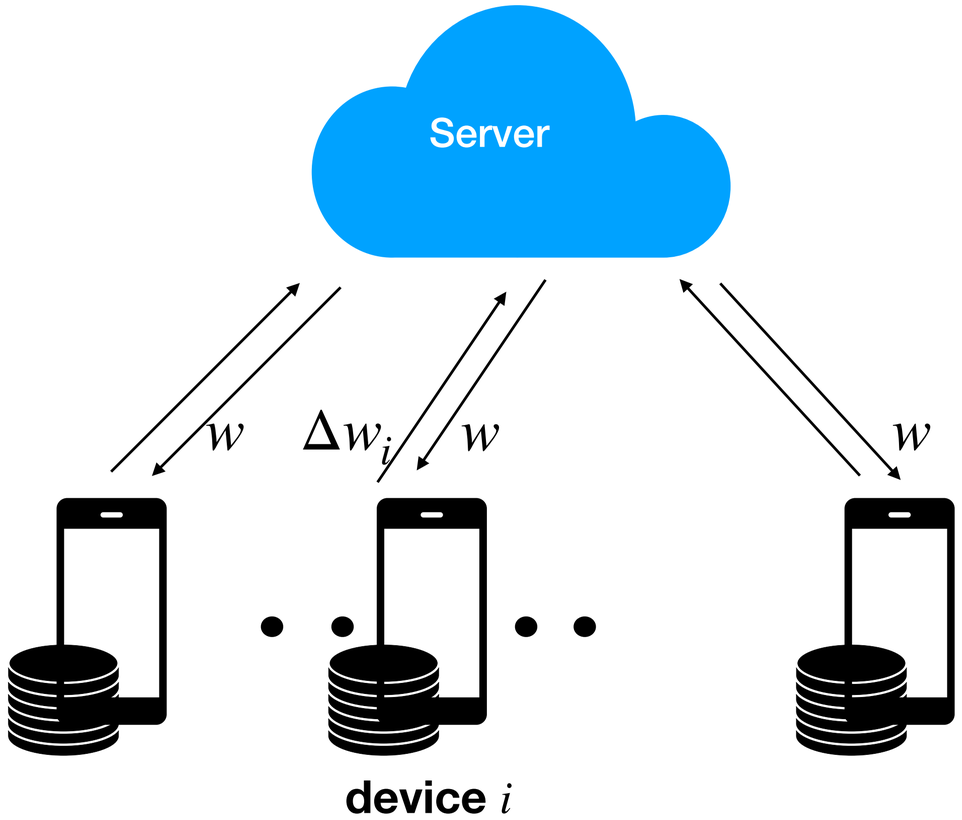
Accelerated & Adaptive Federated Multi-task Learning

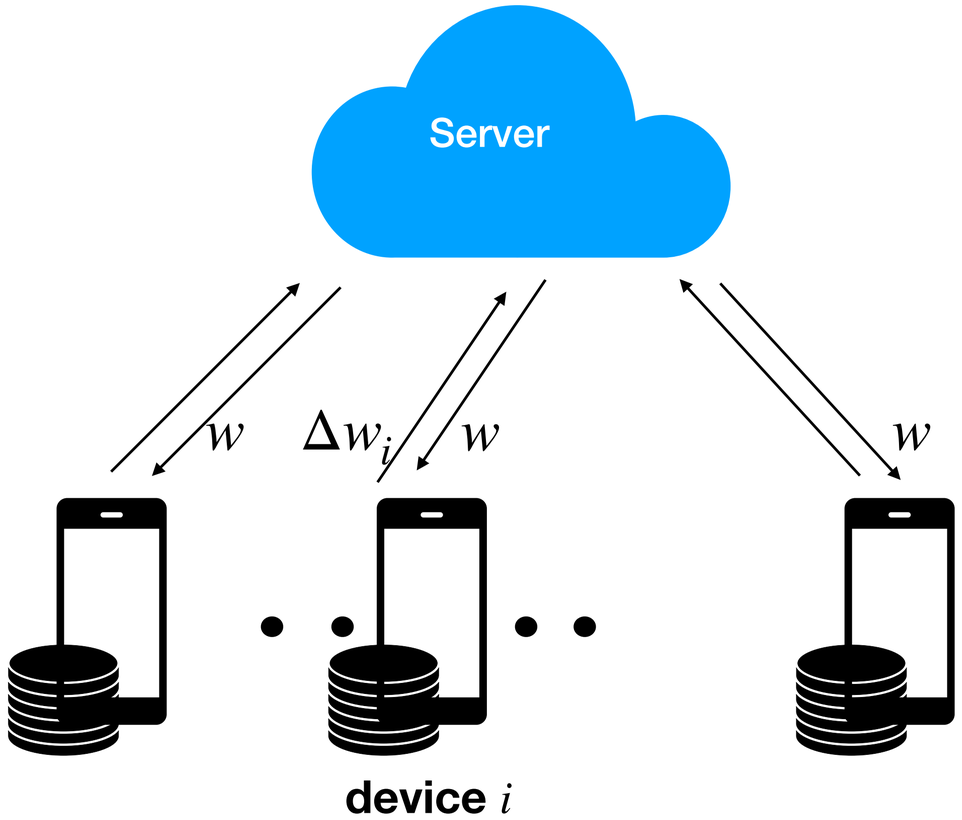
Yang Zhang, Pratik Patil, Kin Gutierrez

Federated Multi-Task Learning, V. Smith et. al., NIPS 2017

Federated Learning



Federated Learning



Why? ✓Relief on network ✓privacy ✓flexible addition of new data ✓model aggregation

Challenges of Federated Learning

Non-IID

Unbalanced

Underlying structure among nodes

Statistical challenges

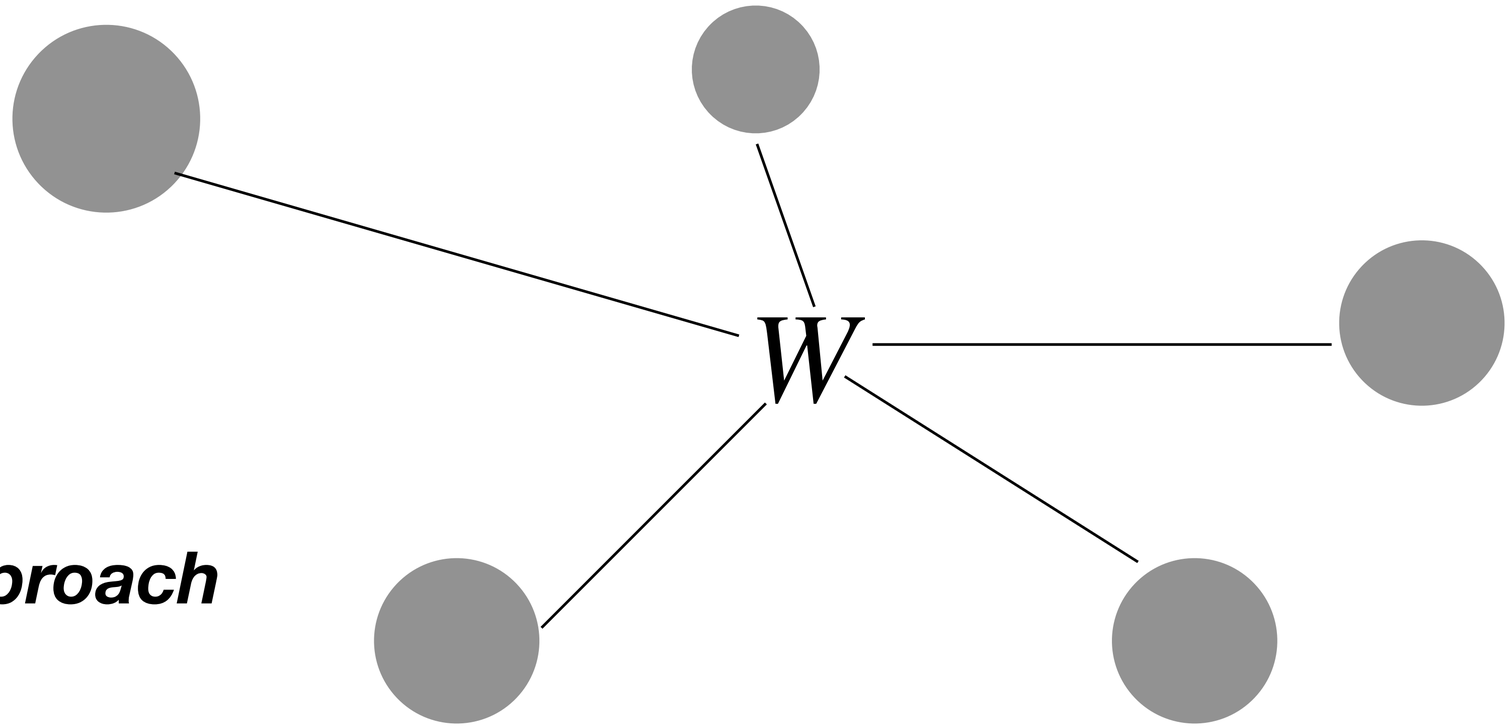
Massively distributed

Limited communication

Straggler effect

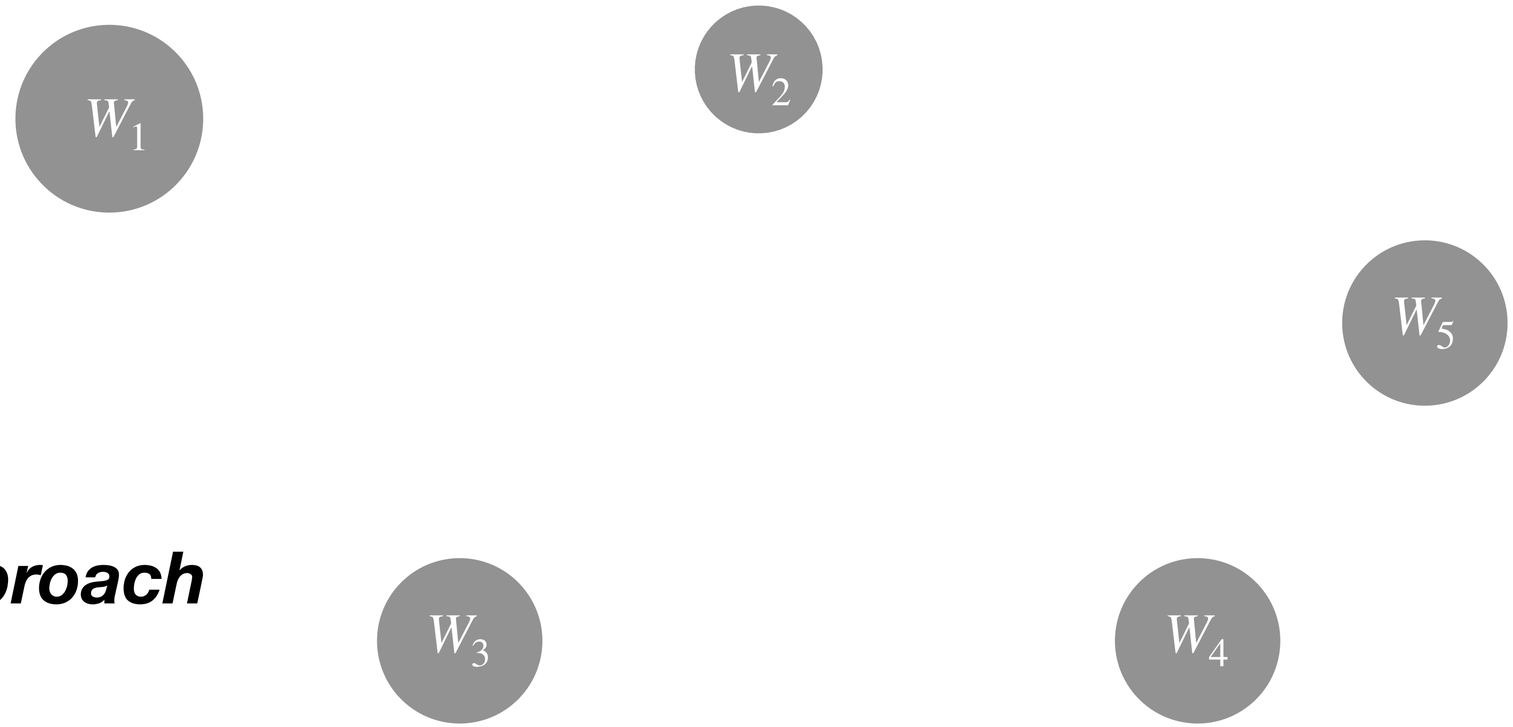
System challenges

Why Multi-task Learning?



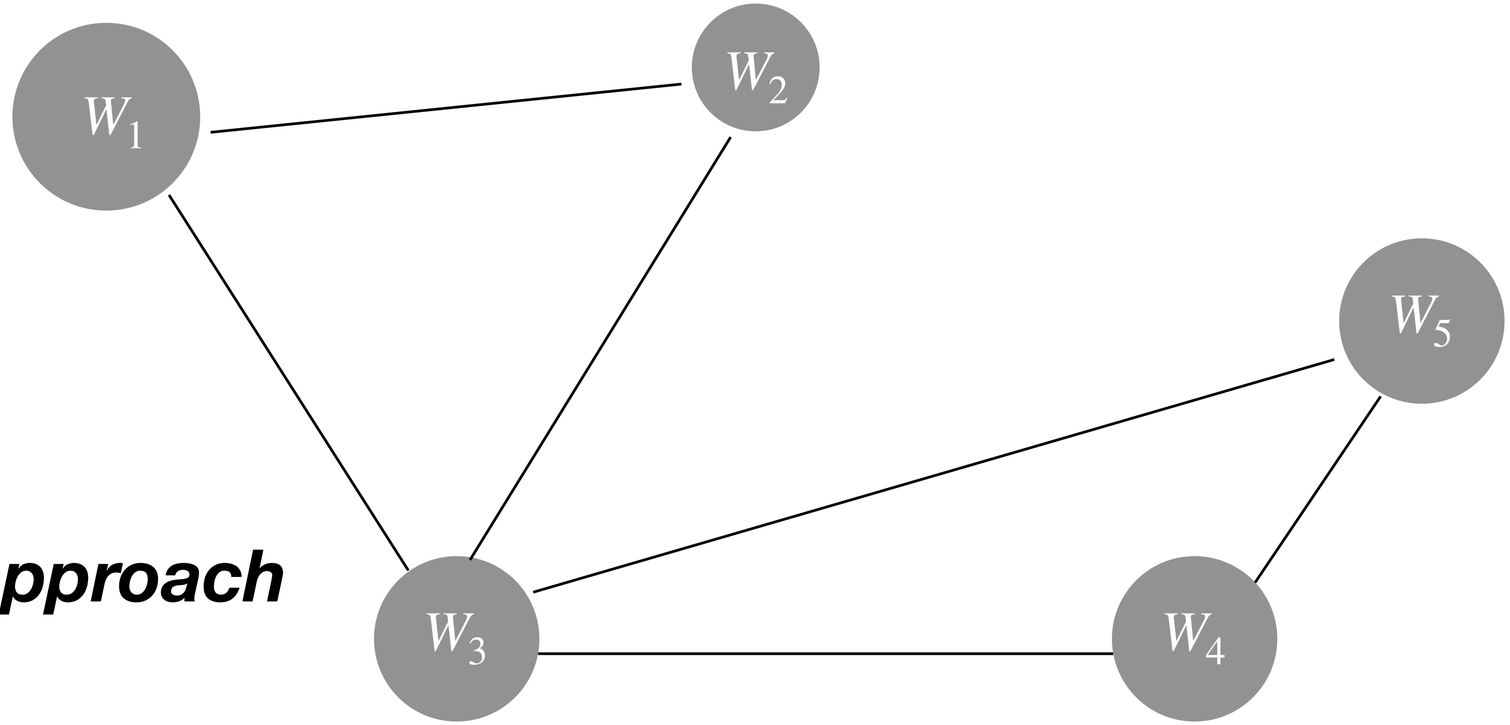
Global approach

Why Multi-task Learning?



Local approach

Multi-task Learning Uses Underlying Structure



Multi-task approach

Mocha Framework: Multi-task Federated Learning

Primal

$$\min_{\mathbf{W},\mathbf{\Omega}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \ell_k \left(\mathbf{w}_k^T \mathbf{x}_{k,j}, y_{k,j} \right) + \mathscr{R}(\mathbf{W}, \mathbf{\Omega})$$

Mocha Framework: Multi-task Federated Learning

Primal

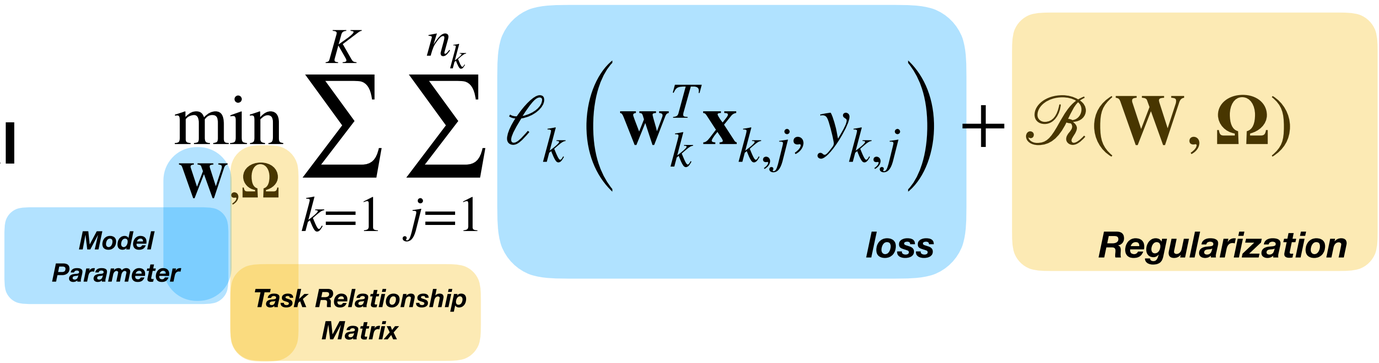
$$\min_{\mathbf{W},\mathbf{\Omega}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \ell_k\left(\mathbf{w}_k^T \mathbf{x}_{k,j}, y_{k,j}\right) + \mathscr{R}(\mathbf{W}, \mathbf{\Omega})$$

Model Parameter

loss

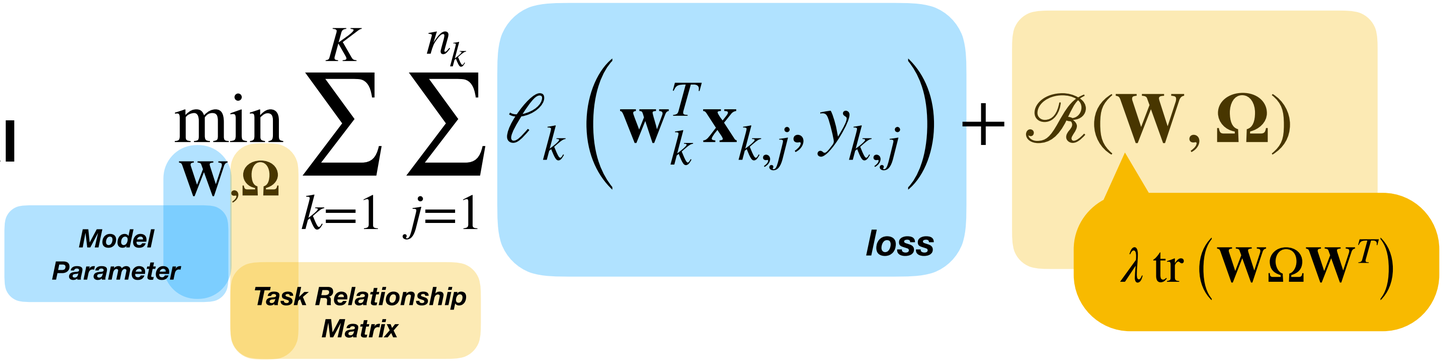
Mocha Framework: Multi-task Federated Learning

Primal



Mocha Framework: Multi-task Federated Learning

Primal



Mocha Framework: Multi-task Federated Learning

Primal

$$\min_{\mathbf{W},\boldsymbol{\Omega}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \ell_k \left(\mathbf{w}_k^T \mathbf{x}_{k,j}, y_{k,j} \right) + \mathscr{R}(\mathbf{W}, \boldsymbol{\Omega})$$

$$\geq$$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} - \ell_k^* \left(-\alpha_{k,j} \right) - \mathscr{R}^*(\mathbf{X}\boldsymbol{\alpha})$$

Mocha Framework: Multi-task Federated Learning

Primal

$$\min_{\mathbf{W},\boldsymbol{\Omega}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \ell_k \left(\mathbf{w}_k^T \mathbf{x}_{k,j}, y_{k,j} \right) + \mathscr{R}(\mathbf{W}, \boldsymbol{\Omega})$$

$$\geq$$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} - \ell_k^* \left(-\alpha_{k,j} \right) - \mathscr{R}^*(\mathbf{X}\boldsymbol{\alpha})$$

Mocha Framework: Multi-task Federated Learning

Primal

$$\min_{\mathbf{W},\boldsymbol{\Omega}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \frac{1}{n_k} \left[1 - \left(\mathbf{w}_k^T x_{k,j} \right) y_{k,j} \right]_+ + \frac{\lambda}{2} \mathbf{w}^T \bar{\boldsymbol{\Omega}} \mathbf{w}$$

$$\geq$$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} y_{k,j}\, \alpha_{k,j} - \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{X}^T \bar{\boldsymbol{\Omega}}^{-1} \mathbf{X} \boldsymbol{\alpha}$$

Dual Framework

Global objective

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} y_{k,j}\, \alpha_{k,j} \; - \; \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{X}^T \bar{\boldsymbol{\Omega}}^{-1} \mathbf{X} \boldsymbol{\alpha}$$

Dual Framework

Global objective

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} y_{k,j}\, \alpha_{k,j} \; - \; \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{X}^T \bar{\boldsymbol{\Omega}}^{-1} \mathbf{X} \boldsymbol{\alpha}$$

Dual Framework

Global objective

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} y_{k,j}\, \alpha_{k,j} \; - \; \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{X}^T \bar{\boldsymbol{\Omega}}^{-1} \mathbf{X} \boldsymbol{\alpha}$$

Local objective

$$\max_{\boldsymbol{\alpha}_k^+} \sum_{j=1}^{n_k} y_{k,j}\, \alpha_{k,j} \; - \; \left\langle \mathbf{w}\left(\boldsymbol{\alpha}^{(t)}\right)_k, \mathbf{X}_k \left(\boldsymbol{\alpha}_k^{(t+1)} - \boldsymbol{\alpha}_k^{(t)}\right) \right\rangle$$

$$- \frac{\sigma'}{2\lambda} \left\| \mathbf{X}_k \left(\boldsymbol{\alpha}_k^{(t+1)} - \boldsymbol{\alpha}_k^{(t)}\right) \right\|^2_{\bar{\boldsymbol{\Omega}}_k^{-1}} - \; \frac{1}{K} \mathscr{R}^*(\mathbf{X}\boldsymbol{\alpha})$$

Mocha: Primal-Dual Framework

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over* K *nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\ \boldsymbol{\alpha}_k^{(t+1)};\ \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Mocha: Primal-Dual Framework

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$

Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**

  Set subproblem σ' and number of federated iterations T_i

  for $t = 0, 1, \ldots T_i$ **do**

    Send $\mathbf{w}_k^{(t)}$ to each node

    for $k = 1, \ldots, K$ *in parallel over* K *nodes* **do**

      Solve local dual problem $\mathcal{D}_k(\ \boldsymbol{\alpha}_k^{(t+1)};\ \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$

        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$

      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server

    end

    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server

  end

  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$

end

return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Communication Round

Mocha: Primal-Dual Framework

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over* K *nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Communication Round

Local Computation

Mocha: Primal-Dual Framework

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over* K *nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Communication Round

Local Computation

Mocha Drawbacks

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over K nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\ \boldsymbol{\alpha}_k^{(t+1)};\ \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Intensive Local Computation

Mocha Drawbacks

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over K nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Intensive Local Computation

-> naive acceleration

Mocha Drawbacks

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

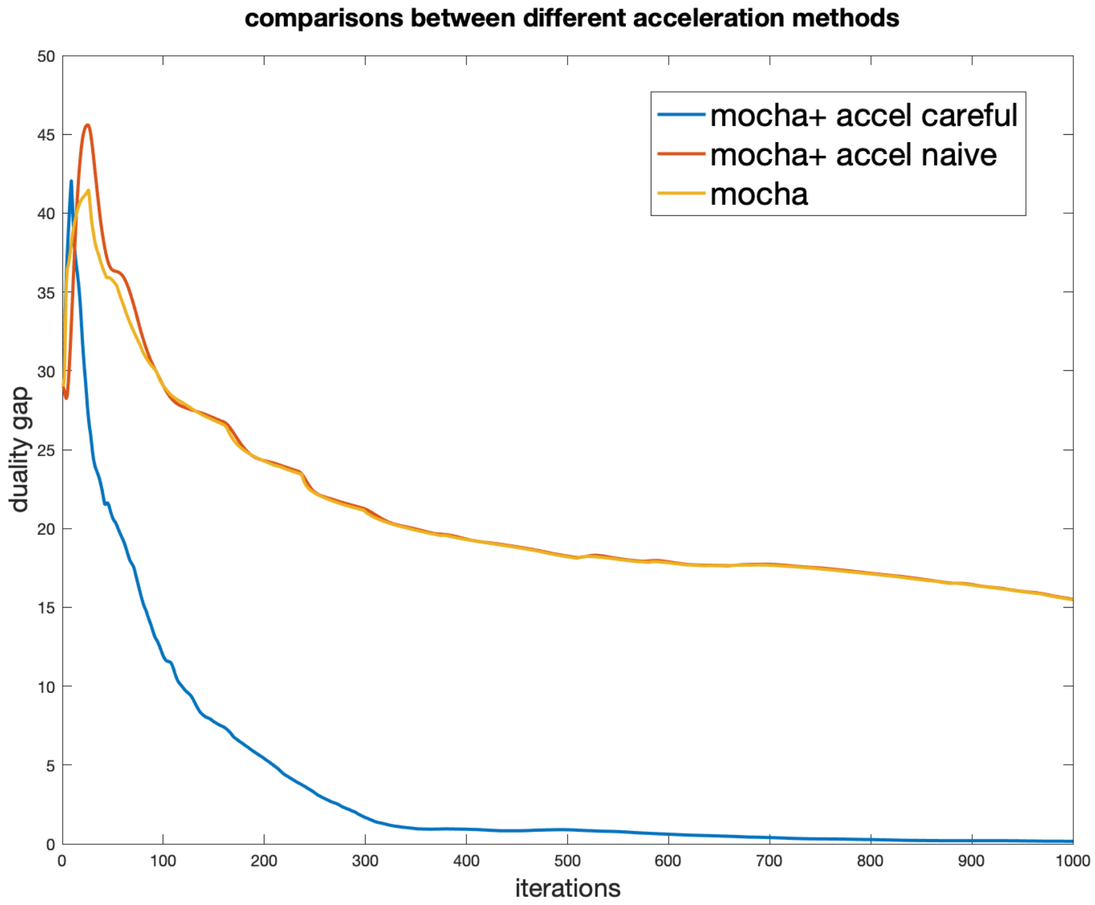
for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over K nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

Intensive Local Computation

-> careful acceleration

Why Acceleration?

Faster local computation



Mocha Drawbacks

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over* K *nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

High Communication

Mocha Drawbacks

Algorithm 1: Vanilla Mocha: Primal Dual Framework

Input: Data $\mathbf{X}_k$ from $k = 1, \ldots, K$ tasks, initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^{K \times d}$

for $i = 0, 1, \ldots$ **do**
  Set subproblem σ' and number of federated iterations T_i
  for $t = 0, 1, \ldots T_i$ **do**
    Send $\mathbf{w}_k^{(t)}$ to each node
    for $k = 1, \ldots, K$ *in parallel over K nodes* **do**
      Solve local dual problem $\mathcal{D}_k(\boldsymbol{\alpha}_k^{(t+1)}; \mathbf{X}_k, \boldsymbol{\alpha}_k^{(t)}, \mathbf{w}_k^{(t)}, \mathbf{\Omega}_{(k,k)}^{-1}, \sigma')$
        returning approximate solution $\boldsymbol{\alpha}_k^{(t+1)}$
      return $\mathbf{X}_k \boldsymbol{\alpha}_k^{(t+1)}$ to server
    end
    Compute $\mathbf{w}^{(t+1)} = \nabla \mathcal{R}^\star(\mathbf{X}\boldsymbol{\alpha}^{(t+1)})$ on server
  end
  Update $\mathbf{\Omega}$ centrally based on latest $\mathbf{w}$
end
return $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_K]$

High Communication

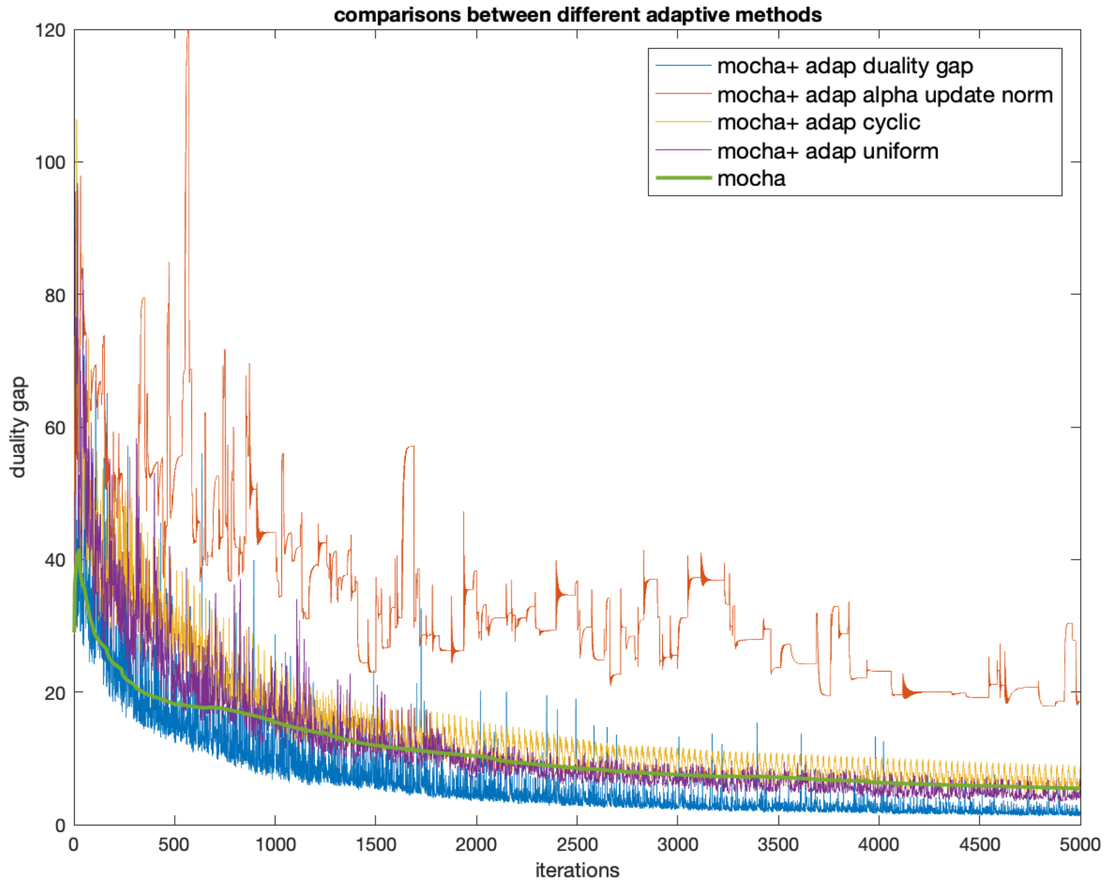
-> adaptive node selection using duality gap

Why Adaptiveness?

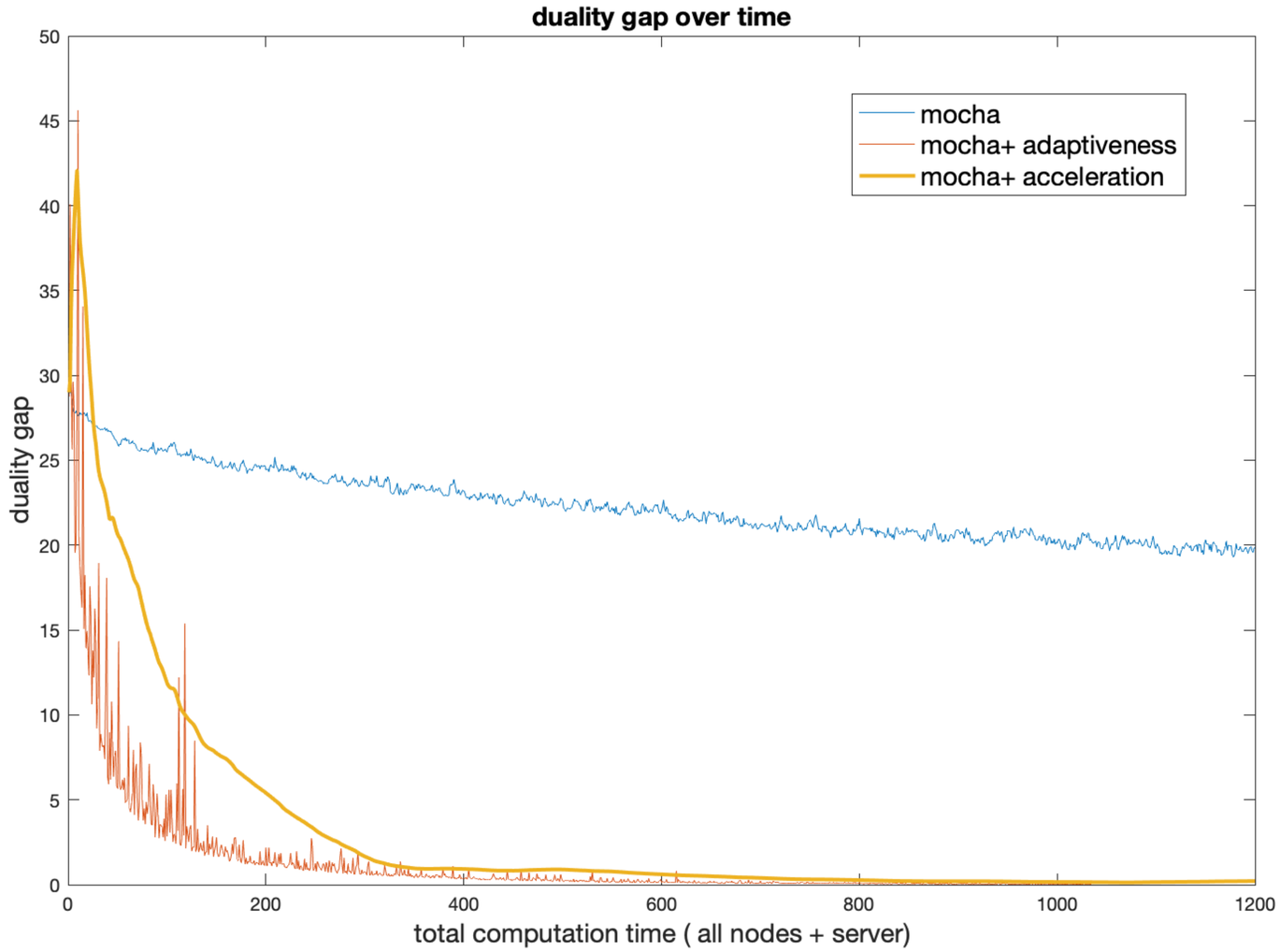
Less communication

-> Faster iterations

-> more tolerant to stragglers



Total Computation Comparison



Summary & Future Work

Federated multi-task learning framework

Extension:

- ✔ Adaptive node selection -> lower communication cost
- ✔ Accelerated local computation -> lower computation cost
- Acceleration on top of adaptiveness