Different Aspects of Communication Organization in Distributed and Federated learning

Optimization methods

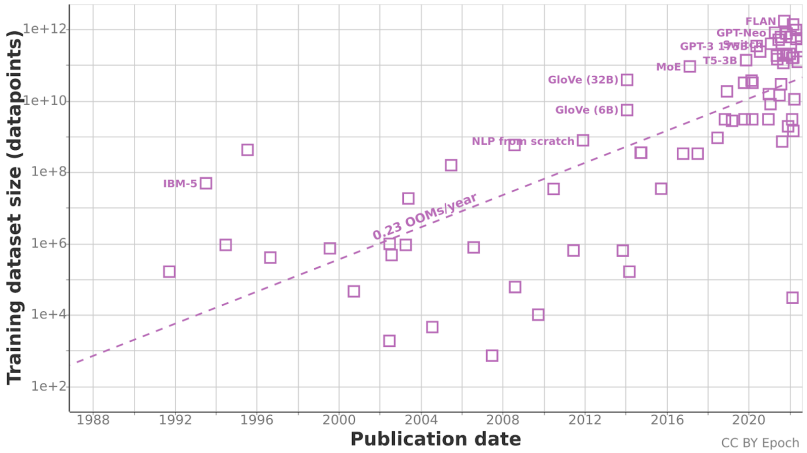
Aleksandr Beznosikov

MIPT

29 April 2025

Modern trends in machine learning

- Exponential growth in model sizes and data volumes.



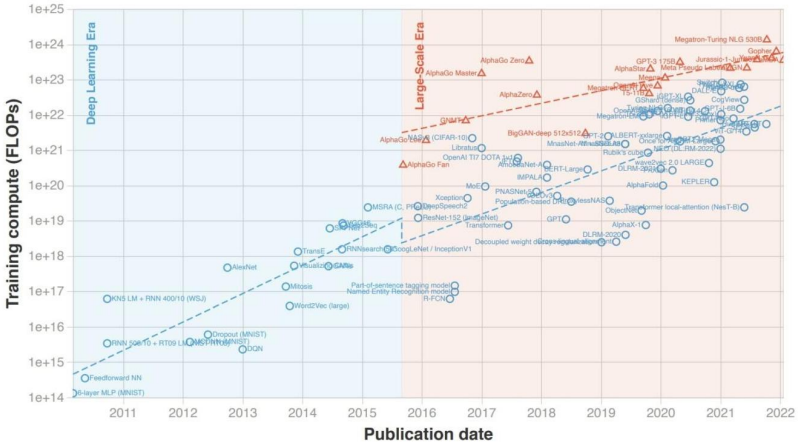


Figure: Trends in machine learning tasks

Varieties of distributed learning

- Cluster learning (big players): training within one large and powerful computing cluster
- Collaborative learning (all players): pooling computing resources over the Internet

Varieties of distributed learning

- Cluster learning (big players): training within one large and powerful computing cluster
- Collaborative learning (all players): pooling computing resources over the Internet
- Federated learning (another paradigm): learning from users' local data using their computing powers

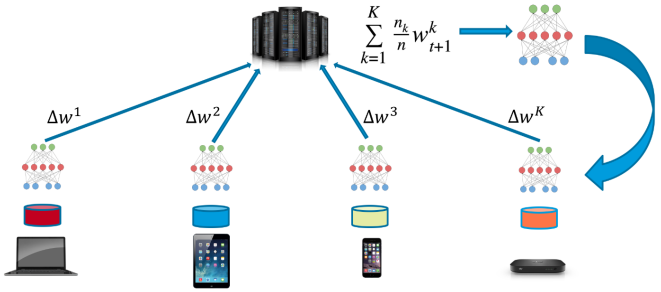


Figure: Federated Learning

Distributed optimization problem

- Distributed optimization problem:

$$\min_{x \in \mathbb{R}^d} \left[f(x) := \frac{1}{M} \sum_{m=1}^{M} f_m(x) = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{N} \sum_{i=1}^{N} l(g(x, a_i^m), b_i^m) \right],$$

  where x – optimization variables, (a_i^m, b_i^m) – data sample , l – loss function, g – model.
- The problem is divided into M pieces, each node m having access only to the function f_m (and its derivatives).

Communicating through the server

- Let us look at an example of how an ordinary GD becomes distributed.

Algorithm 1 Centralized GD

Input: Step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  Send x_k to all nodes ▷ by server
  for $m = 1, \ldots, M$ in parallel **do**
    Receive x_k forom server ▷ by nodes
    Compute gradient $\nabla f_m(x_k)$ at x_k ▷ by nodes
    Send $\nabla f_m(x_k)$ to server ▷ by nodes
  end for
  Receive $\nabla f_m(x_k)$ from nodes ▷ by server
  Compute $\nabla f(x_k) = \frac{1}{M} \sum_{m=1}^{M} \nabla f_m(x_k)$ ▷ by server
  $x_{k+1} = x_k - \gamma \nabla f(x_k)$ ▷ by server
end for

Output: x_K

What are we fighting for?

- **Q**: If we are talking about a federated learning paradigm, what would users want to demand from the learning process?

What are we fighting for?

- **Q:** If we are talking about a federated learning paradigm, what would users want to demand from the learning process?
- Since the data is private and personal, one would like to get its **privacy** and security.

What are we fighting for?

- **Q:** If we are talking about a federated learning paradigm, what would users want to demand from the learning process?
- Since the data is private and personal, one would like to get its **privacy** and security.
- The user wants a better result in terms of **personalization** of learning.

What are we fighting for?

- **Q:** If we are talking about a federated learning paradigm, what would users want to demand from the learning process?
- Since the data is private and personal, one would like to get its **privacy** and security.
- The user wants a better result in terms of **personalization** of learning.
- **Q:** distributivity is necessary for parallelization, but why can't we achieve full parallelization?

What are we fighting for?

- **Q:** If we are talking about a federated learning paradigm, what would users want to demand from the learning process?
- Since the data is private and personal, one would like to get its **privacy** and security.
- The user wants a better result in terms of **personalization** of learning.
- **Q:** distributivity is necessary for parallelization, but why can't we achieve full parallelization?
- Communication costs are a waste of time. The problem of communication bottleneck is relevant for all distributed problems. There are many ways to fight for **fast and effective** communication.

Local updates

The idea – more local computation

- In the basic approach, communications occur every iteration.
- If computing (stochastic) gradients is much cheaper, why not count multiple times between communications.

Parallel SGD/FedAvg/Local SGD

The idea:

- Make local steps (local training):

$$x_{k+1}^m = x_k^m - \gamma \nabla f_m(x_k^m).$$

- Every Tth iteration, forward the current x_k^m to the server. The server averages $x_k = \frac{1}{M} \sum_{m=1}^{M} x_k^m$, and forwards x_k to the workers. The workers update: $x_k^m = x_k$.
- Centralized distributed SGD is a Local SGD with $T = 1$.

Mangasarian O. Parallel Gradient Distribution in Unconstrained Optimization

McMahan B. et al. Communication-Efficient Learning of Deep Networks from Decentralized Data

How it works

- Problems: 1) LSTM on 10 million public posts, 2) CNN on CIFAR-10.

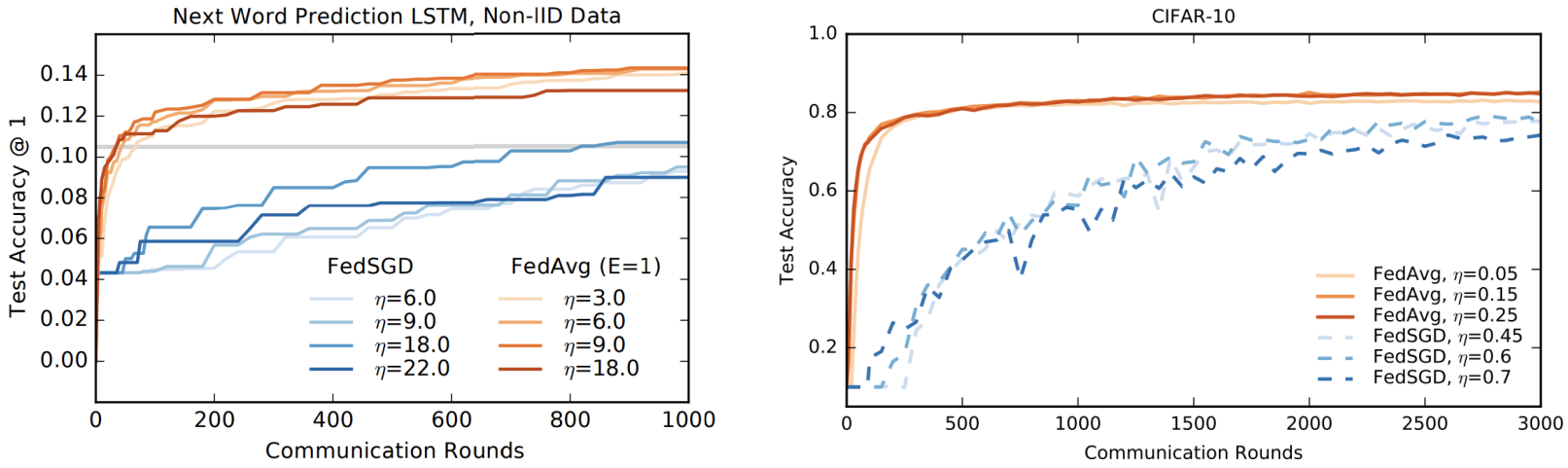


Figure: Comparison of Local SGD (FedAvg) and Centralized Distributed SGD (FedSGD).

McMahan B. et al. Communication-Efficient Learning of Deep Networks from Decentralized Data

How it convergences

- Problem: logistic regression on a5a LibSVM dataset.

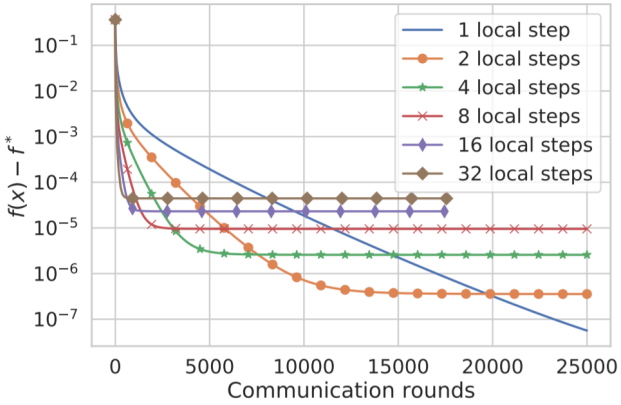


Figure: Dependence of convergence of Local SGD on number of local steps

- Typical convergence of this type of methods: faster in terms of communications, worse quality of ultimate accuracy.

Khaled A. et al. Tighter Theory for Local SGD on Identical and Heterogeneous Data

How it convergences

- **Q**: what causes this effect?

How it convergences

- **Q:** what causes this effect? It occurs due to the heterogeneity of local data on different devices.
- In the theoretical estimates of the convergence of the method, this also shows up:

$$\mathcal{O}\left(\frac{\|x^0 - x^*\|^2}{\gamma K} + \gamma \cdot \frac{1}{M}\sum_{m=1}^{M}\|\nabla f_m(x^*)\|^2\right),$$

where $\gamma \leq \mathcal{O}\left(\frac{1}{LT}\right)$ – stepsize, K – number of local iterations on each device. The estimation is given for the case of convex and L-smooth f_m.

Khaled A. et al. Tighter Theory for Local SGD on Identical and Heterogeneous Data

How it convergences

- **Q:** what causes this effect? It occurs due to the heterogeneity of local data on different devices.
- In the theoretical estimates of the convergence of the method, this also shows up:

$$\mathcal{O}\left(\frac{\|x^0 - x^*\|^2}{\gamma K} + \gamma \cdot \frac{1}{M}\sum_{m=1}^{M}\|\nabla f_m(x^*)\|^2\right),$$

where $\gamma \leq \mathcal{O}\left(\frac{1}{LT}\right)$ – stepsize, K – number of local iterations on each device. The estimation is given for the case of convex and L-smooth f_m.

Khaled A. et al. Tighter Theory for Local SGD on Identical and Heterogeneous Data

- Moreover, the σ_{opt}^2 factor is not eliminated at all.

Glasgow M.R. et al. Sharp bounds for federated averaging (local sgd) and continuous perspective

Solving the problem

- **Q**: the problem of the local method is convergence to a neighbourhood. How can it be interpreted and then solved?

Solving the problem

- **Q:** the problem of the local method is convergence to a neighbourhood. How can it be interpreted and then solved?
- Local task regularization as a defence against local overfitting (FedProx):

$$\tilde{f}_m(x) := f_m(x) + \frac{\lambda}{2}\|x - v\|^2,$$

  where v – certain reference point.
- Run local iterations not for f_m, but for $\tilde{f}_m$.

Li T. et al. Federated Optimization in Heterogeneous Networks

More modifications and generalizations

- Using so-called shifts to control bias due to heterogeneity.

  Karimireddy S. P. et al. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning

  Mishchenko K. et al. ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

- Using of consensus gossip procedures when centralized communications are unavailable

  Beznosikov A. et al. Decentralized Local Stochastic Extra-Gradient for Variational Inequalities

How it works

- Problem: logistic regression on EMNIST (letters).

	Epochs	0% similarity (sorted) Num. of rounds	Speedup	10% similarity Num. of rounds	Speedup	100% similarity (i.i.d.) Num. of rounds	Speedup
SGD	1	317	(1×)	365	(1×)	416	(1×)
SCAFFOLD	1	**77**	(4.1×)	62	(5.9×)	60	(6.9×)
	5	152	(2.1×)	20	(18.2×)	10	(41.6×)
	10	286	(1.1×)	16	(22.8×)	7	(59.4×)
	20	266	(1.2×)	**11**	(33.2×)	**4**	(104×)
FedAvg	1	258	(1.2×)	74	(4.9×)	83	(5×)
	5	428	(0.7×)	34	(10.7×)	10	(41.6×)
	10	711	(0.4×)	25	(14.6×)	6	(69.3×)
	20	1k+	(< 0.3×)	18	(20.3×)	**4**	(104×)
FedProx	1	1k+	(< 0.3×)	979	(0.4×)	459	(0.9×)
	5	1k+	(< 0.3×)	794	(0.5×)	351	(1.2×)
	10	1k+	(< 0.3×)	894	(0.4×)	308	(1.4×)
	20	1k+	(< 0.3×)	916	(0.4×)	351	(1.2×)

Figure: Comparison of Local SGD (FedAvg), FedProx and SCAFFOLD and Centralized Distributed SGD.

Karimireddy S. P. et al. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning

What do we want to achieve, anyway?

- Lower bounds:

$$K = \Omega\left(\sqrt{\frac{L}{\varepsilon}}\right).$$

  L – smoothness constant of f.

- What method will give such estimates?

What do we want to achieve, anyway?

- Lower bounds:

$$K = \Omega\left(\sqrt{\frac{L}{\varepsilon}}\right).$$

  L – smoothness constant of f.
- What method will give such estimates? Distributed version of Nesterov's accelerated method with 1 local step between communications.
- Note that local methods were invented for stochastic setups. But even here there is no improvement in the general case:

  Woodworth B. The Min-Max Complexity of Distributed Stochastic Convex Optimization with Intermittent Communication
- But there are settings where localised methods shoot out.

Data similarity

- Distributed learning problem:

$$f(x) = \frac{1}{M}\sum_{m=1}^{M} f_m(x) = \frac{1}{M}\sum_{m=1}^{M}\left[\frac{1}{N}\sum_{i=1}^{N}\ell(x, z_i^m)\right],$$

where z_i^m – data sample $(a_i^m,\ b_i^m)$, ℓ – loss of model with weights x on sample z_i^m.

Data similarity

- Distributed learning problem:

$$f(x) = \frac{1}{M}\sum_{m=1}^{M} f_m(x) = \frac{1}{M}\sum_{m=1}^{M}\left[\frac{1}{N}\sum_{i=1}^{N}\ell(x, z_i^m)\right],$$

  where z_i^m – data sample (a_i^m, b_i^m), ℓ – loss of model with weights x on sample z_i^m.
- Suppose we can partition the training data uniformly across devices. E.g., if cluster or collaborative computing on open data is used. In fact, we will further understand that it's enough to put a large uniform sample on just one device.
- This gives the similarity of the local loss functions.
- It is asserted that for any x:

$$\|\nabla^2 f_m(x) - \nabla^2 f(x)\| \leq \delta.$$

Matrix Hoeffding

Theorem (Matrix Hoeffding)

Consider a finite sequence of random square matrices $\{X_i\}_{i=1}^N$. Let the matrices in this sequence be independent, Hermitian and of dimension d. Suppose also that $\mathbb{E}[X_i] = 0$, and $X_i^2 \preceq A^2$ is almost surely, where A is a non-random Hermitian matrix. Then with probability $1 - p$ it is satisfied that

$$\left\| \sum_{i=1}^N X_i \right\| \leq \sqrt{8N\|A^2\| \cdot \ln(d/p)}.$$

Tropp J. An introduction to matrix concentration inequalities

Tropp J. User-friendly tail bounds for sums of random matrices

Similarity parameter

- Local loss function:

$$f_m(x) = \frac{1}{N}\sum_{i=1}^{N} \ell(x, z_i).$$

- ℓ – L-smooth (L-Lipschitz gradient), convex, twice differentiable function (e.g., quadratic or logreg). Then we have $\nabla^2 \ell(x, z_i) \preceq LI$ for any x and z_i (here I is a unit matrix).

Similarity parameter

- Local loss function:

$$f_m(x) = \frac{1}{N}\sum_{i=1}^{N} \ell(x, z_i).$$

- ℓ – L-smooth (L-Lipschitz gradient), convex, twice differentiable function (e.g., quadratic or logreg). Then we have $\nabla^2 \ell(x, z_i) \preceq LI$ for any x and z_i (here I is a unit matrix).
- Let us divide all data uniformly over all workers. $X_i = \frac{1}{N}\left[\nabla^2 \ell(x, z_i) - \nabla^2 f(x)\right]$. It is easy to check that all conditions of Hoeffding inequality are satisfied for it, in particular, $A^2 = \frac{4L^2}{N^2} I$.

Similarity parameter: summary

- As a result, we have

$$\|\nabla^2 f_m(x) - \nabla^2 f(x)\| \leq \delta \sim \frac{L}{\sqrt{N}}.$$

- Conclusion: the larger the local sample size, the smaller the similarity parameter (hessians are similar to each other).

Method in general terms (not only for similarity)

- Consider Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where $V(x, y)$ is the Bregman divergence generated by the strictly convex function $\varphi(x)$:

$$V(x, y) = \varphi(x) - \varphi(y) - \langle \nabla \varphi(y); x - y \rangle.$$

Method in general terms (not only for similarity)

- Consider Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where $V(x, y)$ is the Bregman divergence generated by the strictly convex function $\varphi(x)$:

$$V(x, y) = \varphi(x) - \varphi(y) - \langle \nabla \varphi(y); x - y \rangle.$$

- **Q**: Which method do we have if $\varphi(x) = \frac{1}{2}\|x\|^2$?

Method in general terms (not only for similarity)

- Consider Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where $V(x, y)$ is the Bregman divergence generated by the strictly convex function $\varphi(x)$:

$$V(x, y) = \varphi(x) - \varphi(y) - \langle \nabla \varphi(y); x - y \rangle.$$

- **Q**: Which method do we have if $\varphi(x) = \frac{1}{2}\|x\|^2$? Gradient descent.

Convergence in general terms

Definition (Relative smoothness and strong convexity)

Let $\varphi : \mathbb{R}^d \to \mathbb{R}$ is convex and twice differentiable. Let us say that the function f is L_φ-smooth and μ_φ-strongly convex with respect to φ if for any $x \in \mathbb{R}^d$ the following holds

$$\mu_\varphi \nabla^2 \varphi(x) \preceq \nabla^2 f(x) \preceq L_\varphi \nabla^2 \varphi(x),$$

or equivalently for any $x, y \in \mathbb{R}^d$

$$\mu_\varphi V(x, y) \leq f(x) - f(y) - \langle \nabla f(y); x - y \rangle \leq L_\varphi V(x, y).$$

Lu H. et al. Relatively-Smooth Convex Optimization by First-Order Methods, and Applications

Convergence in general terms: proof

- The optimality condition for the Mirror Descent step:

$$\gamma \nabla f(x_k) + \nabla \varphi(x_{k+1}) - \nabla \varphi(x_k) = 0.$$

Convergence in general terms: proof

- The optimality condition for the Mirror Descent step:

$$\gamma \nabla f(x_k) + \nabla \varphi(x_{k+1}) - \nabla \varphi(x_k) = 0.$$

- From it (here x^* – optimal point):

$$\langle \gamma \nabla f(x_k) + \nabla \varphi(x_{k+1}) - \nabla \varphi(x_k), x_{k+1} - x^* \rangle = 0.$$

$$\begin{aligned} \langle \gamma \nabla f(x_k), x^{k+1} - x^* \rangle &= \langle \nabla \varphi(x_k) - \nabla \varphi(x_{k+1}), x^{k+1} - x^* \rangle \\ &= V(x^*, x_k) - V(x^*, x_{k+1}) - V(x_{k+1}, x_k). \end{aligned}$$

(the last statement is called the Pythagoras' theorem for the Bregman divergence and is verified by the definition)

Convergence in general terms: proof

- The optimality condition for the Mirror Descent step:

$$\gamma \nabla f(x_k) + \nabla \varphi(x_{k+1}) - \nabla \varphi(x_k) = 0.$$

- From it (here x^* – optimal point):

$$\langle \gamma \nabla f(x_k) + \nabla \varphi(x_{k+1}) - \nabla \varphi(x_k), x_{k+1} - x^* \rangle = 0.$$

$$\begin{aligned} \langle \gamma \nabla f(x_k), x^{k+1} - x^* \rangle &= \langle \nabla \varphi(x_k) - \nabla \varphi(x_{k+1}), x^{k+1} - x^* \rangle \\ &= V(x^*, x_k) - V(x^*, x_{k+1}) - V(x_{k+1}, x_k). \end{aligned}$$

(the last statement is called the Pythagoras' theorem for the Bregman divergence and is verified by the definition)

- Small permutations give:

$$\begin{aligned} &\langle \gamma \nabla f(x_k), x_{k+1} - x_k \rangle + V(x_{k+1}, x_k) \\ &\qquad = V(x^*, x_k) - V(x^*, x_{k+1}) - \langle \gamma \nabla f(x_k), x_k - x^* \rangle. \end{aligned}$$

Convergence in general terms: proof

- Substitute $\gamma = \frac{1}{L_\varphi}$:

$$\begin{aligned}\langle \nabla f(x_k), x_{k+1} - x_k \rangle + L_\varphi V(x_{k+1}, x_k) \\ = L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) \\ - \langle \nabla f(x_k), x_k - x^* \rangle.\end{aligned}$$

Convergence in general terms: proof

- Substitute $\gamma = \frac{1}{L_\varphi}$:

$$\begin{aligned}\langle \nabla f(x_k), x_{k+1} - x_k \rangle + L_\varphi V(x_{k+1}, x_k) \\ = & L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) \\ & - \langle \nabla f(x_k), x_k - x^* \rangle.\end{aligned}$$

- Let us use the definition of smoothness with respect to φ c $x = x_{k+1}$, $y = x_k$:

$$f(x_{k+1}) - f(x_k) \leq \langle \nabla f(x_k); x_{k+1} - x_k \rangle + L_\varphi V(x_{k+1}, x_k).$$

Convergence in general terms: proof

- Substitute $\gamma = \frac{1}{L_\varphi}$:

$$\begin{aligned}\langle \nabla f(x_k), x_{k+1} - x_k \rangle + L_\varphi V(x_{k+1}, x_k) \\ = L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) \\ - \langle \nabla f(x_k), x_k - x^* \rangle.\end{aligned}$$

- Let us use the definition of smoothness with respect to φ c $x = x_{k+1}$, $y = x_k$:

$$f(x_{k+1}) - f(x_k) \leq \langle \nabla f(x_k); x_{k+1} - x_k \rangle + L_\varphi V(x_{k+1}, x_k).$$

- Combine the previous two:

$$f(x_{k+1}) - f(x_k) \leq L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) - \langle \nabla f(x_k), x_k - x^* \rangle.$$

Convergence in general terms: proof

- From the previous slide:

$$f(x_{k+1}) - f(x_k) \leq L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) - \langle \nabla f(x_k), x_k - x^* \rangle.$$

Convergence in general terms: proof

- From the previous slide:

$$f(x_{k+1}) - f(x_k) \leq L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) - \langle \nabla f(x_k), x_k - x^* \rangle.$$

- Relative strong convexity:

$$\mu_\varphi V(x^*, x_k) \leq f(x^*) - f(x_k) - \langle \nabla f(x_k); x^* - x_k \rangle$$

Convergence in general terms: proof

- From the previous slide:

$$f(x_{k+1}) - f(x_k) \leq L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) - \langle \nabla f(x_k), x_k - x^* \rangle.$$

- Relative strong convexity:

$$\mu_\varphi V(x^*, x_k) \leq f(x^*) - f(x_k) - \langle \nabla f(x_k); x^* - x_k \rangle$$

- Sum the previous two together and shuffle them a bit:

$$f(x_{k+1}) - f(x^*) \leq (L_\varphi - \mu_\varphi) V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}).$$

Convergence in general terms: proof

- From the previous slide:

$$f(x_{k+1}) - f(x_k) \leq L_\varphi V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}) - \langle \nabla f(x_k), x_k - x^* \rangle.$$

- Relative strong convexity:

$$\mu_\varphi V(x^*, x_k) \leq f(x^*) - f(x_k) - \langle \nabla f(x_k); x^* - x_k \rangle$$

- Sum the previous two together and shuffle them a bit:

$$f(x_{k+1}) - f(x^*) \leq (L_\varphi - \mu_\varphi) V(x^*, x_k) - L_\varphi V(x^*, x_{k+1}).$$

- By virtue of the fact that x^* – optimum:

$$V(x^*, x_{k+1}) \leq \left(1 - \frac{\mu_\varphi}{L_\varphi}\right) V(x^*, x_k).$$

Convergence in general terms

Theorem (Convergence of Mirror Descent)

Let φ and f satisfy the definition above, then Mirror Descent with step $\gamma = \frac{1}{L_\varphi}$ converges and is satisfied:

$$V(x^*, x_K) \leq \left(1 - \frac{\mu_\varphi}{L_\varphi}\right)^K V(x^*, x_0).$$

Method for the data similarity problem

- Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where the Bregman divergence of $V(x, y)$ generated by the function $\varphi(x)$ (here we need to require that f_1 is convex):

$$\varphi(x) = f_1(x) + \frac{\delta}{2} \|x\|^2.$$

The function f_1 is stored on the server.

Method for the data similarity problem

- Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where the Bregman divergence of $V(x, y)$ generated by the function $\varphi(x)$ (here we need to require that f_1 is convex):

$$\varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2.$$

The function f_1 is stored on the server.

- **Q:** What is the number of communications that occur in K iterations of Mirror Descent?

Method for the data similarity problem

- Mirror Descent:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

where the Bregman divergence of $V(x, y)$ generated by the function $\varphi(x)$ (here we need to require that f_1 is convex):

$$\varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2.$$

The function f_1 is stored on the server.

- **Q:** What is the number of communications that occur in K iterations of Mirror Descent? K of communications (number of ∇f gradient counts), computing arg min requires only computations on the server.

Method for the data similarity problem

Algorithm 2 Mirror Descent for the data similarity problem

Input: Stepsize $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  Send x_k to all workers ▷ server
  for $m = 1, \ldots, M$ in parallel **do**
    Receive x_k from server ▷ workers
    Compute gradient $\nabla f_m(x_k)$ at x_k ▷ workers
    Send $\nabla f_m(x_k)$ to server ▷ workers
  end for
  Receive $\nabla f_m(x_k)$ from all workers ▷ server
  Compute $\nabla f(x_k) = \frac{1}{M} \sum_{m=1}^{M} \nabla f_m(x_k)$ ▷ server
  $x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma \langle \nabla f(x_k), x \rangle + V(x, x_k)\right)$ ▷ server
end for

Convergence for the data similarity problem: proof

- Recall that convergence is defined in terms of constants from the relation:

$$\mu_\varphi \nabla^2 \varphi(x) \preceq \nabla^2 f(x) \preceq L_\varphi \nabla^2 \varphi(x),$$

Convergence for the data similarity problem: proof

- Recall that convergence is defined in terms of constants from the relation:

$$\mu_\varphi \nabla^2 \varphi(x) \preceq \nabla^2 f(x) \preceq L_\varphi \nabla^2 \varphi(x),$$

- In our case:

$$\mu_\varphi \left(\delta I + \nabla^2 f_1(x)\right) \preceq \nabla^2 f(x) \preceq L_\varphi \left(\delta I + \nabla^2 f_1(x)\right)$$

Convergence for the data similarity problem: proof

- Recall that convergence is defined in terms of constants from the relation:

$$\mu_\varphi \nabla^2 \varphi(x) \preceq \nabla^2 f(x) \preceq L_\varphi \nabla^2 \varphi(x),$$

- In our case:

$$\mu_\varphi \left(\delta I + \nabla^2 f_1(x)\right) \preceq \nabla^2 f(x) \preceq L_\varphi \left(\delta I + \nabla^2 f_1(x)\right)$$

- Let us find L_φ:

$$\|\nabla^2 f_1(x) - \nabla^2 f(x)\| \leq \delta \Rightarrow \nabla^2 f(x) - \nabla^2 f_1(x) \preceq \delta I$$
$$\Rightarrow \nabla^2 f(x) \preceq \delta I + \nabla^2 f_1(x) \Rightarrow \boxed{L_\varphi = 1.}$$

Convergence for the data similarity problem: proof

- Let us find μ_φ. From the strong convexity of f:

$$\mu I \preceq \nabla^2 f(x) \Rightarrow \delta I \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

Convergence for the data similarity problem: proof

- Let us find μ_φ. From the strong convexity of f:

$$\mu I \preceq \nabla^2 f(x) \Rightarrow \delta I \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

- From $\|\nabla^2 f_1(x) - \nabla^2 f(x)\| \leq \delta$ we have:

$$\nabla^2 f_1(x) - \nabla^2 f(x) \preceq \delta I.$$

Convergence for the data similarity problem: proof

- Let us find μ_φ. From the strong convexity of f:

$$\mu I \preceq \nabla^2 f(x) \Rightarrow \delta I \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

- From $\|\nabla^2 f_1(x) - \nabla^2 f(x)\| \leq \delta$ we have:

$$\nabla^2 f_1(x) - \nabla^2 f(x) \preceq \delta I.$$

- Combining the previous two points:

$$\nabla^2 f_1(x) - \nabla^2 f(x) \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

Convergence for the data similarity problem: proof

- Let us find μ_φ. From the strong convexity of f:

$$\mu I \preceq \nabla^2 f(x) \Rightarrow \delta I \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

- From $\|\nabla^2 f_1(x) - \nabla^2 f(x)\| \leq \delta$ we have:

$$\nabla^2 f_1(x) - \nabla^2 f(x) \preceq \delta I.$$

- Combining the previous two points:

$$\nabla^2 f_1(x) - \nabla^2 f(x) \preceq \frac{2\delta}{\mu}\nabla^2 f(x) - \delta I.$$

- And we get

$$\nabla^2 f_1(x) + \delta I \preceq \frac{2\delta + \mu}{\mu}\nabla^2 f(x) \Rightarrow \boxed{\mu_\varphi = \frac{\mu}{2\delta + \mu}.}$$

Convergence for the data similarity problem: theorem

Theorem (Convergence for the data similarity problem)

Let f be strongly convex, f_1 be convex, and ℓ be smooth, and $\varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2$, then Mirror Descent with step $\gamma = 1$ converges and is satisfied:

$$V(x^*, x_K) \leq \left(1 - \frac{\mu}{\mu + 2\delta}\right)^K V(x^*, x_0).$$

Convergence for the data similarity problem: theorem

Theorem (Convergence for the data similarity problem)

Let f be strongly convex, f_1 be convex, and ℓ be smooth, and $\varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2$, then Mirror Descent with step $\gamma = 1$ converges and is satisfied:

$$V(x^*, x_K) \leq \left(1 - \frac{\mu}{\mu + 2\delta}\right)^K V(x^*, x_0).$$

- This means that if we want to achieve an accuracy ε ($V(x^*, x_K) \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{V(x^*, x_0)}{\varepsilon}\right) \text{ communications.}$$

Hendrikx H. et al. Statistically Preconditioned Accelerated Gradient Method for Distributed Optimization

How it works

- Problem: ResNet-18 on CIFAR-10.

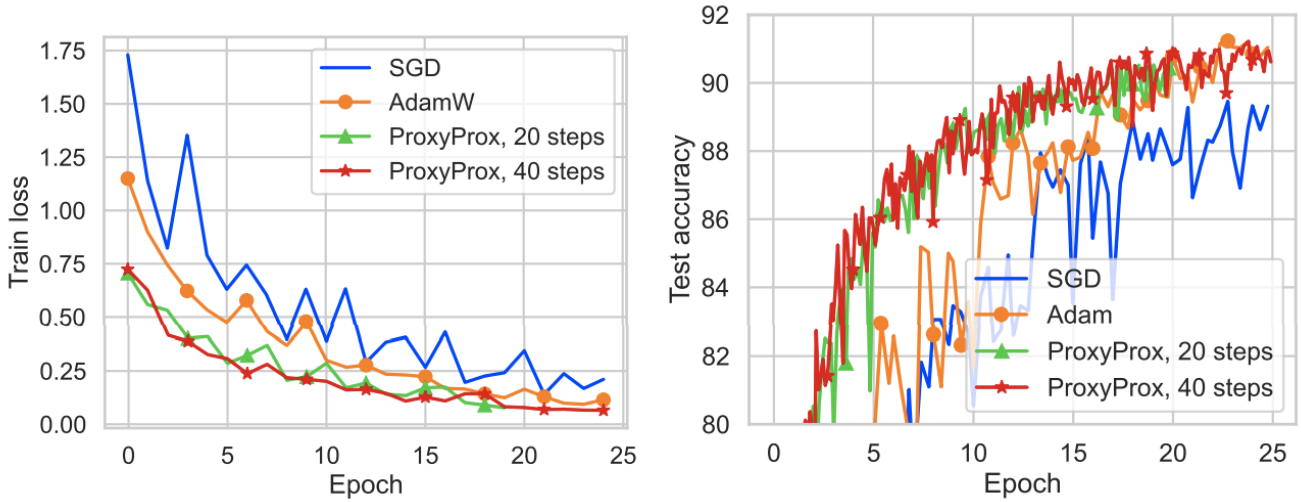


Figure: Comparison of Mirror Descent (ProxyProx) with SOTA optimizers on **non-distributed** problems.

Woodworth B. et al. Two Losses Are Better Than One: Faster Optimization Using a Cheaper Proxy

Research question: acceleration

- Estimate on the number of communications under data similarity:
$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{1}{\varepsilon}\right).$$
- Estimate on the number of communications for Centralized Distributed Gradient Descent:
$$K = \mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\varepsilon}\right).$$

Research question: acceleration

- Estimate on the number of communications under data similarity:
$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{1}{\varepsilon}\right).$$
- Estimate on the number of communications for Centralized Distributed Gradient Descent:
$$K = \mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\varepsilon}\right).$$
- Given that δ can be small, we see the improvement.

Research question: acceleration

- Estimate on the number of communications under data similarity:
$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{1}{\varepsilon}\right).$$
- Estimate on the number of communications for Centralized Distributed Gradient Descent:
$$K = \mathcal{O}\left(\frac{L}{\mu} \log \frac{1}{\varepsilon}\right).$$
- Given that δ can be small, we see the improvement.
- But there's also the distributed version of Accelerated Gradient Method that gives estimate:
$$K = \mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right).$$
- It is not clear which is better. Is it possible to accelerate the method for the problem with data similarity?

Another look at Mirror Descent

- Mirror Descent with $\gamma = 1$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

with the Bregman divergence $V(x, y)$, generated by the function $\varphi(x)$:

$$\varphi(x) = f_1(x) + \frac{\delta}{2} \|x\|^2.$$

Another look at Mirror Descent

- Mirror Descent with $\gamma = 1$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

with the Bregman divergence $V(x, y)$, generated by the function $\varphi(x)$:

- Substitute $\varphi(x)$: $\qquad \varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2.$

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(f_1(x) + \langle \nabla f(x_k) - \nabla f_1(x_k), x \rangle + \frac{\delta}{2}\|x - x_k\|^2 \right).$$

- Or a little differently:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(f_1(x) + \frac{\delta}{2} \left\| x - \left(x_k - \frac{1}{\delta}(\nabla f(x_k) - \nabla f_1(x_k)) \right) \right\|^2 \right).$$

Another look at Mirror Descent

- Mirror Descent with $\gamma = 1$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\langle \nabla f(x_k), x \rangle + V(x, x_k) \right),$$

with the Bregman divergence $V(x, y)$, generated by the function $\varphi(x)$:

- Substitute $\varphi(x)$: $\qquad \varphi(x) = f_1(x) + \frac{\delta}{2}\|x\|^2.$

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(f_1(x) + \langle \nabla f(x_k) - \nabla f_1(x_k), x \rangle + \frac{\delta}{2}\|x - x_k\|^2 \right).$$

- Or a little differently:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(f_1(x) + \frac{\delta}{2} \left\| x - \left(x_k - \frac{1}{\delta}(\nabla f(x_k) - \nabla f_1(x_k)) \right) \right\|^2 \right).$$

- **Q:** What method does it look like?

Another look at Mirror Descent

- Proximal Gradient Method for the composite problem $g_1(x) + g_2(x)$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma g_2(x) + \frac{1}{2} \|x - (x_k - \gamma g_1(x_k))\|^2 \right).$$

Another look at Mirror Descent

- Proximal Gradient Method for the composite problem $g_1(x) + g_2(x)$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma g_2(x) + \frac{1}{2} \|x - (x_k - \gamma g_1(x_k))\|^2 \right).$$

- The argmin problem at each iteration can be solved inexactly somehow by a numerical method (e.g., by Gradient Descent or Nesterov's method). The peculiarity of such a method is that ∇g_1 is called much less frequently than ∇g_2. This kind of algorithms for composite problems that devide oracle complexities are sometimes called slidings.

Another look at Mirror Descent

- Proximal Gradient Method for the composite problem $g_1(x) + g_2(x)$:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left(\gamma g_2(x) + \frac{1}{2} \|x - (x_k - \gamma g_1(x_k))\|^2 \right).$$

- The argmin problem at each iteration can be solved inexactly somehow by a numerical method (e.g., by Gradient Descent or Nesterov's method). The peculiarity of such a method is that ∇g_1 is called much less frequently than ∇g_2. This kind of algorithms for composite problems that devide oracle complexities are sometimes called slidings.
- Usually, this kind of algorithms are proposed for composite problems of the form: convex + convex.
- In our case, $g_1 = f - f_1$, $g_2 = f_1$. And this is the problem of the form: non-convex + convex = convex.

Acceleration and optimal algorithm

- Long history:

Reference	Communication complexity	Local gradient complexity	Order	Limitations
DANE [42]	$\mathcal{O}\left(\frac{\delta^2}{\mu^2}\log\frac{1}{\varepsilon}\right)$	— (2)	1st	quadratic
DiSCO [51]	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}(\log\frac{1}{\varepsilon}+C^2\Delta F_0)\log\frac{L}{\mu}\right)$	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}(\log\frac{1}{\varepsilon}+C^2\Delta F_0)\log\frac{L}{\mu}\right)$	2nd	C - self-concordant (3)
AIDE [40]	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}\log\frac{L}{\delta}\right)$	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\log\frac{L}{\delta}\right)$ (4)	1st	quadratic
DANE-LS [50]	$\mathcal{O}\left(\frac{\delta}{\mu}\log\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\frac{\delta^{3/2}}{\mu^{3/2}}\log\frac{1}{\varepsilon}\right)$ (5)	1st/2nd	quadratic (6)
DANE-HB [50]	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\frac{\delta}{\mu}\log\frac{1}{\varepsilon}\right)$ (5)	1st/2nd	quadratic (6)
SONATA [45]	$\mathcal{O}\left(\frac{\delta}{\mu}\log\frac{1}{\varepsilon}\right)$	— (2)	1st	decentralized
SPAG [21]	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right)$ (1)	— (2)	1st	M - Lipshitz hessian
DiRegINA [12]	$\mathcal{O}\left(\frac{\delta}{\mu}\log\frac{1}{\varepsilon}+\sqrt{\frac{M\delta R_0}{\mu}}\right)$	— (2)	2nd	M -Lipshitz hessian
ACN [1]	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}+\sqrt[3]{\frac{M\delta R_0}{\mu}}\right)$	— (2)	2nd	M -Lipshitz hessian
Acc SONATA [46]	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}\log\frac{\delta}{\mu}\right)$	— (2)	1st	decentralized
This paper	$\mathcal{O}\left(\sqrt{\frac{\delta}{\mu}}\log\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right)$	1st	

Optimal algorithm

For the problem:

$$f(x) = g_1(x) + g_2(x),$$

where $g_1 = f - f_1$ и $g_2 = f_1$.

Algorithm 3 Accelerated Extragradient

Input: Stepsizes γ and θ, momentums α, τ, starting point $x_0 = x_0^f \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
2: $\quad x_k^g = \tau x_k + (1-\tau) x_k^f$
3: $\quad x_{k+1}^f \approx \arg\min\limits_{x \in \mathbb{R}^d} \left[\langle \nabla g_1(x_k^g), x - x_k^g \rangle + \frac{1}{2\theta} \|x - x_k^g\|^2 + g_2(x) \right]$
4: $\quad x_{k+1} = x_k + \eta\alpha(x_{k+1}^f - x_k) - \eta \nabla f(x_{k+1}^f)$
5: **end for**

Kovalev D. et al. Optimal Gradient Sliding and its Application to Distributed Optimization Under Similarity

Three ideas

- 1 idea – Nesterov acceleration.
- 2 idea – Sliding.
- 3 idea – Extragradient/Tseng's method = method for VIs.
- first two ideas are clear, but the third idea is the key.

How it works

- Problem: logistic regression on different data.

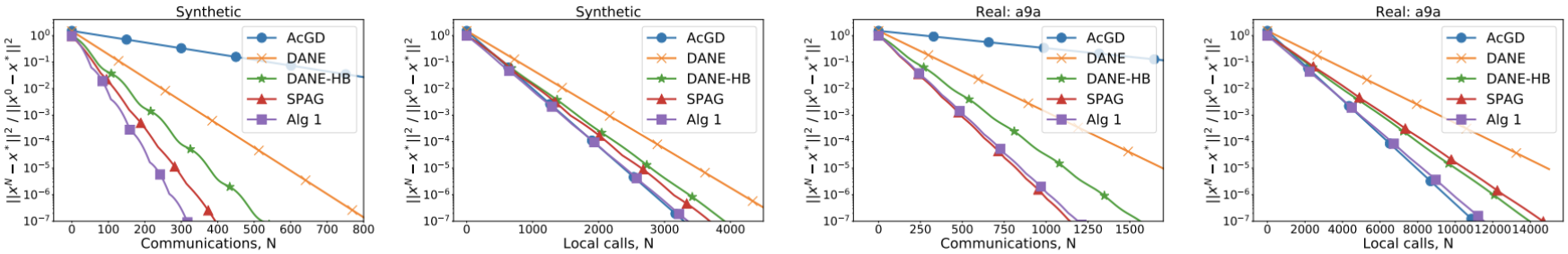


Figure: Comparison of Accelerated ExtraGradient (Alg. 1) with SOTA optimizers for distributed minimization problems under data similarity.

Compression

Unbiased compression (quantization)

Unbiased compression (quantization)

Let us call the stochastic operator $\mathcal{Q}(x)$ an unbiased compression (quantization) operator if for any $x \in \mathbb{R}^d$ it is fulfilled:

$$\mathbb{E}[\mathcal{Q}(x)] = x, \quad \mathbb{E}[\|\mathcal{Q}(x)\|_2^2] \leq \omega \|x\|_2^2,$$

where $\omega \geq 1$.

Unbiased compression: examples

Random sparsification (selection of random components)

Consider a stochastic operator

$$\text{Randk}(x) = \frac{d}{k} \sum_{i \in S} [x]_i e_i,$$

where k — some fixed number from the set $\{1, \ldots, d\}$ (the number of components of vector x that we pass; for example, we can choose $k = 1$), S —a random subset of the set $\{1, \ldots, d\}$ of size k (the subset S is chosen randomly and equally likely among all possible subsets of size d), $[\cdot]_i$ – i-th component of the vector, $(e_1, \ldots, e_d)$ —the standard basis in $\mathbb{R}^d$.

Richtárik P. and Takáč M. Parallel coordinate descent methods for big data optimization

Unbiased compression: examples

Random sparsification (selection of random components)

Consider a stochastic operator

$$\text{Randk}(x) = \frac{d}{k} \sum_{i \in S} [x]_i e_i,$$

where k — some fixed number from the set $\{1, \ldots, d\}$ (the number of components of vector x that we pass; for example, we can choose $k = 1$), S —a random subset of the set $\{1, \ldots, d\}$ of size k (the subset S is chosen randomly and equally likely among all possible subsets of size d), $[\cdot]_i$ – i-th component of the vector, $(e_1, \ldots, e_d)$ —the standard basis in $\mathbb{R}^d$.

Richtárik P. and Takáč M. Parallel coordinate descent methods for big data optimization

- **Q**: why do we need the multiplier $\frac{d}{k}$?

Unbiased compression: examples

Random sparsification (selection of random components)

Consider a stochastic operator

$$\text{Randk}(x) = \frac{d}{k} \sum_{i \in S} [x]_i e_i,$$

where k — some fixed number from the set $\{1, \ldots, d\}$ (the number of components of vector x that we pass; for example, we can choose $k = 1$), S —a random subset of the set $\{1, \ldots, d\}$ of size k (the subset S is chosen randomly and equally likely among all possible subsets of size d), $[\cdot]_i$ – i-th component of the vector, $(e_1, \ldots, e_d)$ —the standard basis in $\mathbb{R}^d$.

Richtárik P. and Takáč M. Parallel coordinate descent methods for big data optimization

- **Q**: why do we need the multiplier $\frac{d}{k}$? For unbiasedness.

Unbiased compression: examples

- **Q**: What is ω for random sparsification?

Unbiased compression: examples

- **Q**: What is ω for random sparsification? $\frac{d}{k}$.
  Each coordinate takes part in $\mathcal{Q}(x)$ with probability $\frac{k}{d}$, so

$$\begin{aligned}\mathbb{E}\left[\|\mathcal{Q}(x)\|^2\right] &= \mathbb{E}\left[\sum_{i=1}^{d}[\mathcal{Q}(x)]_i^2\right] \\ &= \frac{d^2}{k^2}\left[\sum_{i=1}^{d}\frac{k}{d}[x]_i^2\right] \\ &= \frac{d}{k}\|x\|^2.\end{aligned}$$

  Here $[\cdot]_i$ – ith coordinate of the vector.

Unbiased compression: examples

Three-level ℓ_2-quantization

Consider the following operator: $[\mathcal{Q}(x)]_i = \|x\|_2 \text{sign}(x_i)\xi_i, \ i = 1, \ldots, d$, where $[\cdot]_i$ is the i-th component of the vector, and ξ_i —a random variable having a Bernoulli distribution with parameter $\frac{|x_i|}{\|x\|_2}$, i.e.

$$\xi_i = \begin{cases} 1 & \text{with probability } \frac{|x_i|}{\|x\|_2}, \\ 0 & \text{with probability } 1 - \frac{|x_i|}{\|x\|_2}. \end{cases}$$

Thus, if we want to pass a vector $\mathcal{Q}(x)$, we need to pass a vector consisting of zeros and ± 1, and a real number $\|x\|_2$, with the probability of zeroing a component the greater the component is smaller modulo it. It can be shown that this operator is an unbiased compression with constant $\omega = \sqrt{d}$.

Alistarh D. et al. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding

Unbiased compression: examples

- **Q**: Is rounding be an unbiased operator?
- **Q**: Which rounding seems to be the most natural for calculations on a computer?

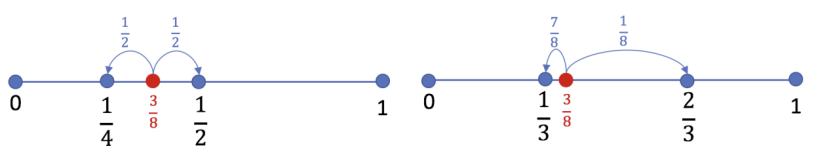
Unbiased compression: examples

Natural compression (random rounding to the power of two)

Consider the following operator:

$$[\mathcal{Q}(x)]_i = \begin{cases} \lfloor [x]_i \rfloor_2 \,, & \text{with probabilty } p = \frac{[x]_i - \lceil [x]_i \rceil_2}{\lceil [x]_i \rceil_2 - \lfloor [x]_i \rfloor_2} \\ \lceil [x]_i \rceil_2 \,, & \text{with probability } 1 - p \end{cases} ,$$

where $[\cdot]_i$ – i-vector component, $\lfloor \cdot \rfloor_2$ – is the nearest degree of two from the bottom, $\lceil \cdot \rceil_2$ is the nearest degree of two from the top. We round to the two nearest powers of two, the probability of rounding is greater the closer the real number is to the corresponding power of two. It can be shown that $\omega = \frac{9}{8}$.



Horváth S. et al. Natural compression for distributed deep learning

Unbiased compression: an idea

- The simplest idea that comes to mind is to use parallel GD, but apply unbiased compression to the gradients sent from the workers to the server.

Quantized GD (QGD)

Algorithm 1 QGD

Input: step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  Send x_k to all workers ▷ by server
  for $m = 1, \ldots, M$ in parallel **do**
    Recieve x_k from server ▷ by workers
    Compute $\nabla f_m(x_k)$ in point x_k ▷ by workers
    Independently generate $g_{k,m} = \mathcal{Q}(\nabla f_m(x_k))$ ▷ by workers
    Send $g_{k,m}$ to master ▷ by workers
  end for
  Recieve $g_{k,m}$ from all workers ▷ by server
  Compute $g_k = \frac{1}{M} \sum_{m=1}^{M} g_{k,m}$ ▷ by server
  $x_{k+1} = x_k - \gamma g_k$ ▷ by server
end for

Output: x_K

QGD: convergence

Theorem (QGD)

Let all local functions f_m be μ-simply convex and have L-Lipschitz gradient, then if $\gamma \leq L^{-1}\left(\frac{2\omega}{M}+1\right)^{-1}$ then

$$\mathbb{E}\left[\|x_K - x^*\|^2\right] = \mathcal{O}\left((1-\gamma\mu)^K\|x_0 - x^*\|^2 + \gamma \cdot \frac{2\omega}{\mu M^2}\sum_{m=1}^{M}\|\nabla f_m(x^*)\|^2\right).$$

The selection of γ from the paper was also used to obtain this result:

Stich S. Unified Optimal Analysis of the (Stochastic) Gradient Method

QGD: convergence

Theorem (QGD)

Let all local functions f_m be μ-simply convex and have L-Lipschitz gradient, then if $\gamma \leq L^{-1}\left(\frac{2\omega}{M}+1\right)^{-1}$ then

$$\mathbb{E}\left[\|x_K - x^*\|^2\right] = \mathcal{O}\left((1-\gamma\mu)^K\|x_0 - x^*\|^2 + \gamma \cdot \frac{2\omega}{\mu M^2}\sum_{m=1}^{M}\|\nabla f_m(x^*)\|^2\right).$$

The selection of γ from the paper was also used to obtain this result:

Stich S. Unified Optimal Analysis of the (Stochastic) Gradient Method

- **Q**: what are the problems with this estimation? (recall the convergence estimate GD)

QGD: convergence

Theorem (QGD)

Let all local functions f_m be μ-simply convex and have L-Lipschitz gradient, then if $\gamma \leq L^{-1}\left(\frac{2\omega}{M}+1\right)^{-1}$ then

$$\mathbb{E}\left[\|x_K - x^*\|^2\right] = \mathcal{O}\left((1-\gamma\mu)^K\|x_0 - x^*\|^2 + \gamma \cdot \frac{2\omega}{\mu M^2}\sum_{m=1}^{M}\|\nabla f_m(x^*)\|^2\right).$$

The selection of γ from the paper was also used to obtain this result:

Stich S. Unified Optimal Analysis of the (Stochastic) Gradient Method

- **Q**: what are the problems with this estimation? (recall the convergence estimate GD) Sublinear convergence (depends on the heterogeneity of the data).

QGD: convergence

- Behaviour in practice:

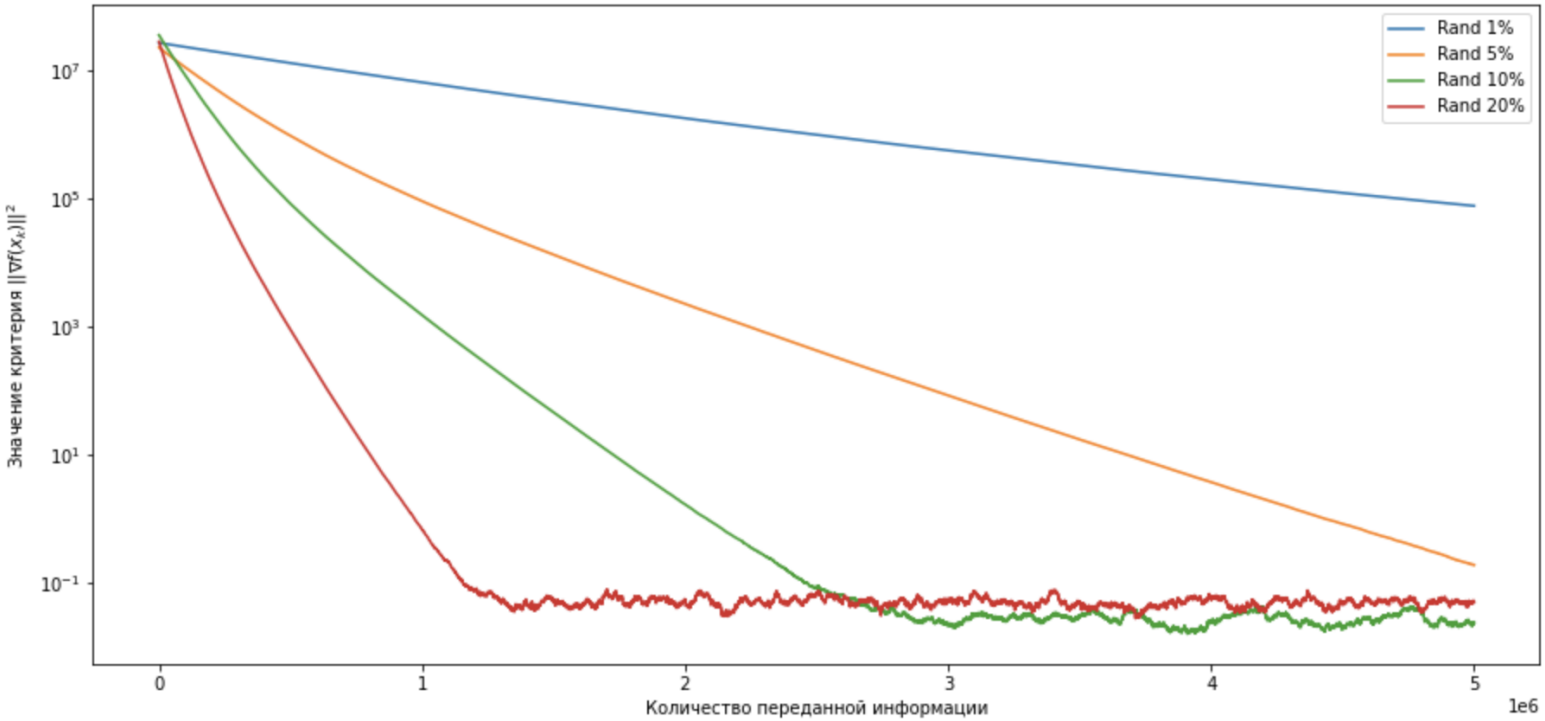


Figure: Behaviour of methods with unbiased compression operator and constant step size

- In theory the pitch was chosen cleverly, with constant step the theory predicts exactly the same effect – early plateauing.

Unbiased compression: solving the plateau problem

- Method DIANA – QGD with memory:

Algorithm 1 DIANA (sketch)

1: Each device m possesses a vector of "memory" $h_0^m = 0$

2: The server stores $h_0 = \frac{1}{M}\sum\limits_{m=1}^{M} h_0^m = 0$

3: Send a compressed version of the difference to the server $\mathcal{Q}(\nabla f_m(x_k) - h_k^m)$

4: Refreshing the memory $h_{k+1}^m = h_k^m + \alpha\mathcal{Q}(\nabla f_m(x_k) - h_k^m)$

5: The server calculates $g_k = h_k + \frac{1}{M}\sum\limits_{m=1}^{M} \mathcal{Q}(\nabla f_m(x_k) - h_k^m)$

6: For the update $x_{k+1} = x_k - \gamma g_k$

7: The server updates $h_{k+1} = h_k + \alpha\frac{1}{M}\sum\limits_{m=1}^{M} \mathcal{Q}(\nabla f_m(x_k) - h_k^m)$

Mishchenko K. et al. Distributed Learning with Compressed Gradient Differences

QGD and DIANA: convergence

- **Q**: What are some other questions about convergence/estimates of convergence?

QGD and DIANA: convergence

- **Q**: What are some other questions about convergence/estimates of convergence? Does it converge better at all?
- Best estimate per number of communications for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[1+\frac{\omega}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- Estimate on the number of communications for GD:

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

- In terms of **number** of communications, compressed methods are inferior to basic methods – this is to be expected (compression fees). BUT!

QGD and DIANA: convergence

- Compressors compress information β times and it is typical that $\beta \geq \omega$.

QGD and DIANA: convergence

- Compressors compress information β times and it is typical that $\beta \geq \omega$.
- Best estimate on the number of information for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[\frac{1}{\beta} + \frac{1}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- Estimate on the number of information for GD:

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

QGD and DIANA: convergence

- Compressors compress information β times and it is typical that $\beta \geq \omega$.
- Best estimate on the number of information for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[\frac{1}{\beta} + \frac{1}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- Estimate on the number of information for GD:

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

- The unbiased compressor provably improves the number of transmitted information, an improvement factor: $\left[\frac{1}{\beta} + \frac{1}{M}\right]$.

There may not be a server

- Often in practice "centralised communications via a server" are implemented without a "server".
- Architecture with AllGather/AllReduce procedure: some graph of links/communications is given, messages are exchanged according to this graph, including averaging can be organised.

Chan, E. et al. Collective communication: theory, practice, and experience

Centralised communications without a server

Operation	Before				After			
	Node 0	Node 1	Node 2	Node 3	Node 0	Node 1	Node 2	Node 3
Broadcast	x				x	x	x	x
Reduce(-to-one)	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$\sum_j x^{(j)}$			
Scatter	x_0 x_1 x_2 x_3				x_0	x_1	x_2	x_3
Gather	x_0	x_1	x_2	x_3	x_0 x_1 x_2 x_3			
Allgather	x_0	x_1	x_2	x_3	x_0 x_1 x_2 x_3	x_0 x_1 x_2 x_3	x_0 x_1 x_2 x_3	x_0 x_1 x_2 x_3
Reduce-scatter	$x_0^{(0)}$ $x_1^{(0)}$ $x_2^{(0)}$ $x_3^{(0)}$	$x_0^{(1)}$ $x_1^{(1)}$ $x_2^{(1)}$ $x_3^{(1)}$	$x_0^{(2)}$ $x_1^{(2)}$ $x_2^{(2)}$ $x_3^{(2)}$	$x_0^{(3)}$ $x_1^{(3)}$ $x_2^{(3)}$ $x_3^{(3)}$	$\sum_j x_0^{(j)}$	$\sum_j x_1^{(j)}$	$\sum_j x_2^{(j)}$	$\sum_j x_3^{(j)}$
Allreduce	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$\sum_j x^{(j)}$	$\sum_j x^{(j)}$	$\sum_j x^{(j)}$	$\sum_j x^{(j)}$

Ring AllReduce

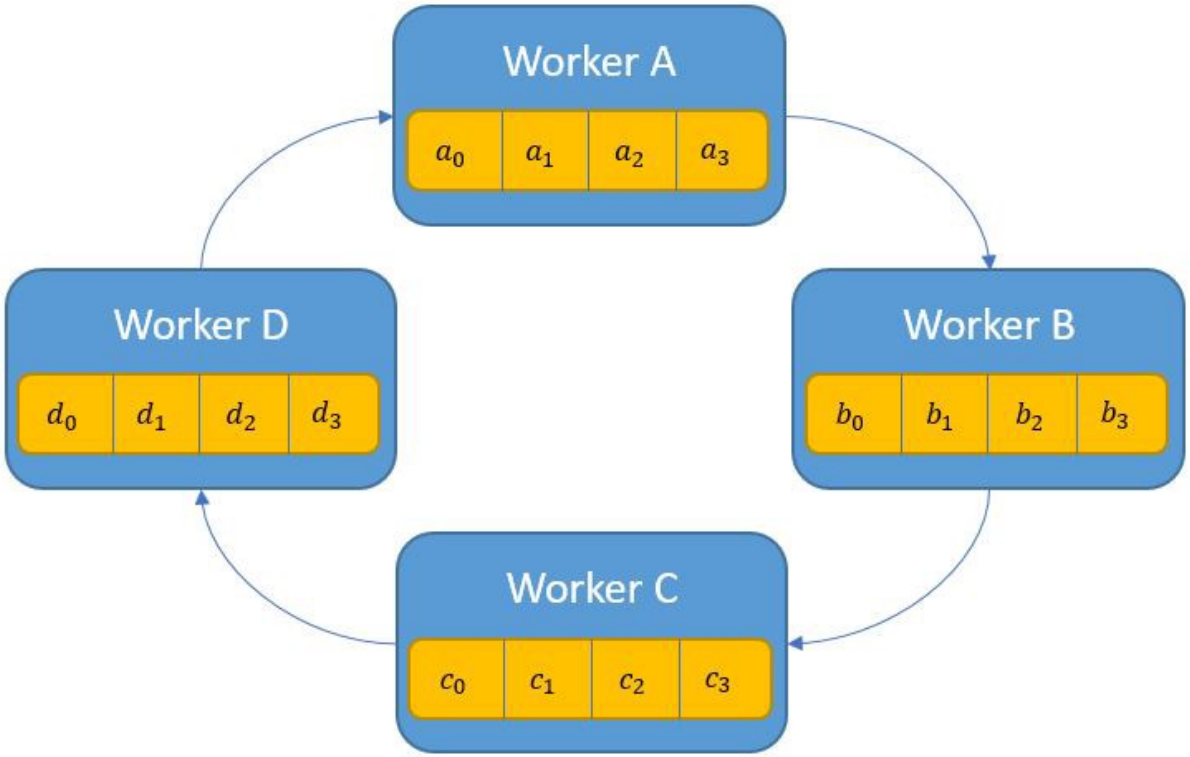


Figure: Picture from here

Ring AllReduce: first step of averaging

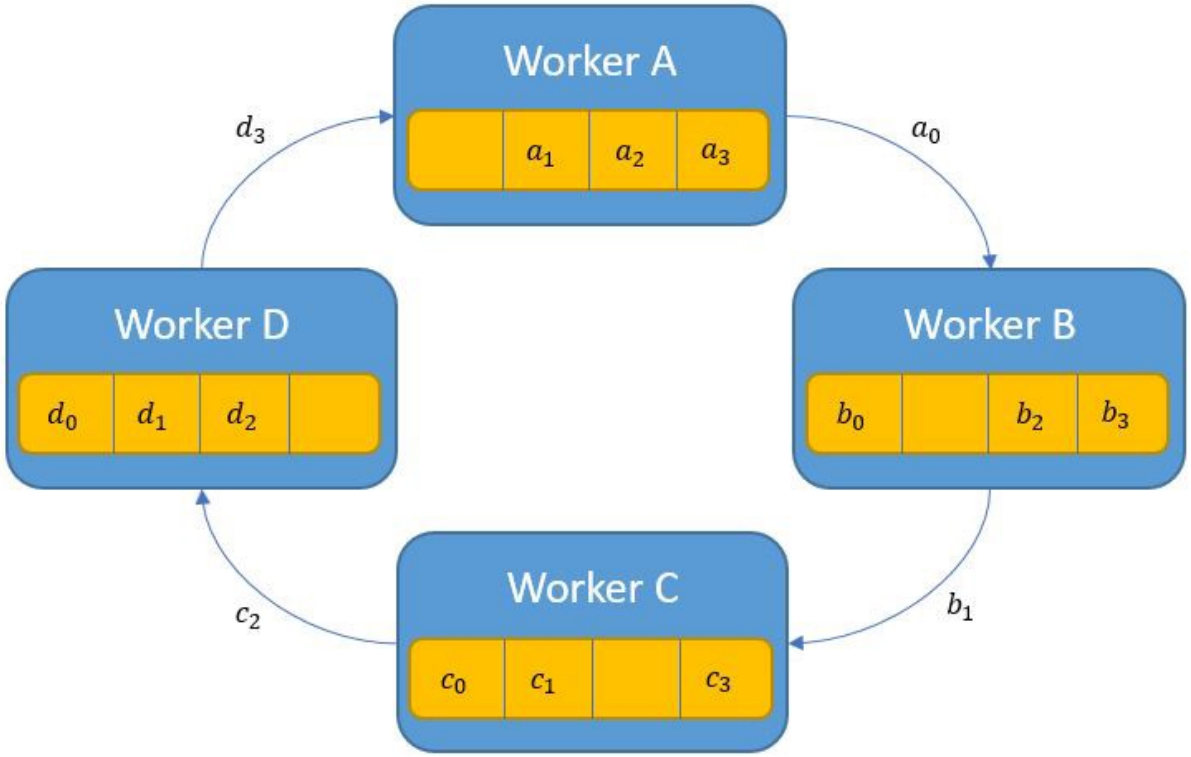


Figure: Picture from here

Ring AllReduce: second step of averaging

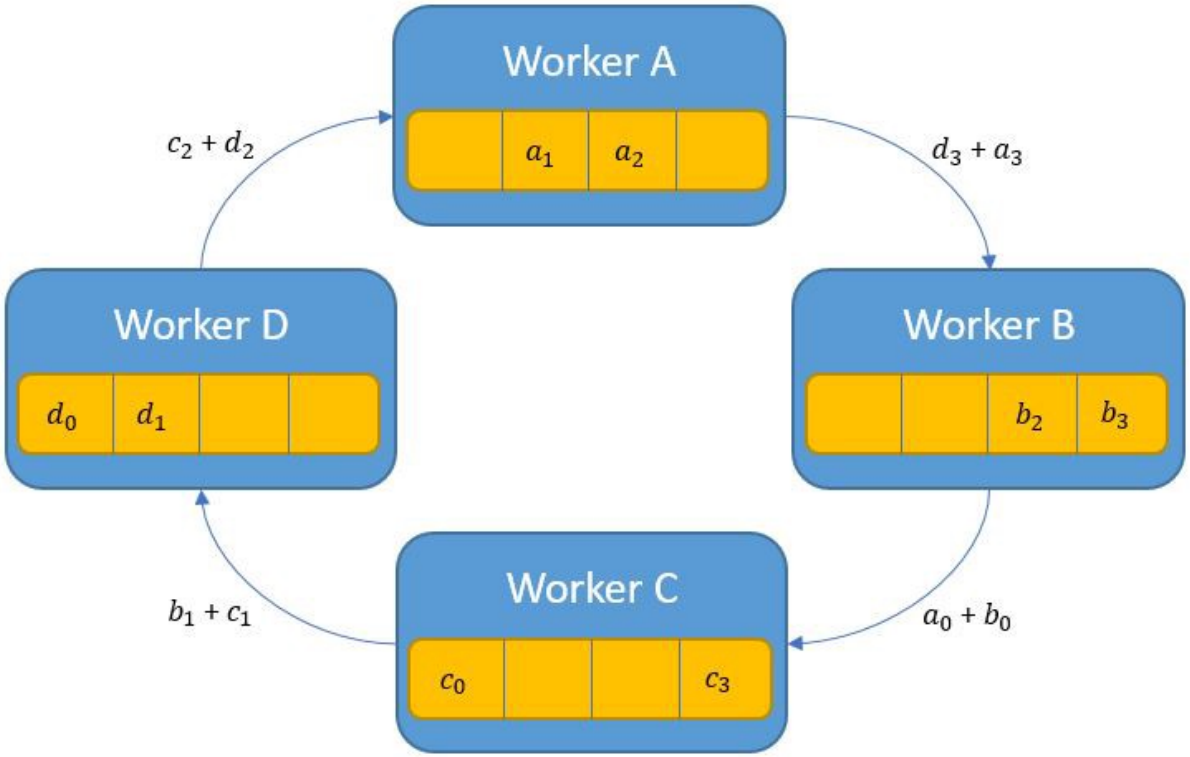


Figure: Picture from here

Ring AllReduce: first step of backprogation

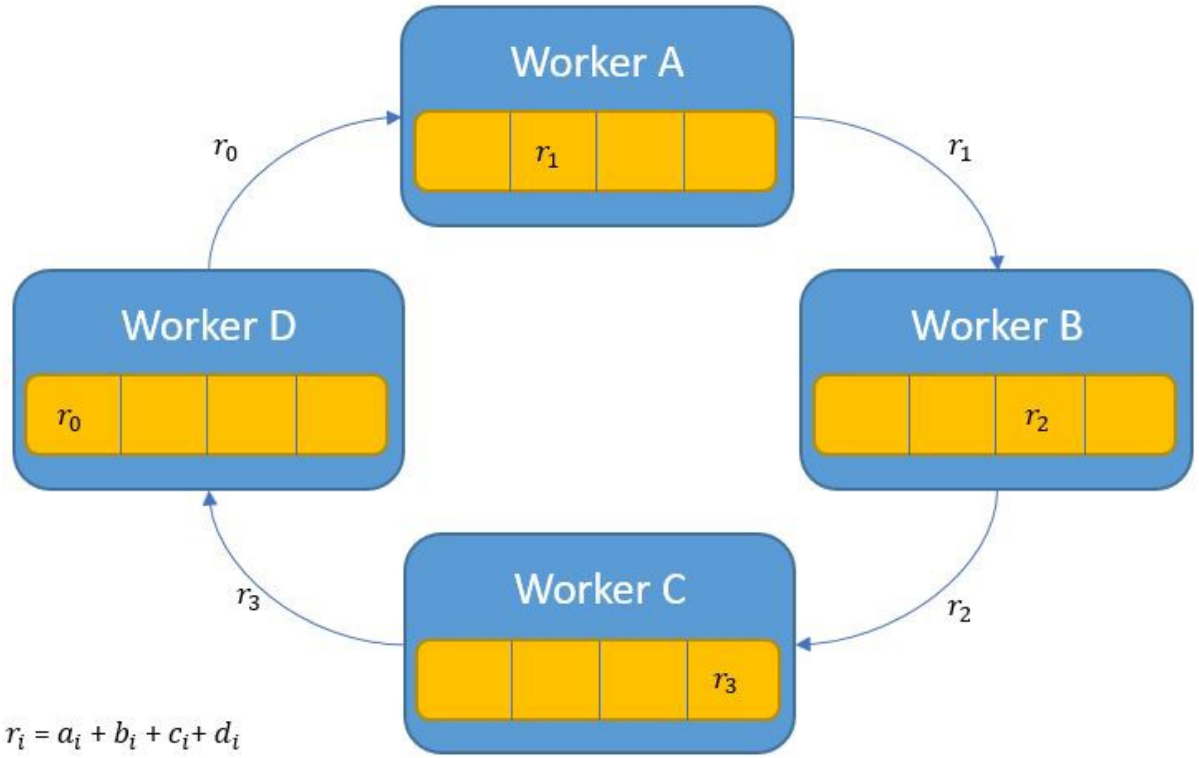


Figure: Картинка отсюда

Ring AllReduce: final

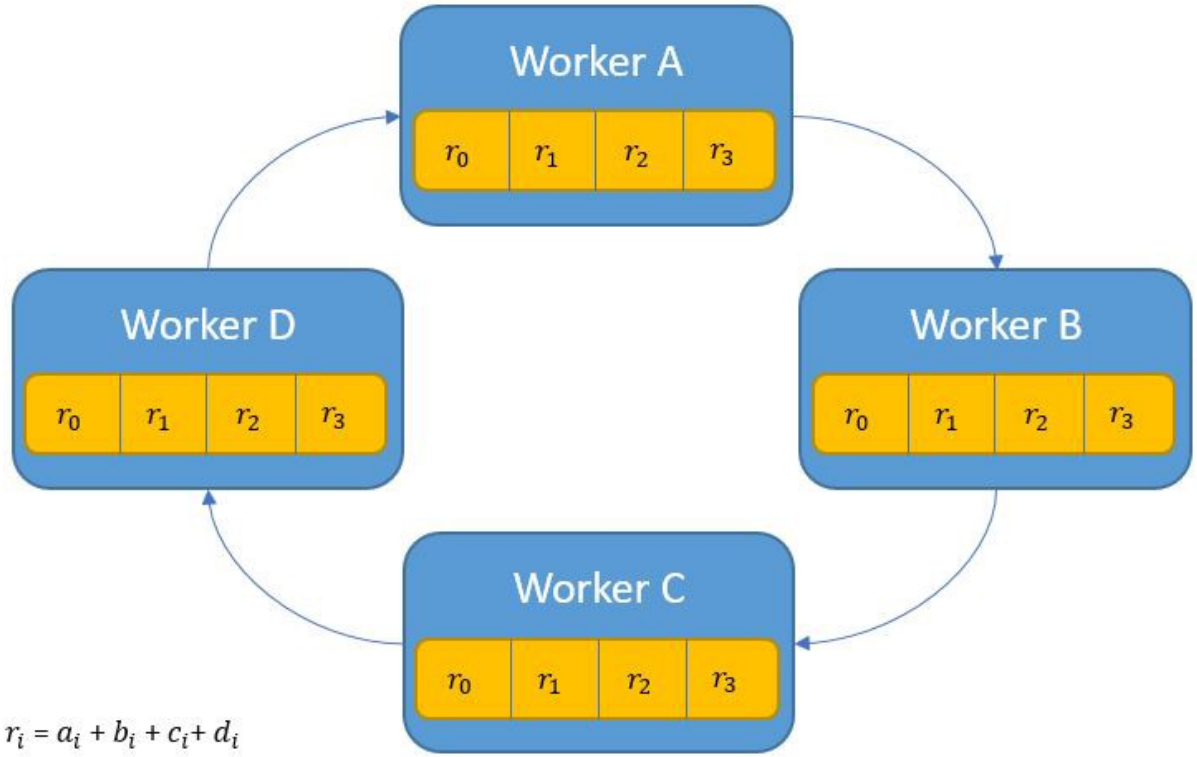


Figure: Картинка отсюда

Quantized GD (QGD) with AllReduce

Algorithm 1 QGD

Input: step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  for $m = 1, \ldots, M$ in parallel **do**
    Compute $\nabla f_m(x_k)$ in x_k
    Indipendetly generate $g_{k,m} = \mathcal{Q}(\nabla f_m(x_k))$
    Run AllReduce $\{g_{k,m}\}$ and get $g_k = \frac{1}{M}\sum_{m=1}^{M} g_{k,m}$
    $x_{k+1} = x_k - \gamma g_k$
  end for
end for

Output: x_K

Quantized GD (QGD) with AllReduce

Algorithm 1 QGD

Input: step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  for $m = 1, \ldots, M$ in parallel **do**
    Compute $\nabla f_m(x_k)$ in x_k
    Indipendetly generate $g_{k,m} = \mathcal{Q}(\nabla f_m(x_k))$
    Run AllReduce $\{g_{k,m}\}$ and get $g_k = \frac{1}{M}\sum_{m=1}^{M} g_{k,m}$
    $x_{k+1} = x_k - \gamma g_k$
  end for
end for

Output: x_K

- **Q**: What kind of problems might appear with (for example) Randk?

Quantized GD (QGD) with AllReduce

Algorithm 1 QGD

Input: step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  for $m = 1, \ldots, M$ in parallel **do**
    Compute $\nabla f_m(x_k)$ in x_k
    Indipendetly generate $g_{k,m} = \mathcal{Q}(\nabla f_m(x_k))$
    Run AllReduce $\{g_{k,m}\}$ and get $g_k = \frac{1}{M}\sum_{m=1}^{M} g_{k,m}$
    $x_{k+1} = x_k - \gamma g_k$
  end for
end for

Output: x_K

- **Q:** What kind of problems might appear with (for example) Randk? The same non-zero coordinates on different devices can cause collisions.

PermK: do a dependent randomisation

Permutation compressor (dependent RandK)

Suppose that $d \geq n$ and $d = qn$, where $q \geq 1$ is an integer. Let $\pi = (\pi_1, \ldots, \pi_d)$ be a random permutation of $\{1, \ldots, d\}$. Then for each $i \in \{1, 2, \ldots, n\}$ we have the following compression operator

$$\mathcal{Q}_i(u) = n \cdot \sum_{j=q(i-1)+1}^{qi} u_{\pi_j} e_{\pi_j}.$$

Szlendak, R. et al. Permutation Compressors for Provably Faster Distributed Nonconvex Optimization

PermK: do a dependent randomisation

Permutation compressor (dependent RandK)

Suppose that $d \geq n$ and $d = qn$, where $q \geq 1$ is an integer. Let $\pi = (\pi_1, \ldots, \pi_d)$ be a random permutation of $\{1, \ldots, d\}$. Then for each $i \in \{1, 2, \ldots, n\}$ we have the following compression operator

$$\mathcal{Q}_i(u) = n \cdot \sum_{j=q(i-1)+1}^{qi} u_{\pi_j} e_{\pi_j}.$$

Szlendak, R. et al. Permutation Compressors for Provably Faster Distributed Nonconvex Optimization

- Friendly to centralised communications without a server.
- In the homogeneous case have the physics of cheap full gradient forwarding.

Biased compression

- Randomisation is good, but there's gap for improvement.

Biased compression

We call the (stochastic) operator $C(x)$ a shifted compression operator if for any $x \in \mathbb{R}^d$ it is fulfilled:

$$\mathbb{E}[\|C(x) - x\|_2^2] \leq \left(1 - \frac{1}{\delta}\right) \|x\|_2^2,$$

where $\delta \geq 0$.

Biased compression: examples

"Greedy" sparsification (selection of the largest modular components)

Consider an operator

$$\mathrm{Top}_k(x) = \sum_{i=d-k+1}^{d} x_{(i)} e_{(i)},$$

where k — some fixed number from the set $\{1, \ldots, d\}$ (the number of components of vector x that we pass; for example, we can choose $k = 1$), with the coordinates sorted modulo: $|x_{(1)}| \leq |x_{(2)}| \leq \ldots \leq |x_{(d)}|$, $(e_1, \ldots, e_d)$ — the standard basis in $\mathbb{R}^d$. It can be shown that this operator is a shifted compression with constant $\delta = \frac{d}{k} k$.

Alistarh D. et al. The convergence of sparsified gradient methods

Biased compression: examples

- Various examples of compressors (sparsifiers, rounding, etc):
  Beznosikov A. et al. On Biased Compression for Distributed Learning
- A practical biased compressor based on iterative SVD decomposition:
  Vogels T. et al. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization

Biased compression: a theorem in the case of 1 node

Theorem (convergence of QGD with shifted compression in case of 1 node)

Let f μ be strongly convex (or PL) and have L-Lipschitz gradient, then QGD for one node with step $\gamma \leq 1/L$ and with biased compressor with parameter δ converges and is satisfied:

$$f(w_K) - f(w^*) \leq \left(1 - \frac{\gamma\mu}{\delta}\right)^K (f(w_0) - f(w^*)).$$

Beznosikov A. et al. On Biased Compression for Distributed Learning

Biased compression: it's not that simple

- Consider the following distributed problem with $M = 3$, $d = 3$ and local functions:

$$f_1(x) = \langle a, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_2(w) = \langle b, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_3(x) = \langle c, x\rangle^2 + \frac{1}{4}\|x\|^2,$$

where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.

Biased compression: it's not that simple

- Consider the following distributed problem with $M = 3$, $d = 3$ and local functions:

$$f_1(x) = \langle a, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_2(w) = \langle b, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_3(x) = \langle c, x\rangle^2 + \frac{1}{4}\|x\|^2,$$

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Q:** where is her optimum?

Biased compression: it's not that simple

- Consider the following distributed problem with $M = 3$, $d = 3$ and local functions:

$$f_1(x) = \langle a, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_2(w) = \langle b, x\rangle^2 + \frac{1}{4}\|x\|^2,\ f_3(x) = \langle c, x\rangle^2 + \frac{1}{4}\|x\|^2,$$

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Q:** where is her optimum? $(0, 0, 0)$.

Biased compression: it's not that simple

- Consider the following distributed problem with $M = 3$, $d = 3$ and local functions:

$$f_1(x) = \langle a, x\rangle^2 + \tfrac{1}{4}\|x\|^2,\ f_2(w) = \langle b, x\rangle^2 + \tfrac{1}{4}\|x\|^2,\ f_3(x) = \langle c, x\rangle^2 + \tfrac{1}{4}\|x\|^2,$$

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Q:** where is her optimum? $(0, 0, 0)$.
- Let the starting point $x_0 = (t, t, t)$ for some $t > 0$. Then the local gradients are:

$$\nabla f_1(x_0) = \tfrac{t}{2}(-11, 9, 9),\quad \nabla f_2(x_0) = \tfrac{t}{2}(9, -11, 9),\quad \nabla f_3(x_0) = \tfrac{t}{2}(9, 9, -11).$$

- **Q:** what will the `QGD` (gradient descent with compressions) step look like if we use $Top1$ compression?

Biased compression: it's not that simple

- Consider the following distributed problem with $M = 3$, $d = 3$ and local functions:

$$f_1(x) = \langle a, x\rangle^2 + \tfrac{1}{4}\|x\|^2,\ f_2(w) = \langle b, x\rangle^2 + \tfrac{1}{4}\|x\|^2,\ f_3(x) = \langle c, x\rangle^2 + \tfrac{1}{4}\|x\|^2,$$

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Q:** where is her optimum? $(0, 0, 0)$.
- Let the starting point $x_0 = (t, t, t)$ for some $t > 0$. Then the local gradients are:

$$\nabla f_1(x_0) = \tfrac{t}{2}(-11, 9, 9),\quad \nabla f_2(x_0) = \tfrac{t}{2}(9, -11, 9),\quad \nabla f_3(x_0) = \tfrac{t}{2}(9, 9, -11).$$

- **Q:** what will the `QGD` (gradient descent with compressions) step look like if we use $Top1$ compression?

$$x_1 = (t, t, t) + \gamma \cdot \frac{11}{6}(t, t, t) = \left(1 + \frac{11\gamma}{6}\right) x_0.$$

- We move away from the solution geometrically for any $\gamma > 0$.

Biased compression: error compensation

- Let's try to remember what we didn't pass on in the communication process:

$$e_{1,m} = 0 + \gamma \nabla f_m(x_0) - C(0 + \gamma \nabla f_m(x_0)).$$

Biased compression: error compensation

- Let's try to remember what we didn't pass on in the communication process:

$$e_{1,m} = 0 + \gamma \nabla f_m(x_0) - C(0 + \gamma \nabla f_m(x_0)).$$

- And add this to future parcels:

$$C(e_{1,m} + \gamma \nabla f_m(x_1))$$

Biased compression: error compensation

- Let's try to remember what we didn't pass on in the communication process:

$$e_{1,m} = 0 + \gamma \nabla f_m(x_0) - C(0 + \gamma \nabla f_m(x_0)).$$

- And add this to future parcels:

$$C(e_{1,m} + \gamma \nabla f_m(x_1))$$

- In an arbitrary iteration, it is written as follows:

$$\text{Parcel:} C(e_{k,m} + \gamma \nabla f_m(x_k)),$$
$$e_{k+1,m} = e_{k,m} + \gamma \nabla f_m(x_k) - C(e_{k,m} + \gamma \nabla f_m(x_k))$$

Biased compression: error compensation

- Let's try to remember what we didn't pass on in the communication process:

$$e_{1,m} = 0 + \gamma \nabla f_m(x_0) - C(0 + \gamma \nabla f_m(x_0)).$$

- And add this to future parcels:

$$C(e_{1,m} + \gamma \nabla f_m(x_1))$$

- In an arbitrary iteration, it is written as follows:

$$\text{Parcel:} C(e_{k,m} + \gamma \nabla f_m(x_k)),$$
$$e_{k+1,m} = e_{k,m} + \gamma \nabla f_m(x_k) - C(e_{k,m} + \gamma \nabla f_m(x_k))$$

- This technique is called error compensation (error feedback).

Stich S. et al. Sparsified SGD with memory

QGD with error feedback

Algorithm 1 QGD с error feedback

Input: step size $\gamma > 0$, starting point $x_0 \in \mathbb{R}^d$, starting errors $e_{0,m} = 0$ for all m from 1 to M, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  Send x_k to all workers ▷ by server
  for $m = 1, \ldots, M$ in parallel **do**
    Recieve x_k from the master ▷ by workers
    Compute $\nabla f(w_k)$ in x_k ▷ by workers
    Generate $g_{k,m} = C(e_{k,m} + \gamma \nabla f(x_k))$ ▷ by workers
    Вычислить $e_{k+1,m} = e_{k,m} + \gamma \nabla f_m(x_k) - g_{k,m}$ ▷ by workers
    Send $g_{k,m}$ to master ▷ by workers
  end for
  Recieve $g_{k,m}$ from all workers ▷ by server
  Compute $g_k = \frac{1}{M} \sum_{m=1}^{M} g_{k,m}$ ▷ by server
  $x_{k+1} = x_k - g_k$ ▷ by server
end for

Output: x_K

QGD with error feedback: convergence

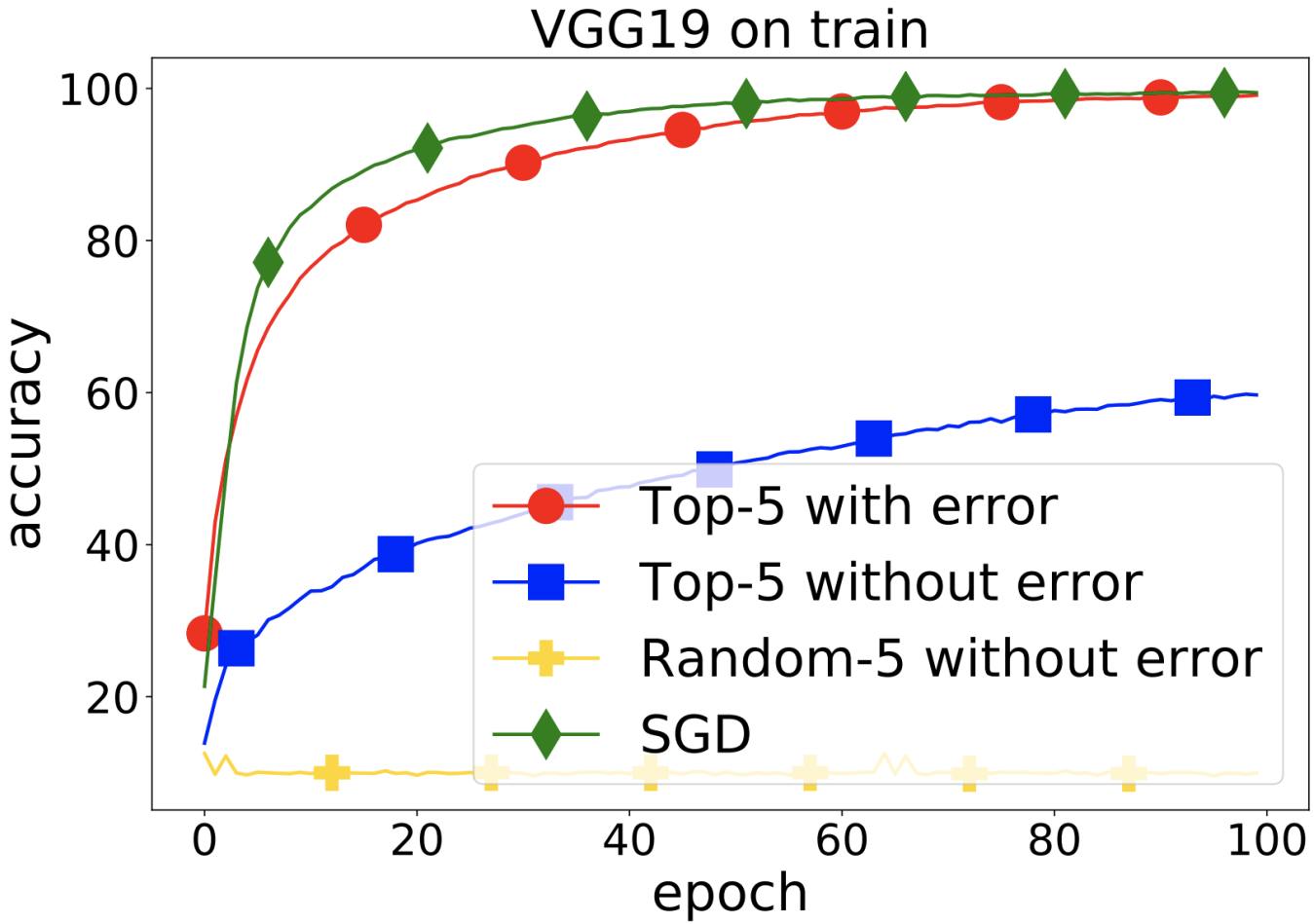


Figure: Accuracy during training of VGG19 on CIFAR10 using different compressors

QGD с error feedback: convergence

Theorem GD with error feedback

Let all local functions f_m be μ-simply convex and have L-Lipschitz gradient, then if $\gamma \leq \frac{1}{28\delta L}$ then

$$\mathbb{E}\left[f(\tilde{x}_K) - f(x^*)\right] \leq \mathcal{O}\left(\delta L \|x_0 - x^*\|^2 \exp\left(-\frac{\gamma\mu K}{2}\right) + \gamma \cdot \frac{\delta}{\mu} \cdot \frac{1}{M} \sum_{m=1}^{M} \|\nabla f_m(x^*)\|^2\right).$$

Stich S. and Karimireddy S. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication

Beznosikov A. et al. On Biased Compression for Distributed Learning

- Same problem as QGD – the second term in the evaluation should be eliminated

Biased compression: solving the plateau issue

- Idea like DIANA: memory + compression of difference

Algorithm 1 EF21 (sketch)

1: Each device m possesses a vector of "memory" $g_0^m = 0$

2: The server stores $h_0 = \frac{1}{M}\sum_{m=1}^{M} h_0^m = 0$

3: Send a compressed version of the difference to the server $C(\nabla f_m(w^k) - h_k^m)$

4: Refreshing the memory $h_{k+1}^m = h_k^m + C(\nabla f_m(x^k) - h_k^m)$

5: Server compute $h_{k+1} = h_k + \frac{1}{M}\sum_{m=1}^{M} C(\nabla f_m(x^k) - h_k^m)$

6: Update: $x_{k+1} = x_k - \gamma h_{k+1}$

Richtarik P. et al. EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback

Unbiased vs biased

- Best estimate on the number of communications for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[1+\frac{\omega}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- Best estimate on the number of communications for the unaccelerated method with biased compression (EF-21):

$$\mathcal{O}\left([1+\delta]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- It has already been discussed that these estimates are worse than for the baseline GD.

Unbiased vs biased

- Compressors compress information β times and it is typical that $\beta \geq \omega$ and $\beta \geq \delta$.

Unbiased vs biased

- Compressors compress information β times and it is typical that $\beta \geq \omega$ and $\beta \geq \delta$.
- Best estimate on the number of information for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[\frac{1}{\beta} + \frac{1}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- As discussed, the unbiased compressor provably improves the number of transmitted information.

Unbiased vs biased

- Compressors compress information β times and it is typical that $\beta \geq \omega$ and $\beta \geq \delta$.
- Best estimate on the number of information for the unaccelerated method with unbiased compression (DIANA):

$$\mathcal{O}\left(\left[\frac{1}{\beta} + \frac{1}{M}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- As discussed, the unbiased compressor provably improves the number of transmitted information.
- The biased compressor has an estimate:

$$\mathcal{O}\left(\left[\frac{1}{\beta} + \frac{\delta}{\beta}\right]\frac{L}{\mu}\log\frac{1}{\varepsilon}\right).$$

- The biased compressor does not improve the number of transmitted information in the general case. **And this is an open question: how to see the theoretical superiority of biased operators**.

Differential privacy

Why not existing approaches?

- Anonymization – Netflix example.

Why not existing approaches?

- Anonymization – Netflix example.
- Encryption – long and computationally expensive.

Differential privacy

- Suppose we are given data size of n where each sample is generated from a domain $\mathcal{X}$.
- We call two datasets neighboring if they differ by exactly one element.
- Suppose we consider some (possibly stochastic) algorithm $\mathcal{A}$ that, using a given date from $\mathcal{X}^d$, produces an answer Y (possibly a random variable) from some set E.

Definition

We say that the mechanism $\mathcal{A}$ is (ε, δ) - DP (differential private) if for all neighboring datasets $D, D' \in \mathcal{X}^n$ and for all $E \subseteq \mathcal{Y}$ it follows that

$$\mathbb{P}\{\mathcal{A}(D) \in E\} \leq e^{\varepsilon}\mathbb{P}\{\mathcal{A}(D') \in E\} + \delta.$$

- The essence - change one element in the sample, the algorithm's answer should not change much. I.e. nothing depends on data of one user!

Properties of differential privacy

- If $\varepsilon \to 0$, then $e^{\varepsilon} \to 1$. If additionally $\delta \to 0$, we have ideal situation.
- Typically, we want to have $\delta \leq \frac{1}{d}$

Differential privacy mechanisms

- Gaussian:

$$\tilde{\mathcal{A}}(D) = \mathcal{A}(D) + Z, \quad \text{with} \quad Z \sim \mathcal{N}(0, \sigma^2).$$

- If we want to have the (ε, δ) - DP algorithm $\tilde{\mathcal{A}}$, we need

$$\sigma = \frac{\Delta \ln \frac{5}{4\delta}}{\varepsilon}.$$

- Here we use sensitivity of the algorithm $\mathcal{A}$:

$$\Delta = \max_{\text{neighboring } D, D'} \|\mathcal{A}(D) - \mathcal{A}(D')\|$$

Differential privacy mechanisms

- Gaussian:

$$\tilde{\mathcal{A}}(D) = \mathcal{A}(D) + Z, \quad \text{with} \quad Z \sim \mathcal{N}(0, \sigma^2).$$

- If we want to have the (ε, δ) - DP algorithm $\tilde{\mathcal{A}}$, we need

$$\sigma = \frac{\Delta \ln \frac{5}{4\delta}}{\varepsilon}.$$

- Here we use sensitivity of the algorithm $\mathcal{A}$:

$$\Delta = \max_{\text{neighboring } D, D'} \|\mathcal{A}(D) - \mathcal{A}(D')\|$$

- Laplacian:

$$\tilde{A}(D) = A(D) + Z, \quad \text{with} \quad Z \sim \text{Lap}(\alpha).$$

- If we want to have the $(\varepsilon, 0)$ - DP algorithm $\tilde{A}$, we need

$$\alpha = \Delta / \varepsilon.$$

DP-ERM

- Consider

$$\min_{x \in \mathbb{R}^d} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(g(x, a_i), b_i)) + \lambda \varphi(x) \right].$$

DP-ERM

- Consider

$$\min_{x \in \mathbb{R}^d} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(g(x, a_i), b_i)) + \lambda \varphi(x) \right].$$

- For simplicity, we assume that we store all data on single device, but we forward something during training.
- **Q**: what do we send?

DP-ERM

- Consider

$$\min_{x \in \mathbb{R}^d} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(g(x, a_i), b_i)) + \lambda \varphi(x) \right].$$

- For simplicity, we assume that we store all data on single device, but we forward something during training.
- **Q**: what do we send? Models and gradients.

DP-ERM

- Model perturbation:

$$\tilde{x}_k = x_k + b \quad \text{with} \quad b \sim \mathcal{N}(0, \sigma \cdot I)$$

DP-ERM

- Model perturbation:

$$\tilde{x}_k = x_k + b \quad \text{with} \quad b \sim \mathcal{N}(0, \sigma \cdot I)$$

- Limiting result: if we add noise to x^* (optimal model) and $\sigma^2 \sim \frac{\log \delta^{-1}}{\lambda^2 \varepsilon^2}$, then we have (ε, δ)-DP.

DP-ERM

- Gradient perturbation:

$$g(x) = \nabla f(x) + b,$$

DP-ERM

- Gradient perturbation:

$$g(x) = \nabla f(x) + b,$$

but in practice to do in the other way with clipping:

$$g(x) = \min\left\{\frac{C}{\|\nabla f(x)\|}; 1\right\}\nabla f(x) + b.$$

Q: why do we need the clipping technique here?

DP-ERM

- Gradient perturbation:

$$g(x) = \nabla f(x) + b,$$

but in practice to do in the other way with clipping:

$$g(x) = \min\left\{\frac{C}{\|\nabla f(x)\|}; 1\right\}\nabla f(x) + b.$$

Q: why do we need the clipping technique here? To control gradient and in fact sensitivity in the ball.

DP-ERM

- Gradient perturbation:

$$g(x) = \nabla f(x) + b,$$

but in practice to do in the other way with clipping:

$$g(x) = \min\left\{\frac{C}{\|\nabla f(x)\|}; 1\right\}\nabla f(x) + b.$$

Q: why do we need the clipping technique here? To control gradient and in fact sensitivity in the ball.

- Result: if $\sigma^2 \sim \frac{C^2 \log \frac{1}{\delta}}{\varepsilon^2}$, then we have (ε, δ)-DP

Personalization

4 classical options

What kind of model can be used on local devices?

- Golbal model trained on a big open dataset on the server.
- The same as previous, but with local tuning.
- Global federated learning model.
- The same as previous, but with local tuning.

How to make better?

Change the objective:

- Difference penalization

$$\min_{x_1,\ldots,x_M \in \mathbb{R}^d} \left[\frac{1}{M} \sum_{m=1}^{M} f_m(x_m) + \frac{\lambda}{2M} \sum_{m=1}^{M} \|x_m - \bar{x}\|_2^2 \right],$$

where $\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x_m$.

How to make better?

Change the objective:

- Difference penalization

$$\min_{x_1,\ldots,x_M \in \mathbb{R}^d} \left[\frac{1}{M} \sum_{m=1}^{M} f_m(x_m) + \frac{\lambda}{2M} \sum_{m=1}^{M} \|x_m - \bar{x}\|_2^2 \right],$$

  where $\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x_m$.

- Train the network:

$$\min_{x_1,\ldots,x_M \in \mathbb{R}^d} \min_{W \in \mathcal{C}} \left[\frac{1}{M} \sum_{m=1}^{M} f_m(x_m) + \frac{\lambda}{2} X^T W X \right],$$

  where X is a matrix of x_m, W is a matrix with properties of the communication network.