Personalized learning with the existence of harmful workers

by

Brahim Erraji

Thesis submitted to the
Deanship of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
ML

Deprtement of Machine Learning
Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):	Dr. Martin Takáč Associate Professor, Dept. of ML, Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
Internal Member:	Dr. Samuel Horvath Assistant Professor, Dept. of ML, Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
	Dr. Martin Takáč Associate Professor, Dept. of ML, Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the increasing demand for powerful machine-learning models, data is becoming a valuable resource. However, centralized servers can only host a limited amount. Thus, added to privacy concerns, Federated Learning (FL) is currently the go-to framework to train large and data-hungry models without compromising privacy or the accuracy of the model. The success of FL is tightly linked with particular properties of the distributed data, namely homogeneity. The involved workers, each holding their own share of the data, are assumed -optimistically- to provide unbiased gradients of the objective function that will contribute to the convergence of the optimization method to a minimum. However, in many cases, this assumption is violated, either unintentionally because the data in each worker is inherently different. Or intentionally, in what is referred to in the literature as *byzantine behavior*. Therefore, it is no longer wise to trust the updates sent by the workers blindly and hope for the best, especially in large-scale applications where the involved workers are not necessarily known entities and might not be liable for misuse. To this end, we propose a corrupt update filtering mechanism that adopts adaptive aggregation weights on the received update to represent their historical usefulness throughout the training. We consider α (the mixing weights vector) a parameter to be optimized as well. We leverage optimization techniques (gradient descent, derivative-free optimization) to update the α before applying it to update the original parameters x. We get a comparable convergence rate to standard minibatch SGD in the smooth and strongly convex case $\mathcal{O}(\frac{\sigma^2 m^2}{T|H|^3})$, where m is the number of workers, σ is the variance of the stochastic gradients, T is the iteration counter, and $|H|$ is the cardinality of the helpful workers' set. Additionally, we empirically prove the method's merit through a series of experiments involving imagined byzantine behaviors that the method successfully overcomes.

Acknowledgements

Throughout my master's training, I have been lucky to receive guidance and support from my supervisor, Dr. Martin Takac. I would like to express my utmost gratitude to Dr. Martin, as well as to all the talented people I had the pleasure of interacting with, including Dr. Eduard Gorbunov, Dmitry Vilensky-Pasechnyuk, Artem Agafonov, Dr. Dmitry Kamzolov, and many others.

I would also like to thank Dr. Sebastian Stich for hosting me at CISPA, during which time this project was developed. I was fortunate to have had the eye-opening opportunity to experience research in two labs. The access to top faculty and researchers who answered any questions I had and were willing to join lengthy meetings with me greatly shaped my thinking.

Finally, I would like to express my love for the country of the UAE and its people. Staying here during the last two years has been an immense pleasure, and I will leave with so many happy memories that I will look back on. And of course, I would like to thank everyone at Martin's lab and the MBZUAI community as a whole.

Dedication

To Mama and Baba.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
FL	Federated Learning
PFL	Personalized Federated learning
GD	Gradient Descent
SGD	Stochastic Gradient Descent
DFO	Derivative Free Optimization

Chapter 1

Background and motivation

1.1 Introduction

In the last decades, the scientific contributions in Artificial Intelligence (AI) have allowed unprecedented results in many distinct fields [8, 9, 16], in many cases, machine learning lies at the core of these achievements, by automating complicated tasks through learning techniques, beating state of the art in any given task becomes a matter of time. A typical *supervised* machine-learning problem consists of minimizing the number of instances where the machine does the demanded task wrongly (e.g. give a wrong label to an image, regress a very far value for a known data point, ...). This formulation turns a machine learning problem into an optimization one, referred to as optimizing the loss function, which is a function that measures the performance of the machine in the given task (i.e. the number of misclassifications, accumulated error in a regression problem ...). Concretely, a machine-learning problem aims to solve the following problem:

$$\min \mathcal{L}_{(x,y)\sim\mathcal{D}}(\hat{y}, y), \tag{1.1}$$

where $\mathcal{L}$ is a loss function (e.g. euclidean distance), $\mathcal{D}$ is the underlying data distribution for the problem, and $\hat{y}$ is the predicted label for x using a w-parameterized (or non-parameterized) model $h(x, w)$.

The loss function as per (1.1) is by nature stochastic, as we can never access all the possible data points in a given problem. If we did, the problem would become trivial because all possible inputs would already have an output.

Optimizing a stochastic loss objective when having access to a limited dataset has been a challenge because of concerns about overfitting to the empirical samples, causing poor generalization to the underlying data distribution. An immediate solution to this is to provide an unlimited number of data points that will represent the actual distribution with high confidence. But in practice, that is far from being possible, forcing existing optimization methods to provide an approximate solution belonging to a neighborhood of the true optimum proportionate to the local dataset size.

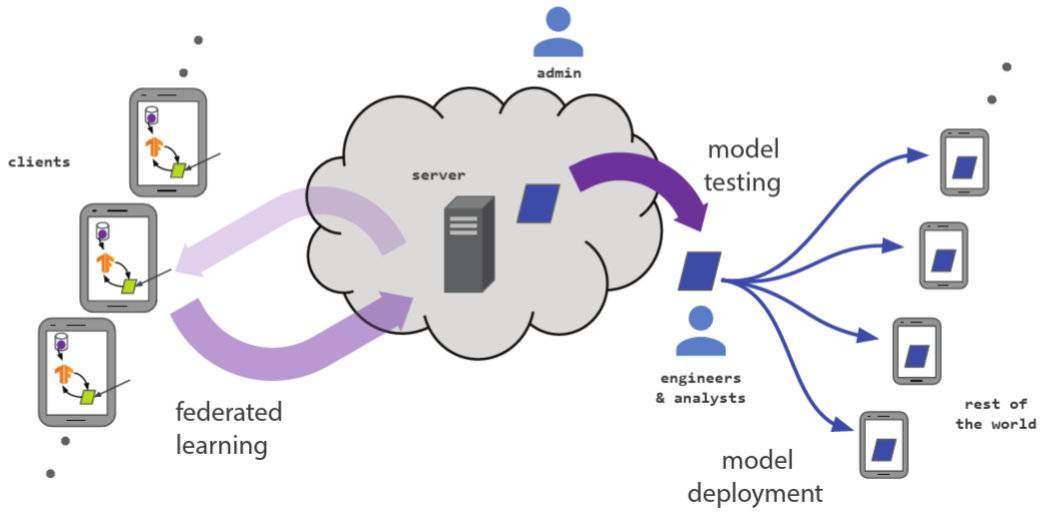


Figure 1.1: Generic lifecycle of an FL model from training to deployment. (image Source [12]).

Therefore, in many settings, where data is stored inherently in a distributed manner and many data access regulations apply (General Data Protection Regulation (GDPR) in Europe and California Consumer Privacy Act (CCPA) in the United States), attention has switched towards collaborative learning, where multiple workers share the same objective and hold their own share of the data. They then provide privacy-preserving information to a trusted server that runs a defined optimization method and sends back the updated model to the workers until a stopping criterion is met. This collaborative learning process is referred to as *Federated learning* [15].

The canonical form for *Federated Learning* is the following optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x), \text{ where } f_i(x) = \mathbb{E}_{\zeta_i \sim \mathcal{D}_i}[f(x, \zeta_i)], \tag{1.2}$$

where $\mathbb{R}^d$ is the parameter space, x is the parameters to be optimized (e.g., weights of a neural network), and m is the number of workers.

	Multi core	FL
N# Workers	4-32	$10^6 - 10^{10}$
Identical Objective	Yes	No
Reliable workers[1]	Yes	No

Table 1.1: Comparison between multi-core optimization and federated learning.

Equation (1.2) happens to be the same objective for an ordinary multi-core distributed optimization problem for the case where $f_1(x) = f_2(x) = \cdots = f_n(x)$, therefore it is insightful to compare the similarities between these two settings. Table 1.1 highlights the key differences between multi-core optimization and FL.

To solve (1.2), the baseline method is distributed gradient descent:

$$x^+ = x - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x). \tag{1.3}$$

at each time, all the workers provide the full gradient over local data on the point formed by the local model parameters state.

Theorem 1.1.1 *For the right choice of step size η, the sequence generated by* (1.3) *will converge to a critical point of* (1.2) *at a rate of:*

$$\|\nabla f(\hat{x})\|^2 \leq \mathcal{O}(\frac{1}{\sqrt{T}}),$$

where $\hat{x} \in \{x^1, \ldots, x^T\}$.

In many cases, only a stochastic estimator of the gradients is available; thus, the update turns into stochastic gradient descent.

$$x^+ = x - \eta \frac{1}{m} \sum_{i=1}^{m} g_i(x), \tag{1.4}$$

where g_i is a usually unbiased estimator of the full gradient ∇f_i.

the stochastic gradients used in (1.4) are usually assumed to have a bounded variance. (i.e. $\mathbb{E}||\nabla f_i(x) - g_i(x)||^2 \leq \sigma^2$). The lower bound complexity for Stochastic Gradient Descent (SGD) is presented in the following theorem.

Theorem 1.1.2 *For the right choice of step size η, the sequence generated by* (1.3) *will converge to a critical point of* (1.2) *at a rate of:*

$$\|\nabla f(\hat{x})\|^2 \leq \mathcal{O}(\frac{\sigma}{\sqrt{mT}}), \tag{1.5}$$

where $\hat{x} \in \{x^1, \ldots, x^T\}$.

Under Theorem 1.1.2, the benefit of collaborating is immediately obvious. Because, We can reach a stationary point much more quickly for a large number of workers.

The rate presented in Theorem 1.1.2 is optimal for first-order methods in the nonconvex setting [1] and thus any alteration of (1.4) will get the same rate or worsen it.

It is not hard to see the bottlenecks of the baseline distributed gradient descent. First, the method is assuming access to full gradients of the workers, which is unpractical and unattainable. Secondly, the method is requiring communication from all the workers all the time, and thus can only progress as fast as the slowest worker.

Each of these challenges motivated an explosion of works in the literature proposing creative solutions to overcome each burden. Stochastic gradient descent and its variations are now used instead of full gradient descent, LocalSGD and asynchronous SGD are solutions to the communication bottleneck and many other branches of study.

Data heterogeneity

Since its emergence FL has been implemented ubiquitously to solve large-scale learning problems [19, 27, 20, 18]. Allowing for the training of increasingly big models in terms of architecture and training data sizes. In all cases in which FL gave impressive results the key assumption was that the workers draw from the same underlying data distribution $\mathcal{D}_1 \equiv \mathcal{D}_2 \equiv \cdots \equiv \mathcal{D}_m$, and thus it makes sense to share a common model between all the involved workers. This assumption is as strong as it seems. One can think of many scenarios where this does not necessarily hold (e.g. two mobile devices storing messages from two different languages cannot collaborate to construct a sentence completer). This observation gave birth to a new line of work focusing on settings where f_i's do not sample from the same underlying distribution.

In fact, the defacto method to solve (1.2) is *Federated Averaging (FedAvg)* (Algorithm 9) first introduced in [15] also known as LocalSGD, it consists of extending the traditional SGD to the distributed setting. The novel part is using an inner loop to update local models before averaging. Using this extra step reduces the number of communication rounds between the server and the workers by a factor of K which is the number of local steps done between two communication rounds. But due to heterogeneity of clients' objectives, *FedAvg* struggles to find a minimum of (1.2) because of shifts introduces in each round. [13] discussed this thoroughly and highlighted the effect of heterogeneity when using local steps.

Therefore, not only does heterogeneity cancel the benefits of collaboration, but it also prevents convergence of LocalSGD-like methods to a minimum when collaboration can be helpful.

While heterogeneity usually arises from the nature of data stored in each worker, in some cases, it can be intentionally amplified by harmful workers in an attempt to drive the training trajectory away from a minimum. This behavior is referred to in the literature as *byzantine attacks* [3, 4].

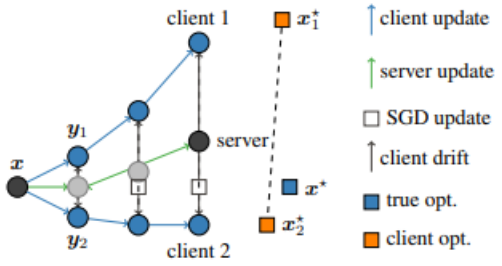


Figure 1.2: Heterogeniety can cause clients' optimum to be randomly placed in the search space, and the minimum of the averages of local objectives might not be the average of the minimums, (image source [13]).

The *byzantine generals* problem [14] is a classic problem in computer science. The simplified version of the problem is the following: assuming a fixed number of generals of the Byzantine army are trying to agree on a battle plan, but a subset of them are traitors, and their goal is to confuse the others and make them agree on a bad plan. The challenge is to come up with an algorithm that would eliminate the effect of the traitor generals. In the main work [14], it is shown that the problem is solvable with more than two-thirds of the generals being loyal, i.e. one traitor general is able to confuse two loyal generals at most.

In the context of federated learning, a *byzantine attack* consists of one or a group of workers providing falsified updates to the server that do not represent their actual gradients. If those updates are applied with the same weight as the correct updates, it can result in an undefined behavior of the optimization method.

To showcase the dangers of *byzantine attacks*, we present the following lemma from [4].

Lemma 1.1.3 (Lemma 1 in [4]) *For any vector $U \in \mathbb{R}^d$, a single byzantine worker can derail the training following the update* (1.3) *in the direction of U.*

Proof: In fact, if a byzantine worker j provides a falsified gradient $\tilde{\nabla} f_j(x) = mU - \sum_{i \neq j} \nabla f_i(x)$, then (1.3) becomes:

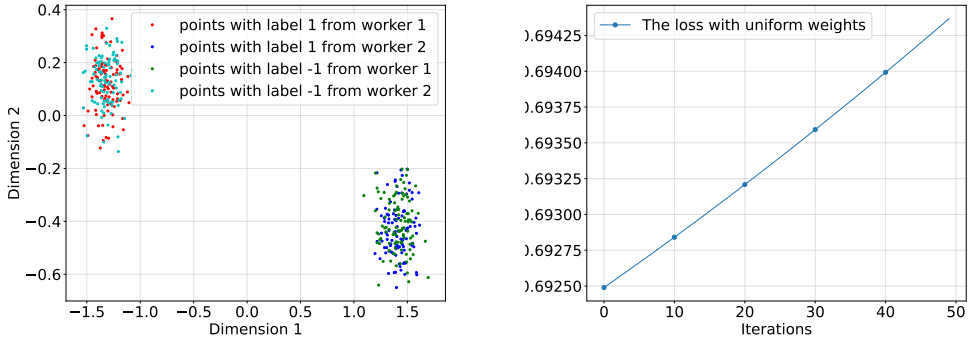
$$x^+ = x - \eta \frac{1}{m}(\sum_{i \neq j} \nabla f_i(x) + \tilde{\nabla} f_j(x)) \quad = x - \eta \frac{1}{m} \sum i \neq j \nabla f_i(x) + \eta \frac{1}{m} \sum_{i \neq j} \nabla f_i(x) - \eta U,$$

$$x^+ = x - \eta U.$$

□

Under Lemma 1.1.3, the harmful worker can choose U to be 0_d and effectively stop the training.

While Lemma 1.1.3 assumes that the byzantine workers have access to the gradients provided by other workers, which is unrealistic and breaches the conditionality that the



(a) Two workers with opposite data distributions

(b) Failure to convergence to an optimum when using uniform weights

Figure 1.3: In this simulated example of heterogeneous workers, the collaboration led to stopping the training and preventing it from reaching a minimum.

server claims, it is still insightful for understanding the effect of providing wrong gradients. In fact, in many cases just providing a random vector, or worse providing a vector U that is pointing to the opposite direction from the expected gradient $\langle U, \nabla f_j(x) \rangle < 0$, can harm the training greatly.

Experiencing heterogeneity empirically

To further motivate the problem with heterogeneity and byzantine behavior, let us set up imagined scenarios where this could take place:

Synthetic data. An illustrative example is shown in Figure 1.3(a), we artificially create two workers with "opposite" data distributions, with the feature space being $\mathcal{X} = \mathbb{R}^2$ and the label space being $\mathcal{Y} = \{-1, 1\}$, and then we choose a logistic regression loss for the objectives of *worker 1* and *worker 2*.

$$f_i(x) = \frac{1}{n_i} \sum_{j=1}^{n_i} \log(1 + \exp(-y_{ij} a_{ij}^T x)), i \in \{1, 2\},$$

where a_i is the data matrix stored in each worker, y_i is the corresponding label vector and n_i are the number of data points in each worker i.

We then run the standard gradient descent algorithm with the gradients received from both workers to optimize f_1. i.e. we generate the following iterative sequence:

$$x^+ = x - \frac{\eta}{2}(\nabla f_1(x) + \nabla f_2(x))$$

the results of running standard distributed gradient descent are shown in Figure 1.3(b), where we notice a clear divergence of the loss objective. illustrating the effect of heterogeneity on the training process.

Real data:
The problems with byzantine workers rise in real-world applications as well. In this part, we train a neural network to classify images of digits using the popular MNIST dataset [7]. The setting involves ten workers holding data in form of images of digits and their respective labels. To simulate the behavior of byzantine workers, we shuffle the labels of the images stored in half of the involved workers, this will result in five workers having mislabelled images and thus will send the wrong gradients at each time. The results are shown in Figure 1.4

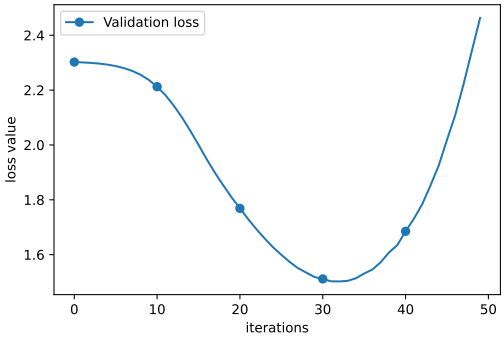


Figure 1.4: With half of the ten involved workers being byzantine, distributed SGD could not converge to an optimum

In light of the empirical experiments, it is obvious that the success of FL is tightly linked with the involved workers being homogeneous in terms of their data. To that end, a new line of work emerged recently, in which homogeneity is no longer taken for granted. Namely: *Personalized federated learning* (PFL) [23].

In PFL, each worker cares about their objective primarily, resulting in a custom model for each worker. As such, a filtering mechanism to eliminate workers who do not share the same objective has to be implemented before or during the training. To that end, many personalization techniques have been experimented with in the literature (see review paper [26] and references therein). The broad technique branches are:

- **No collaboration:** The question to collaborate or not to collaborate is an ongoing research direction, as in many cases the concerns of overfitting to the local data turn into concerns of underfitting because of how different the newly introduced data by the other workers is.
- **Collaboration and then fine-tuning** [11]: In these methods, the first step is to run a standard federated training process and after that, fine-tune the resulting model to each worker, this approach showed potential in real-world applications [22, 24].
- **Clustering:** This line of work consists of grouping the workers into clusters and then running standard federated learning on each cluster, effectively eliminating the negative effect of heterogeneous workers on each other. In this thesis, we focus on this approach for personalization.

1.2 Related work

Clustering the workers is a very intuitive solution to solve the heterogeneity issue. For example, one can opt for the geographical proximity of the workers to form the clusters assuming that workers belonging to the same geographical zone will likely sample from the data distribution (e.g. two hospitals admitting patients from the same neighborhood will likely have similar data). Yet, this information (The location of each worker) is not always available to the server. Moreover, the assumption that the closer is better is not always correct. Therefore the attention switched towards clustering based on the provided updates by the workers (a worker is only as helpful as the information they provide).

The pioneering work that questioned the usefulness of each update received from the m workers is *Krum* [3]. In which the authors formulated the problem of the existence of harmful workers while training a machine learning model concretely. Their contribution lies in answering the question of what is a good aggregation rule. Effectively generalizing (1.3) to:

$$x^+ = x - \eta F(g_1, g_2, \ldots, g_m), \tag{1.6}$$

where F is an aggregation rule used for the received vectors $g_1, \ldots, g_m$ from the m workers, e.g. simple arithmetic average is the case of standard SGD.

To answer that question, they leverage the observation that *good workers* will give updates that are close to each other in expectation. Assuming that there is f byzantine among the m workers, the update rule F should look for the vector that minimizes the distance between the vector itself and the closest $n-f$ workers. (The paper used $n-f-2$ closest vectors instead due to technicalities.). That is using the notation $i \to j$ to say g_j belongs to the the set of $n-f-2$ vectors closest to g_i, the update rule should be $F(g_1, g_2, \ldots, g_m) = g_{i^*}$ such that $s(i^*) \leq s(i) := \sum_{i \to j} \|g_i - g_j\|^2$. Intuitively, F should pick the vector that is in the middle of a group $n-f-2$ of vectors that are close to each other. The authors labeled this rule function as *Krum* or Kr for short.

While the work has theoretical and empirical convergence guarantees when using the Krum function as an update rule, it is very hard to ignore the strong assumptions leading to those guarantees. staring by the assumption on the number of byzantine workers: $2f + 2 < m$. This is in many cases unachievable and even worse unverifiable as the information on the exact number of byzantine workers f is unknown. Also, the method is unimplementable without knowing f. Moreover, the update rule F is only applying one update from the m available at each time, this reduces the benefits of collaboration.

The second work that we estimate most related to this thesis is PFL-TC [25], the approach is to leverage clustering algorithms theory to put the updates into fixed K clusters and apply the updates from the workers into those clusters, a simplified version of the method is resumed in Algorithm 1.

Algorithm 1 PFL-TC

Require: Learning rate η ,momentum parameter α, Number-Clusters K

1: **for** round $t \in [T]$ **do**

2: Server shares current states with worker $x_{i,t-1}$

3: Each client i generates their local stochastic gradient $g_i(x_{i,t-1})$

4: Compute the momentum and share it with the server

$$m_{i,t} = \alpha g_i(x_{i,t-1}) - (1-\alpha)m_{i,t-1}, i \in [m] \tag{1.7}$$

5: Server runs **Threshhold-clustering** $2(\{m_{i,t}\}_{i\in m}, \{v_{k_0^t}\}_{k\in K})$ and get the new cluster means $\{v_{k^t}\}_{k\in[K]}$

6: The server updates the model of each worker

$$x_{i,t} = x_{i,t-1} - \eta v_{k^t}, i \in k^t \tag{1.8}$$

7: **end for**

Algorithm 2 Threshhld-clustering

Require: Points to cluster $\{x_1, \ldots, x_m\}$, K number of clusters, initial cluster means $\{v_{1,0}, \ldots, v_{K,0}\}$

1: **for** round $r \in [R]$ **do**

2: **for** cluster $k \in [K]$ **do**

3: Set a threshold $\tau_{k,r}$

4: Update cluster means as follows

$$y_i = \begin{cases} x_i & \|x_i - v_{k,l}\| \leq \tau_{k,r} \\ v_{k,l-1} & \|x_i - v_{k,l}\| > \tau_{k,r} \end{cases}$$
$$v_{k,l} = \frac{1}{n}\sum_{i=1}^{m} y_i \tag{1.9}$$

5: **end for**

6: **end for**

7: **return** cluster means $\{v_k = v_{k,R}\}$ and the clusters $\{C_k = \{x_i : \|x_i - v_{k,R}\| \leq \tau_{k,R}\}\}_{k\in[K]}$

To understand Algorithm 1, one should think of the updates (momentum or stochastic gradient) as points in the space $\mathbb{R}^d$. As with any scatter of points, they can be clustered. in other words, we can leverage the extensive clustering literature to help us eliminate heterogeneity.

Starting with this observation, The authors were able to get a rate (Theorem 3 in [25]) comparable to the optimal rate:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla f_i(x_{i,t-1})\|^2 \leq \mathcal{O}\left(\sqrt{\frac{\sigma^2}{n_i T}}\right), \tag{1.10}$$

with $\mathbb{E}\|g_i(x,\zeta_i) - \nabla f_i(x)\|^2 \leq \sigma$ and n_i is the number of workers in the same cluster as worker i.

Although successful in getting an optimal rate, the method is far from practical. In fact, the assumptions that ensure convergence is very hard to verify as they depend on the updates received from the workers on each round r. Yet, the work still serves as an illustration of what update-based clustering can achieve.

Chapter 2

Preliminaries

For a smooth and coherent reading of this thesis, we invite the reader to refer to this chapter for the clarification of any notations or results that were used in the analysis phases. This chapter is independent of the other chapters and can be accessed anytime during the reading of this thesis.

2.1 Notation

We list the notation used throughout this thesis as follows

- We use m to denote the number of workers, and d to denote the parameter space dimension. subsequently, $\mathbb{R}^d$ and $\mathbb{R}^m$ are the parameters and mixing weight spaces, respectively. we use x and α to refer to the parameters and weights, respectively.
- Throughout the thesis, α is forced to stay in the probability simplex Δ of $\mathbb{R}^m$ with Δ defined as follows

$$\Delta := \left\{(\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbb{R}^m, \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0, \forall i \in [m]\right\}. \tag{2.1}$$

- We use subscript to index a worker and superscript to index the round (e.g. α_i^r refers to the mixing weight of worker i at the r-th round.). occasionally, double superscripts are used when needed and the notation is explained therein.
- 0_m or 0_d is used to denote the null vector of $\mathbb{R}^m$ and $\mathbb{R}^d$, respectively.
- ζ_i is used to denote the random variables behind the randomness in the stochastic gradients sent by the m workers.
- f_{target} is used to denote the objective function and can be referred to as f for short.
- V is the stochastic validation function that the worker having f_{target} as objective provides to the server.

- We use superscript c to denote the complementary of a set in a mother set, e.g. $\{1,2\}^c$ in $\{1,2,3\}$ is $\{3\}$.

2.2 Smoothness and strong convexity

Note that the choice of the norms in all the following equations is irrelevant since all norms defined over $\mathbb{R}^d$ are equivalent (d is finite). All objectives in this thesis are assumed to be smooth and strongly convex with parameters to be stated when needed.

2.2.1 Lipschitz smoothness

The term "smoothness" used throughout this thesis, refers to the Lipschitz continuity of the gradients of a function f. i.e. the function f satisfies

$$\forall x, y \in \mathbb{R}^d : \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|. \tag{2.2}$$

Note that (2.2) has many equivalent definitions (Theorem 2.1.5 in [17]). We primarily use the quadratic upper bound definition as in (2.3)

$$\forall x, y \in \mathbb{R}^d, f(x) \leq f(y) + \langle \nabla f(y), x - y\rangle + \frac{L}{2}\|x - y\|^2. \tag{2.3}$$

2.2.2 Strong convexity

Strong convexity is a generalization of the standard convexity, for a $\mu \geq 0$, a function is μ-strongly convex if it verifies:

$$\forall x, y \in \mathbb{R}^d, f(x) \geq f(y) + \langle \nabla f(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2. \tag{2.4}$$

For the case where $\mu = 0$, the definition (2.4) turns into the standard definition of a convex function.

Similarly, (2.4) has many equivalent definitions (Thereom 2.1.10 in [17]).

The strongly convex class of functions satisfies a very important property that makes acquiring convergence rate for a global minimum of a given method $\mathcal{M}$ possible:

$$f \textit{ is strongly convex}, x^* \textit{ is a local minimum} \implies x^* \textit{ is a global minimum and it is unique.} \tag{2.5}$$

2.3 Projection onto the simplex properties

The projection operator $\mathcal{P}$ that sends a vector $\hat{\alpha} \in \mathbb{R}^m$ to $\alpha \in \Delta$ requires algorithms of similar complexity to the sorting algorithm to be applied. Additionality, it holds certain useful properties that we shall list:

Proposition 2.3.1 (Proposition 2 in [6]) *Let $\hat{\alpha}$ be in $\mathbb{R}^m$ and α be the projection of $\hat{\alpha}$ onto the probability simplex Δ:*

$$\exists \tau \in \mathbf{R}, \forall i \in [m] : \alpha = \max\{\hat{\alpha} - \tau, 0\}. \tag{2.6}$$

Proposition 2.3.2 *The projection operator to probability simplex Δ is insensitive to shifting:*

$$\forall \hat{\alpha} \in \mathbb{R}^m, \forall c \in \mathbb{R} : \mathcal{P}(\hat{\alpha} + c) = \mathcal{P}(\hat{\alpha}). \tag{2.7}$$

Following Proposition 2.3.1, let α be the projection onto Δ of $\hat{\alpha}$ and denote $\mathcal{I}$ the set of coordinates i such as: $\max\{\hat{\alpha}_i - \tau, 0\} = \hat{\alpha}_i - \tau$. then $\sum_{i=1}^m y_i = \sum_{i \in \mathcal{I}}^m (\hat{\alpha}_i - \tau) = 1$, resulting in $\tau = \frac{\sum_{i \in \mathcal{I}}^m \hat{\alpha}_i - 1}{|\mathcal{I}|}$. Intuitively, the following algorithm to find τ can be constructed as in Algorithm 3.

Algorithm 3 Projection onto probability simplex (Sorting)

Require: $\hat{\alpha} \in \mathbb{R}^m$ the vector to be projected

1: Sort $\hat{\alpha}$ into u: $u_1 \geq u_2 \cdots \geq u_m$
2: Set $K := \max_{1 \leq k \leq m} \left[k : \frac{\sum_{i=1}^k \hat{\alpha}_i - 1}{k} < u_k \right]$
3: Set $\tau = \frac{\sum_{i=1}^K \hat{\alpha}_i - 1}{K}$
4: For $i \in [m]$, $\alpha_i = \max\{\hat{\alpha}_i - \tau, 0\}$
5: **return** α

Chapter 3

New approach to solve heterogeneity

Our approach to solving the heterogeneity issue is based on the observation that applying uniform weights to all the workers is not ideal, therefore it is intuitive to opt for a filtering mechanism that lowers the impact of harmful workers. To address this, we leverage the fact that ideally, the server will assign a 0 weight to the updates received from harmful workers. Assuming the server's access to a validation loss, we can evaluate the performance of the workers on the validation loss that is representative of the objective function we are trying to optimize. The intuition is that workers that worsen the validation loss will get decreased weight and vice-versa. In this new framework, we take personalization to its limit by putting each worker in the center of optimization and finding a custom model for it before moving on to the next worker. Concretely we turn the canonical form of FL (1.2) to (3.1).

$$\arg\min_{x\in\mathbb{R}^d} f_{target}(x) := \mathbb{E}_{\zeta\sim\mathcal{D}}[f(x,\zeta)], \tag{3.1}$$
$$\text{with access to unbiased gradients of } f_i(x) := \mathbb{E}_{\zeta_i\sim\mathcal{D}_i}[f(x,\zeta_i)], i \in [n].$$

Under (3.1), it becomes clear that the task is to figure out which of the workers sample from $\mathcal{D}$. Subsequently, we transform the standard update from (1.3) to (3.2), a weighted averaging rule:

$$x^+ = x - \eta_x \sum_{i=1}^{n} \alpha_i \nabla f_i(x), \tag{3.2}$$

where we consider α a parameter to be optimized as well.

The approach to optimizing α is by providing a validation set to the server that can be used to construct a differentiable function with respect to α validation function. Algorithm 4 summarizes the initial approach. The intuition is that if V is representative of f_{target} then those workers who increase the validation loss will decrease weights and vice versa.

The choice to project the mixing weights α to the probability simplex $\mathcal{P}$ allows the method to be comparable with the standard *SGD* for example if all the workers are helpful α following the update (4) and (5) will converge to the uniform weights as in standard *SGD*.

Algorithm 4 Adaptive mixing weights SGD

Require: x^0 – starting point, α^0 initial averaging vector, Validation loss V

1: **for** $r = 0, 1, \ldots$ **do**
2: Receive M stochastic gradients $\nabla f_i(x^r, \zeta_i^r)$ from all workers
3: $x^{r+1} = x^r - \eta_x \sum_{i=0}^{M-1} \alpha_i^r \nabla f_i(x^r, \zeta_i^r)$
4: $\alpha^{r+1} = \alpha^r - \eta_\alpha \frac{\partial V(x^{r+1})}{\partial \alpha}$
5: Project α^{r+1} onto probability simplex
6: **end for**

Contributions

In light of the challenges and concerns discussed above, we list our contributions as follows

- We introduce a new personalized federated learning framework where one worker is in the center of optimization and the others are *helpers*.
- Our methodology allows the elimination of updates received from harmful workers.
- Theoretically prove convergence to the optimum x^* in the smooth and strongly convex case at the optimal rate of $\mathcal{O}(\frac{\sigma^2}{T|H|})$.
- We run extensive experiments in both deterministic and stochastic cases with imagined byzantine behaviors to prove the merit of the method.

3.1 Convergence guarantees

A good starting point to get a convergence rate for our approach is to concretely define the notion of a *helpful/harmful* worker. While in standard *SGD*, workers are supposed to provide unbiased stochastic gradients throughout the training, we adopt a more general framework as per Definition 3.1.1.

Definition 3.1.1 *For any $x \in \mathbb{R}^d$, the cluster of helpful workers $H(x)$ at point x is defined as followed.*

$$\forall x \in \mathbb{R}^d, i \in H(x) : \langle \nabla V(x), \nabla f_i(x, \zeta_i) \rangle \geq c \|\nabla f(x)\|^2,$$
$$\text{for some } c \in (0, 1].$$

Also, denote H as the set of workers who are always helpful.

$$H = \bigcap_{k \geq 0} H(x^k).$$

Algorithm 5 Adaptive mixing weights SignSGD

Require: x^0 – starting point, α^0 initial averaging vector. Validation loss V

1: **for** $r = 0, 1, \ldots$ **do**
2: Receive M stochastic gradients $\nabla f_i(x^r, \zeta_i^r)$ from all workers
3: $x^{r+1} = x^r - \eta_x \sum_{i=0}^{M-1} \alpha_i^r \nabla f_i(x^r, \zeta_i^r)$
4: $\alpha^{r+1} = \alpha^r - \eta_\alpha \operatorname{Sign}(\nabla_\alpha V(x^{r+1}))$
5: Project α^{r+1} onto probability simplex
6: **end for**

An implicit remark in Definition 3.1.1 is that H is never empty. Despite the fact that it is decreasing in cardinality but it never touches 0 because at worst the worker that provided f_{target} is always in H. Or at best $H := [m]$ and the setting becomes the same as that of standard SGD.

Throughout this section, the following assumptions are made.

Assumption 3.1.2 *The validation loss V is representative of the objective loss f_{target}:*

$$\forall x \in \mathbb{R}^d, \mathbb{E}[V(x)] = f_{target}(x),$$

where the expectation is taken with respect to the sampling of V.

Assumption 3.1.3 *the objective loss f_{target} is μ-strongly convex and $L-$smooth*

$$\forall x, y \in \mathbb{R}^d : \frac{\mu}{2}\|x - y\|^2 \leq f(x) - f(y) - \langle \nabla f(y), x - y \rangle \leq \frac{L}{2}\|x - y\|^2.$$

Assumption 3.1.4 *The received stochastic gradients are bounded*

$$\exists B \geq 0, \forall x \in \mathbb{R}^d : \|\nabla f_i(x, \zeta_i)\| \leq B.$$

Assumption 3.1.5 *The random variables $\zeta_i, i \in [m]$, are independent of each other.*

$$\forall x \in \mathbb{R}^d, \forall i \neq j \in [m] : \langle \nabla f_i(x, \zeta_i) - \mathbb{E}[\nabla f_i(x, \zeta_i)], \nabla f_i(x, \zeta_j) - \mathbb{E}[\nabla f_j(x, \zeta_j)] \rangle = 0.$$

3.2 SignSGD for the update of α

Algorithm 4 serves as the baseline for our contribution, to analyze the method we propose a more controlled version of 4 based on *SignSGD* [2].

The modification in Algorithm 5 is to update α following $\operatorname{Sign}(\nabla_\alpha V(x^{r+1}))$. This change gives the same reward to all the helpful workers at time k. As opposed to before where the size of the gradient influenced the respective reward. To see this more clearly, we construct an example where two simple two quadratic functions try to collaborate to find the minimum of the first function, the results are shown in Figure 3.1.

Throughout the rest of this section, we refer to f_{target} as f for simplicity.

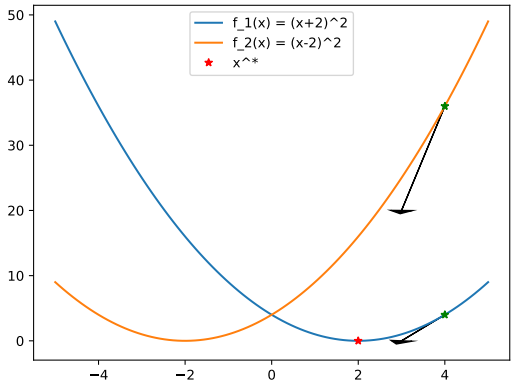


Figure 3.1: Running Algorithm 4 to find the mixing weights for gradients received from f_1 and f_2 will give a higher weight to f_2, despite the fact that we are optimizing for f_1.

Lemma 3.2.1 *Let $g \in \mathbb{R}^d$ be a random vector, for any vector h, we have:*

$$\mathbb{E}[\|g - \mathbb{E}[g]\|^2] = E[\|g - h\|^2] - E[\|E[g] - h\|^2].$$

Following Definition 3.1.1 and Algorithm 5, we can track the way the mixing weights for the workers H evolve through the following Lemma:

Lemma 3.2.2 *For any time $k \geq 0$:*

$$\forall k \geq 0, \eta_x \leq \frac{c\|\nabla f(x^k)\|^2}{LB^2} \implies \sum_{i \in H} \alpha_i^{k+1} \geq \sum_{i \in H} \alpha_i^k.$$

Proof: Denote $\hat{\alpha}^{k+1}$ as α^{k+1} before projection to simplex.
In fact the update for $i \in H(x^k)$:

$$\begin{aligned}
\hat{\alpha}_i^{k+1} &= \alpha_i^k - \eta_\alpha \operatorname{Sign}[\nabla_\alpha V(x^{k+1})] \\
&= \alpha_i^k - \eta_\alpha \operatorname{Sign}[\nabla_\alpha V(x^{k+1})] \\
&\overset{\text{Chain rule}}{=} \alpha_i^k - \eta_\alpha \operatorname{Sign}\left[\left\langle \frac{\partial V(x^{k+1})}{\partial w}, \frac{\partial x^{k+1}}{\partial \alpha} \right\rangle\right] \\
&\overset{(3)}{=} \alpha_i^k - \eta_\alpha \operatorname{Sign}\left[- \eta_x \langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i^k) \rangle\right] \\
&= \alpha_i^k + \eta_\alpha \operatorname{Sign}\left[\langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i^k) \rangle\right],
\end{aligned}$$

And in the other side:

$$\begin{aligned}\langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i^k)\rangle &= \langle \nabla V(x^{k+1}) - \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k)\rangle + \langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k)\rangle \\ &\overset{(3.1.1)}{\geq} \langle \nabla V(x^{k+1}) - \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k)\rangle + c\|\nabla f(x^k)\|^2 \\ &\overset{\textit{cauchy-schwartz}}{\geq} -\|\nabla V(x^{k+1}) - \nabla V(x^k)\|\|\nabla f_i(x^k, \zeta_i^k)\| + c\|\nabla f(x^k)\|^2 \\ &\overset{A\ 3.1.3, A\ 3.1.4}{\geq} -L\mathbb{E}\|x^{k+1} - x^k\|B + c\|\nabla f(x^k)\|^2 \\ &\overset{(3), \sum_{i=1}^m \alpha_i^k = 1}{\geq} -\eta_x L B^2 + c\|\nabla f(x^k)\|^2,\end{aligned}$$

Therefore for $\eta_x \geq \frac{c\|\nabla f(x^k)\|^2}{LB^2} \implies \text{Sign}\left[\langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i^k)\rangle\right] = 1.$

And thus:

$$\eta_x \geq \frac{c\|\nabla f(x^k)\|^2}{LB^2} \implies \hat{\alpha}_i^{k+1} = \alpha_i^k + \eta_\alpha. \tag{3.3}$$

For the second part of the proof we consider two cases:

The first case, $\forall i \in [m] : \mathbb{E}[\hat{\alpha}_i^{k+1}] = \hat{\alpha}_i^k + \eta_\alpha$ then $\hat{\alpha}^{k+1} = \alpha^k + \eta_\alpha$, projection is invariant by shifting:

$$\begin{aligned}&\implies \alpha^{k+1} = \mathcal{P}(\hat{\alpha}^{k+1}) = \mathcal{P}(\alpha^k) = \alpha^k, \\ &\implies \sum_{i\in H} \alpha_i^{k+1} \geq \sum_{i\in H} \alpha_i^k.\end{aligned}$$

The second case: $\exists i \in [m]/\{H\} : \hat{\alpha}_i^{k+1} = \alpha_i^k$ or $\hat{\alpha}_i^{k+1} = \alpha_i^k - \eta_\alpha$

Following Proposition 2.3.1:

$$\exists \tau \in \mathbb{R} : \alpha_i^{k+1} = \max\{\hat{\alpha}_i^{k+1} - \tau, 0\}.$$

Assume $\tau > \eta_\alpha$ then:

$$\begin{aligned}&\implies \sum_{i=1}^m \alpha_i^{k+1} = 1 \\ &\implies \sum_{i=1}^m \max\{\hat{\alpha}_i^{k+1} - \tau, 0\} = 1 \\ &\implies \sum_{i=1}^m \max\{\hat{\alpha}_i^{k+1} - \eta_\alpha, 0\} > 1 \\ &\overset{\sum_{i=1}^m \max\{\hat{\alpha}_i^{k+1} - \eta_\alpha, 0\} \leq \sum_{i=1}^m \max\{\alpha_i^k, 0\}}{\implies} \sum_{i=1}^m \max\{\alpha_i^k, 0\} > 1 \\ &\implies \sum_{i=1}^m \alpha_i^k > 1.\end{aligned}$$

Contradiction with $\sum_{i=1}^{m} \alpha_i^k = 1$, therefore $\tau \leq \eta_\alpha$
And:

$$\begin{aligned}
\alpha_i^{k+1} &= \max\{\hat{\alpha}_i^{k+1} - \tau, 0\} \\
&\geq \max\{\hat{\alpha}_i^{k+1} - \eta_\alpha, 0\}, \\
\alpha_i^{k+1} &\overset{(3.3)}{\geq} \alpha_i^k.
\end{aligned}$$

And finally, we conclude:

$$\forall k \geq 0, \eta_x \leq \frac{c\|\nabla f(x^k)\|^2}{LB^2} \implies \sum_{i \in H} \alpha_i^{k+1} \geq \sum_{i \in H} \alpha_i^k.$$

□

Corollary 3.2.3 *For a $k \in \mathbb{N}$, if $\sum\limits_{i \in H} \alpha_i^k = 1$ then:*

$$\forall i \in [m]/H^k : \forall K \geq k : \alpha^K = 0.$$

Lemma 3.2.4 *Let $k \geq 0$, for an $i \in [m]$: if $\langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k) \rangle < 0$, and $\eta_x \leq \frac{-\langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i) \rangle}{LB^2}$ and $\eta_\alpha \geq 1 - \frac{1}{m} + \frac{1}{|H|}$ then:*

$$\alpha_i^{k+1} = 0.$$

Proof: Denote $\hat{\alpha}_i^{k+1}$ as α_i^{k+1} before projection. Assume $\langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k) \rangle < 0$

$$\hat{\alpha}_i^{k+1} = \alpha_i^k + \eta_\alpha \operatorname{Sign}\left[\langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i) \rangle\right]$$

From the other side:

$$\begin{aligned}
\langle \nabla V(x^{k+1}), \nabla f_i(x^k, \zeta_i) \rangle &= \langle \nabla V(x^{k+1}) - V(x^k), \nabla f_i(x^k, \zeta_i) \rangle + \langle \nabla V(x^k), g_i^k \rangle \\
&\leq \|\nabla V(x^{k+1}) - \nabla V(x^k)\| \|\nabla f_i(x^k, \zeta_i)\| + \langle \nabla V(x^k), g_i^k \rangle \\
&\leq L\|x^{k+1} - x^k\| \|\nabla f_i(x^k, \zeta_i)\| + \langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i) \rangle \\
&\leq \eta_x L \sum_{i=1}^{m} \alpha_i^k \|\nabla f_i(x^k, \zeta_i)\| B + \mathbb{E}[\langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i) \rangle] \\
&\leq \eta_x L B^2 + \langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i) \rangle,
\end{aligned}$$

We conclude that for $\eta_x < \frac{-\langle \nabla V(x^k), \nabla f_i(x^k,\zeta_i)\rangle}{LB^2} \implies \langle \nabla V(x^{k+1}), \nabla f_i(x^k,\zeta_i)\rangle < 0$. And thus: $\forall i \in H^c : \hat{\alpha}_i^{k+1} = \alpha_i^k - \eta_\alpha$.

Following Proposition 2.3.1:

$$\exists \tau \in \mathbb{R} : \alpha_i^{k+1} = \max\{\hat{\alpha}_i^{k+1} - \tau, 0\} = \max\{\alpha_i^k - \eta_\alpha - \tau, 0\},$$

Assume $\tau < \frac{1}{m} - \frac{1}{|H|} \implies \forall i \in H : \alpha_i^{k+1} = \alpha_i^k + \eta_\alpha - \tau > \frac{1}{m} + \eta_\alpha - \frac{1}{m} + \frac{1}{|H|} > \frac{1}{|H|}$ (contradiction with $\forall i \in [m], \alpha_i^k \leq \frac{1}{|H|}$), thus $\tau \geq \frac{1}{m} - \frac{1}{|H|}$.
And finally choosing $\eta_\alpha \geq 1 - \frac{1}{m} + \frac{1}{|H|} \implies$

$$\forall i \in Neg(x^k) : \hat{\alpha}_i^{k+1} - \tau = \alpha_i^k - \eta_\alpha - \tau \leq 0 \implies \alpha_i^{k+1} = 0. \tag{3.4}$$

□

Corollary 3.2.5 *Following Lemma 3.2.2 and Lemma 3.2.4, if* $\exists \hat{i} \in [m]$ *such as:* $\langle \nabla V(x^k), \nabla f_i(x^k, \zeta_i^k)\rangle < 0$ *for a small enough* η_x*, we have*

$$\sum_{i \in H^k} \alpha_i^{k+1} \geq \sum_{i \in H^k} \alpha_i^k + \frac{|H^k|}{m-1} \alpha_{\hat{i}}^k. \tag{3.5}$$

We now present the main convergence theorem for Algorithm 5.

Theorem 3.2.6 *Let all assumptions hold, starting from a uniform mixing weights* $\alpha^0 = (\frac{1}{m}, \ldots, \frac{1}{m})$*,and choose* $\eta_w^k = \min_{i \in Neg(x^k)} \{\frac{(2k+1)m}{(k+1)^2|H|\mu c}, \frac{c\|\nabla f(x^k)\|^2}{LB^2}, \frac{-\langle \nabla V(x^k), \nabla f_i(x^k,\zeta_i)\rangle}{LB^2}\}$*, and* $\eta_\alpha \geq 1 - \frac{1}{m} + \frac{1}{|H|}$*. we have:*

$$\mathbb{E}[f(x^{k+1}) - f^* | x^k] \leq \frac{mB^2(m-|H|)}{|H|(k+1)\mu c} + \frac{4L\sigma^2}{(k+1)\mu c} \frac{m^2}{|H|^3} + L\frac{4m^2}{(k+1)|H|^2\mu^2 c^2}|H^c|B^2, \tag{3.6}$$

with $Neg(w) := \{i \in [H(w)]^c : \langle \nabla V(w), \nabla f_i(w, \zeta_i^k)\rangle < 0\}$.

Proof: Applying the smoothness of f:

$$\begin{aligned}
\mathbb{E}[f(x^{k+1})|x^k,\alpha^k] &\leq f(x^k)+\mathbb{E}\left[\langle\nabla f(x^k),x^{k+1}-x^k\rangle\middle|x^k,\alpha^k\right]+\frac{L}{2}\mathbb{E}\left[\|x^{k+1}-x^k\|^2\middle|x^k,\alpha^k\right]\\
&\overset{(3)}{=} f(x^k)-\eta_x\mathbb{E}\left[\langle\nabla f(x^k),\sum_{i=1}^m\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\rangle\middle|x^k,\alpha^k\right]\\
&+\frac{L\eta_x^2}{2}\mathbb{E}\left[\|\sum_{i=1}^m\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\|^2\middle|x^k,\alpha^k\right]\\
&=f(x^k)-\eta_x\sum_{i=1}^m\alpha_i^k\left\langle\nabla f(x^k),\mathbb{E}[\nabla f_i(x^k,\zeta_i^k)|x^k,\alpha^k]\right\rangle\\
&+\frac{L\eta_x^2}{2}\mathbb{E}\left[\left\|\sum_{i=1}^m\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\right\|^2\middle|x^k,\alpha^k\right]\\
&\overset{\text{Def (3.1.1),Assump (3.1.4)}}{\leq} f(x^k)-\eta_x c\|\nabla f(x^k)\|^2\sum_{i\in H(x^k)}\alpha_i^k\\
&-\eta_x\sum_{i\in[H(x^k)]^c}\alpha_i^k\mathbb{E}[\langle\nabla f(x^k),\nabla f_i(x^k,\zeta_i^k)\rangle|x^k,\alpha^k]+\frac{L\eta_x^2}{2}\mathbb{E}\left[\left\|\sum_{i=1}^m\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\right\|^2\middle|x^k,\alpha^k\right]\\
&\leq f(x^k)-\eta_x c\|\nabla f(x^k)\|^2\sum_{i\in H(x^k)}\mathbb{E}[\alpha_i^k]-\eta_x\sum_{i\in[H(x^k)]^c}\alpha_i^k\mathbb{E}[\langle\nabla f(x^k),\nabla f_i(x^k,\zeta_i^k)\rangle|x^k,\alpha^k]\\
&+L\eta_x^2\left[\mathbb{E}\left\|\sum_{i\in H(x^k)}\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\right\|^2+\mathbb{E}\left\|\sum_{i\in[H(x^k)]^c}\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\right\|^2\right]\\
&\overset{\text{Lemma 3.2.1},\zeta_i\text{ are independent}}{\leq} f(x^k)-\eta_x c\|\nabla f(x^k)\|^2\sum_{i\in H(x^k)}\mathbb{E}[\alpha_i^k]+\eta_x B^2\sum_{i\in[H(x^k)]^c}\mathbb{E}[\alpha_i^k]\\
&+L\eta_x^2\left[\left\|\sum_{i\in H(x^k)}\alpha_i^k\nabla f_i(x^k)\right\|^2+\sum_{i\in H(x^k)}(\alpha_i^k)^2\mathbb{E}\left[\|\nabla f_i(x^k,\zeta_i^k)-\nabla f_i(x^k)\|^2\right]\right.\\
&\left.+\mathbb{E}\left\|\sum_{i\in[H(x^k)]^c}\alpha_i^k\nabla f_i(x^k,\zeta_i^k)\right\|^2\right]\\
&\overset{l_2\text{ norm is convex, jensen's inequality}}{=} f(x^k)-\eta_x\sum_{i\in H(x^k)}\alpha_i^k(c-L\eta_x\sum_{i\in H(x^k)}\alpha_i^k)\|\nabla f(x^k)\|^2\\
&-\eta_x\sum_{i\in[H(x^k)]^c}\alpha_i^k\mathbb{E}\left[\langle\nabla f(x^k),\nabla f_i(x^k,\zeta_i^k)\rangle|x^k\right]\\
&+L\eta_x^2\sigma^2\sum_{i\in H(x^k)}(\alpha_i^k)^2+L\eta_x^2\sum_{i\in[H(x^k)]^c}\alpha_i^k\mathbb{E}\left[\|\nabla f_i(x^k,\zeta_i^k)\|^2\middle|x^k\right]
\end{aligned}$$

$$\overset{\eta_x \leq \frac{c}{2L\sum_{i\in H(x^k)}\alpha_i^k},\ \text{Assump 3.1.4}}{\leq} f(x^k) - c\sum_{i\in H(x^k)}\alpha_i^k\eta_x\|\nabla f(x^k)\|^2 \tag{3.7}$$
$$- \eta_x\sum_{i\in[H(x^k)]^c}\alpha_i^k\mathbb{E}\left[\langle\nabla f(x^k),\nabla f_i(x^k,\zeta_i^k)\rangle\middle| x^k,\alpha^k\right] + L\eta_x^2\sigma^2\sum_{i\in H(x^k)}(\alpha_i^k)^2$$
$$+ L\eta_x^2B^2\sum_{i\in[H(x^k)]^c}\alpha_i^k$$

Denote $Neg(x) := \{i \in [H^k(w)]^c : \mathbb{E}\langle\nabla f(x),\nabla f_i(x,\zeta_i^k)\rangle < 0\}$

Line (3.7) becomes:

$$\mathbb{E}[f(x^{k+1})] \leq f(x^k) - c\sum_{i\in H(x^k)}\alpha_i^k\eta_x\|\nabla f(x^k)\|^2 + \eta_xB^2\sum_{i\in Neg(x^k)}\alpha_i^k$$
$$+ L\eta_x^2\sigma^2\sum_{i\in H(x^k)}(\alpha_i^k)^2 + L\eta_x^2B^2\sum_{i\in H^c(x^k)}\alpha_i^k$$
$$\mathbb{E}[f(x^{k+1}) - f^*] \overset{SC}{\leq} \left(1 - c\mu\frac{|H|}{m}\eta_x\right)[f(x^k) - f^*] + \eta_xB^2\sum_{i\in Neg(x^k)}\alpha_i^k$$
$$+ L\eta_x^2\sigma^2\sum_{i\in H}(\alpha_i^k)^2 + L\eta_x^2B^2\sum_{i\in H^c}\alpha_i^k$$

Since $\forall i,j \in H : \alpha_i^k = \alpha_j^k$ and $\sum_{i\in H}\alpha_i^k \leq 1$, Therefore: $\forall\ i \in H : \alpha_i^k \leq \frac{1}{|H|} \implies \sum_{i\in H}(\alpha_i^k)^2 \leq \frac{1}{|H|}$

$$\implies \mathbb{E}[f(x^{k+1}) - f^*] \leq \left(1 - c\mu\frac{|H|}{m}\eta_x\right)[f(x^k) - f^*] + \eta_xB^2\sum_{i\in Neg(x^k)}\alpha_i^k$$
$$+ \frac{L\eta_x^2\sigma^2}{|H|} + L\eta_x^2|H^c|B^2\sum_{i\in H^c}(\alpha_i^k)^2$$

Strongly convex case: $\mu > 0 : \eta_x = \frac{(2k+1)m}{(k+1)^2|H|\mu c}$

$$\mathbb{E}[f(x^{k+1}) - f^*] \leq \frac{k^2}{(k+1)^2}[f(x^k) - f^*] + \frac{(2k+1)m}{(k+1)^2|H|\mu c}B^2 \sum_{i \in Neg(x^k)} \alpha_i^k + \frac{L(2k+1)^2\sigma^2}{(k+1)^4\mu c}\frac{m^2}{|H|^3}$$
$$+ L\frac{(2k+1)^2m^2}{(k+1)^4|H|^2\mu^2c^2}|H^c|B^2$$
$$\implies \beta(k+1) \leq \beta(k) + \frac{(2k+1)m}{|H|\mu c}B^2 \sum_{i \in Neg(x^k)} \alpha_i^k + \frac{L(2k+1)^2\sigma^2}{(k+1)^2\mu c}\frac{m^2}{|H|^3} + L\frac{(2k+1)^2m^2}{(k+1)^2|H|^2\mu^2c^2}|H^c|B^2$$
$$\overset{\frac{2j+1}{j+1} \leq 2}{\leq} \beta(k) + \frac{(2k+1)m}{|H|\mu c}B^2 \sum_{i \in Neg(x^k)} \alpha_i^k + \frac{4L\sigma^2}{\mu c}\frac{m^2}{|H|^3} + L\frac{4m^2}{|H|^2\mu^2c^2}|H^c|B^2$$
$$\overset{recusion}{\leq} \frac{mB^2}{|H|\mu c}\sum_{j=0}^{k}\left[(2j+1)\sum_{i \in Neg(x^k)} \alpha_i^j\right] + \frac{4L\sigma^2}{\mu c}\frac{m^2}{|H|^3}(k+1) + L\frac{4m^2}{|H|^2\mu^2c^2}|H^c|B^2(k+1)$$

with $\beta(k) = k^2\mathbb{E}[f(x^k) - f^*]$

Let's bound: $\sum_{j=0}^{k}\left[(2j+1)\sum_{i \in Neg(x^k)} \alpha_i^j\right]$

Following Lemma 3.2.4, the term $\sum_{i \in Neg(x^k)} \alpha_i^j$ will be 0 if $Neg(x^k) = \emptyset$ or $\forall i \in Neg(x^k), \exists k' < k : i \in Neg(x^{k'})$. We use those observations to conclude that:

$$\sum_{j=0}^{k}\left[(2j+1)\sum_{i \in Neg(x^j)} \alpha_i^j\right] \leq (m - |H|)(2k+1) \tag{3.8}$$

We finally conclude that:

$$\mathbb{E}[f(x^{k+1}) - f^*] \leq \frac{mB^2(m-|H|)}{|H|\mu c}\frac{2k+1}{(k+1)^2} + \frac{4L\sigma^2}{(k+1)\mu c}\frac{m^2}{|H|^3} + L\frac{4m^2}{(k+1)|H|^2\mu^2c^2}|H^c|B^2 \tag{3.9}$$

$$\mathbb{E}[f(x^{k+1}) - f^*] \overset{\frac{2j+1}{j+1} \leq 2}{\leq} \frac{mB^2(m-|H|)}{|H|(k+1)\mu c} + \frac{4L\sigma^2}{(k+1)\mu c}\frac{m^2}{|H|^3} + L\frac{4m^2}{(k+1)|H|^2\mu^2c^2}|H^c|B^2. \tag{3.10}$$

□

3.3 An inner loop for α

Another logical variation of the baseline Algorithm 4, is instead of doing only one gradient step for α, we will permit it to run for a sufficient number of iterations each round, the

Algorithm 6 Adaptive mixing weights SGD with inner loop for α

Require: x^0 – starting point.

1: **for** $r = 0, 1, \ldots$ **do**
2: receive M stochastic gradients $\nabla f_i(x^r, \zeta_i^r)$ from all workers.
3: Run Algorithm 7 for K iterations and receive α^r
4: $x^{r+1} = x^r - \eta_x \sum_{i=0}^{M-1} \alpha_i^r \nabla f_i(x^r, \zeta_i^r)$
5: **end for**

Algorithm 7 Adaptive mixing weights SGD

Require: α^0 initial averaging vector. x^r current model state , $i \in [m], \nabla f_i(x^r, \zeta_i^r)$, Validation loss V.

1: **for** $l = 0, 1, \ldots, L$ **do**
2: $x^+ = x^r - \eta_x \sum_{i=1}^{m} \alpha_i^l f_i(x^r, \zeta_i^r)$
3: $\hat{\alpha}^{l+1} = \alpha^l - \eta_\alpha \frac{\partial V(x^+)}{\partial \alpha}$
4: Project α^{l+1} onto probability simplex
5: **end for**
6: **return** α^L as α^r

idea behind this variation is to find the solution of the following problem:

$$\underset{\alpha \in \mathcal{P}}{\operatorname{argmin}}\, h_r(\alpha) := f_{target}(x^r - \eta_x \sum_{i=1}^{m} \alpha_i \nabla f_i(x^r, \zeta_i^r)). \tag{3.11}$$

Saying differently, each time r the goal is to find the best mixing weights α^* to apply for the gradients received at time r. Concretely, the methods are as described in Algorithms 6 and 7.

3.3.1 Analysis

To analyze the method described in Algorithms 6 and 7, we will start by the presenting the following Lemma:

Lemma 3.3.1 *Let $r \geq 0$, and x^k is generated by Algorithm 6, denote $h_r(\alpha) := f(x^r - \eta_x \sum_{i=1}^{m} \alpha_i \nabla f_i(x^r, \zeta_i^r))$*

$$f \text{ is } L\text{-smooth} \implies h_r \text{ is } L_h\text{-smooth}, \tag{3.12}$$

$$f \text{ is convex} \implies h_r \text{ is convex}, \tag{3.13}$$

with $L_h := L\eta_x^2 m B^2$.

Proof: Let α^1 and α^2 be in $\mathbb{R}^m$.
denote $x^1 := x^k - \eta_x \sum_{i=1}^{m} \alpha_i^1 \nabla f_i(x^k, \zeta_i^k)$ and $x^2 := x^k - \eta_x \sum_{i=1}^{m} \alpha_i^2 \nabla f_i(x^k, \zeta_i^k)$

following the smoothness of f:

$$
\begin{aligned}
f(x^1) &\leq f(x^2) + \langle \nabla f(x^2), x^1 - x^2 \rangle + \frac{L}{2}\|x^2 - x^1\|^2 \\
&= f(x^2) + \eta_x \langle \nabla f(x^2), \sum_{i=1}^{m} (\alpha_i^2 - \alpha_i^1) \nabla f_i(x^k, \zeta_i^k) \rangle + \frac{L\eta_x^2}{2} \| \sum_{i=1}^{m} (\alpha_i^2 - \alpha_i^1) \nabla f_i(x^k, \zeta_i^k) \|^2 \\
&\overset{\| \sum_{i=1}^m a_i \|^2 \leq m \sum_{i=1}^m \|a_i\|^2}{\leq} f(x^2) + \eta_x \langle \nabla f(x^2), \sum_{i=1}^{m} (\alpha_i^2 - \alpha_i^1) \nabla f_i(x^k, \zeta_i^k) \rangle \\
&+ \frac{L\eta_x^2 m}{2} \sum_{i=1}^{m} (\alpha_i^2 - \alpha_i^1)^2 \|\nabla f_i(x^k, \zeta_i^k)\|^2
\end{aligned}
$$

$$
\leq f(x^2) + \eta_x \sum_{i=1}^{m} (\alpha_i^2 - \alpha_i^1) \langle \nabla f(x^2), \nabla f_i(x^k, \zeta_i^k) \rangle + \frac{L\eta_x^2 m B^2}{2} \|\alpha^1 - \alpha^2\|_2^2, \tag{3.14}
$$

From the other side:

$$
\begin{aligned}
\frac{\partial h_k(\alpha^2)}{\partial \alpha} &= \frac{\partial f(x^k - \eta_x \sum_{i=1}^m \alpha_i^2 \nabla f_i(x^k, \zeta_i^k))}{\partial \alpha} \\
&= \left[\frac{\nabla f(x^2)}{\partial x} \right]^T \left[\frac{\partial (x^k - \eta_x \sum_{i=1}^m \alpha_i^2 \nabla f_i(x^k, \zeta_i^k))}{\partial \alpha} \right] \\
&= -\eta_x \left[\frac{\nabla f(x^2)}{\partial x} \right]^T \left[\nabla f_1(x^k, \zeta_1^k), \ldots, \nabla f_m(x^k, \zeta_m^k)) \right] \\
&= -\eta_x \left[\langle \nabla f(x^2), \nabla f_1(x^k, \zeta_1^k) \rangle, \ldots, \langle \nabla f(x^2), \nabla f_m(x^k, \zeta_m^k) \rangle \right],
\end{aligned}
$$

Thus:

$$
\langle \frac{\partial h_k(\alpha^2)}{\partial \alpha}, \alpha^1 - \alpha^2 \rangle = -\eta_x \sum_{i=1}^{m} (\alpha_i^1 - \alpha_i^2) \langle \nabla f(x^2), \nabla f_i(x^k, \zeta_i^k) \rangle, \tag{3.15}
$$

Plugging 3.15 into 3.14: we get

$$
h_k(\alpha_1) \leq h_k(\alpha_2) + \langle \frac{\partial h_k(\alpha^2)}{\partial \alpha}, \alpha^1 - \alpha^2 \rangle + \frac{L\eta_x^2 m B^2}{2} \|\alpha^1 - \alpha^2\|_2^2.
$$

Therefore h_k is smooth with respect to α with parameter $L_h = L\eta_x^2 m B^2$.

The second statement of the lemma is easy enough to see. □

In addition to Lemma 3.3.1 that proved the smoothness of h_r is linked with the smoothness of f. We need an additional assumption of the strong convexity of h_r.

Assumption 3.3.2 *for any round $r \geq 0$, with x^r generated by Algorithm 6, the function h_r is strongly convex with parameter μ_h:*

$$
\forall \alpha^1, \alpha^2 \in \mathbb{R}^m : h_r(\alpha^1) \geq h_r(\alpha^2) + \langle \nabla h_r(\alpha^2), \alpha^1 - \alpha^2 \rangle + \frac{\mu_h}{2} \|\alpha^1 - \alpha^2\|^2
$$

Algorithm 8 DFO Adaptive mixing weights

Require: x^0 – starting point, α^0 initial averaging vector. Validation loss V

1: **for** $r = 0, 1, \ldots$ **do**
2: Receive M stochastic gradients $\nabla f_i(x^r, \zeta_i^r)$ from all workers
3: Run a defined DFO algorithm to find the best α
4: $x^{r+1} = x^r - \eta_x \sum_{i=0}^{M-1} \alpha_i^r \nabla f_i(x^r, \zeta_i^r)$
5: **end for**

3.4 DFO to preserve privacy

A major flaw in our proposed approach is the added requirement by the server of having an unbiased validation set of the wanted stochastic objective. This violates one of the premises on which standard FL is built: **Interdiction of raw data sharing**. To solve this, we propose a variation where we use a *Derivative Free Optimization* (DFO) algorithm to update α. This change eliminates the need for a validation set to construct a differentiable validation function. Instead, depending on the used DFO algorithm, the server queries the worker that has f_{target} as objective with candidates model states $\{x^j\}, j \in [\text{candidates } \#]$ formed with candidate α's, after which the worker responds with the validation loss value, and then the DFO algorithm is executed to find the best α among the candidates. We refer the reader to [21] for a comprehensive survey on DFO literature. Concretely, the baseline Algorithm turns into Algorithm 8.

Chapter 4

Experiments

The goal behind the experiments is to illustrate the harm caused by byzantine workers when giving the updates received by them the same weights as those received from helpful workers. as well as how our approach eliminates their effect.

We perform experiments using a toy example and real datasets with multiple byzantine behaviors to show that the method converges towards an optimum in both parameters and mixing weight spaces. The experiments include differentiation and derivative-free methods. all experiments were done within *PyTorch 1.13.0* environment , using *MNIST* dataset and the automatic differentiation therein.

4.1 Experiments on synthetic data

Setup: we consider a two-workers case for visualization purposes and assume *target* has logistic loss: $f_{target}(w) = \frac{1}{n}\sum_{j=1}^{n} \log(1 + \exp(-y_{ij}a_{ij}^T w))$, where a is the data matrix stored in *target* with $n = 1000$ and $d = 2$, data points and feature space dimension, respectively. and y is the label vector of size n, *worker 2* has their own data matrix and label vector $\bar{a}, \bar{y}$ with the same shapes. we artificially shift apart the point with 1 and -1 labels by adding two different gaussian vectors to each label. To simulate byzantine behavior, we switch the added vectors for *worker 2*. Practically, making points of label 1 in *target* similar to points of label -1 in *worker 2* and vice versa.

To learn the best mixing weights using our approach, we reserve 20% of the data stored in *target* for validation, we use a shared step size $\eta_x = \eta_\alpha = 0.1$.

We compare SGD with uniform mixing weights and SGD with the weights learned using our method. The results are summarized in figure 4.2. We notice that mixing weights converge towards $(1, 0)$ as expected since *worker 2* is harmful by construction. And subsequently, we notice divergence of uniform SGD and convergence with learned optimal mixing weights.

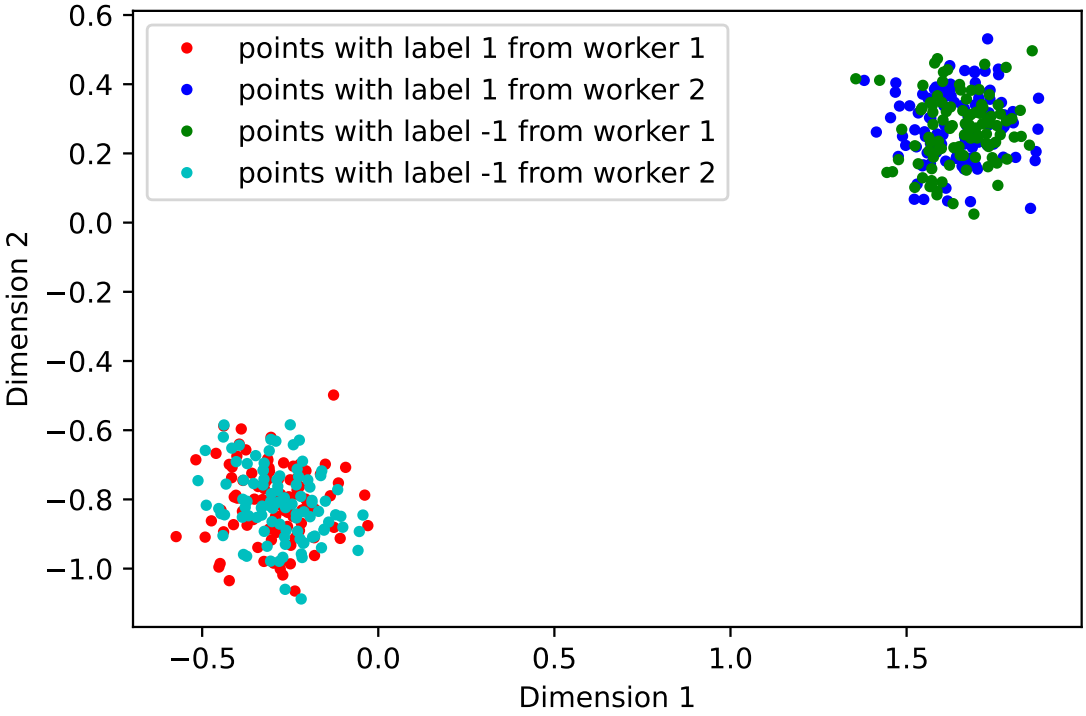
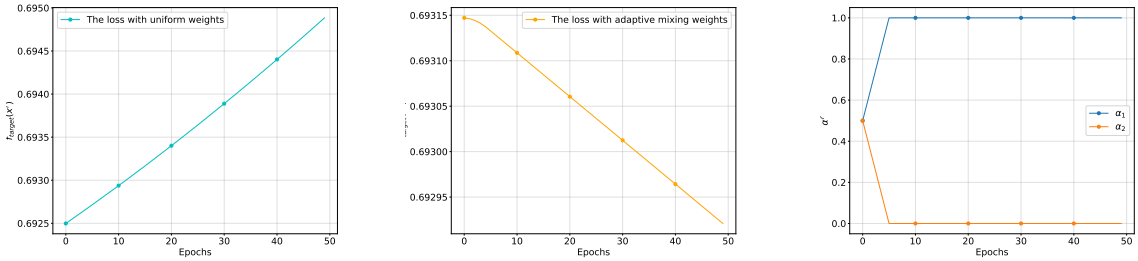


Figure 4.1: *target* and *Worker 2* have different data distributions (opposite), qualifying *worker 2* as harmful in the training process.

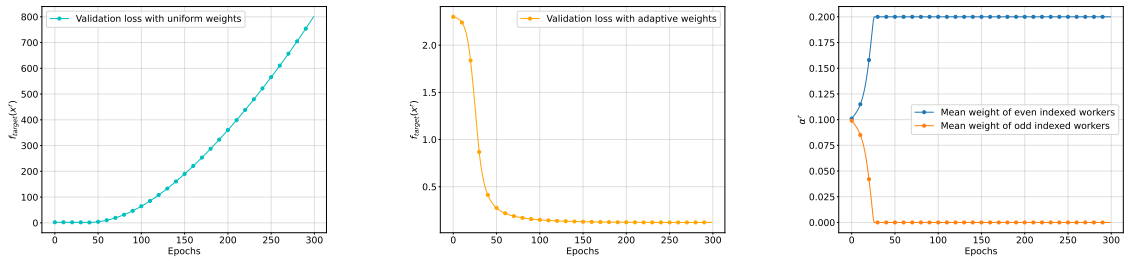


(a) Validation loss with uniform weights

(b) Validation loss with adaptive weights

(c) Mixing weights

Figure 4.2: SGD with uniform mixing weights (orange line) and with adaptive mixing weights (blue line), -step size for SGD with uniform alpha had to be chosen smaller for illustration purposes-



(a) Validation loss with uniform weights for MNIST Disjoint labels case

(b) Validation loss with adaptive weights for MNIST Disjoint labels case

(c) Mixing weights averages for MNIST Disjoint labels case

Figure 4.3: Odd-indexed workers holding different digits succeeded in straying the training from reaching an optimum and increasing the validation loss when uniform weights are used. adaptive weights gradually reduced their impact and canceled it completely very early in the training.

4.2 Experiments on real datasets

4.2.1 MNIST dataset

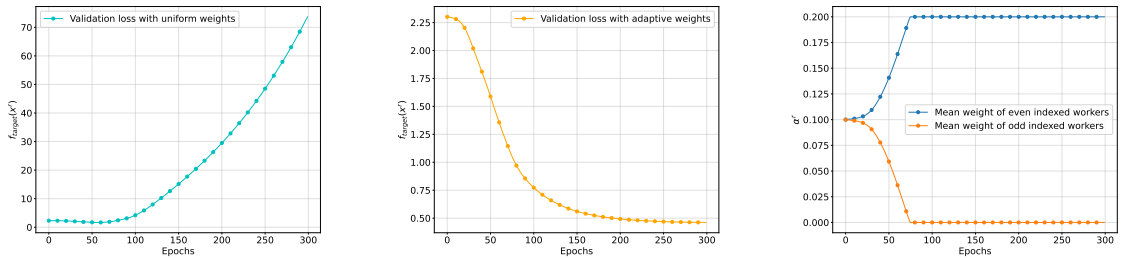
The experiments that were done using *MNIST* dataset [7] for digit classification represent a real-world deployment of the method to show its effectiveness in cases where only stochastic gradients are available. The experiments aim to optimize the weights of a 2-layer fully connected neural network, each layer has 64 nodes, generating a nonconvex loss function. 10 workers are involved in the training with such as workers holding *odd* indexes are harmful and those with *even* indexes are helpful (Under the convention that *target* hold index 0). During all the following experiments, a batch size of 10% of the data stored in each worker is used to generate the stochastic gradients, To simulate harmful workers, we test multiple scenarios that are listed in the following:

Disjoint labels

In this scenario, we provide even indexed workers with images of digits from 0 to 4 and the others with digits from 5 to 9. Obviously, odd-indexed workers have a different distribution from the even-indexed, and thus they are harmful. The results when experimenting with this scenario are summarized in Figure 4.3.

Shuffled labels

In this scenario, we provide images from all the digits to all the workers but shuffled the labels in the odd-indexed workers. effectively, making the data stored in them wrong and untrustworthy and qualifying them as harmful for the training. the results are summarized in Figure 4.4.



(a) Validation loss with uniform weights for MNIST shuffled labels case

(b) Validation loss with adaptive weights for MNIST shuffled labels case

(c) Mixing weights averages for MNIST shuffled labels case

Figure 4.4: Odd-indexed workers holding shuffled labels derailed the training when the uniform mixing weights, while adaptive weights insured convergence.

4.2.2 EMNIST Dataset

In this part, we experiment with the EMNIST dataset [5]. the dataset is an extension of the standard MNIST dataset. by adding letters in both lowercase and uppercase, this dataset is more complete. In this collaboration scenario, we set up a scenario where there are 4 workers, the first three workers hold images of digits, uppercase letters, and lowercase letters respectively. and the fourth worker is byzantine images with wrong labels. the validation set has images from all three categories and therefore to improve the validation accuracy all the workers have to collaborate. The model used is a neural network following ResNet18 [10] architecture. the results are resumed in Figure 4.5.

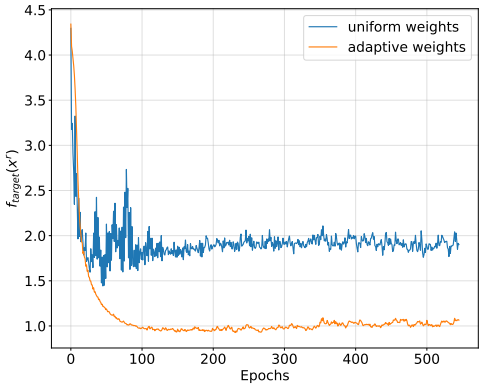
4.3 Experiments with variations of the baseline algorithm

All the experiments so far were implemented using the standard baseline Algorithm 4, This section presents the results using the proposed variations that each solve a particular issue with the baseline algorithm.

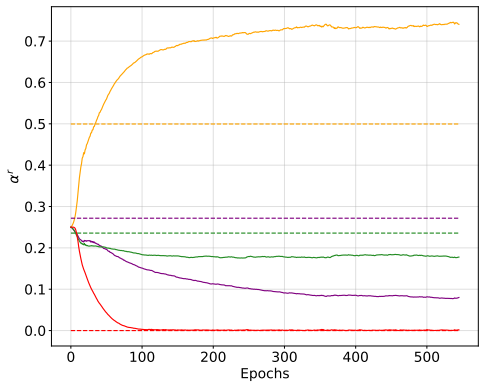
4.3.1 SignSGD for α

The experiments in this part were performed on MNIST dataset [7] with a 2-layer network architecture. The labels in half of the 10 involved workers (odd-indexed) are shuffled to simulate byzantine behavior. The results are presented in Figure 4.6.

4.3.2 Inner loop for α

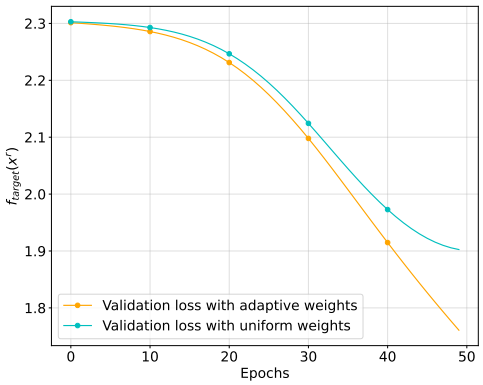


(a) With the existence of one harmful worker, The validation loss using uniform weights stagnated much earlier than with adaptive weights

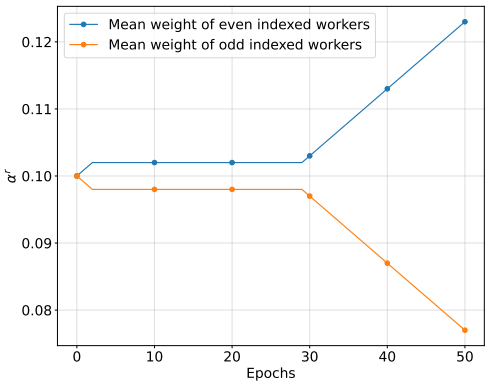


(b) The evolution of α throughout the training, the dashed lines show the optimal values for α calculated by the proportion of useful data stored in each worker.

Figure 4.5: Heterogeneity embedded in the updates received from worker 4, harmed the training greatly when using standard SGD.



(a) Using uniform weights widened the neighborhood of the minimum that the method converged to throughout the training.



(b) Using SignSGD to update α lead to a more smooth and monotonic evolution of α.

Figure 4.6: SignSGD update for α allowed the elimination of the effects of the heterogeneous/byzantine workers

References

[1] Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization, 2022.

[2] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Byzantine-tolerant machine learning. *CoRR*, abs/1703.02757, 2017.

[5] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

[6] Laurent Condat. Fast projection onto the simplex and the l 1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

[7] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[8] John David Funge. *Artificial intelligence for computer games: an introduction*. AK Peters/CRC Press, 2004.

[9] Pavel Hamet and Johanne Tremblay. Artificial intelligence in medicine. *Metabolism*, 69:S36–S40, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning, 2019.

[12] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[13] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning, 2019.

[14] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

[15] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. 2016.

[16] Daniel B. Neill. Using artificial intelligence to improve hospital inpatient care. *IEEE Intelligent Systems*, 28(2):92–95, 2013.

[17] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[18] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[19] Solmaz Niknam, Harpreet S. Dhillon, and Jeffrey H. Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, 2020.

[20] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119, 2020.

[21] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[22] Khe Sim, Angad Chandorkar, Fan Gao, Mason Chua, Tsendsuren Munkhdalai, and Françoise Beaufays. Robust continuous on-device personalization for automatic speech recognition. pages 1284–1288, 08 2021.

[23] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[24] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization, 2019.

[25] Mariel Werner, Lie He, Sai Praneeth Karimireddy, Michael Jordan, and Martin Jaggi. Towards provably personalized federated learning via threshold-clustering of similar clients. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[26] Shanshan Wu, Tian Li, Zachary Charles, Yu Xiao, Ken Liu, Zheng Xu, and Virginia Smith. Motley: Benchmarking heterogeneity and personalization in federated learning. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[27] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 20(3):1935–1949, 2020.

APPENDICES

Appendix A

Algorithms

A.1 Federated Averaging

Algorithm 9 FedAvg: Federated Averaging

Require: x^0 Starting point, Step size η, Number of local steps L.

for $k = 0, 1, \ldots$ **do**
  Sample a subset of clients $\mathcal{S} \subset [n]$.
  Broadcast current model state x to $\mathcal{S}$.
  on parallel for $i \in \mathcal{S}$ **do:**
    Initialize local models $y_i^0 \leftarrow x$.
    for $l = 0, \ldots, L$ **do**
      Generate a stochastic gradient g_i^l.
      $y_i \leftarrow y_i - \eta g_i^l$.
    end for
    Send $\Delta y_i \leftarrow y_i - x$ to server
  end parallel computation
  $x \leftarrow x + \frac{\eta}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta y_i$.
end for