Byzantine Resilient and Fast Federated Few-Shot Learning

Ankit Pratap Singh [1] **Namrata Vaswani** [1]

Abstract

This work introduces a Byzantine resilient solution for learning low-dimensional linear representation. Our main contribution is the development of a provably Byzantine-resilient AltGDmin algorithm for solving this problem in a federated setting. We argue that our solution is sample-efficient, fast, and communication-efficient. In solving this problem, we also introduce a novel secure solution to the federated subspace learning meta-problem that occurs in many different applications.

1. Introduction

Multi-task representation learning refers to the problem of jointly estimating the model parameters for a set of related tasks. This is typically done by learning a common "representation" for all of their source vectors (feature vectors). This learned representation can then be used for solving the meta-learning or learning-to-learn problem: learning model parameters in a data-scarce environment. This strategy is referred to as "few-shot" learning. In recent work (Du, Hu, Kakade, Lee, & Lei, 2020), a very interesting low-dimensional linear representation was introduced and the corresponding low rank matrix learning optimization problem was defined. However, (Du et al., 2020) assumed that this optimization problem (see eq. (1)), which is non-convex, can be correctly solved. It is mentioned that it should be possible to solve it by solving a nuclear norm based convex relaxation of it. However, there are no known guarantees to ensure that the solution to the relaxation is indeed also a solution of the original problem. Moreover, convex relaxations are known to be very slow to solve (compared with direct iterative solutions) (Jain, Kar, et al., 2017; Netrapalli, Jain, & Sanghavi, 2013): these need order $1/\sqrt{\epsilon}$ number of iterations to obtain an ϵ accurate solution. In follow-up work, (Tripuraneni, Jin, & Jordan, 2021) studied a special case in which all the source vectors for the different tasks are the same. It introduced a method of moments estimator that is faster, but needs many more samples; sample complexity grows as $1/\epsilon^2$.

In interesting parallel works (Nayer & Vaswani, 2023, on arXiv since Feb. 2021; Collins, Hassani, Mokhtari, & Shakkottai, 2021), a fast and communication-efficient GD-based algorithm, that was referred to as Alternating GD and Minimization (AltGDmin) and FedRep respectively, was introduced for solving the mathematical problem given in (1), when the available number of training samples per task is much lesser than the regression vector length. Follow-up work (Vaswani, 2024) improved the guarantees for AltGDmin while also simplifying the proof. AltGDmin and FedRep algorithms are identical except for the initialization step. AltGDmin uses a better initialization and hence also has a better sample complexity by a factor of r. The latter (FedRep) paper referred to the problem of (1) as multi-task linear representation learning. The former (AltGDmin) paper used federated sketching, dynamic MRI (Babu, Lingala, & Vaswani, 2023) as motivating applications. It also solved the phaseless generalization of (1) called low rank phase retrieval. In older work (Nayer & Vaswani, 2021; Nayer, Narayanamurthy, & Vaswani, 2020, 2019), an alternating minimization (AltMin) solution to this problem was developed and analyzed as well. Since (1) is a special case of this more general problem, this AltMin solution also solves (1). All these works study the centralized setting or the attack-free federated setting. Other somewhat related works include (Shen, Ye, Kang, Hassani, & Shokri, 2023; Tziotis, Shen, Pedarsani, Hassani, & Mokhtari, 2022). A longer version of the mathematical problem being solved in this work (Byzantine resilient low rank column-wise compressive sensing) is at (Singh & Vaswani, 2024).

1.1. Contributions

We adapt the altGDmin algorithm described above to show how it can solve the multi-task linear representation learning and few shot learning problems. Our main contribution is the development of a *provably Byzantine-resilient* AltGDmin-based solution for solving this problem in a federated setting. Our solution is communication-efficient along with being fast and sample-efficient. In this setting, resilience to adversarial attacks on some nodes is an impor-

[1]Department of Electrical and Computer Engineering, Iowa State University, Ames IA, USA. Correspondence to: Ankit Pratap Singh <sankit@iastate.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

tant requirement. The most general attack is the Byzantine attack. For this, the attacking nodes can collude; and all the attacking nodes know the outputs of all the nodes, the algorithm being implemented by the center, and the algorithm parameters.

In solving the above problem, we introduce a novel solution approach, called *Subspace Median*, for combining subspace estimates from multiple federated nodes when some of them can be malicious. This approach and its guarantee (Lemma 3.1) are of independent interest for developing a secure solution to the federated subspace learning meta-problem that occurs in many applications – (online) PCA, subspace tracking, initializing many sparse recovery, low rank matrix recovery, or phase retrieval problems.

1.2. Related Work

Few-shot learning is applied across various tasks such as image classification (Vinyals, Blundell, Lillicrap, Wierstra, et al., 2016), sentiment analysis from short texts (Yu et al., 2018), and object recognition (Fei-Fei, Fergus, & Perona, 2006), with much of the focus on practical experimentation over theoretical development (Snell, Swersky, & Zemel, 2017; Ravi & Larochelle, 2016; Sung et al., 2018; Boudiaf et al., 2020). Representation learning, a significant method within this field, has been highlighted in several studies (Sun, Shrivastava, Singh, & Gupta, 2017; Goyal, Mahajan, Gupta, & Misra, 2019), though they often fall short of providing algorithmic guarantees for provably solving the representation learning problem (Du et al., 2020; Baxter, 2000; Maurer, Pontil, & Romera-Paredes, 2016; Tripuraneni et al., 2021; Tripuraneni, Jordan, & Jin, 2020; Y. Li, Ildiz, Papailiopoulos, & Oymak, 2023). Recent work by (Collins et al., 2021) and (Nayer & Vaswani, 2023, on arXiv since Feb. 2021; Vaswani, 2024) developed a provable algorithm to solve the low-dimensional linear representation learning problem, although they do not consider Byzantine attacks. There are other line of works which extends the low-dimensional linear representation learning problem (Shen et al., 2023), which focuses on Differential Privacy. The algorithm CENTAUR, presented in their work, aligns with the server and client procedures outlined in the study by (Collins et al., 2021), with the notable addition of the Gaussian mechanism. The work presented in (Tziotis et al., 2022) addresses the challenge of stragglers. To combat the straggler effect, the paper introduces a novel sampling mechanism that utilizes a "doubling" strategy.

Geometric Median is one of the aggregation method to handle Byzantine attacks. (Chen, Su, & Xu, 2017) develops non-asymptotic analysis in stochastic gradient descent utilized the geometric median of means, giving convergence guarantees under specific conditions. Follow-up work uses coordinate-wise mean and trimmed-mean estimators (Yin, Chen, Kannan, & Bartlett, 2018) but with assumption of bounded variance and coordinate-wise bounded skewness (or coordinate-wise sub-exponential) on the gradient distribution. (Alistarh, Allen-Zhu, & Li, 2018; Allen-Zhu, Ebrahimian, Li, & Alistarh, 2020) provided non-asymptotic guarantees for Byzantine resilient stochastic gradient descent, assuming a consistent set of Byzantine nodes across iterations.

Some studies have explored heterogeneous data distributions, establishing results within bounds of heterogeneity (Pillutla, Kakade, & Harchaoui, 2019; Data & Diggavi, 2021; L. Li, Xu, Chen, Giannakis, & Ling, 2019; Ghosh, Hong, Yin, & Ramchandran, 2019). While (Regatti, Chen, & Gupta, 2022; Lu, Li, Chen, & Ma, 2022; Cao, Fang, Liu, & Gong, 2020; Cao & Lai, 2019; Xie, Koyejo, & Gupta, 2019) use detection methods to manage heterogeneous gradients with a trusted dataset at central server.

1.3. Problem Set up

First consider the centralized setting. Suppose that there are q source tasks, each task $k \in [q]$ associated with a distribution over the input-output space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \Re^n$ and $\mathcal{Y} \subseteq \Re$. The aim is to learn prediction functions for all tasks simultaneously, leveraging a shared representation $\varphi : \mathcal{X} \to \mathcal{Z}$ that maps inputs to a feature space $\mathcal{Z}$. We let the representation function class be Low-Dimensional Linear Representations i.e., $\{\boldsymbol{x} \mapsto \boldsymbol{U}^T\boldsymbol{x} | \boldsymbol{U} \in \Re^{n\times r}\}$ (Du et al., 2020). An example is the two-layer ReLU neural network. The goal is to find the optimal representation φ^*, represented by $\boldsymbol{U}^*$ and the true linear predictors $\boldsymbol{b}_k^*$ for all tasks $k \in [q]$ to minimize the difference between the predicted and actual outputs. Arranging the m input features for task k as rows of an $m \times n$ matrix $\boldsymbol{X}_k$, and the outputs in an $m \times 1$ vector $\boldsymbol{y}_k$, we have the following model

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_q] := [\boldsymbol{X}_1\boldsymbol{U}^*\boldsymbol{b}_1^*, ..., \boldsymbol{X}_q\boldsymbol{U}^*\boldsymbol{b}_q^*] + \boldsymbol{V}$$

where $\boldsymbol{V}$ is the modeling error that is assumed to be i.i.d. zero mean Gaussian with variance σ_v^2. We have assumed an r-dimensional linear model for the regression coefficients, i.e., $\boldsymbol{\theta}_k^* = \boldsymbol{U}^*\boldsymbol{b}_k^*$, with $r \ll \min(n, q)$. In other words, the $n\times q$ regression coefficients' matrix $\boldsymbol{\Theta}^* = \boldsymbol{U}^*\boldsymbol{B}^*$ is rank r. Our goal is to learn the column span of the $n \times r$ matrix $\boldsymbol{U}^*$ (and in the process also learn $\boldsymbol{\Theta}^*$), from the $m \times q$ matrix $\boldsymbol{Y}$. We assume that all the feature vectors for all the tasks are i.i.d. standard Gaussian, i.e., all the $\boldsymbol{X}_k$s are i.i.d. and have i.i.d. standard Gaussian entries. Solving this problem requires solving

$$\min_{\substack{\tilde{\boldsymbol{U}}\in\Re^{n\times r}\\ \tilde{\boldsymbol{B}}\in\Re^{r\times q}}} \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{X}_k\tilde{\boldsymbol{U}}\tilde{\boldsymbol{b}}_k\|^2 \tag{1}$$

In the federated setting, we assume that there are a total of L nodes. Each observes a different disjoint subset ($\widetilde{m} =$

m/L) of rows of $\boldsymbol{Y}$. Denoting the set of rows observed at node ℓ by $\mathcal{S}_\ell$, this means that $\mathcal{S}_\ell$s are disjoint and $\cup_{\ell=1}^{L}\mathcal{S}_\ell = [q]$. At most τL nodes can be Byzantine with $\tau < 0.4$. The nodes can only communicate with the center.

In this work, all $\boldsymbol{U}$ matrices are $n \times r$ and are used to denote the subspaces spanned by their columns. We use $\|.\|$ to denote the (induced) ℓ_2 norm and $\|.\|_F$ for the Frobenius norm. For $\boldsymbol{U}_1, \boldsymbol{U}_2$ with orthonormal columns, we use $\boldsymbol{SD}_2(\boldsymbol{U}_1, \boldsymbol{U}_2) := \|(\boldsymbol{I} - \boldsymbol{U}_1\boldsymbol{U}_1{}^\top)\boldsymbol{U}_2\|$ or $\boldsymbol{SD}_F(\boldsymbol{U}_1, \boldsymbol{U}_2) := \|(\boldsymbol{I} - \boldsymbol{U}_1\boldsymbol{U}_1{}^\top)\boldsymbol{U}_2\|_F$ to quantify the Subspace Distance (SD). Clearly $\boldsymbol{SD}_F(\boldsymbol{U}_1, \boldsymbol{U}_2) \leq \sqrt{r}\boldsymbol{SD}_2(\boldsymbol{U}_1, \boldsymbol{U}_2)$.

2. Centralized Multi-task Representation Learning and Few Shot Learning

Below, we first give the AltGDmin algorithm from (Nayer & Vaswani, 2023, on arXiv since Feb. 2021) to learn $\boldsymbol{U}^*$. This is also similar to the FedRep algorithm of (Collins et al., 2021), with the difference only being that the AltGDmin initialization is better (has a better sample complexity). Next, we give details about few-shot learning.

2.1. Multi-task Linear Representation Learning

Recall that the goal is to minimize $f(\boldsymbol{U}, \boldsymbol{B}) := \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{X}_k\boldsymbol{U}\boldsymbol{b}_k\|^2$ where $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_q]$. AltGDmin (Nayer & Vaswani, 2023, on arXiv since Feb. 2021; Collins et al., 2021; Vaswani, 2024) proceeds as follows. We first initialize $\boldsymbol{U}$ as explained below; this is needed since the our optimization problem is clearly non-convex. After this, at each iteration, we alternatively update $\boldsymbol{U}$ and $\boldsymbol{B}$ as follows: (1) Keeping $\boldsymbol{U}$ fixed, update $\boldsymbol{B}$ by solving $\min_{\boldsymbol{B}} f(\boldsymbol{U}, \boldsymbol{B}) = \min_{\boldsymbol{B}} \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{X}_k\boldsymbol{U}\boldsymbol{b}_k\|^2$. (2) Keeping $\boldsymbol{B}$ fixed, update $\boldsymbol{U}$ by a GD step, followed by orthonormalizing its columns: $\boldsymbol{U}^+ \leftarrow QR(\boldsymbol{U} - \eta\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B})))$. Here $\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}) = \sum_{k\in[q]} \boldsymbol{X}_k^\top(\boldsymbol{X}_k\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)\boldsymbol{b}_k^\top$, η is the step-size for GD. We initialize $\boldsymbol{U}$ by (Nayer & Vaswani, 2023, on arXiv since Feb. 2021) computing the top r singular vectors of

$$\boldsymbol{\Theta}_0 := \sum_k \boldsymbol{X}_k^\top(\boldsymbol{y}_k)_{\text{trunc}}\boldsymbol{e}_k^\top, \ \boldsymbol{y}_{\text{trunc}} := (\boldsymbol{y} \circ \mathbb{1}_{|\boldsymbol{y}|\leq\sqrt{\alpha}})$$

Here $\alpha := 9\kappa^2\mu^2\sum_k \|\boldsymbol{y}_k\|^2/mq$. Here and below, $\boldsymbol{y}_{\text{trunc}}$ refers to a truncated version of the vector $\boldsymbol{y}$ obtained by zeroing out entries of $\boldsymbol{y}$ with magnitude larger than α (the notation $\mathbb{1}_{\boldsymbol{z}\leq\alpha}$ returns a 1-0 vector with 1 where $\boldsymbol{z}_j < \alpha$ and zero everywhere else, and $\boldsymbol{z}_1 \circ \boldsymbol{z}_2$ is the Hadamard product (.* operation in MATLAB)). The algorithm is summarized in Algorithm 1. We can show the following.

Theorem 2.1 ((Vaswani, 2024)). *Assume $\sigma_v^2 = 0$ and that $\max_k \|\boldsymbol{b}_k^*\| \leq \mu\sqrt{r/q}\sigma_1(\boldsymbol{\Theta}^*)$ for a constant $\mu \geq 1$ (incoherence of right singular vectors of $\boldsymbol{\Theta}^*$). Let κ denote the ratio of the first to the r-th singular value of $\boldsymbol{\Theta}^*$. Consider Algorithm 1 with $\eta = 0.4/m\sigma_1^{*2}$ and $T = C\kappa^2\log(1/\epsilon)$. If $mq \geq C\kappa^6\mu^2(n+q)r(\kappa^2 r + \log(1/\epsilon))$ and $m \geq C\max(\log n, \log q, r)\log(1/\epsilon)$, then, with probability (w.p.) at least $1 - n^{-10}$, $\text{SD}_2(\boldsymbol{U}, \boldsymbol{U}^*) \leq \epsilon$ and $\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^*\| \leq \epsilon\|\boldsymbol{\theta}_k^*\|$ for all $k \in [q]$.*

The time cost is $mqnr \cdot T = C\kappa^2 mqnr\log(1/\epsilon)$. The communication cost is nr per node per iteration.

This result shows that, as long as the total number of samples per task, m, is roughly order nr^2/q, the learning error decays exponentially with iterations even with a step-size η being a numerical constant (fast decay). Thus, after $T = C\kappa^2\log(1/\epsilon)$ iterations, $\boldsymbol{SD}(\boldsymbol{U}, \boldsymbol{U}^*) \leq \epsilon$, i.e. the low-dimensional subspace is accurately learned.

Treating κ, μ as numerical constants and assuming $n \approx q$, notice that the AltGDmin sample complexity is $mq \gtrsim nr\max(r, \log(1/\epsilon))$. On the other hand, FedRep (Collins et al., 2021) needs to assume $mq \gtrsim nr^2\max(r, \log(1/\epsilon))$ which is worse by a factor of r. In fact this complexity is comparable to that for the AltMin solution from (Nayer & Vaswani, 2021) that solved this problem and its LRPR generalization. The older result of (Nayer & Vaswani, 2023, on arXiv since Feb. 2021) for AltGDmin needed $mq \gtrsim nr^2\log(1/\epsilon)$. This is worse by a factor of $\max(1, r/\log(1/\epsilon))$.

The FedRep guarantee is worse because its initialization involves computing $\boldsymbol{U}_0$ as top r singular vectors of the matrix $\sum_{ki} \boldsymbol{y}_{ki}^2\boldsymbol{x}_{ki}\boldsymbol{x}_{ki}^\top\mathbb{1}(\boldsymbol{y}_{ki}^2 \leq (9\kappa^2\mu^2\sum_{ki}\boldsymbol{y}_{ki}^2/mq))$, and its analysis of the GD step is not as tight as can be (similar to that of (Nayer & Vaswani, 2023, on arXiv since Feb. 2021)). The advantage of the result of (Collins et al., 2021) was (i) a slightly better dependence κ, and (ii) it studied the low rank column-wise sensing problem in the $\sigma_v^2 \neq 0$ setting, while the result of (Vaswani, 2024) assumes $\sigma_v^2 = 0$. As we explain in the remark given next, this result can easily extend to the $\sigma_v^2 \neq 0$ setting as well with no change to its sample complexity.

Remark 2.2 (Theorem 2.1 with $\sigma_v^2 \neq 0$). Assume everything from Theorem 1 and that $0 < \sigma_v^2 \leq c\frac{\|\boldsymbol{\Theta}^*\|_F^2}{q}$. Let $\epsilon_{noise} := Cq\kappa^2\frac{\sigma_v^2}{\sigma_1^{*2}}$. Then, $\text{SD}_2(\boldsymbol{U}, \boldsymbol{U}^*) \leq \max(\epsilon, \epsilon_{noise})$. In words, the error decays exponentially until it reaches the (normalized) "noise-level", but saturates after that.

2.2. Few-Shot Learning

Few-shot learning refers to learning in data-scarce environments (Du et al., 2020). Once an estimate $\boldsymbol{U}$ for the true representation $\boldsymbol{U}^*$ is obtained, the problem simplifies to learning a predictor function $\boldsymbol{b}_k : \Re^r \to \Re$ defined on $\Re^r$

and specialized for each task k. Now, each source task can easily compute the local predictor $\boldsymbol{b}_k^*$ using the available samples, as $r \ll m$.

We want to bound the excess risk of the learned predictor on target task new. We are given m_{new} input, output training data pairs arranged into an $m \times n$ matrix $\boldsymbol{X}_{new}$, and an $m \times 1$ vector $\boldsymbol{y}_{new}$ and we need to solve the regression problem. However, this is data-scarce setting, i.e., $m_{new} \ll n$ and consequently without the low-dimensional linear representation, it is impossible to solve the regression problem. However, using the learned $\boldsymbol{U}$, we can easy learn an r-dimensional vector of regression coefficients as long as $m_{new} > r$. Excess risk on the learned predictor is given by $\|\boldsymbol{x}_{new}^\top \boldsymbol{\theta}_{new}^* + \boldsymbol{v}_{new} - \boldsymbol{x}_{new}^\top \boldsymbol{U}\boldsymbol{b}_{new}\|$, where $\boldsymbol{\theta}_{new}^* = \boldsymbol{U}^* \boldsymbol{b}_{new}^*$.

We compute $\boldsymbol{b}_{new}$ as $\boldsymbol{b}_{new} = (\boldsymbol{X}_{new}\boldsymbol{U})^\dagger \boldsymbol{y}_{new}$. Here, $\boldsymbol{U}$ is the final estimate from the AltGDmin algorithm described above. $\boldsymbol{M}^\dagger := (\boldsymbol{M}^\top \boldsymbol{M})^{-1}\boldsymbol{M}^\top$. We can prove the following for it.

We have the following bound on the expected value of the excess risk (ER) for the few-shot learning task. Recall from (Du et al., 2020) that $\mathbb{E}[ER(\boldsymbol{U}, \boldsymbol{b}_{new}] = \mathbb{E}[(y - \hat{y})^2]$ where $y = \boldsymbol{\theta}_{new}^{*\top} \boldsymbol{x} + v$ and we predict it as $\hat{y} = \boldsymbol{b}_{new}^\top \boldsymbol{U}^\top \boldsymbol{x}$ with $\boldsymbol{b}_{new}$ as given by the last step of Algorithm 1 and $\boldsymbol{U}$ is the output of its learning representation step.

Corollary 2.3. *Let $\boldsymbol{U}$ be the final output of the learning steps of Algorithm 1. If $m_{new} \geq C \max(r, \log q, \log n)$, then, the excess risk $\mathbb{E}[ER(\boldsymbol{U}, \boldsymbol{b}_{new}] = \|\boldsymbol{\theta}^* - \boldsymbol{U}\boldsymbol{b}_{new}\|^2 + \sigma_v^2 \leq C \max(\sigma_v^2, \epsilon \|\boldsymbol{b}_{new}^*\|^2)$.*

Notice that, with just order r samples, we are able to learn the regression coefficients for n-dimensional features.

Algorithm 1 Few-Shot Learning via altGDmin. Let $\boldsymbol{M}^\dagger := (\boldsymbol{M}^\top \boldsymbol{M})^{-1}\boldsymbol{M}^\top$.

Input: $\boldsymbol{y}_k, \boldsymbol{X}_k, k \in [q]$
Parameters: GD step size, η; Number of iterations, T
Sample-split: Partition the data into $2T + 1$ equal-sized disjoint sets: $\boldsymbol{y}_k^{(\tau)}, \boldsymbol{X}_k^{(\tau)}, \tau = 0, 1, \ldots 2T$.

Learning Representation:
Initialization:
set $\alpha \leftarrow 9\kappa^2\mu^2 \frac{1}{mq} \sum_{ki} \left|\boldsymbol{y}_{ki}\right|^2$,
Using $\boldsymbol{y}_k \equiv \boldsymbol{y}_k^{(0)}, \boldsymbol{X}_k \equiv \boldsymbol{X}_k^{(0)}$,
set $\boldsymbol{y}_{k,trunc}(\alpha) \leftarrow \boldsymbol{y}_{k,trnc} := \text{trunc}(\boldsymbol{y}_k, \alpha)$,
set $\boldsymbol{\Theta}_0 \leftarrow (1/m) \sum_{k \in [q]} \boldsymbol{X}_k^\top \boldsymbol{y}_{k,trunc}(\alpha) \boldsymbol{e}_k^\top$
set $\boldsymbol{U}_0 \leftarrow$ top-r-singular-vectors of $\boldsymbol{\Theta}_0$
GDmin iterations:
for $t = 1$ **to** T **do**
  Let $\boldsymbol{U} \leftarrow \boldsymbol{U}_{t-1}$.
  Using $\boldsymbol{y}_k \equiv \boldsymbol{y}_k^{(t)}, \boldsymbol{X}_k \equiv \boldsymbol{X}_k^{(t)}$,
  set $\boldsymbol{b}_k \leftarrow (\boldsymbol{X}_k \boldsymbol{U})^\dagger \boldsymbol{y}_k$, $\boldsymbol{\theta}_k \leftarrow \boldsymbol{U}\boldsymbol{b}_k$ for all $k \in [q]$
  Using $\boldsymbol{y}_k \equiv \boldsymbol{y}_k^{(T+t)}, \boldsymbol{X}_k \equiv \boldsymbol{X}_k^{(T+t)}$, compute
  set $\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}) = \sum_k \boldsymbol{X}_k^\top (\boldsymbol{X}_k \boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)\boldsymbol{b}_k^\top$
  set $\hat{\boldsymbol{U}}^+ \leftarrow \boldsymbol{U} - (\eta/m)\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}_t)$.
  compute $\hat{\boldsymbol{U}}^+ \overset{\text{QR}}{=} \boldsymbol{U}^+ \boldsymbol{R}^+$.
  Set $\boldsymbol{U}_t \leftarrow \boldsymbol{U}^+$.
end for

Few-shot Learning: Prediction on new **source**
$\boldsymbol{b}_{new} \leftarrow (\boldsymbol{X}_{new}\boldsymbol{U})^\dagger \boldsymbol{y}_{new}$
$\boldsymbol{\theta}_{new} \leftarrow \boldsymbol{U}\boldsymbol{b}_{new}$

3. Resilient Federated Multi-Task and Few-Shot Learning

Recall the federated setting problem from Sec. 1.3: there a total of L federated nodes and we assume that at most τL of them may be Byzantine with $\tau < 0.4$. Denote the set of good (non-Byzantine) nodes by $\mathcal{J}_{good}$. Equivalently, this means that $|\mathcal{J}_{good}| > (1 - \tau)L$.

We develop a solution approach for making AltGDmin Byzantine resilient that relies on the geometric median (GM). The most challenging part in doing this is modifying the initialization step. For the rest of the algorithm, we can borrow ideas from the existing extensive literature on Byzantine resilient GD discussed earlier. One popular approach in this area is to replace the summation in the gradient computation step by a "median" for vector-valued quantities. A well-studied one is the geometric median (GM) (Minsker, 2015; Chen et al., 2017), which we will use. The minimization step for update of columns of $\boldsymbol{B}$ can be done locally at the nodes. These are also used only in the local partial gradient computation and hence never need to be transmitted to the center. We should mention though that the analysis of the GD step is not a direct extension of existing ideas because of the important differences between our problem and most standard problems. In our problem, the GD step is not a standard GD or projected GD step for a given cost function.

For L data vectors, $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L$, the geometric median (GM) is defined as $\boldsymbol{z}_{gm} = \min_{\boldsymbol{z}} \sum_{\ell=1}^{L} \|\boldsymbol{z}_\ell - \boldsymbol{z}\|$. Here and below, $\|.\|$ with a subscript denotes the l_2 norm. The GM cannot be computed in closed form but various algorithms exist to accurately approximate it.

3.1. GM-based Resilient Spectral Initialization: Subspace Median and Subspace Median of Means

This consists of two steps. First a resilient estimate of the truncation threshold $\alpha = \frac{\tilde{C}}{mq} \sum_k \sum_i y_{ki}^2$ needs to be computed. For this, we use the scalar median of means of the partial estimates computed by each node. Next, we need to

compute $\boldsymbol{U}_0$ which is the matrix of top r left singular vectors of $\boldsymbol{\Theta}_0$. Node ℓ has data to compute the $n \times q$ matrix $(\boldsymbol{\Theta}_0)_\ell$, defined as

$$(\boldsymbol{\Theta}_0)_\ell := \sum_{k=1}^{q} (\boldsymbol{X}_k)_\ell{}^\top ((\boldsymbol{y}_k)_\ell)_{\text{trunc}} \boldsymbol{e}_k^\top, \tag{2}$$

Observe that $\boldsymbol{\Theta}_0 = \sum_\ell (\boldsymbol{\Theta}_0)_\ell / L$. If all nodes were good, we would use this fact to implement the federated power method (PM) for this case: starting with a random initialization $\boldsymbol{U}$, this involves iterating the following: compute $\boldsymbol{V} := \sum_\ell \boldsymbol{V}_\ell / L$ (where $\boldsymbol{V}_\ell = (\boldsymbol{\Theta}_0)_\ell{}^\top \boldsymbol{U}$) in a federated fashion, followed by computing $\tilde{\boldsymbol{U}}^+ = \sum_\ell (\boldsymbol{\Theta}_0)_\ell \boldsymbol{V}/L$ in a federated fashion, and then obtain $\boldsymbol{U}^+ = QR(\tilde{\boldsymbol{U}}^+)$ at the center. To deal with Byzantine attacks, the most obvious solution is to replace the averaging at the center by the GM. However, this works with high probability only if all the $(\boldsymbol{\Theta}_0)_\ell$'s are extremely accurate estimates of $\boldsymbol{\Theta}^*$ [1]. This further implies that its required sample complexity is very large. We provide a detailed discussion of this fact for the simpler PCA problem in Sec. 4 and Table 1.

Subspace-Median. Since the GM is defined for quantities whose distance can be measured using the vector l_2 norm (equivalently, matrix Frobenius norm), it cannot be directly used for subspaces (or their basis matrices): these do not lie in on a Euclidean space (but instead on the Stiefel manifold). To understand this simply, notice that $\boldsymbol{U}, -\boldsymbol{U}$ specify the same subspace even though $\|\boldsymbol{U} - (-\boldsymbol{U})\|_F = 2\sqrt{r} \neq 0$. Notice though that the Frobenius norm between the projection matrices of two subspaces is also a measure of subspace distance: $\|\mathcal{P}_{\boldsymbol{U}} - \mathcal{P}_{\boldsymbol{U}^*}\|_F = \sqrt{2}\boldsymbol{SD}_F(\boldsymbol{U}, \boldsymbol{U}^*)$ (Chen, Chi, Fan, Ma, et al., 2021, Lemma 2.5). Here $\mathcal{P}_{\boldsymbol{U}} := \boldsymbol{U}\boldsymbol{U}^\top$ is the projection matrix for subspace $\boldsymbol{U}$ (assumes $\boldsymbol{U}$ has orthonormal columns). We use this idea to develop a simple but useful approach called the "Subspace Median": Node ℓ computes $\hat{\boldsymbol{U}}_\ell$ as the top r singular vectors of the matrix $(\boldsymbol{\Theta}_0)_\ell$ that it has data for, and sends it to the center. If node ℓ is good, then $\hat{\boldsymbol{U}}_\ell$ already has orthonormal columns; however if the node is Byzantine, then it is not. The center first orthonormalizes the columns of all the received $\hat{\boldsymbol{U}}_\ell$: $\boldsymbol{U}_\ell = QR(\hat{\boldsymbol{U}}_\ell)$ for all $\ell \in [L]$. It then computes the projection matrices $\mathcal{P}_{\boldsymbol{U}_\ell} := \boldsymbol{U}_\ell \boldsymbol{U}_\ell^\top$, $\ell \in [L]$, followed by vectorizing them, computing their GM, and then converting the GM into a matrix. Denote this by $\mathcal{P}_{gm}$. Finally, the center finds the ℓ for which $\mathcal{P}_{\boldsymbol{U}_\ell}$ is closest to $\mathcal{P}_{gm}$ in Frobenius norm and outputs the corresponding $\boldsymbol{U}_\ell$. Denote this $\boldsymbol{U}_\ell$ by $\boldsymbol{U}_{out}$ We can show the following for this estimator

Lemma 3.1. *(Subspace Median) Suppose that $|\mathcal{J}_{good}| \geq (1-\tau)L$ for a $\tau < 0.4$. If $\min_{\ell \in \mathcal{J}_{good}} \Pr(\text{SD}_F(\boldsymbol{U}_\ell, \boldsymbol{U}^*) \leq \delta) \geq 1 - p$. Then, with probability at least $1 - c_0 - \exp(L\psi(0.4 - \tau, p))$, $\text{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq 23\delta$.*

Here $\psi(a, b) := (1-a)\log\frac{1-a}{1-b} + a\log\frac{a}{b}$ is the binary KL divergence.

Subspace Median of Means. A median-based estimator can be robust to almost 50% outliers (here Byzantine attacks), but, as is well known, the use of median also wastes samples. In our context, this means that the estimate of each node needs to be accurate enough. If the maximum number of Byzantine nodes is known to be much lesser than 50%, a better approach is to use the median of means (MoM) estimator. We explain how to develop this for our problem. For a parameter $\tilde{L} \leq L$, we would like to form $\tilde{L}$ mini-batches of $\rho = L/\tilde{L}$ nodes; w.l.o.g. ρ is an integer. For the ℓ-th node in the ϑ-th mini-batch we use the short form notation $(\vartheta, \ell) = (\vartheta - 1)\rho + \ell$, for $\ell \in [\rho]$.

In our setting, combining samples means combining the rows of $(\boldsymbol{X}_k)_\ell$ and $(\boldsymbol{y}_k)_\ell$ for ρ nodes to obtain $(\boldsymbol{\Theta}_0)_{(\vartheta)}$ with k-th column given by $\sum_{\ell=1}^{\rho} (\boldsymbol{X}_k)_{(\vartheta,\ell)}^\top (\boldsymbol{y}_{k,\text{trunc}})_{(\vartheta,\ell)}/\rho$. To compute this in a communication-efficient and private fashion, we use a federated power method for each of the $\tilde{L}$ mini-batches. The output of each of these power methods is $\boldsymbol{U}_{(\vartheta)}$, $\vartheta \in [\tilde{L}]$. Then we do subspace-median on $\boldsymbol{U}_{(\vartheta)}$, $\vartheta \in [\tilde{L}]$ to obtain the final subspace estimate $\boldsymbol{U}_{out}$. To explain the federation details simply, we explain them for $\vartheta = 1$. The power method needs to federate $\boldsymbol{U} \leftarrow QR((\boldsymbol{\Theta}_0)_{(1)}(\boldsymbol{\Theta}_0)_{(1)}^\top \boldsymbol{U}) = QR(\sum_{\ell'=1}^{\rho} (\boldsymbol{\Theta}_0)_{\ell'} (\sum_{\ell=1}^{\rho} (\boldsymbol{\Theta}_0)_\ell^\top \boldsymbol{U}))$. This needs two steps of information exchange between the nodes and center at each power method iteration. In the first step, we compute $\boldsymbol{V} = \sum_{\ell \in [\rho]} (\boldsymbol{\Theta}_0)_\ell{}^\top \boldsymbol{U}$, and in the second one we compute $\tilde{\boldsymbol{U}} = \sum_{\ell \in [\rho]} (\boldsymbol{\Theta}_0)_\ell \boldsymbol{V}$, followed by its QR decomposition.

We summarize the complete algorithm in Algorithm 2.

Guarantee. We can prove the following. It needs to assume that the same set of τL nodes are Byzantine for all the power method iterations needed for the initialization step[2].

Theorem 3.2 (Initialization via Subs-MoM)**.** *Assume $\sigma_v^2 = 0$ and that $\max_k \|\boldsymbol{b}_k^*\| \leq \mu\sqrt{r/q}\sigma_1(\boldsymbol{\Theta}^*)$ for a constant $\mu \geq 1$. Consider Algorithm 2 with $T_{gm} =$*

[1] The reason for this is that it computes the GM of the node outputs $\boldsymbol{V}_\ell = (\boldsymbol{\Theta}_0)_\ell^\top \boldsymbol{U}$ at each iteration including the first one. At the first iteration, $\boldsymbol{U}$ is a randomly generated matrix and thus, w.h.p., this is a bad approximation of the desired subspace $span(\boldsymbol{U}^*)$. Consequently, unless the various $(\boldsymbol{\Theta}_0)_\ell$'s are very close approximations of $\boldsymbol{\Theta}^*$, the different $\boldsymbol{V}_\ell$'s are likely to be bad approximations of $span(\boldsymbol{B}^*)$. In particular, this means that the estimates at the different nodes may be quite different even for all the good nodes. As a result, their GM is unable to distinguish between the good and Byzantine ones, and, there is a good chance it approximates the Byzantine one(s). A similar argument can be repeated for $\tilde{\boldsymbol{U}}_\ell$s and so on. Thus, unless all the $(\boldsymbol{\Theta}_0)_\ell$'s are very close approximations of $\boldsymbol{\Theta}^*$ (and hence very similar), there is a good chance that the subspace estimates do not improve over iterations.

[2] This can be relaxed if we instead assume that a much tighter bound on the number of bad nodes per iteration.

$C\log(\frac{\tilde{L}r}{\delta_0}), T_{pow} = C\kappa^2\log(\frac{n}{\delta_0})$. *Assume that the set of Byzantine nodes remains fixed for all iterations in this algorithm and is of size at most τL with $\tau < 0.4\tilde{L}/L$. If $\widetilde{m}q \geq C\frac{\tilde{L}}{L}\cdot\kappa^6\mu^2(n+q)r^2/\delta_0^2$, then*

Then, w.p. at least $1 - c_0 - \exp(-\tilde{L}\psi(0.4-\tau, n^{-10} + \exp(-c(n+q)))) - L\exp(-\tilde{c}\widetilde{m}q\delta_0^2/r^2\kappa^4)$

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{out}) \leq \delta_0$$

The communication cost per node is order $nr\log(\frac{n}{\delta_0})$. The computational cost at any node is order $nqr\log(\frac{n}{\delta_0})$ while that at the center is $n^2\tilde{L}\log^3(\tilde{L}r/\delta_0)$.

The extension of the above result for the $\sigma_v^2 \neq 0$ case will be straightforward and can be proved using the same ideas as those used for Remark 2.2.

Proof. This follows by using Lemma 3.1 along with the Davis Kahan $\sin\Theta$ theorem and concentration bounds from (Vershynin, 2018) applied to analyze the output of each node. We apply the latter two to $\boldsymbol{\Phi}_{(\vartheta)} = \sum_{\ell=1}^{\rho}(\boldsymbol{\Theta}_0)_{(\vartheta,\ell)}(\boldsymbol{\Theta}_0)_{(\vartheta,\ell)}^\top/\rho$ and $\boldsymbol{\Phi}^* = \mathbb{E}[(\boldsymbol{\Theta}_0)_\ell|\alpha]\mathbb{E}[(\boldsymbol{\Theta}_0)_\ell|\alpha]^\top$ for $\vartheta \in [\tilde{L}]$. □

3.2. GM-based Resilient Federated GDmin Iterations

We can make the altGDmin iterations resilient as follows. In the minimization step, each node computes its own estimate $(\boldsymbol{b}_k)_\ell$ of $\boldsymbol{b}_k^*$ as follows:

$$(\boldsymbol{b}_k)_\ell = ((\boldsymbol{X}_k)_\ell\boldsymbol{U})^\dagger(\boldsymbol{y}_k)_\ell,\ k \in [q]$$

Each node then uses this to compute its estimate of the gradient w.r.t. $\boldsymbol{U}$ as $\nabla f_\ell = \sum_{k\in\boldsymbol{S}_\ell}(\boldsymbol{X}_k)_\ell^\top((\boldsymbol{X}_k)_\ell\boldsymbol{U}(\boldsymbol{b}_k)_\ell - (\boldsymbol{y}_k)_\ell)(\boldsymbol{b}_k)_\ell^\top$. The center receives the gradients from the different nodes, computes their GM and uses this for the projected GD step. Since the gradient norms are not bounded, the GM computation needs to be preceded by the thresholding step.

To improve sample complexity (while reducing Byzantine tolerance), we can replace GM of the gradients by their GM of means: form $\tilde{L}$ batches of size $\rho = L/\tilde{L}$ each, compute the mean gradient within each batch, compute the GM of the $\tilde{L}$ mean gradients. Use appropriate scaling. We summarize the GMoM algorithm in Algorithm 3. The GM case corresponds to $\tilde{L} = L$. Given a good enough initialization, a small enough fraction of Byzantine nodes, enough samples $\widetilde{m}q$ at each node at each iteration, we can prove the following for the GD iterations.

Lemma 3.3. *(AltGDmin-SubsMoM: Error Decay) Consider Algorithm 3 with sample-splitting, and with step-size $\eta \leq 0.5/\sigma_1^{*2}$. If, at each iteration t, $\widetilde{m}q \geq C_1\kappa^4\mu^2(n+r)r^2(\tilde{L}/L)$, $\widetilde{m} > C_2\max(\log q, \log n)$; if $\tau < 0.4\tilde{L}/L$; and if the initial estimate $\boldsymbol{U}_0$ satisfies $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_0) \leq \delta_0 = 0.1/\kappa^2$, then w.p. at least $1 - c_0 - t\big[Ln^{-10} + \exp(-L\psi(0.4-\tau, n^{-10}))\big]$, $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{t+1}) \leq \delta_{t+1} := \big(1 - (\eta\sigma_1^{**2})\frac{0.12}{\kappa^2}\big)^{t+1}\delta_0$*

Algorithm 2 Byz-AltGDmin-Learn: Initialization step.

Input: Batch ϑ : $\{(\boldsymbol{X}_k)_\ell, \boldsymbol{Y}_\ell, k \in [q]\}, \ell \in [L]$
Parameters: T_{pow}, T_{gm},
Nodes $\ell = 1, ..., L$
Compute $\alpha_\ell \leftarrow \frac{\tilde{C}}{\widetilde{m}q}\sum_k \|(\boldsymbol{y}_k)_\ell\|^2$, with $\tilde{C} = 9\kappa^2\mu^2$.
Central Server
$\alpha \leftarrow \text{Median}\{\alpha_{(\vartheta)}\}_{\vartheta=1}^{\tilde{L}}$, where $\alpha_{(\vartheta)} = \sum_{\ell=1}^{\rho}\alpha_{(\vartheta,\ell)}/\rho$
Central Server
Let $\boldsymbol{U}_0 = \boldsymbol{U}_{rand}$ where $\boldsymbol{U}_{rand}$ is an $n \times r$ matrix with i.i.d standard Gaussian entries.
for $\tau \in [T_{pow}]$ **do**
  Nodes $\ell = 1, ..., L$
  Compute $\boldsymbol{V}_\ell \leftarrow (\boldsymbol{\Theta}_0)_\ell^\top(\boldsymbol{U}_{(\vartheta)})_{\tau-1}$ for $\ell \in (\vartheta-1)\rho + [\rho]$, $\vartheta \in [\tilde{L}]$. Push to center.
  Central Server
  Compute $\boldsymbol{V}_{(\vartheta)} \leftarrow \sum_{\ell=1}^{\rho}\boldsymbol{V}_{(\vartheta-1)\rho+\ell}$
  Push $\boldsymbol{V}_{(\vartheta)}$ to nodes $\ell \in (\vartheta-1)\rho + [\rho]$.
  Nodes $\ell = 1, ..., L$
  Compute $\boldsymbol{U}_\ell \leftarrow \sum_k(\boldsymbol{\Theta}_0)_\ell\boldsymbol{V}_{(\vartheta)}$ for $\ell \in (\vartheta-1)\rho + [\rho]$, $\vartheta \in [\tilde{L}]$. Push to center.
  Central Server
  Compute $\boldsymbol{U}_{(\vartheta)} \leftarrow QR(\sum_{\ell=1}^{\rho}\boldsymbol{U}_{(\vartheta-1)\rho+\ell})$
  Let $(\boldsymbol{U}_{(\vartheta)})_\tau \leftarrow \boldsymbol{U}_{(\vartheta)}$. Push to nodes $\ell \in (\vartheta-1)\rho + [\rho]$.
end for
Central Server (implements Subspace Median on $\boldsymbol{U}_{(\vartheta)}, \vartheta \in [\tilde{L}]$)
Orthonormalize: $\boldsymbol{U}_\vartheta \leftarrow QR((U_\vartheta)_0), \vartheta \in [\rho]$
Compute $\mathcal{P}_{\boldsymbol{U}_\vartheta} \leftarrow \boldsymbol{U}_\vartheta\boldsymbol{U}_\vartheta^\top, \vartheta \in [\rho]$
Compute GM: $\mathcal{P}_{gm} \leftarrow \text{approxGM}\{\mathcal{P}_{\boldsymbol{U}_\vartheta}, \vartheta \in [\rho]\}$ (Use (Cohen, Lee, Miller, Pachocki, & Sidford, 2016, Algorithm 1) with parameter T_{gm}).
Find $\vartheta_{best} = \arg\min_\vartheta \|\mathcal{P}_{\boldsymbol{U}_\vartheta} - \mathcal{P}_{gm}\|_F$
Output $\boldsymbol{U}_{out} = \boldsymbol{U}_{\vartheta_{best}}$

We prove this lemma in the long version (Singh & Vaswani, 2024, Section V). The complete algorithm is obtained by using Algorithm 3 initialized using Algorithm 2 with sample-splitting. Combining Theorem 3.2 and Lemma 3.3, and setting $\eta = 0.5/\sigma_1^{*2}$ and $\delta_0 = 0.1/\kappa^2$, we can show that, at iteration $t+1$, $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{t+1}) \leq \delta_{t+1} = (1 - 0.06/\kappa^2)^{t+1}0.1/\kappa^2$ whp. Thus, in order for this to be $\leq \epsilon$, we need to set $T = C\kappa^2\log(1/\epsilon)$. Also, since we are using fresh samples at each iteration (sample-splitting), this also means that our sample complexity needs to be multiplied by T. We have the following final result.

Algorithm 3 Byz-AltGDMin-Learn: Complete algorithm

Obtain $\boldsymbol{U}_0$ using Algorithm 2.
for $t = 1$ to T **do**
  Nodes $\ell = 1, ..., L$
  Set $\boldsymbol{U} \leftarrow \boldsymbol{U}_{t-1}$
  $(\boldsymbol{b}_k)_\ell \leftarrow ((\boldsymbol{X}_k)_\ell \boldsymbol{U})^\dagger (\boldsymbol{y}_k)_\ell,\ \forall\, k \in [q]$
  $(\boldsymbol{\theta}_k)_\ell \leftarrow \boldsymbol{U}(\boldsymbol{b}_k)_\ell,\ \forall\, k \in [q]$
  $(\nabla f_k)_\ell \leftarrow \sum_{k\in[q]} (\boldsymbol{X}_k)_\ell^\top ((\boldsymbol{X}_k)_\ell \boldsymbol{U}(\boldsymbol{b}_k)_\ell - (\boldsymbol{y}_k)_\ell)(\boldsymbol{b}_k)_\ell{}^\top,\ \forall\, k \in [q]$
  Push $\nabla f_\ell \leftarrow \sum_{k\in[q]} (\nabla_{\boldsymbol{U}} f_k)_\ell$
  Central Server
  Compute $\nabla f_{(\vartheta)} \leftarrow \sum_{\ell\in\vartheta} \nabla f_\ell$
  $\nabla f^{GM} \leftarrow approxGMthresh(\nabla f_{(\vartheta)}, \vartheta = 1, 2, \ldots \tilde{L})$.
  (Use (Cohen et al., 2016, Algorithm 1) with T_{gm} iterations on $\{\nabla f_{(\vartheta)},\ \vartheta \in [\tilde{L}] \setminus \{\ell : \|\nabla f_{(\vartheta)}\| > \omega\}\}$)
  Compute $\boldsymbol{U}^+ \leftarrow QR(\boldsymbol{U}_{t-1} - \frac{\eta}{\rho\tilde{m}} \nabla f^{GM})$
  return Set $\boldsymbol{U}_t \leftarrow \boldsymbol{U}^+$. Push $\boldsymbol{U}_t$ to nodes.
end for

Theorem 3.4. *(AltGDmin-SubsMoM: Complete guarantee) Assume $\sigma_v^2 = 0$ and that $\max_k \|\boldsymbol{b}_k^*\| \le \mu\sqrt{r/q}\sigma_1(\boldsymbol{\Theta}^*)$ for a constant $\mu \ge 1$. Consider Algorithm 3 and the setting of Theorem 3.2 and Lemma 3.3. Set $T = C\kappa^2 \log(1/\epsilon)$. If $\tilde{m}q \ge C\kappa^4\mu^2(n+q)r^2\log(1/\epsilon)(\tilde{L}/L)$ and $\tilde{m} > C\kappa^2 \max(\log q, \log n)\log(1/\epsilon)$, then, w.p. at least $1 - TLn^{-10}$, $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}) \le \epsilon$, and $\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^*\| \le \epsilon\|\boldsymbol{\theta}_k^*\|$ for all $k \in [q]$. The communication cost per node is order $nr\log(\frac{n}{\epsilon})$. The computational cost at any node is order $nqr\log(\frac{n}{\epsilon})$ while that at the center is $n^2\tilde{L}\log^3(\tilde{L}r/\epsilon)$.*

The extension of the above result for the $\sigma_v^2 \neq 0$ case will be straightforward and can be proved using the same ideas as those used for Remark 2.2.

3.3. Numerical Experiments

In the Figure 1 we plot Error vs Iteration where $Error = \frac{\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U})}{\sqrt{r}}$. *We report mean $\boldsymbol{SD}_F$ over 100 Monte Carlo runs.* We compare Byz-Fed-AltGDmin-Learn (GMoM) with the baseline algorithm - AltGDmin-Learn (Mean) in the no attack setting. We also provide results for Byz-Fed-AltGDmin-Learn (GM) for both values of L_{byz}. All these are compared in Figure 1. We also compare the initialization errors in Figure 1 Table. As can be seen Byz-Fed-AltGDmin-Learn (GMoM) based initialization error is quite a bit lower than that with Byz-Fed-AltGDmin-Learn (GM). The same is true for the GDmin iterations.

Method	$L_{byz} = 1$	$L_{byz} = 2$
Byz-Fed-AltGDmin-Learn (GM)	0.716(0.665)	0.717(0.667)
Byz-Fed-AltGDmin-Learn (GMoM)	0.477(0.457)	0.475(0.459)

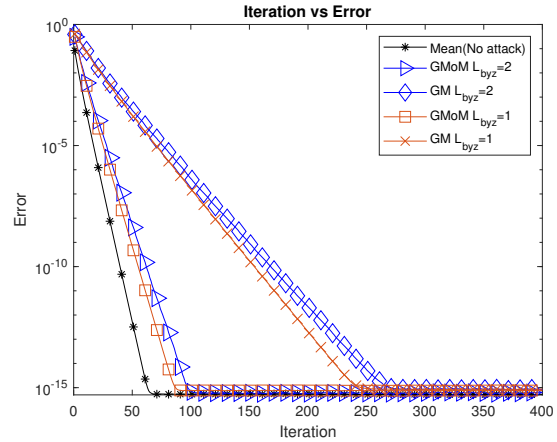

Figure 1: *Table:* Initialization errors. We report "$\max \boldsymbol{SD}_F(\text{mean}\boldsymbol{SD}_F)$" in each column. *Figure:* Byz-Fed-AltGDmin-Learn (GMoM), AltGDmin-Learn (Mean), Byz-Fed-AltGDmin-Learn (GM) for $L_{byz} = 1, 2$; $L = 18$.

4. Resilient Federated PCA

Given q data vectors $\boldsymbol{d}_k \in \Re^n$, that are *zero mean*, mutually independent, sub-Gaussian, and have covariance matrices that share the same principal subspace, the goal is to find this subspace. We can arrange the data vectors into an $n \times q$ matrix, $\boldsymbol{D} := [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots \boldsymbol{d}_q]$. The data is vertically federated, this means that each node ℓ has $q_\ell = \tilde{q} = \frac{q}{L}$ $\boldsymbol{d}_k$'s. Denote the corresponding sub-matrix of $\boldsymbol{D}$ by $\boldsymbol{D}_\ell$. Suppose that $\boldsymbol{d}_k$ has covariance matrix $\boldsymbol{\Sigma}_k^*$ of the form $\boldsymbol{\Sigma}_k^* \overset{\text{EVD}}{=} [\boldsymbol{U}^*, \boldsymbol{U}_{\perp,k}^*]\boldsymbol{S}_k[\boldsymbol{U}^*, \boldsymbol{U}_{\perp,k}^*]^\top$: all the covariance matrices share the same principal subspace $\boldsymbol{U}^*$, but the lower eigenvectors and all eigenvalues can be different. We use K to denote the maximum sub-Gaussian norm (Vershynin, 2018, Chap 2) of $\boldsymbol{\Sigma}_k^{*-1/2}\boldsymbol{d}_k$ for any $k \in [q]$. The goal is to obtain a resilient estimate of the r-dimensional subspace $\boldsymbol{U}^*$ of $\Re^n$ in a federated setting.

The subspace median idea developed for initializing the AltGDmin algorithm described earlier is in fact much more generally applicable for a generic subspace learning meta-problem: given L subspace estimates $\boldsymbol{U}_\ell$ of an unknown subspace $\boldsymbol{U}^*$, one can compute their subspace median using the exact same idea as that given in Sec. 3.1. For PCA, the individual node subspace estimates $\boldsymbol{U}_\ell$ are computed as the top r singular vectors of the data matrix $\boldsymbol{D}_\ell$.

Moreover, we can also develop and analyze a subspace median of means generalization of it well. This requires some different ideas described next because, for the current problem, we are assuming vertical federation. Pick an integer $\tilde{L} \le L$. In order to implement the "mean"

Methods→	**SVD-ResCovEst** (Minsker, 2015, Cor 4.3)	**ResPowMeth** Modification of (Minsker, 2015; Hardt & Price, 2014)	**SubsMed (Proposed)**	**PowMeth, no attack (baseline)**
Sample Comp for PCA (lower bound on q)	$\frac{n^2 L}{\epsilon^2}$	$\max\left(n^2 r^2, \frac{n}{\epsilon^2}\right) \cdot L$	$\frac{nrL}{\epsilon^2}$	$\frac{nr}{\epsilon^2}$
Communic Cost	n^2	$nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	nr	$nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$
Compute Cost - node	$n^2 q_\ell$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$
Compute Cost - center	$n^2 L\log^3\left(\frac{Ln}{\epsilon}\right)$	$nrL\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})\log^3\left(\frac{Ln}{\epsilon}\right)$	$n^2 L\log^3\left(\frac{Ln}{\epsilon}\right)$	$nrL\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$

Table 1: *Comparisons for solving the resilient federated PCA problem (Sec. 4). We compare the proposed Subspace Median (SubsMed) algorithm with the two obvious (but bad) solutions – SVD-Resilient Covariance Estimation (SVD-ResCovEst): SVD on GM of Covariance matrices, and Resilient Power Method (ResPowMeth): GM based modification of the power method – and with the baseline (power method for a no-attack setting). Observe that SubsMed needs the smallest sample complexity and has the lowest communication cost.*

step, we need to combine samples from $\rho = L/\tilde{L}$ nodes, i.e., we need to find the r-SVD of matrices $\boldsymbol{D}_{(\vartheta)} = [\boldsymbol{D}_{(\vartheta,1)}, \boldsymbol{D}_{(\vartheta,2)}, \ldots, \boldsymbol{D}_{(\vartheta,\rho)}]$, for all $\vartheta \in [\tilde{L}]$; we are using the notation $(\vartheta, \ell) = (\vartheta - 1)\rho + \ell$. This needs to be done without sharing the entire data matrix. We do this by implementing $\tilde{L}$ different federated power methods, each of which combines samples from a different minibatch of ρ nodes. The output of this step is $\tilde{L}$ subspace estimates $\boldsymbol{U}_{(\vartheta)}$, $\vartheta \in [\tilde{L}]$. These serve as inputs to a basic Subspace-Median algorithm to obtain the final Subspace-MoM estimator. $\tilde{L} = L$ is its subspace median special case.

Theorem 4.1 (Resilient Federated PCA)**.** *Consider Subspace Median of Means. For a $\Delta > 0$, assume that $\min_\ell((\sigma_r^*)_\ell - (\sigma_{r+1}^*)_\ell) \geq \Delta$. Here $\boldsymbol{\Sigma}_\ell^* = \frac{1}{\tilde{q}}\sum_{k\in\boldsymbol{S}_\ell}\boldsymbol{\Sigma}_k^*$. Assume that the set of Byzantine nodes remains fixed for all iterations in this algorithm and the size of this set is at most τL with $\tau < 0.4\tilde{L}/L$. If*

$$q \geq CK^4\frac{\sigma_1^{*2}}{\Delta^2}\frac{nr}{\epsilon^2}\cdot\tilde{L}$$

then, then w.p. at least $1 - c_0 - \exp(-L\psi(0.4 - \tau, 2\exp(-n)))$, $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq \epsilon$. The communication cost is $T_{pow}nr = nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$ per node. The computational cost at the center is order $n^2\tilde{L}\log^3\left(\frac{\tilde{L}r}{\epsilon}\right)$. The computational cost at any node is order $nq_\ell rT_{pow}$.

Comparison with attack-free federated PCA. Observe that the total sample complexity (lower bound on q) needed by the above result to guarantee $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}) \leq \epsilon$ is order $nr\tilde{L}/\epsilon^2$. Here we are quantifying subspace distance using $\boldsymbol{SD}_F$. However, even if we use the more common distance measure $\boldsymbol{SD}_2(\boldsymbol{U}^*, \boldsymbol{U}) := \|(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{U}^*\|$ and require just $\boldsymbol{SD}_2(\boldsymbol{U}^*, \boldsymbol{U}) \leq \epsilon$, this is the required sample complexity. The reason is we need Frobenius norm is for the GM computation. On the other hand, standard attack-free PCA needs a sample complexity of only n/ϵ^2 to guarantee $\boldsymbol{SD}_2(\boldsymbol{U}^*, \boldsymbol{U}) \leq \epsilon$ (Vershynin, 2018, Remark 4.7.2). Our complexity also has an extra factor of $\tilde{L}$; this is because we are computing the individual node estimates using $\tilde{q} = q/L$ data points and we need each of the node estimates to be accurate (to ensure that their "median" is accurate). This extra factor is needed also in other work that uses (geometric) median, e.g., in (Chen et al., 2017).

Two more obvious solutions for Resilient PCA and why they fail. Consider the symmetric matrix $\boldsymbol{\Phi}_\ell := (\boldsymbol{\Theta}_0)_\ell(\boldsymbol{\Theta}_0)_\ell^\top$. In a centralized setting, the most obvious solution to the above problem would be to compute the GM of the vectorized matrices $\boldsymbol{\Phi}_\ell$ followed by obtaining the principal subspace (r-SVD) of the GM matrix; this was studied in (Minsker, 2015). However, in a federated setting, this is communication inefficient because it requires each node to share an $n \times n$ matrix. For the same reason it is not private either. Moreover, this is extremely sample inefficient; see Table 1. For a communication-efficient solution, in the attack-free federated setting, one would use the distributed power method (Golub & Van Loan, 1989; Wu, Wai, Li, & Scaglione, 2018). A direct modification of this to deal with attacks is to use its GM based modification: at each iteration, instead of summing the $n \times r$ matrices, $\tilde{\boldsymbol{U}}_\ell := (\boldsymbol{\Phi}_\ell\boldsymbol{U})$ received from each node, we compute the GM of their vectorized versions. We refer to this as *Resilient Power Method (ResPowMeth)*. However, this works w.h.p. only if all the $\boldsymbol{\Phi}_\ell$'s are extremely accurate estimates of $\boldsymbol{\Phi}^* = \boldsymbol{\Theta}^*\boldsymbol{\Theta}^{*\top}$ (Singh & Vaswani, 2024). We summarize this discussion in Table 1.

5. Conclusions and Future Work

We developed a Byzantine-resilient, sample-, time-, and communication-efficient solution, called Byz-AltGDmin, for few shot learning. We also introduced a novel solution approach, called Subspace Median, for combining subspace estimates from multiple federated nodes when some of them can be malicious. This is likely to be of independent interest for developing a secure initialization approach for various federated low rank matrix recovery, and subspace learning and tracking problems.

The few shot learning problem is almost synonymous with the online subspace tracking problem studied in (Babu et al., 2023) for real-time dynamic MRI. Mini-batch subspace tracking ideas of this work can be useful for few shot learning as well. We will explore real data applications in future.

Acknowledgements

The authors would like to thank Prof. Shana Moothedath of Iowa State University for sharing the linear representation learning (Du et al., 2020) paper with us.

Addressing reviewer comments

We revised the part of Abstract, Introduction, Contributions, and Related Work section to include the work by (Collins et al., 2021) which we missed by mistake. We significantly shortened the section 2. This work does not have the space to add real-world data applications. However we will do this as part of future work by modifying the approaches developed in (Babu et al., 2023).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alistarh, D., Allen-Zhu, Z., & Li, J. (2018). Byzantine stochastic gradient descent. *Advances in Neural Information Processing Systems*, *31*.

Allen-Zhu, Z., Ebrahimian, F., Li, J., & Alistarh, D. (2020). Byzantine-resilient non-convex stochastic gradient descent. *arXiv preprint arXiv:2012.14368*.

Babu, S., Lingala, S. G., & Vaswani, N. (2023). Fast low rank compressive sensing for accelerated dynamic MRI. *IEEE Trans. Comput. Imag.*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, *12*, 149–198.

Boudiaf, M., Ziko, I., Rony, J., Dolz, J., Piantanida, P., & Ben Ayed, I. (2020). Information maximization for few-shot learning. *Advances in Neural Information Processing Systems*, *33*, 2445–2457.

Cao, X., Fang, M., Liu, J., & Gong, N. Z. (2020). Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*.

Cao, X., & Lai, L. (2019). Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing*, *67*(22), 5850–5864.

Chen, Y., Chi, Y., Fan, J., Ma, C., et al. (2021). Spectral methods for data science: A statistical perspective. *Foundations and Trends textregistered in Machine Learning*, *14*(5), 566–806.

Chen, Y., Su, L., & Xu, J. (2017). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *1*(2), 1–25.

Cohen, M. B., Lee, Y. T., Miller, G., Pachocki, J., & Sidford, A. (2016). Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing* (pp. 9–21).

Collins, L., Hassani, H., Mokhtari, A., & Shakkottai, S. (2021). Exploiting shared representations for personalized federated learning. In *International conference on machine learning* (pp. 2089–2099).

Data, D., & Diggavi, S. (2021). Byzantine-resilient SGD in high dimensions on heterogeneous data. In *2021 IEEE International Symposium on Information Theory (ISIT)* (pp. 2310–2315).

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., & Lei, Q. (2020). Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*.

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, *28*(4), 594–611.

Ghosh, A., Hong, J., Yin, D., & Ramchandran, K. (2019). Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*.

Golub, G. H., & Van Loan, C. F. (1989). Matrix computations. *The Johns Hopkins University Press, Baltimore, USA*.

Goyal, P., Mahajan, D., Gupta, A., & Misra, I. (2019). Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 6391–6400).

Hardt, M., & Price, E. (2014). The noisy power method: A meta algorithm with applications. *Advances in neural information processing systems*, *27*.

Jain, P., Kar, P., et al. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, *10*(3-4), 142–363.

Li, L., Xu, W., Chen, T., Giannakis, G. B., & Ling, Q. (2019). RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 1544–1551).

Li, Y., Ildiz, M. E., Papailiopoulos, D., & Oymak, S. (2023). Transformers as algorithms: Generalization and stability in in-context learning. In *International conference on machine learning* (pp. 19565–19594).

Lu, S., Li, R., Chen, X., & Ma, Y. (2022). Defense against local model poisoning attacks to byzantine-robust federated learning. *Frontiers of Computer Science*, *16*(6), 166337.

Maurer, A., Pontil, M., & Romera-Paredes, B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, *17*(81), 1–32.

Minsker, S. (2015). Geometric median and robust estimation in banach spaces.

Nayer, S., Narayanamurthy, P., & Vaswani, N. (2019). Phaseless PCA: Low-rank matrix recovery from column-wise phaseless measurements. In *Intnl. conf. machine learning (icml)*.

Nayer, S., Narayanamurthy, P., & Vaswani, N. (2020, March). Provable low rank phase retrieval. *IEEE Trans. Info. Th.*.

Nayer, S., & Vaswani, N. (2021). Sample-efficient low rank phase retrieval. *IEEE Trans. Info. Th.*.

Nayer, S., & Vaswani, N. (2023, on arXiv since Feb. 2021, Feb.). Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections. *IEEE Trans. Info. Th.*.

Netrapalli, P., Jain, P., & Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization..

Pillutla, K., Kakade, S. M., & Harchaoui, Z. (2019). Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*.

Ravi, S., & Larochelle, H. (2016). Optimization as a model for few-shot learning. In *International conference on learning representations.*

Regatti, J., Chen, H., & Gupta, A. (2022). Byzantine Resilience With Reputation Scores. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (pp. 1–8).

Shen, Z., Ye, J., Kang, A., Hassani, H., & Shokri, R. (2023). Share your representation only: Guaranteed improvement of the privacy-utility tradeoff in federated learning. *arXiv preprint arXiv:2309.05505*.

Singh, A. P., & Vaswani, N. (2024). Byzantine-resilient federated pca and low rank column-wise sensing. *arXiv preprint arXiv:2309.14512*.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, *30*.

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the ieee international conference on computer vision* (pp. 843–852).

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1199–1208).

Tripuraneni, N., Jin, C., & Jordan, M. (2021). Provable meta-learning of linear representations. In *International conference on machine learning* (pp. 10434–10443).

Tripuraneni, N., Jordan, M., & Jin, C. (2020). On the theory of transfer learning: The importance of task diversity. *Advances in neural information processing systems*, *33*, 7852–7862.

Tziotis, I., Shen, Z., Pedarsani, R., Hassani, H., & Mokhtari, A. (2022). Straggler-resilient personalized federated learning. *arXiv preprint arXiv:2206.02078*.

Vaswani, N. (2024). Efficient federated low rank matrix recovery via alternating gd and minimization: A simple proof. *IEEE Trans. Info. Th.*.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science* (Vol. 47). Cambridge university press.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, *29*.

Wu, S. X., Wai, H.-T., Li, L., & Scaglione, A. (2018). A review of distributed algorithms for principal component analysis. *Proceedings of the IEEE*, *106*(8), 1321–1340.

Xie, C., Koyejo, S., & Gupta, I. (2019). Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning* (pp. 6893–6901).

Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning* (pp. 5650–5659).

Yu, M., Guo, X., Yi, J., Chang, S., Potdar, S., Cheng, Y., … Zhou, B. (2018). Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*.