Published in Image Processing On Line on 2023–02–12.
Submitted on 2022–10–12, accepted on 2022–10–13.
ISSN 2105–1232 © 2023 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2023.440

Incidence of the Sample Size Distribution on One-Shot Federated Learning

Marie Garin, Gonzalo Iñaki Quintana

Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
{marie.garin, gonzalo.quintana}@ens-paris-saclay.fr

Communicated by Jean-Michel Morel *Demo edited by* Marie Garin and Gonzalo Quintana

Abstract

Federated Learning (FL) is a learning paradigm where multiple nodes collaboratively train a model by only exchanging updates or parameters. This enables to keep data locally, therefore enhancing privacy – statement requiring nuance, e.g. memorization of training data in language models. Depending on the application, the number of samples that each node contains can be very different, which can impact the training and the final performance. This work studies the impact of the per-node sample size distribution on the mean squared error (MSE) of the one-shot federated estimator. We focus on one-shot aggregation of statistical estimations made across disjoint, independent and identically distributed (i.i.d.) data sources, in the context of empirical risk minimization. In distributed learning, it is well-known that for a total number of m nodes, each node should contain at least m samples to equal the performance of centralized training. In a federated scenario, this result remains true, but now applies to the mean of the per-node sample size distribution. The demo enables to visualize this effect as well as to compare the behavior of the FESC (Federated Estimation with Statistical Correction) algorithm – a weighting scheme which depends on the local sample size – with respect to the classical federated estimator and the centralized one, for a large collection of distributions, number of nodes, and features space dimension.

Source Code

The source code and documentation for the algorithm described in this article are available from the web page[1]. Usage instructions are included in the `README` file of the archive. The original implementation of the algorithm is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: federated learning; distributed learning; supervised learning; one-shot learning; empirical risk minimization; mean squared error

[1] https://doi.org/10.5201/ipol.2023.440
[2] https://github.com/gonzaq94/one-shot-fed-learning

1 Introduction

Machine learning algorithms are typically trained in a centralized way, where data is collected from possibly multiple sources and stored at a central server or location, for analysis and training. However, the data collection process can sometimes be slow, costly, and raise privacy issues. To handle the rapidly increasing amount of data produced from a large, diverse, and heterogeneous number of sources, researchers study more distributed architectures. Federated learning (FL) [4, 5] has stood out as a promising field of growing emphasis, advocating for an alternative to centralized learning.

FL consists of a set of m nodes that contain local datasets and computing capabilities, which enable them to collectively train a machine learning model. The nodes exchange only model updates or gradients, and the data never leaves its sources (which partially alleviates the data privacy issues). We focus on the server orchestrated setting [3], in which a central server collects and aggregates the updates produced locally by the nodes to provide a global outcome. Each of the m nodes contains a variable number of observations n_i, that make up the local dataset. When performing a distributed training, the distribution of the sample size across the nodes (i.e., n_i) raises new statistical challenges.

In this demo, we focus on one-shot federated learning [2, 8, 6], where the aggregation of the computations of all the nodes is made on one communication round, in contrast to the *multi-round setting*. We provide insights on the effect of the sample size distribution on the mean squared error (MSE), and the leverage effect obtained with the FESC algorithm [1], an aggregation procedure that relies on the optimization of an upper bound of the MSE. We compare the behavior of two one-shot federated learning estimators (the plain federated estimator and FESC) with the centralized one, in the context of empirical risk minimization. The MSE of the two weighting schemes and of the centralized estimator is plotted according to the mean of the sample size distribution: the relationship of this quantity to the total number of nodes m plays a crucial role in the final performance. Indeed, the regular federated estimator reaches the performance of the centralized estimator when the sample sizes average is of the same order as m, the number of nodes [1]. The demo displays the MSE of the three estimators with respect to the mean of the sample size distribution, for a broad collection of distributions (e.g., log-normal, Gaussian, Pareto, etc.). It shows that the FESC weighting scheme converges faster to the performance of the centralized estimator than the plain federated estimator. This effect is more visible when the sample size distribution has higher variance. To perform the simulation, we synthesize data from a linear model and estimate the sought parameter with ridge regression. The demo enables to choose the sample size distribution, the features dimension, and the number of nodes that participate in the training.

2 Federated Learning with Statistical Correction

In this work, the model class $\mathcal{F}$ is the set of parametrized linear functions given by

$$\mathcal{F}_\Theta = \{f_\theta : \forall x \in \mathbb{R}^d,\ f_\theta(x) = x^T\theta,\ \theta\}, \tag{1}$$

with $\Theta \subset \mathbb{R}^d$ the parameter space and d the features dimension. We consider a low-dimensional regime and an i.i.d. (independent and identically distributed) sampling. As we focus on the federated setting, the data is distributed across m nodes. Although one of the major challenges of federated learning is to deal with non-identically distributed sampling, the framework is restricted to the i.i.d. hypothesis. Indeed, as this demo aims at providing some insights on the effect of the distribution of sample sizes, we focus on this specific issue. Each node i contains a subset of n_i samples of two random variables X, with values in $\mathbb{R}^d$, and Y, with values in $\mathbb{R}$: $\{(x_{ij}, y_{ij}), 1 \leq j \leq n_i\}$. Given some loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$, the global risk is defined by

$$R(\theta) = \mathbb{E}[\ell(f_\theta(X), Y)], \tag{2}$$

and we set $\theta^* \in \arg\min_{\theta\in\Theta} R(\theta)$. The goal is to estimate the hidden parameter θ^*. Since the random variable distributions are not accessible, we focus on the empirical risk, defined as

$$\hat{R}(\theta) = \frac{1}{N}\sum_{i}^{m}\sum_{j}^{n_i} \ell(f_\theta(x_{ij}), y_{ij}), \tag{3}$$

with $N = \sum_i n_i$ the total number of samples. In the centralized setting, a widespread approach is to estimate θ by *empirical risk minimization* (ERM), i.e.

$$\hat{\theta}_{\mathrm{c}} \in \arg\min_{\theta\in\Theta} \hat{R}(\theta), \tag{4}$$

with $\hat{\theta}_{\mathrm{c}}$ the centralized estimator. In one-shot federated learning, each node i estimates its local parameter $\hat{\theta}_i$ through ERM

$$\hat{\theta}_i \in \arg\min_{\theta\in\Theta} \hat{R}_i(\theta), \tag{5}$$

with $\hat{R}_i$ the local empirical risk of node i defined as

$$\hat{R}_i(\theta) = \frac{1}{n_i}\sum_{j}^{n_i} \ell(f_\theta(x_{ij}), y_{ij}). \tag{6}$$

The final federated estimator is usually a convex combination of the locally estimated parameters

$$\hat{\theta}_{\mathrm{w}} = \sum_{i}^{m} w_i\hat{\theta}_i, \tag{7}$$

with $w_i \geq 0$ and $\sum_i w_i = 1$. By setting $w_i = \frac{n_i}{N}$, we obtain the plain federated estimator

$$\hat{\theta}_{\mathrm{s}} \;=\; \sum_{i}^{m} \frac{n_i}{N}\hat{\theta}_i. \tag{8}$$

We will use this estimator as a baseline to compare with FESC, an improved weighting scheme detailed below.

The FESC aggregation algorithm selects a subset of K nodes – the nodes with the largest sample sizes – to participate in the final aggregated parameter. The proposed weighting scheme is given by

$$\hat{w}_{(i)} = \begin{cases} \frac{n_{(i)}}{2}\frac{2+\sum_j^K \frac{1}{n_{(j)}}}{\sum_j^K n_{(j)}} - \frac{1}{2n_{(i)}}, \ \forall i \leq K \\ 0, \ \forall i > K, \end{cases} \tag{9}$$

where $n_{(i)}$ are the reordered n_i in descending order, i.e., $n_{(1)} \geq \cdots \geq n_{(m)}$, and $\hat{w}_{(i)}$ the corresponding value for $n_{(i)}$, i.e., if $n_{(i)} = n_p$, then $\hat{w}_{(i)} = \hat{w}_p$. The number of nodes, K, participating in the learning is defined as

$$K = \arg\max_{k\in[m]} \left\{ \frac{1}{n_{(k)}^2} \leq \frac{2+\sum_j^k \frac{1}{n_{(j)}}}{\sum_j^k n_{(j)}} \right\}. \tag{10}$$

The federated estimator with statistical correction is then obtained by

$$\hat{\theta}_{\hat{\mathrm{w}}} = \sum_{i=1}^{m} \hat{w}_{(i)} \hat{\theta}_{(i)}, \tag{11}$$

where the $\hat{\theta}_{(i)}$ are the corresponding values for $n_{(i)}$. The FESC algorithm is summarized in Algorithm 1.

Algorithm 1 FESC pseudo-code

Inputs: number of nodes m, features dimension d.
for each node $i = 1, \ldots, m$ **do**
    The node sends n_i and $\hat{\theta}_i$ to the server
end for
Order all the n_i in decreasing order.
$K, \hat{w}_{(i)} \leftarrow$ from Equations (9) and (10)
$\hat{\theta}_{\hat{\mathrm{w}}} \leftarrow \sum_i \hat{w}_{(i)} \hat{\theta}_{(i)}$
return $\hat{\theta}_{\hat{\mathrm{w}}}$

3 Experiments

The experiments were performed on synthetic data generated as follows. X is a random variable with values in $\mathbb{R}^d$ and Y with values in $\mathbb{R}$. We assume a linear model, i.e. $Y = X^T\theta^* + \epsilon$, with ϵ sampled according to a standard normal distribution. The sought parameter θ^* is sampled according to a multivariate uniform distribution with support $[0, 1]^d$ and X according to a multivariate standard normal distribution. We compare the MSE of 3 estimations of θ: the centralized $\hat{\theta}_{\mathrm{c}}$, the federated $\hat{\theta}_{\mathrm{s}}$ and the federated with statistical correction $\hat{\theta}_{\hat{\mathrm{w}}}$. The number of nodes m, the features dimension d and the distribution of the sample size are parameters of the demo chosen by the user. We denote by η the number of samples per node, which is a random variable following a sample size distribution chosen by the user. The mean of η is expressed as a function of m

$$\mathbb{E}[\eta] = m^{\gamma}. \tag{12}$$

We vary γ (and hence the mean sample size per node, as the number of nodes is fixed) from 0.2 to 1.2, in steps of 0.1. For each value of γ, we first generate the m samples sizes (Equation (12)) with the associated samples, and estimate the hidden parameter through ridge regression in a central way and locally at each node. We then derive the weights of the standard federated estimator and those of the FESC algorithm using Equations (8) and (11). Finally, all the local estimations are combined with Equation (7) and, using the ground-truth value of θ^*, the MSE is obtained. We fix the regularization parameter of the ridge regression to the inverse of the squared root of the global sample size ($\lambda_i = \frac{1}{N}$), see [7] for further explanations. For the sake of computation time, the experiments are realized in only one run. The pseudo-code of the entire demo is shown in Algorithm 2 (see Appendix).

4 Demo

In this section, we discuss the basic aspects of the demonstration, and give insights on the required input parameters, and the outputs it generates.

4.1 Input Parameters

The demo takes three input parameters:

Number of nodes: the number of nodes that are involved in training. Ranges from 2 to 100, defaults to 100.

Features dimension: the dimension of the hidden parameter vector θ. Ranges from 1 to 100, defaults to 10.

Sample size probability distribution: the probability distribution of the sample sizes in each node, the associated random variable being denoted by η. Several distributions are available, all of mean $\mathbb{E}[\eta] = m^\gamma$. The distributions are sampled using `scipy.stats` with the method `rvs`. Defaults to log-normal of shape parameter 1, and scale parameter set so that the mean is equal to $\mathbb{E}[\eta] = m^\gamma$. The available distributions are the following:

- Log-normal of shape parameter $\mathtt{s} = 1$ and scale parameter $\mathtt{scale} = m^\gamma e^{\frac{-1}{2}}$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = (e^{\mathtt{s}^2} - 1)e^{2m^\gamma} = (e-1)e^{2m^\gamma}$ (denoted as **log-normal (low variance)**);
- Log-normal of shape parameter $\mathtt{s} = 2$ and scale parameter $\mathtt{scale} = m^\gamma e^2$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = (e^{\mathtt{s}^2} - 1)e^{2m^\gamma} = (e^4-1)e^{2m^\gamma}$ (denoted as **log-normal (high variance)**);
- Poisson of shape parameter $\mathtt{mu} = m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \mathtt{mu} = m^\gamma$ (denoted as **poisson**);
- Gaussian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = 0.1m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \mathtt{scale}^2 = 0.01m^{2\gamma}$ (denoted as **gaussian (low variance)**);
- Gaussian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = 0.5m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \mathtt{scale}^2 = 0.25m^{2\gamma}$ (denoted as **gaussian (medium variance)**);
- Gaussian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \mathtt{scale}^2 = m^{2\gamma}$ (denoted as **gaussian (high variance)**);
- Uniform of location parameter $\mathtt{loc} = 0.5m^\gamma$, scale parameter $\mathtt{scale} = m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \frac{\mathtt{scale}^2}{12} = \frac{m^{2\gamma}}{12}$ (denoted as **uniform (low variance)**);
- Uniform of location parameter $\mathtt{loc} = 0$, scale parameter $\mathtt{scale} = 2m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \frac{\mathtt{scale}^2}{12} = \frac{m^{2\gamma}}{3}$ (denoted as **uniform (high variance)**);
- Laplacian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = 0.1m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = 2\mathtt{scale}^2 = 0.02m^{2\gamma}$ (denoted as **laplacian (low variance)**);
- Laplacian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = 0.5m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = 2\mathtt{scale}^2 = 0.5m^{2\gamma}$ (denoted as **laplacian (medium variance)**);
- Laplacian of location parameter $\mathtt{loc} = m^\gamma$, scale parameter $\mathtt{scale} = m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = 2\mathtt{scale}^2 = 2m^{2\gamma}$ (denoted as **laplacian (high variance)**);
- Pareto of of shape parameter $\mathtt{b} = 2.5$ and scale parameter $\mathtt{scale} = 0.6m^\gamma$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \frac{\mathtt{scale}^2\mathtt{b}}{(\mathtt{b}-1)^2(\mathtt{b}-2)} = 0.8m^{2\gamma}$ (denoted as **pareto**);
- Weibull of shape parameter $\mathtt{c} = 1.5$ and scale parameter $\mathtt{scale} = \frac{m^\gamma}{\Gamma(1+\frac{2}{3})}$, thus $\mathbb{E}[\eta] = m^\gamma$ and $\mathbb{V}(\eta) = \mathtt{scale}^2\big(\Gamma(1+\frac{2}{\mathtt{c}}) - \Gamma^2(1+\frac{1}{\mathtt{c}})\big) = m^{2\gamma}\big(\frac{\Gamma(1+\frac{4}{3})}{\Gamma(1+\frac{2}{3})} - 1\big)$ (denoted as **weibull**).

Increasing the parameters "number of nodes" and "features dimension" increases the computation time. If the simulation is running too slow, you are invited to decrease the value of those parameters.

4.2 Outputs

The demo generates two plots as outputs:

Sample size histogram: the histogram of the sample size distribution across the nodes (η variable) for $\gamma = 1.2$ (largest γ in the simulation). This is used to illustrate the chosen probability distribution. The choice of using the largest γ (and therefore, the largest mean) is arbitrary.

MSE vs. γ curve: this shows the variation of the MSE for the three studied algorithms (centralized, federated, and FESC), as a function of γ. The objective of this plot is to show the minimal value of γ (and, therefore, of $\mathbb{E}[\eta]$) that is needed for the federated algorithms to equal the performance of the centralized learning.

Figure 1 shows the two outputs of the demo, when the input parameters are set to their default values (100 nodes, features of dimension 10, log-normal distribution). In the MSE vs. γ plot, we can see that for $\gamma < 1$ ($\mathbb{E}[\eta] = m$), the federated algorithm cannot attain the performance of centralized learning. The FESC algorithm manages to reduce the smallest γ necessary for equaling the performance of centralized learning. We remark that in order to keep the execution time of the demo under 30 seconds, we were constrained to use only one Montecarlo iteration for the simulation. Increasing the number of Montecarlo iterations would smooth the MSE vs. γ plot, and make the plots better coincide with the theoretical results.

Figure 1: Output plots generated by the demo for the default parameters: the sample size histogram (left), and the MSE vs. γ curve (right).

5 Conclusions

A demo that allows to visualize the impact of the sample size distribution on the one-shot federated estimator was constructed. The demo compares the MSE of the plain federated estimator and FESC to that of a centralized training, varying the mean of the sample size distribution. It shows that, in order for the federated algorithms to attain the performance of the centralized training, the mean number of samples per node has to be at least equal to the number of nodes in the training. In order to comply with the max execution time of the IPOL journal, the simulation uses only one Montecarlo iteration. To obtain results that better agree with the theory, more Montecarlo iterations should be used.

Acknowledgments

The authors would like to thank all the organizers and mentors of the ML Briefs workshop.

Appendix: Demo Pseudo-Code

Algorithm 2 Demo pseudo-code

Inputs: number of nodes m, features' dimension d, sample-size distribution of the random variable η.
for each $\gamma = 0.2, 0.3, \ldots, 1.2$ **do**
  $n = (n_1, \ldots, n_m) \leftarrow m$ realizations of η (with $\mathbb{E}[\eta] = m^\gamma$)
  $x, y, \theta^* \leftarrow$ **GenerateData**(d, n)
  $\lambda \leftarrow \frac{1}{\sqrt{N}}$ ▷ ridge regularization parameter
  $\hat{\theta}_{\mathrm{c}} \leftarrow$ **RidgeRegression**(λ, x, y) ▷ centralized estimator
  for each node $i = 1, \ldots, m$ **do**
    $\hat{\theta}_i \leftarrow$ **RidgeRegression**(λ, x_i, y_i)
  end for
  $\theta \leftarrow [\theta_1, \ldots, \theta_m]$
  $\hat{\theta}_{\mathrm{s}} \leftarrow \sum_i \frac{n_i}{N} \hat{\theta}_i$ ▷ plain federated estimator
  $\hat{\theta}_{\hat{\mathrm{w}}} \leftarrow$ **FESC**(n, θ) ▷ FESC estimator
  $\mathrm{MSE}(\hat{\theta}_{\mathrm{c}}) \leftarrow \|\hat{\theta}_{\mathrm{c}} - \theta^*\|^2$, $\mathrm{MSE}(\hat{\theta}_{\mathrm{s}}) \leftarrow \|\hat{\theta}_{\mathrm{s}} - \theta^*\|^2$, $\mathrm{MSE}(\hat{\theta}_{\hat{\mathrm{w}}}) \leftarrow \|\hat{\theta}_{\hat{\mathrm{w}}} - \theta^*\|^2$
end for
plot$(\mathrm{MSE}(\hat{\theta}_{\mathrm{c}}), \mathrm{MSE}(\hat{\theta}_{\mathrm{s}}), \mathrm{MSE}(\hat{\theta}_{\hat{\mathrm{w}}}), \gamma)$

GenerateData(n, d): ▷ $n = [n_1, \ldots, n_m]$ sample sizes array, d features dimensions
$\theta^* \leftarrow$ from $\mathcal{U}([0,1]^d)$
for each node $i = 1, \ldots, m$ **do**
  $x_i \leftarrow$ from $\mathcal{N}(0,1)^{\otimes(n_i, d)}$
  $\epsilon_i \leftarrow$ from $\mathcal{N}(0,1)^{\otimes n_i}$
  $y_i \leftarrow x_i^T \theta^* + \epsilon_i$
end for
$x \leftarrow [x_1, \ldots, x_m], y \leftarrow [y_1, \ldots, y_m]$
return x, y, θ^*

RidgeRegression(λ, x, y): ▷ x: input features, y: targets, λ regularization weight
$w^* \leftarrow \arg\min_w (\|y - Xw\|_2^2 + \lambda * \|w\|_2^2)$
return w^*

FESC(n, θ): ▷ $n = [n_1, \ldots, n_m]$ sample sizes array, θ parameters array
$n \leftarrow$ sort n in decreasing order.
$K \leftarrow$ from Equation (10) and n
$\hat{w}_{(i)} \leftarrow$ from Equation (9) and K
$\hat{\theta} \leftarrow \sum_i \hat{w}_{(i)} \hat{\theta}_{(i)}$
return $\hat{\theta}$

References

[1] M. Garin, T. Evgeniou, and N. Vayatis, *Weighting schemes for one-shot federated learning*, SSRN, (2022). https://doi.org/10.2139/ssrn.4035732.

[2] N. Guha, A. Talwalkar, and V. Smith, *One-shot federated learning*, CoRR, abs/1902.11175 (2019). https://doi.org/10.48550/arXiv.1902.11175.

[3] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., *Advances and open problems in federated learning*, Now Foundations and Trends, (2021). http://dx.doi.org/10.1561/2200000083.

[4] J. Konečný, H.B. McMahan, and D. Ramage, *Federated optimization: Distributed optimization beyond the datacenter*, CoRR, abs/1511.03575 (2015). https://doi.org/10.48550/arXiv.1511.03575.

[5] J. Konečný, H.B. McMahan, D. Ramage, and P. Richtárik, *Federated optimization: Distributed machine learning for on-device intelligence*, CoRR, abs/1610.02527 (2016). https://doi.org/10.48550/arXiv.1610.02527.

[6] S. Salehkaleybar, A. Sharifnassab, and S.J. Golestani, *One-shot federated learning: theoretical limits and algorithms to achieve them*, Journal of Machine Learning Research, 22 (2021), pp. 1–47. https://dl.acm.org/doi/abs/10.5555/3546258.3546447.

[7] Y. Zhang, J.C. Duchi, and M.J. Wainwright, *Communication-efficient algorithms for statistical optimization*, Journal of Machine Learning Research, 14 (2013), pp. 3321–3363. https://dl.acm.org/doi/10.5555/2567709.2567769.

[8] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, *Distilled one-shot federated learning*, 2020. https://doi.org/10.48550/arXiv.2009.07999.