

On 5th Generation of Local Training Methods in Federated Learning

Peter Richtárik

*King Abdullah University of Science and Technology
Kingdom of Saudi Arabia*



Talk @ Apple
May 23, 2023



Konstantin Mishchenko



Arto Maranjyan



Dmitry Kovalev



Michal Grudzien



Laurent Condat



Sebastian Stich



Ivan Agarský



Mher Safaryan



Grigory Malinovsky



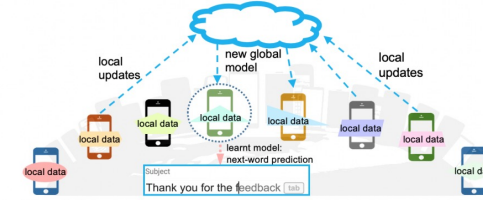
Kai Yi



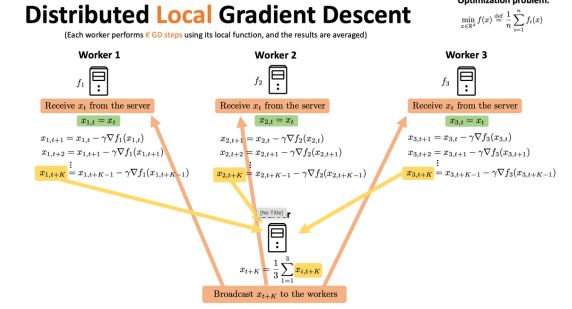
Abdurakhmon Sadiev

Coauthors

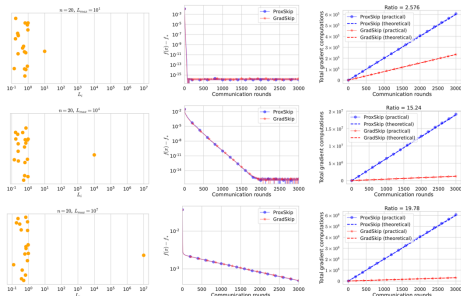
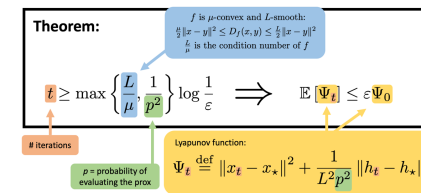
Outline of the Talk



1. Local Training
2. Brief History of Local Training
3. 5th Generation of Local Training Methods
4. ProxSkip
5. RandProx
6. GradSkip



ProxSkip: Bounding the # of Iterations

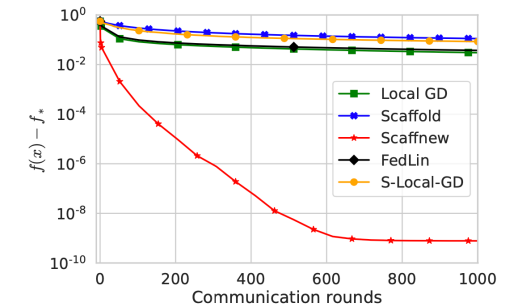


Algorithm 1 ProxSkip

```

1: stepsize  $\gamma > 0$ , probability  $p > 0$ , initial iterate  $x_0 \in \mathbb{R}^d$ , initial control variate  $h_0 \in \mathbb{R}^d$ , number of iterations  $T \geq 1$ 
2: for  $t = 0, 1, \dots, T - 1$  do
3:    $\hat{x}_{t+1} = x_t - \gamma(\nabla f(x_t) - h_t)$  ◊ Take a gradient-type step adjusted via the control variate  $h_t$ 
4:   Flip a coin  $\theta_t \in \{0, 1\}$  where  $\text{Prob}(\theta_t = 1) = p$  ◊ Flip a coin that decides whether to skip the prox or not
5:   if  $\theta_t = 1$  then
6:      $x_{t+1} = \text{prox}_{\frac{\gamma}{p}\psi}(\hat{x}_{t+1} - \frac{\gamma}{p}h_t)$  ◊ Apply prox, but only very rarely! (with small probability  $p$ )
7:   else
8:      $x_{t+1} = \hat{x}_{t+1}$  ◊ Skip the prox!
9:   end if
10:   $h_{t+1} = h_t + \frac{p}{\gamma}(x_{t+1} - \hat{x}_{t+1})$  ◊ Update the control variate  $h_t$ 
11: end for

```



(c) theoretical hyper-parameters



Part 1

Local Training

Optimization Formulation of Federated Learning

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

model parameters / features

$$\frac{1}{n} \sum_{i=1}^n$$

$$f_i(x)$$

devices /
machines

Loss on local data \mathcal{D}_i stored on device i

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

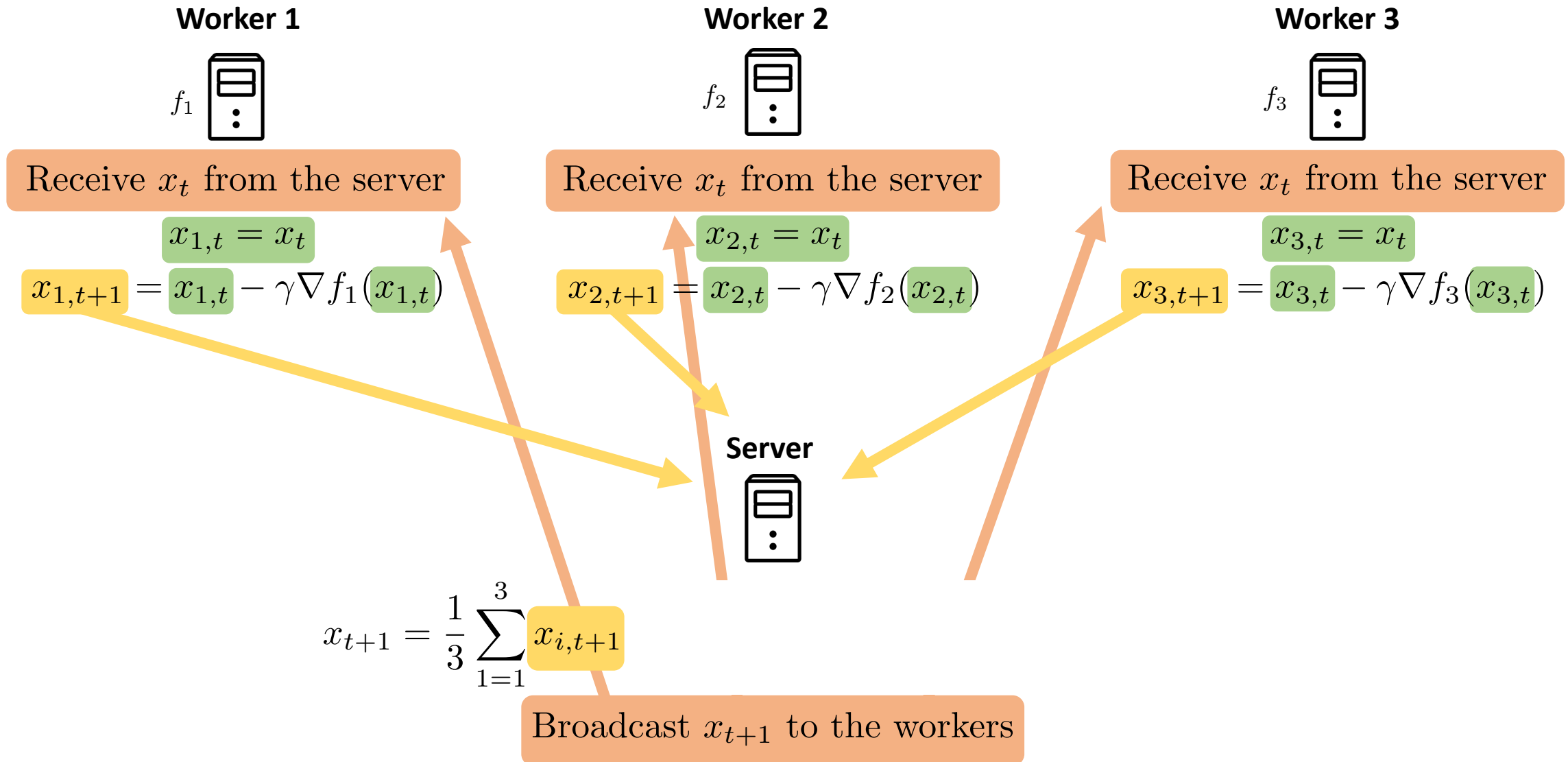
The datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be arbitrarily heterogeneous

Distributed Gradient Descent

(Each worker performs 1 GD step using its local function, and the results are averaged)

Optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

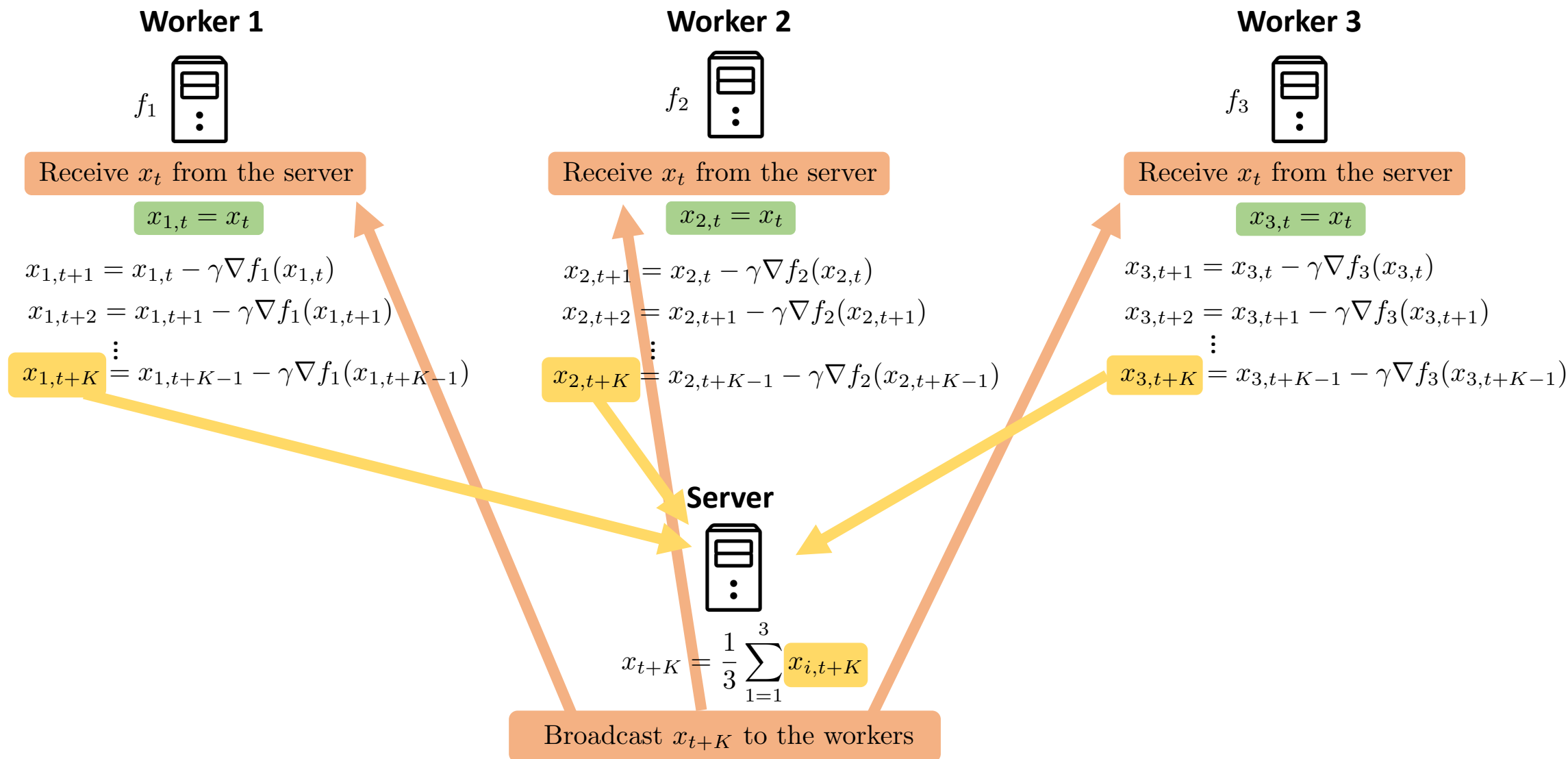


Distributed Local Gradient Descent

(Each worker performs K GD steps using its local function, and the results are averaged)

Optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$



Part 2

Brief History of Local Training



Grigory Malinovsky, Kai Yi and P.R.

Variance reduced ProxSkip: algorithm, theory and application to federated learning

NeurIPS 2022

Brief History of Local Training Methods

Table 1: Five generations of local training (LT) methods summarizing the progress made by the ML/FL community over the span of 7+ years in the understanding of the *communication acceleration properties of LT*.

Generation ^(a)	Theory	Assumptions	Comm. Complexity ^(b)	Selected Key References
1. Heuristic	✗	—	empirical results only	LocalSGD [Povey et al., 2015]
	✗	—	empirical results only	SparkNet [Moritz et al., 2016]
	✗	—	empirical results only	FedAvg [McMahan et al., 2017]
2. Homogeneous	✓	bounded gradients	sublinear	FedAvg [Li et al., 2020b]
	✓	bounded grad. diversity ^(c)	linear but worse than GD	LFGD [Haddadpour and Mahdavi, 2019]
3. Sublinear	✓	standard ^(d)	sublinear	LGD [Khaled et al., 2019]
	✓	standard	sublinear	LSGD [Khaled et al., 2020]
4. Linear	✓	standard	linear but worse than GD	Scaffold [Karimireddy et al., 2020]
	✓	standard	linear but worse than GD	S-Local-GD [Gorbunov et al., 2020a]
	✓	standard	linear but worse than GD	FedLin [Mitra et al., 2021]
5. Accelerated	✓	standard	linear & better than GD	ProxSkip/Scaffnew [Mishchenko et al., 2022]
	✓	standard	linear & better than GD	ProxSkip-VR [THIS WORK]

^(a) Since client sampling (CS) and data sampling (DS) can only *worsen* theoretical communication complexity, our historical breakdown of the literature into 5 generations of LT methods focuses on the full client participation (i.e., no CS) and exact local gradient (i.e., no DS) setting. While some of the referenced methods incorporate CS and DS techniques, these are irrelevant for our purposes. Indeed, from the viewpoint of communication complexity, all these algorithms enjoy best theoretical performance in the no-CS and no-DS regime.

^(b) For the purposes of this table, we consider problem (1) in the *smooth* and *strongly convex* regime only. This is because the literature on LT methods struggles to understand even in this simplest (from the point of view of optimization) regime.

^(c) *Bounded gradient diversity* is a uniform bound on a specific notion of gradient variance depending on client sampling probabilities. However, this assumption (as all homogeneity assumptions) is very restrictive. For example, it is not satisfied the standard class of smooth and strongly convex functions.

^(d) The notorious FL challenge of handling non-i.i.d. data by LT methods was solved by Khaled et al. [2019] (from the viewpoint of *optimization*). From generation 3 onwards, there was no need to invoke any data/gradient homogeneity assumptions. Handling non-i.i.d. data remains a challenge from the point of view of *generalization*, typically by considering *personalized* FL models.



Grigory Malinovsky, Kai Yi and P.R.

Variance Reduced ProxSkip: Algorithm, Theory and Application to Federated Learning

NeurIPS 2022

Brief History of Local Training Methods

Generation 1: Heuristic

“No theory”

10/2014



Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur
Parallel Training of DNNs with Natural Gradient and Parameter Averaging
ICLR Workshops 2015

11/2015



Philipp Moritz, Robert Nishihara, Ion Stoica, Michael I. Jordan
SparkNet: Training Deep Networks in Spark
ICLR 2015

02/2016



H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas
Communication-Efficient Learning of Deep Networks from Decentralized Data
AISTATS 2017

Accepted as a workshop contribution at ICLR 2015

PARALLEL TRAINING OF DNNs WITH NATURAL GRADIENT AND PARAMETER AVERAGING

Daniel Povey, Xiaohui Zhang & Sanjeev Khudanpur
Center for Language and Speech Processing, 40 Brown Language Technology Center of Excellence,
The Johns Hopkins University, Baltimore, MD 21218, USA
{dpovey@jhu.edu, xzhang@jhu.edu, sanjeev@jhu.edu}

ABSTRACT

We describe the neural-network training framework used in the Kaldi speech recognition toolkit, which is geared towards training DNNs with large amounts of training data using multiple GPUs, supported by multi-core machines, leading to a 10x reduction in training time. We describe the framework in detail, showing how the natural gradient is approximated by the natural gradient, and how the natural gradient is used to train the network. The framework is designed to be efficient and scalable, and we show that it can be used to train DNNs with large amounts of training data.

1 INTRODUCTION

Parallel training of neural networks generally involves use of some combination of model partitioning and data partitioning (Dean et al., 2012), and the natural approach to data partitioning involves communication of model parameters for each sub-model. Here we describe our natural gradient training framework which uses a different version of data partitioning: we have multiple GPUs processing on separate machines, and each independently computes updates to the model parameters and communicates them to the individual machines. This is very efficient for us as the large-scale training of recurrent neural networks (RNNs) is a very memory-intensive task, and we cannot fit the entire model on a single machine. In our case, it is only worth using a natural gradient with an efficient implementation of natural gradient methods (gradient descent (NG-GD)) that we have developed. We don't attempt in this paper to develop a framework that explains why parameter averaging should work well despite non-convexity of DNNs, or why NG-GD is so helpful. The point of this paper is to describe our methods and to establish that empirically they work well. The significance of this work is that we show that it is possible to get a faster speedup when increasing the number of GPUs, without requiring frequent data transfer (dramatic data only being up to around 1 MB/GPU).

In Section 2 we describe our problem setting, which is Deep Neural Networks (DNNs) applied to speech recognition, although our ideas are more general than this. In Section 3 we introduce the parallel training method. In Section 4 we describe the general idea behind our natural gradient method, although most of the technical details have been relegated to the appendix. In this paper we also give the proofs, but we do not discuss in Section 5 what we think we can use our code for. We present our conclusions in Section 6.

There are two versions of our NG-GD method: a “simple” version and an “advanced” one. Technical details for these are in Appendixes A and B respectively. Appendix C has background information on our DNN implementation.

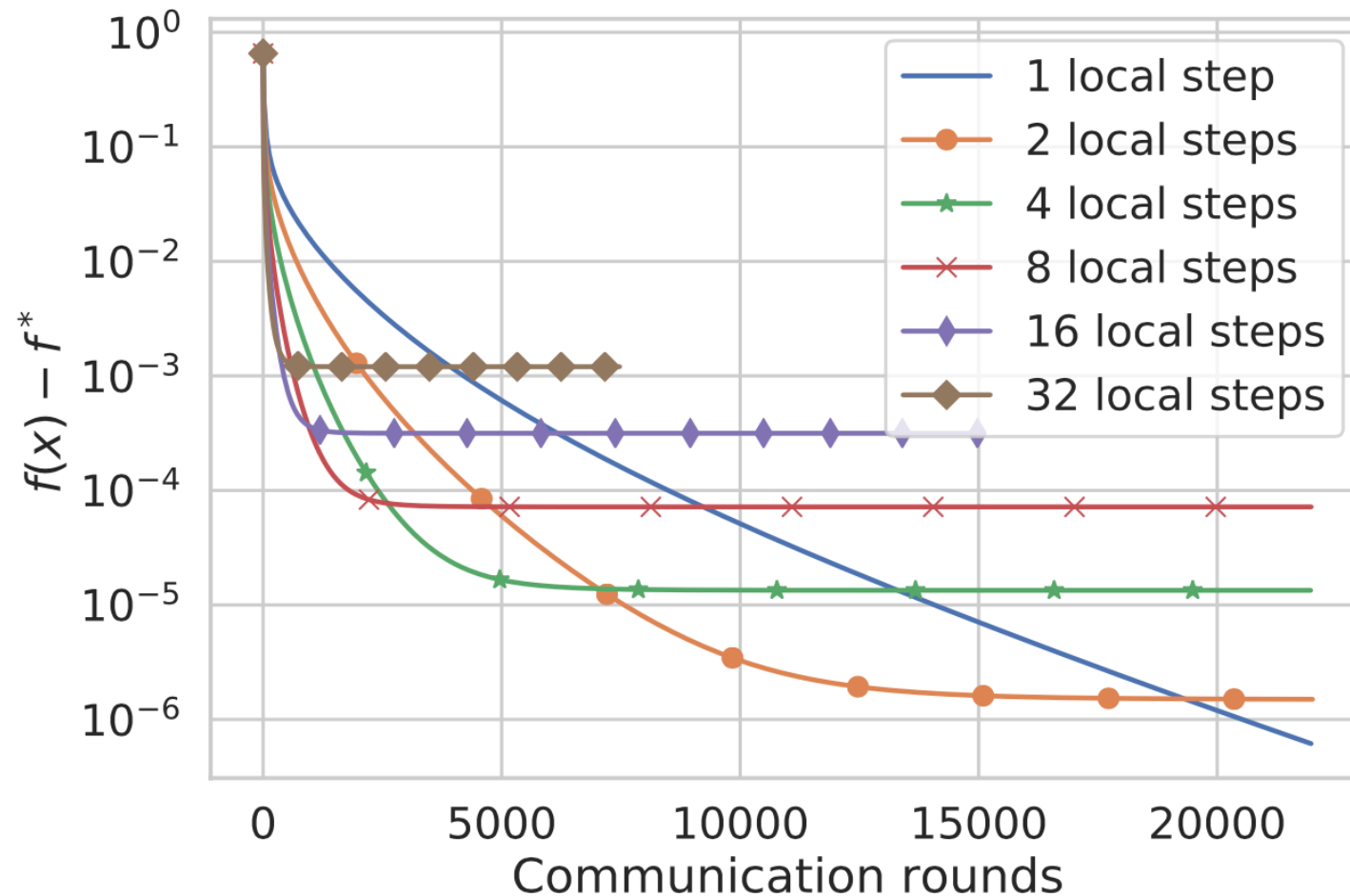
2 PROBLEM SETTING

When training DNNs for speech recognition, the immediate problem is that of choosing vectors $x \in \mathbb{R}^D$ as corresponding to classes $k \in \{1, \dots, K\}$. The dimension D is typically around hundred.

1

Brief History of Local Training Methods

Generation 1: Heuristic



L2-regularized logistic regression
LibSVM mushrooms dataset

Brief History of Local Training Methods

Generation 2: Homogeneous

“Theory requires data to be similar/homogeneous across the clients”

07/2019



Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang and Zhihua Zhang
On the Convergence of FedAvg on Non-IID Data
ICLR 2020

Bounded gradients:

$$\|\nabla f_i(x)\| \leq B \quad \forall x \in \mathbb{R}^d \quad \forall i \in \{1, 2, \dots, n\}$$

10/2019



Farzin Haddadpour and Mehrdad Mahdavi
On the Convergence of Local Descent Methods in Federated Learning
arXiv:1910.14425, 2019

Bounded gradient diversity (aka strong growth):

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq C \|\nabla f(x)\|^2 \quad \forall x \in \mathbb{R}^d$$

Brief History of Local Training Methods

Generation 3: Sublinear

“Heterogeneous data is allowed, but the rate is worse than GD”

10/2019



Ahmed Khaled, Konstantin Mishchenko and P.R.

First Analysis of Local GD on Heterogeneous Data

NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality, 2019

10/2019



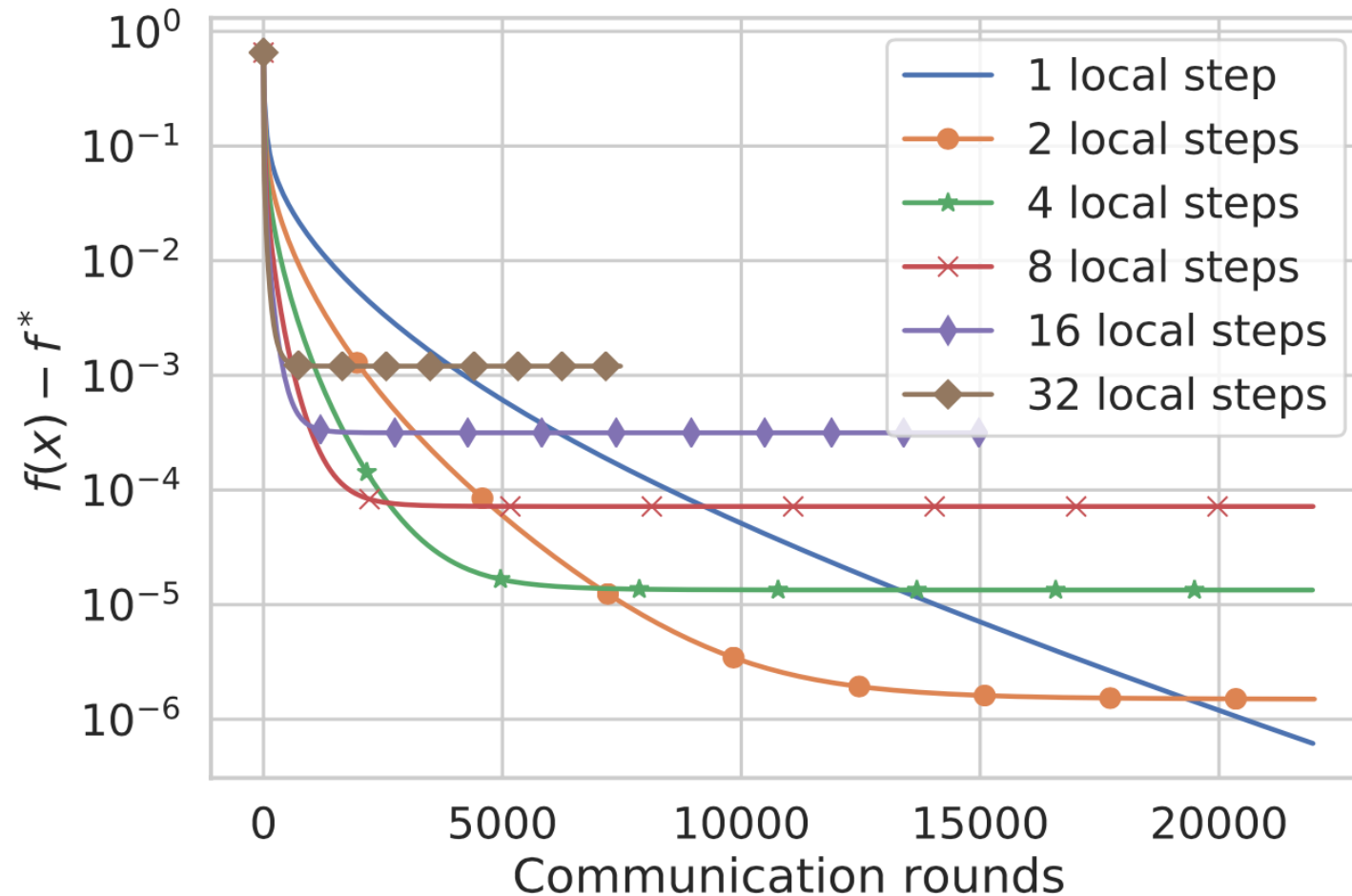
Ahmed Khaled, Konstantin Mishchenko and P.R.

Tighter Theory for Local SGD on Identical and Heterogeneous Data

AISTATS 2020

Brief History of Local Training Methods

Generation 3: Sublinear



L2-regularized logistic regression
LibSVM mushrooms dataset

Brief History of Local Training Methods

Generation 4: Linear

“Heterogeneous data is allowed, but the rate at best matches that of GD”

10/2019
Scaffold



Sai P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh
SCAFFOLD: Stochastic Controlled Averaging for Federated Learning
ICML 2020

11/2020
S-Local-GD, Local-GD*
S-Local-SVRG



Eduard Gorbunov, Filip Hanzely and P.R.
Local SGD: Unified Theory and New Efficient Methods
AISTATS 2021

02/2021
FedLin



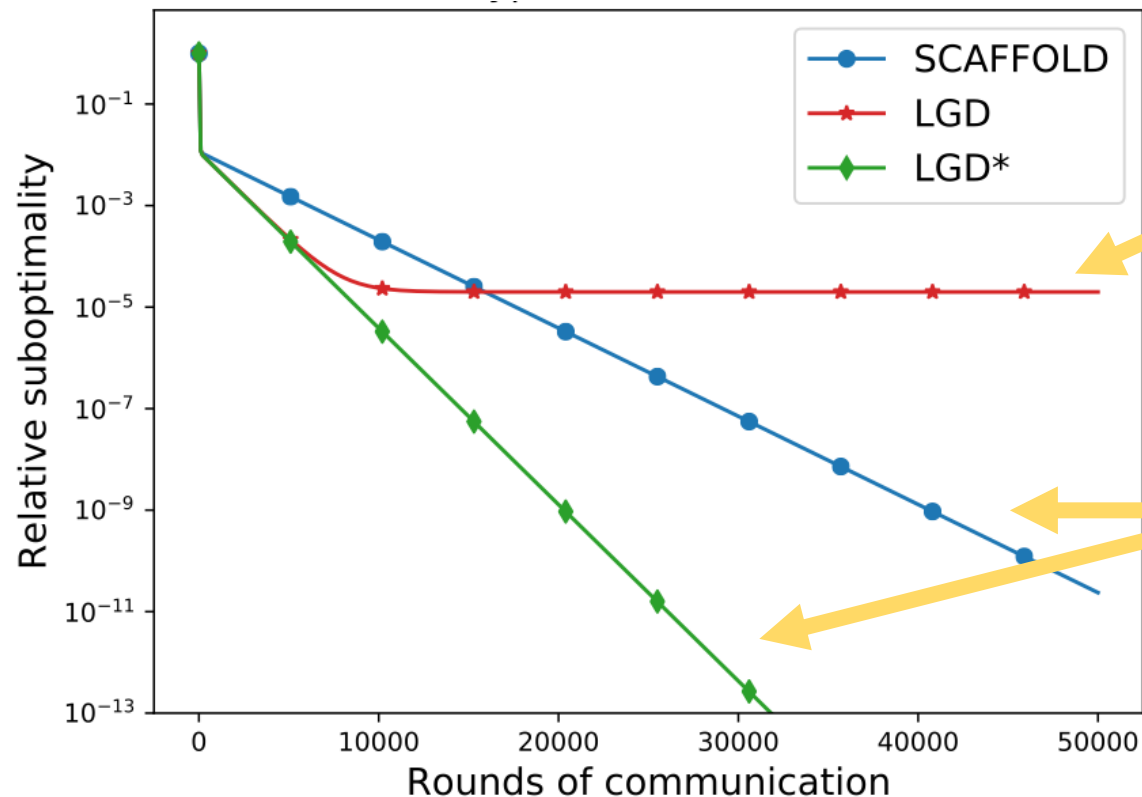
Aritra Mitra, Rayana Jaafar, George J. Pappas, Hamed Hassani
Linear Convergence in Federated Learning: Tackling Client Heterogeneity & Sparse Gradients
NeurIPS 2021

Method	$\alpha^*, \beta^*, \eta^*$	Complexity	Setting	Sec
Local-SGD, Alg. 1 (Karimireddy et al., 2020)	$f_0(\alpha^*), 0, -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	UBV, CHet	G.1.1
Local-SGD, Alg. 1 (Karimireddy et al., 2020)	$f_0(\alpha^*), 0, -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	UBV, Het	G.1.1
Local-SGD, Alg. 1 (Karimireddy et al., 2020)	$f_0(\alpha^*), 0, -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	ES, CHet	G.1.2
Local-SGD, Alg. 1 (Karimireddy et al., 2020)	$f_0(\alpha^*), 0, -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	ES, Het	G.1.2
Local-SVRG, Alg. 2 (new)	$\frac{\nabla f_0(\alpha^*) - \nabla f_0(\alpha^*)}{\nabla f_0(\alpha^*)}$	$m + \frac{L(d^2 + \epsilon^2)}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	simple, CHet	G.2
Local-SVRG, Alg. 2 (new)	$\frac{\nabla f_0(\alpha^*) - \nabla f_0(\alpha^*)}{\nabla f_0(\alpha^*)}$	$m + \frac{L(d^2 + \epsilon^2)}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	simple, Het	G.2
S-Local-SGD, Alg. 3 (new)	$f_0(\alpha^*), \nabla f_0(\alpha^*), -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	UBV, Het	G.3
SS-Local-SGD, Alg. 4 (Karimireddy et al., 2021)	$f_0(\alpha^*), \nabla f_0(\alpha^*), -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	UBV, Het	G.4.1
SS-Local-SGD, Alg. 4 (new)	$f_0(\alpha^*), \nabla f_0(\alpha^*), -$	$\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}}$	ES, Het	G.4.2
S-Local-SGD*, Alg. 5 (new)	$\frac{\nabla f_0(\alpha^*) - \nabla f_0(\alpha^*)}{\nabla f_0(\alpha^*)}$	$\left(\frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}} \right) \log \frac{1}{\epsilon}$	simple, Het	G.5
S-Local-SVRG, Alg. 6 (new)	$\nabla f_0(\alpha^*) - \nabla f_0(\alpha^*)$	$\left(m + \frac{L}{\epsilon} + \frac{d^2}{\epsilon^2} + \sqrt{\frac{L(d^2 + \epsilon^2)}{\epsilon^2}} \right) \log \frac{1}{\epsilon}$	simple, Het	G.6

Brief History of Local Training Methods

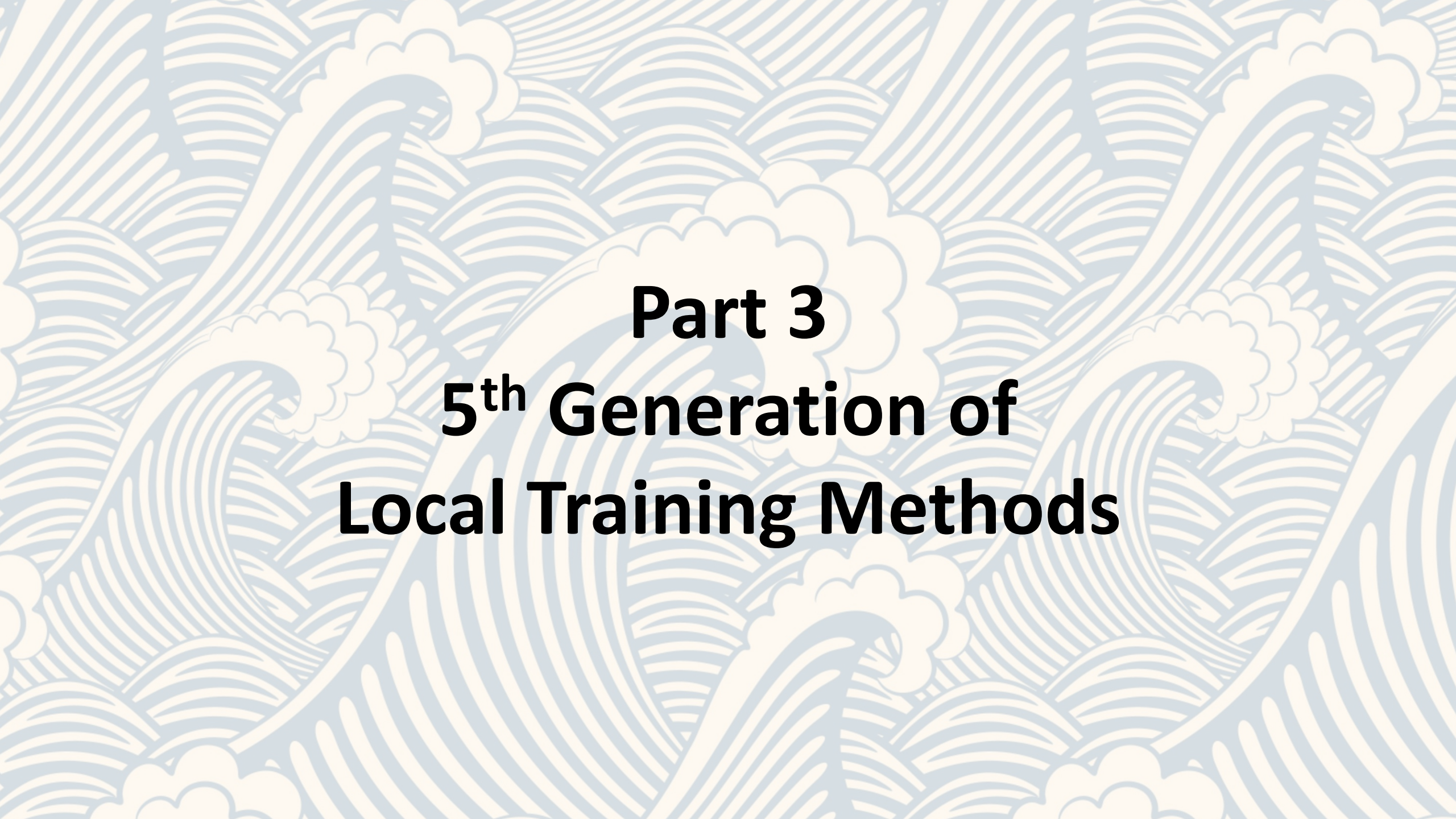
Generation 4: Linear

“Heterogeneous data is allowed, but the rate at best matches that of GD”



Generation 3

Generation 4



Part 3

5th Generation of Local Training Methods

Brief History of Local Training Methods

Generation 5: Accelerated

“Communication complexity is better than GD for heterogeneous data”



In practice, local training significantly improves communication efficiency.

However, there is no theoretical result explaining this!

Is the situation hopeless, or can we show/prove that local training helps?

Key Property of 5th Generation Local Training Methods

Communication complexity
of 4th generation
local training methods

$$\mathcal{O} \left(\frac{L}{\mu} \log \frac{1}{\varepsilon} \right)$$

Communication complexity
of 5th generation
local training methods

$$\mathcal{O} \left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon} \right)$$

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!†

Konstantin Mishchenko¹ Grigory Malinovsky² Sebastian Stich³ Peter Richtárik²

Abstract

We introduce **ProxSkip**—a surprisingly simple and provably efficient method for minimizing the sum of a smooth (f) and an expensive nonsmooth proximable (ψ) function. The canonical approach to solving such problems is via the proximal gradient descent (**ProxGD**) algorithm, which is based on the evaluation of the gradient of f and the prox operator of ψ in each iteration. In this work we are specifically interested in the regime in which the evaluation of prox is costly relative to the evaluation of the gradient, which is the case in applications. **ProxSkip** allows for the expensive operator to be skipped in most iterations: its iteration complexity is $\mathcal{O}(\kappa \log 1/\epsilon)$, where κ is the condition number of f , the number of evaluations is $\mathcal{O}(\sqrt{\kappa} \log 1/\epsilon)$ on ψ . Our motivation comes from federated learning, where evaluation of the gradient operator corresponds to exchanging a local GD step independently on all devices and evaluation of prox corresponds to exchanging communication in the form of gradient evaluations. In this context, **ProxSkip** offers a *relative acceleration* of communication complexity. Unlike other local gradient-type methods such as **FedAvg**, **SCAFFOLD**, **S-Local-GD** and others, whose theoretical communication complexity is worse than, or at best matching, that of **ProxGD** in the heterogeneous data regime, we achieve a provable and large improvement with heterogeneous data-bounding assumptions.

where $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is a smooth function, and $\psi: \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex regularizer.

Such problem are ubiquitous, and appear in numerous applications associated with virtually all areas of science and engineering, including signal processing (Combettes & Pesquet, 2009), image processing (Luke, 2020), data science (Parikh & Boyd, 2014) and machine learning (Shalev-Shwartz & Ben-David, 2014).

1.1. Proximal gradient descent

† Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods*.

1. Introduction

We study optimization problems of the form

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \quad (1)$$

¹INRS, ENS, Inria Sierra, Paris, France ²Computer Science, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia ³CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Peter Richtárik <peter.richtarik@kaust.edu.sa>.

† Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods*.

$\text{prox}_{\gamma\psi}$. This is the case for many regularizers, including the L_1 norm ($\psi(x) = \|x\|_1$), the L_2 norm ($\psi(x) = \|x\|_2^2$), and elastic net (Zhou & Hastie, 2005). For many further examples, we refer the reader to the books (Parikh & Boyd, 2014; Beck, 2017).

1.2. Expensive proximity operators

However, in this work we are interested in the situation when the evaluation of the *proximity operator* is *expensive*. That is, we assume that the computation of $\text{prox}_{\gamma\psi}$ (the backward step) is costly relative to the evaluation of the gradient of f (the forward step).

A conceptually simple yet rich class of expensive proximity operators arises from regularizers ψ encoding a

The Beginning



ICML
International Conference
On Machine Learning



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.
ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!
ICML 2022

Brief History of Local Training Methods

Generation 5: Accelerated

“Communication complexity is better than GD for heterogeneous data”

02/2022

ProxSkip



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

ICML 2022

07/2022

APDA; APDA-Inexact



Abdurakhmon Sadiev, Dmitry Kovalev and P.R.

Communication Acceleration of Local Gradient Methods via an Accelerated Primal-Dual Algorithm with Inexact Prox

NeurIPS 2022

07/2022

ProxSkip-LSVRG



Grigory Malinovsky, Kai Yi and P.R.

Variance Reduced ProxSkip: Algorithm, Theory and Application to Federated Learning

NeurIPS 2022

07/2022

RandProx



Laurent Condat and P.R.

RandProx: Primal-Dual Optimization Algorithms with Randomized Proximal Updates

ICLR 2023

Brief History of Local Training Methods

Generation 5: Accelerated

“Communication complexity is better than GD for heterogeneous data”

10/2022

GradSkip



Artavazd Maranjyan, Mher Safaryan and P.R.

GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

arXiv:2210.16402, 2022

10/2022

Compressed-
Scaffnew



Laurent Condat, Ivan Agarský and P.R.

Provably Doubly Accelerated Federated Learning: The First Theoretically Successful Combination of Local Training and Compressed Communication

arXiv:2210.13277, 2022

12/2022

5GCS



Michal Grudzien, Grigory Malinovsky and P.R.

Can 5th Generation Local Training Methods Support Client Sampling? Yes!

AISTATS 2023

02/2023

TAMUNA



Laurent Condat, Grigory Malinovsky and P.R.

TAMUNA: Accelerated Federated Learning with Local Training and Partial Participation

arXiv:2302.09832, 2023

Brief History of Local Training Methods

Generation 5: Accelerated

	# Comm. Rounds	Local Optimizer	# Local Training Steps	Total Complexity (Comm. + Compute)	Client Sampling?	Comm. Compression?	Supports Decentralized Setup?	Key Insight
ProxSkip 2/22, ICML 22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	$\sqrt{\frac{L}{\mu}}$	=	✗	✗	✓	First 5th generation local training method
APDA-Inexact 7/22, NeurIPS 22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	any	better	better	✗	✗	✓	Can use more powerful local solvers which take fewer local GD-type steps
VR-ProxSkip 7/22, NeurIPS 22	worse	VR-SGD	worse	better	✗	✗	✗	Running variance reduced SGD locally can lead to better total complexity than ProxSkip
RandProx 7/22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	$\sqrt{\frac{L}{\mu}}$	=	✗	✗	✓	ProxSkip = VR mechanism for compressing the prox
GradSkip 10/22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	better	better	✗	✗	✗	Workers containing less important data can do fewer local training steps!
Compressed Scaffnew 10/22	worse	GD	worse	better	✗	✓	✗	Can compress uplink, leads to better overall communication complexity than ProxSkip.
5GCS 10/22	worse	any	$\sqrt{\frac{L}{\mu}}$	worse	✓	✗	✗	Can do client sampling
TAMUNA 2/23	worse	GD / SGD	worse	better	✓	✓	✗	First 5th gen LT method that can do CS + CC!

Part 4

ProxSkip: Local Training Provably Leads to Communication Acceleration



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

ICML 2022

Federated Learning: ProxSkip vs Baselines

Table 1. The performance of federated learning methods employing multiple local gradient steps in the strongly convex regime.

	method	# local steps per round	# floats sent per round	stepsize on client i	linear rate?	# rounds	rate better than GD?
Gen 3	GD (Nesterov, 2004)	1	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
	LocalGD (Khaled et al., 2019; 2020)	τ	d	$\frac{1}{\tau L}$	✗	$\mathcal{O}\left(\frac{G^2}{\mu n \tau \varepsilon}\right)$ ^(d)	✗
Gen 4	Scaffold (Karimireddy et al., 2020)	τ	$2d$	$\frac{1}{\tau L}$ ^(e)	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
	S-Local-GD ^(a) (Gorbunov et al., 2021)	τ	$d < \# < 2d$ ^(f)	$\frac{1}{\tau L}$	✓	$\tilde{\mathcal{O}}(\kappa)$	✗
	FedLin ^(b) (Mitra et al., 2021)	τ_i	$2d$	$\frac{1}{\tau_i L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
Gen 5	Scaffnew ^(g) (this work) for any $p \in (0, 1]$	$\frac{1}{p}$ ^(h)	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}\left(p\kappa + \frac{1}{p}\right)$ ^(c)	✓ (for $p > \frac{1}{\kappa}$)
	Scaffnew ^(g) (this work) for optimal $p = \frac{1}{\sqrt{\kappa}}$	$\sqrt{\kappa}$ ^(h)	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\sqrt{\kappa})$ ^(c)	✓

^(a) This is a special case of S-Local-SVRG, which is a more general method presented in (Gorbunov et al., 2021). S-Local-GD arises as a special case when full gradient is computed on each client.

^(b) FedLin is a variant with a fixed but different number of local steps for each client. Earlier method S-Local-GD has the same update but random loop length.

^(c) The $\tilde{\mathcal{O}}$ notation hides logarithmic factors.

^(d) G is the level of dissimilarity from the assumption $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq G^2 + 2LB^2 (f(x) - f_*)$, $\forall x$.

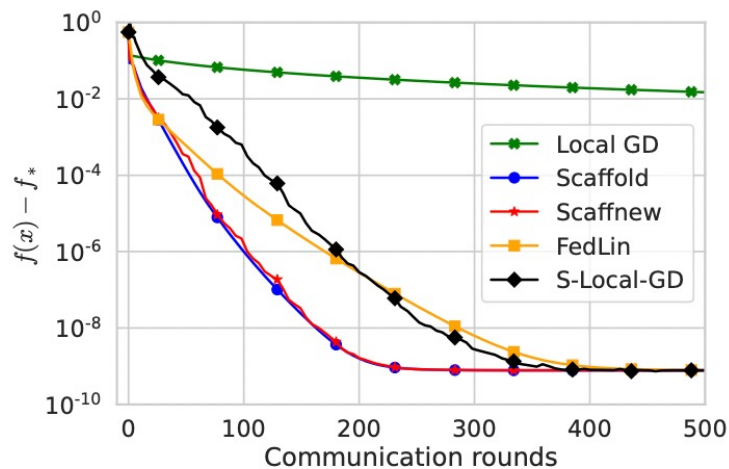
^(e) We use Scaffold's cumulative local-global stepsize $\eta_i \eta_g$ for a fair comparison.

^(f) The number of sent vectors depends on hyper-parameters, and it is randomized.

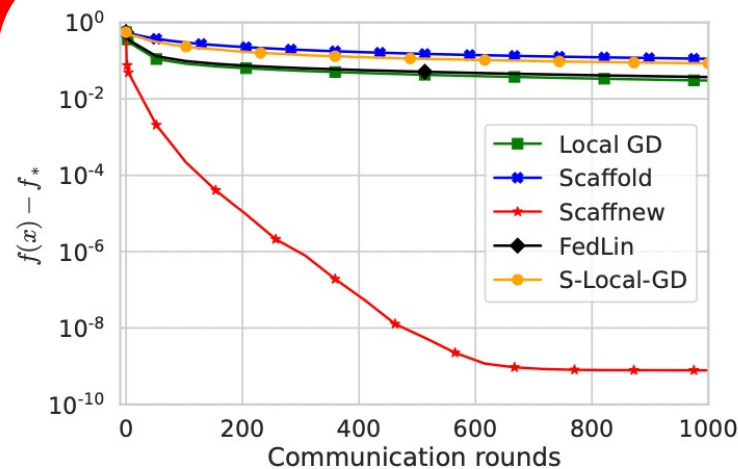
^(g) Scaffnew (Algorithm 2) = ProxSkip (Algorithm 1) applied to the consensus formulation (6) + (7) of the finite-sum problem (5).

^(h) ProxSkip (resp. Scaffnew) takes a *random* number of gradient (resp. local) steps before prox (resp. communication) is computed (resp. performed). What is shown in the table is the *expected* number of gradient (resp. local) steps.

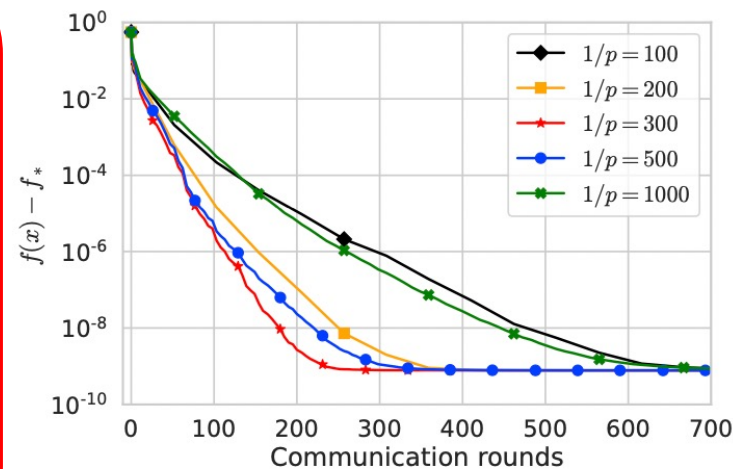
ProxSkip + Deterministic Gradients



(a) tuned hyper-parameters



(b) theoretical hyper-parameters



(c) different options of p

Figure 1. Deterministic Case. Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs. In (c), we compare communication rounds with the algorithm stepsize set to the best theoretical stepsize and different options of parameter p .

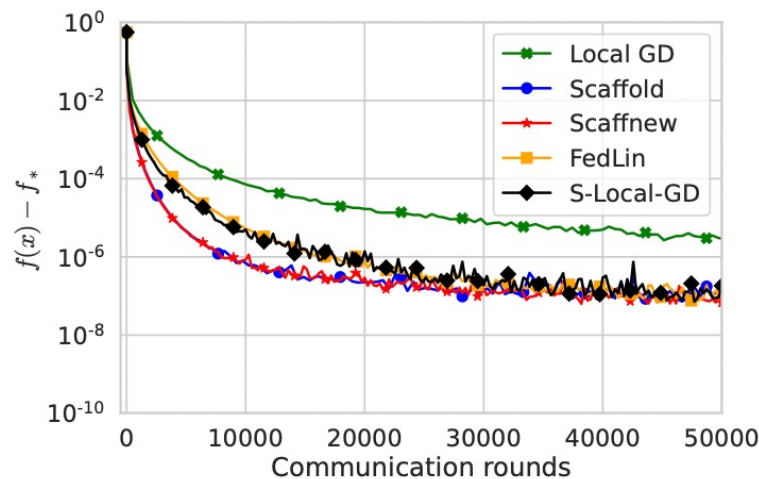
L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

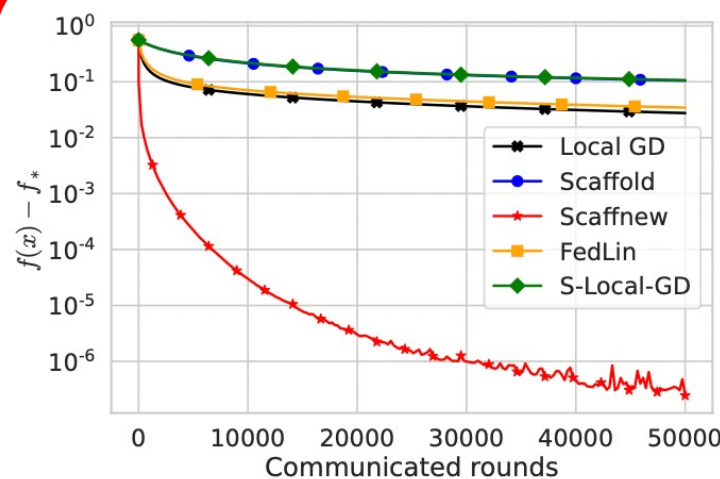
$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)

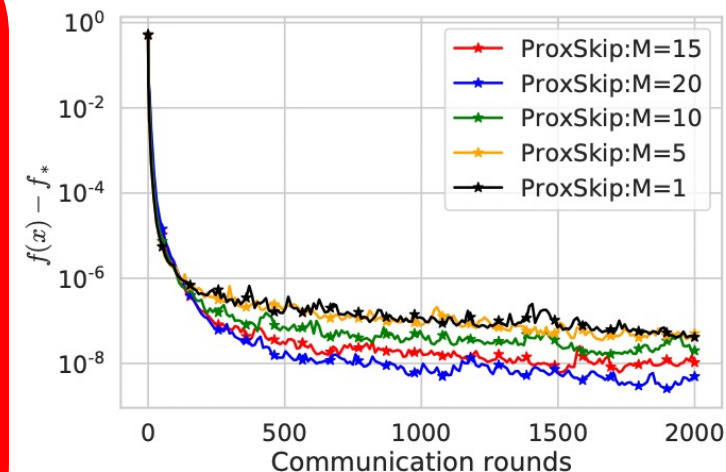
ProxSkip + Stochastic Gradients



(a) tuned hyper-parameters



(b) theoretical hyper-parameters



(c) different number of clients

Figure 2. Stochastic Case. Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalSGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs. In (c), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs and different number of clients.

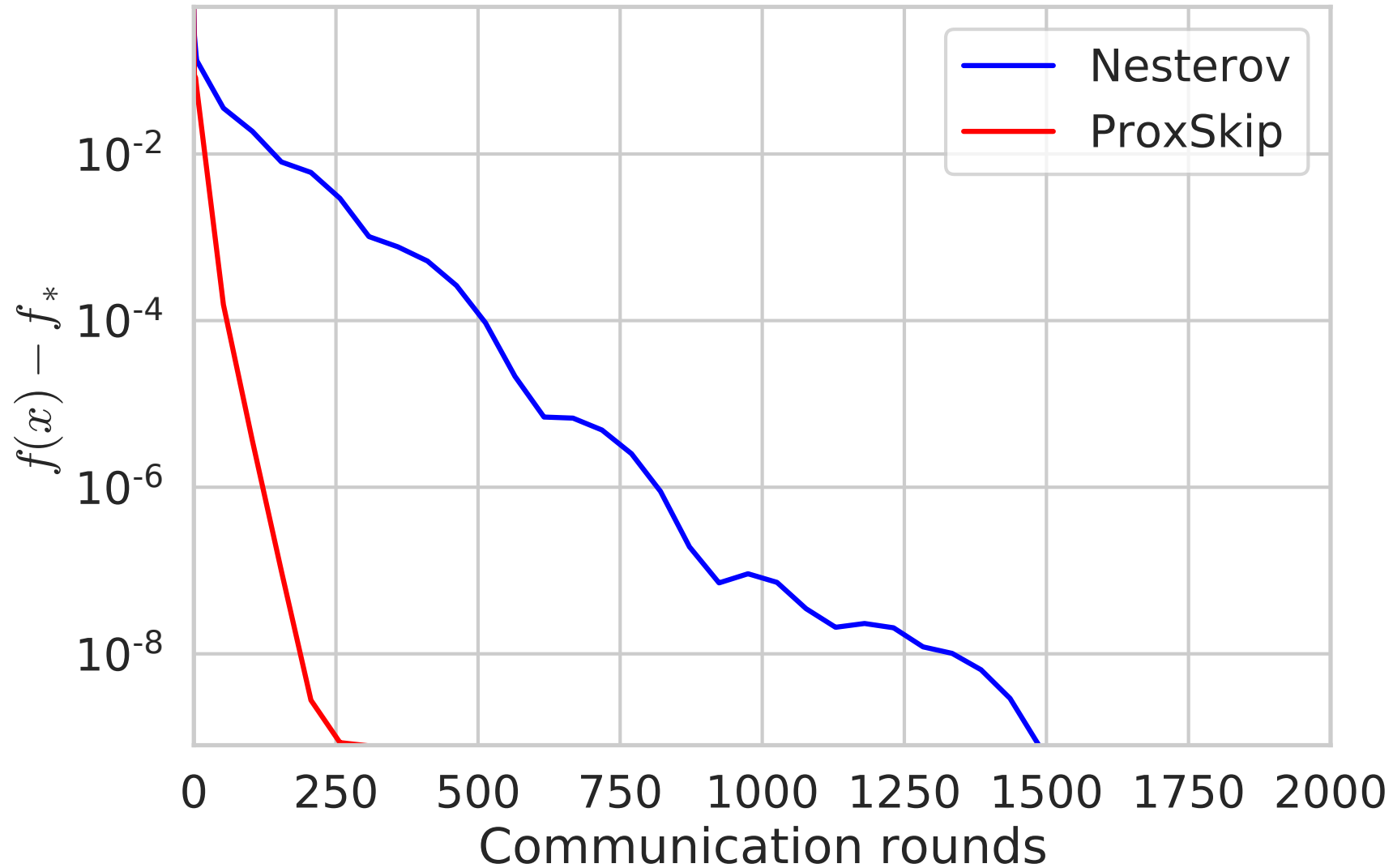
L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)

ProxSkip vs Nesterov's Accelerated GD



Consensus Reformulation

Original problem:
optimization in \mathbb{R}^d

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

Consensus reformulation:
optimization in \mathbb{R}^{nd}

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

Bad: non-differentiable

Good: Indicator function of a nonempty closed convex set

$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x_1 = \dots = x_n, \\ +\infty, & \text{otherwise.} \end{cases}$$

Consensus Reformulation

Original problem:
optimization in \mathbb{R}^d

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

Bad: non-differentiable

Good: proper closed convex

Consensus reformulation:
optimization in \mathbb{R}^{nd}

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

$\psi(x_1, \dots, x_n) : \mathbb{R}^{nd} \rightarrow \mathbb{R} \cup \{+\infty\}$
is a proper closed convex function

$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \mid \psi(x) \leq t\}$ The epigraph of ψ is a closed and convex set

Three Assumptions

The epigraph of ψ is a closed and convex set

$$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \in \mathbb{R}^d \times \mathbb{R} \mid \psi(x) \leq t\}$$

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$

A1 f is μ -convex and L -smooth:

$$\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$$

A2 $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is proper, closed, and convex

A3 ψ is proximable

Bregman divergence of f :

$$D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(y), x - y \rangle$$

The proximal operator $\text{prox}_\psi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ defined by

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left(\psi(u) + \frac{1}{2} \|u - x\|^2 \right)$$

can be evaluated exactly (e.g., in closed form)

Key Method: Proximal Gradient Descent

proximal operator:

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left(\psi(u) + \frac{1}{2} \|u - x\|^2 \right)$$



stepsize

$$x_t - \gamma \nabla f(x_t)$$



gradient operator

$$x \mapsto x - \gamma \nabla f(x)$$

Proximal Gradient Descent: Theory

f is μ -convex and L -smooth:
 $\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$
 $\frac{L}{\mu}$ is the condition number of f

Theorem:

$$t \geq \frac{L}{\mu} \log \frac{1}{\varepsilon} \quad \Rightarrow \quad \|x_t - x_\star\|^2 \leq \varepsilon \|x_0 - x_\star\|^2$$

(for stepsize $\gamma = \frac{1}{L}$)

iterations

Error tolerance

$$x_\star \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$

What to do When the Prox is Expensive?

Can we somehow get away with
fewer evaluations of the proximity operator
in the Proximal GD method?

Approach 1



We'll skip ALL prox evaluations!



The method is NOT implementable!



Serves as an inspiration for Approach 2

Approach 2 (ProxSkip)



We'll skip MANY prox evaluations!



The method is implementable!

Approach 1:
Simple, Extreme but
Practically Useless Variant

Removing ψ via a Reformulation

$$\min_{x \in \mathbb{R}^d} f(x) - \langle h_\star, x \rangle$$

$$\begin{aligned} h_\star &\stackrel{\text{def}}{=} \nabla f(x_\star) \\ x_\star &\stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x) \end{aligned}$$



x_\star is a solution of the above problem!

By the 1st order optimality conditions, the solution satisfies $\nabla f(x) - \nabla f(x_\star) = 0$



We do not know $h_\star = \nabla f(x_\star)$!

Apply Gradient Descent to the Reformulation

$$\begin{aligned} h_\star &\stackrel{\text{def}}{=} \nabla f(x_\star) \\ x_\star &\stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x) \end{aligned}$$

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_\star)$$



We do not need to evaluate the prox of ψ at all!



We do not know h_\star and hence can't implement the method!

Approach 2:

The ProxSkip Method

Idea: Try to “Learn” the Optimal Gradient Shift

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

Desire: $h_t \rightarrow h_\star$



Perhaps we can learn h_\star with only occasional access to ψ ?

ProxSkip: Bird's Eye View

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$

1

$$\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

2a

with probability $1 - p$ do
 $1 - p \approx 1$

$$x_{t+1} = \hat{x}_{t+1}$$

$$h_{t+1} = h_t$$

2b

with probability p do
 $p \approx 0$

evaluate $\text{prox}_{\frac{\gamma}{p}\psi}(?)$

$$x_{t+1} = ?$$

$$h_{t+1} = ?$$

ProxSkip: The Algorithm (Detailed View)

Algorithm 1 ProxSkip

1: stepsize $\gamma > 0$, probability $p > 0$, initial iterate $x_0 \in \mathbb{R}^d$, initial control variate $h_0 \in \mathbb{R}^d$, number of iterations $T \geq 1$
2: **for** $t = 0, 1, \dots, T - 1$ **do**
3: $\hat{x}_{t+1} = x_t - \gamma(\nabla f(x_t) - h_t)$ ◇ Take a gradient-type step adjusted via the control variate h_t
4: Flip a coin $\theta_t \in \{0, 1\}$ where $\text{Prob}(\theta_t = 1) = p$ ◇ Flip a coin that decides whether to skip the prox or not
5: **if** $\theta_t = 1$ **then**
6: $x_{t+1} = \text{prox}_{\frac{\gamma}{p}\psi}(\hat{x}_{t+1} - \frac{\gamma}{p}h_t)$ ◇ Apply prox, but only very rarely! (with small probability p)
7: **else**
8: $x_{t+1} = \hat{x}_{t+1}$ ◇ Skip the prox!
9: **end if**
10: $h_{t+1} = h_t + \frac{p}{\gamma}(x_{t+1} - \hat{x}_{t+1})$ ◇ Update the control variate h_t
11: **end for**

ProxSkip: Bounding the # of Iterations

Theorem:

f is μ -convex and L -smooth:
 $\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$
 $\frac{L}{\mu}$ is the condition number of f

$$t \geq \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \log \frac{1}{\varepsilon} \implies \mathbb{E} [\Psi_t] \leq \varepsilon \Psi_0$$

iterations

p = probability of
evaluating the prox

Lyapunov function:

$$\Psi_t \stackrel{\text{def}}{=} \|x_t - x_\star\|^2 + \frac{1}{L^2 p^2} \|h_t - h_\star\|^2$$

ProxSkip: Optimal Prox-Evaluation Probability

Since in each iteration we evaluate the prox with probability p , the expected number of prox evaluations after t iterations is:

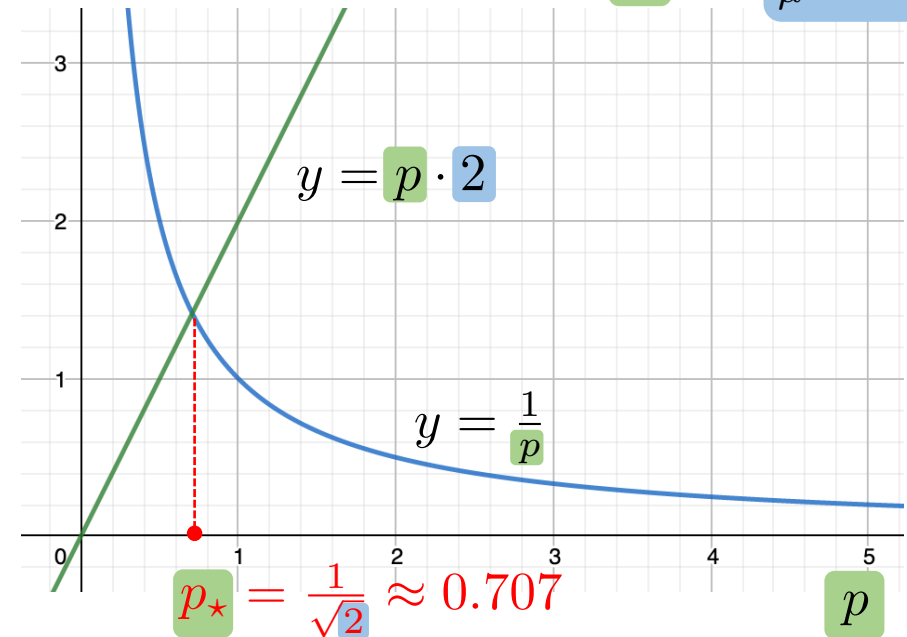
$\frac{L}{\mu}$ is the condition number of f

$$p \cdot t = p \cdot \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \cdot \log \frac{1}{\varepsilon} = \max \left\{ p \cdot \frac{L}{\mu}, \frac{1}{p} \right\} \cdot \log \frac{1}{\varepsilon}$$

Minimized for p satisfying $p \cdot \frac{L}{\mu} = \frac{1}{p}$

$$\Rightarrow p_{\star} = \frac{1}{\sqrt{L/\mu}}$$

Computation of optimal p_{\star} for $\frac{L}{\mu} = 2$



From Gradients to Stochastic Gradients

- As described, in ProxSkip each worker computes the **full gradient** of its local function
- It's often better to consider a **cheap stochastic approximation of the gradient** instead
 - We consider this extension in the paper
 - We provide theoretical convergence rates

$$\nabla f_i(x_t) \Rightarrow g_i(x_t)$$

Full gradient Stochastic gradient

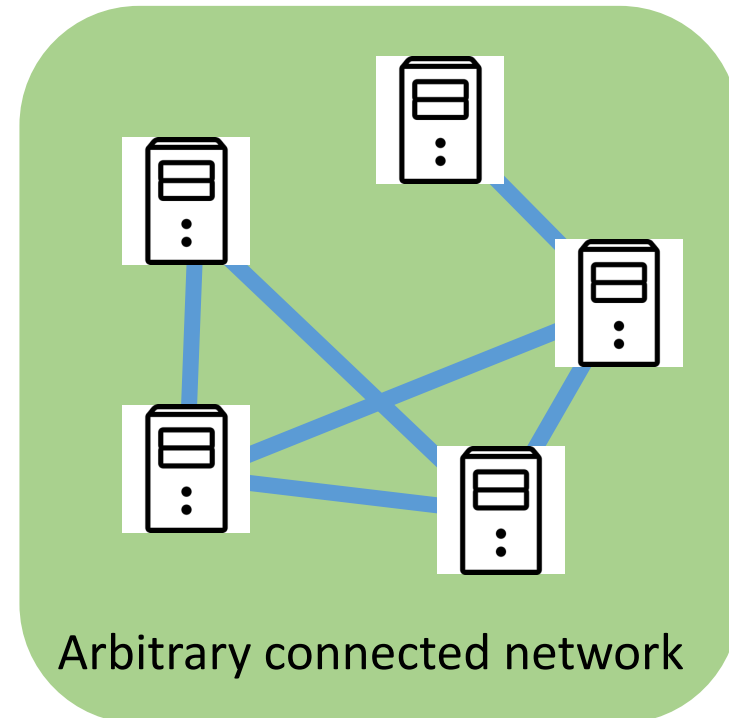
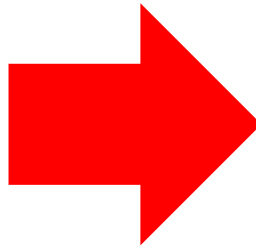
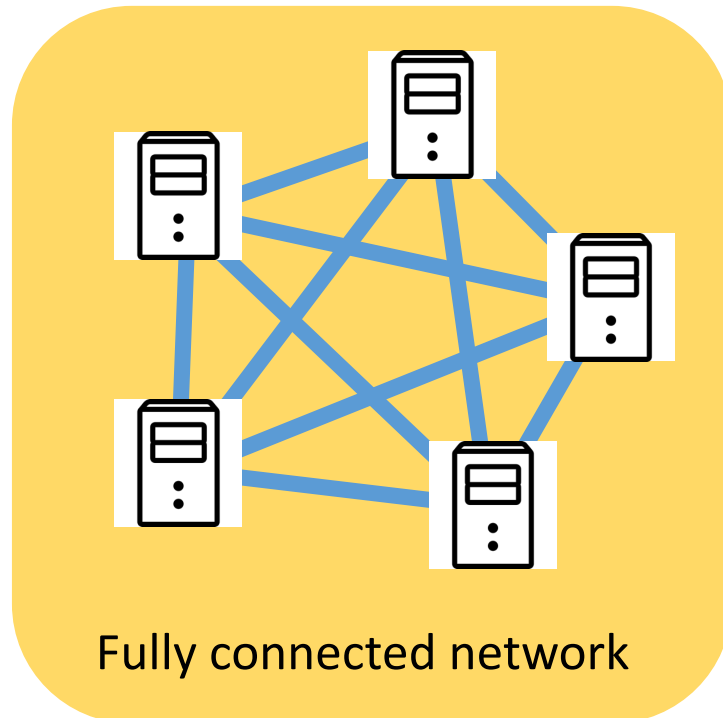
Assumptions:

(unbiasedness) $\mathbb{E}[g_{i,t}(x_t) \mid x_t] = \nabla f_i(x_t)$

(expected smoothness) $\mathbb{E}[\|g_{i,t}(x_t) - \nabla f(x_*)\|^2 \mid x_t] \leq 2AD_f(x_t, x_*) + C$
(Gower et al, 2019)

From Fully Connected Networks to Arbitrary Connected Networks

- In each communication round of ProxSkip, **each worker sends messages to all other workers** (e.g., through a server).
 - We can think of ProxSkip workers as the nodes of a **fully-connected network**.
 - In each communication round, all **workers communicate with their neighbors**.
- In the paper we provide extension to **arbitrary connected networks**.



“Philosophically” Related Literature

Gradient Sliding by **George Lan**



Guanghui Lan

Gradient Sliding for Composite Optimization

Mathematical Programming 2016



Guanghui Lan and Yi Zhou

Conditional Gradient Sliding for Convex Optimization

SIAM Journal on Optimization 2016



Guanghui Lan and Yuyuan Ouyang

Accelerated Gradient Sliding for Structured Convex Optimization

COAP 2022



Dmitry Kovalev, Aleksandr Beznosikov, Ekaterina Dmitrievna Borodich, Alexander Gasnikov, Gesualdo Scutari

Optimal Gradient Sliding and its Application to Optimal Distributed Optimization Under Similarity

NeurIPS 2022

Part 5

RandProx: Primal–Dual Optimization Algorithms with Randomized Proximal Updates

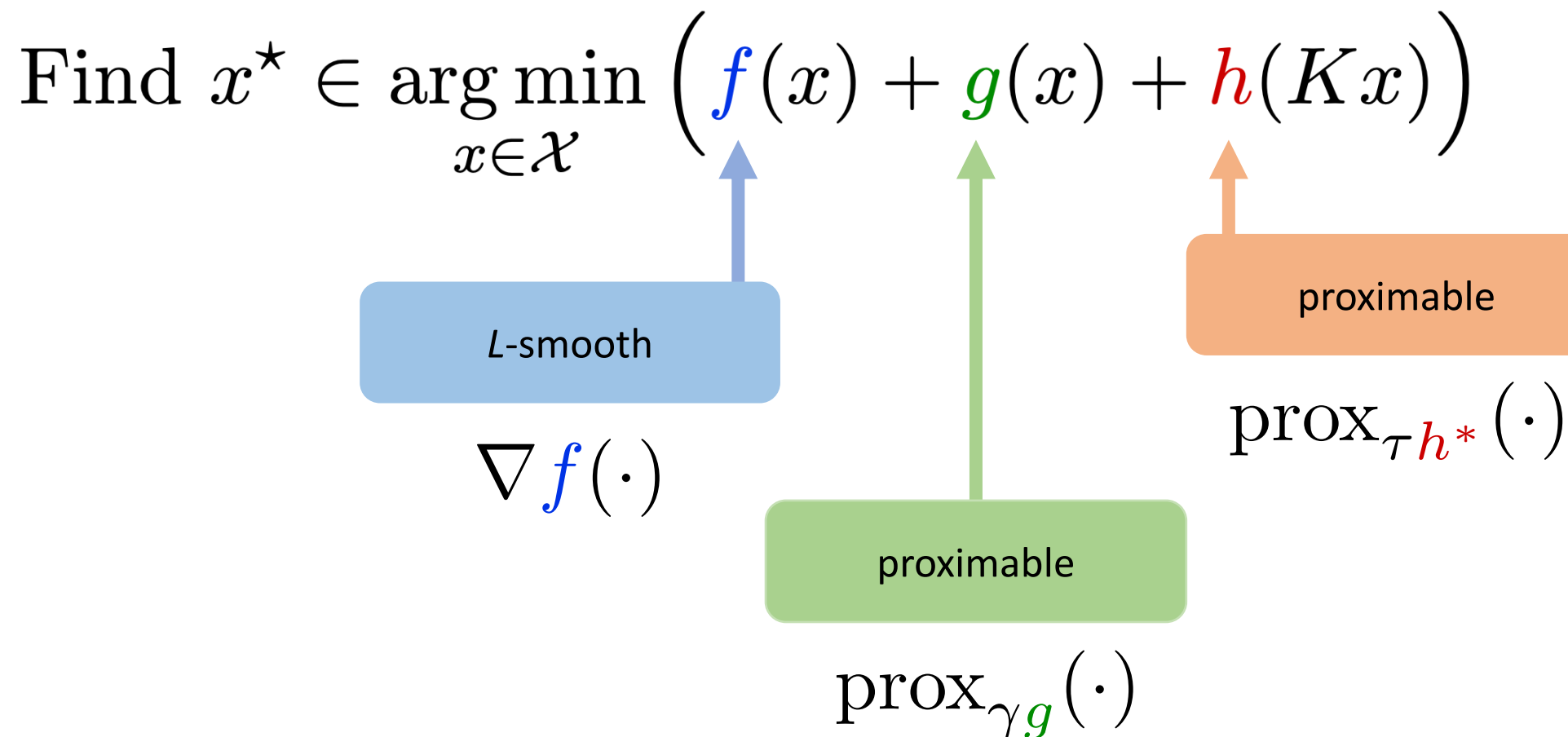


Laurent Condat and P.R.

RandProx: Primal–Dual Optimization Algorithms with Randomized Proximal Updates

ICLR 2023 (arXiv:2207.12891)

Minimization of the Sum of Three Functions



PDDY: Primal-Dual Davis Yin Splitting

$$\text{Find } x^* \in \arg \min_{x \in \mathcal{X}} \left(\textcolor{blue}{f}(x) + \textcolor{green}{g}(x) + \textcolor{red}{h}(Kx) \right)$$

Algorithm **PDDY** [Salim et al., 2022b]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;

stepsizes $\gamma > 0$, $\tau > 0$

for $t = 0, 1, \dots$ **do**

$$\hat{x}^t := \text{prox}_{\gamma \textcolor{green}{g}} \left(x^t - \gamma \nabla \textcolor{blue}{f}(x^t) - \gamma K^* u^t \right)$$

$$\textcolor{yellow}{u^{t+1}} := \text{prox}_{\tau \textcolor{red}{h}^*} \left(\textcolor{yellow}{u^t} + \tau K \hat{x}^t \right)$$

$$x^{t+1} := \hat{x}^t - \gamma K^* (u^{t+1} - u^t)$$

end for



Adil Salim, Laurent Condat, Konstantin Mishchenko and P.R.

Dualize, Split, Randomize: Toward Fast Nonsmooth Optimization Algorithms

JOTA 2022

Special Cases of PDDY

$$\arg \min_{x \in \mathcal{X}} \left(\underset{\text{blue}}{f}(x) + \underset{\text{green}}{g}(x) + \underset{\text{red}}{h}(Kx) \right)$$

f	g	h	K	Deterministic algorithm
any	any	any	any	PDDY
any	0	any	any	PAPC
any	0	any	Id	forward-backward (FB)
any	0	$\iota_{\{b\}}$	any	PAPC
0	any	any	any	Chambolle–Pock (CP)
0	any	any	Id	ADMM
any	any	any	Id	Davis–Yin (DY)

[Salim, Condat, Mishchenko & R 2022]

[Drori, Sabach & Teboulle 2015]

[Drori, Sabach & Teboulle 2015]

[Chambolle & Pock 2011]

[Boyd et al 2011]

[Davis & Yin 2017]

From PDDY to RandProx: Compress the Prox!

Algorithm **PDDY** [Salim et al., 2022b]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;
 stepsizes $\gamma > 0$, $\tau > 0$
for $t = 0, 1, \dots$ **do**
 $\hat{x}^t := \text{prox}_{\gamma g}(x^t - \gamma \nabla f(x^t) - \gamma K^* u^t)$
 $u^{t+1} := \text{prox}_{\tau h^*}(u^t + \tau K \hat{x}^t)$
 $x^{t+1} := \hat{x}^t - \gamma K^*(u^{t+1} - u^t)$
end for

Algorithm **RandProx** [new]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;
 stepsizes $\gamma > 0$, $\tau > 0$; $\omega \geq 0$
for $t = 0, 1, \dots$ **do**
 $\hat{x}^t := \text{prox}_{\gamma g}(x^t - \gamma \nabla f(x^t) - \gamma K^* u^t)$
 $u^{t+1} := u^t + \frac{1}{1+\omega} \mathcal{R}^t(\text{prox}_{\tau h^*}(u^t + \tau K \hat{x}^t) - u^t)$
 $x^{t+1} := \hat{x}^t - \gamma (1 + \omega) K^*(u^{t+1} - u^t)$
end for

Variance reduction technique known as EF21:



P. R., Igor Sokolov and Ilyas Fatkhullin

EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback

NeurIPS 2021

Compression operator: $\mathbb{E} [\mathcal{R}^t(x)] = x$ $\mathbb{E} [\|\mathcal{R}^t(x) - x\|^2] \leq \omega \|x\|^2$

Part 6

GradSkip: Clients with Less Important Data can do Less Local Training

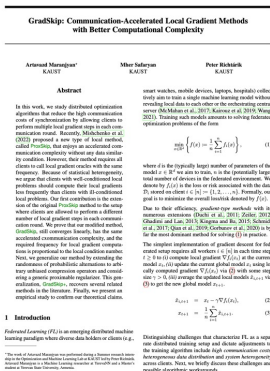
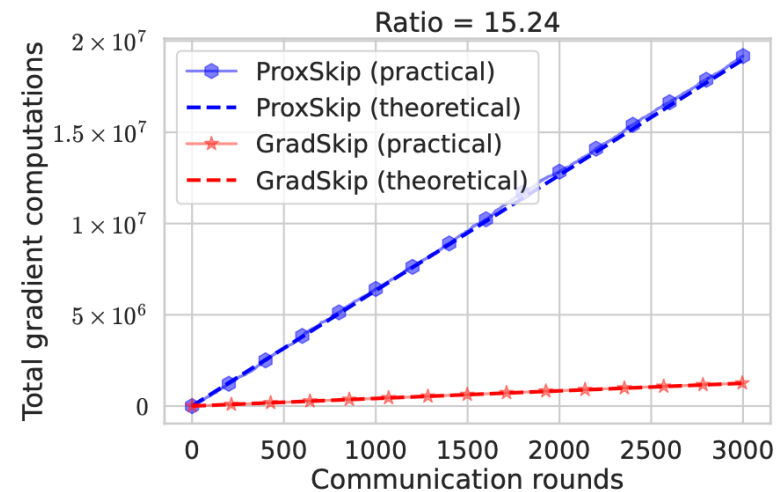
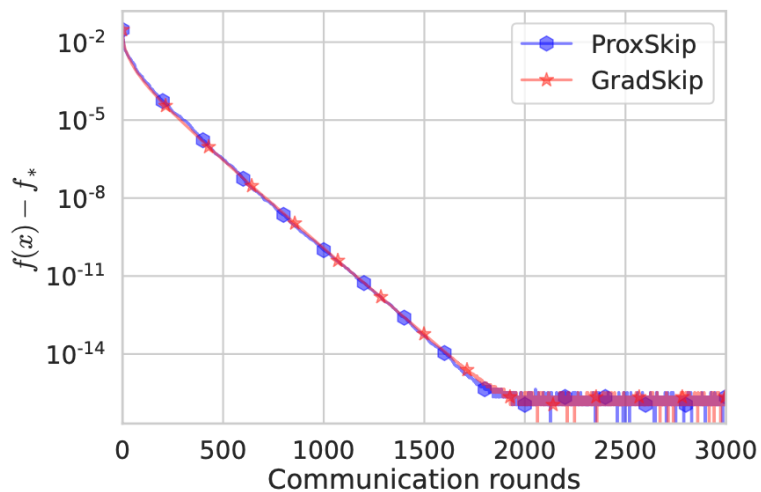
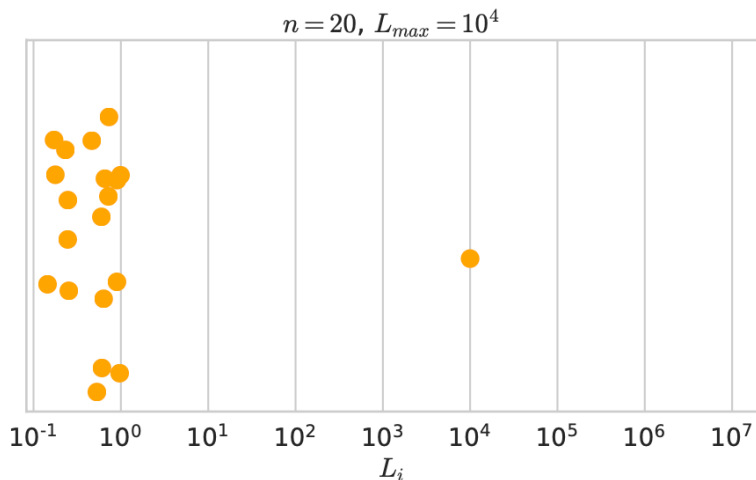


Artavazd Maranjyan, Mher Safaryan and P.R.

GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

arXiv:2210.16402, 2022

GradSkip



Algorithm 2 GradSkip+

- Parameters:** stepsize $\gamma > 0$, compressors $\mathcal{C}_\omega \in \mathbb{B}^d(\omega)$ and $\mathcal{C}_\Omega \in \mathbb{B}^d(\Omega)$.
- Input:** initial iterate $x_0 \in \mathbb{R}^d$, initial control variate $h_0 \in \mathbb{R}^d$, number of iterations $T \geq 1$.
- for** $t = 0, 1, \dots, T - 1$ **do**
- $\hat{h}_{t+1} = \nabla f(x_t) - (\mathbf{I} + \Omega)^{-1} \mathcal{C}_\Omega (\nabla f(x_t) - h_t)$ ◇ Update the shift $\hat{h}_{i,t}$ via shifted compression
- $\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - \hat{h}_{t+1})$ ◇ Update the iterate $\hat{x}_{i,t}$ via shifted gradient step
- $\hat{g}_t = \frac{1}{\gamma(1+\omega)} \mathcal{C}_\omega \left(\hat{x}_{t+1} - \text{prox}_{\gamma(1+\omega)\psi} \left(\hat{x}_{t+1} - \gamma(1+\omega) \hat{h}_{t+1} \right) \right)$ ◇ Estimate the proximal gradient
- $x_{t+1} = \hat{x}_{t+1} - \gamma \hat{g}_t$ ◇ Update the main iterate $x_{i,t}$
- $h_{t+1} = \hat{h}_{t+1} + \frac{1}{\gamma(1+\omega)} (x_{t+1} - \hat{x}_{t+1})$ ◇ Update the main shift $h_{i,t}$
- end for**



Artavazd Maranjyan, Mher Safaryan and P.R.

GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

arXiv:2210.16402, 2022



The End