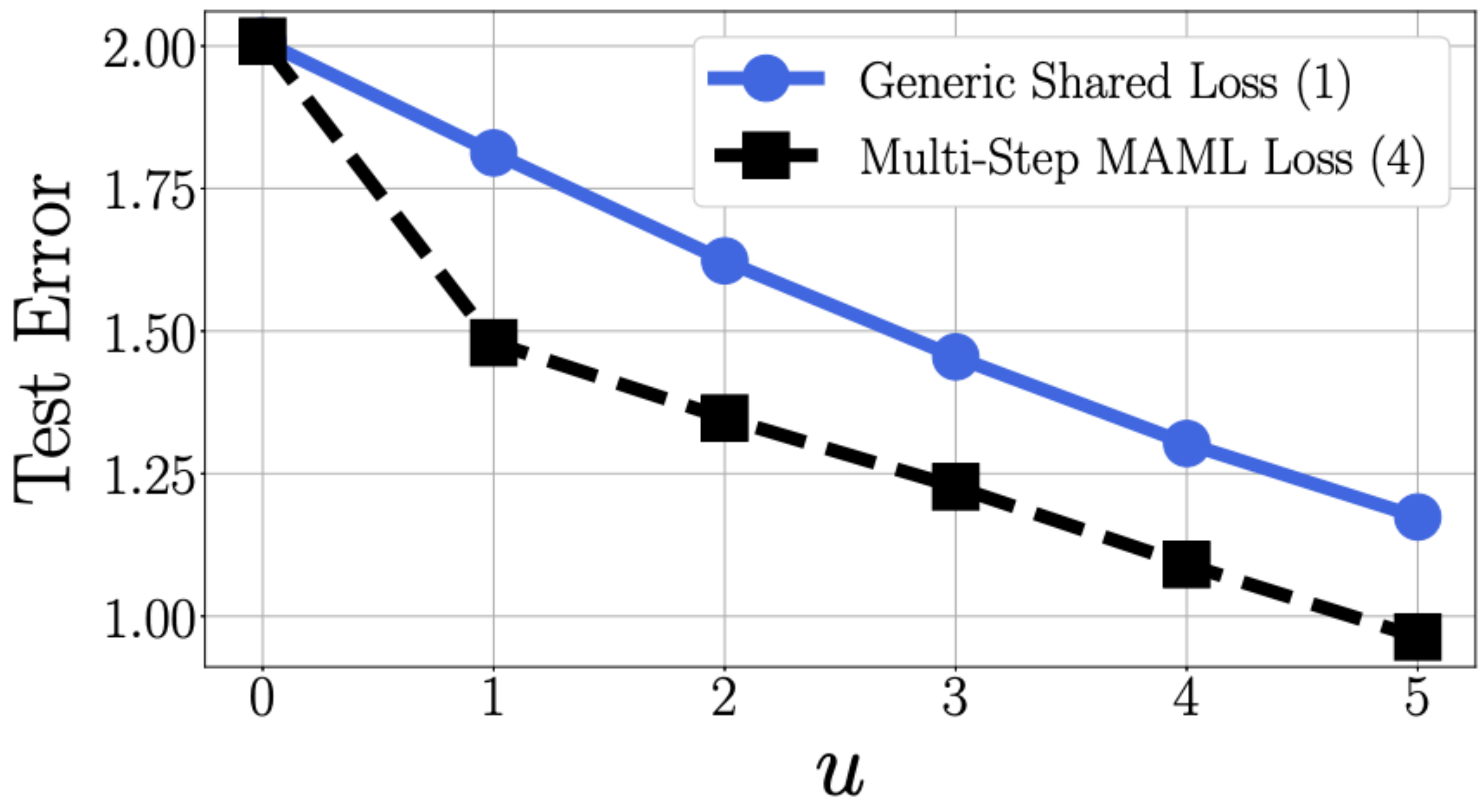
$\vartheta_i = \{\mathcal{D}_{i,r}\}_{r=0}^{u}$

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\| \nabla F^{(u)} \left( \frac{\mathbf{X}(t)^\top \mathbf{1}}{n} \right) \right\|^2 = \mathcal{O} \left( \frac{2\hat{L}(u) F^{(u)}(\mathbf{0}) + \hat{\sigma}(u)^2}{(nT)^{\frac{1}{2}}} \right) + \mathcal{O} \left( \frac{1}{T} \right)$$

# Personalization Impact

$$b_{iq} = \mathbf{a}_{iq}^\top \boldsymbol{\beta}_i^* + \zeta_{iq}$$

$$f_i(\mathbf{z}) = \mathbb{E}_{\xi_{iq} \sim p_i} \left[ \left( b_{iq} - \mathbf{a}_{iq}^\top \mathbf{z} \right)^2 + \frac{1}{2n} \|\mathbf{z}\|^2 \right]$$



# Robustness to Asynchrony

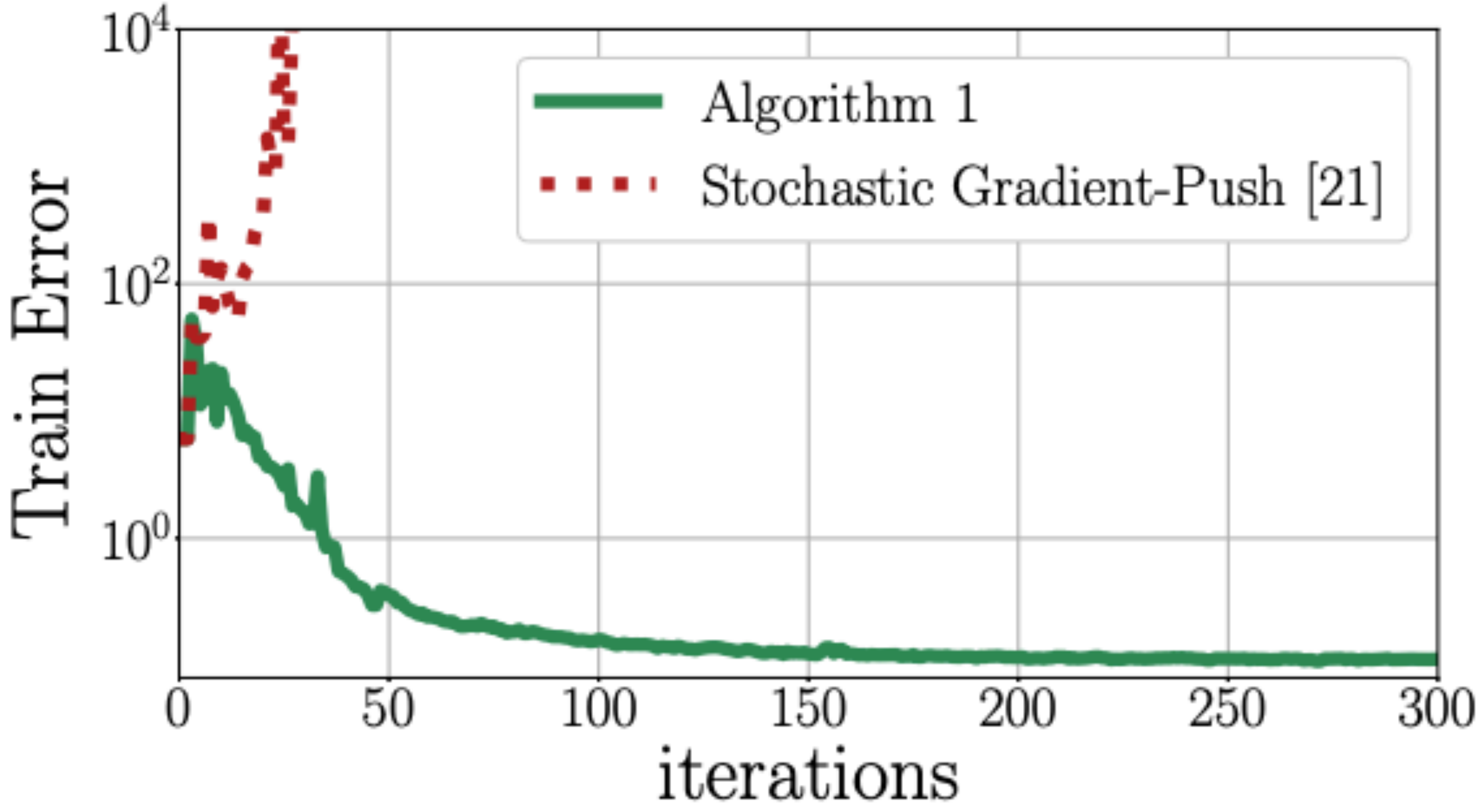
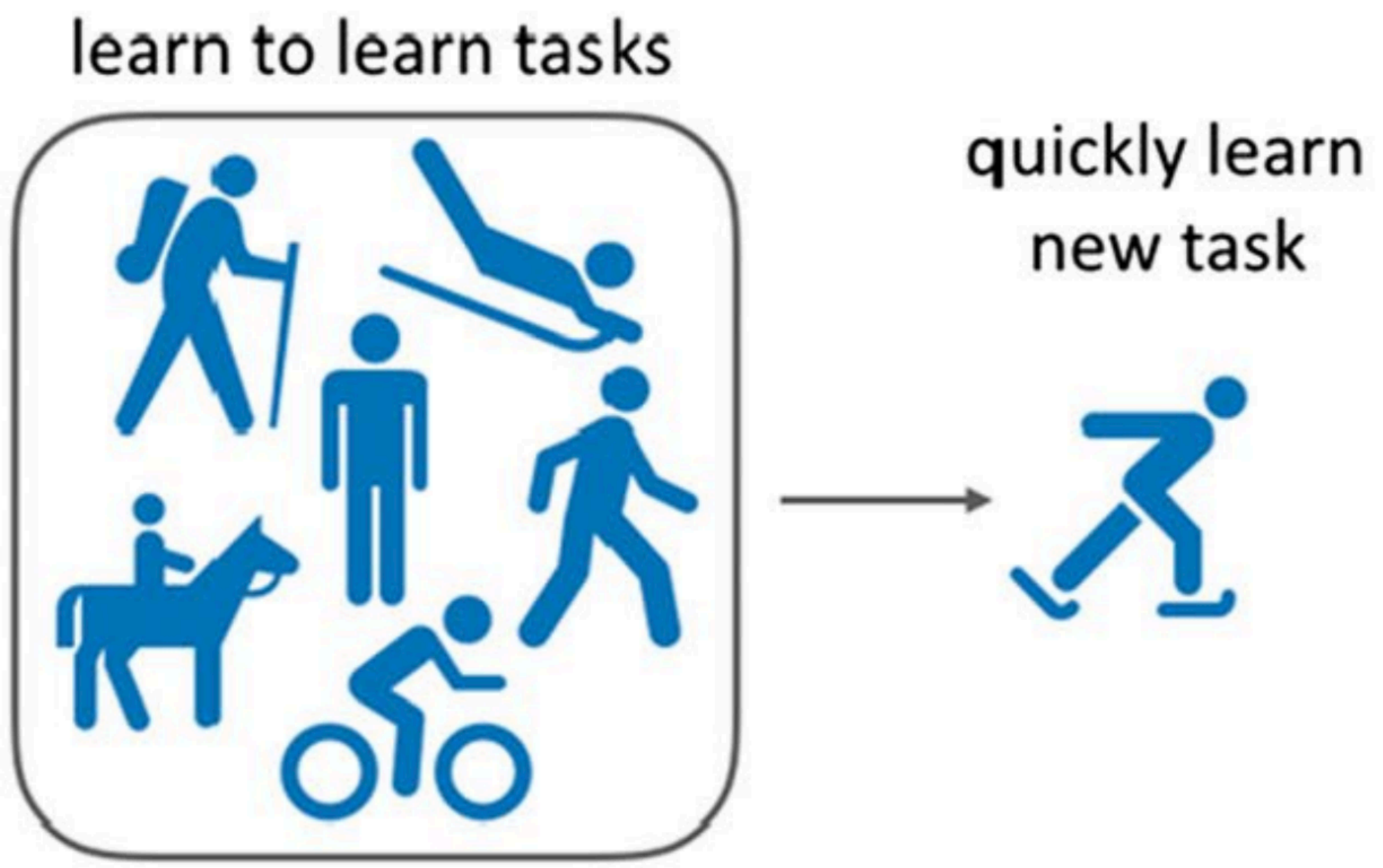


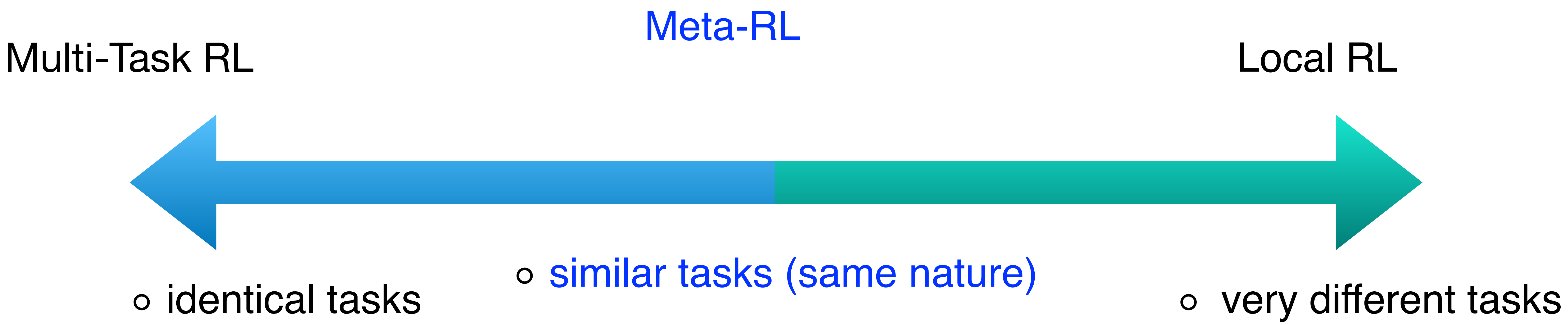

Fig. 3: Robustness to asynchronous communications, idle agents, message losses and delays.

# PART III

# On First-Order Meta-Reinforcement Learning with Moreau Envelopes

# Motivation


learn to learn tasks
quickly learn
new task


Meta-RL
Multi-Task RL
Local RL
identical tasks
similar tasks (same nature)
very different tasks

# Multi-Task RL Setup

- a set of Markov Decision Processes (MDPs) $\{\mathcal{M}_i\}_{i \in \mathcal{I}}$ from distribution $p$
- maximize the expected discounted reward over a finite number of steps $\{0, 1, \ldots, H\}$
- for each task $i \in \mathcal{I}$, the states and actions are $\mathcal{S}_i$ and $\mathcal{A}_i$
- initial state distribution $\mu_i : \mathcal{S}_i \to \Delta(\mathcal{S}_i)$
- transition kernel $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A}_i \to \Delta(\mathcal{S}_i)$, $\mathcal{P}_i(s_i'|s_i, a_i)$ is the probability of transitioning from state $s_i \in \mathcal{S}_i$ to $s_i' \in \mathcal{S}_i$ by taking action $a_i \in \mathcal{A}_i$
- reward function $r_i : \mathcal{S}_i \times \mathcal{A}_i \to [0, R]$
- discounted factor $\gamma \in (0, 1)$
- $\mathcal{M}_i = (\mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, r_i, \mu_i, \gamma)$
- value of a trajectory $\tau_i = (s_i^0, a_i^0, \ldots, a_i^{H-1}, s_i^H)$:

$$\mathcal{R}_i(\tau_i) := \sum_{h=0}^{H-1} \gamma^h r_i(s_i^h, a_i^h),$$

# Policy Gradient RL

- policy function $\pi_i : \mathcal{S}_i \to \Delta(\mathcal{A}_i)$ determines the probability of each action $a_i$ given a state $s_i$ as $\pi_i(a_i|s_i)$
- Policy Gradient Reinforcement Learning (PGRL): parameterize the policy by a $d$-dimensional parameter $w \in \mathbb{R}^d$ , i.e., $\pi_i(\cdot|\cdot; w)$
- the probability of trajectory $\tau_i = (s_i^0, a_i^0, \ldots, a_i^{H-1}, s_i^H)$ is

$$q_i(\tau_i; w) := \mu_i(s_i^0) \prod_{h=0}^{H-1} \pi_i(a_i^h|s_i^h; w) \prod_{h=0}^{H-1} \mathcal{P}_i(s_i^{h+1}|s_i^h, a_i^h),$$

- the average reward value for each task $i \in \mathcal{I}$ is

$$J_i(w) := \mathbb{E}_{\tau_i \sim q_i(\cdot; w)} \left[\mathcal{R}_i(\tau_i)\right],$$

- in multi-task reinforcement learning, we seek to optimize

$$J(w) := \mathbb{E}_{i \sim p} \left[J_i(w)\right].$$

# Policy Gradient Approach

- the full gradient of the value function is

$$\nabla J_i(w) := \mathbb{E}_{\tau_i \sim q_i(\cdot; w)} \left[ g_i(\tau_i; w) \right],$$

with stochastic policy gradient $g_i(\cdot; w)$

$$g_i(\tau_i; w) := \sum_{h=0}^{H-1} \nabla_w \log \pi_i(a_i^h | s_i^h; w) \, \mathcal{R}_i^h(\tau_i),$$

$$\text{where} \quad \mathcal{R}_i^h(\tau_i) := \sum_{l=h}^{H-1} \gamma^l \, r_i(s_i^l, a_i^l).$$

- To deal with the computational intractability of the full gradient, we approximate this term by a stochastic policy gradient over a batch $\mathcal{D}_i$ of trajectories sampled from distribution $q_i(\cdot; w)$, i.e.,

$$\nabla \tilde{J}_i(\mathcal{D}_i; w) := \frac{1}{|\mathcal{D}_i|} \sum_{\tau_i \in \mathcal{D}_i} g_i(\tau_i; w),$$

where $\nabla J_i(w) = \mathbb{E}\left[ \nabla \tilde{J}_i(\mathcal{D}_i; w) \right]$,

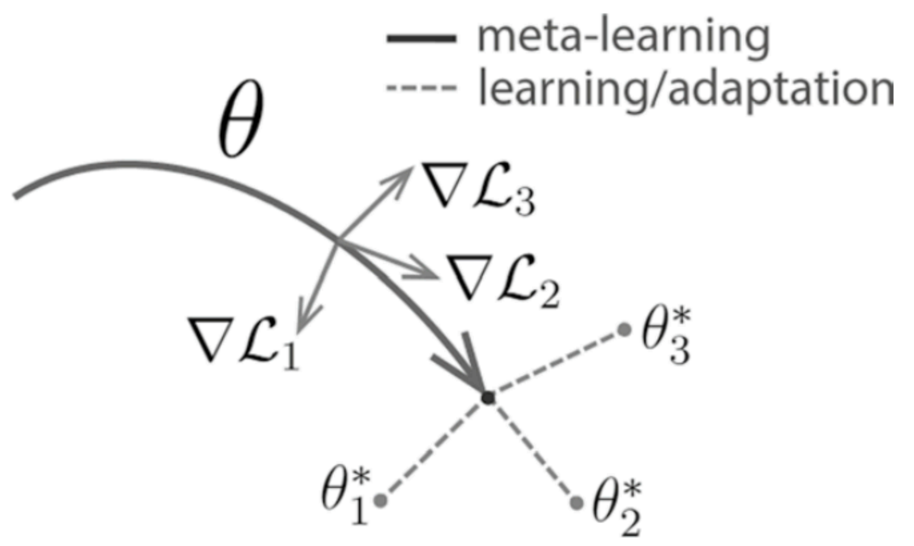
# Meta-Reinforcement Learning

- we formulate the joint multi-task setup via *Moreau Envelope Meta-Reinforcement Learning* cost (MEMRL)

$$\max_{w \in \mathbb{R}^d} V(w) := \mathbb{E}_{i\sim p}\left[V_i(w)\right]$$

$$\text{with} \quad V_i(w) := \max_{\theta_i \in \mathbb{R}^d}\left[J_i(\theta_i) - \frac{\lambda}{2}\|\theta_i - w\|^2\right],$$

- in Model-Agnostic Meta- Reinforcement Learning (MAML) framework, the goal is to maximize the following cost function:




$$\max_{w \in \mathbb{R}^d} V'(w) := \mathbb{E}_{i\sim p}\left[V_i'(w)\right],$$

$$\text{with} \quad V_i'(w) := J_i(w + \alpha \nabla J_i(w)).$$

# Moreau Envelope Meta-Reinforcement Learning (MEMRL)

**Algorithm 1** MEMRL: First-Order Moreau Envelope Meta-Reinforcement Learning

**input:** regularization parameter $\lambda$, inexact approximation precision $\nu$, meta stepsize $\alpha$, task batch size $B$, trajectory batch size $D$.
**initialize:** $w^0 \in \mathbb{R}^d, t \leftarrow 0$
**repeat**
  sample a batch of tasks $\mathcal{B}^t \subseteq \mathcal{I}$ with size $B$
  **for** all tasks $i \in \mathcal{B}^t$ **do**
    find $\tilde{\theta}_i(w^t)$ such that for a batch of trajectories $\mathcal{D}_i^t$ (of size $D$) sampled from $q_i(\cdot; \tilde{\theta}_i(w^t))$ to maximize $\tilde{F}_i(\cdot; \cdot, w^t)$ up to accuracy level $\nu$ with

$$\left\| \nabla \tilde{F}_i \left( \mathcal{D}_i^t; \tilde{\theta}_i(w^t), w^t \right) \right\| \leq \nu$$

  **end for**
  $w^{t+1} \leftarrow (1-\alpha\lambda) w^t + \frac{\alpha\lambda}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} \tilde{\theta}_i(w^t)$
  $t \leftarrow t+1$
**until** not converged
**output:**

$$\tilde{F}_i(\mathcal{D}_i; \theta_i, w) := \tilde{J}_i(\mathcal{D}_i; \theta_i) - \frac{\lambda}{2} \|\theta_i - w\|^2$$

1) $\theta_i^{t,0} \leftarrow w^t, k \leftarrow 0$,
2) sample a batch of trajectories $\mathcal{D}_i^{t,0}$ with size $D$ with respect to $q_i(\cdot; \theta_i^{t,0})$,
3) While not $\left\| \nabla \tilde{F}_i \left( \mathcal{D}_i^{t,k}; \theta_i^{t,k}, w^t \right) \right\| \leq \nu$:
   a) sample a batch of trajectories $\mathcal{D}_i^{t,k}$ with size $D$ with respect to $q_i(\cdot; \theta_i^{t,k})$,
   b) $\theta_i^{t,k+1} \leftarrow \theta_i^{t,k} + \beta \left[ \nabla \tilde{J}_i(\mathcal{D}_i^{t,k}; \theta_i^{t,k}) - \lambda(\theta_i^{t,k} - w^t) \right]$,
   c) $k \leftarrow k+1$,
4) $\tilde{\theta}_i(w^t) \leftarrow \theta_i^{t,k}$.

Bi-level optimization

# Convergence Result

**Lemma 2** (Properties of $V_i$). *Let Assumption 1 hold and $\lambda \geq \kappa \hat{L}$ for some $\kappa > 1$, and $\hat{G}, \hat{L}$ as in Lemma 1. Then, for all $i \in \mathcal{I}$ and $w, v \in \mathbb{R}^d$, the following properties hold:*

$$\left\|\nabla V_i(w)\right\| \leq \hat{G},$$
$$\left\|\nabla V_i(w) - \nabla V_i(v)\right\| \leq \tilde{L} \left\|w - v\right\|,$$

*where* $\tilde{L} := \frac{\lambda}{\kappa - 1}$.

**Theorem 1** (MEMRL Convergence). *Let Assumption 1 hold, $\lambda > \hat{L}$, and $\alpha = \frac{1}{4\tilde{L}}$. Then for any timestep $T \geq 4\tilde{L}^2$, the following property holds for the iterates of Algorithm 1:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla V(w^t)\|^2 \leq \frac{8R}{(1-\gamma)\sqrt{T}} + \frac{\lambda^2\nu^2}{(\lambda-\hat{L})^2} + \frac{8\tilde{L}\hat{G}^2}{B\sqrt{T}} + \frac{8\tilde{L}\lambda^2\nu^2}{(\lambda-\hat{L})^2 B\sqrt{T}} + \frac{8\alpha\tilde{L}\lambda^2\hat{G}^2}{(\lambda-\hat{L})^2 BD\sqrt{T}},$$

*where $\hat{G}, \hat{L}$ as in Lemma 1, and $\tilde{L}$ as in Lemma 2.*

# Numerical Experiment

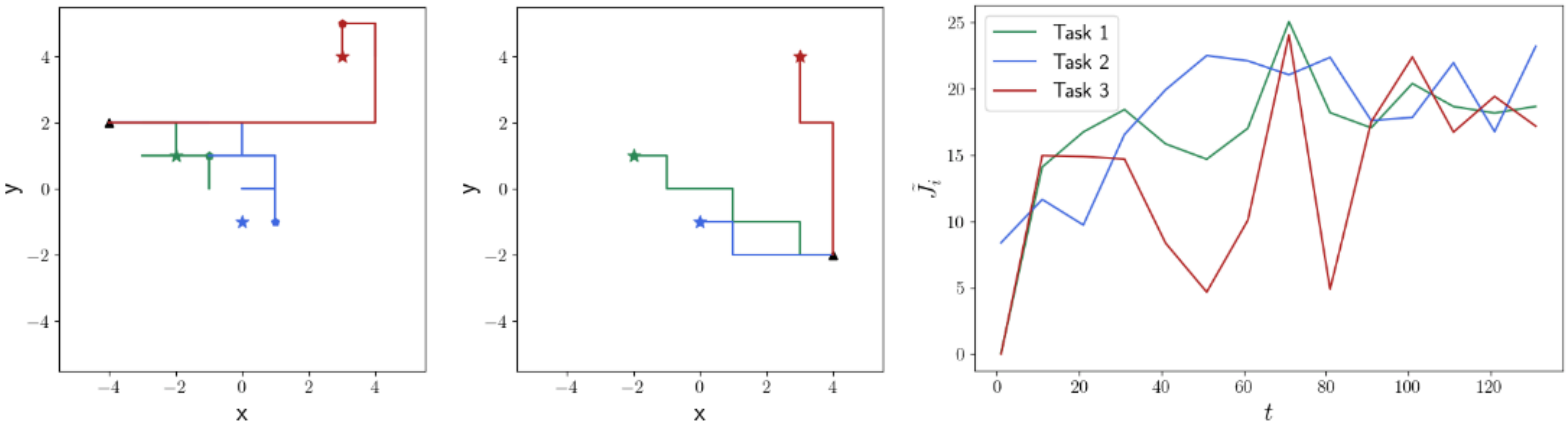



Fig. 1: The performance of our `MEMRL` algorithm on discrete 2D-navigation for $|\mathcal{I}|=3$ tasks with different underlying MDPs. **(Left)** The navigation map at iteration $t = 0$ starting from a random location (black triangle) on the grid. The stars indicate the destination of each task $i \in \mathcal{I}$. Pentagons indicate the end of a trajectory when it fails to reach its destination (star). **(Middle)** The navigation map at iteration $t = 120$, where the adapted meta-policy for each task is optimal. **(Right)** The evolution of individual reward functions given the adapted meta-policy on each task. Each curve is the empirical mean of the reward obtain over 10 independent trajectories conditioned on the approximated policy parameter $\tilde{\theta}_i^t$.

# Conclusion

We:

- studied federated learning under personalization and asynchronous updates
- proposed PersA-Fl algorithm to address this problem
- showed a first-order stationary convergence for our proposed algorithm under both MAML and ME personalization costs
- compared the performance of our algorithm with its counterparts on heterogeneous data

# Conclusion

We:

- studied decentralized optimization under personalization and asynchronous updates with message loss and delay,
- proposed PARS-Push algorithm for personalized, asynchronous, and robust decentralized optimization,
- showed the convergence of our algorithm for strongly-convex and non-convex function classes.

# Discussion

- Formulated the Meta-Reinforcement Learning problem with Moreau Envelopes
- Studied the convergence analysis of this problem for non-convex setups
- Provided numerical results of the performance of this formulation on $2$D navigation problem

---

- Extending the theoretical analysis to the convex function class
- Study this problem for distributed multi-agent setups
- Exploring the connections of this problem to LP