On Distributed Methods for Variational Inequalities and Beyond

Aleksandr Beznosikov

MIPT

17 January 2024

Variational Inequality

Definition (Stampacchia VIs)

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

where $F : \mathbb{R}^d \to \mathbb{R}^d$ is some operator and g is a proper convex lower semicontinuous function.

Definition (Minty VIs)

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

where $F : \mathbb{R}^d \to \mathbb{R}^d$ is some operator and g is a proper convex lower semicontinuous function.

Variational Inequality: facts

- Two formulations are equivariant for smooth monotone operators.
- In the case when $g \equiv 0$ and $\mathcal{Z} = \mathbb{R}^d$, then VI is equal to

$$\text{Find } z^* \in \mathcal{Z} \text{ such that } F(z) = 0.$$

Variational Inequality: examples

- Minimization:

$$\min_{z \in \mathbb{R}^d} f(z).$$

  We take $F(z) \stackrel{\text{def}}{=} \nabla f(z)$.

- Saddle point problem:

$$\min_{x \in \mathbb{R}^{d_x}} \min_{y \in \mathbb{R}^{d_y}} g(x, y).$$

  Here $F(z) \stackrel{\text{def}}{=} F(x, y) = [\nabla_x g(x, y), -\nabla_y g(x, y)]$.

- Fixed point problem:

$$\text{Find } z^* \in \mathbb{R}^d \text{ such that } T(z^*) = z^*,$$

  where $T : \mathbb{R}^d \to \mathbb{R}^d$ is an operator. We take $F(z) = z - T(z)$.

Variational Inequality: classical examples

- Game theory and economy (comes from von Neumann). Simple example – matrix game (bilinear sadddle point problem on simplexes):

$$\min_{x \in \Delta^{d_x}} \max_{y \in \Delta^{d_y}} x^T A y,$$

  where A – cost matrix, x и y – probability of actions.
- Constrained optimization and Lagrange multipliers.

Variational Inequality: ML example

- From classical minimization problem:

$$\min_{z \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i, z), y_i),$$

  where $\{x_i, y_i\}_{i=1}^n$ – data, f – model z, l – loss.
- To robust formulation via saddle point problem:

$$\min_{z \in \mathbb{R}^d} \max_{\|\delta_i\| \leq e} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i + \delta_i, z), y_i),$$

  where δ_i – adversarial noise.

Variational Inequality: GANs

- GAN represents two models generator G and discriminator D.
- D takes an element x as input and determines whether this element is real (from a data sample) or artificially generated by the generator.
- The generator is given some random vector z as input, from which the generator constructs a "fake" instance similar to the real sample.
- Formally, the GAN training problem is formulated as a saddle point problem:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Variational Inequality: method

- We know what to do with:

$$\min_{z \in \mathbb{R}^d} f(z).$$

  Gradient descent:

$$z^{k+1} = z^k - \gamma \nabla f(z^k).$$

- What to do with VIs and saddle point problems? The same idea – descent-ascent:

$$z^{k+1} = z^k - \gamma F(z^k).$$

Variational Inequality: method

- The idea of descent-ascent isn't bad and often works, but physical intuition tells that it has some not-so-pleasant aspects.

Variational Inequality: method

- The idea of descent-ascent isn't bad and often works, but physical intuition tells that it has some not-so-pleasant aspects.
- Consider $\min_{x \in R} \max_{y \in \mathbb{R}} xy$. With starting point $(1, 1)$. Where is the solution? Point $(0, 0)$.
- Vector: $\begin{pmatrix} \nabla_x g(x^k, y^k) \\ -\nabla_y g(x^k, y^k) \end{pmatrix}$ is always orthogonal to $\begin{pmatrix} x^k - x^* \\ y^k - y^* \end{pmatrix}$. What does it means? Method diverges.
- Intuition is not strict, but it can tell us to try something a little different.

Variational Inequality: Extragradient method

Algorithm Extragradient method

Вход: stepsize $\gamma > 0$, staring $z^0 \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: $\quad z^{k+1/2} = z^k - \gamma F(z^k)$
3: $\quad z^{k+1} = z^k - \gamma F(z^{k+1/2})$
4: **end for**

It is easy to check that for this method on the problem $\min_{x \in R} \max_{y \in R} xy$, the directions of the final step in the scalar product with the direction to the solution gives a number greater than 0, hence an acute angle.

Contemporary challenges

- Exponential growth in model sizes and data volumes.

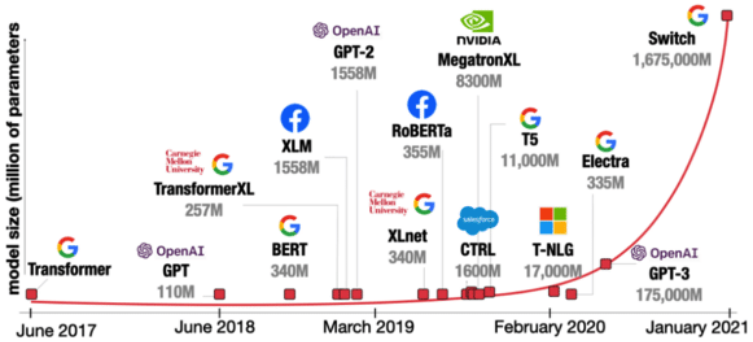

Figure: Dynamics of growth of modern language models

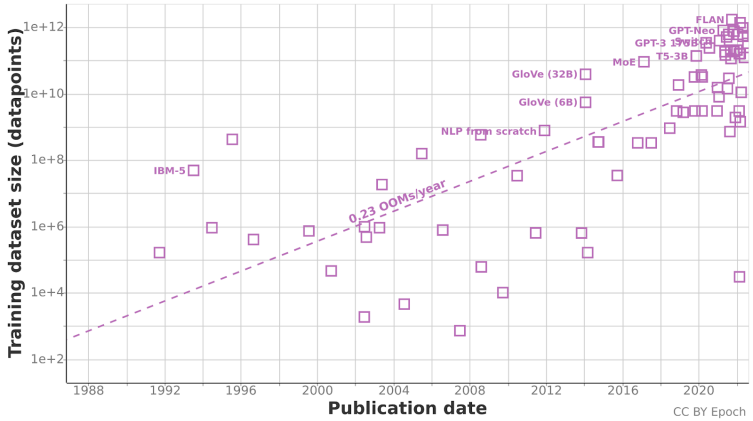

Figure: Dynamics of dataset growth

Varieties of distributed learning

- Cluster learning (large players): we train within one large and powerful computing cluster
- Collaborative learning (all players): pooling computing resources over the Internet
- Federated learning (another paradigm): learn on users' local data using their computational power

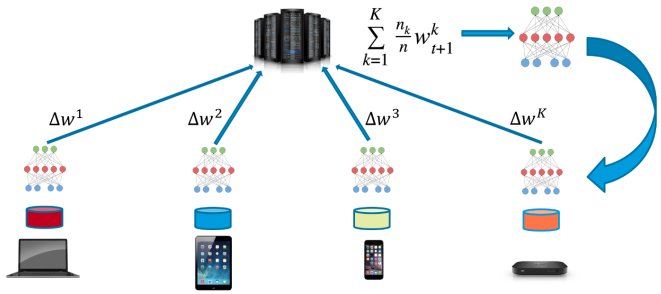

Figure: Federated learning

The most popular distributed setup

- Formulation (horizontal):

$$F(z) := \frac{1}{M}\sum_{m=1}^{M} F_m(z) := \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\xi\sim\mathcal{D}_m}[F_m(z,\xi)].$$

- The problem is shared among M computing devices, each device m has access only to its own operator F_m or its stochastic realization.

Communication setups

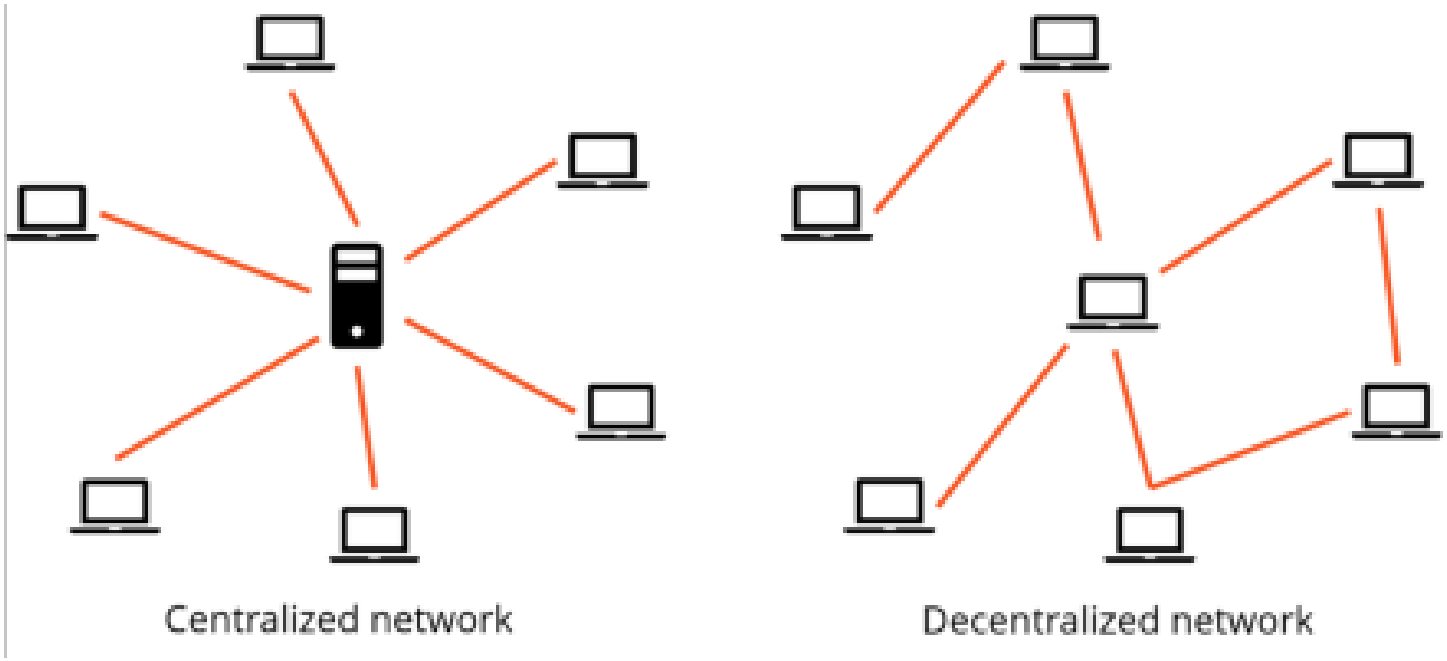

Figure: Centralized and decentralized connections

Communicating through the server

- Let us look at an example of how Extragradient becomes centralized.

Algorithm Centralized Extragradient

Вход: Stepsize $\gamma > 0$, starting point $z_0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**
  Send z^k to all workers ▷ by server
  for $m = 1, \ldots, M$ in parallel **do**
    Recieve z^k from server ▷ by workers
    Compute $F_m(z^k)$ in z^k ▷ by workers
    Send $F_m(z^k)$ to server ▷ by workers
  end for
  Recieve $F_m(z^k)$ from all workers ▷ by server
  Compute $F(z^k) = \frac{1}{M}\sum_{m=1}^{M} F_m(z^k)$ ▷ by server
  $z^{k+1/2} = z^k - \gamma F(z^k)$ ▷ by server
  Similarly for z^{k+1}
end for

- In the decentralized setting, it does not work, there is no server.

Assumptions

- **Assumption 1.** F_m is Lipschitz with constant L, i.e. for all $z_1, z_2 \in \mathcal{Z}$

$$\|F_m(z_1) - F_m(z_2)\| \leq L\|z_1 - z_2\|.$$

(smoothness)

- **Assumption 2.** F_m is strongly monotone with constant μ, i.e. for all $z_1, z_2 \in \mathcal{Z}$

$$\langle F_m(z_1) - F_m(z_2), z_1 - z_2\rangle \geq \mu\|z_1 - z_2\|^2.$$

(strong convexity and strong-convexity-strong-concavity)

- **Assumption 3.** (for decentralized setting) F_m is stored locally on its own device. All devices are connected in a network (undirected may be time-varying graph $G_k(\mathcal{V}_k, \mathcal{E}_k)$ with max diameter Δ and max condition number χ of Laplace matrix).

Plan

- Deterministic case
- Stochastic: bounded variance
- Stochastic: finite sum
- Compression
- Similarity
- Similarity + compression

Deterministic case

- We can compute full F_m on each device:

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

Lower bounds

Lower bounds for distributed algorithms with K communications.

centralized	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{32\mu K}{L}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L}}\right)\right)$
decentralized (fixed network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\sqrt{\chi}}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L}\sqrt{\chi}}\right)\right)$
decentralized (time-varying network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\chi}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L\chi}}\right)\right)$

Table: Lower bounds for distributed VIs.

Lower bounds

- No "problem" acceleration (unlike minimization) since VIs is a broader class of problems
- No "network" acceleration in the time-varying setting

Lower bounds: idea

- Problem:

$$f_m(x,y) = \begin{cases} f_1(x,y) = \frac{M}{2|B_d|} \cdot \frac{L}{2} x^T A_1 y + \frac{\mu}{2}\|x\|^2 - \frac{\mu}{2}\|y\|^2 + \frac{M}{2|B_d|} \cdot \frac{L^2}{2\mu} e_1^T y, & m \in B_d \\ f_2(x,y) = \frac{M}{2|B|} \cdot \frac{L}{2} x^T A_2 y + \frac{\mu}{2}\|x\|^2 - \frac{\mu}{2}\|y\|^2, & m \in B \\ f_3(x,y) = \frac{\mu}{2}\|x\|^2 - \frac{\mu}{2}\|y\|^2, & \text{otherwise} \end{cases}$$

where $e_1 = (1, 0 \ldots, 0)$ and

$$A_1 = \begin{pmatrix} 1 & 0 & & & & & & \\ & 1 & -2 & & & & & \\ & & 1 & 0 & & & & \\ & & & 1 & -2 & & & \\ & & & & \ldots & \ldots & & \\ & & & & & 1 & -2 & \\ & & & & & & 1 & 0 \\ & & & & & & & 1 \end{pmatrix}, \; A_2 = \text{analogically}$$

- Network – chain

Optimal algorithms

- In the centralized case, just Centralized Extragradient
- In the decentralized case, we can simulate server communication via gossip procedures.

Algorithm FastMix

Parameters: Vectors $z_1, ..., z_M$, communic. rounds P.

Initialization: Construct matrix $\mathbf{z}$ with rows $z_1^T, ..., z_M^T$, choose $\mathbf{z}^{-1} = \mathbf{z}$, $\mathbf{z}^0 = \mathbf{z}$, $\eta = \frac{1-\sqrt{1-\lambda_2^2(\tilde{W})}}{1+\sqrt{1-\lambda_2^2(\tilde{W})}}$.

for $h = 0, 1, 2, \ldots, P-1$ **do**

$\mathbf{z}^{h+1} = (1+\eta)\tilde{W}\mathbf{z}^h - \eta\mathbf{z}^{h-1}$,

end for

Output: rows $z_1, ..., z_M$ of $\mathbf{z}^P$.

Stochastic setting: bounded variance

- We can compute only stochastic realizations $F_m(z, \xi)$ for each device:

$$F(z) := \frac{1}{M}\sum_{m=1}^{M} F_m(z) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\xi\sim\mathcal{D}_m}[F_m(z,\xi)].$$

- **Assumption.** $F_m(z, \xi)$ is unbiased and has bounded variance, i.e. for all $z \in \mathcal{Z}$

$$\mathbb{E}[F_m(z,\xi)] = F_m(z),\ \mathbb{E}[\|F_m(z,\xi) - F_m(z)\|^2] \le \sigma^2.$$

Lower bounds

Lower bounds for distributed algorithms with K communications and T local computations ($T > K$).

centralized	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{32\mu K}{L}\right) + \frac{\sigma^2}{\mu^2 MT}\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L}}\right) + \frac{\sigma^2}{\mu^2 MT}\right)$
decentralized (fixed network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\sqrt{\chi}}\right) + \frac{\sigma^2}{\mu^2 MT}\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L}\sqrt{\chi}} + \frac{\sigma^2}{\mu^2 MT}\right)\right)$
decentralized (time-varying network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\chi}\right) + \frac{\sigma^2}{\mu^2 MT}\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu} K}{\sqrt{L}\chi}\right) + \frac{\sigma^2}{\mu^2 MT}\right)$

Lower bounds: idea

- Consider

$$\min_{x \in \mathbb{R}} f(x) = \frac{\mu}{2}(x - x_0)^2,$$

  where we do not know the constant $x_0 \neq 0$.
- Using stochastic first order oracle

$$\nabla f(x, \xi) = \mu(x + \xi - x_0), \text{ where } \xi \in \mathcal{N}\left(0, \frac{\sigma^2}{\mu^2}\right).$$

Optimal algorithms: centralized

Batching as additional idea to the deterministic algorithm

Algorithm 1 Centralized Extra Step Method

Parameters: Stepsize $\gamma \leq \frac{1}{4L}$; Communication rounds K, number of local steps T.
Initialization: Choose $(x^0, y^0) = z^0 \in \mathcal{Z}$, $k = \left\lfloor \frac{K}{r} \right\rfloor$ and batch size $b = \left\lfloor \frac{T}{2k} \right\rfloor$.
for $t = 0, 1, 2, \ldots, k$ **do**
  for each machine m **do**
    $g_m^t = \frac{1}{b} \sum_{i=1}^{b} F_m(z^t, \xi_m^{t,i})$, send g_m^t,
  on server:
    $z^{t+1/2} = \text{proj}_{\mathcal{Z}}(z^t - \frac{\gamma}{M} \sum_{m=1}^{M} g_m^t)$, send $z^{t+1/2}$,
  for each machine m **do**
    $g_m^{t+1/2} = \frac{1}{b} \sum_{i=1}^{b} F_m(z^{t+1/2}, \xi_m^{t+1/2,i})$, send $g_m^{t+1/2}$,
  on server:
    $z^{t+1} = \text{proj}_{\mathcal{Z}}(z^t - \frac{\gamma}{M} \sum_{m=1}^{M} g_m^{t+1/2})$, send z^{t+1},
end for
Output: z^{k+1} or z_{avg}^{k+1}.

Optimal algorithms: decentralized

Algorithm 2 Decentralized Extra Step Method

Parameters: Stepsize $\gamma \leq \frac{1}{4L}$; Communication rounds K, number of local calls T.
Initialization: Choose $(x^0, y^0) = z^0 \in \mathcal{Z}$, $z_m^0 = z^0$, $k = \left\lfloor \frac{K}{H} \right\rfloor$ and batch size $b = \left\lfloor \frac{T}{2k} \right\rfloor$.
for $t = 0, 1, 2, \ldots, k$ **do**
  for each machine m **do**
    $g_m^t = \frac{1}{b} \sum_{i=1}^{b} F_m(z_m^t, \xi_m^{t,i}), \quad \hat{z}_m^{t+1/2} = z_m^t - \gamma g_m^t,$
  communication
    $\tilde{z}_1^{t+1/2}, ..., \tilde{z}_M^{t+1/2} = \text{FastMix}(\hat{z}_1^{t+1/2}, ..., \hat{z}_M^{t+1/2}, H),$
  for each machine m **do**
    $z_m^{t+1/2} = \text{proj}_{\mathcal{Z}}(\tilde{z}_m^{t+1/2}),$
    $g_m^{t+1/2} = \frac{1}{b} \sum_{i=1}^{b} F_m(z_m^{t+1/2}, \xi_m^{t+1/2,i}),$
    $\hat{z}_m^{t+1} = z_m^t - \gamma g_m^{t+1/2},$
  communication
    $\tilde{z}_1^{t+1}, ..., \tilde{z}_M^{t+1} = \text{FastMix}(\hat{z}_1^{t+1}, ..., \hat{z}_M^{t+1}, H),$
  for each machine m **do**
    $z_m^{t+1} = \text{proj}_{\mathcal{Z}}(\tilde{z}_m^{t+1}),$
end for
Output: $\bar{z}^{k+1}$ or $\bar{z}_{avg}^{k+1}$.

Stochastic case: finite-sum

- We can compute full F_m on each device:

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

- But we don't want to do it since expensive, we compute only random part $F_{m,i}$.

Lower bounds

Lower bounds for distributed algorithms with K communications and T local computations.

centralized	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{32\mu K}{L}\right) + R_0^2 \exp\left(-\frac{16\mu K}{\sqrt{n}L}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{L}}\right) + R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{n}\sqrt{L}}\right)\right)$
decentralized (fixed network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\sqrt{\chi}}\right) + R_0^2 \exp\left(-\frac{16\mu K}{\sqrt{n}L}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{L}\sqrt{\chi}}\right) + R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{n}\sqrt{L}}\right)\right)$
decentralized (time-varying network)	
VIs	$\Omega\left(R_0^2 \exp\left(-\frac{128\mu K}{L\chi}\right) + R_0^2 \exp\left(-\frac{16\mu K}{\sqrt{n}L}\right)\right)$
Minimization (exists)	$\Omega\left(R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{L}\chi}\right) + R_0^2 \exp\left(-\frac{\sqrt{\mu}K}{\sqrt{n}\sqrt{L}}\right)\right)$

Lower bounds: idea

- Double separation
- Random choice of batch

Optimal algorithms: non-distributed with bathching

- New variance reduction algorithm

Algorithm 4

Parameters: Stepsizes $\eta > 0$, momentums α, γ, batchsize $b \in \{1, \ldots, n\}$, probability $p \in (0, 1)$
Initialization: Choose $z^0 = w^0 \in \operatorname{dom} g$. Put $z^{-1} = z^0, w^{-1} = w^0$
for $k = 0, 1, 2 \ldots$ **do**
  Sample $j_1^k, \ldots, j_b^k$ independently from $\{1, \ldots, m\}$ uniformly at random
  $S^k = \{j_1^k, \ldots, j_b^k\}$
  $\Delta^k = \frac{1}{b} \sum_{j \in S^k} \left(F_j(x^k) - F_j(w^{k-1}) + \alpha(F_j(x^k) - F_j(x^{k-1}))\right) + F(w^{k-1})$
  $x^{k+1} = \operatorname{prox}_{\eta g}(x^k + \gamma(w^k - x^k) - \eta \Delta^k)$
  $w^{k+1} = \begin{cases} x^{k+1}, & \text{with probability } p \\ w^k, & \text{with probability } 1 - p \end{cases}$
end for

Optimal algorithms: fixed network

Algorithm 1

Parameters: Stepsizes $\eta, \theta > 0$, momentums α, β, γ, batchsize $b \in \{1, \dots, n\}$, probability $p \in (0, 1)$

Initialization: Choose $\mathbf{z}^0 = \mathbf{w}^0 \in (\operatorname{dom} g)^M$, $\mathbf{y}^0 \in \mathcal{L}^\perp$. Put $\mathbf{z}^{-1} = \mathbf{z}^0, \mathbf{w}^{-1} = \mathbf{w}^0, \mathbf{y}^{-1} = \mathbf{y}^0$

for $k = 0, 1, 2 \dots$ **do**

  Sample $j^k_{m,1}, \dots, j^k_{m,b}$ independently from $[n]$

  $S^k = \{j^k_{m,1}, \dots, j^k_{m,b}\}$

  Sample $j^{k+1/2}_{m,1}, \dots, j^{k+1/2}_{m,b}$ independently from $[n]$

  $S^{k+1/2} = \{j^{k+1/2}_{m,1}, \dots, j^{k+1/2}_{m,b}\}$

  $$\delta^k = \frac{1}{b} \sum_{j \in S^k} \Big(\mathbf{F}_j(\mathbf{z}^k) - \mathbf{F}_j(\mathbf{w}^{k-1}) + \alpha[\mathbf{F}_j(\mathbf{z}^k) - \mathbf{F}_j(\mathbf{z}^{k-1})] \Big) + \mathbf{F}(\mathbf{w}^{k-1})$$

  $$\Delta^k = \delta^k - (\mathbf{y}^k + \alpha(\mathbf{y}^k - \mathbf{y}^{k-1}))$$

  $$\mathbf{z}^{k+1} = \operatorname{prox}_{\eta \mathbf{g}}(\mathbf{z}^k + \gamma(\mathbf{w}^k - \mathbf{z}^k) - \eta \Delta^k)$$

  $$\Delta^{k+1/2} = \frac{1}{b} \sum_{j \in S^{k+1/2}} \left(\mathbf{F}_j(\mathbf{z}^{k+1}) - \mathbf{F}_j(\mathbf{w}^k) \right) + \mathbf{F}(\mathbf{w}^k)$$

  $$\mathbf{y}^{k+1} = \mathbf{y}^k - \theta(\mathbf{W} \otimes \mathbf{I}_d)(\mathbf{z}^{k+1} - \beta(\Delta^{k+1/2} - \mathbf{y}^k))$$

  $$\mathbf{w}^{k+1} = \begin{cases} \mathbf{z}^k, & \text{with probability } p \\ \mathbf{w}^k, & \text{with probability } 1 - p \end{cases}$$

end for

Optimal algorithms: fixed network

Algorithm 2

Parameters: Stepsizes $\eta_z, \eta_y, \eta_x, \theta > 0$, momentums $\alpha, \gamma, \omega, \tau$, parameters ν, β, batchsize $b \in \{1, \dots, n\}$, probability $p \in (0, 1)$

Initialization: Choose $\mathbf{z}^0 = \mathbf{w}^0 \in (\operatorname{dom} g)^M$, $\mathbf{y}^0 \in (\mathbb{R}^d)^M$, $\mathbf{x}^0 \in \mathcal{L}^\perp$. Put $\mathbf{z}^{-1} = \mathbf{z}^0, \mathbf{w}^{-1} = \mathbf{w}^0$, $\mathbf{y}_f = \mathbf{y}^{-1} = \mathbf{y}^0, \mathbf{x}_f = \mathbf{x}^{-1} = \mathbf{x}^0$, $m_0 = \mathbf{0}^{dM}$

for $k = 0, 1, 2, \dots$ **do**

Sample $j^k_{m,1}, \dots, j^k_{m,b}$ independently from $[n]$

$S^k = \{j^k_{m,1}, \dots, j^k_{m,b}\}$

Sample $j^{k+1/2}_{m,1}, \dots, j^{k+1/2}_{m,b}$ independently from $[n]$

$S^{k+1/2} = \{j^{k+1/2}_{m,1}, \dots, j^{k+1/2}_{m,b}\}$

$$\delta^k = \frac{1}{b}\sum_{j \in S^k} \Big(\mathbf{F}_j(\mathbf{z}^k) - \mathbf{F}_j(\mathbf{w}^{k-1}) + \alpha[\mathbf{F}_j(\mathbf{z}^k) - \mathbf{F}_j(\mathbf{z}^{k-1})]\Big) + \mathbf{F}(\mathbf{w}^{k-1})$$

$$\Delta^k_z = \delta^k - \nu\mathbf{z}^k - \mathbf{y}^k - \alpha(\mathbf{y}^k - \mathbf{y}^{k-1})$$

$$\mathbf{z}^{k+1} = \operatorname{prox}_{\eta_z \mathbf{g}}(\mathbf{z}^k + \omega(\mathbf{w}^k - \mathbf{z}^k) - \eta_z \Delta^k_z)$$

$$\mathbf{y}^k_c = \tau\mathbf{y}^k + (1-\tau)\mathbf{y}^k_f$$

$$\mathbf{x}^k_c = \tau\mathbf{x}^k + (1-\tau)\mathbf{x}^k_f$$

$$\Delta^k_y = \nu^{-1}(\mathbf{y}^k_c + \mathbf{x}^k_c) + \mathbf{z}^{k+1} + \gamma(\mathbf{y}^k + \mathbf{x}^k + \nu\mathbf{z}^k)$$

$$\delta^{k+1/2} = \frac{1}{b}\sum_{j \in S^{k+1/2}} \big(\mathbf{F}_j(\mathbf{z}^{k+1}) - \mathbf{F}_j(\mathbf{w}^k)\big) + \mathbf{F}(\mathbf{w}^k)$$

$$\Delta^k_x = \nu^{-1}(\mathbf{y}^k_c + \mathbf{x}^k_c) + \beta(\mathbf{x}^k + \delta^{k+1/2})$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \eta_y \Delta^k_y$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (\mathbf{W}_T(Tk) \otimes \mathbf{I}_d)(\eta_x \Delta^k_x + m^k)$$

$$m^{k+1} = \eta_x \Delta^k_x + m^k - (\mathbf{W}_T(Tk) \otimes \mathbf{I}_d)(\eta_x \Delta^k_x + m^k)$$

$$\mathbf{y}^{k+1}_f = \mathbf{y}^k_c + \tau(\mathbf{y}^{k+1} - \mathbf{y}^k)$$

$$\mathbf{x}^{k+1}_f = \mathbf{x}^k_c - \theta(\mathbf{W}_T(Tk) \otimes \mathbf{I}_d)(\mathbf{y}^k_c + \mathbf{x}^k_c)$$

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{z}^k, & \text{with probability } p \\ \mathbf{w}^k, & \text{with probability } 1-p \end{cases}$$

end for

Quantization and compression

Definition (Quantization)

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

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

Expected/average compression (how much less the compressed vector takes up in memory): $\beta^{-1} \stackrel{\text{def}}{=} \frac{\mathsf{E}\|Q(z)\|_{\text{bits}}}{\|z\|_{\text{bits}}}$. Note that $\beta \geq 1$.

Examples: random selection of coordinates.

Definition (Compression)

(Stochastic) operator $C : \mathbb{R}^d \to \mathbb{R}^d$ is called *compression* if there exists $\delta \geq 1$ such that

$$\mathsf{E}\|C(z) - z\|^2 \leq (1 - 1/\delta)\|z\|^2, \quad \forall z \in \mathbb{R}^d.$$

Expected/average compression (how much less the compressed vector occupies in memory): $\beta^{-1} \stackrel{\text{def}}{=} \frac{\mathsf{E}\|C(z)\|_{\text{bits}}}{\|z\|_{\text{bits}}}$. Отметим, что $\beta \geq 1$.

Examples: Greedy choice of coordinates, low-rank decompositions,

Ideas

- For example, quantized extragradient method

$$z^{k+1/2} = z^k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q_1(F_m(z^k))$$

$$z^{k+1} = z^k - \gamma \cdot \frac{1}{M} \sum_{m=1}^{M} Q_2(F_m(z^{k+1/2}))$$

- Different Q are taken here. In fact it can be the same operator in terms of physics, but with different or the same randomness.

Ideas

- Good idea: variance reduction.
- Non-distributed problem:

$$\min_{z \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(z)$$

And the next method:

$$z^{k+1} = z^k - \gamma \cdot (\nabla f_{i_k}(z^k) - \nabla f_{i_k}(w^k) + \nabla f(w^k))$$

$$w^{k+1} = \begin{cases} w^k & \text{with prob.} \quad \tau \\ z^k & \text{with prob} \quad 1 - \tau \end{cases}$$

Algorithm 1 MASHA1

Parameters: Stepsize $\gamma > 0$, parameter $\tau \in (0; 1)$, number of iterations K.
Initialization: Choose $z^0 = w^0 \in \mathcal{Z}$.
Devices send $F_m(w^0)$ to server and get $F(w^0)$
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for each device m in parallel **do**
    $z^{k+1/2} = \tau z^k + (1-\tau)w^k - \gamma F(w^k)$
    Sends $g_m^k = Q_m^{\text{dev}}(F_m(z^{k+1/2}) - F_m(w^k))$ to server
  end for
  for server **do**
    Sends to devices $g^k = Q^{\text{serv}}\left[\frac{1}{M}\sum_{m=1}^M g_m^k\right]$
    Sends to devices one bit b_k : 1 with probability $1-\tau$, 0 with with probability τ
  end for
  for each device m in parallel **do**
    $z^{k+1} = z^{k+1/2} - \gamma g^k$
    If $b_k = 1$ then $w^{k+1} = z^k$, sends $F_m(w^{k+1})$ to server and gets $F(w^{k+1})$
    else $w^{k+1} = w^k$
  end for
end for

Convergence MASHA1

- Convergence of MASHA1 in transmitted information:

$$\mathcal{O}([1 + \sqrt{\frac{1}{M} + \frac{1}{\beta}} \cdot \frac{L}{\mu}] \log \frac{1}{\varepsilon});$$

- Extragradient without quantization:

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

- Quantization gives boost.

Compression the same?

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

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

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

Compression the same?

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

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

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

Compression the same?

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

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

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

Compression the same?

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

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

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Question:** where is her optimum? $(0, 0, 0)$.
- Let the starting point $w_0 = (t, t, t)$ for some $t > 0$. Then the local gradients are:

$$\nabla f_1(w_0) = \frac{t}{2}(-11, 9, 9), \quad \nabla f_2(w_0) = \frac{t}{2}(9, -11, 9), \quad \nabla f_3(w_0) = \frac{t}{2}(9, 9, -11).$$

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

Compression the same?

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

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

  where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ и $c = (2, 2, -3)$.
- **Question:** where is her optimum? $(0, 0, 0)$.
- Let the starting point $w_0 = (t, t, t)$ for some $t > 0$. Then the local gradients are:

$$\nabla f_1(w_0) = \tfrac{t}{2}(-11, 9, 9),\quad \nabla f_2(w_0) = \tfrac{t}{2}(9, -11, 9),\quad \nabla f_3(w_0) = \tfrac{t}{2}(9, 9, -11).$$

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

$$w_1 = (t, t, t) + \gamma \cdot \frac{11}{6}(t, t, t) = \left(1 + \frac{11\gamma}{6}\right) w_0.$$

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

Error compensation

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

$$e_{1,m} = 0 + \gamma F_m(z_0) - C(0 + \gamma F_m(z_0)).$$

- And add this to future parcels:

$$C(e_{1,m} + \gamma F_m(z_1))$$

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

$$\text{Parcel:} C(e_{k,m} + \gamma F_m(w_k)),$$
$$e_{k+1,m} = e_{k,m} + \gamma F_m(z_k) - C(e_{k,m} + \gamma F_m(z_k))$$

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

MASHA2 для компрессий

Algorithm 2 MASHA2

Parameters: Stepsize $\gamma > 0$, parameter τ, number of iterations K.
Initialization: Choose $z^0 = w^0 \in \mathcal{Z}$, $e_m^0 = 0$, $e^0 = 0$.
Devices send $F_m(w^0)$ to server and get $F(w^0)$
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for each device m in parallel **do**
    $z^{k+1/2} = \tau z^k + (1-\tau)w^k - \gamma F(w^k)$
    Sends $g_m^k = C_m^{\text{dev}}(\gamma F_m(z^{k+1/2}) - \gamma F_m(w^k) + e_m^k)$ to server
    $e_m^{k+1} = e_m^k + \gamma F_m(z^{k+1/2}) - \gamma F_m(w^k) - g_m^k$
  end for
  for server **do**
    Sends to devices $g^k = C^{\text{serv}}\left[\frac{1}{M}\sum_{m=1}^M g_m^k + e^k\right]$
    $e^{k+1} = e^k + \frac{1}{M}\sum_{m=1}^M g_m^k - g^k$
    Sends to devices one bit b_k : 1 with probability $1-\tau$, 0 with with probability τ
  end for
  for each device m in parallel **do**
    $z^{k+1} = z^{k+1/2} - \gamma g^k$
    If $b_k = 1$ then $w^{k+1} = z^k$, sends $F_m(w^{k+1})$ to server and gets $F(w^{k+1})$
    else $w^{k+1} = w^k$
  end for
end for

Convergence MASHA2

- Convergence of MASHA2 in transmitted information:

$$\mathcal{O}([1 + \frac{L}{\mu}] \log \frac{1}{\varepsilon});$$

- Convergence of MASHA1 in transmitted information:

$$\mathcal{O}([1 + \sqrt{\frac{1}{M} + \frac{1}{\beta}} \cdot \frac{L}{\mu}] \log \frac{1}{\varepsilon});$$

- Extragradient without quantization:

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

- Compression does not give boost in theory.

Convergence MASHA2

- Convergence of MASHA2 in transmitted information:

$$\mathcal{O}([1 + \frac{L}{\mu}] \log \frac{1}{\varepsilon});$$

- Convergence of MASHA1 in transmitted information:

$$\mathcal{O}([1 + \sqrt{\frac{1}{M} + \frac{1}{\beta}} \cdot \frac{L}{\mu}] \log \frac{1}{\varepsilon});$$

- Extragradient without quantization:

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

- Compression does not give boost in theory.

Similarity

- The operators $\{F_m\}$ is δ-related in mean. It means that for any j operators $\{F_m - F_j\}$ is δ-Lipschitz continuous in mean, i.e. for all u, v we have

$$\frac{1}{M}\sum_{m=1}^{M}\|F_m(u) - F_j(u) - F_m(v) + F_j(v)\|^2 \leq \delta^2\|u - v\|^2.$$

- Comes from hessian (second derivatives similarity):

$$\|\nabla^2 f_m(z) - \nabla^2 f_j(z)\| \leq \delta$$

- Natural assumption since from Hoeffding: $\delta = \tilde{\mathcal{O}}(L/\sqrt{b})$ or even $\delta = \tilde{\mathcal{O}}(L/b)$, where b is the number of local data points on each of the devices.

Method for data similarity

- Mirror descent for minimization problems:

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

where $V(x, y)$ is the Bregman divergence 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 data similarity

- Mirror descent for minimization problems:

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

where $V(x, y)$ is the Bregman divergence 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.

- What is the number of communications that occur in K iterations of such a mirror descent? K of communications (the number of gradient counts ∇f), computing arg min requires only computations on the server.

Convergence for data similarity: theorem

Theorem (convergence for data similarity)

Let f be strongly convex, f_i be convex, and ℓ be smooth, and $\varphi(w) = f_1(w) + \delta\|w\|^2$, then mirror descent with step $\gamma = 1$ converges and is satisfied:

$$V(w^*, w_K) \leq \left(1 - \frac{\mu}{\mu + 2\delta}\right)^K V(w^*, w_0).$$

Convergence for data similarity: theorem

Theorem (convergence for data similarity)

Let f be strongly convex, f_i be convex, and ℓ be smooth, and $\varphi(w) = f_1(w) + \delta\|w\|^2$, then mirror descent with step $\gamma = 1$ converges and is satisfied:

$$V(w^*, w_K) \leq \left(1 - \frac{\mu}{\mu + 2\delta}\right)^K V(w^*, w_0).$$

- It means that if we need to achieve an accuracy ε ($V(w^*, w_K) \sim \varepsilon$), then we need to

$$K = \left(\left[1 + \frac{\delta}{\mu}\right] \log \frac{V(w^*, w_0)}{\varepsilon}\right) \text{ communications.}$$

Similarity + compression

- Double kill of two ideas

Algorithm 1 `Three Pillars Algorithm`

Parameters: stepsizes γ and η, momentum τ, probability $p \in (0;1]$, number of local steps H;
Initialization: Choose $z^0 = m^0 = (x^0, y^0) \in \mathcal{Z}$;

for $k = 0, 1, \ldots, K-1$ **do**
  Server takes $u_0^k = z^k$;
  for $t = 0, 1, \ldots, H-1$ **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