Federated Multi-Task Learning

Virginia Smith
Stanford
smithv@stanford.edu

Chao-Kai Chiang*
USC
chaokaic@usc.edu

Maziar Sanjabi*
USC
maziarsanjabi@gmail.com

Ameet Talwalkar
CMU
talwalkar@cmu.edu

Abstract

Federated learning poses new statistical and systems challenges in training machine learning models over distributed networks of devices. In this work, we show that multi-task learning is naturally suited to handle the statistical challenges of this setting, and propose a novel systems-aware optimization method, MOCHA, that is robust to practical systems issues. Our method and theory for the first time consider issues of high communication cost, stragglers, and fault tolerance for distributed multi-task learning. The resulting method achieves significant speedups compared to alternatives in the federated setting, as we demonstrate through simulations on real-world federated datasets.

1 Introduction

Mobile phones, wearable devices, and smart homes are just a few of the modern distributed networks generating massive amounts of data each day. Due to the growing storage and computational power of devices in these networks, it is increasingly attractive to store data locally and push more network computation to the edge. The nascent field of *federated learning* explores *training* statistical models directly on devices [37]. Examples of potential applications include: learning sentiment, semantic location, or activities of mobile phone users; predicting health events like low blood sugar or heart attack risk from wearable devices; or detecting burglaries within smart homes [3, 39, 42]. Following [25, 36, 26], we summarize the unique challenges of federated learning below.

1. **Statistical Challenges**: The aim in federated learning is to fit a model to data, $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, generated by m distributed nodes. Each node, $t \in [m]$, collects data in a *non-IID* manner across the network, with data on each node being generated by a distinct distribution $\mathbf{X}_t \sim P_t$. The number of data points on each node, n_t, may also vary significantly, and there may be an underlying structure present that captures the relationship amongst nodes and their associated distributions.
2. **Systems Challenges**: There are typically a large number of nodes, m, in the network, and communication is often a significant bottleneck. Additionally, the storage, computational, and communication capacities of each node may differ due to variability in hardware (CPU, memory), network connection (3G, 4G, WiFi), and power (battery level). These systems challenges, compounded with unbalanced data and statistical heterogeneity, make issues such as stragglers and fault tolerance significantly more prevalent than in typical data center environments.

In this work, we propose a modeling approach that differs significantly from prior work on federated learning, where the aim thus far has been to train a single global model across the network [25, 36, 26]. Instead, we address statistical challenges in the federated setting by learning separate models for each node, $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$. This can be naturally captured through a *multi-task learning (MTL)* framework, where the goal is to consider fitting separate but related models simultaneously [14, 2, 58, 28]. Unfortunately, current multi-task learning methods are not suited to handle the systems challenges that arise in federated learning, including high communication cost, stragglers, and fault tolerance. Addressing these challenges is therefore a key component of our work.

*Authors contributed equally.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1.1 Contributions

We make the following contributions. First, we show that MTL is a natural choice to handle statistical challenges in the federated setting. Second, we develop a novel method, MOCHA, to solve a general MTL problem. Our method generalizes the distributed optimization method COCOA [22, 31] in order to address systems challenges associated with network size and node heterogeneity. Third, we provide convergence guarantees for MOCHA that carefully consider these unique systems challenges and provide insight into practical performance. Finally, we demonstrate the superior empirical performance of MOCHA with a new benchmarking suite of federated datasets.

2 Related Work

Learning Beyond the Data Center. Computing SQL-like queries across distributed, low-powered nodes is a decades-long area of research that has been explored under the purview of query processing in sensor networks, computing at the edge, and fog computing [32, 12, 33, 8, 18, 15]. Recent works have also considered training machine learning models centrally but serving and storing them locally, e.g., this is a common approach in mobile user modeling and personalization [27, 43, 44]. However, as the computational power of the nodes within distributed networks grows, it is possible to do even more work locally, which has led to recent interest in federated learning.[2] In contrast to our proposed approach, existing federated learning approaches [25, 36, 26, 37] aim to learn a single global model across the data.[3] This limits their ability to deal with non-IID data and structure amongst the nodes. These works also come without convergence guarantees, and have not addressed practical issues of stragglers or fault tolerance, which are important characteristics of the federated setting. The work proposed here is, to the best of our knowledge, the first federated learning framework to consider these challenges, theoretically and in practice.

Multi-Task Learning. In multi-task learning, the goal is to learn models for multiple related tasks simultaneously. While the MTL literature is extensive, most MTL modeling approaches can be broadly categorized into two groups based on how they capture relationships amongst tasks. The first (e.g., [14, 4, 11, 24]) assumes that a clustered, sparse, or low-rank structure between the tasks is known *a priori*. A second group instead assumes that the task relationships are not known beforehand and can be learned directly from the data (e.g., [21, 58, 16]). In this work, we focus our attention on this latter group, as task relationships may not be known beforehand in real-world settings. In comparison to learning a single global model, these MTL approaches can directly capture relationships amongst non-IID and unbalanced data, which makes them particularly well-suited for the statistical challenges of federated learning. We demonstrate this empirically on real-world federated datasets in Section 5. However, although MTL is a natural modeling choice to address the statistical challenges of federated learning, currently proposed methods for distributed MTL (discussed below) do not adequately address the systems challenges associated with federated learning.

Distributed Multi-Task Learning. Distributed multi-task learning is a relatively new field, in which the aim is to solve an MTL problem when data for each task is distributed over a network. While several recent works [1, 35, 54, 55] have considered the issue of distributed MTL training, the proposed methods do not allow for flexibility of communication versus computation. As a result, they are unable to efficiently handle concerns of fault tolerance and stragglers, the latter of which stems from both data and system heterogeneity. The works of [23] and [7] allow for asynchronous updates to help mitigate stragglers, but do not address fault tolerance. Moreover, [23] provides no convergence guarantees, and the convergence of [7] relies on a bounded delay assumption that is impractical for the federated setting, where delays may be significant and devices may drop out completely. Finally, [30] proposes a method and setup leveraging the distributed framework COCOA [22, 31], which we show in Section 4 to be a special case of the more general approach in this work. However, the authors in [30] do not explore the federated setting, and their assumption that the same amount of work is done locally on each node is prohibitive in federated settings, where unbalance is common due to data and system variability.

[2] The term *on-device learning* has been used to describe both the task of model training and of model serving. Due to the ambiguity of this phrase, we exclusively use the term federated learning.

[3] While not the focus of our work, we note privacy is an important concern in the federated setting, and that the privacy benefits associated with global federated learning (as discussed in [36]) also apply to our approach.

3 Federated Multi-Task Learning

In federated learning, the aim is to learn a model over data that resides on, and has been generated by, m distributed nodes. As a running example, consider learning the activities of mobile phone users in a cell network based on their individual sensor, text, or image data. Each node (phone), $t \in [m]$, may generate data via a distinct distribution, and so it is natural to fit separate models, $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$, to the distributed data—one for each local dataset. However, structure between models frequently exists (e.g., people may behave similarly when using their phones), and modeling these relationships via *multi-task learning* is a natural strategy to improve performance and boost the effective sample size for each node [10, 2, 5]. In this section, we suggest a general MTL framework for the federated setting, and propose a novel method, MOCHA, to handle the systems challenges of federated MTL.

3.1 General Multi-Task Learning Setup

Given data $\mathbf{X}_t \in \mathbb{R}^{d \times n_t}$ from m nodes, multi-task learning fits separate weight vectors $\mathbf{w}_t \in \mathbb{R}^d$ to the data for each task (node) through arbitrary convex loss functions ℓ_t (e.g., the hinge loss for SVM models). Many MTL problems can be captured via the following general formulation:

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

where $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_m] \in \mathbb{R}^{d \times m}$ is a matrix whose t-th column is the weight vector for the t-th task. The matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times m}$ models relationships amongst tasks, and is either known a priori or estimated while simultaneously learning task models. MTL problems differ based on their assumptions on $\mathcal{R}$, which takes $\mathbf{\Omega}$ as input and promotes some suitable structure amongst the tasks.

As an example, several popular MTL approaches assume that tasks form clusters based on whether or not they are related [14, 21, 58, 59]. This can be expressed via the following bi-convex formulation:

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

with constants $\lambda_1, \lambda_2 > 0$, and where the second term performs L_2 regularization on each local model. We use a similar formulation with variable clusters (12) in our experiments in Section 5, and provide details on other common classes of MTL models that can be formulated via (1) in Appendix B.

3.2 MOCHA: A Framework for Federated Multi-Task Learning

In the federated setting, the aim is to train statistical models directly on the edge, and thus we solve (1) while assuming that the data $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ is distributed across m nodes or devices. Before proposing our federated method for solving (1), we make the following observations:

- ***Observation 1****: In general,* (1) *is not jointly convex in* $\mathbf{W}$ *and* $\mathbf{\Omega}$*, and even in the cases where* (1) *is convex, solving for* $\mathbf{W}$ *and* $\mathbf{\Omega}$ *simultaneously can be difficult [5].*
- ***Observation 2****: When fixing* $\mathbf{\Omega}$*, updating* $\mathbf{W}$ *depends on both the data* $\mathbf{X}$*, which is distributed across the nodes, and the structure* $\mathbf{\Omega}$*, which is known centrally.*
- ***Observation 3****: When fixing* $\mathbf{W}$*, optimizing for* $\mathbf{\Omega}$ *only depends on* $\mathbf{W}$ *and not on the data* $\mathbf{X}$*.*

Based on these observations, it is natural to propose an alternating optimization approach to solve problem (1), in which at each iteration we fix either $\mathbf{W}$ or $\mathbf{\Omega}$ and optimize over the other, alternating until convergence is reached. Note that solving for $\mathbf{\Omega}$ is not dependent on the data and therefore can be computed centrally; as such, we defer to prior work for this step [59, 21, 58, 16]. In Appendix B, we discuss updates to Ω for several common MTL models.

In this work, we focus on developing an efficient distributed optimization method for the $\mathbf{W}$ step. In traditional data center environments, the task of distributed training is a well-studied problem, and various communication-efficient frameworks have been recently proposed, including the state-of-the-art primal-dual COCOA framework [22, 31]. Although COCOA can be extended directly to update $\mathbf{W}$ in a distributed fashion across the nodes, it cannot handle the unique systems challenges of the federated environment, such as stragglers and fault tolerance, as discussed in Section 3.4. To this end, we extend COCOA and propose a new method, MOCHA, for federated multi-task learning. Our method is given in Algorithm 1 and described in detail in Sections 3.3 and 3.4.

Algorithm 1 MOCHA: Federated Multi-Task Learning Framework

Input: Data $\mathbf{X}_t$ from $t = 1, \ldots, m$ tasks, stored on one of m nodes, and initial matrix $\mathbf{\Omega}_0$
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^b$
for iterations $i = 0, 1, \ldots$ **do**
  Set subproblem parameter σ' and number of federated iterations, H_i
  for iterations $h = 0, 1, \cdots, H_i$ **do**
    for tasks $t \in \{1, 2, \ldots, m\}$ **in parallel over** m **nodes do**
      call local solver, returning θ_t^h-approximate solution $\Delta\boldsymbol{\alpha}_t$ of the local subproblem (4)
      update local variables $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_t + \Delta\boldsymbol{\alpha}_t$
      return updates $\Delta\mathbf{v}_t := \mathbf{X}_t\Delta\boldsymbol{\alpha}_t$
    reduce: $\mathbf{v}_t \leftarrow \mathbf{v}_t + \Delta\mathbf{v}_t$
  Update $\mathbf{\Omega}$ centrally based on $\mathbf{w}(\boldsymbol{\alpha})$ for latest $\boldsymbol{\alpha}$
Central node computes $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$ based on the lastest $\boldsymbol{\alpha}$
return: $\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_m]$

3.3 Federated Update of W

To update $\mathbf{W}$ in the federated setting, we begin by extending works on distributed primal-dual optimization [22, 31, 30] to apply to the generalized multi-task framework (1). This involves deriving the appropriate dual formulation, subproblems, and problem parameters, as we detail below.

Dual problem. Considering the dual formulation of (1) will allow us to better separate the global problem into distributed subproblems for federated computation across the nodes. Let $n := \sum_{t=1}^m n_t$ and $\mathbf{X} := \text{Diag}(\mathbf{X}_1, \cdots, \mathbf{X}_m) \in \mathbb{R}^{md \times n}$. With $\mathbf{\Omega}$ fixed, the dual of problem (1), defined with respect to dual variables $\boldsymbol{\alpha} \in \mathbb{R}^n$, is given by:

$$\min_{\boldsymbol{\alpha}} \left\{ \mathcal{D}(\boldsymbol{\alpha}) := \sum_{t=1}^{m} \sum_{i=1}^{n_t} \ell_t^*(-\boldsymbol{\alpha}_t^i) + \mathcal{R}^*(\mathbf{X}\boldsymbol{\alpha}) \right\}, \tag{3}$$

where ℓ_t^* and $\mathcal{R}^*$ are the conjugate dual functions of ℓ_t and $\mathcal{R}$, respectively, and $\boldsymbol{\alpha}_t^i$ is the dual variable for the data point $(\mathbf{x}_t^i, y_t^i)$. Note that $\mathcal{R}^*$ depends on $\mathbf{\Omega}$, but for the sake of simplicity, we have removed this in our notation. To derive distributed subproblems from this global dual, we make an assumption described below on the regularizer $\mathcal{R}$.

Assumption 1. *Given $\mathbf{\Omega}$, we assume that there exists a symmetric positive definite matrix $\mathbf{M} \in \mathbb{R}^{md \times md}$, depending on $\mathbf{\Omega}$, for which the function $\mathcal{R}$ is strongly convex with respect to $\mathbf{M}^{-1}$. Note that this corresponds to assuming that $\mathcal{R}^*$ will be smooth with respect to matrix $\mathbf{M}$.*

Remark 1. *We can reformulate the MTL regularizer in the form of $\bar{\mathcal{R}}(\mathbf{w}, \bar{\mathbf{\Omega}}) = \mathcal{R}(\mathbf{W}, \mathbf{\Omega})$, where $\mathbf{w} \in \mathbb{R}^{md}$ is a vector containing the columns of $\mathbf{W}$ and $\bar{\mathbf{\Omega}} := \mathbf{\Omega} \otimes \mathbf{I}_{d\times d} \in \mathbb{R}^{md \times md}$. For example, we can rewrite the regularizer in* (2) *as $\bar{\mathcal{R}}(\mathbf{w}, \bar{\mathbf{\Omega}}) = tr\big(\mathbf{w}^T(\lambda_1\bar{\mathbf{\Omega}} + \lambda_2\mathbf{I})\mathbf{w}\big)$. Writing the regularizer in this form, it is clear that it is strongly convex with respect to matrix $\mathbf{M}^{-1} = \lambda_1\bar{\mathbf{\Omega}} + \lambda_2\mathbf{I}$.*

Data-local quadratic subproblems. To solve (1) across distributed nodes, we define the following data-local subproblems, which are formed via a careful quadratic approximation of the dual problem (3) to separate computation across the nodes. These subproblems find updates $\Delta\boldsymbol{\alpha}_t \in \mathbb{R}^{n_t}$ to the dual variables in $\boldsymbol{\alpha}$ corresponding to a single node t, and only require accessing data which is available locally, i.e., $\mathbf{X}_t$ for node t. The t-th subproblem is given by:

$$\min_{\Delta\boldsymbol{\alpha}_t} \mathcal{G}_t^{\sigma'}(\Delta\boldsymbol{\alpha}_t; \mathbf{v}_t, \boldsymbol{\alpha}_t) := \sum_{i=1}^{n_t} \ell_t^*(-\boldsymbol{\alpha}_t^i - \Delta\boldsymbol{\alpha}_t^i) + \langle \mathbf{w}_t(\boldsymbol{\alpha}), \mathbf{X}_t\Delta\boldsymbol{\alpha}_t \rangle + \frac{\sigma'}{2} \|\mathbf{X}_t\Delta\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2 + c(\boldsymbol{\alpha}), \tag{4}$$

where $c(\boldsymbol{\alpha}) := \frac{1}{m}\mathcal{R}^*(\mathbf{X}\boldsymbol{\alpha})$, and $\mathbf{M}_t \in \mathbb{R}^{d\times d}$ is the t-th diagonal block of the symmetric positive definite matrix $\mathbf{M}$. Given dual variables $\boldsymbol{\alpha}$, corresponding primal variables can be found via $\mathbf{w}(\boldsymbol{\alpha}) = \nabla\mathcal{R}^*(\mathbf{X}\boldsymbol{\alpha})$, where $\mathbf{w}_t(\boldsymbol{\alpha})$ is the t-th block in the vector $\mathbf{w}(\boldsymbol{\alpha})$. Note that computing $\mathbf{w}(\boldsymbol{\alpha})$ requires the vector $\mathbf{v} = \mathbf{X}\boldsymbol{\alpha}$. The t-th block of $\mathbf{v}$, $\mathbf{v}_t \in \mathbb{R}^d$, is the only information that must be communicated between nodes at each iteration. Finally, $\sigma' > 0$ measures the difficulty of the data partitioning, and helps to relate progress made to the subproblems to the global dual problem. It can be easily selected based on $\mathbf{M}$ for many applications of interest; we provide details in Lemma 9 of the Appendix.

3.4 Practical Considerations

During MOCHA's federated update of $\mathbf{W}$, the central node requires a response from all workers before performing a synchronous update. In the federated setting, a naive execution of this communication protocol could introduce dramatic straggler effects due to node heterogeneity. To avoid stragglers, MOCHA provides the t-th node with the flexibility to *approximately solve* its subproblem $\mathcal{G}_t^{\sigma'}(\cdot)$, where the quality of the approximation is controled by a per-node parameter θ_t^h. The following factors determine the quality of the t-th node's solution to its subproblem:

1. **Statistical challenges**, such as the size of $\mathbf{X}_t$ and the intrinsic difficulty of subproblem $\mathcal{G}_t^{\sigma'}(\cdot)$.
2. **Systems challenges**, such as the node's storage, computational, and communication capacities due to hardware (CPU, memory), network connection (3G, 4G, WiFi), and power (battery level).
3. A **global clock cycle** imposed by the central node specifying a deadline for receiving updates.

We define θ_t^h as a function of these factors, and assume that each node has a controller that may derive θ_t^h from the current clock cycle and statistical/systems setting. θ_t^h ranges from zero to one, where $\theta_t^h = 0$ indicates an exact solution to $\mathcal{G}_t^{\sigma'}(\cdot)$ and $\theta_t^h = 1$ indicates that node t made no progress during iteration h (which we refer to as a *dropped node*). For instance, a node may 'drop' if it runs out of battery, or if its network bandwidth deteriorates during iteration h and it is thus unable to return its update within the current clock cycle. A formal definition of θ_t^h is provided in (5) of Section 4.

MOCHA mitigates stragglers by enabling the t-th node to define its own θ_t^h. On every iteration h, the local updates that a node performs and sends in a clock cycle will yield a specific value for θ_t^h. As discussed in Section 4, MOCHA is additionally robust to a small fraction of nodes periodically dropping and performing no local updates (i.e., $\theta_t^h := 1$) under suitable conditions, as defined in Assumption 2. In contrast, prior work of COCOA may suffer from severe straggler effects in federated settings, as it requires a *fixed $\theta_t^h = \theta$ across all nodes and all iterations* while still maintaining synchronous updates, and it does not allow for the case of dropped nodes ($\theta := 1$).

Finally, we note that asynchronous updating schemes are an alternative approach to mitigate stragglers. We do not consider these approaches in this work, in part due to the fact that the bounded-delay assumptions associated with most asynchronous schemes limits fault tolerance. However, it would be interesting to further explore the differences and connections between asynchronous methods and approximation-based, synchronous methods like MOCHA in future work.

4 Convergence Analysis

MOCHA is based on a bi-convex alternating approach, which is guaranteed to converge [17, 45] to a stationary solution of problem (1). In the case where this problem is jointly convex with respect to $\mathbf{W}$ and $\mathbf{\Omega}$, such a solution is also optimal. In the rest of this section, we therefore focus on the convergence of solving the $\mathbf{W}$ update of MOCHA in the federated setting. Following the discussion in Section 3.4, we first introduce the following per-node, per-round approximation parameter.

Definition 1 (Per-Node-Per-Iteration-Approximation Parameter)**.** At each iteration h, we define the accuracy level of the solution calculated by node t to its subproblem (4) as:

$$\theta_t^h := \frac{\mathcal{G}_t^{\sigma'}(\Delta\boldsymbol{\alpha}_t^{(h)}; \mathbf{v}^{(h)}, \boldsymbol{\alpha}_t^{(h)}) - \mathcal{G}_t^{\sigma'}(\Delta\boldsymbol{\alpha}_t^{\star}; \mathbf{v}^{(h)}, \boldsymbol{\alpha}_t^{(h)})}{\mathcal{G}_t^{\sigma'}(\mathbf{0}; \mathbf{v}^{(h)}, \boldsymbol{\alpha}_t^{(h)}) - \mathcal{G}_t^{\sigma'}(\Delta\boldsymbol{\alpha}_t^{\star}; \mathbf{v}^{(h)}, \boldsymbol{\alpha}_t^{(h)})}\,, \tag{5}$$

where $\Delta\boldsymbol{\alpha}_t^{\star}$ is the minimizer of subproblem $\mathcal{G}_t^{\sigma'}(\cdot\,; \mathbf{v}^{(h)}, \boldsymbol{\alpha}_t^{(h)})$. We allow this value to vary between $[0, 1]$, with $\theta_t^h := 1$ meaning that no updates to subproblem $\mathcal{G}_t^{\sigma'}$ are made by node t at iteration h.

While the flexible per-node, per-iteration approximation parameter θ_t^h in (5) allows the consideration of stragglers and fault tolerance, these additional degrees of freedom also pose new challenges in providing convergence guarantees for MOCHA. We introduce the following assumption on θ_t^h to provide our convergence guarantees.

Assumption 2. *Let $\mathcal{H}_h := (\boldsymbol{\alpha}^{(h)}, \boldsymbol{\alpha}^{(h-1)}, \cdots, \boldsymbol{\alpha}^{(1)})$ be the dual vector history until the beginning of iteration h, and define $\Theta_t^h := \mathbb{E}[\theta_t^h | \mathcal{H}_h]$. For all tasks t and all iterations h, we assume $p_t^h := \mathbb{P}[\theta_t^h = 1] \leq p_{\max} < 1$ and $\hat{\Theta}_t^h := \mathbb{E}[\theta_t^h | \mathcal{H}_h, \theta_t^h < 1] \leq \Theta_{\max} < 1$.*

This assumption states that at each iteration, the *probability* of a node sending a result is non-zero, and that the quality of the returned result is, on average, better than the previous iterate. Compared to [49, 30] which assumes $\theta_t^h = \theta < 1$, our assumption is significantly less restrictive and better models the federated setting, where nodes are unreliable and may periodically drop out.

Using Assumption 2, we derive the following theorem, which characterizes the convergence of the federated update of MOCHA in finite horizon when the losses ℓ_t in (1) are smooth.

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*

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

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

Here, $\bar{\Theta} := p_{\max} + (1 - p_{\max})\Theta_{\max} < 1$. While Theorem 1 is concerned with finite horizon convergence, it is possible to get asymptotic convergence results, i.e., $H \to \infty$, with milder assumptions on the stragglers; see Corollary 8 in the Appendix for details.

When the loss functions are non-smooth, e.g., the hinge loss for SVM models, we provide the following sub-linear convergence for L-Lipschitz losses.

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*

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

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

then $\bar{\boldsymbol{\alpha}} := \frac{1}{H-H0} \sum_{h=H_0+1}^{H} \boldsymbol{\alpha}^{(h)}$ will satisfy $\mathbb{E}[\mathcal{D}(\bar{\boldsymbol{\alpha}}) - \mathcal{D}(\boldsymbol{\alpha}^\star)] \leq \epsilon_{\mathcal{D}}$.

These theorems guarantee that MOCHA will converge in the federated setting, under mild assumptions on stragglers and capabilities of the nodes. While these results consider convergence in terms of the dual, we show that they hold analogously for the duality gap. We provide all proofs in Appendix C.

Remark 2. *Following from the discussion in Section 3.4, our method and theory generalize the results in [22, 31]. In the limiting case that all θ_t^h are identical, our results extend the results of* COCOA *to the multi-task framework described in* (1)*.*

Remark 3. *Note that the methods in [22, 31] have an aggregation parameter $\gamma \in (0, 1]$. Though we prove our results for a general γ, we simplify the method and results here by setting $\gamma := 1$, which has been shown to have the best performance, both theoretically and empirically [31].*

5 Simulations

In this section we validate the empirical performance of MOCHA. First, we introduce a benchmarking suite of real-world federated datasets and show that multi-task learning is well-suited to handle the statistical challenges of the federated setting. Next, we demonstrate MOCHA's ability to handle stragglers, both from statistical and systems heterogeneity. Finally, we explore the performance of MOCHA when devices periodically drop out. Our code is available at: `github.com/gingsmith/fmtl`.

5.1 Federated Datasets

In our simulations, we use several real-world datasets that have been generated in federated settings. We provide additional details in the Appendix, including information about data sizes, n_t.

- **Google Glass (GLEAM)**[4]: This dataset consists of two hours of high resolution sensor data collected from 38 participants wearing Google Glass for the purpose of activity recognition. Following [41], we featurize the raw accelerometer, gyroscope, and magnetometer data into 180 statistical, spectral, and temporal features. We model each participant as a separate task, and predict between eating and other activities (e.g., walking, talking, drinking).

[4] `http://www.skleinberg.org/data/GLEAM.tar.gz`

- **Human Activity Recognition**[5]: Mobile phone accelerometer and gyroscope data collected from 30 individuals, performing one of six activities. We use the provided 561-length feature vectors of time and frequency domain variables generated for each instance [3]. We model each individual as a separate task and predict between sitting and other activities (e.g., walking, lying down).
- **Land Mine**[6]: Radar image data collected from 29 land mine fields. Each instance consists of nine features extracted from the images [56]. We model each field as a task, and predict whether or not landmines are present in each field. Notably, the data is collected from two different terrains—highly foliated and desert regions—and the tasks therefore naturally form two clusters.
- **Vehicle Sensor**[7]: Acoustic, seismic, and infrared sensor data collected from a distributed network of 23 sensors, deployed with the aim of classifying vehicles driving by a segment of road [13]. Each instance is described by 50 acoustic and 50 seismic features. We model each sensor as a separate task and predict between AAV-type and DW-type vehicles.

5.2 Multi-Task Learning for the Federated Setting

We demonstrate the benefits of multi-task learning for the federated setting by comparing the error rates of a multi-task model to that of a fully local model (i.e., learning a model for each task separately) and a fully global model (i.e., combining the data from all tasks and learning one single model). Work on federated learning thus far has been limited to the study of fully global models [25, 36, 26].

We use a cluster-regularized multi-task model [59, 21], as described in Section 3.1. For each dataset from Section 5.1, we randomly split the data into 75% training and 25% testing, and learn multi-task, local, and global support vector machine models, selecting the best regularization parameter, $\lambda \in \{$1e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10$\}$, for each model using 5-fold cross-validation. We repeat this process 10 times and report the average prediction error across tasks, averaged across these 10 trials.

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

Model	Human Activity	Google Glass	Land Mine	Vehicle Sensor
Global	2.23 (0.30)	5.34 (0.26)	27.72 (1.08)	13.4 (0.26)
Local	1.34 (0.21)	4.92 (0.26)	23.43 (0.77)	7.81 (0.13)
MTL	**0.46 (0.11)**	**2.02 (0.15)**	**20.09 (1.04)**	**6.59 (0.21)**

In Table 1, we see that for each dataset, multi-task learning significantly outperforms the other models in terms of achieving the lowest average error across tasks. The global model, as proposed in [25, 36, 26] performs the worst, particularly for the Human Activity and Vehicle Sensor datasets. Although the datasets are already somewhat unbalanced, we note that a global modeling approach may benefit tasks with a very small number of instances, as information can be shared across tasks. For this reason, we additionally explore the performance of global, local, and multi-task modeling for highly skewed data in Table 4 of the Appendix. Although the performance of the global model improves slightly relative to local modeling in this setting, the global model still performs the worst for the majority of the datasets, and MTL still significantly outperforms both global and local approaches.

5.3 Straggler Avoidance

Two challenges that are prevalent in federated learning are stragglers and high communication. Stragglers can occur when a subset of the devices take much longer than others to perform local updates, which can be caused either by statistical or systems heterogeneity. Communication can also exacerbate poor performance, as it can be slower than computation by many orders of magnitude in typical cellular or wireless networks [52, 20, 48, 9, 38]. In our experiments below, we simulate the time needed to run each method by tracking the operations and communication complexities, and scaling the communication cost relative to computation by one, two, or three orders of magnitude, respectively. These numbers correspond roughly to the clock rate vs. network bandwidth/latency (see, e.g., [52]) for modern cellular and wireless networks. Details are provided in Appendix E.

[5] https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[6] http://www.ee.duke.edu/~lcarin/LandmineData.zip

[7] http://www.ecs.umass.edu/~mduarte/Software.html

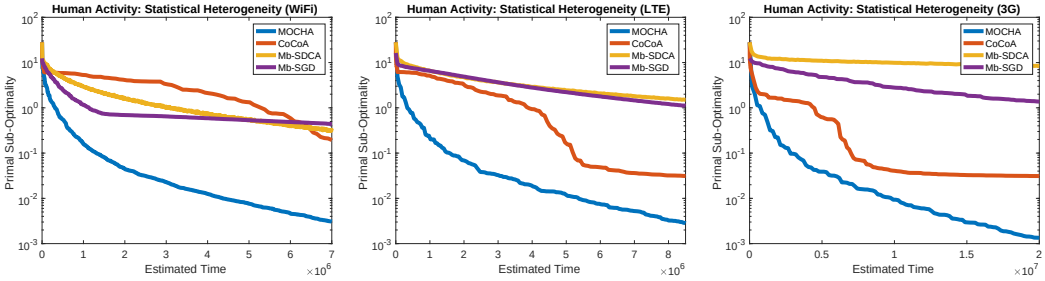

Figure 1: The performance of MOCHA compared to other distributed methods for the $\mathbf{W}$ update of (1). While increasing communication tends to *decrease* the performance of the mini-batch methods, MOCHA performs well in high communication settings. In all settings, MOCHA with varied approximation values, Θ_t^h, performs better than without (i.e., naively generalizing COCOA), as it avoids stragglers from statistical heterogeneity.

Statistical Heterogeneity. We explore the effect of statistical heterogeneity on stragglers for various methods and communication regimes (3G, LTE, WiFi). For a fixed communication network, we compare MOCHA to COCOA, which has a single θ parameter, and to mini-batch stochastic gradient descent (Mb-SGD) and mini-batch stochastic dual coordinate ascent (Mb-SDCA), which have limited communication flexibility depending on the batch size. We tune all compared methods for best performance, as we detail in Appendix E. In Figure 1, we see that while the performance degrades for mini-batch methods in high communication regimes, MOCHA and COCOA are robust to high communication. However, COCOA is significantly affected by stragglers—because θ is fixed across nodes and rounds, difficult subproblems adversely impact convergence. In contrast, MOCHA performs well regardless of communication cost and is robust to statistical heterogeneity.

Systems Heterogeneity. MOCHA is also equipped to handle heterogeneity from changing systems environments, such as battery power, memory, or network connection, as we show in Figure 2. In particular, we simulate systems heterogeneity by randomly choosing the number of local iterations for MOCHA or the mini-batch size for mini-batch methods, between 10% and 100% of the minimum number of local data points for high variability environments, to between 90% and 100% for low variability (see Appendix E for full details). We do not vary the performance of COCOA, as the impact from statistical heterogeneity alone significantly reduces performance. However, adding systems heterogeneity would reduce performance even further, as the maximum θ value across all nodes would only increase if additional systems challenges were introduced.

5.4 Tolerance to Dropped Nodes

Finally, we explore the effect of nodes dropping on the performance of MOCHA. We do not draw comparisons to other methods, as to the best of our knowledge, no other methods for distributed multi-task learning directly address fault tolerance. In MOCHA, we incorporate this setting by allowing $\theta_t^h := 1$, as explored theoretically in Section 4. In Figure 3, we look at the performance of MOCHA, either for one fixed $\mathbf{W}$ update, or running the entire MOCHA method, as the probability that nodes drop at each iteration (p_t^h in Assumption 2) increases. We see that the performance of MOCHA is robust to relatively high values of p_t^h, both during a single update of $\mathbf{W}$ and in how this affects the performance of the overall method. However, as intuition would suggest, if one of the nodes *never* sends updates (i.e., $p_1^h := 1$ for all h, green dotted line), the method does not converge to the correct solution. This provides validation for our Assumption 2.

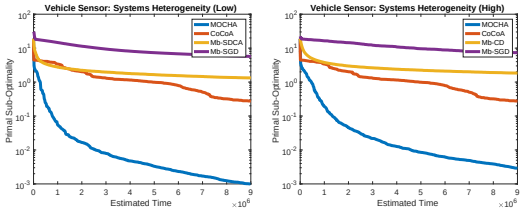

Figure 2: MOCHA can handle variability from systems heterogeneity.

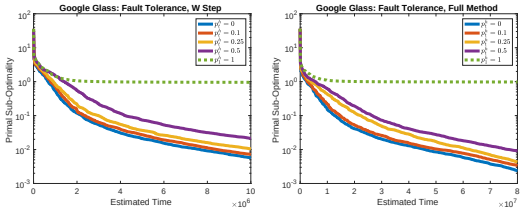

Figure 3: The performance of MOCHA is robust to nodes periodically dropping out (fault tolerance).

6 Discussion

To address the statistical and systems challenges of the burgeoning federated learning setting, we have presented MOCHA, a novel systems-aware optimization framework for federated multi-task learning. Our method and theory for the first time consider issues of high communication cost, stragglers, and fault tolerance for multi-task learning in the federated environment. While MOCHA does not apply to non-convex deep learning models in its current form, we note that there may be natural connections between this approach and "convexified" deep learning models [6, 34, 51, 57] in the context of kernelized federated multi-task learning.

Acknowledgements

We thank Brendan McMahan, Chloé Kiddon, Jakub Konečný, Evan Sparks, Xinghao Pan, Lisha Li, and Hang Qi for valuable discussions and feedback.

References

[1] A. Ahmed, A. Das, and A. J. Smola. Scalable hierarchical multitask learning algorithms for conversion optimization in display advertising. In *Conference on Web Search and Data Mining*, 2014.

[2] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Neural Information Processing Systems*, 2007.

[5] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[6] Ö. Aslan, X. Zhang, and D. Schuurmans. Convex deep learning via normalized kernels. In *Advances in Neural Information Processing Systems*, 2014.

[7] I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou. Asynchronous multi-task learning. In *International Conference on Data Mining*, 2016.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *SIGCOMM Workshop on Mobile Cloud Computing*, 2012.

[9] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *USENIX Annual Technical Conference*, 2010.

[10] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[11] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Conference on Knowledge Discovery and Data Mining*, 2011.

[12] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *VLDB Journal*, 14(4):417–443, 2005.

[13] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

[14] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Conference on Knowledge Discovery and Data Mining*, 2004.

[15] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: Vision and challenges. *SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[16] A. R. Gonçalves, F. J. Von Zuben, and A. Banerjee. Multi-task sparse structure learning with gaussian copula models. *Journal of Machine Learning Research*, 17(33):1–30, 2016.

[17] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007.

[18] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. Mobile fog: A programming model for large-scale applications on the internet of things. In *SIGCOMM Workshop on Mobile Cloud Computing*, 2013.

[19] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse Inverse Covariance Matrix Estimation Using Quadratic Approximation. In *Neural Information Processing Systems 27*, 2014.

[20] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An in-depth study of lte: Effect of network protocol and application behavior on performance. In *ACM SIGCOMM Conference*, 2013.

[21] L. Jacob, J.-p. Vert, and F. R. Bach. Clustered multi-task learning: A convex formulation. In *Neural Information Processing Systems*, 2009.

[22] M. Jaggi, V. Smith, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-Efficient Distributed Dual Coordinate Ascent. In *Neural Information Processing Systems*, 2014.

[23] X. Jin, P. Luo, F. Zhuang, J. He, and Q. He. Collaborating between local and global learning for distributed online multiple tasks. In *Conference on Information and Knowledge Management*, 2015.

[24] S. Kim and E. P. Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS Genet*, 5(8):e1000587, 2009.

[25] J. Konečnỳ, H. B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv:1511.03575*, 2015.

[26] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv:1610.05492*, 2016.

[27] T. Kuflik, J. Kay, and B. Kummerfeld. Challenges and solutions of ubiquitous user modeling. In *Ubiquitous display environments*, pages 7–30. Springer, 2012.

[28] A. Kumar and H. Daumé. Learning task grouping and overlap in multi-task learning. In *International Conference on Machine Learning*, 2012.

[29] S. L. Lauritzen. *Graphical Models*, volume 17. Clarendon Press, 1996.

[30] S. Liu, S. J. Pan, and Q. Ho. Distributed multi-task relationship learning. *Conference on Knowledge Discovery and Data Mining*, 2017.

[31] C. Ma, V. Smith, M. Jaggi, M. I. Jordan, P. Richtárik, and M. Takáč. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, 2015.

[32] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Symposium on Operating Systems Design and Implementation*, 2002.

[33] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[34] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2014.

[35] D. Mateos-Núñez and J. Cortés. Distributed optimization for multi-task learning via nuclear-norm approximation. In *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2015.

[36] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Conference on Artificial Intelligence and Statistics*, 2017.

[37] H. B. McMahan and D. Ramage. `http://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/`. *Google*, 2017.

[38] A. P. Miettinen and J. K. Nurminen. Energy efficiency of mobile clients in cloud computing. In *USENIX Conference on Hot Topics in Cloud Computing*, 2010.

[39] A. Pantelopoulos and N. G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(1):1–12, 2010.

[40] H. Qi, E. R. Sparks, and A. Talwalkar. Paleo: A performance model for deep neural networks. In *International Conference on Learning Representations*, 2017.

[41] S. A. Rahman, C. Merck, Y. Huang, and S. Kleinberg. Unintrusive eating recognition using google glass. In *Conference on Pervasive Computing Technologies for Healthcare*, 2015.

[42] P. Rashidi and D. J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on systems, man, and cybernetics*, 39(5):949–959, 2009.

[43] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 2016.

[44] S. Ravi. `https://research.googleblog.com/2017/02/on-device-machine-intelligence.html`. *Google*, 2017.

[45] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[46] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *International Conference on Machine Learning*, June 2007.

[47] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

[48] D. Singelée, S. Seys, L. Batina, and I. Verbauwhede. The communication and computation cost of wireless security. In *ACM Conference on Wireless Network Security*, 2011.

[49] V. Smith, S. Forte, C. Ma, M. Takác, M. I. Jordan, and M. Jaggi. CoCoA: A general framework for communication-efficient distributed optimization. *arXiv:1611.02189*, 2016.

[50] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro. Mini-Batch Primal and Dual Methods for SVMs. In *International Conference on Machine Learning*, 2013.

[51] C.-Y. Tsai, A. M. Saxe, and D. Cox. Tensor switching networks. In *Advances in Neural Information Processing Systems*, 2016.

[52] C. Van Berkel. Multi-core for mobile phones. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1260–1265. European Design and Automation Association, 2009.

[53] H. Wang, A. Banerjee, C.-J. Hsieh, P. K. Ravikumar, and I. S. Dhillon. Large scale distributed sparse precision estimation. In *Neural Information Processing Systems*, 2013.

[54] J. Wang, M. Kolar, and N. Srebro. Distributed multi-task learning. In *Conference on Artificial Intelligence and Statistics*, 2016.

[55] J. Wang, M. Kolar, and N. Srebro. Distributed multi-task learning with shared representation. *arXiv:1603.02185*, 2016.

[56] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[57] Y. Zhang, P. Liang, and M. J. Wainwright. Convexified convolutional neural networks. *International Conference on Machine Learning*, 2017.

[58] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

[59] J. Zhou, J. Chen, and J. Ye. Clustered multi-task learning via alternating structure optimization. In *Neural Information Processing Systems*, 2011.

A Preliminaries

Notation. We use $\mathbf{I}_{d\times d}$ to represent an identity matrix of size $d \times d$. When the context allows, we use the notation $\mathbf{I}$ to denote an identity matrix of an appropriate size. We also use $\otimes$ to denote the Kronecker product between two matrices.

Definition 2 (Matrix norm)**.** Given a symmetric positive definite matrix $\mathbf{M}$, the norm of $\mathbf{u}$ with respect to $\mathbf{M}$ is given by $\|\mathbf{u}\|_{\mathbf{M}} := \sqrt{\mathbf{u}^T\mathbf{M}\mathbf{u}}$.

Definition 3 (L-smooth)**.** A convex function f is L-smooth with respect to $\mathbf{M}$ if

$$f(\mathbf{u}) \leq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v}\rangle + \frac{L}{2}\|\mathbf{u} - \mathbf{v}\|_{\mathbf{M}}^2 \qquad \forall \mathbf{u}, \mathbf{v}\,. \tag{8}$$

If $\mathbf{M} = \mathbf{I}$ then, we simply say f is L-smooth.

Definition 4 (τ-strongly convex)**.** A function f is τ-strongly convex with respect to $\mathbf{M}$ if

$$f(\mathbf{u}) \geq f(\mathbf{v}) + \langle \mathbf{z}, \mathbf{u} - \mathbf{v}\rangle + \frac{\tau}{2}\|\mathbf{u} - \mathbf{v}\|_{\mathbf{M}}^2 \qquad \forall \mathbf{u}, \mathbf{v},\ \mathbf{z} \in \partial f(\mathbf{v})\,, \tag{9}$$

where $\partial f(\mathbf{v})$ is the set of sub-differentials of function f at $\mathbf{v}$. If $\mathbf{M} = \mathbf{I}$ then, we simply say f is τ-strongly convex.

Definition 5. The function f is called L-Lipchitz if for any $\mathbf{x}$ and $\mathbf{y}$ in its domain

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|_2\,. \tag{10}$$

If a function f is L-Lipchitz then its dual will be L-bounded, i.e., for any $\boldsymbol{\alpha}$ such that $\|\boldsymbol{\alpha}\|_2 > L$, then $f^*(\boldsymbol{\alpha}) = +\infty$.

B Multi-Task Learning

In this section, we summarize several popular multi-task learning formulations that can be written in the form of (1) and can therefore be addressed by our framework, MOCHA. While the $\mathbf{W}$ update is discussed in Section 3, we provide details here on how to solve the Ω update for these formulations.

B.1 Multi-Task Learning Formulations

MTL with cluster structure. In MTL models that assume a cluster structure, the weight vectors for each task, $\mathbf{w}_t$, are assumed to 'close' according to some metric to other weight vectors from tasks in the same cluster. This idea goes back to mean-regularized MTL [14], which assumes that all the tasks form one cluster, and that the weight vectors are close to their mean. Such a regularizer could be formulated in the form of (1) by choosing $\boldsymbol{\Omega} = (\mathbf{I}_{m\times m} - \frac{1}{m}\mathbf{1}\mathbf{1}^T)^2$, where $\mathbf{I}_{m\times m}$ is the identity matrix of size $m \times m$ and $\mathbf{1}_m$ represents a vector of all ones with size m. In this case, we set $\mathcal{R}$ to be

$$\mathcal{R}(\mathbf{W}, \boldsymbol{\Omega}) = \lambda_1 \operatorname{tr}\big(\mathbf{W}\boldsymbol{\Omega}\mathbf{W}^T\big) + \lambda_2\|\mathbf{W}\|_F^2\,, \tag{11}$$

where $\lambda_1, \lambda_2 > 0$ are parameters. Note that in this formulation, the structural dependence matrix $\boldsymbol{\Omega}$ is known a-priori. However, it is natural to assume multiple clusters exist, and to learn this clustering structure directly from the data [59]. For such a model, the problem formulation is non-convex if a perfect clustering structure is imposed [59, 21]. However, by performing a convex relaxation, the following regularizer is obtained [59, 21]

$$\mathcal{R}(\mathbf{W}, \boldsymbol{\Omega}) = \lambda \operatorname{tr}\big(\mathbf{W}(\eta\mathbf{I} + \boldsymbol{\Omega})^{-1}\mathbf{W}^T\big),\ \boldsymbol{\Omega} \in \mathcal{Q} = \Big\{\mathbf{Q} \mid \mathbf{Q} \succeq \mathbf{0},\ \operatorname{tr}(\mathbf{Q}) = k,\ \mathbf{Q} \preceq \mathbf{I}\Big\}, \tag{12}$$

where λ and η are regularization parameters, k is the number of clusters, and $\boldsymbol{\Omega}$ defines the clustering structure.

MTL with probabilistic priors. Another set of MTL models that can be realized by our framework enforce structure by putting probabilistic priors on the dependence among the columns of $\mathbf{W}$. For example, in [58] it is assumed that the weight matrix $\mathbf{W}$ has a prior distribution of the form:

$$\mathbf{W} \ \sim\ \left(\prod_{i=1}^{m}\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})\right)\ \mathcal{MN}(\mathbf{0}, \mathbf{I}_{d\times d} \otimes \boldsymbol{\Omega})\,, \tag{13}$$

where $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ denotes the normal distribution with mean $\mathbf{0}$ and covariance $\sigma^2\mathbf{I}$, and $\mathcal{MN}(\mathbf{0}, \mathbf{I}_{d\times d} \otimes \mathbf{\Omega})$ denotes the matrix normal distribution with mean $\mathbf{0}$, row covariance $\mathbf{I}_{d\times d}$, and column covariance $\mathbf{\Omega}$. This prior generates a regularizer of the following form [58]:

$$\mathcal{R}(\mathbf{W}, \mathbf{\Omega}) = \lambda \left(\frac{1}{\sigma^2} \|\mathbf{W}\|^2 + \operatorname{tr}\big(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T\big) \; + d \; \log|\mathbf{\Omega}| \right), \; \lambda > 0 \,.$$

Unfortunately, such a regularizer is non-convex with respect to $\mathbf{\Omega}$ due to the concavity of $\log|\mathbf{\Omega}|$. To obtain a jointly convex formulation in $\mathbf{\Omega}$ and $\mathbf{W}$, the authors in [58] propose omitting $\log|\mathbf{\Omega}|$ and controling the complexity of $\mathbf{\Omega}$ by adding a constraint on $\operatorname{tr}(\mathbf{\Omega})$:

$$\mathcal{R}(\mathbf{W}, \mathbf{\Omega}) = \lambda \left(\frac{1}{\sigma^2} \|\mathbf{W}\|^2 + \operatorname{tr}\big(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T\big) \right), \; \mathbf{\Omega} \in \mathcal{Q} = \left\{ \mathbf{Q} \mid \mathbf{Q} \succeq \mathbf{0}, \; \operatorname{tr}(\mathbf{Q}) = 1 \right\}. \quad (14)$$

It is worth noting that unlike the clustered MTL formulations, such as (2), the probabilistic formulation in (14) can model both positive and negative relationships among the tasks through the covariance matrix.

MTL with graphical models. Another way of modeling task relationships is through the precision matrix. This is popular in graphical models literature [29] because it encodes conditional independence among variables. In other words, if we denote the precision matrix among tasks in matrix variate Gaussian prior with $\mathbf{\Omega}$, then $\mathbf{\Omega}_{i,j} = 0$ if and only if tasks weights $\mathbf{w}_i$ and $\mathbf{w}_j$ are independent given the rest of the task weights [16]. Therefore, assuming sparsity in the structure among the tasks translates to sparsity in matrix $\mathbf{\Omega}$. As a result, we can formulate a sparsity-promoting regularizer by:

$$\mathcal{R}(\mathbf{W}, \mathbf{\Omega}) = \lambda \left(\frac{1}{\sigma^2} \|\mathbf{W}\|^2 + \operatorname{tr}\big(\mathbf{W}\mathbf{\Omega}\mathbf{W}^T\big) - d \; \log|\mathbf{\Omega}| \right) + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \|\mathbf{\Omega}\|_1 \,, \quad (15)$$

where $\lambda_1, \lambda_2 \geq 0$ control the sparsity of $\mathbf{W}$ and $\mathbf{\Omega}$ respectively [16]. It is worth noting that although this problem is jointly non-convex in $\mathbf{W}$ and $\mathbf{\Omega}$, it is bi-convex.

B.2 Strong Convexity of MTL Regularizers

Recall that in Assumption 1, we presumed that the vectorized formulation of the MTL regularizer is strongly convex with respect to a matrix $\mathbf{M}^{-1}$. In this subsection we discuss the choice of matrix $\mathbf{M}$ for the widely-used MTL formulations introduced in Section B.1.

Using the notation from Remark 1 for the clustered MTL formulation (11), it is easy to see that $\bar{\mathcal{R}}(\mathbf{w}, \bar{\mathbf{\Omega}}) = \lambda_1 \mathbf{w}^T \bar{\mathbf{\Omega}} \mathbf{w} + \lambda_2 \|\mathbf{w}\|_2^2$, where $\bar{\mathbf{\Omega}} := \mathbf{\Omega} \otimes \mathbf{I}_{d\times d}$. As a result, it is clear that $\bar{\mathcal{R}}(\mathbf{w}, \bar{\mathbf{\Omega}})$ is 1-strongly convex with respect to $\mathbf{M}^{-1} = \lambda_1 \bar{\mathbf{\Omega}} + \lambda_2 \mathbf{I}_{md\times md}$.

Using a similar reasoning, it is easy to see that the matrix $\mathbf{M}$ can be chosen as $\lambda^{-1}(\eta\mathbf{I} + \bar{\mathbf{\Omega}})$, $\lambda^{-1}(\frac{1}{\sigma^2}\mathbf{I} + \bar{\mathbf{\Omega}}^{-1})^{-1}$ and $\lambda^{-1}(\frac{1}{\sigma^2}\mathbf{I} + \bar{\mathbf{\Omega}})^{-1}$ for (12), (14) and (15) respectively.

B.3 Optimizing Ω in MTL Formulations

In this section, we briefly cover approaches to update $\mathbf{\Omega}$ in the MTL formulations introduced in Section B.1. First, it is clear that (2) does not require any updates to $\mathbf{\Omega}$, as it is assumed to be fixed. In (12), it can be shown [59, 21] that the optimal solution for $\mathbf{\Omega}$ has the same column space as the rows of $\mathbf{W}$. Therefore, the problem boils down to solving a simple convex optimization problem over the eigenvalues of $\mathbf{\Omega}$; see [59, 21] for details. Although outside the scope of this paper, we note that the bottleneck of this approach to finding $\mathbf{\Omega}$ is computing the SVD of $\mathbf{W}$, which can be a challenging problem when m is large. In the probabilistic model of (14), the $\mathbf{\Omega}$ update is given in [58] by $(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}$, which requires computing the eigenvalue decomposition of $\mathbf{W}^T\mathbf{W}$. For the graphical model formulation, the problem of solving for $\mathbf{\Omega}$ is called sparse precision matrix estimation or graphical lasso [16]. This is a well-studied problem, and many scalable algorithms have been proposed to solve it [53, 16, 19].

B.3.1 Reducing the Size of Ω by Sharing Tasks

One interesting aspect of MOCHA is that the method can be easily modified to accommodate the sharing of tasks among the nodes without any change to the local solvers. This property helps the central node to reduce the size of $\mathbf{\Omega}$ and the complexity of its update with minimal changes to the whole system. The following remark highlights this capability.

Remark 4. MOCHA *can be modified to solve problems when there are tasks that are shared among nodes. In this case, each node still solves a data local sub-problem based on its own data for the task, but the central node needs to do an additional aggregation step to add the results for all the nodes that share the data of each task. This reduces the size of matrix* $\mathbf{\Omega}$ *and simplifies its update.*

C Convergence Analysis

Notation. In the rest of this section we use the superscript (h) or h to denote the corresponding variable at iteration (h) of the federated update in MOCHA. When context allows, we drop the superscript to simplify notation.

In order to provide a general convergence analysis, similar to the ones provided in [22, 31, 49], we assume an aggregation parameter $\gamma \in (0, 1]$ in this section. With such an aggregation parameter, the updates in each federated iteration would be $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_t + \gamma \Delta\boldsymbol{\alpha}_t$ and $\mathbf{v}_t \leftarrow \mathbf{v}_t + \gamma \Delta\mathbf{v}_t$. For a more detailed discussion on the role of aggregation parameter, see Appendix D.1. Note that in Algorithm 1, MOCHA is presented assuming $\gamma = 1$ for simplicity.

Before proving our convergence guarantees, we provide several useful definitions and key lemmas.

Definition 6. For each task t, define

$$\sigma_t := \max_{\boldsymbol{\alpha} \in \mathbb{R}^{n_t}} \frac{\|\mathbf{X}_t \boldsymbol{\alpha}\|_{M_t}^2}{\|\boldsymbol{\alpha}\|^2} \text{ and } \sigma_{\max} := \max_{t \in [m]} \sigma_t. \tag{16}$$

Definition 7. For any $\boldsymbol{\alpha}$, define the duality gap as

$$G(\boldsymbol{\alpha}) := \mathcal{D}(\boldsymbol{\alpha}) - (-\mathcal{P}(\mathbf{W}(\boldsymbol{\alpha}))), \tag{17}$$

where $\mathcal{P}(\mathbf{W}) := \sum_{t=1}^{m} \sum_{i=1}^{n_t} \ell_t(\mathbf{w}_t^T \mathbf{x}_t^i, y_t^i) + \mathcal{R}(\mathbf{W}, \mathbf{\Omega})$ as in (1).

The following lemma uses Assumption 2 to bound the average performance of θ_t^h, which is crucial in providing global convergence guarantees for MOCHA.

Lemma 3. *Under Assumption 2,* $\Theta_t^h \leq \bar{\Theta} = p_{\max} + (1 - p_{\max})\Theta_{\max} < 1$.

Proof. Recalling the definitions $p_t^h := \mathbb{P}[\theta_t^h = 1]$ and $\hat{\Theta}_t^h := \mathbb{E}[\theta_t^h | \theta_t^h < 1, \mathcal{H}_h]$, we have

$$\begin{aligned}
\Theta_t^h &= \mathbb{E}[\theta_t^h | \mathcal{H}_h] \\
&= \mathbb{P}[\theta_t^h = 1] \cdot \mathbb{E}[\theta_t^h | \theta_t^h = 1, \mathcal{H}_h] + (1 - \mathbb{P}[\theta_t^h < 1]) \cdot \mathbb{E}[\theta_t^h | \theta_t^h < 1, \mathcal{H}_h] \\
&= p_t^h \cdot 1 + (1 - p_t^h) \cdot \hat{\Theta}_t^h \leq \bar{\Theta} < 1,
\end{aligned}$$

where the last inequality is due to Assumption 2, and the fact that $\hat{\Theta}_t^h < 1$ by definition. □

The next key lemma bounds the dual objective of an iterate based on the dual objective of the previous iterate and the objectives of local subproblems.

Lemma 4. *For any* $\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha} \in \mathbb{R}^n$ *and* $\gamma \in (0, 1]$ *if* σ' *satisfies* (28)*, then*

$$\mathcal{D}\left(\boldsymbol{\alpha} + \gamma \Delta\boldsymbol{\alpha}\right) \leq (1 - \gamma)\mathcal{D}(\boldsymbol{\alpha}) + \gamma \sum_{t=1}^{m} \mathcal{G}_t^{\sigma'}(\Delta\boldsymbol{\alpha}_t; \mathbf{v}, \boldsymbol{\alpha}_t)\,. \tag{18}$$

Proof. The proof of this lemma is similar to [49, Lemma 1] and follows from the definition of local sub-problems, smoothness of $\mathcal{R}^*$ and the choice of σ' in (28). □

Recall that if the functions ℓ_t are $(1/\mu)$-smooth, their conjugates ℓ_t^* will be μ-strongly convex. The lemma below provides a bound on the amount of improvement in dual objective in each iteration.

Lemma 5. *If the functions* ℓ_t^* *are* μ*-strongly convex for some* $\mu \geq 0$*. Then, for any* $s \in [0, 1]$*.*

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^{(h+1)}) | \mathcal{H}_h] \geq \gamma \sum_{t=1}^{m} (1 - \bar{\Theta}) \left(s G_t(\boldsymbol{\alpha}^{(h)}) - \frac{\sigma' s^2}{2} J_t \right), \tag{19}$$

where

$$G_t(\boldsymbol{\alpha}) := \sum_{i=1}^{n_t} \left[\ell_t^*(-\boldsymbol{\alpha}_t^i) + \ell_t(\mathbf{w}_t(\boldsymbol{\alpha})^\top \mathbf{x}_t^i, y_t^i) + \boldsymbol{\alpha}_t^i \mathbf{w}_t(\boldsymbol{\alpha})^\top \mathbf{x}_t^i\right] , \tag{20}$$

$$J_t := -\tfrac{\mu(1-s)}{\sigma' s}\|(\mathbf{u}_t - \boldsymbol{\alpha}_t^{(h)})\|^2 + \|\mathbf{X}_t(\mathbf{u}_t - \boldsymbol{\alpha}_t^{(h)})\|_{M_t}^2 , \tag{21}$$

for $\mathbf{u}_t \in \mathbb{R}^{n_t}$ *with*

$$\mathbf{u}_t^i \in \partial \ell_t(\mathbf{w}_t(\boldsymbol{\alpha})^\top \mathbf{x}_t^i, y_t^i) . \tag{22}$$

Proof. Applying Lemma 4 and recalling $\mathcal{D}(\boldsymbol{\alpha}) = \sum_{t=1}^m \mathcal{G}_t^{\sigma'}(\mathbf{0}; \mathbf{v}, \boldsymbol{\alpha}_t)$, we can first bound the improvement for each task separately. Following a similar approach as in the proof of [49, Lemma 7] we can obtain the bound (19) which bounds the improvement from $\boldsymbol{\alpha}^{(h)}$ to $\boldsymbol{\alpha}^{(h+1)}$. □

The following lemma relates the improvement of the dual objective in one iteration to the duality gap for the smooth loss functions ℓ_t.

Lemma 6. *If the loss functions ℓ_t are $(1/\mu)$-smooth, then there exists a proper constants $s \in (0, 1]$, such that for any $\gamma \in (0, 1]$ at any iteration h*

$$\mathbb{E}\left[\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^{(h+1)})|\mathcal{H}_h\right] \geq s\gamma(1-\bar{\Theta})G(\boldsymbol{\alpha}^{(h)}), \tag{23}$$

where $G(\boldsymbol{\alpha}^{(h)})$ is the duality gap of $\boldsymbol{\alpha}^{(h)}$ which is defined in (17)*.*

Proof. Recall the definition of $\sigma_{\max}$ in (16). Now, if we carefully choose $s = \mu/(\mu + \sigma_{\max}\sigma')$, it is easy to show that $J_t \leq 0$ in (19); see [49, Theorem 11] for details. The final result follows as a consequence of Lemma 5. □

Note that Lemma 5 holds even if the functions are non-smooth, i.e. $\mu = 0$. However, we cannot infer sufficient decrease of Lemma 6 from Lemma 5 when $\mu = 0$. Therefore, we need additional tools when the losses are L-Liptchitz. The first is the following lemma, which bounds the J term in (19).

Lemma 7. *Assume that the loss functions ℓ_t are L-Lipschitz. Denote $J := \sum_{t=1}^m J_t$, where J_t is defined in* (21)*, then*

$$J \leq 4L^2 \sum_{t=1}^m \sigma_t n_t := 4L^2\sigma, \tag{24}$$

where σ_t is defined in (16)*.*

Proof. The proof is similar to [31, Lemma 6] and using the definitions of σ and σ_t and the fact that the losses are L-Lipchitz. □

C.1 Convergence Analysis for Smooth Losses

C.1.1 Proof of Theorem 1

Let us rewrite (23) from Lemma 6 as

$$\begin{aligned}\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^{(h+1)})|\mathcal{H}_h] &= \mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^\star) + \mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^\star) - \mathcal{D}(\boldsymbol{\alpha}^{(h+1)})|\mathcal{H}_h] \\ &\geq s\gamma(1-\bar{\Theta})G(\boldsymbol{\alpha}^{(h)}) \\ &\geq s\gamma(1-\bar{\Theta})\left(\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)\right),\end{aligned}$$

where the last inequality is due to weak duality, i.e. $G(\boldsymbol{\alpha}^{(h)}) \geq \mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)$. Re-arranging the terms in the above inequality, we can easily get

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(h+1)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)|\mathcal{H}_h] \leq \left(1 - s\gamma(1-\bar{\Theta})\right)\left(\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)\right) \tag{25}$$

Recursively applying this inequality and taking expectations from both sides, we arrive at

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(h+1)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)] \leq \left(1 - s\gamma(1-\bar{\Theta})\right)^{h+1} \left(\mathcal{D}(\boldsymbol{\alpha}^{(0)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)\right). \tag{26}$$

Now we can use a simple bound on the initial duality gap [49, Lemma 10], which states that $\mathcal{D}(\boldsymbol{\alpha}^{(0)}) - \mathcal{D}(\boldsymbol{\alpha}^\star) \leq n$, to get the final result. It is worth noting that we can translate the bound on the dual distance to optimality to the bound on the duality gap using the following inequalities

$$s\gamma(1-\bar{\Theta})\,\mathbb{E}[G(\boldsymbol{\alpha}^{(H)})] \leq \mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(H)}) - \mathcal{D}(\boldsymbol{\alpha}^{(H+1)})] \leq \mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(H)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)] \leq \epsilon_{\mathcal{D}}, \tag{27}$$

where the first inequality is due to (23), the second inequality is due to the optimality of $\boldsymbol{\alpha}^\star$, and the last inequality is the bound we just proved for the dual distance to optimality.

C.1.2 Asymptotic Convergence

In the case of smooth loss functions, it is possible to get asymptotic convergence results under milder assumptions. The following corollary is an extension of Theorem 1.

Corollary 8. *If the loss functions ℓ_t are μ-smooth, then under Assumption 1, $\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(H)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)] \to 0$ as $H \to \infty$ if either of the following conditions hold*

- $\limsup_{h\to\infty} p_t^h < 1$ *and* $\limsup_{h\to\infty} \hat{\Theta}_t^h < 1$.
- *For any task t, $\left(1 - p_t^h\right) \times \left(1 - \hat{\Theta}_t^h\right) = \omega(\frac{1}{h})$. Note that in this case $\lim_{h\to\infty} p_t^h$ can be equal to 1.*

Proof. The proof is similar to the proof of Theorem 1. We can use the same steps to get a sufficient decrease inequality like the one in (25), with $\bar{\Theta}$ replaced with $\bar{\Theta}^h := \max_t\ \Theta_t^h$.

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(h+1)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)|\mathcal{H}_h] \leq \left(1 - s\gamma(1-\bar{\Theta}^h)\right) \left(\mathcal{D}(\boldsymbol{\alpha}^{(h)}) - \mathcal{D}(\boldsymbol{\alpha}^\star)\right)$$

The rest of the argument follows by applying this inequality recursively and using the assumptions in the corollary. □

C.2 Convergence Analysis for Lipschitz Losses: Proof for Theorem 2

Proof. For L-Lipschitz loss functions, the proof follows the same line of reasoning as the proof of Theorem 8 in [31] and therefore we do not cover it in detail. Unlike the case with smooth losses, it is not possible to bound the decrease in dual objective by (23). However, we can use Lemma 5 with $\mu = 0$. The next step is to bound $J = \sum_{t=1}^m J_t$ in (19), which can be done via Lemma 7. Finally, we apply the inequalities recursively, choose s carefully, and bound the terms in the final inequality. We refer the reader to the proof of Theorem 8 in [31] for more details. It is worth noting that similar to Theorem 1, we can similarly get bounds on the expected duality gap, instead of the dual objective. □

D Choosing σ'

In order to guarantee the convergence of the federated update of MOCHA, the parameter σ' must satisfy:

$$\sigma' \sum_{t=1}^m \|\mathbf{X}_t\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2 \geq \gamma\|\mathbf{X}\boldsymbol{\alpha}\|_{\mathbf{M}}^2 \ \ \forall \boldsymbol{\alpha} \in \mathbb{R}^n\,, \tag{28}$$

where $\gamma \in (0, 1]$ is the aggregation parameter for MOCHA Algorithm. Note that in Algorithm 1 we have assumed that $\gamma = 1$. Based on Remark 1, it can be seen that the matrix $\mathbf{M}$ in Assumption 1 can be chosen of the form $\mathbf{M} = \bar{\mathbf{M}} \otimes \mathbf{I}_{d\times d}$, where $\bar{\mathbf{M}}$ is a positive definite matrix of size $m \times m$. For such a matrix, the following lemma shows how to choose σ'.

Lemma 9. *For any positive definite matrix $\mathbf{M} = \bar{\mathbf{M}} \otimes \mathbf{I}_{d\times d}$,*

$$\sigma' := \gamma \max_t \sum_{t'=1}^m \frac{|\bar{\mathbf{M}}_{tt'}|}{\bar{\mathbf{M}}_{tt}} \tag{29}$$

satisfies the inequality (28).

Proof. First of all it is worth noting that for any t, $\mathbf{M}_t = \bar{\mathbf{M}}_t \otimes \mathbf{I}_{d\times d}$. For any $\boldsymbol{\alpha} \in \mathbb{R}^n$

$$\begin{aligned}
\gamma\|\mathbf{X}\boldsymbol{\alpha}\|_{\mathbf{M}}^2 &= \gamma \sum_{t,t'} \bar{\mathbf{M}}_{tt'} \langle \mathbf{X}_t\boldsymbol{\alpha}_t, \mathbf{X}_{t'}\boldsymbol{\alpha}_{t'}\rangle \\
&\leq \gamma \sum_{t,t'} \frac{1}{2}|\bar{\mathbf{M}}_{tt'}| \left(\frac{1}{\bar{\mathbf{M}}_{tt}}\|\mathbf{X}_t\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2 + \frac{1}{\bar{\mathbf{M}}_{t't'}}\|\mathbf{X}_{t'}\boldsymbol{\alpha}_{t'}\|_{\mathbf{M}_{t'}}^2 \right) \\
&= \gamma \sum_{t} \left(\sum_{t'} \frac{|\bar{\mathbf{M}}_{tt'}|}{\bar{\mathbf{M}}_{tt}} \right) \|\mathbf{X}_t\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2 \\
&\leq \sigma' \sum_{t} \|\mathbf{X}_t\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2,
\end{aligned}$$

where the first inequality is due to Cauchy-Schwartz and the second inequality is due to definition of σ'. □

Remark 5. *Based on the proof of Lemma 9, it is easy to see that we can choose σ' differently across the tasks in our algorithm to allow tasks that are more loosely correlated with other tasks to update more aggressively. To be more specific, if we choose $\sigma'_t = \gamma \sum_{t'} \frac{|\bar{\mathbf{M}}_{tt'}|}{\bar{\mathbf{M}}_{tt}}$, then it it is possible to show that $\gamma\|\mathbf{X}\boldsymbol{\alpha}\|_{\mathbf{M}}^2 \leq \sum_{t=1}^m \sigma'_t \|\mathbf{X}_t\boldsymbol{\alpha}_t\|_{\mathbf{M}_t}^2$ for any $\boldsymbol{\alpha}$, and the rest of the convergence proofs will follow.*

D.1 The Role of Aggregation Parameter γ

The following remark highlights the role of aggregation parameter γ.

Remark 6. *Note that the when $\gamma < 1$ the chosen σ' in* (28) *would be smaller compared to the case where $\gamma = 1$. This means that the local subproblems would be solved with less restrictive regularizer. Therefore, the resulting $\Delta\boldsymbol{\alpha}$ would be more aggressive. As a result, we need to do a more conservative update $\boldsymbol{\alpha} + \gamma\Delta\boldsymbol{\alpha}$ in order to guarantee the convergence.*

Although aggregation parameter γ is proposed to capture this trade off between aggressive subproblems and conservative updates, in most practical scenarios $\gamma = 1$ has the best empirical performance.

E Simulation Details

In this section, we provide additional details and results of our empirical study.

E.1 Datasets

In Table 2, we provide additional details on the number of tasks (m), feature size (d), and per-task data size (n_t) for each federated dataset described in Section 5. The standard deviation n_σ is a measure data skew, and calculates the deviation in the sizes of training data points for each task, n_t. All datasets are publicly available.

Table 2: Federated Datasets for Empirical Study.

Dataset	**Tasks** (m)	**Features** (d)	**Min** n_t	**Max** n_t	**Std. Deviation** n_σ
Human Activity	30	561	210	306	26.75
Google Glass	38	180	524	581	11.07
Land Mine	29	9	333	517	65.39
Vehicle Sensor	23	100	872	1,933	267.47

E.2 Multi-Task Learning with Highly Skewed Data

To generate highly skewed data, we sample from the original training datasets so that the task dataset sizes differ by at least two orders of magnitude. The sizes of these highly skewed datasets are shown

Table 3: Skewed Datasets for Empirical Study.

Dataset	**Tasks** (m)	**Features** (d)	**Min** n_t	**Max** n_t	**Std. Deviation** σ
HA-Skew	30	561	3	306	84.41
GG-Skew	38	180	6	581	111.79
LM-Skew	29	9	5	517	181.92
VS-Skew	23	100	19	1,933	486.08

in Table 3. When looking at the performance of local, global, and multi-task models for these datasets (Table 4), the global model performs slightly better in this setting (particularly for the Human Activity and Land Mine datasets). However, multi-task learning still significantly outperforms all models.

Table 4: Average prediction error for skewed data: means and standard errors over 10 random shuffles.

Model	HA-Skew	GG-Skew	LM-Skew	VS-Skew
Global	2.41 (0.30)	5.38 (0.26)	29.28 (2.47)	13.58 (0.23)
Local	3.87 (0.37)	4.96 (0.20)	27.63 (1.15)	8.15 (0.19)
MTL	**1.93 (0.44)**	**3.28 (0.15)**	**24.12 (1.08)**	**6.91 (0.21)**

E.3 Implementation Details

In this section, we provide thorough details on our experimental setup and compared methods.

Methods.

- **Mb-SGD.** Mini-batch stochastic gradient descent is a standard, widely used method for parallel and distributed optimization. See, e.g., a discussion of this method for the SVM models of interest [46]. We tune both the mini-batch size and step size for best performance using grid search.
- **Mb-SDCA.** Mini-batch SDCA aims to improve mini-batch SGD by employing coordinate ascent in the dual, which has encouraging theoretical and practical backings [47, 50]. For all experiments, we scale the updates for mini-batch stochastic dual coordinate ascent at each round by $\frac{\beta}{b}$ for mini-batch size b and $\beta \in [1, b]$, and tune both parameters with grid search.
- **CoCoA.** We generalize CoCoA [22, 31] to solve (1), and tune θ, the fixed approximation parameter, between $[0, 1)$ via grid search. For both CoCoA, and MOCHA, we use coordinate ascent as a local solver for the dual subproblems (4).
- **MOCHA.** The only parameter necessary to tune for MOCHA is the level of approximation quality θ_t^h, which can be directly tuned via H_i, the number of local iterations of the iterative method run locally. In Section 4, our theory relates this parameter to global convergence, and we discuss the practical effects of this parameter in Section 3.4.

Computation and Communication Complexities. We provide a brief summary of the above methods from the point of view of computation, communication, and memory complexities. MOCHA is superior in terms of its computation complexity compared to other distributed optimization methods, as MOCHA allows for flexibility in its update of W. At one extreme, the update can be based on a single data point per iteration in parallel, similar to parallel SGD. At the other extreme, MOCHA can completely solve the subproblems on each machine, similar to methods such as ADMM. This flexibility of computation yields direct benefits in terms of communication complexity, as performing additional local computation will result in fewer communication steps. Note that all methods, including MOCHA, communicate the same size vector at each iteration, and so the main difference is in how many communication rounds are necessary for convergence. In terms of memory, MOCHA must maintain the task matrix, Ω, on the master server. While this overhead is greater than most *non-MTL* (global or local) approaches, the task matrix is typically low-rank by design and the overhead is thus manageable. We discuss methods for computing Ω in further detail in Section B.3.

Estimated Time. To estimate the time to run methods in the federated setting, we carefully count the floating-point operations (FLOPs) performed in each local iteration for each method, as well as the size and frequency of communication. We convert these counts to estimated time (in milliseconds), using known clock rate and bandwidth/latency numbers for mobile phones in 3G, LTE, and wireless networks [52, 20, 48, 9, 38]. In particular, we use the following standard model for the cost of one round, h, of local computation / communication on a node t:

$$\textit{Time}(h,t) := \frac{\textit{FLOPs}(h,t)}{\textit{Clock Rate}(t)} + \textit{Comm}(h,t) \tag{30}$$

Note that the communication cost $\textit{Comm}(h,t)$ includes both bandwidth and latency measures. Detailed models of this type have been used to closely match the performance of real-world systems [40].

Statistical Heterogeneity. To account for statistical heterogeneity, MOCHA and the mini-batch methods (Mb-SGD and Mb-SDCA) can adjust the number of local iterations or batch size, respectively, to account for difficult local problems or high data skew. However, because COCOA uses a fixed accuracy parameter θ across both the tasks and rounds, changes in the subproblem difficulty and data skew can make the computation on some nodes much slower than on others. For COCOA, we compute θ via the duality gap, and carefully tune this parameter between $[0, 1)$ for best performance. Despite this, the number of local iterations needed for θ varies significantly across nodes, and as the method runs, the iterations tend to increase as the subproblems become more difficult.

Systems Heterogeneity. Beyond statistical heterogeneity, there can be variability in the systems themselves that cause changes in performance. For example, low battery levels, poor network connections, or low memory may reduce the ability a solver has on a local node to compute updates. As discussed in Section 3.4, MOCHA assumes that the central node sets some global clock cycle, and the t-th worker determines the amount of feasible local computation given this clock cycle along with its systems constraints. This specified amount of local computation corresponds to some implicit value of θ_t^h based on the underlying systems and statistical challenges for the t-th node.

To model this setup in our simulations, it suffices to fix a global clock sycle and then randomly assign various amounts of local computation to each local node at each iteration. Specifically, in our simulations we charge all nodes the same fixed computation cost at each iteration over an LTE network, but force some nodes to perform less updates given their current systems constraints. At each round, we assign the number of updates for node t between $[0.1n_{\min}, n_{\min}]$ for *high variability* environments, and between $[0.9n_{\min}, n_{\min}]$ for *low variability* environments, where $n_{\min} := \min_t n_t$ is the minimum number of local data points across tasks.

For the mini-batch methods, we vary the mini-batch size in a similar fashion. However, we do not follow this same process for COCOA, as this would require making the θ parameter worse than what was optimally tuned given statistical heterogeneity. Hence, in these simulations we do not introduce any additional variability (and thus present overly optimistic results for COCOA). In spite of this, we see that in both low and high variability settings, MOCHA significantly outperforms all other methods and is robust to systems-related heterogeneity.

Fault Tolerance. Finally, we demonstrate that MOCHA can handle nodes periodically dropping out, which is also supported in our convergence results in Section 4. We perform this simulation using the notation defined in Assumption 2, i.e., that each node t temporarily drops on iteration h with probability p_t^h. In our simulations, we modify this probability directly and show that MOCHA is robust to fault tolerance in Figure 3. However, note that this robustness is not merely due to statistical redundancy: If we are to drop out a node entirely (as shown in the green dotted line), MOCHA will not converge to the correct solution. This provides insight into our Assumption 2, which requires that the probability that a node drops at each round cannot be exactly equal to one.