Accelerated Methods with Compression for Horizontal and Vertical Federated Learning

Sergey Stanko, Timur Karimullin, Aleksandr Beznosikov

Moscow Institute of physics and technology

Introduction

To address the problem of the time-consuming process of function minimization, the community came to distributed algorithms, where the calculation process is divided among different devices. Such parallel computation can be used in situations, where data is distributed across several machines, as in the case of federated learning approach. The latter can be divided into two different regimes. One of these is horizontal federated learning, where each worker possesses their own collection of samples, but share the same set of features. A different way of data division is considered in the vertical case, where every device has a unique set of features of the same samples.

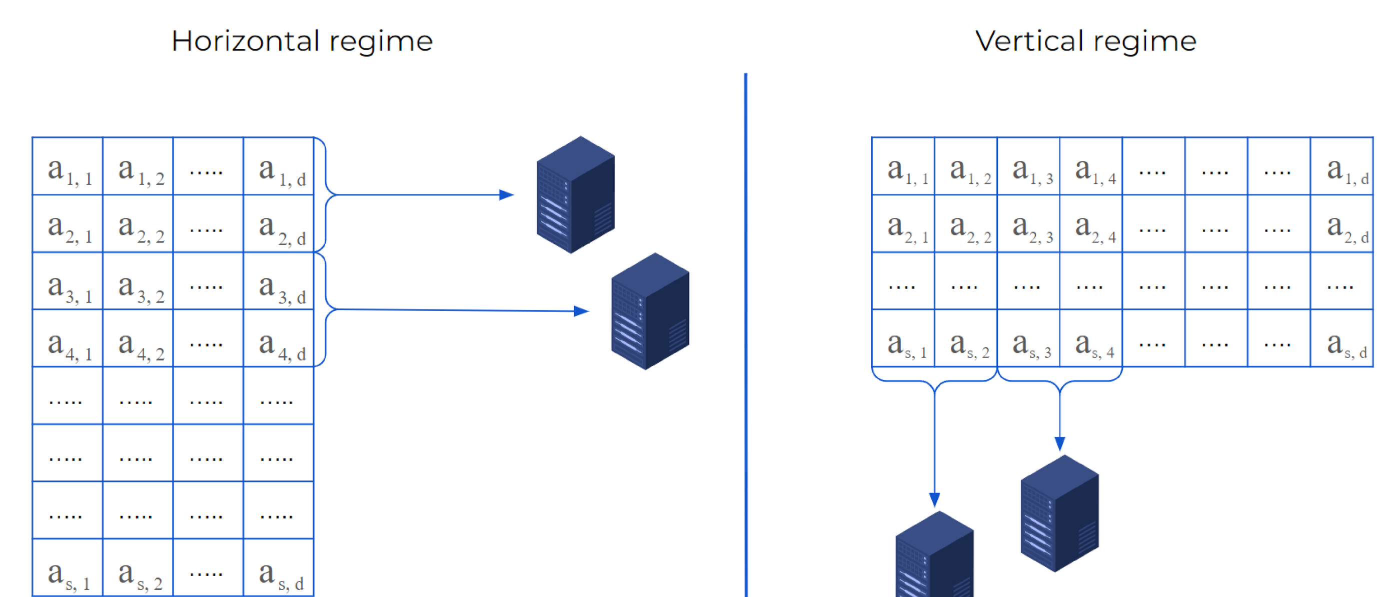


Figure 1. Difference between regimes of data division

Compression

Despite the considerable practical utility of distributed methods, they are not without significant drawbacks. In particular, the parallelisation of a task does not necessarily result in an optimal reduction in time. That means that having n devices does not accelerate task by n times. This happens because of limited ability of networks to exchange information. Thus, the key bottleneck of parallel computation is the communication part. There have been considered several ways of dealing with this issue [2], but in our paper we concentrate solely on reducing communication cost of each iteration by decreasing the size of sending information also known as compression technique [1].

The main results

Table 1: Summary of bounds for iteration complexities for finding an ε-solution. Convergence is measured by the distance to the solution.

Regime	Reference	Iteration complexity
Horizontal	QSGD Alistarh et al. [2017] (1)	$\mathcal{O}\left(\frac{L}{\mu}\left(1+\frac{\omega}{n}\right)\log\frac{1}{\varepsilon}+\frac{\sigma_*^2\omega}{\mu^2\varepsilon}\right)$
	DIANA Mishchenko et al. [2023]	$\mathcal{O}\left(\left(\frac{L}{\mu}\left(1+\frac{2\omega}{n}\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
	VR-DIANA Horváth et al. [2019]	$\mathcal{O}\left(\left(\frac{L}{\mu}\left(1+\frac{\omega}{n}\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
	ADIANA Li et al. [2020]	$\mathcal{O}\left(\left(\sqrt{\frac{L}{\mu}}\left(1+\sqrt{\left(\frac{\omega}{n}+\sqrt{\frac{\omega}{n}}\right)\omega}\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
	ADIANA He et al. [2024]	$\mathcal{O}\left(\left(\sqrt{\frac{L}{\mu}}\left(\sqrt{\frac{\omega^2}{n}}+1\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
	MARINA Gorbunov et al. [2022] (2)	$\mathcal{O}\left(\left(\frac{L}{\mu}\left(1+\frac{\omega}{\sqrt{n}}\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
	MASHA Beznosikov et al. [2023] (3)	$\mathcal{O}\left(\left(\frac{L}{\mu}\sqrt{\left(w+\frac{\omega^2}{n}\right)}+\omega\right)\log\frac{1}{\varepsilon}\right)$
	Three Pillars Algorithm Beznosikov and Gasnikov [2023] (3, 4)	$\mathcal{O}\left(\left(\frac{L}{\mu}\sqrt{n}+n\right)\log\frac{1}{\varepsilon}\right)$
	Algorithm 1(This paper)	$\mathcal{O}\left(\left(\sqrt{\frac{L}{\mu}}\left(\sqrt{\frac{\omega^2}{n}}+1\right)+\omega\right)\log\frac{1}{\varepsilon}\right)$
Vertical	AVFL Cai et al. [2022]; CE-VFL Sun et al. [2023]; SecureBoost+ Chen et al. [2021]; eHE-SecureBoost Xu et al. [2021];	No theoretical results
	CVFL Castiglia et al. [2023]	No concrete number of iterations (5)
	Algorithm 2(This paper) (4)	$\mathcal{O}\left(\left(\sqrt{\frac{L\omega}{\mu}}+s\right)\log\frac{1}{\varepsilon}\right)$

(1) with tuned step size, no convergence with fixed step size; (2) under Polyak-Lojasiewicz condition; (3) for variational inequalities and saddle point problems; (4) for special compressor only; (5) for special compressor convergence rate is $\mathcal{O}(\frac{1}{\sqrt{N}})$, no guaranties in the case of arbitrary compressor;

Notation: μ = constant of strong convexity, L = smoothness constant, ω = compression constant (see Definition 1.1), n = number of workers, σ_* = sum of workers' gradients in the optimal point, s = total number of samples.

Problem statement

We state the standard distributed learning problem, which, formally, can be written in the following form.

$$\min_{x\in\mathbb{R}^d}\left[f(x):=\frac{1}{n}\sum_{i=1}^{n}f_i(x)\right].$$

We assume functions to have the following properties.

Definition 2: The functions $f_i:\mathbb{R}^d\to\mathbb{R}$ are L-smooth for some $L>0, \forall i\in\overline{1,n}$:

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

Definition 3: The function $f:\mathbb{R}^d\to\mathbb{R}$ is μ - strongly convex for some $\mu>0$:

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

Compressor properties

Compression mathematically can be represented in the form of a vector function. In our paper, we consider them having the following properties:

Definition 1 We say that the compression operator Q is unbiased, if

$$\mathbb{E}\left[Q(x)\right]=x\ ,\forall x\in\mathbb{R}^d.$$

We also assume that there exists a constant $\omega\ge 0$, such:

$$\mathbb{E}\left[\|Q(x)-x\|^2\right]\le\omega\|x\|^2\ ,\forall x\in\mathbb{R}^d.$$

DHPL-Katyusha

The `DHPL-Katyusha`, the algorithm for the horizontal data division regime, is inspired by the original `L-Katyusha` [3]. We use the variance reduction technique for compressor's stochastics, thus in our method, every device calculates the compressed gradient difference and broadcasts it to other workers. After that, the new points are found using the momentum part, similar to `L-Katyusha`.

Algorithm 1 `DHPL-Katyusha`

Parameters: $\theta_1,\theta_2,\gamma\in(0,1]$, probability $p\in(0,1]$ (every worker has the same random seed for random, connected with p).

Initialization: Choose $y^0=w^0=z^0\in\mathbb{R}^d$, stepsize $\eta=\frac{\theta_2\gamma}{(1+\theta_2)\theta_1}$, set $\widetilde{L}=L\max\left\{\frac{\omega}{n},1\right\}$, $\sigma=\frac{\mu}{\widetilde{L}}$.

for $k=0,1,2,\dots K$ **do**
  for $i=1\dots n$ in parallel **do**
    $x^k\leftarrow\theta_1 z^k+\theta_2 w^k+(1-\theta_1-\theta_2)y^k$
    $g_i^k\leftarrow\nabla f_i(x^k)-\nabla f_i(w^k)$
    $\widetilde{g}_i^k\leftarrow Q_i(g_i^k)$
    Using communications broadcast $\widetilde{g}_i^k$
    Compute $\widetilde{g}^k\leftarrow\frac{1}{n}\sum_{i=1}^{n}\widetilde{g}_i^k+\nabla f(w^k)$
    $z^{k+1}\leftarrow\frac{1}{1+\eta\sigma}(\eta\sigma x^k+z^k-\frac{\eta}{\widetilde{L}}\widetilde{g}^k)$
    $y^{k+1}\leftarrow x^k+\theta_1(z^{k+1}-z^k)$
    $w^{k+1}\leftarrow\begin{cases}y^k, & \text{with probability } p\\ w^k, & \text{with probability } 1-p\end{cases}$
    if $w^{k+1}=y^k$ **then**
      Using communications broadcast $\nabla f_i(w^{k+1})$
      Compute $\nabla f(w^{k+1})=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(w^{k+1})$
    end if
  end for
end for

DVPL-Katyusha

In `DVPL-Katyusha` every worker has its own set of features. Therefore, unlike the horizontal regime, in the vertical case all operations are performed on subvectors, corresponding to individual worker's components.

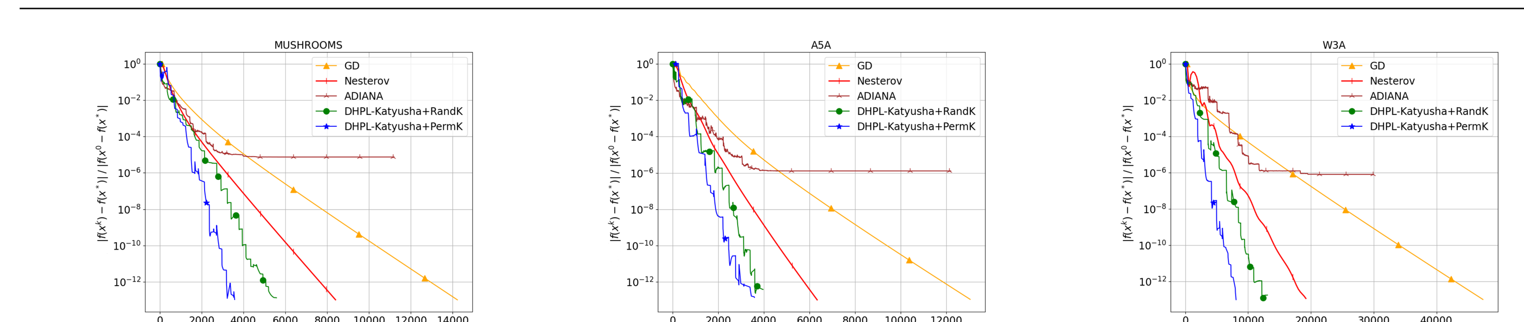
Algorithm 2 `DVPL-Katyusha`

Parameters: $\theta_1,\theta_2,\gamma\in(0,1]$, probability $p\in(0,1]$, RandK select j-th sample with probability $p_j=\frac{L_j}{s\bar{L}}$(every worker has the same random seed for random, connected with p and p_j).

Initialization: Choose $y^0=w^0=z^0\in\mathbb{R}^d$, stepsize $\eta=\frac{\theta_2\gamma}{(1+\theta_2)\theta_1}$, set $\widetilde{L}=\max\left\{\frac{\bar{L}}{K},L\right\}$, $\sigma=\frac{\mu}{\widetilde{L}}$.

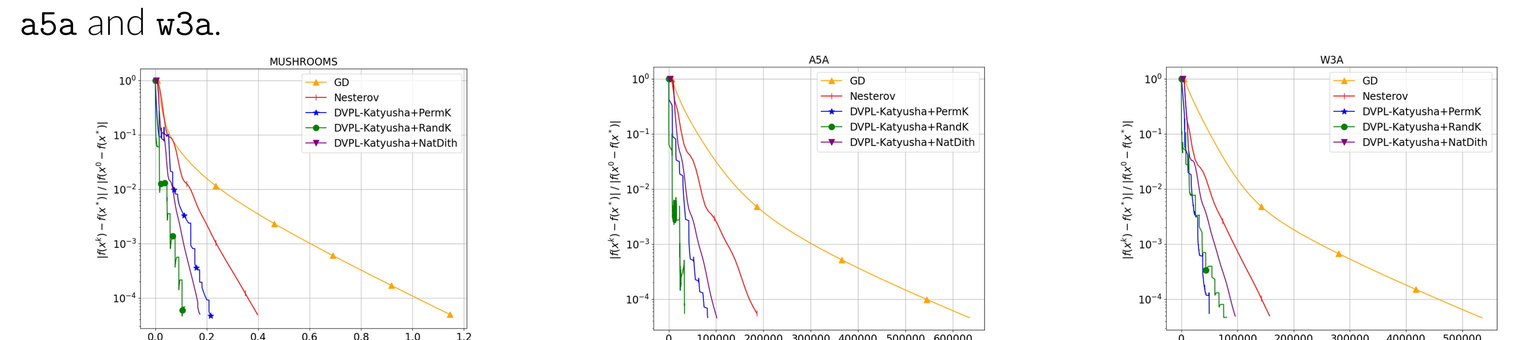
for $k=0,1,2,\dots K$ **do**
  for $i=1\dots n$ in parallel **do**
    $x_i^k\leftarrow\theta_1 z_i^k+\theta_2 w_i^k+(1-\theta_1-\theta_2)y_i^k$
    Compute $X_i^k=\mathrm{RandK}\left(\left\|\left\langle A_{ji}^T,x_i^k\right\rangle\right\|_{j=\overline{1,s}}\right)$
    Compute $W_i^k=\mathrm{RandK}\left(\left\|\left\langle A_{ji}^T,w_i^k\right\rangle\right\|_{j=\overline{1,s}}\right)$
    Using communications broadcast X_i^k and W_i^k
    $J^k=\{j_1^k,\cdots,j_n^k\}$ - indices, selected by RandK
    $\widetilde{g}_i^k\leftarrow\frac{1}{K}\sum_{j\in J^k}\frac{1}{sp_j}\nabla\mathcal{L}_j\left(\sum_{i=1}^{n}X_{ij}^k,b_j\right)_i-\frac{1}{K}\sum_{j\in J^k}\frac{1}{sp_j}\nabla\mathcal{L}_j\left(\sum_{i=1}^{n}W_{ij}^k,b_j\right)_i+\nabla\mathcal{L}\left(Aw^k,b\right)_i$
    $z_i^{k+1}\leftarrow\frac{1}{1+\eta\sigma}(\eta\sigma x_i^k+z_i^k-\frac{\eta}{\widetilde{L}}\widetilde{g}_i^k)$
    $y_i^{k+1}\leftarrow x_i^k+\theta_1(z_i^{k+1}-z_i^k)$
    $w_i^{k+1}\leftarrow\begin{cases}y_i^k, & \text{with probability } p\\ w_i^k, & \text{with probability } 1-p\end{cases}$
    if $w_i^{k+1}=y_i^k$ **then**
      for $j=1\dots s$ **do**
        Compute $\left\langle A_{ji}^T,w_i^k\right\rangle$
        Using communications broadcast $\left\langle A_{ji}^T,w_i^k\right\rangle$
      end for
      Compute $\nabla\mathcal{L}\left(Aw^k,b\right)_i$
    end if
  end for
end for

Numerical experiments



(a) `mushrooms` (b) `a5a` (c) `w3a`

Figure 2. Comparison of different algorithms in horizontal case on LIBSVM datasets `mushrooms`, `a5a` and `w3a`.



(a) `mushrooms` (b) `a5a` (c) `w3a`

Figure 3. Comparison of different algorithms in vertical case on LIBSVM datasets `mushrooms`, `a5a` and `w3a`.

References

[1] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX symposium on operating systems design and implementation (OSDI 14)*, pages 571–582, 2014.

[2] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[3] Dmitry Kovalev, Samuel Horvath, and Peter Richtarik. Don't jump through hoops and remove those loops: Svrg and katyusha are better without the outer loop, 2019.