Stochastic variance reduction for variational inequalities methods

Yura Malitsky

September 13, 2021

One World Optimization Seminar

LINKÖPING UNIVERSITY

Intro

- variance reduction in minimization
- variational inequalities (VIs)
- variance reduction in VIs

Joint work with: Ahmet Alacaoglu,
University of Wisconsin-Madison

Reference: arxiv:2102.08352
Support: WASP Program

Empirical risk minimization

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

Assumption: f is convex and smooth. Both d and N can be large

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$

Empirical risk minimization

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

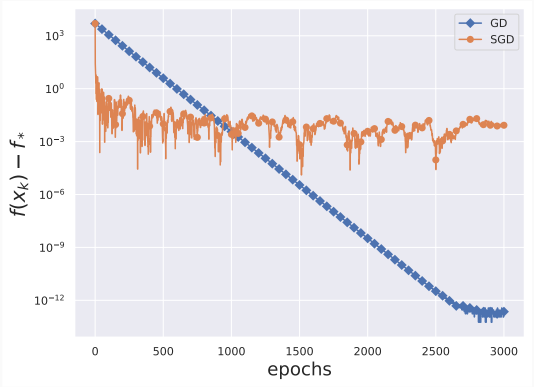
Assumption: f is convex and smooth. Both d and N can be large

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$

- fast, expensive iteration

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$

- slow, cheap iteration



Empirical risk minimization

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

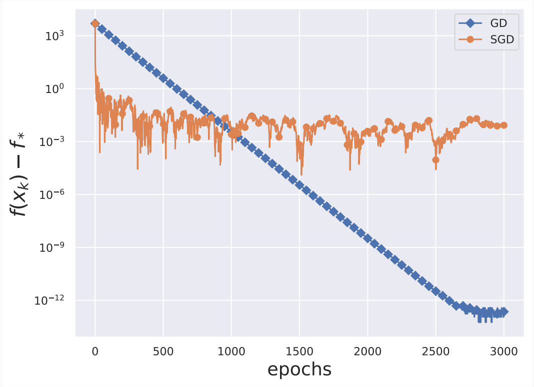
Assumption: f is convex and smooth. Both d and N can be large

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$

- fast, expensive iteration

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$

- slow, cheap iteration



Empirical risk minimization

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

Assumption: f is convex and smooth. Both d and N can be large

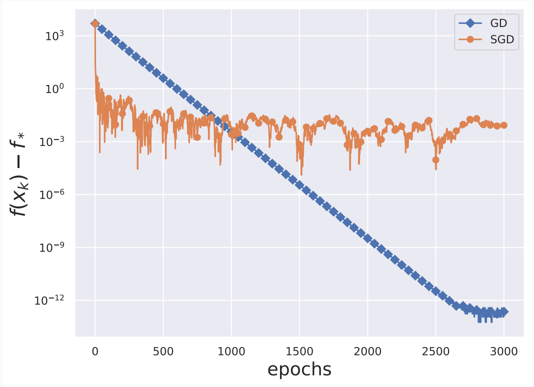
GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$

- fast, expensive iteration

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$

- slow, cheap iteration

We want the best of both worlds



Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Because $\nabla f_{\xi_k}(x_k) \neq \nabla f(x_k) \implies \mathrm{Var}[\nabla f_{\xi_k}(x_k)] = \mathbb{E}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|^2\right] \neq 0$

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Because $\nabla f_{\xi_k}(x_k) \neq \nabla f(x_k) \implies \mathrm{Var}[\nabla f_{\xi_k}(x_k)] = \mathbb{E}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|^2\right] \neq 0$

Goal: decrease variance.

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Because $\nabla f_{\xi_k}(x_k) \neq \nabla f(x_k) \implies \mathrm{Var}[\nabla f_{\xi_k}(x_k)] = \mathbb{E}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|^2\right] \neq 0$

Goal: decrease variance.

Minibatch-SGD:

$$\text{Sample } \mathcal{S}_k \subset \{1, 2, \dots, N\}$$

$$x_{k+1} = x_k - \frac{\tau}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k)$$

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Because $\nabla f_{\xi_k}(x_k) \neq \nabla f(x_k) \implies \mathrm{Var}[\nabla f_{\xi_k}(x_k)] = \mathbb{E}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|^2\right] \neq 0$

Goal: decrease variance.

Minibatch-SGD:

$$\text{Sample } \mathcal{S}_k \subset \{1, 2, \dots, N\}$$

$$x_{k+1} = x_k - \frac{\tau}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k)$$

Idea: increase batch size $|\mathcal{S}_k|$ $\leftarrow$ closer to GD update

Naïve idea

SGD: $x_{k+1} = x_k - \tau_k \nabla f_{\xi_k}(x_k)$ Convergence rate: $\mathcal{O}(\frac{1}{\varepsilon^2})$

Why is it slow?

Because $\nabla f_{\xi_k}(x_k) \neq \nabla f(x_k) \implies \mathrm{Var}[\nabla f_{\xi_k}(x_k)] = \mathbb{E}\left[\|\nabla f_{\xi_k}(x_k) - \nabla f(x_k)\|^2\right] \neq 0$

Goal: decrease variance.

Minibatch-SGD:

$$\text{Sample } \mathcal{S}_k \subset \{1, 2, \dots, N\}$$
$$x_{k+1} = x_k - \frac{\tau}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k)$$

Idea: increase batch size $|\mathcal{S}_k|$ $\leftarrow$ closer to GD update

Good for convergence, not good for complexity.

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

Good idea

Basic update: $$x_{k+1} = x_k - \tau_k g_k$$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$

Good idea

Basic update: $$x_{k+1} = x_k - \tau_k g_k$$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$ $\longleftarrow$ expensive

Good idea

Basic update: $$x_{k+1} = x_k - \tau_k g_k$$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$ $\longleftarrow$ expensive
- If $y = x_{k-2}$ and $x_k - x_{k-2} \to 0 \implies$

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$ $\longleftarrow$ expensive
- If $y = x_{k-2}$ and $x_k - x_{k-2} \to 0 \implies$

$$\nabla f(x_k) - g_k = (\nabla f(x_k) - \nabla f(x_{k-2})) - (\nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x_{k-2})) \to 0$$

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies$ variance $= 0$ $\longleftarrow$ expensive
- If $y = x_{k-2}$ and $x_k - x_{k-2} \to 0 \implies$ $\longleftarrow$ less expensive

$$\nabla f(x_k) - g_k = (\nabla f(x_k) - \nabla f(x_{k-2})) - (\nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x_{k-2})) \to 0$$

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \nabla f(y)$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$ $\longleftarrow$ expensive
- If $y = x_{k-2}$ and $x_k - x_{k-2} \to 0 \implies$ $\longleftarrow$ less expensive

$$\nabla f(x_k) - g_k = (\nabla f(x_k) - \nabla f(x_{k-2})) - (\nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x_{k-2})) \to 0$$

- Can we take $y = x_{k-100}$? $\longleftarrow$ even cheaper

Good idea

Basic update: $x_{k+1} = x_k - \tau_k g_k$

Variance reduction update:

$$g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(y) + \boxed{\nabla f(y)}$$

For any y, vector g_k is unbiased: $\mathbb{E}[g_k] = \nabla f(x_k)$

- Ideally, $y = x_k \implies g_k = \nabla f(x_k) \implies \text{variance} = 0$ $\longleftarrow$ expensive
- If $y = x_{k-2}$ and $x_k - x_{k-2} \to 0 \implies$ $\longleftarrow$ less expensive

$$\nabla f(x_k) - g_k = (\nabla f(x_k) - \nabla f(x_{k-2})) - (\nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x_{k-2})) \to 0$$

- Can we take $y = x_{k-100}$? $\longleftarrow$ even cheaper

Infrequently compute a full gradient $\boxed{\nabla f(y)}$

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \ldots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \ldots, N$ do
4:         Sample $\xi_k \in \{1, \ldots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample  $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

SVRG

Superscript x^s for expensive update with $\nabla f(x^s)$

Subscript x_k for cheap update with $\nabla f_{\xi_k}(x_k)$

Stochastic Variance Reduced Gradient Algorithm:

```
1: for epoch $s = 1, \ldots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \ldots, N$ do
4:         Sample $\xi_k \in \{1, \ldots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

Loopless SVRG

Adding another source of randomness removes one loop

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

Loopless SVRG

Adding another source of randomness removes one loop

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

```
1: for $k = 1, \dots, K$ do
2:     Sample $\xi_k \in \{1, \dots, N\}$.
3:     $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(w_k) + \nabla f(w_k)$
4:     $x_{k+1} = x_k - \tau g_k$
5:     $w_{k+1} = \begin{cases} x_{k+1}, & \text{with probability } \frac{1}{N} \\ w_k, & \text{with probability } 1 - \frac{1}{N} \end{cases}$
6: end for
```

[Kovalev et al., *ALT*, 2020]

Loopless SVRG

Adding another source of randomness removes one loop

```
1: for epoch $s = 1, \ldots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \ldots, N$ do
4:         Sample $\xi_k \in \{1, \ldots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

```
1: for $k = 1, \ldots, K$ do
2:     Sample $\xi_k \in \{1, \ldots, N\}$.
3:     $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(w_k) + \nabla f(w_k)$
4:     $x_{k+1} = x_k - \tau g_k$
5:     $w_{k+1} = \begin{cases} x_{k+1}, & \text{with probability } \frac{1}{N} \\ w_k, & \text{with probability } 1 - \frac{1}{N} \end{cases}$
6: end for
```

[Kovalev et al., *ALT*, 2020]

Loopless SVRG

Adding another source of randomness removes one loop

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

```
1: for $k = 1, \dots, K$ do
2:     Sample $\xi_k \in \{1, \dots, N\}$.
3:     $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(w_k) + \nabla f(w_k)$
4:     $x_{k+1} = x_k - \tau g_k$
5:     $w_{k+1} = \begin{cases} x_{k+1}, & \text{with probability } \frac{1}{N} \\ w_k, & \text{with probability } 1 - \frac{1}{N} \end{cases}$
6: end for
```

[Kovalev et al., *ALT*, 2020]

Loopless SVRG

Adding another source of randomness removes one loop

```
1: for epoch $s = 1, \dots, S$ do
2:     $x_0 = x^s$
3:     for $k = 1, \dots, N$ do
4:         Sample $\xi_k \in \{1, \dots, N\}$
5:         $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(x^s) + \nabla f(x^s)$
6:         $x_{k+1} = x_k - \tau g_k$
7:     end for
8:     $x^{s+1} = x_{N+1}$
9: end for
```

[Johnson et al., *NIPS*, 2013]

```
1: for $k = 1, \dots, K$ do
2:     Sample $\xi_k \in \{1, \dots, N\}$.
3:     $g_k = \nabla f_{\xi_k}(x_k) - \nabla f_{\xi_k}(w_k) + \nabla f(w_k)$
4:     $x_{k+1} = x_k - \tau g_k$
5:     $w_{k+1} = \begin{cases} x_{k+1}, & \text{with probability } \frac{1}{N} \\ w_k, & \text{with probability } 1 - \frac{1}{N} \end{cases}$
6: end for
```

[Kovalev et al., *ALT*, 2020]

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau \left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k)\right)$

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau \left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k)\right)$

Assumption: f_i is convex and L-smooth; cost of ∇f_i is $\mathcal{O}(1)$
f is L_f-smooth

Complexity table

$$\text{Convergence rate} = \quad \#\text{ iterations to reach } \varepsilon\text{-accuracy: } f(x_k) - f_* \leqslant \varepsilon.$$

$$\text{Total complexity} = \quad \text{amount of computation to reach } \varepsilon\text{-accuracy.}$$

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau \left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k)\right)$

Assumption: f_i is convex and L-smooth; cost of ∇f_i is $\mathcal{O}(1)$
f is L_f-smooth

Iteration cost
Convergence rate
Total complexity

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau \left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k)\right)$

Assumption: f_i is convex and L-smooth; cost of ∇f_i is $\mathcal{O}(1)$
f is L_f-smooth

	GD
Iteration cost	$N \times \mathcal{O}(1)$
Convergence rate	$\mathcal{O}\left(\frac{L_f}{\varepsilon}\right)$
Total complexity	$\mathcal{O}\left(\frac{NL_f}{\varepsilon}\right)$

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau\left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k)\right)$

Assumption: f_i is convex and L-smooth; cost of ∇f_i is $\mathcal{O}(1)$
f is L_f-smooth

	GD	**SGD**
Iteration cost	$N \times \mathcal{O}(1)$	$\mathcal{O}(1)$
Convergence rate	$\mathcal{O}\left(\frac{L_f}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$
Total complexity	$\mathcal{O}\left(\frac{NL_f}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$

Complexity table

Convergence rate = # iterations to reach ε-accuracy: $f(x_k) - f_* \leqslant \varepsilon$.

Total complexity = amount of computation to reach ε-accuracy.

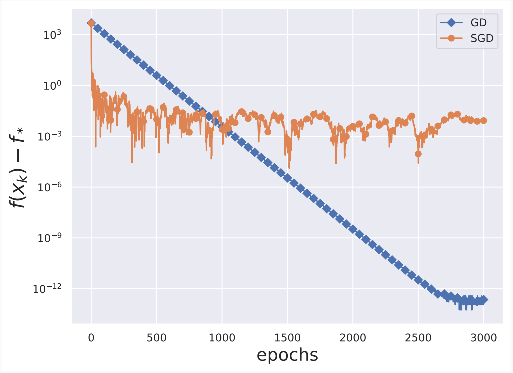
GD: $x_{k+1} = x_k - \tau \nabla f(x_k)$
SGD: $x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$
SVRG: $x_{k+1} = x_k - \tau \left(\nabla f(w_k) + \nabla f_i(x_k) - \nabla f_i(w_k) \right)$

Assumption: f_i is convex and L-smooth; cost of ∇f_i is $\mathcal{O}(1)$
f is L_f-smooth

	GD	**SGD**	**SVRG**
Iteration cost	$N \times \mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Convergence rate	$\mathcal{O}\left(\frac{L_f}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{\sqrt{N}L}{\varepsilon}\right)$
Total complexity	$\mathcal{O}\left(\frac{NL_f}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{\sqrt{N}L}{\varepsilon}\right)$

Take-away 1

- VR takes the best from GD and SGD

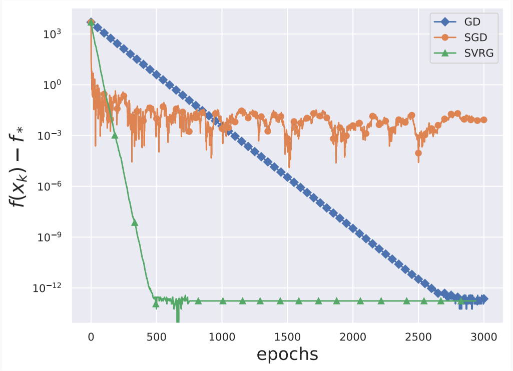


- References:
  - **SAG**: [Schmidt, et al., *NIPS*, 2013]
  - **SVRG:** [Johnson, et al., *NIPS*, 2013]
  - **SAGA:** [Defazio, et al., *NIPS*, 2014]
  - **L-SVRG:** [Kovalev et al., *ALT*, 2020]

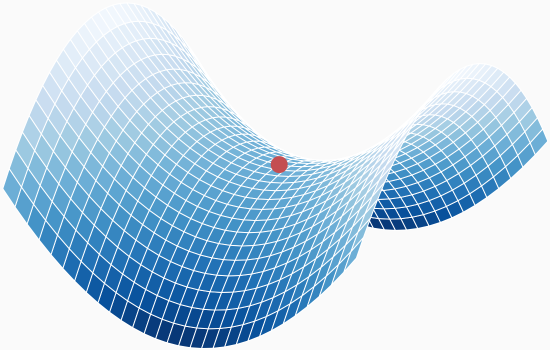
Take-away 1

- VR takes the best from GD and SGD

- References:
  - **SAG**: [Schmidt, et al., *NIPS*, 2013]
  - **SVRG:** [Johnson, et al., *NIPS*, 2013]
  - **SAGA:** [Defazio, et al., *NIPS*, 2014]
  - **L-SVRG:** [Kovalev et al., *ALT*, 2020]



Saddle points



Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y)$$

Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Optimality condition:

$$x: \qquad \langle \nabla_x f(x^*, y^*), x - x^* \rangle \geqslant 0 \quad \forall x \in \mathcal{X}$$

Saddle point problems

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Optimality condition:

$$\begin{aligned} x: &\quad \langle \nabla_x f(x^*, y^*), x - x^* \rangle \geqslant 0 \quad \forall x \in \mathcal{X} \\ y: &\quad \langle -\nabla_y f(x^*, y^*), y - y^* \rangle \geqslant 0 \quad \forall y \in \mathcal{Y} \end{aligned}$$

Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x,y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X},\mathcal{Y}$ are closed convex sets

Optimality condition:

$$\begin{aligned} x: &\qquad \langle \nabla_x f(x^*,y^*), x-x^*\rangle \geqslant 0 \quad \forall x\in\mathcal{X} \\ y: &\qquad \langle -\nabla_y f(x^*,y^*), y-y^*\rangle \geqslant 0 \quad \forall y\in\mathcal{Y} \end{aligned}$$

$$\iff$$

$$\left\langle \begin{pmatrix} \nabla_x f(x^*,y^*) \\ -\nabla_y f(x^*,y^*) \end{pmatrix}, \begin{pmatrix} x-x^* \\ y-y^* \end{pmatrix} \right\rangle \geqslant 0 \quad \forall (x,y)\in\mathcal{X}\times\mathcal{Y}$$

Saddle point problems

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Optimality condition:

$$\begin{aligned} x: \quad & \langle \nabla_x f(x^*, y^*), x - x^* \rangle \geqslant 0 \quad \forall x \in \mathcal{X} \\ y: \quad & \langle -\nabla_y f(x^*, y^*), y - y^* \rangle \geqslant 0 \quad \forall y \in \mathcal{Y} \end{aligned}$$

$$\iff$$

$$\left\langle \underbrace{\begin{pmatrix} \nabla_x f(x^*, y^*) \\ -\nabla_y f(x^*, y^*) \end{pmatrix}}_{F(z^*)}, \underbrace{\begin{pmatrix} x - x^* \\ y - y^* \end{pmatrix}}_{z - z^*} \right\rangle \geqslant 0 \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$$

Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Optimality condition:

$$\begin{aligned} x: &\quad \langle \nabla_x f(x^*, y^*), x - x^* \rangle \geqslant 0 \quad \forall x \in \mathcal{X} \\ y: &\quad \langle -\nabla_y f(x^*, y^*), y - y^* \rangle \geqslant 0 \quad \forall y \in \mathcal{Y} \end{aligned}$$

$$\Longleftrightarrow$$

$$\left\langle \underbrace{\begin{pmatrix} \nabla_x f(x^*, y^*) \\ -\nabla_y f(x^*, y^*) \end{pmatrix}}_{F(z^*)}, \underbrace{\begin{pmatrix} x - x^* \\ y - y^* \end{pmatrix}}_{z - z^*} \right\rangle \geqslant 0 \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$$

Variational inequality:

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

Saddle point problems

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y)$$

Assumption: f is a convex-concave smooth function, $\mathcal{X}, \mathcal{Y}$ are closed convex sets

Optimality condition:

$$\begin{aligned} x: &\quad \langle \nabla_x f(x^*, y^*), x - x^* \rangle \geqslant 0 \quad \forall x \in \mathcal{X} \\ y: &\quad \langle -\nabla_y f(x^*, y^*), y - y^* \rangle \geqslant 0 \quad \forall y \in \mathcal{Y} \end{aligned}$$

$$\Longleftrightarrow$$

$$\Bigg\langle \underbrace{\begin{pmatrix} \nabla_x f(x^*, y^*) \\ -\nabla_y f(x^*, y^*) \end{pmatrix}}_{F(z^*)}, \underbrace{\begin{pmatrix} x - x^* \\ y - y^* \end{pmatrix}}_{z - z^*} \Bigg\rangle \geqslant 0 \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$$

Variational inequality:

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

Assumption: F is monotone: $\langle F(z) - F(z'), z - z' \rangle \geqslant 0 \quad \forall z, z' \in \mathcal{Z}$

Motivation

Motivation

1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$

Motivation

1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0$

Motivation

1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y \geqslant 0} f(x) + y h(x)$

Motivation

1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y \geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax)$

Motivation

1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y \geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax) \quad \longrightarrow \quad \min_x \max_y f(x) + \langle Ax, y \rangle - g^*(y)$

Motivation

1. Matrix games: $\min_{x\in\Delta^n} \max_{y\in\Delta^m} \langle Ax, y\rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y\geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax) \quad \longrightarrow \quad \min_x \max_y f(x) + \langle Ax, y\rangle - g^*(y)$
4. Structural minimization-2: $\min_x \max_{i=1,\dots,m} f_i(x)$

Motivation

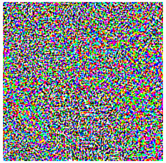
1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y \geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax) \quad \longrightarrow \quad \min_x \max_y f(x) + \langle Ax, y \rangle - g^*(y)$
4. Structural minimization-2: $\min_x \max_{i=1,\dots,m} f_i(x) \quad \longrightarrow \min_x \max_{y \in \Delta^m} \sum y_i f_i(x)$

Motivation

1. Matrix games: $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x)\leqslant 0 \quad \longrightarrow \quad \min_x \max_{y\geqslant 0} f(x)+yh(x)$
3. Structural minimization-1: $\min_x f(x)+g(Ax) \quad \longrightarrow \quad \min_x\max_y f(x)+\langle Ax, y\rangle - g^*(y)$
4. Structural minimization-2: $\min_x \max_{i=1,\dots,m} f_i(x) \quad \longrightarrow \quad \min_x \max_{y\in\Delta^m}\sum y_i f_i(x)$
5. Robustness (adversarial training) [Goodfellow et al., ICLR 2015]

"panda"

$+\ .007\ \times$

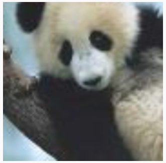

adversarial noise

$=$

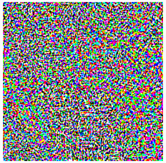
"gibbon"

Motivation

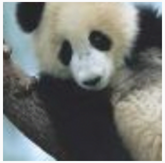
1. Matrix games: $\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle Ax, y \rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x) \leqslant 0 \quad \longrightarrow \quad \min_x \max_{y \geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax) \quad \longrightarrow \quad \min_x \max_y f(x) + \langle Ax, y \rangle - g^*(y)$
4. Structural minimization-2: $\min_x \max_{i=1,\dots,m} f_i(x) \quad \longrightarrow \min_x \max_{y \in \Delta^m} \sum y_i f_i(x)$
5. Robustness (adversarial training) [Goodfellow et al., ICLR 2015]


"panda"

$+ .007 \times$


adversarial noise

$=$


"gibbon"

$$\underbrace{\min_x \sum_i f(x, a_i, b_i)}_{\text{ERM}} \longrightarrow \underbrace{\min_x \max_{\delta \in \mathcal{S}} \sum_i f(x, a_i + \delta, b_i)}_{\text{Robust ERM}}$$

Motivation

1. Matrix games: $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$
2. Constrained optimization: $\min_x f(x) \quad \text{s.t.} \quad h(x)\leqslant 0 \quad \longrightarrow \quad \min_x \max_{y\geqslant 0} f(x) + yh(x)$
3. Structural minimization-1: $\min_x f(x) + g(Ax) \quad \longrightarrow \quad \min_x \max_y f(x) + \langle Ax, y\rangle - g^*(y)$
4. Structural minimization-2: $\min_x \max_{i=1,\dots,m} f_i(x) \quad \longrightarrow \min_x \max_{y\in\Delta^m} \sum y_i f_i(x)$
5. Robustness (adversarial training)

$$\underbrace{\min_x \sum_i f(x, a_i, b_i)}_{\text{ERM}} \quad \longrightarrow \quad \underbrace{\min_x \max_{\delta\in\mathcal{S}} \sum_i f(x, a_i+\delta, b_i)}_{\text{Robust ERM}}$$

6. Generative adversarial networks, GANs.
   Two player game between two neural networks.

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

Example: gradient descent-ascent

$$x_{k+1} = x_k - \tau \nabla_x f(x_k, y_k)$$

$$y_{k+1} = y_k + \tau \nabla_y f(x_k, y_k)$$

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

Example: gradient descent-ascent

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - \tau \begin{bmatrix} \nabla_x f(x_k, y_k) \\ -\nabla_y f(x_k, y_k) \end{bmatrix}$$

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

Example: gradient descent-ascent

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix}}_{z_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \end{bmatrix}}_{z_k} - \tau \underbrace{\begin{bmatrix} \nabla_x f(x_k, y_k) \\ -\nabla_y f(x_k, y_k) \end{bmatrix}}_{F(z_k)}$$

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

Example: gradient descent-ascent

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix}}_{z_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \end{bmatrix}}_{z_k} - \tau \underbrace{\begin{bmatrix} \nabla_x f(x_k, y_k) \\ -\nabla_y f(x_k, y_k) \end{bmatrix}}_{F(z_k)}$$

$$z_{k+1} = z_k - \tau F(z_k)$$

What doesn't work

$$\min_x \max_y f(x, y)$$

Assumption: f is a convex-concave smooth function

Harder than basic minimization $\min_x f(x)$: worse rates, more complicated algorithms, etc.

Example: gradient descent-ascent

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix}}_{z_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \end{bmatrix}}_{z_k} - \tau \underbrace{\begin{bmatrix} \nabla_x f(x_k, y_k) \\ -\nabla_y f(x_k, y_k) \end{bmatrix}}_{F(z_k)}$$

$$z_{k+1} = z_k - \tau F(z_k) \qquad \longleftarrow \quad \text{doesn't work}$$

What works

$$\langle F(z^*), z - z^* \rangle \geqslant 0 \quad \forall z \in \mathcal{Z}$$

Assumption: F is monotone and L_F-Lipschitz

What works

$$\langle F(z^*), z - z^* \rangle \geqslant 0 \quad \forall z \in \mathcal{Z}$$

Assumption: F is monotone and L_F-Lipschitz

Extragradient method: [Korpelevich, 1976]

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(z_k - \tau F(z_k)) \\ z_{k+1} &= P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2})) \end{aligned}$$

What works

$$\langle F(z^*), z - z^* \rangle \geqslant 0 \quad \forall z \in \mathcal{Z}$$

Assumption: F is monotone and L_F-Lipschitz

Extragradient method: [Korpelevich, 1976]

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(z_k - \tau F(z_k)) \\ z_{k+1} &= P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2})) \end{aligned}$$

Def. $w \in \mathcal{Z}$ is ε-solution if $\mathrm{Gap}(w) \leqslant \varepsilon$, where $\mathrm{Gap}(w) = \max_{z \in \mathcal{C}} \langle F(z), w - z \rangle$

What works

$$\langle F(z^*), z - z^* \rangle \geqslant 0 \quad \forall z \in \mathcal{Z}$$

Assumption: F is monotone and L_F-Lipschitz

Extragradient method: [Korpelevich, 1976]

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(z_k - \tau F(z_k)) \\ z_{k+1} &= P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2})) \end{aligned}$$

Def. $w \in \mathcal{Z}$ is ε-solution if $\mathrm{Gap}(w) \leqslant \varepsilon$, where $\mathrm{Gap}(w) = \max_{z \in \mathcal{C}} \langle F(z), w - z \rangle$

Theorem

Let $\tau \in \left(0, \frac{1}{L_F}\right)$. *Then* $z_k \to z_* \in \mathrm{Sol}$ *and* $\mathrm{Gap}(z^K) = \mathcal{O}\left(\frac{L_F}{K}\right)$.

(z^K) is the average of $(z_{k+1/2})$

Finite-sum structure

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) = f_1(x, y) + \cdots + f_N(x, y)$$

Finite-sum structure

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) = f_1(x, y) + \cdots + f_N(x, y)$$

$$F(z) = \underbrace{\begin{bmatrix} \nabla_x f_1(x, y) \\ -\nabla_y f_1(x, y) \end{bmatrix}}_{F_1(z)} + \cdots + \underbrace{\begin{bmatrix} \nabla_x f_N(x, y) \\ -\nabla_y f_N(x, y) \end{bmatrix}}_{F_N(z)}$$

Finite-sum structure

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) = f_1(x, y) + \cdots + f_N(x, y)$$

$$F(z) = \underbrace{\begin{bmatrix} \nabla_x f_1(x, y) \\ -\nabla_y f_1(x, y) \end{bmatrix}}_{F_1(z)} + \cdots + \underbrace{\begin{bmatrix} \nabla_x f_N(x, y) \\ -\nabla_y f_N(x, y) \end{bmatrix}}_{F_N(z)}$$

$$= F_1(z) + \cdots + F_N(z)$$

Finite-sum structure

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x,y) = f_1(x,y) + \cdots + f_N(x,y)$$

$$F(z) = \underbrace{\begin{bmatrix} \nabla_x f_1(x,y) \\ -\nabla_y f_1(x,y) \end{bmatrix}}_{F_1(z)} + \cdots + \underbrace{\begin{bmatrix} \nabla_x f_N(x,y) \\ -\nabla_y f_N(x,y) \end{bmatrix}}_{F_N(z)}$$

$$= F_1(z) + \cdots + F_N(z)$$

State of the art:

[Balamurugan, Bach, *NIPS*, 2016]: Strongly-convex-strongly-concave case is covered.

[Iusem et al., *SIOPT*, 2017]: With increasing minibatch size.

[Carmon, et al., *NeurIPS*, 2019]: Matrix games, generic case requires unnatural assumptions.

Proposed algorithm

Proposed algorithm

```
1: **Input:** Probability $p \in (0,1]$, step size $\tau$, $\alpha \in (0,1)$, and $z_0 = w_0$
2: **for** $k = 0, 1, \dots$ **do**
3:     $\bar{z}_k = \alpha z_k + (1-\alpha) w_k$
4:     Sample $\xi_k \in \{1, \dots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1-p \end{cases}$
8: **end for**
```

Proposed algorithm

$$z_{k+1/2} = P_{\mathcal{Z}}(z_k - \tau F(z_k))$$
$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2}))$$

```
1: **Input:** Probability $p \in (0,1]$, step size $\tau$, $\alpha \in (0,1)$, and $z_0 = w_0$
2: **for** $k = 0, 1, \ldots$ **do**
3:     $\bar{z}_k = \alpha z_k + (1-\alpha) w_k$
4:     Sample $\xi_k \in \{1, \ldots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1-p \end{cases}$
8: **end for**
```

Proposed algorithm

$$z_{k+1/2} = P_{\mathcal{Z}}(z_k - \tau F(z_k))$$
$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2}))$$

```
1: **Input:** Probability $p \in (0,1]$, step size $\tau$, $\alpha \in (0,1)$, and $z_0 = w_0$
2: **for** $k = 0, 1, \dots$ **do**
3:     $\bar{z}_k = \alpha z_k + (1-\alpha) w_k$
4:     Sample $\xi_k \in \{1, \dots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1-p \end{cases}$
8: **end for**
```

Proposed algorithm

$$z_{k+1/2} = P_{\mathcal{Z}}(z_k - \tau F(z_k))$$
$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2}))$$

```
1: Input: Probability $p \in (0,1]$, step size $\tau$, $\alpha \in (0,1)$, and $z_0 = w_0$
2: for $k = 0, 1, \ldots$ do
3:     $\bar{z}_k = \alpha z_k + (1-\alpha) w_k$
4:     Sample $\xi_k \in \{1, \ldots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1-p \end{cases}$
8: end for
```

Proposed algorithm

$$z_{k+1/2} = P_{\mathcal{Z}}(z_k - \tau F(z_k))$$
$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2}))$$

```
1: Input: Probability $p \in (0, 1]$, step size $\tau$, $\alpha \in (0, 1)$, and $z_0 = w_0$
2: for $k = 0, 1, \ldots$ do
3:     $\bar{z}_k = \alpha z_k + (1 - \alpha) w_k$
4:     Sample $\xi_k \in \{1, \ldots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1 - p \end{cases}$
8: end for
```

Proposed algorithm

$$z_{k+1/2} = P_{\mathcal{Z}}(z_k - \tau F(z_k))$$
$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau F(z_{k+1/2}))$$

```
1: Input: Probability $p \in (0, 1]$, step size $\tau$, $\alpha \in (0, 1)$, and $z_0 = w_0$
2: for $k = 0, 1, \ldots$ do
3:     $\bar{z}_k = \alpha z_k + (1 - \alpha) w_k$
4:     Sample $\xi_k \in \{1, \ldots, N\}$
5:     $z_{k+1/2} = P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k))$
6:     $z_{k+1} = P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)])$
7:     $w_{k+1} = \begin{cases} z_{k+1}, & \text{with probability } p \\ w_k, & \text{with probability } 1 - p \end{cases}$
8: end for
```

When there is no randomness, i.e., $p = 1$ and $F_\xi = F \implies w_{k+1} = z_{k+1} \implies \bar{z}_k = z_k$

Analysis

$$\begin{aligned} \mathsf{z}_{k+1/2} &= P_{\mathcal{Z}}(\bar{\mathsf{z}}_k - \tau F(\mathsf{w}_k)) \\ \mathsf{z}_{k+1} &= P_{\mathcal{Z}}(\bar{\mathsf{z}}_k - \tau[F(\mathsf{w}_k) + F_{\xi_k}(\mathsf{z}_{k+1/2}) - F_{\xi_k}(\mathsf{w}_k)]) \end{aligned}$$

Analysis

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k)) \\ z_{k+1} &= P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)]) \end{aligned}$$

- F is monotone
- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2\|z - z'\|^2$

Analysis

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(\bar{z}_k - \tau F(w_k)) \\ z_{k+1} &= P_{\mathcal{Z}}(\bar{z}_k - \tau[F(w_k) + F_{\xi_k}(z_{k+1/2}) - F_{\xi_k}(w_k)]) \end{aligned}$$

- F is monotone
- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2\|z - z'\|^2$

Theorem

Let $p \in (0,1]$, $\alpha = 1 - p$, *and* $\tau \in \left(0, \frac{\sqrt{p}}{L}\right)$. *Then,* $z_k \to z_* \in \mathrm{Sol}$ *a.s. and*

$$\mathbb{E}\left[\mathrm{Gap}(z^K)\right] = \mathcal{O}\left(\frac{L}{\sqrt{p}K}\right).$$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right)$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times \mathrm{cost}(F) = \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times \mathrm{cost}(F) = \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$$

Stochastic setting

- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2 \|z - z'\|^2$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times \mathrm{cost}(F) = \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$$

Stochastic setting

- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2 \|z - z'\|^2$

Convergence rate: $\mathcal{O}\left(\frac{L}{\sqrt{p}\varepsilon}\right)$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times \mathrm{cost}(F) = \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$$

Stochastic setting

- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2 \|z - z'\|^2$

Convergence rate: $\mathcal{O}\left(\frac{L}{\sqrt{p}\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L}{\sqrt{p}\varepsilon}\right) \times (pN + 2)\mathrm{cost}(F_\xi) = \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$$

Complexity

Deterministic setting

- F is L_F-Lipschitz: $\|F(z) - F(z')\| \leqslant L_F \|z - z'\|$

Convergence rate: $\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times \mathrm{cost}(F) = \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$$

Stochastic setting

- Unbiasedness: $\mathbb{E}\left[F_\xi(z)\right] = F(z)$
- Lipschitzness: $\mathbb{E}\left[\|F_\xi(z) - F_\xi(z')\|^2\right] \leqslant L^2 \|z - z'\|^2$

Convergence rate: $\mathcal{O}\left(\frac{L}{\sqrt{p}\varepsilon}\right) \implies$ Complexity:

$$\mathcal{O}\left(\frac{L}{\sqrt{p}\varepsilon}\right) \times (pN + 2)\mathrm{cost}(F_\xi) = \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$$

The same improvement as in the minimization case!

How big is improvement?

$$F(z) = F_1(z) + \cdots + F_N(z), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

How big is improvement?

$$F(\mathrm{z}) = F_1(\mathrm{z}) + \cdots + F_N(\mathrm{z}), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\implies \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\implies \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$.

How big is improvement?

$$F(z) = F_1(z) + \cdots + F_N(z), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\implies \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\implies \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$. What is good $F_\xi(z)$?

Example:

How big is improvement?

$$F(\mathrm{z}) = F_1(\mathrm{z}) + \cdots + F_N(\mathrm{z}), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$. What is good $F_\xi(\mathrm{z})$?

Example:

- Uniform sampling: $F_\xi(\mathrm{z}) = N \cdot F_i(\mathrm{z})$, where $\xi = i$ with prob $p_i = \frac{1}{N}$

How big is improvement?

$$F(z) = F_1(z) + \cdots + F_N(z), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$. What is good $F_\xi(z)$?

Example:

- Uniform sampling: $F_\xi(z) = N \cdot F_i(z)$, where $\xi = i$ with prob $p_i = \frac{1}{N}$

$$\Longrightarrow \qquad L = N \max_i L_i$$

How big is improvement?

$$F(z) = F_1(z) + \cdots + F_N(z), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$. What is good $F_\xi(z)$?

Example:

- Uniform sampling: $F_\xi(z) = N \cdot F_i(z)$, where $\xi = i$ with prob $p_i = \frac{1}{N}$
$$\Longrightarrow \qquad L = N \max_i L_i$$
- Weighted (importance) sampling: $F_\xi(z) = \frac{1}{p_i} F_i(z)$, where $\xi = i$ with prob $p_i = \frac{L_i}{\sum L_j}$

How big is improvement?

$$F(z) = F_1(z) + \cdots + F_N(z), \qquad \text{each } F_i \text{ is } L_i\text{-Lipschitz}$$

Deterministic: F is L_F-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L_F}{\varepsilon}\right) \times N \times \mathrm{cost}(F_\xi)$

Stochastic VR: F_ξ is L-Lipschitz $\Longrightarrow \mathcal{O}\left(\frac{L}{\varepsilon}\right) \times \sqrt{N} \times \mathrm{cost}(F_\xi)$

Unfortunately, $L_F \leqslant L$. What is good $F_\xi(z)$?

Example:

- Uniform sampling: $F_\xi(z) = N \cdot F_i(z)$, where $\xi = i$ with prob $p_i = \frac{1}{N}$
$$\Longrightarrow \qquad L = N \max_i L_i$$
- Weighted (importance) sampling: $F_\xi(z) = \frac{1}{p_i} F_i(z)$, where $\xi = i$ with prob $p_i = \frac{L_i}{\sum L_j}$
$$\Longrightarrow \qquad L = L_1 + \cdots + L_N$$

Generalizations

1. Bregman setting (require two loops)

Generalizations

1. Bregman setting (require two loops)
2. General VIs: $\langle F(z^*), z - z^* \rangle + g(z) - g(z^*) \geqslant 0$: $P_{\mathcal{Z}} \longrightarrow \operatorname{prox}_g$

Generalizations

1. Bregman setting (require two loops)
2. General VIs: $\langle F(z^*), z - z^* \rangle + g(z) - g(z^*) \geqslant 0$: $P_{\mathcal{Z}} \longrightarrow \mathrm{prox}_g$
3. Another algorithms:

Generalizations

1. Bregman setting (require two loops)
2. General VIs: $\langle F(z^*), z - z^* \rangle + g(z) - g(z^*) \geqslant 0$: $P_{\mathcal{Z}} \longrightarrow \mathrm{prox}_g$
3. Another algorithms:
   - Forward-backward-forward

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(z_k - \tau F(z_k)) \\ z_{k+1} &= z_{k+1/2} - \tau[F(z_{k+1/2}) - F(z_k)] \end{aligned}$$

Generalizations

1. Bregman setting (require two loops)
2. General VIs: $\langle F(z^*), z - z^* \rangle + g(z) - g(z^*) \geqslant 0$: $\quad P_{\mathcal{Z}} \longrightarrow \mathrm{prox}_g$
3. Another algorithms:
    - Forward-backward-forward

$$\begin{aligned} z_{k+1/2} &= P_{\mathcal{Z}}(z_k - \tau F(z_k)) \\ z_{k+1} &= z_{k+1/2} - \tau[F(z_{k+1/2}) - F(z_k)] \end{aligned}$$

    - Forward-reflected-backward

$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau[2F(z_k) - F(z_{k-1})])$$

Subtleties

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[z^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(z^K)\right]$

Subtleties

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[\mathsf{z}^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(\mathsf{z}^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

Subtleties

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[z^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(z^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry
(ii) Variable distributions

Subtleties

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[\mathsf{z}^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(\mathsf{z}^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry

(ii) Variable distributions

$$F(\mathsf{z}) = \begin{pmatrix} A^\top \mathsf{y} \\ -A\mathsf{x} \end{pmatrix}$$

$$A\mathsf{x} = x_1 A_{:1} + \cdots + x_n A_{:n} \implies \text{stochastic oracle } A_\xi \text{ is easy}$$

Subtleties

$$\mathrm{z}_{k+1} = P_{\mathcal{Z}}(\mathrm{z}_k - \tau[F(\mathrm{w}_k) + \underbrace{F_\xi(\mathrm{z}_{k+1/2}) - F_\xi(\mathrm{w}_k)}_{\text{linear}}])$$

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[\mathrm{z}^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(\mathrm{z}^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry

(ii) Variable distributions

$$F(\mathrm{z}) = \begin{pmatrix} A^\top \mathrm{y} \\ -A\mathrm{x} \end{pmatrix}$$

$A\mathrm{x} = x_1 A_{:1} + \cdots + x_n A_{:n} \implies$ stochastic oracle A_ξ is easy

But we need it for $F_\xi(\mathrm{z}_{k+1/2} - \mathrm{w}_k)$

Subtleties

$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau[F(w_k) + \underbrace{F_\xi(z_{k+1/2}) - F_\xi(w_k)}_{\text{linear}}])$$

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[z^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(z^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry
(ii) Variable distributions

$$F(z) = \begin{pmatrix} A^\top y \\ -Ax \end{pmatrix}$$

$Ax = x_1 A_{:1} + \cdots + x_n A_{:n} \implies$ stochastic oracle A_ξ is easy
But we need it for $F_\xi(z_{k+1/2} - w_k) \quad \implies$

$$A_\xi(x - x') = \frac{1}{p_j} A_{:j}(x_j - x_j'), \qquad p_j = \mathrm{Prob}\{\xi = j\} = \frac{|x_j - x_j'|}{\|x - x'\|_1}$$

Subtleties

$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau[F(w_k) + \underbrace{F_\xi(z_{k+1/2}) - F_\xi(w_k)}_{\text{linear}}))$$

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[z^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(z^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry
(ii) Variable distributions

$$F(z) = \begin{pmatrix} A^\top y \\ -Ax \end{pmatrix}$$

$Ax = x_1 A_{:1} + \cdots + x_n A_{:n} \implies$ stochastic oracle A_ξ is easy
But we need it for $F_\xi(z_{k+1/2} - w_k) \qquad \implies$

$$A_\xi(x - x') = \frac{1}{p_j} A_{:j}(x_j - x_j'), \qquad p_j = \mathrm{Prob}\{\xi = j\} = \frac{|x_j - x_j'|}{\|x - x'\|_1}$$

x, x' are parts of $z_{k+1/2}, w_k \implies$ every iteration we have to change the distribution!

Subtleties

$$z_{k+1} = P_{\mathcal{Z}}(z_k - \tau[F(w_k) + \underbrace{F_\xi(z_{k+1/2}) - F_\xi(w_k))}_{\text{linear}}$$

- Easy to get rate for $\mathrm{Gap}(\mathbb{E}\left[z^K\right])$, harder for $\mathbb{E}\left[\mathrm{Gap}(z^K)\right]$
- Good complexity for matrix games $\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax, y\rangle$ require

(i) Bregman case, ℓ_1 geometry
(ii) Variable distributions

$$F(z) = \begin{pmatrix} A^\top y \\ -Ax \end{pmatrix}$$

$Ax = x_1 A_{:1} + \cdots + x_n A_{:n} \implies$ stochastic oracle A_ξ is easy
But we need it for $F_\xi(z_{k+1/2} - w_k) \qquad \implies$

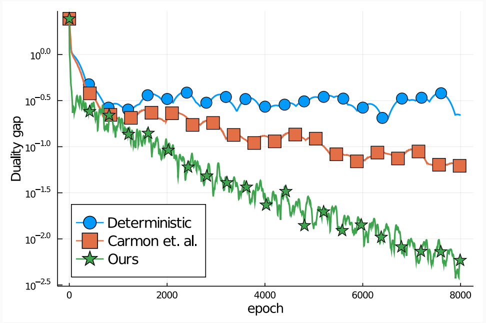
$$A_\xi(x - x') = \frac{1}{p_j} A_{:j}(x_j - x_j'), \qquad p_j = \mathrm{Prob}\{\xi = j\} = \frac{|x_j - x_j'|}{\|x - x'\|_1}$$

x, x' are parts of $z_{k+1/2}, w_k \implies$ every iteration we have to change the distribution!

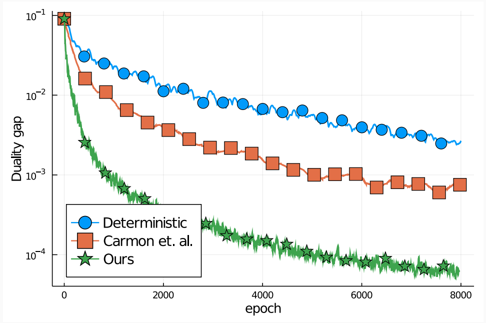
[Grigoriadis & Khachiyan, 1995], [Carmon et al. 2019]

Illustration

$$\min_{x\in\Delta^n}\max_{y\in\Delta^n}\langle Ax, y\rangle$$



'policeman and robber' problem, $n = 500$



random iid matrix from $[0,1]^{n\times n}$, $n = 500$

Take-away 2

- variance reduction is possible for saddle point problems/VIs (without any extra assumptions)
- good complexity requires good stochastic oracles

Take-away 2

- variance reduction is possible for saddle point problems/VIs (without any extra assumptions)
- good complexity requires good stochastic oracles

Open question: good lower bounds are unknown. Can we go beyond $\sqrt{N}$ improvement?

Take-away 2

- variance reduction is possible for saddle point problems/VIs (without any extra assumptions)
- good complexity requires good stochastic oracles

~~**Open question:** *good lower bounds are unknown. Can we go beyond $\sqrt{N}$ improvement?*~~

Y. Han et al. *"Lower Complexity Bounds of Finite-Sum Optimization Problems: The Results and Construction"* arxiv:2103.08280