Local Methods of Distributed Optimization

Aleksandr Beznosikov

3 May 2024

Modern trends in machine learning

- Exponential growth in model sizes and data volumes.

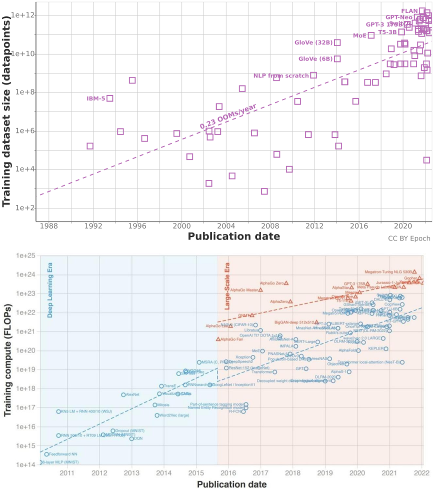


Figure: Trends in machine learning tasks

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

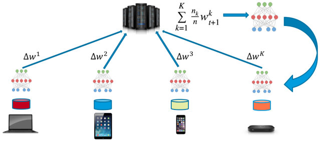


Figure: Federated Learning

The most common distributed setting

- Horizontal, offline distributed learning:

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

  where x – model weights, g – model, l – loss function.
- The data is shared among M computational devices, each device m has its own local subsample $\{a_i^m, b_i^m\}_{i=1}^{n_m}$ of size n_m.
- The focus of this presentation.

Communicate centralized via server

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

Algorithm Centralized Distributed GD

```
Input: Stepsize $\gamma > 0$, 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                                   ▷ server
    for $i = 1, \ldots, n$ in parallel do
        Receive $x_k$ from server                               ▷ workers
        Compute gradient $\nabla f_m(x_k)$ at point $x_k$       ▷ 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
    $x_{k+1} = x_k - \gamma \nabla f(x_k)$                      ▷ server
end for
```

What are we fighting for?

- **Question:** distributivity is necessary for parallelization, but why can't we achieve full parallelization?

What are we fighting for?

- **Question:** 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 actual for all distributed learning productions.
- There are many ways to fight for effective communication.

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):

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

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

PDF Mangasarian O. Parallel Gradient Distribution in Unconstrained Optimization

PDF 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.

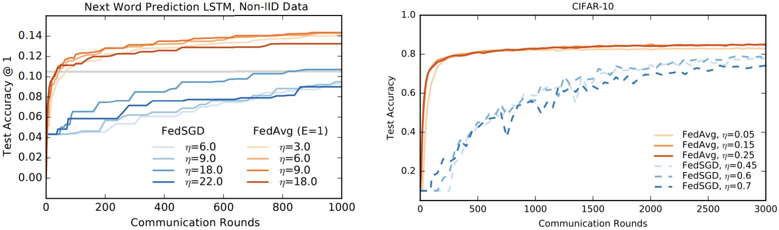


Figure: Comparison of Local SGD (FedAvg) and Centralized Distributed SGD (FedSGD).

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

How it convergences

- Problem: logistic regression on a5a LibSVM dataset.

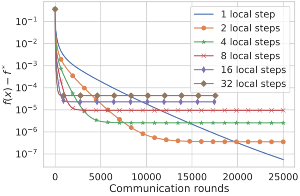


Figure: Dependence of convergence of Local SGD on number of local steps

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

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

How it convergences

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

How it convergences

- **Question:** 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:

$$\mathcal{O}\left(\frac{\|x^0 - x^*\|^2}{\gamma K} + \frac{\gamma \sigma_{opt}^2}{M}\right),$$

  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

- **Question:** 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:

$$\mathcal{O}\left(\frac{\|x^0 - x^*\|^2}{\gamma K} + \frac{\gamma \sigma_{opt}^2}{M}\right),$$

  where $\gamma \le \mathcal{O}\left(\frac{1}{LT}\right)$ – 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

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

Solving the problem

- **Question:** 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 $\tilde{f}_m$.

PDF 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

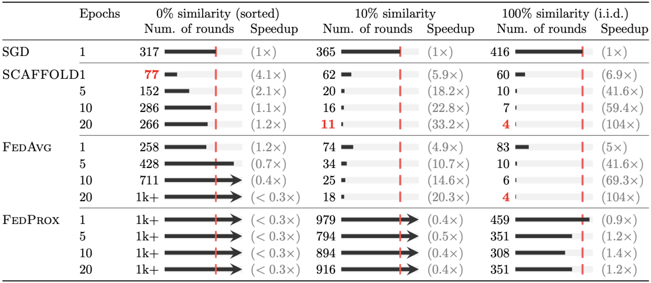
  Mishchenko K. et al. ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

- 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)		10% similarity		100% similarity (i.i.d.)	
		Num. of rounds	Speedup	Num. of rounds	Speedup	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 $\mathrm{E}[X_i] = 0$, 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

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

📄 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 $\varphi(x)$:

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

- **Question:** 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 $\varphi(x)$:

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

- **Question:** 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 с $x = x_{k+1}$, $y = x_k$:

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

- 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.

- **Question:** 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.

- **Question:** 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 Mirror Descent for the data similarity problem

```
Input: Stepsize $\gamma > 0$, starting point $x^0 \in \mathbb{R}^d$, number iterations $K$
for $k = 0, 1, \dots, K-1$ do
    Send $x_k$ to all workers                                        ▷ server
    for $m = 1, \dots, M$ 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

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

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

How it works

- Problem: ResNet-18 on CIFAR-10.

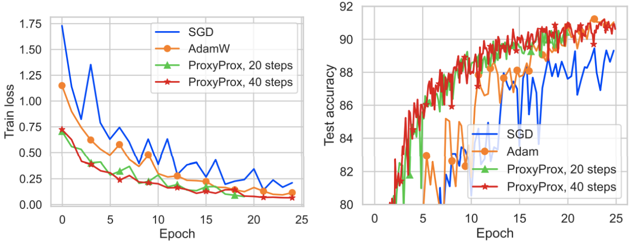


Figure: Comparison of Mirror Descent (ProxyProx) with SOTA optimizers on **non-distributed** problems.

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

Research questions: 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 questions: 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 questions: 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 distrbuted 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?

Research questions: extension

- We look at the story with similarity only for minimization problems, but there are other classes of problems.

Research questions: extension

- We look at the story with similarity only for minimization problems, but there are other classes of problems.
- The hessian similarity can be rewritten as δ-smoothness of $f - f_i$:

$$\|\nabla f(x) - \nabla f_i(x) - \nabla f(y) + \nabla f_i(y)\| \leq \delta \|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

Research questions: extension

- We look at the story with similarity only for minimization problems, but there are other classes of problems.
- The hessian similarity can be rewritten as δ-smoothness of $f - f_i$:

$$\|\nabla f(x) - \nabla f_i(x) - \nabla f(y) + \nabla f_i(y)\| \leq \delta \|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

- Let us consider the variational inequality problem:

$$\text{Find } z^* \in \mathcal{Z} \text{ such that } \langle F(z^*), z - z^* \rangle \geq 0, \; \forall z \in \mathcal{Z},$$

  where $F : \mathbb{R}^d \to \mathbb{R}^d$ is some operator.
- Minimization is also a VI with $F(z) := \nabla f(x)$. Also saddle point problem ($\min_x \max_y g(x, y)$) is a VI with $F(z) := (\nabla_x g(x, y), -\nabla_y g(x, y))$.

Research questions: extension

- We look at the story with similarity only for minimization problems, but there are other classes of problems.
- The hessian similarity can be rewritten as δ-smoothness of $f - f_i$:

$$\|\nabla f(x) - \nabla f_i(x) - \nabla f(y) + \nabla f_i(y)\| \leq \delta \|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

- Let us consider the variational inequality problem:

$$\text{Find } z^* \in \mathcal{Z} \text{ such that } \langle F(z^*), z - z^* \rangle \geq 0, \ \forall z \in \mathcal{Z},$$

  where $F : \mathbb{R}^d \to \mathbb{R}^d$ is some operator.
- Minimization is also a VI with $F(z) := \nabla f(x)$. Also saddle point problem ($\min_x \max_y g(x, y)$) is a VI with $F(z) := (\nabla_x g(x, y), -\nabla_y g(x, y))$.
- Make sense to consider distributed VIs under the following assumption:

$$\|F(x) - F_i(x) - F(y) + F_i(y)\| \leq \delta \|x - y\| \quad \forall x, y \in \mathbb{R}^d.$$

Research questions: break lower bounds

- The lower bounds say we achieve optimality and there's nothing more to do here.

Research questions: break lower bounds

- The lower bounds say we achieve optimality and there's nothing more to do here.
- **Question:** Is the Nesterov's method always optimal?

Research questions: break lower bounds

- The lower bounds say we achieve optimality and there's nothing more to do here.
- **Question:** Is the Nesterov's method always optimal? No! E.g., if we consider the specificity that the target function can be of the sum species $f(x) = \frac{1}{n} \sum\limits_{i=1}^{n} f_i(x)$.
- Katyusha has the following upper estimate of convergence (oracle complexity on the call f_i): $\mathcal{O}\left(\left[n + \sqrt{n\frac{L}{\mu}}\right] \log \frac{1}{\varepsilon}\right)$.

  Allen-Zhu Z. Katyusha: the first direct acceleration of stochastic gradient methods
- Upper bound on oracle complexity for the Nesterov's method is $\mathcal{O}\left(n\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$.
- Let us add specificity in communications, break out the lower bounds and get an even faster method.

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 $\varphi(x)$:

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

- Substitute $\varphi(x)$:

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

- 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 $\varphi(x)$:

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

- Substitute $\varphi(x)$:

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

- Or a little differently:

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

- **Question:** 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.

Extension to VIs

- We consider the following problem:

$$\text{Find } z^* \in \mathcal{Z} \text{ such that } \langle F(z^*), z - z^* \rangle \geq 0,\ \forall z \in \mathcal{Z},$$

  where $F : \mathbb{R}^d \to \mathbb{R}^d$ is some operator.

- As before, the problem is shared among M computing nodes, each device m has access only to its own operator F_m:

$$F(z) := \frac{1}{M} \sum_{m=1}^{M} F_m(z).$$

- We assume that F is μ-strongly monotone, F_1 is monotone and $F - F_1$ is δ-Lipschitz.

Extension to VIs: idea

- Optimal methods for VIs with Lipschitz operator are different from those for smooth minimization problems.
- We need to base on specific methods, e.g., ExtraGradient, Tseng's etc.
- And also use idea of sliding. Again: we need sliding for non-monotone+monotone=monotone.

Extension to VIs: method

Algorithm Star Min-Max Data Similarity Algorithm

```
Input: Stepsize $\gamma$, accuracy $e$, starting point $z^0 \in \mathcal{Z}$, $z_m^0 = z^0$, for all $m \in [M]$, number of iterations $K$
for $k = 0, 1, \ldots, K-1$ do
    for  $m = 1, \ldots, M$ in parallel do
        Compute $F_m(z^k)$ and sends it to server                                  ▷ workers
    end for
    Compute $v^k = z^k - \gamma \cdot \big(F(z^k) - F_1(z^k)\big)$                ▷ server
    Find $u^k$, s.t. $\|u^k - \hat{u}^k\|^2 \leq e$, where $\hat{u}^k$ is solution of    ▷ server
        $\langle \gamma F_1(\hat{u}^k) + \hat{u}^k - v^k, \hat{u}^k - z \rangle \leq 0, \quad \forall z \in \mathcal{Z}$
    Update $z^{k+1} = \mathrm{proj}_{\mathcal{Z}}\left[u^k + \gamma \cdot (F(z^k) - F_1(z^k) - F(u^k) + F_1(u^k))\right]$
and send $z^{k+1}$ to workers                                                       ▷ server
end for
```

Beznosikov A. et al. Distributed Saddle-Point Problems Under Similarity

Convergence

- If we want to achieve an accuracy ε ($\|z_K - z^*\|^2 \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

Convergence

- If we want to achieve an accuracy ε ($\|z_K - z^*\|^2 \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- Is optimal?

Convergence

- If we want to achieve an accuracy ε ($\|z_K - z^*\|^2 \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- Is optimal? For minimization no.

Convergence

- If we want to achieve an accuracy ε ($\|z_K - z^*\|^2 \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- Is optimal? For minimization no.
- But for VIs the lower bounds on communications are

$$K = \Omega\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right).$$

Convergence

- If we want to achieve an accuracy ε ($\|z_K - z^*\|^2 \sim \varepsilon$), then we need to

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- Is optimal? For minimization no.
- But for VIs the lower bounds on communications are

$$K = \Omega\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right).$$

- It gives optimality of the proposed method.

How it works

- Problem: robust regression problem (regression with adversarial noise) on synthetic data.

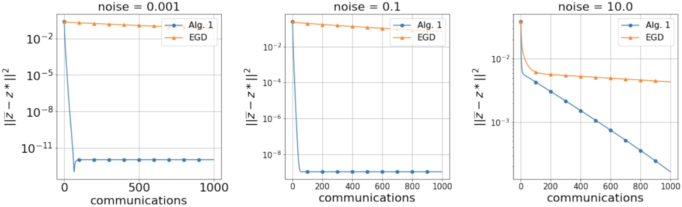


Figure: Comparison of Star Min-Max Data Similarity Algorithm (Alg. 1) with Centralized Distributed ExtraGradient (EGD).

Break lower bounds

- We discuss: more details about problem can give acceleration and break lower bounds.
- We consider the setting, where we can compress transmitted information.

Definition (Compression)

A stochastic operator $Q : \mathbb{R}^d \to \mathbb{R}^d$ is called *compression* if there exists a constant $q \geq 1$ such that

$$Q(z) = z, \quad \mathbb{E}\|Q(z)\|^2 \leq q\|z\|^2, \quad \forall z \in \mathbb{R}^d.$$

Break lower bounds: how to use compression

Direct compression is not super good even for minimization:

$$x_{k+1} = x_k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(\nabla f_m(x_k)).$$

Question: why?

Break lower bounds: how to use compression

Direct compression is not super good even for minimization:

$$x_{k+1} = x_k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(\nabla f_m(x_k)).$$

Question: why? Variance of stochastic gradient does not tend to 0.

Break lower bounds: how to use compression

Direct compression is not super good even for minimization:

$$x_{k+1} = x_k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(\nabla f_m(x_k)).$$

Question: why? Variance of stochastic gradient does not tend to 0.

- The same problem as in classical SGD:

$$x_{k+1} = x_k - \gamma \nabla f_{i_k}(x_k),$$

where i_k is generated randomly. **Question:** how to solve?

Break lower bounds: how to use compression

Direct compression is not super good even for minimization:

$$x_{k+1} = x_k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(\nabla f_m(x_k)).$$

Question: why? Variance of stochastic gradient does not tend to 0.

- The same problem as in classical SGD:

$$x_{k+1} = x_k - \gamma \nabla f_{i_k}(x_k),$$

where i_k is generated randomly. **Question:** how to solve? Variance reduction:

$$x_{k+1} = x_k - \gamma g_k \quad \text{with} \quad g_k = \nabla f_{i_k}(x_k) - \nabla f_{i_k}(w_k) + \nabla f(w_k),$$

where w_k is a reference point (coin flip update with small probability).

Break lower bounds: how to use compression

Direct compression is not super good even for minimization:

$$x_{k+1} = x_k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q(\nabla f_m(x_k)).$$

Question: why? Variance of stochastic gradient does not tend to 0.

- The same problem as in classical SGD:

  $$x_{k+1} = x_k - \gamma \nabla f_{i_k}(x_k),$$

  where i_k is generated randomly. **Question:** how to solve? Variance reduction:

  $$x_{k+1} = x_k - \gamma g_k \quad \text{with} \quad g_k = \nabla f_{i_k}(x_k) - \nabla f_{i_k}(w_k) + \nabla f(w_k),$$

  where w_k is a reference point (coin flip update with small probability).
- The same idea can be used for the stochasticity from compression:

  $$g_k = \frac{1}{M} \sum_{m=1}^{M} [F_m(x_k) - F_m(w_k)] + F(w_k)$$

Break lower bounds: what compression

- We need to take into account similarity.

Definition (Compression)

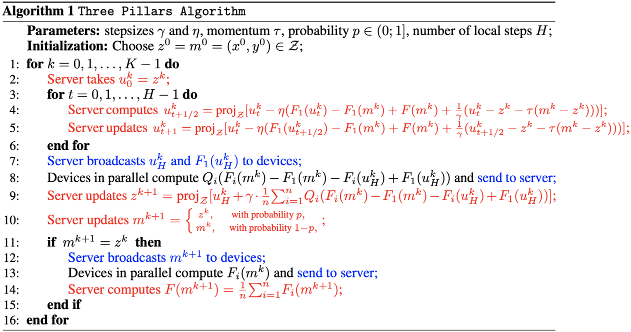
Assume that $d \geq M$ and $d = qM$, where $q \geq 1$ is an integer. Let $\pi = (\pi_1, \ldots, \pi_d)$ be a random permutation of $\{1, \ldots, d\}$. Then for all $u \in \mathbb{R}^d$ and each $i \in \{1, 2, \ldots, M\}$ we define

$$Q_m(u) = M \cdot \sum\nolimits_{j=q(m-1)+1}^{qm} u_{\pi_j} e_{\pi_j}.$$

Break lower bounds: idea

- Sliding
- Specific methods for VIs
- Variance reduction to deal with compression
- Permutation compression

Break lower bounds: method



```
Algorithm 1 Three Pillars Algorithm
  Parameters: stepsizes $\gamma$ and $\eta$, momentum $\tau$, probability $p \in (0;1]$, number of local steps $H$;
  Initialization: Choose $z^0 = m^0 = (x^0, y^0) \in \mathcal{Z}$;
  for $k = 0, 1, \dots, K-1$ do
      Server takes $u_0^k = z^k$;
      for $t = 0, 1, \dots, H-1$ do
          Server computes $u_{t+1/2}^k = \text{proj}_{\mathcal{Z}}[u_t^k - \eta(F_1(u_t^k) - F_1(m^k) + F(m^k) + \frac{1}{\gamma}(u_t^k - z^k - \tau(m^k - z^k)))]$;
          Server updates $u_{t+1}^k = \text{proj}_{\mathcal{Z}}[u_t^k - \eta(F_1(u_{t+1/2}^k) - F_1(m^k) + F(m^k) + \frac{1}{\gamma}(u_{t+1/2}^k - z^k - \tau(m^k - z^k)))]$;
      end for
      Server broadcasts $u_H^k$ and $F_1(u_H^k)$ to devices;
      Devices in parallel compute $Q_i(F_i(m^k) - F_1(m^k) - F_i(u_H^k) + F_1(u_H^k))$ and send to server;
      Server updates $z^{k+1} = \text{proj}_{\mathcal{Z}}[u_H^k + \gamma \cdot \frac{1}{n}\sum_{i=1}^n Q_i(F_i(m^k) - F_1(m^k) - F_i(u_H^k) + F_1(u_H^k))]$;
      Server updates $m^{k+1} = \begin{cases} z^k, & \text{with probability } p, \\ m^k, & \text{with probability } 1-p, \end{cases}$;
      if $m^{k+1} = z^k$ then
          Server broadcasts $m^{k+1}$ to devices;
          Devices in parallel compute $F_i(m^k)$ and send to server;
          Server computes $F(m^{k+1}) = \frac{1}{n}\sum_{i=1}^n F_i(m^{k+1})$;
      end if
  end for
```

PDF Beznosikov A. et al. Similarity, Compression and Local Steps: Three Pillars of Efficient Communications for Distributed Variational Inequalities

Convergence

- Method without compression needs

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- New method without compression needs

$$K = \mathcal{O}\left(\left[M + \frac{\delta\sqrt{M}}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

Question: which better?

Convergence

- Method without compression needs

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- New method without compression needs

$$K = \mathcal{O}\left(\left[M + \frac{\delta\sqrt{M}}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

Question: which better? In terms of communications, the basic is better, but by new method we transmitted M times less. And for new method complexity in terms of transmitted information is

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu\sqrt{M}}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

$\sqrt{M}$ times better than for the basic method!

Convergence

- Method without compression needs

$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

- New method without compression needs

$$K = \mathcal{O}\left(\left[M + \frac{\delta\sqrt{M}}{\mu}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

Question: which better? In terms of communications, the basic is better, but by new method we transmitted M times less. And for new method complexity in terms of transmitted information is

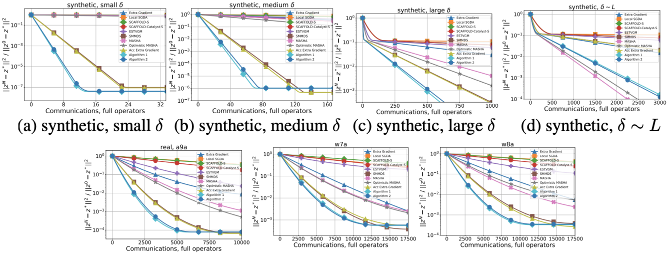
$$K = \mathcal{O}\left(\left[1 + \frac{\delta}{\mu\sqrt{M}}\right] \log \frac{\|z_0 - z^*\|^2}{\varepsilon}\right) \text{ communications.}$$

$\sqrt{M}$ times better than for the basic method!

- Lower bound gives optimality of the proposed method.

How it works

- Problem: robust regression on different data.



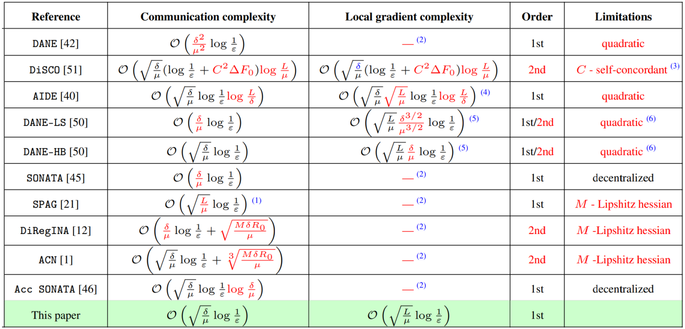
(a) synthetic, small δ (b) synthetic, medium δ (c) synthetic, large δ (d) synthetic, $\delta \sim L$

(e) real, a9a (f) real, w7a (g) real, w8a

Figure: Comparison of Three Pillars Algorithm with SOTA optimizers for distributed saddle point problems.

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 Accelerated Extragradient

```
Input: Stepsizes $\gamma$ and $\theta$, momentums $\alpha, \tau$, starting point $x_0 = x_0^f \in \mathbb{R}^d$,
    number of iterations $K$
for $k = 0, 1, 2, \dots, K-1$ do
    $x_k^g = \tau x_k + (1-\tau) x_k^f$
    $x_{k+1}^f \approx \arg\min_{x \in \mathbb{R}^d} \left[ \langle \nabla g_1(x_k^g), x - x_k^g \rangle + \frac{1}{2\theta} \| x - x_k^g \|^2 + g_2(x) \right]$
    $x_{k+1} = x_k + \eta\alpha(x_{k+1}^f - x_k) - \eta \nabla g(x_{k+1}^f)$
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.

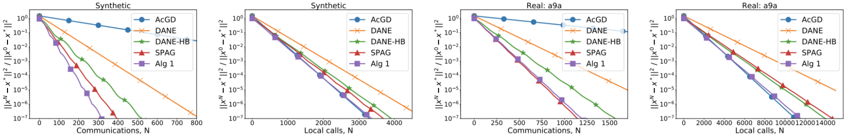


Figure: Comparison of Accelerated ExtraGradient (Alg. 1) with SOTA optimizers for distributed minimization problems under data similarity.