entropy

MDPI

Article

Robust Aggregation for Federated Learning by Minimum γ-Divergence Estimation

Cen-Jhih Li [1], Pin-Han Huang [2], Yi-Ting Ma [1], Hung Hung [3] and Su-Yun Huang [1,*]

[1] Institute of Statistical Science, Academia Sinica, Taipei City 11529, Taiwan; licz82@stat.sinica.edu.tw (C.-J.L.); erics910085@gmail.com (Y.-T.M.)
[2] Data Science Degree Program, National Taiwan University, Taipei City 10617, Taiwan; peterkonerko@gmail.com
[3] Institute of Epidemiology and Preventive Medicine, National Taiwan University, Taipei City 10055, Taiwan; hhung@ntu.edu.tw
* Correspondence: syhuang@stat.sinica.edu.tw

Abstract: Federated learning is a framework for multiple devices or institutions, called local clients, to collaboratively train a global model without sharing their data. For federated learning with a central server, an aggregation algorithm integrates model information sent from local clients to update the parameters for a global model. Sample mean is the simplest and most commonly used aggregation method. However, it is not robust for data with outliers or under the Byzantine problem, where Byzantine clients send malicious messages to interfere with the learning process. Some robust aggregation methods were introduced in literature including marginal median, geometric median and trimmed-mean. In this article, we propose an alternative robust aggregation method, named γ-mean, which is the minimum divergence estimation based on a robust density power divergence. This γ-mean aggregation mitigates the influence of Byzantine clients by assigning fewer weights. This weighting scheme is data-driven and controlled by the γ value. Robustness from the viewpoint of the influence function is discussed and some numerical results are presented.

Keywords: byzantine problem; density power divergence; federated learning; γ-divergence; influence function; robustness

Citation: Li, C.-J.; Huang, P.-H.; Ma, Y.-T.; Hung, H.; Huang, S.-Y. Robust Aggregation for Federated Learning by Minimum γ-Divergence Estimation. *Entropy* **2022**, *24*, 686. https://doi.org/10.3390/e24050686

Academic Editor: Mohamed Medhat Gaber

Received: 30 March 2022
Accepted: 11 May 2022
Published: 13 May 2022

1. Introduction

Federated learning (FL), a distributed learning paradigm, was proposed by Google in 2016 [1] for distributing the optimization of model parameters over multiple machines. The distributed framework allows the data to be stored in local devices without sharing with each other. These machines, called clients, collaboratively train a global model. In this article, we consider the case that there is a central server, which works as a coordinator and aggregates the gradient information sent by the local clients to update the global model through an iterative process [2]. In real-world applications, clients, such as hospitals and clinical institutions, hold highly sensitive and private data such as electronic healthcare records, medical images, etc. [3]. However, FL still encounters several challenges in a real world setting. In its training process, some unreliable clients may produce outlying data information or even send malicious values [4], which leads to a biased result or even a failure of the training process. These arbitrary clients are called Byzantine clients. With a properly designed aggregation scheme, the effect by Byzantine clients can be reduced or excluded. Several papers have made significant contributions to developing Byzantine-resilience methods with specific use of robust aggregators, including marginal median [5], geometric median [6], and trimmed-mean [7]. Another aggregation framework is the Byzantine-robust stochastic aggregation method [8], which adds a regularization term in the objective function to handle the Byzantine and heterogeneous data problem.

In this article, we consider the case of FL, which consists of a central server and m clients. Assume that there is an α-fraction of Byzantine clients. Our goal is to minimize the following objective function:

$$L(\theta) = \sum_{i=1}^{m} \pi_i \mathbb{E}_{\eta_i \sim \mathcal{D}}[l_{\eta_i}(\theta)], \tag{1}$$

where $\theta \in \mathbb{R}^p$ is the global model parameter, $l_{\eta_i}(\theta)$ is the loss function of client i with local data η_i having an unknown distribution $\mathcal{D}$, and $\{\pi_i\}_{i=1}^m$ are the prior weights assigned to local clients. At the tth round of FL iterative updates, the server broadcasts the current parameter value $\theta^t \in \mathbb{R}^p$ to local clients. Normal clients faithfully compute an estimate of the local gradients $\mathbb{E}_{\eta_i \sim \mathcal{D}} \nabla l_{\eta_i}(\theta^t)$ and send the gradient information back to the server. On the contrary, Byzantine clients send arbitrary erroneous messages to obstruct the optimization process. The central server aggregates the gradient information from clients and updates the model parameter from θ^t to θ^{t+1}. At the end, the server should output an estimate of the optimal θ, which aims to minimize the objective loss (1). In this work we consider uniform client weights, i.e., $\pi_1 = \cdots = \pi_m = 1$. The main reason is that, under the Byzantine problem, the local sample sizes claimed by Byzantine clients might not be reliable.

Our main contributions of this work are listed below.

- We propose the γ-mean as a robust aggregator in federated learning. This robust aggregator mitigates the influence of Byzantine clients by assigning fewer weights. The weighting scheme is data-driven and controlled by the tuning parameter γ.
- We have a discussion on robustness from the influence function point of view. Benefits of adopting γ-mean can be seen from its influence function in comparison to other robust alternatives such as marginal median, geometric median and trimmed mean.
- The robustness of γ-mean is then verified through simulation study and real data experiments. The γ-mean based aggregation outperforms other robust aggregators such as marginal median, geometric median and trimmed mean.

The rest of the article is organized as follows. In Section 2, we review some related work in robust federated learning. In Section 3, we propose our main method for robust FL aggregation and provide some theoretical viewpoints by the influence function. In Section 4, we conduct an extensive simulation study. In Section 5, we provide some further numerical examples using MNIST, fashion MNIST and chest X-ray images (pneumonia). In Section 6, we make some concluding remarks.

2. Related Work

McMahan et al. introduced the *FedAvg* algorithm [9], which used the mean as its aggregator. The sample mean aggregation is a very popular FL framework [9,10]. However, the sample mean is known to be vulnerable to outliers and heavy-tailed distribution. For instance, Byzantine clients [7,8,11] may send extreme values to strongly influence the aggregation of sample mean. There are some robust alternatives to the mean aggregator, such as marginal median, geometric median and trimmed mean. In contrast to mean, marginal median is a relatively robust aggregator by computing coordinate-wise medians. Another Byzantine-resilient aggregator is geometric median, which computes the geometric median of $\{\boldsymbol{x}_i\}_{i=1}^m$ given by (see Weiszfeld's algorithm [12] for solution computation).

$$\text{GeoMed}(\{\boldsymbol{x}_i\}_{i=1}^m) = \arg\min_{\boldsymbol{x} \in \mathbb{R}^p} \sum_{i=1}^{m} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2. \tag{2}$$

The trimmed mean aggregator computes coordinate-wise trimmed means by removing the β-fraction of data points from each of the two tails [11]. Both the marginal median and geometric median, though are robust to outliers, but are not efficient, as they only use the most centrally-lying data for inference. As for trimmed mean, first it needs a proper selection of β. Second, it often does not work well for outliers lying on one side

of a coordinate instead of on two sides. To have a better balance between robustness and efficiency, we will propose another robust aggregator based on a robust density power divergence, namely, the γ-divergence.

3. Proposed Aggregator and Its Robustness

In this section, we introduce an aggregator, called "γ-mean", which is based on the minimum γ-divergence estimation. Two versions of implementation algorithms are given and some robustness viewpoints based on the influence function are provided.

3.1. Minimum γ-Divergence Estimation

The γ-divergence [13,14], which is also known as the density power divergence of type zero [15], is a robust divergence against outliers with a tuning parameter $\gamma > 0$. The γ-divergence between the data distribution with probability density function g and the model distribution f_τ (indexed by τ) is defined by

$$D_\gamma(g, f_\tau) = \frac{\|g\|_{\gamma+1}}{\gamma(\gamma+1)}\left\{1 - \int \left(\frac{f_\tau(\boldsymbol{x})}{\|f_\tau\|_{\gamma+1}}\right)^\gamma \frac{g(\boldsymbol{x})}{\|g\|_{\gamma+1}} d\boldsymbol{x}\right\},$$

where $\|f_\tau\|_{\gamma+1} = (\int f_\tau^{\gamma+1}(\boldsymbol{x})d\boldsymbol{x})^{1/(\gamma+1)}$. In the limiting case, it reduces to the Kullback-Leibler divergence, i.e., $\lim_{\gamma\to 0} D_\gamma(g, f_\tau) = \int \ln(\frac{g(\boldsymbol{x})}{f_\tau(\boldsymbol{x})}) g(\boldsymbol{x})d\boldsymbol{x}$. In the population level, the minimum γ-divergence estimation of τ is given by

$$\tau_\gamma^* = \arg\min_\tau D_\gamma(g, f_\tau) = \arg\max_\tau \int \left(\frac{f_\tau(\boldsymbol{x})}{\|f_\tau\|_{\gamma+1}}\right)^\gamma g(\boldsymbol{x})d\boldsymbol{x}. \tag{3}$$

In the sample level with empirical data $\{\boldsymbol{x}_i\}_{i=1}^m$, the data distribution will be replaced by the empirical distribution to get the estimate $\widehat{\tau}_\gamma$:

$$\widehat{\tau}_\gamma = \arg\min_\tau D_\gamma(g, f_\tau) = \arg\max_\tau \frac{1}{m}\sum_{i=1}^m \left(\frac{f_\tau(\boldsymbol{x}_i)}{\|f_\tau\|_{\gamma+1}}\right)^\gamma.$$

In this article, we use a Gaussian working model, i.e., $f_\tau \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Suppose g takes a contaminated form: $g(\boldsymbol{x}) = (1-\epsilon)f_\tau(\boldsymbol{x}) + \epsilon h(\boldsymbol{x})$, where $h(\boldsymbol{x})$ is the probability density function of contamination distribution. It is assumed that $\int f_\tau^\gamma(\boldsymbol{x})h(\boldsymbol{x})$ is small for certain $\gamma > 0$, so that the contamination has little effect in the learning process. By taking derivatives with respect to $\tau = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and setting them to zero, the estimates $(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})$ have to satisfy the following estimating equations:

$$\frac{1}{m}\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}) = 0,$$

$$\frac{1}{m}\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})\left\{\frac{1}{\gamma+1}\widehat{\boldsymbol{\Sigma}} - (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top\right\} = 0,$$

where $d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}) = \exp\left(\frac{-\gamma}{2}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top \widehat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})\right)$. The solution pair, $(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})$, has to satisfy the following stationary equations:

$$\widehat{\boldsymbol{\mu}} = \frac{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})\boldsymbol{x}_i}{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})}, \tag{4}$$

$$\widehat{\boldsymbol{\Sigma}} = (\gamma+1)\frac{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top}{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})}. \tag{5}$$

3.2. Robust Aggregation by γ-Mean

In view of the stationary Equations (4) and (5), we use a fixed-point iteration. Two algorithms are provided below. Algorithm 1 is the usual γ-mean, which uses $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as the working model, and Algorithm 2 is the simple γ-mean, which adopts $N(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$ as the working model. Note that, in the former case when the sample size is not sufficiently large enough to have a stable estimate of the covariance inverse, we will use $\{\text{diag}(\widehat{\boldsymbol{\Sigma}})\}^{-1}$ instead, where $\widehat{\boldsymbol{\Sigma}}$ is the minimum γ-divergence estimator for the covariance and $\text{diag}(\widehat{\boldsymbol{\Sigma}})$ denotes the diagonal matrix consisting of diagonal elements of $\widehat{\boldsymbol{\Sigma}}$. Also note that, in the latter case of simple γ-mean, σ^2 can be merged into the hyperparameter γ. Thus, we will simply use the standard Gaussian as the working model.

Algorithm 1 γ-mean with a Gaussian working model.

Input: Gradient information $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m]$ and the maximum number of iterations S
Output: $\boldsymbol{\mu}_\gamma = \widehat{\boldsymbol{\mu}}$

Start with initials $\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\mu}}_{ini}$ and $\widehat{\boldsymbol{\Sigma}} = \widehat{\boldsymbol{\Sigma}}_{ini}$.
for $s = 1, 2, \ldots, S$ (while $s \leq S$ and iterations not yet converge) **do**
  for $i = 1, 2, \ldots, m$ **do**
    Calculate $d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}) \leftarrow \exp\left(-\frac{\gamma}{2}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top \widehat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})\right)$ at the ith local client.
  end for
  Denote $w_i^{(\gamma)} = \frac{d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})}{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})}$.
  $\widehat{\boldsymbol{\mu}} \leftarrow \sum_{i=1}^m w_i^{(\gamma)} \cdot \boldsymbol{x}_i$,
  $\widehat{\boldsymbol{\Sigma}} \leftarrow (1+\gamma) \cdot \sum_{i=1}^m w_i^{(\gamma)} \cdot (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top$.
end for

Algorithm 2 simple γ-mean with the standard Gaussian as the working model.

Input: Gradient information $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m]$ and the maximum number of iterations S
Output: $\boldsymbol{\mu}_\gamma = \widehat{\boldsymbol{\mu}}$

Start with initial $\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\mu}}_{ini}$.
for $s = 1, 2, \ldots, S$ (while $s \leq S$ and iterations not yet converge) **do**
  for $i = 1, 2, \ldots, m$ **do**
    Calculate $d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}}) \leftarrow \exp\left(-\frac{\gamma}{2}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})^\top (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}})\right)$ at the ith local client.
  end for
  Denote $w_i^{(\gamma)} = \frac{d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}})}{\sum_{i=1}^m d_i^{(\gamma)}(\widehat{\boldsymbol{\mu}})}$.
  $\widehat{\boldsymbol{\mu}} \leftarrow \sum_{i=1}^m w_i^{(\gamma)} \cdot \boldsymbol{x}_i$.
end for

For the extremely large dimension p, such as in a deep neural network model, the simple γ-mean algorithm will be more feasible than the usual γ-mean for better numerical stability.

3.3. Robustness

3.3.1. Influence Function

The influence function is a tool to evaluate the change of an estimator by a small perturbation to the data distribution. An estimator with a lower influence provides better resistance against outliers. Therefore, we analyze the robustness of different estimators by showing their influence functions. Let $\{\boldsymbol{x}_i\}_{i=1}^m \subset \mathbb{R}^p$ be sampled from G and T be the statistical functional for estimation. The robustness of $T(G)$ can be evaluated by the influence function of T:

$$\text{IF}_T(\boldsymbol{x}_0; G) = \frac{\partial}{\partial \epsilon} T\{(1-\epsilon)G + \epsilon \delta_{\boldsymbol{x}_0}\}\Big|_{\epsilon=0},$$

where $\delta_{\boldsymbol{x}_0}$ is the point mass at $\boldsymbol{x}_0$. In other words, the influence function of T is the Gâteaux derivative of $T(G)$ with respect to G along the direction $\delta_{\boldsymbol{x}_0} - G$. In this paper, we consider evaluating the influence function at the Gaussian working model F_τ with $\tau = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and seeing the deviant effect when a point perturbation $\boldsymbol{x}_0$ is added. The influence function of M-estimators have been well-studied in [16], which is related to the inverse of Hessian matrix and the first order derivative. The estimator based on γ-divergence is also an M-estimate. Its influence function derivation can be found in [14] and is given by

$$\mathrm{IF}_{\boldsymbol{\mu}_\gamma}(\boldsymbol{x}_0; G)\big|_{G=F_\tau} = (\gamma+1)^{\frac{p+2}{2}} \exp\left(\frac{-\gamma}{2}(\boldsymbol{x}_0 - \boldsymbol{\mu}_\gamma)^\top \boldsymbol{\Sigma}_\gamma^{-1}(\boldsymbol{x}_0 - \boldsymbol{\mu}_\gamma)\right)(\boldsymbol{x}_0 - \boldsymbol{\mu}_\gamma), \quad (6)$$

where $\boldsymbol{\mu}_\gamma$ and $\boldsymbol{\Sigma}_\gamma^{-1}$ are the minimum γ-divergence estimators for the location and covariance matrix of the working model F_τ, respectively. The influence function of the γ-mean, $\mathrm{IF}_{\boldsymbol{\mu}_\gamma}(\boldsymbol{x}_0; F_\tau)$, is similar to the influence function of the sample mean, which is given by $(\boldsymbol{x}_0 - \boldsymbol{\mu})$, except for an additional multiplicative weight $d_{\boldsymbol{x}_0}^{(\gamma)}(\boldsymbol{\mu}_\gamma, \boldsymbol{\Sigma}_\gamma)$. This weight focuses on the Mahalanobis distance between the perturbing point $\boldsymbol{x}_0$ and $\boldsymbol{\mu}_\gamma$. As a result, when $\boldsymbol{x}_0$ is an outlier away from the target mean of data distribution, this weighting factor produces a down-weighting effect and prevents the estimator from biased estimation caused by the outlier.

3.3.2. Comparison with Other Aggregators

In this subsection, we present influence functions for several robust aggregation methods including marginal median, geometric median and trimmed-mean.

Geometric median $\boldsymbol{\mu}_{\mathrm{geo}}$: The influence function of $\boldsymbol{\mu}_{\mathrm{geo}}$ is given by [17]

$$\mathrm{IF}_{\boldsymbol{\mu}_{\mathrm{geo}}}(\boldsymbol{x}_0; G) = \left(\mathbb{E}_X\left[\|X - \boldsymbol{\mu}_{\mathrm{geo}}\|_2^{-1}\left\{I_p - \|X - \boldsymbol{\mu}_{\mathrm{geo}}\|_2^{-2}(X - \boldsymbol{\mu}_{\mathrm{geo}})(X - \boldsymbol{\mu}_{\mathrm{geo}})^\top\right\}\right]\right)^{-1} \frac{\boldsymbol{x}_0 - \boldsymbol{\mu}_{\mathrm{geo}}}{\|\boldsymbol{x}_0 - \boldsymbol{\mu}_{\mathrm{geo}}\|_2},$$

where the expectation E_X is taken with respect to the data distribution G.

Trimmed-mean $\boldsymbol{\mu}_{\mathrm{trm}}$: The trimmed-mean is an L-esimator, and its influence function is derived in [18]. For the influence function of the marginal trimmed-mean, we derive it coordinate-wise since the coordinates are independent to each other. Therefore, let G_i^{-1} be the quantile function of marginal data distribution on i-th component, and the influence function of $\boldsymbol{\mu}_{\mathrm{trm}}$ is given by

$$\mathrm{IF}_{\boldsymbol{\mu}_{\mathrm{trm}}}(\boldsymbol{x}_0; G) = \frac{1}{1-2\beta}\begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix},$$

where

$$w_i = \begin{cases} G_i^{-1}(\beta) - W_i, & x_{0i} < G_i^{-1}(\beta), \\ \boldsymbol{x}_0 - W_i, & G_i^{-1}(\beta) < x_{0i} < G_i^{-1}(1-\beta), \\ G_i^{-1}(1-\beta) - W_i, & x_{0i} > G_i^{-1}(1-\beta), \end{cases}$$

$\boldsymbol{x}_0 = (x_{01}, \ldots, x_{0p})^\top$, and

$$W_i = (1-2\beta)\int_{G_i^{-1}(\beta)}^{G_i^{-1}(1-\beta)} t\, dG_i(t) + \beta(G_i^{-1}(1-\beta) + G_i^{-1}(\beta)).$$

It is obvious that an influential effect, $\boldsymbol{x}_0 - W_i$, still remains if $\boldsymbol{x}_0$ is outside the trimmed range.

Marginal median $\boldsymbol{\mu}^{(0.5)}$: The influence function of the median is also derived in [18], and we use the same techniques as used in the trimmed-mean to derive the influence function of the marginal median. The influence function is given by

$$\mathrm{IF}_{\boldsymbol{\mu}^{(0.5)}}(\boldsymbol{x}_0; G) = \begin{pmatrix} \frac{x_{01}-\mu_1^{(0.5)}}{2g_1(\mu_1^{(0.5)})|x_{01}-\mu_1^{(0.5)}|} \\ \vdots \\ \frac{x_{0p}-\mu_p^{(0.5)}}{2g_p(\mu_p^{(0.5)})|x_{0p}-\mu_p^{(0.5)}|} \end{pmatrix}, \tag{7}$$

where $\boldsymbol{\mu}^{(0.5)} = (\mu_1^{(0.5)}, \ldots, \mu_p^{(0.5)})^\top$ is the marginal median and g_i is the density function of the marginal distribution G_i.

The influence functions given above reveal that different robust aggregators have different resistance ability against outliers. The trimmed mean can still be influenced by outliers outside the trimmed range. While the marginal median and the geometric median are fairly robust, they may lose too much efficiency due to the use of only the most centrally-lying data point. The γ-mean provides an adjustable control between the robustness and efficiency by the hyper-parameter γ.

4. Simulation Study

In this section, we evaluate our robust γ-mean aggregator and compare it with existing aggregators, including mean, marginal median, geometric median and trimmed mean, through simulation. For computing the geometric median, Weiszfeld's algorithm [12] is used. In this simulation study, we investigate the behavior and inspect the robustness of the γ-mean and other aggregators as an estimator of the location parameter under multivariate Gaussian distribution and multivariate t-distribution with and without Byzantine interference. We build several simulation scenarios, including testing aggregators across the growth of dimension p, different fractions (i.e., α values) of Byzantine attacks and for various settings of γ (which controls the degree of robustness). All our simulation experiments were run on an Nvidia DGX A100 server. We used one A100 GPU card, 16G RAM, and 16 cores of AMD-EPYC-7742 CPUs. However, a personal computer with a moderate performance-efficient GPU card and CPU can also carry out the simulation experiments, but with longer run time. For the real data examples in Section 5, a personal computer will not be sufficient for carrying out the computing job.

4.1. Simulation Settings

- Scenario 1.We focus on the behavior of aggregators for increasing p from 20 to 1000. Other experimental setting is as follows: the number of clients $m = 200$, the fraction of Byzantine attacks $\alpha = 0$ and $\alpha = 0.1$, and the hyper-parameter for controlling the robustness $\gamma = 2/p$.
- Scenario 2. We focus on the behavior of aggregators for increasing contamination fraction α from 0 to 0.5. Other experimental setting is as follows: $m = 200$, $p = 1000$ and $\gamma = 2/p$.
- Scenario 3. We focus on the effect of γ values and set $\gamma = c/p$ for various constants c ranging from 0.5 to 4. Other experimental setting is as follows: $m = 200$, $\alpha = 0.1$ and p ranges from 1 to 1000.
- Scenario 4. After comparison between the γ-mean and other aggregators, we focus on the comparison between two versions of our proposal, the γ-mean versus the simple γ-mean. Other experimental setting is as follows: $m = 200$, $\alpha = 0.1$, $\gamma = 2/p$ and p ranges from 1 to 1000.

For regular clients, gradient vectors are generated from the standard Gaussian distribution and t-distribution with 5 degrees of freedom. For Byzantine clients, gradient vectors are generated from the same Gaussian and t distributions, except with a location

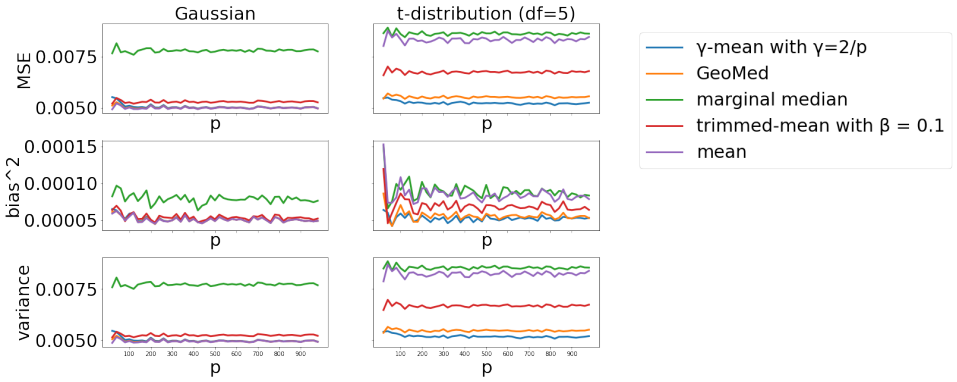
shift $\boldsymbol{\mu} = 100 \times \mathbf{1}_p$, where $\mathbf{1}_p$ is a vector with one in all entries. Each experimental scenario is implemented with 100 replicate runs.

4.2. Results

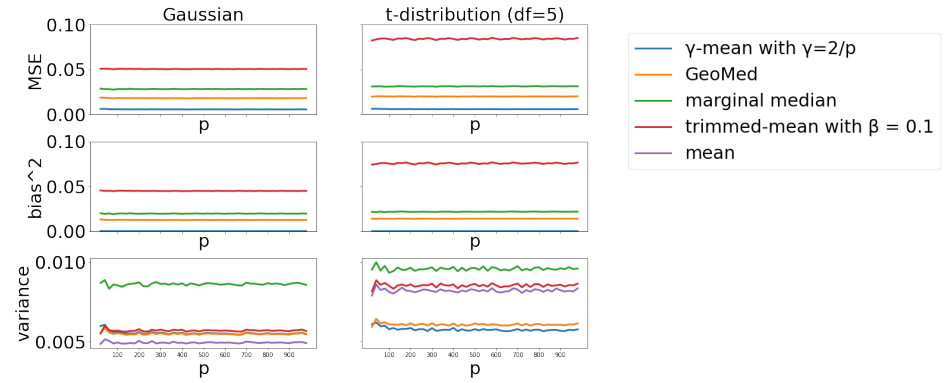
To compare the performance of different aggregators, the mean squared error (MSE) to the true target value is used as a performance metric. MSE can be further decomposed into the squared bias and variance.

4.2.1. Scenario 1

The results are shown in Figure 1. Without contamination (Figure 1a), all aggregators have MSE close to zero. For the Gaussian case, the MSE curves of the mean, geometric median and γ-mean are almost collapsed together. These 3 curves have the lowest MSE values followed by the trimmed-mean and marginal median. For the t-distribution, the γ-mean has the lowest MSE, followed by the geometric median, trimmed-mean, mean and then marginal median. With 10% of contamination, the mean-aggregator fails to perform well and has large values of MSE, verifying that it is not a robust aggregator. Due its large MSE value, the result of the mean-aggregator is removed from Figure 1b. From Figure 1b we can see that γ-mean performs the best among the five aggregators in terms of MSE. In addition, γ-mean remains low and stable in variance under both Gaussian and t distributions.



(a) $\alpha = 0$



(b) $\alpha = 0.1$

Figure 1. Comparison of different aggregators across different dimensions p. (**a**) Case $\alpha = 0$ (no Byzantine client). (**b**) Case $\alpha = 0.1$ (10% Byzantine clients).

4.2.2. Scenario 2

As Figure 2 shows, values of squared bias are close to zero when $\alpha = 0$ and increase as α grows for all aggregators. However, the increasing velocity of aggregators are quite different. Mean-aggregator surges with the highest velocity, while γ-mean progresses with the lowest velocity. Trimmed-mean slowly grows when $\alpha \leq 0.1$; yet, its increasing pace is about the same as the mean after $\alpha > 0.1$ due to the fixed 10% trimming percentage from both tails. The trends in MSE for different aggregators have similar patterns, and the only difference is in the scale with the γ-mean having the lowest MSE values. Also note that the variance of the marginal median increases much faster than other aggregators as α increases, followed by the geometric median, γ-mean, trimmed mean and then the mean. However, the trimmed mean and the mean have dominant bias leading to high MSE, even though they have small variance. As the values of squared bias are large for the mean and trimmed-mean, the bias curves for the γ-mean, geometric median and marginal median in the 2nd row of Figure 2 look collapsed together. To have a better view of these three bias curves, zoomed-in views are provided in the 3rd row of Figure 2, and we can clearly see that the γ-mean has the lowest bias values.

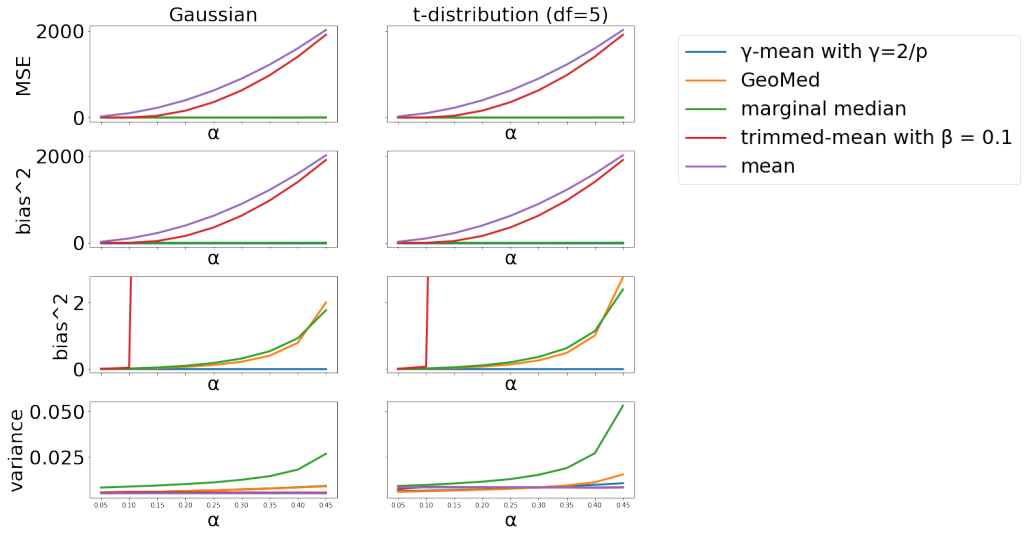


Figure 2. Comparison of different aggregators across different α values with fixed $p = 1000$. The two subplots in the 3rd row are the zoomed-in views of subplots in the 2nd row.

4.2.3. Scenario 3

We conduct further experiments on γ-mean to see the effect of different γ values by setting $\gamma = c/p$ with $c = 0.5, 1, 2, 3, 4$, where $\alpha = 0.1$ and $p \in [1, 1000]$. Results are presented in Figure 3. The squared bias is ignorable, indicating that the γ-mean is quite robust with different γ-values. The main source of MSE comes from the variance. Larger γ values lead to larger stochastic variances, indicating that larger γ values result in lower estimation efficiency.

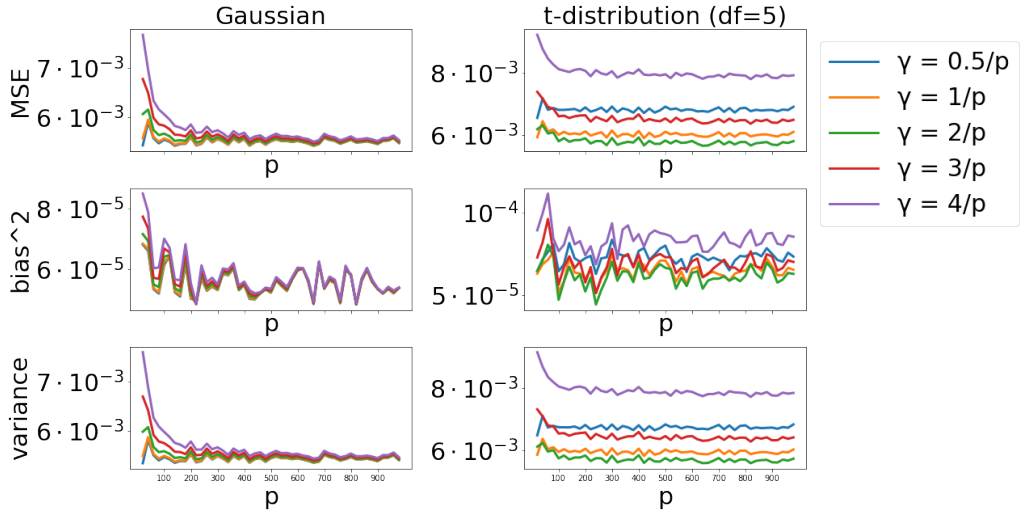


Figure 3. Effect of different γ values across different dimensions $p \in [1, 1000]$ with $\alpha = 0.1$.

4.2.4. Scenario 4

For $p \in [100, 1000]$ and under the Gaussian case, the MSE, bias and variance curves for the γ-mean and simple γ-mean are collapsed together (Figure 4, left panel). For $p \in [100, 1000]$ and under the t-distribution, these curves are nearly indistinguishable (Figure 4, right panel). When p is small ($p \ll 100$), the MSE values of the simple γ-mean are significantly larger than those of γ-mean. However, we did not display the results for $p < 100$, since the relative large MSE values for small p will make the MSE curves of γ-mean and simple γ-mean look lying on the x-axis for $p \geq 100$.

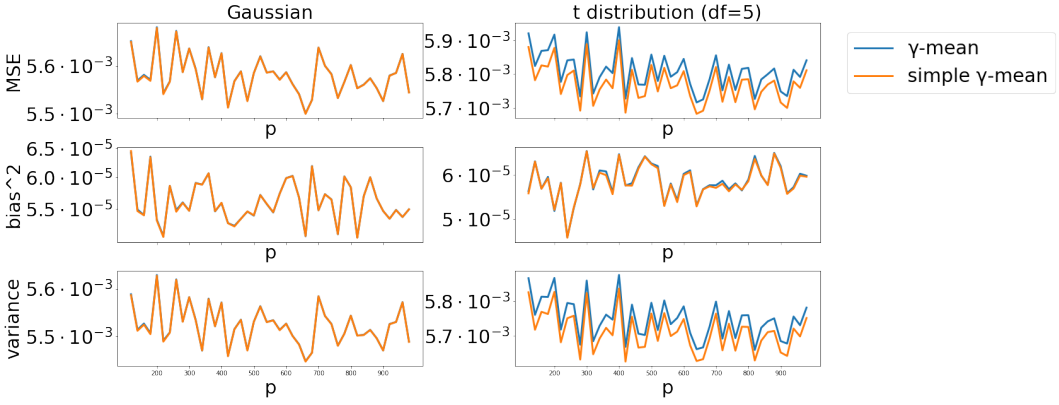


Figure 4. Comparison of γ-mean versus simple γ-mean with $\alpha = 0.1$ and $p \in [100, 1000]$.

5. Real Data Examples

5.1. Datasets

- MNIST [19]. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size, 28 × 28 grayscale, images.
- Fashion MNIST [20]. Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28 × 28 grayscale image, associated with a label from 10 types of clothing, such as shoes, t-shirts, dresses, sandals, sneakers and more.
- Chest X-ray images (pneumonia) [21]. The dataset contains 5856 X-ray images and 2 classes (pneumonia and normal). The 5,856 images consist of 5232 training images (which we further split into 90% for model training and 10% for model validation to implement early stopping) and 624 testing images. Chest X-ray images (anterior-

posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou.

5.2. Experimental Setting

All the data examples were run on an Nvidia DGX A100 server. We used one A100 GPU card, 64 G RAM, and 16 cores of AMD-EPYC-7742 CPUs. The experimental setting is described below.

- For MNIST and fashion MNIST, we set $m = 20$ and the number of Byzantine clients is two. For chest X-ray images, we set $m = 9$ and the number of Byzantine clients is one. Byzantine clients return random values from Gaussian (5, 1).
- γ is set to 0.5. This setting is different from the setting $\gamma = c/p$ in simulation. The main reason is that there is a certain complicated relationship between the γ value and the neural network model adopted, such as the dimensionality, gradient size, learning rate, etc. We have not yet fully understood this relationship, which might govern the selection of γ. We will leave it as a future study.
- We set $\beta = 0.1$ for the trimmed mean.
- To allow for the imbalanced size of clients, we obtain the sample size of each client by sampling from the following steps [11].
  1. Sampling a vector $\boldsymbol{a}$ from Lognormal$(1.5, 3.45^2)$.
  2. Sampling $\boldsymbol{\eta}$ from Dirichlet$(\boldsymbol{a})$. The sum of vector $\boldsymbol{\eta}$ will be 1 due to the property of Dirichlet distribution.
  3. Obtain the sample sizes of clients from multinomial $(n - m \times n_o, \boldsymbol{\eta})$, where n is total sample size and n_o is the minimum sample size guaranteed for each client. We set $n_o = 512$.

In each round of parameter update, the server broadcasts the current parameter values to m clients. Each normal client computes the local gradient and local parameter update, and then returns the change of parameters after iterating k epochs using local data ($k = 1$ for MNIST and fashion MNIST, and $k = 20$ for chest X-ray images). Byzantine clients return random values from a Gaussian distribution $N(5, 1)$. Some further implementation details are given below.

- We run 1000 rounds of FL for MNIST and fashion MNIST, and 100 rounds for chest X-ray images.
- We apply stochastic gradient descent (SGD) with a cosine decay learning rate (decay over rounds), where the decay step is 1000 for MNIST and fashion MNIST, and 100 for chest X-ray images. In each epoch of local clients on MNIST and fashion MNIST, the SGD will go through only 10% of local data to save computing time. This implementation leads to some fluctuations in the early stage of training but the training process will be much faster than going through all local data.
- The initial learning rates are 0.1, 0.5, and 10^{-4} for MNIST, fashion MNIST, and chest X-ray images, respectively. In each epoch in local iterates, 10^{-5} is set as the decay constant.
- We apply gradient clipping to avoid exploding gradients on MNIST and fashion MNIST. If the 2-norm of aggregated gradient is larger than 1, the vector will be scaled to a new vector with norm 1.
- To handle the imbalanced class size, we use weighted cross-entropy as loss function in the chest X-ray example (pneumonia: 0.35, normal: 1.0), where the chest X-ray training dataset contains 3883 pneumonia cases and 1349 normal cases. In addition to the classification accuracy, we also use 'accuracy', 'sensitivity' (also known as 'recall' and 'true positive rate') and 'precision' as our evaluation metrics. In particular, correctly predicting pneumonia is more important than predicting the normal case.

5.3. Models

We use a simple CNN model (Figure A1 in Appendix A) for MNIST and fashion MNIST. For chest X-ray images (pneumonia), we use a pretrained ResNet50 (Figure A2 in Appendix A) with input image size $150 \times 150 \times 3$. We further connect the model to 3 dense-block layers.

5.4. Results

Results for the trimmed mean are not reported here, as they do not perform well due to its reliance on the specification of β and its non-robustness to non-symmetric outliers (note that the trimming procedure is symmetric in two tails of each coordinate).

5.4.1. MNIST

Figure 5 shows the results of FL using different aggregators. In the case of no attack, the simple γ-mean performs as good as the mean-aggregator with 96% testing data classification accuracy. In the Byzantine case, simple γ-mean performs the best with testing accuracy 96% followed by the two median-based methods, geometric median and marginal median, while the mean-aggregator has failed. In addition, the marginal median has some fluctuations in the early stage of training.

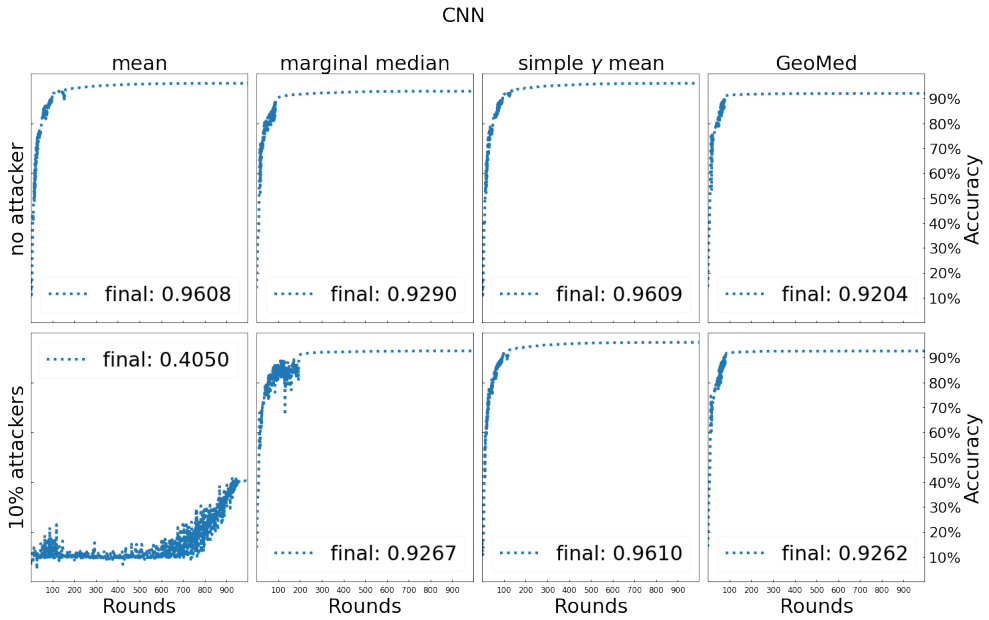


Figure 5. Testing process and comparison of testing accuracy for different aggregators on MNIST.

5.4.2. Fashion MNIST

Figure 6 shows the results of fashion MNIST. The conclusion is similar to MNIST, that the simple γ-mean performs the best. Note that the fluctuations shown in the training process of marginal median look worse than those shown in Figure 5. Although we have applied gradient clipping, still there are some sudden surges in loss function values and they lead to sudden drops in classification accuracy.

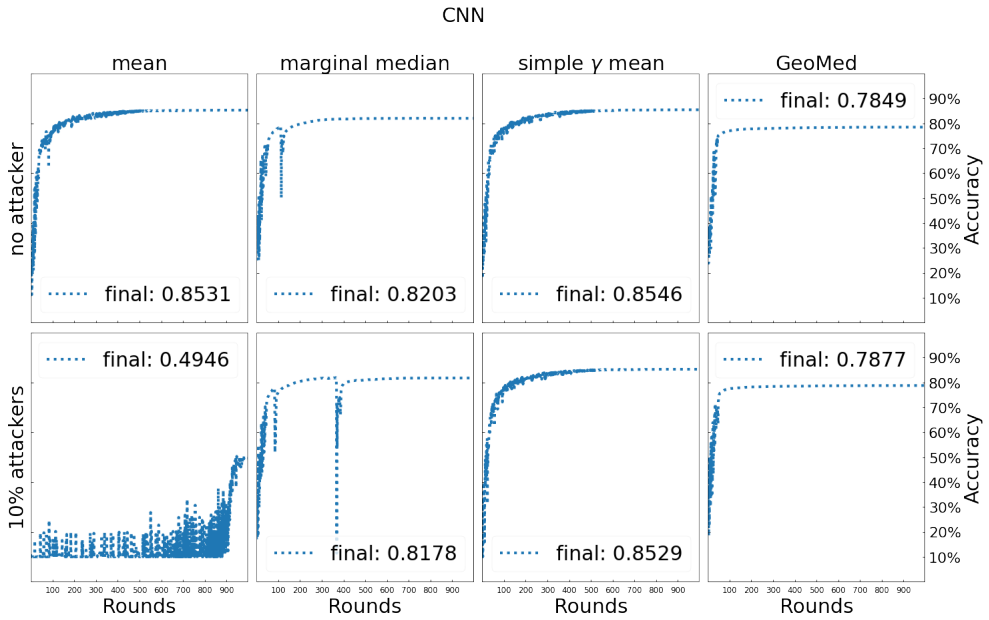


Figure 6. Testing process and comparison of testing accuracy for different aggregators on fashion MNIST.

5.4.3. Chest X-ray Images (Pneumonia)

We train the model on the training dataset without any partitions (i.e., single machine with full training data) as a baseline. In this single machine case, the maximum number of training epochs is 1000 and the training process will stop early if the evaluation metric (loss + accuracy + precision + sensitivity on validation data) does not get better in 10 consecutive epochs. For the FL case, the number of FL rounds is 100 and the maximum training epochs of each client are 20 at every FL round. At each round, the clients will stop early if the evaluation metric (loss + accuracy + precision + sensitivity on validation data) does not get better in 5 consecutive epochs. The simple γ-mean algorithm has adopted the marginal median as initial. All other settings are the same as those listed in Section 5.2. The results are shown in Figure 7.

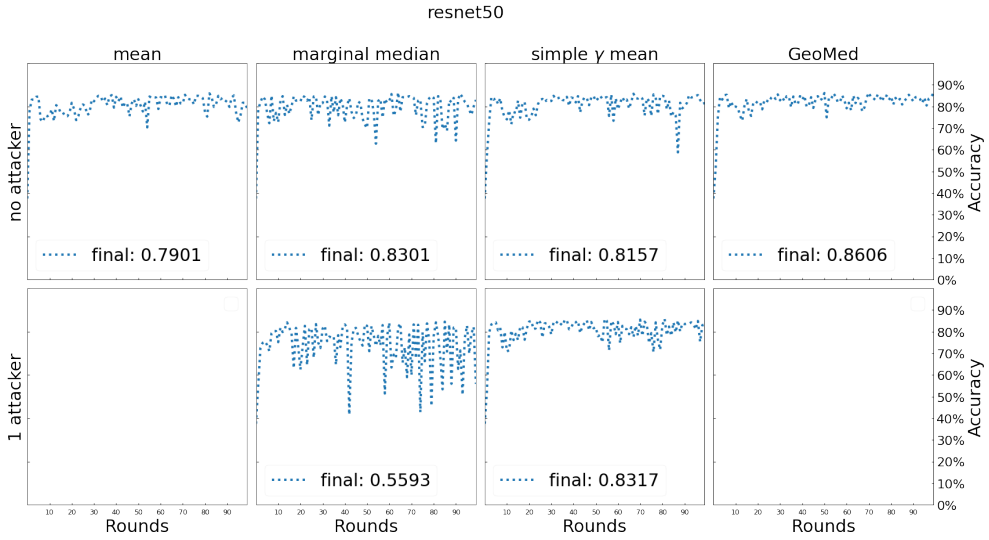


Figure 7. Testing process and comparison of testing accuracy for different aggregators on chest x ray. The aggregators by mean and geometric median cannot tolerate the Byzantine attack and both methods were crushed during model training. Thus, there are no results reported for these two methods.

We further report the following testing data results in Table 1: true positive (TP, which is predicted positive and actually positive), true negative (TN, which is predicted negative and actually negative), false positive (FP, which is predicted positive but actually negative), false negative (FN, which is predicted negative but actually positive), precision (Prec, which is $\frac{TP}{TP+FP}$), sensitivity (Sens, which is $\frac{TP}{TP+FN}$ as well as one minus type-II error) and classification accuracy (Acc). The aggregators by mean and geometric median cannot tolerate the Byzantine (Byz) attack and both methods were crushed during model training. Thus, there are no results reported for these two methods.

Table 1. Pneumonia prediction on test data.

Byz	Aggregator	TN	FN	FP	TP	Prec	Sens (Type II Error)	Acc
	single machine	156	23	78	367	0.8247	0.9410 (0.0590)	0.8381
No	mean	212	103	22	287	0.9288	0.7359 (0.2661)	0.7997
	marginal median	190	63	44	327	0.8814	0.8385 (0.1615)	0.8285
	simple γ-mean	126	8	108	382	0.7796	0.9795 (0.0205)	0.8141
	GeoMed	177	30	57	360	0.8633	0.9231 (0.0769)	0.8606
Yes	mean †	–	–	–	–	–	–	–
	marginal median	228	271	6	119	0.9520	0.3051 (0.6949)	0.5561
	simple γ-mean	140	11	94	379	0.8013	0.9718 (0.0282)	0.8317
	GeoMed †	–	–	–	–	–	–	–

† The symbol "–" indicates model crushed during training.

6. Concluding Remarks

Our major contribution in this article is to propose the γ-mean aggregation in federated learning, which is robust against outliers and Byzantine attacks. We have provided some theoretical discussions on influence functions and carried out numerical studies to justify our proposal. In both simulation and real data experiments, γ-mean based aggregation in general outperforms the existing robust aggregators such as marginal median, geometric median and trimmed mean. In addition, the simple γ-mean does not require complex computation. It can be easily computed by fixed point iteration and works well for extremely high dimensional cases such as in a deep neural network model. The γ-mean (as well as the simple γ-mean) takes a form of weighted average, where γ controls the trade-off between robustness and estimation efficiency. The selection of γ value is important, and a data-driven selection procedure is needed, which will be pursued in our future study.

In our numerical examples (simulation and real data), we have tested γ-mean's ability to guard against the Byzantine attacks. However, when it comes to federated learning in real-world applications, adversarial attacks are also an important issue for model robustness. These attacks create adversarial fake examples, which even humans might not be able to distinguish from genuine ones. Moreover, adversarial examples aim to confuse the model and result in wrong prediction. How to modify the baseline γ-mean algorithm for adversarial attacks will be an interesting and yet challenging problem. We will defer it to a future study.

Author Contributions: Conceptualization, S.-Y.H.; methodology, S.-Y.H.; C.-J.L. and H.H.; software, C.-J.L. and P.-H.H.; formal analysis, C.-J.L.; P.-H.H.; Y.-T.M.; writing—original draft preparation, C.-J.L.; P.-H.H.; Y.-T.M.; writing—review and editing, C.-J.L.; P.-H.H.; Y.-T.M.; H.H. and S.-Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Benchmark datasets are available from the following links: "MNIST" at http://yann.lecun.com/exdb/mnist/ accessed on 31 August 2021, "fashion MNIST" at https://research.zalando.com/project/fashion_mnist/fashion_mnist/ accessed on 31 August 2021, and

"Chest X-ray images (pneumonia)" at https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia accessed on 1 October 2021.

Acknowledgments: Su-Yun Huang thanks the Institute of Statistical Science, Academia Sinica, for a seed grant funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Acc	accuracy
Byz	Byzantine
FL	Federated learning
FN	false negative, predicted negative but actually positive
FP	false positive, predicted positive but actually negative
GeoMed	Geometric median
IF	Influence function
Prec	precision, $\frac{TP}{TP+FP}$
Sens	sensitivity, $\frac{TP}{TP+FN}$ or 1−type II error
TN	true negative, predicted negative and actually negative
TP	true positive, predicted positive and actually positive

Appendix A

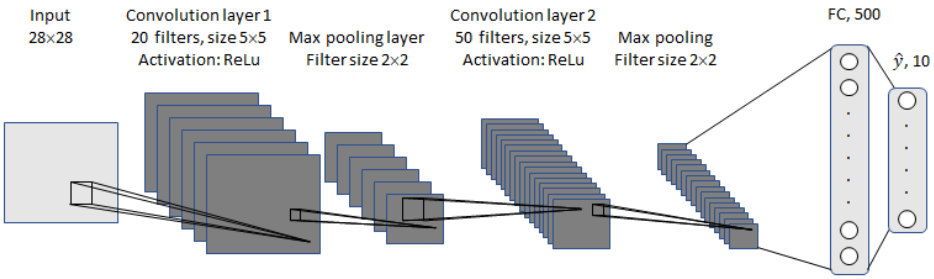


Figure A1. Model used in MNIST and fashion MNIST.

For MNIST and fashion MNIST, we use a model similar to LeNet-5. All filters in the convolution layers are 5×5 maps without padding. The first layer in fully connection is a dense layer with 500 neurons with ReLU activation. The last layer is an output layer with 10 neurons with softmax activation. The number of parameters is 931,080.

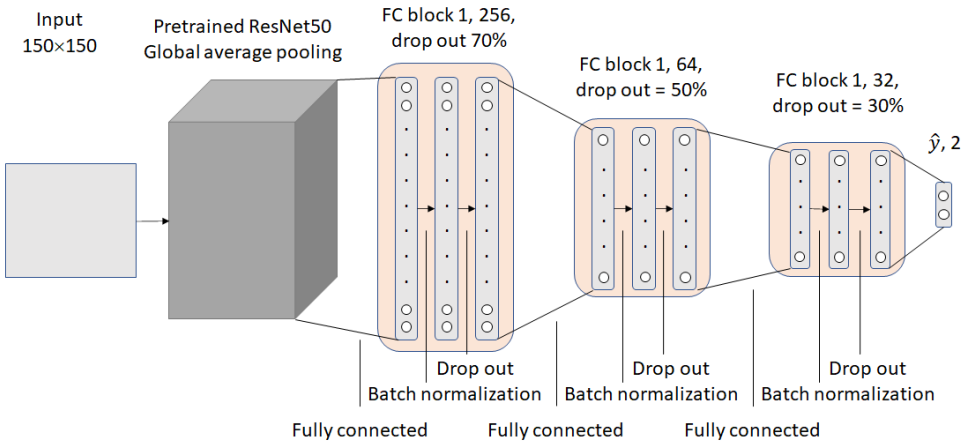


Figure A2. Model used in chest X-ray images.

For chest X-ray images, we use a pretrained ResNet50 with input image size is $150 \times 150 \times 3$ and freeze all ResNet50 parameters. The output dimensionality is $5 \times 5 \times 2048$. A global average pooling is adopted and it leads to the dimensionality 2048. Next, we connect it to 3 dense-block layers. Dense block: dense layer $\rightarrow$ batch normalization $\rightarrow$ drop out. Number of neurons are 256, 64, and 32, respectively. The drop out probabilities are 0.7, 0.5, and 0.3, respectively. The number of trainable parameters is 543,809.

References

1. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtarik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. In Proceedings of the NeurIPS Workshop on Private Multi-Party Machine Learning; 2016. Available online: https://nips.cc/Conferences/2016/ScheduleMultitrack?event=6250 (accessed on 29 March 2022).
2. So, J.; Güler, B.; Avestimehr, A.S. Byzantine-resilient secure federated learning. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 2168–2181. [CrossRef]
3. Xu, J.; Glicksberg, B.S.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated learning for healthcare informatics. *J. Healthc. Inform. Res.* **2021**, *5*, 1–19. [CrossRef] [PubMed]
4. Alistarh, D.; Allen-Zhu, Z.; Li, J. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2018; p. 31.
5. Chen, X.; Chen, T.; Sun, H.; Wu, S.Z.; Hong, M. Distributed training with heterogeneous data: Bridging median- and mean-based algorithms. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; p. 33.
6. Chen, Y.; Su, L.; X, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. Acm Meas. Anal. Comput. Syst.* **2017**, *1*, 1–25. [CrossRef]
7. Xie, C.; Koyejo, O.; Gupta, I. Generalized Byzantine-tolerant SGD. *arXiv* **2018**, arXiv:1802.10116.
8. Li, L.; Xu, W.; Chen, T.; Giannakis, G.B.; Ling, Q. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. *Proc. Aaai Conf. Artif. Intell.* **2019**, *33*, 1544–1551. [CrossRef]
9. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 10 April 2017; Volume 54, pp. 1273–1282.
10. Dayan, I.; Roth, H.R.; Zhong, A.; Harouni, A.; Gentili, A.; Abidin, A.Z.; Li, Q. Federated learning for predicting clinical outcomes in patients with covid-19. *Nat. Med.* **2021**, *27*, 1735–1743. [CrossRef] [PubMed]
11. Portnoy, A.; Tirosh, Y.; Hendler, D. Towards Federated Learning with Byzantine-Robust Client Weighting. In Proceedings of the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML, 2021. Available online: https://federated-learning.org/fl-icml-2021/ (accessed on 29 March 2022).
12. Weiszfeld, E.; Plastria, F. On the point for which the sum of the distances to n given points is minimum. *Ann. Oper. Res.* **2009**, *167*, 7–41. [CrossRef]
13. Fujisawa, H.; Eguchi, S. Robust parameter estimation with a small bias against heavy contamination. *J. Multivar. Anal.* **2008**, *99*, 2053–2081. [CrossRef]
14. Hung, H. A robust removing unwanted variation–testing procedure via γ-divergence. *Biometrics* **2019**, *75*, 650–662. [CrossRef] [PubMed]
15. Jones, M.C.; Hjort, N.L.; Harris, I.R.; Basu, A. A comparison of related density-based minimum divergence estimators. *Biometrika* **2001**, *88*, 865–873. [CrossRef]
16. Huber, P.J. *Robust Statistics*; John Wiley & Sons: Hoboken, NJ, USA, 2004.
17. Chaudhuri, P. On a geometric notion of quantiles for multivariate data. *J. Am. Stat. Assoc.* **1996**, *91*, 862–872. [CrossRef]
18. van der Vaart, A. W. *Asymptotic Statistics*; Cambridge Series in Statistical and Probabilistic Mathematics; Cambridge University Press: Cambridge, UK, 1998.
19. Deng, L. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
20. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST a novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
21. Kermany, D.S.; Goldbaum, M.; Cai, W.; Valentim, C.; Liang, H.; Baxter, S.L.; McKeown, A.; Yang, G.; Wu, X.; Yan, F.; et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* **2018**, *172*, 1122–1131.e9. [CrossRef] [PubMed]