COMPRESSION FOR FEDERATED RANDOM RESHUFFLING

A PREPRINT

Tikhon Antyshev
MIPT

Grigory Malinovsky
KAUST

ABSTRACT

Federated Random Reshuffling (FedRR) is a recently developed method for federated training of supervised machine learning models via empirical risk minimization. It utilizes Random Reshuffling (RR), a variant of Stochastic Gradient Descent (SGD)along with Local Training carried out by the clients. We propose integration of compression techniques in FedRR, reducing the number of communicated bits in order to overcome communication bottleneck, furthermore we integrate server-side optimization (Server Stepsizes) to get improvement in theory and practice. To the best of our knowledge, this is the first time FedRR will be combined with Server Stepsizes and Compressed Iterates at the same time.

Keywords Machine Learning · Federated Learning · Random Reshuffling

1 Introduction

Modern machine learning models heavily rely on Empirical Risk Minimization for supervised training training. The success of this approach itself can be attributed to a plentiful amount of data available and exponential increase in computing power. The latter remains a problem in machine learning tasks, as some models require ridiculous resources for training. One of the proposed solutions to this is parallel computing - distribution of the tasks between clients. It is known as Federated Learning.

1.1 Federated Learning

Federated Learning (FL) is a method of training models, which avoids centralized data-storing and instead relies on clients for greater energy efficiency and privacy. The clients do not share their private data between each other or the server, however each of them partakes in model training by local optimization over personal data.

1.2 Problem Statement

We consider the standard finite-sum optimization formulation of federated learning problem:

$$x_* = \arg\min_{x \in \mathbb{R}^d} \left[f(x) \stackrel{\text{def}}{=} \frac{1}{M} \sum_{m=1}^{M} f_m(x) \right] \tag{1}$$

$$f_m(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_m^i(x)$$

Where M is number of clients in current task, $f_m : \mathbb{R}^d \to \mathbb{R}$ is the loss of the model x on the training data subset owned by client or device m, which has the finite-sum structure, comprised of $f_m^i : \mathbb{R}^d \to \mathbb{R}$ - loss of model x on training point $i \in \overline{1, n}$. We shall also assume that $\forall\ m,\ i:\ f_m^i$ is differentiable and consider strongly convex, convex and non-convex modes.

1.3 Improving FL

Compressed Communication. As the model size increases, so does the time required to transfer iteration of parameters. The end result is performance worsening drastically. In order to overcome this communication bottleneck, we will be using gradient compression scheme and compressed iterates.

Random Reshuffling. It has been experimentally proven that using a random permutation of data set provides better results, which is distinct from standard SGD, which picks data points in an unbiased way.

Local training. This can be seen as minibatch SGD performed on the latest version of model, provided by the server.

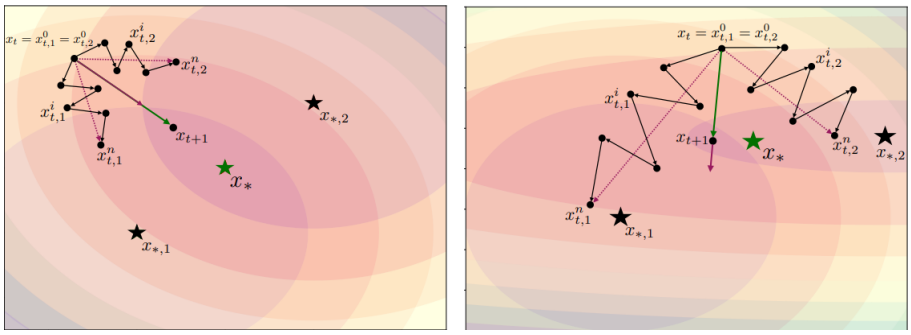
Partial Participation. FL process consists of communication rounds, in each server chooses a subset of clients to participate in optimization. This is done in order to take limited server capability or limited client availability.

Server Stepsizes. After clients finish local optimization they send their results to the server. The most simple action that can be performed is taking average as new global model state. However, empirical evidence suggests that it is more beneficial to treat received model states as gradient information and performing one step of GD over them.

2 Contributions

3 Preliminaries

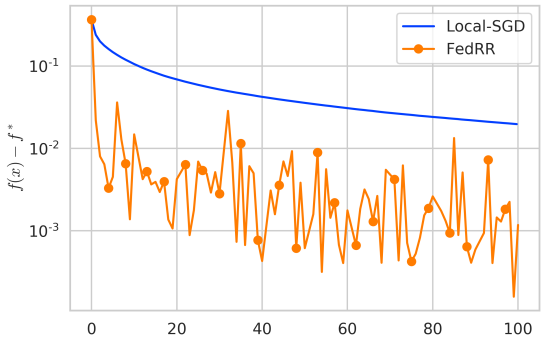
4 Theory



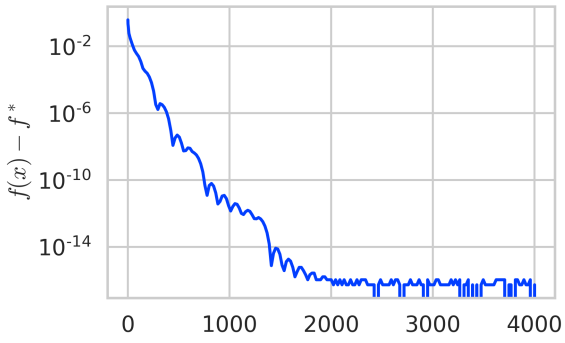
Server Step-Sizes

5 Experiments

The goal of the experiment is to demonstrate improvements of implementing Compressed Iterates with Server Step-Sizes method. We run our experiment on $l2$-regularized logistic regression with the 'a1a' dataset from LibSVM. As a baseline solution we will be demonstrating the results of Proximal and Federated RR. Afterwards, we shall compare performance for proposed algorithm and versions that use either only Server Step-sizes or only Compressed Iterates.



Local SGD and Federated RR



Performance of Nesterov method

6 Examples of citations, figures, tables, references

6.1 Citations

6.2 Figures

6.3 Tables

6.4 Lists

References

Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Proximal and federated random reshuffling. 2021. doi:10.48550/ARXIV.2102.06704. URL https://arxiv.org/abs/2102.06704.

Grigory Malinovsky, Alibek Sailanbayev, and Peter Richtárik. Random reshuffling with variance reduction: New analysis and better rates. 2021. doi:10.48550/ARXIV.2104.09342. URL https://arxiv.org/abs/2104.09342.

Grigory Malinovsky, Konstantin Mishchenko, and Peter Richtárik. Server-side stepsizes and sampling without replacement provably help in federated optimization. 2022. doi:10.48550/ARXIV.2201.11066. URL https://arxiv.org/abs/2201.11066.

Ahmed Khaled and Peter Richtárik. Gradient descent with compressed iterates. 2019. doi:10.48550/ARXIV.1909.04716. URL https://arxiv.org/abs/1909.04716.

Sélim Chraibi, Ahmed Khaled, Dmitry Kovalev, Peter Richtárik, Adil Salim, and Martin Takáč. Distributed fixed point methods with compressed iterates. 2019. doi:10.48550/ARXIV.1912.09925. URL https://arxiv.org/abs/1912.09925.

Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. 2020. doi:10.48550/ARXIV.2006.05988. URL https://arxiv.org/abs/2006.05988.