

Federated Multi-Task Learning

Zijiang Yang

Background

- Federated learning
 - Statistical challenges:
 - Non-IID
 - Number of datapoints differs in each node
 - System challenges:
 - Communication bottleneck
 - Storage, computational and communication capacities differ in each node (stragglers)
- MTL (Multi-Task Learning)
 - System challenges : node heterogeneity, stragglers, fault tolerance✕
 - Goal:
 - Suggest new method MOCHA to solve a general MTL problem (based on COCOA)
 - Prove convergence
 - Evaluate MOCHA with new benchmarking federated datasets

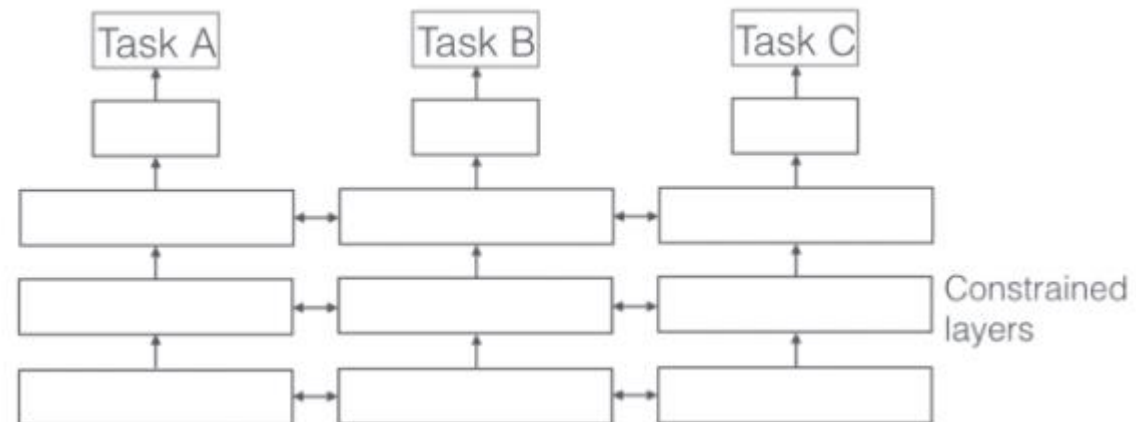
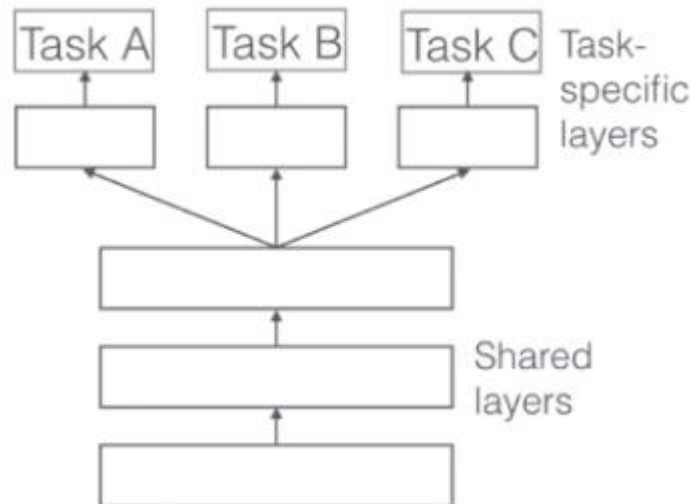
MTL

- Single-task learning:

- Input data 1 \rightarrow model 1 \rightarrow prediction 1
- Input data 2 \rightarrow model 2 \rightarrow prediction 2

- Multi-task learning:

Input data \rightarrow MTL model \rightarrow prediction 1
 \rightarrow prediction 2
 \rightarrow prediction 3



COCO A

- Optimization problem:
$$\min_{\mathbf{w} \in \mathbb{R}^d} \left[P(\mathbf{w}) := \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \ell_i(\mathbf{w}^T \mathbf{x}_i) \right],$$
- Dual problem:
$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left[D(\boldsymbol{\alpha}) := -\frac{\lambda}{2} \|A\boldsymbol{\alpha}\|^2 - \frac{1}{n} \sum_{i=1}^n \ell_i^*(-\alpha_i) \right],$$

Algorithm 1: CoCoA: Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$, scaling parameter $1 \leq \beta_K \leq K$ (default: $\beta_K := 1$).

Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ distributed over K machines

Initialize: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow \mathbf{0}$ for all machines k , and $\mathbf{w}^{(0)} \leftarrow \mathbf{0}$

for $t = 1, 2, \dots, T$

for all machines $k = 1, 2, \dots, K$ *in parallel*

$(\Delta \boldsymbol{\alpha}_{[k]}, \Delta \mathbf{w}_k) \leftarrow \text{LOCALDUALMETHOD}(\boldsymbol{\alpha}_{[k]}^{(t-1)}, \mathbf{w}^{(t-1)})$

$\boldsymbol{\alpha}_{[k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[k]}^{(t-1)} + \frac{\beta_K}{K} \Delta \boldsymbol{\alpha}_{[k]}$

end

reduce $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} + \frac{\beta_K}{K} \sum_{k=1}^K \Delta \mathbf{w}_k$

end

MOCHA

- Data $X_t \in R^{d \times n_t}$, m nodes, separate weight $w_t \in R^d$ and convex loss function ℓ_t for each node

- Global task:
$$\min_{\mathbf{W}, \mathbf{\Omega}} \left\{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(\mathbf{w}_t^T \mathbf{x}_t^i, y_t^i) + \mathcal{R}(\mathbf{W}, \mathbf{\Omega}) \right\}$$

- $\mathbf{W} := [w_1, \dots, w_m] \in R^{d \times m}$ is weight matrix, $\mathbf{\Omega} \in R^{m \times m}$ is models relationships among tasks.

$$\mathcal{R}(\mathbf{W}, \mathbf{\Omega}) = \lambda_1 \text{tr}(\mathbf{W} \mathbf{\Omega} \mathbf{W}^T) + \lambda_2 \|\mathbf{W}\|_F^2,$$

Observations

$$\min_{\mathbf{W}, \Omega} \left\{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(\mathbf{w}_t^T \mathbf{x}_t^i, y_t^i) + \mathcal{R}(\mathbf{W}, \Omega) \right\}$$

- This problem is not always jointly convex in W and Ω . Even it is, it's difficult to solve it for W and Ω simultaneously.
- When fixing Ω , updating W depends on X and Ω
- When fixing W , updating Ω only depends on W and not X
- Fix either W or Ω and optimize over the other at each iteration

COCO A

Algorithm 1: CoCoA: Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$, scaling parameter $1 \leq \beta_K \leq K$ (default: $\beta_K := 1$).

Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ distributed over K machines

Initialize: $\alpha_{[k]}^{(0)} \leftarrow \mathbf{0}$ for all machines k , and $\mathbf{w}^{(0)} \leftarrow \mathbf{0}$

for $t = 1, 2, \dots, T$

for all machines $k = 1, 2, \dots, K$ *in parallel*

$(\Delta \alpha_{[k]}, \Delta \mathbf{w}_k) \leftarrow \text{LOCALDUALMETHOD}(\alpha_{[k]}^{(t-1)}, \mathbf{w}^{(t-1)})$

$\alpha_{[k]}^{(t)} \leftarrow \alpha_{[k]}^{(t-1)} + \frac{\beta_K}{K} \Delta \alpha_{[k]}$

end

reduce $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} + \frac{\beta_K}{K} \sum_{k=1}^K \Delta \mathbf{w}_k$

end

COCOA

Procedure A: LOCALDUALMETHOD: Dual algorithm for prob. (2) on a single coordinate block k

Input: Local $\alpha_{[k]} \in \mathbb{R}^{n_k}$, and $w \in \mathbb{R}^d$ consistent with other coordinate blocks of α s.t. $w = A\alpha$

Data: Local $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_k}$

Output: $\Delta\alpha_{[k]}$ and $\Delta w := A_{[k]}\Delta\alpha_{[k]}$

Procedure B: LOCALSDCA: SDCA iterations for problem (2) on a single coordinate block k

Input: $H \geq 1$, $\alpha_{[k]} \in \mathbb{R}^{n_k}$, and $w \in \mathbb{R}^d$ consistent with other coordinate blocks of α s.t. $w = A\alpha$

Data: Local $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_k}$

Initialize: $w^{(0)} \leftarrow w$, $\Delta\alpha_{[k]} \leftarrow \mathbf{0} \in \mathbb{R}^{n_k}$

for $h = 1, 2, \dots, H$

choose $i \in \{1, 2, \dots, n_k\}$ *uniformly at random*

find $\Delta\alpha$ *maximizing* $-\frac{\lambda n}{2} \|w^{(h-1)} + \frac{1}{\lambda n} \Delta\alpha \mathbf{x}_i\|^2 - \ell_i^*(-(\alpha_i^{(h-1)} + \Delta\alpha))$

$\alpha_i^{(h)} \leftarrow \alpha_i^{(h-1)} + \Delta\alpha$

$(\Delta\alpha_{[k]})_i \leftarrow (\Delta\alpha_{[k]})_i + \Delta\alpha$

$w^{(h)} \leftarrow w^{(h-1)} + \frac{1}{\lambda n} \Delta\alpha \mathbf{x}_i$

end

Output: $\Delta\alpha_{[k]}$ and $\Delta w := A_{[k]}\Delta\alpha_{[k]}$

MOCHA Algorithm

Algorithm 1 MOCHA: Federated Multi-Task Learning Framework

```
1: Input: Data  $\mathbf{X}_t$  from  $t = 1, \dots, m$  tasks, stored on one of  $m$  nodes, and initial matrix  $\mathbf{\Omega}_0$ 
2: Starting point  $\alpha^{(0)} := \mathbf{0} \in \mathbb{R}^n, \mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^b$ 
3: for iterations  $i = 0, 1, \dots$  do
4:   Set subproblem parameter  $\sigma'$  and number of federated iterations,  $H_i$ 
5:   for iterations  $h = 0, 1, \dots, H_i$  do
6:     for tasks  $t \in \{1, 2, \dots, m\}$  in parallel over  $m$  nodes do
7:       call local solver, returning  $\theta_t^h$ -approximate solution  $\Delta\alpha_t$  of the local subproblem (4)
8:       update local variables  $\alpha_t \leftarrow \alpha_t + \Delta\alpha_t$ 
9:       return updates  $\Delta\mathbf{v}_t := \mathbf{X}_t\Delta\alpha_t$ 
10:    reduce:  $\mathbf{v}_t \leftarrow \mathbf{v}_t + \Delta\mathbf{v}_t$ 
11:  Update  $\mathbf{\Omega}$  centrally based on  $\mathbf{w}(\alpha)$  for latest  $\alpha$ 
12: Central node computes  $\mathbf{w} = \mathbf{w}(\alpha)$  based on the latest  $\alpha$ 
13: return:  $\mathbf{W} := [\mathbf{w}_1, \dots, \mathbf{w}_m]$ 
```

$$\mathbf{w}(\alpha) = \nabla R^*(X\alpha), \mathbf{v} = X\alpha$$

Lagrangian

standard form problem (not necessarily convex)

$$\begin{array}{ll}\text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_i(x) = 0, \quad i = 1, \dots, p\end{array}$$

variable $x \in \mathbf{R}^n$, domain \mathcal{D} , optimal value p^\star

Lagrangian: $L : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^p \rightarrow \mathbf{R}$, with $\text{dom } L = \mathcal{D} \times \mathbf{R}^m \times \mathbf{R}^p$,

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x)$$

Lagrange Dual Function

Lagrange dual function: $g : \mathbf{R}^m \times \mathbf{R}^p \rightarrow \mathbf{R}$,

$$\begin{aligned} g(\lambda, \nu) &= \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) \\ &= \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right) \end{aligned}$$

Karush Kuhn Tucker (KKT) conditions

the following four conditions are called KKT conditions (for a problem with differentiable f_i, h_i):

1. primal constraints: $f_i(x) \leq 0, i = 1, \dots, m, h_i(x) = 0, i = 1, \dots, p$
2. dual constraints: $\lambda \succeq 0$
3. complementary slackness: $\lambda_i f_i(x) = 0, i = 1, \dots, m$
4. gradient of Lagrangian with respect to x vanishes:

$$\nabla f_0(x) + \sum_{i=1}^m \lambda_i \nabla f_i(x) + \sum_{i=1}^p \nu_i \nabla h_i(x) = 0$$


Slater's Constraint Qualification

strong duality holds for a convex problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & Ax = b \end{array}$$

if it is strictly feasible, *i.e.*,

$$\exists x \in \text{int } \mathcal{D} : \quad f_i(x) < 0, \quad i = 1, \dots, m, \quad Ax = b$$


$$p^* = d^*$$

Dual problem

$$\min_{\mathbf{W}, \Omega} \left\{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(\mathbf{w}_t^T \mathbf{x}_t^i, y_t^i) + \mathcal{R}(\mathbf{W}, \Omega) \right\}$$

- $X := \text{Diag}(X_1, \dots, X_m)$, Ω fixed

- Primal problem equals to:

$$\min_{Z, W} \{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(z_t^i, y_t^i) + R(W, \Omega) \} \quad \text{with restrictions: } w_t^T x_t^i = z_t^i$$

- Dual function:

$$\begin{aligned} D_0(\alpha) &= \min_{Z, W} \{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(z_t^i, y_t^i) + \alpha_t^i (w_t^T x_t^i - z_t^i) + R(W, \Omega) \} \\ &= \min_{Z, W} \{ \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t(z_t^i, y_t^i) - \alpha_t^i z_t^i + \alpha_t^i w_t^T x_t^i + R(W, \Omega) \} \\ &= - \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t^*(-\alpha_t^i) + R^*(X\alpha) \end{aligned}$$

Dual problem: $\max_{\alpha} D_0(\alpha)$

$$\min_{\alpha} \left\{ \mathcal{D}(\alpha) := \sum_{t=1}^m \sum_{i=1}^{n_t} \ell_t^*(-\alpha_t^i) + \mathcal{R}^*(\mathbf{X}\alpha) \right\},$$

Subproblem

- Quadratic approximation of the dual problem
 - Only access data available only

$$\min_{\Delta \alpha_t} \mathcal{G}_t^{\sigma'}(\Delta \alpha_t; \mathbf{v}_t, \alpha_t) := \sum_{i=1}^{n_t} \ell_t^*(-\alpha_t^i - \Delta \alpha_t^i) + \langle \mathbf{w}_t(\alpha), \mathbf{X}_t \Delta \alpha_t \rangle + \frac{\sigma'}{2} \|\mathbf{X}_t \Delta \alpha_t\|_{\mathbf{M}_t}^2 + c(\alpha)$$

- Get $\Delta \alpha_t$
- $w(\alpha) = \nabla R^*(X\alpha)$, requires $v = X\alpha$ to compute

Practical considerations

- To solve straggler effect: allow node t solve its subproblem approximately, where the quality of the approximation is controlled by θ_t^h
 - Statistical challenges: size of X_t , difficulty of subproblem
 - System challenges: storage, computation, communication capacities
 - Global clock cycle: deadline for receiving updates in central node
- $\theta_t^h=0$: exact solution
- $\theta_t^h=1$: no progress in node t , iteration h
- COCOA: θ for all nodes and iterations
- MOCHA: allows stragglers and fault tolerance

Convergence

$$\theta_t^h := \frac{\mathcal{G}_t^{\sigma'}(\Delta \alpha_t^{(h)}; \mathbf{v}^{(h)}, \alpha_t^{(h)}) - \mathcal{G}_t^{\sigma'}(\Delta \alpha_t^*; \mathbf{v}^{(h)}, \alpha_t^{(h)})}{\mathcal{G}_t^{\sigma'}(\mathbf{0}; \mathbf{v}^{(h)}, \alpha_t^{(h)}) - \mathcal{G}_t^{\sigma'}(\Delta \alpha_t^*; \mathbf{v}^{(h)}, \alpha_t^{(h)})},$$

Theorem 1. Assume that the losses ℓ_t are $(1/\mu)$ -smooth. Then, under Assumptions 1 and 2, there exists a constant $s \in (0, 1]$ such that for any given convergence target $\epsilon_{\mathcal{D}}$, choosing H such that

$$H \geq \frac{1}{(1 - \bar{\Theta})s} \log \frac{n}{\epsilon_{\mathcal{D}}}, \quad (6)$$

will satisfy $\mathbb{E}[\mathcal{D}(\alpha^{(H)}) - \mathcal{D}(\alpha^*)] \leq \epsilon_{\mathcal{D}}$.

Theorem 2. If the loss functions ℓ_t are L -Lipschitz, then there exists a constant σ , defined in (24), such that for any given $\epsilon_{\mathcal{D}} > 0$, if we choose

$$H \geq H_0 + \left\lceil \frac{2}{(1 - \bar{\Theta})} \max \left(1, \frac{2L^2\sigma\sigma'}{n^2\epsilon_{\mathcal{D}}} \right) \right\rceil, \quad (7)$$

$$\text{with } H_0 \geq \left\lceil h_0 + \frac{16L^2\sigma\sigma'}{(1 - \bar{\Theta})n^2\epsilon_{\mathcal{D}}} \right\rceil, h_0 = \left\lceil 1 + \frac{1}{(1 - \bar{\Theta})} \log \left(\frac{2n^2(D(\alpha^*) - D(\alpha^0))}{4L^2\sigma\sigma'} \right) \right\rceil_+,$$

then $\bar{\alpha} := \frac{1}{H - H_0} \sum_{h=H_0+1}^H \alpha^{(h)}$ will satisfy $\mathbb{E}[\mathcal{D}(\bar{\alpha}) - \mathcal{D}(\alpha^*)] \leq \epsilon_{\mathcal{D}}$.

Simulations

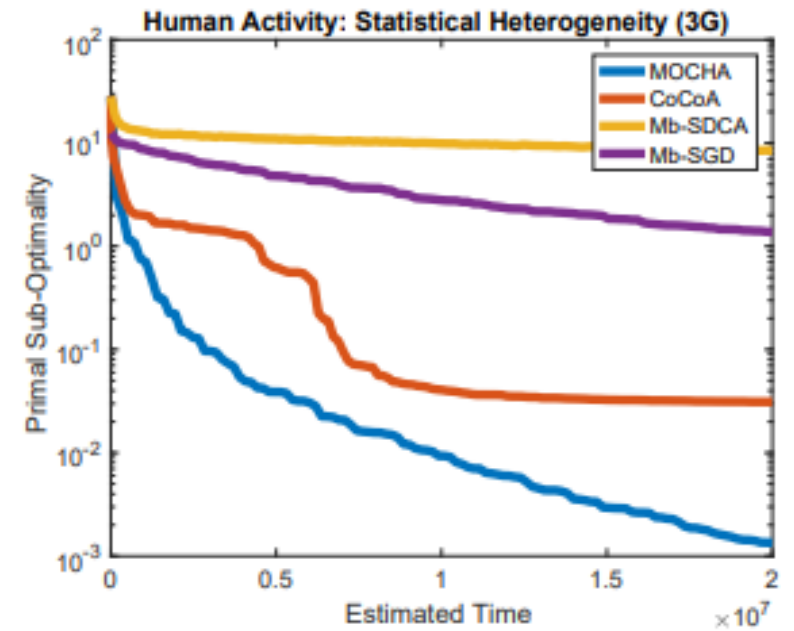
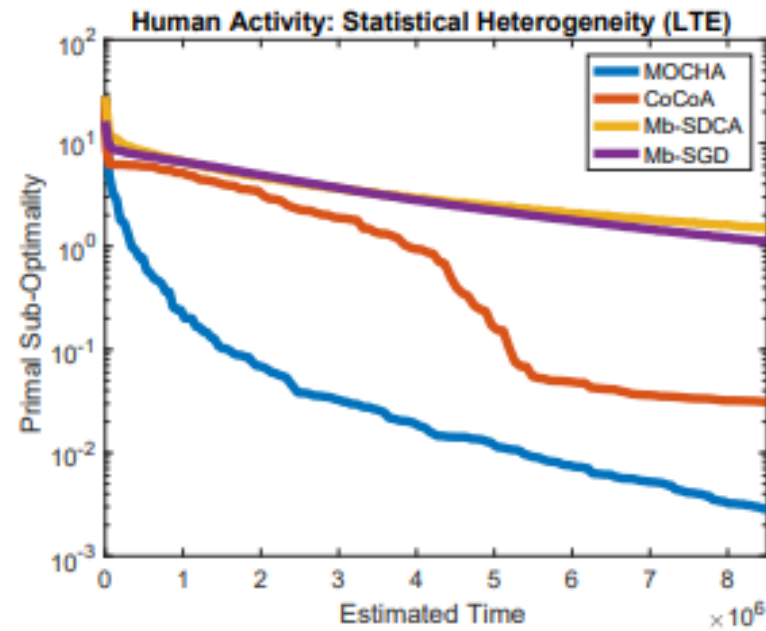
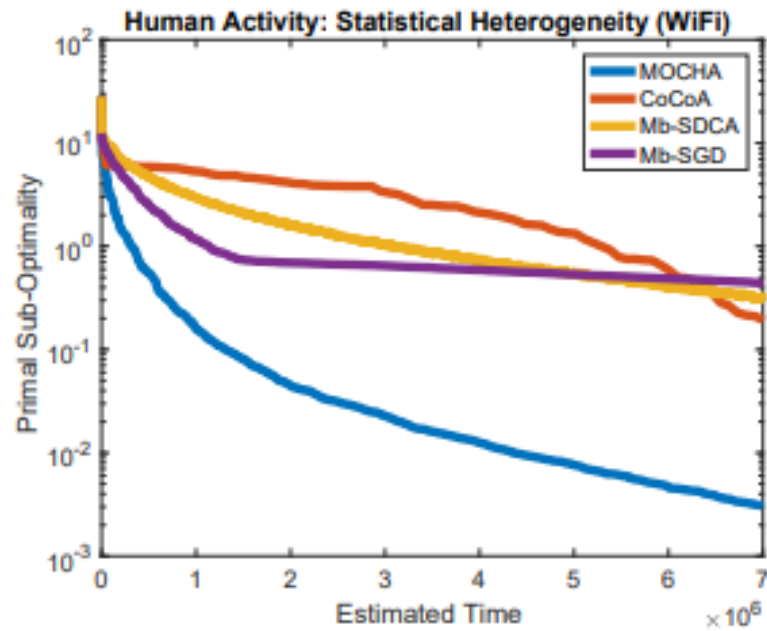
- Real-world datasets: Google Glass, Human Activity Recognition and Vehicle Sensor
- Multi-Task Learning

Table 1: Average prediction error: Means and standard errors over 10 random shuffles.

Model	Human Activity	Google Glass	Vehicle Sensor
Global	2.23 (0.30)	5.34 (0.26)	13.4 (0.26)
Local	1.34 (0.21)	4.92 (0.26)	7.81 (0.13)
MTL	0.46 (0.11)	2.02 (0.15)	6.59 (0.21)

Simulations

- Straggler Avoidance



Simulations

- Heterogeneity

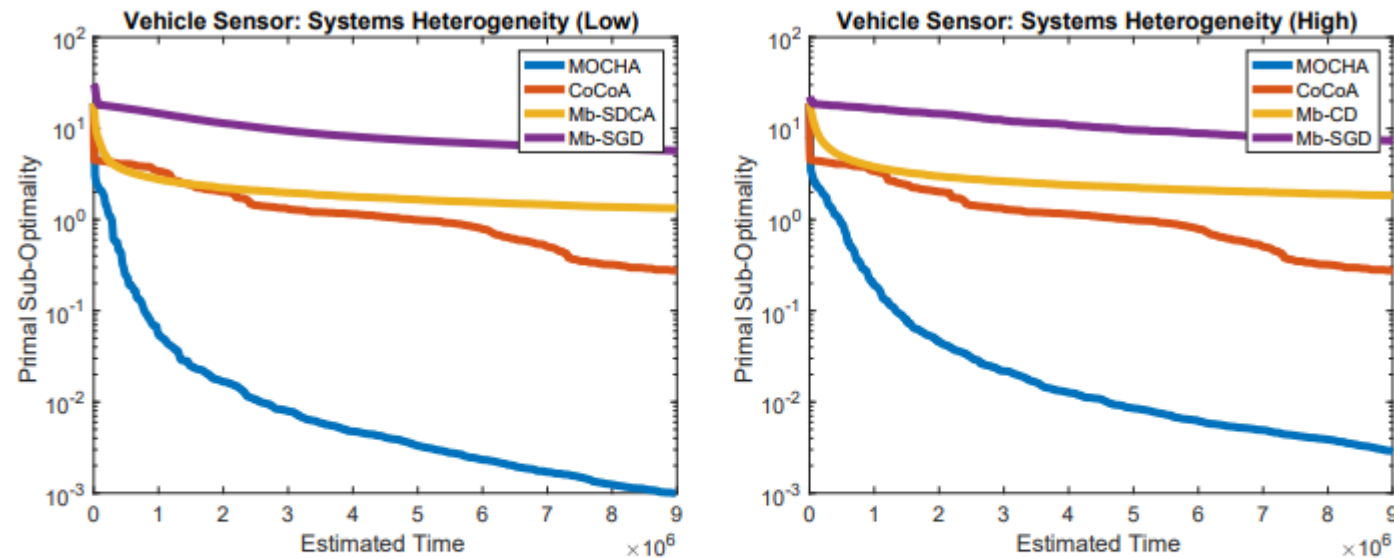


Figure 2: MOCHA can handle variability from systems heterogeneity.

Simulations

- Tolerance to Dropped Nodes
- p_t^h is the probability that nodes drop at each iteration

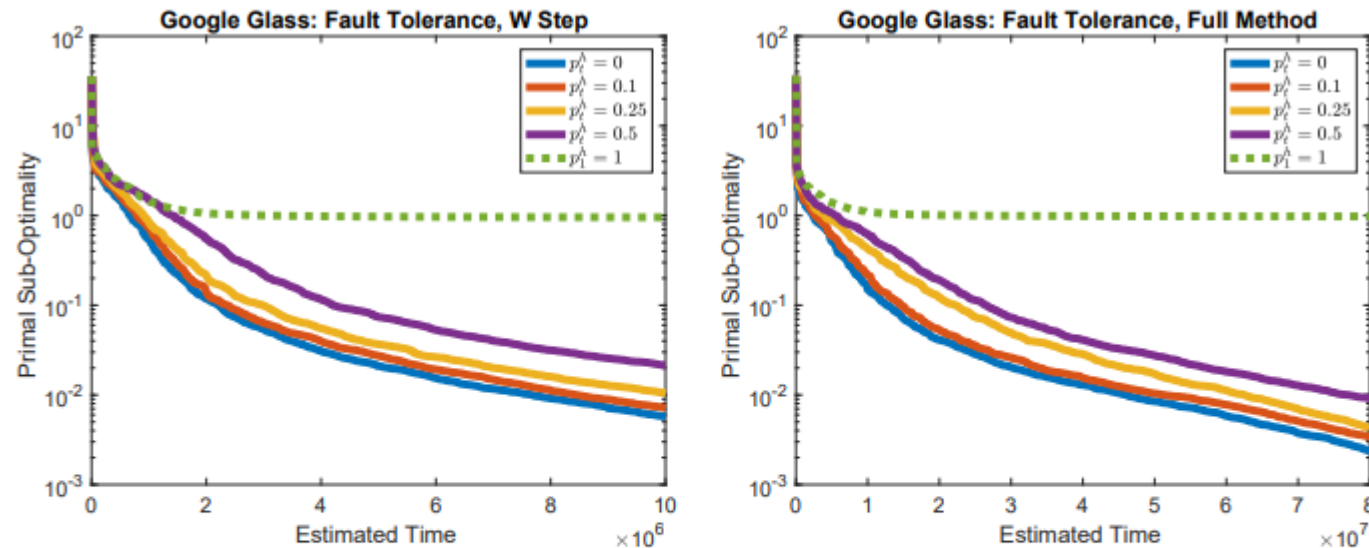


Figure 3: The performance of MOCHA is robust to nodes periodically dropping out (fault tolerance).

Discussion 1

- Q: Why do we solve the dual problem instead of the primal problem in the algorithm?
- The regularization item $R(W, \Omega)$ contains weights and model relations globally, so we cannot upgrade w_t separately in each node.
- For dual problem, we can do approximation to R^* and do separate computation

$$\min_{\Delta \alpha_t} \mathcal{G}_t^{\sigma'}(\Delta \alpha_t; \mathbf{v}_t, \alpha_t) := \sum_{i=1}^{n_t} \ell_t^*(-\alpha_t^i - \Delta \alpha_t^i) + \langle \mathbf{w}_t(\alpha), \mathbf{X}_t \Delta \alpha_t \rangle + \frac{\sigma'}{2} \|\mathbf{X}_t \Delta \alpha_t\|_{\mathbf{M}_t}^2 + c(\alpha)$$

Discussion 2

- Q: The fitted models in MOCCA are linear models (or GLM). If we apply MOCCA to more complex models, can we still prove the convergence of it? Will it still perform well in federated datasets?
- I'll say yes. But there might be additional restrictions on convergence. And if we use more complex models, it will be more difficult to compute the dual problem, the regularizer and its approximation, which increases computation costs

Discussion 3

- Q: Can we apply Mocha to nonconvex function?
- I'll say no. Firstly, if the objective function is nonconvex, we can not guarantee that the dual optimal value equals the primal optimal value. And it is possible that this algorithms can not converge or can not converge to an optimal point.