On Gradient-Based Optimization: Accelerated, Stochastic, Asynchronous, Distributed

Michael I. Jordan
University of California, Berkeley

May 2, 2017

Modern Optimization and Large-Scale Data Analysis

- A need to exploit parallelism,
- while controlling stochasticity,
- and tolerating asynchrony,
- and trying to go as fast as the oracle allows, and maybe even faster

Outline

- Variational, Hamiltonian and symplectic perspectives on Nesterov acceleration
- Avoiding saddle points, efficiently
- Stochastically-controlled stochastic gradient
- A “perturbed iterate” framework for analysis of asynchronous algorithms
- Primal-dual distributed optimization, commoditized
- Ray, a next-generation platform for ML workloads

Variational, Hamiltonian and Symplectic Perspectives on Acceleration

Michael I. Jordan
University of California, Berkeley

April 10, 2017

with Andre Wibisono, Ashia Wilson and Michael Betancourt

Accelerated gradient descent

Setting: Unconstrained convex optimization

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

- Classical gradient descent:

$$x_{k+1} = x_k - \beta \nabla f(x_k)$$

obtains a convergence rate of $O(1/k)$

- Accelerated gradient descent:

$$\begin{aligned} y_{k+1} &= x_k - \beta \nabla f(x_k) \\ x_{k+1} &= (1 - \lambda_k) y_{k+1} + \lambda_k y_k \end{aligned}$$

obtains the (optimal) convergence rate of $O(1/k^2)$

The acceleration phenomenon

Two classes of algorithms:

- **Gradient methods**
  - Gradient descent, mirror descent, cubic-regularized Newton's method (Nesterov and Polyak '06), etc.
  - Greedy descent methods, relatively well-understood
- **Accelerated methods**
  - Nesterov's accelerated gradient descent, accelerated mirror descent, accelerated cubic-regularized Newton's method (Nesterov '08), etc.
  - Important for both theory (optimal rate for first-order methods) and practice (many extensions: FISTA, stochastic setting, etc.)
  - *Not* descent methods, faster than gradient methods, still mysterious

Accelerated methods: Continuous time perspective

- Gradient descent is discretization of gradient flow

$$\dot{X}_t = -\nabla f(X_t)$$

(and mirror descent is discretization of natural gradient flow)

Accelerated methods: Continuous time perspective

- Gradient descent is discretization of gradient flow

$$\dot{X}_t = -\nabla f(X_t)$$

(and mirror descent is discretization of natural gradient flow)

- Su, Boyd, Candes '14: Continuous time limit of accelerated gradient descent is a second-order ODE

$$\ddot{X}_t + \frac{3}{t}\dot{X}_t + \nabla f(X_t) = 0$$

Accelerated methods: Continuous time perspective

- Gradient descent is discretization of gradient flow

$$\dot{X}_t = -\nabla f(X_t)$$

  (and mirror descent is discretization of natural gradient flow)

- Su, Boyd, Candes '14: Continuous time limit of accelerated gradient descent is a second-order ODE

$$\ddot{X}_t + \frac{3}{t}\dot{X}_t + \nabla f(X_t) = 0$$

- These ODEs are obtained by taking continuous time limits. Is there a deeper generative mechanism?

Our work: A general variational approach to acceleration

A systematic discretization methodology

Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

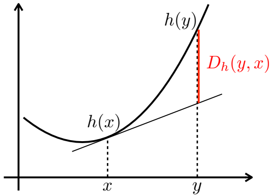
- Function of position x, velocity $\dot{x}$, and time t

Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

- Function of position x, velocity $\dot{x}$, and time t
- $D_h(y, x) = h(y) - h(x) - \langle \nabla h(x), y - x \rangle$ is the Bregman divergence
- h is the convex distance-generating function

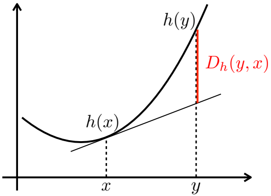


Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

- Function of position x, velocity $\dot{x}$, and time t
- $D_h(y, x) = h(y) - h(x) - \langle \nabla h(x), y - x \rangle$ is the Bregman divergence
- h is the convex distance-generating function
- f is the convex objective function

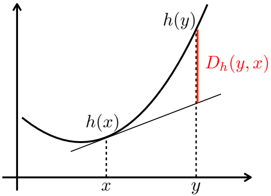


Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t} \dot{x}, x) - e^{\beta_t} f(x) \right)$$

- Function of position x, velocity $\dot{x}$, and time t
- $D_h(y, x) = h(y) - h(x) - \langle \nabla h(x), y - x \rangle$ is the Bregman divergence
- h is the convex distance-generating function
- f is the convex objective function
- $\alpha_t, \beta_t, \gamma_t \in \mathbb{R}$ are arbitrary smooth functions

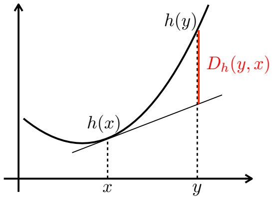


Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t - \alpha_t} \left(\frac{1}{2} \|\dot{x}\|^2 - e^{2\alpha_t + \beta_t} f(x) \right)$$

- Function of position x, velocity $\dot{x}$, and time t
- $D_h(y, x) = h(y) - h(x) - \langle \nabla h(x), y - x \rangle$ is the Bregman divergence
- h is the convex distance-generating function
- f is the convex objective function
- $\alpha_t, \beta_t, \gamma_t \in \mathbb{R}$ are arbitrary smooth functions
- In Euclidean setting, simplifies to damped Lagrangian

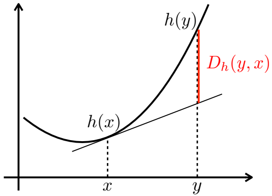


Bregman Lagrangian

Define the **Bregman Lagrangian**:

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t} \dot{x}, x) - e^{\beta_t} f(x) \right)$$

- Function of position x, velocity $\dot{x}$, and time t
- $D_h(y, x) = h(y) - h(x) - \langle \nabla h(x), y - x \rangle$ is the Bregman divergence
- h is the convex distance-generating function
- f is the convex objective function
- $\alpha_t, \beta_t, \gamma_t \in \mathbb{R}$ are arbitrary smooth functions
- In Euclidean setting, simplifies to damped Lagrangian



Ideal scaling conditions:

$$\dot{\beta}_t \leq e^{\alpha_t}$$

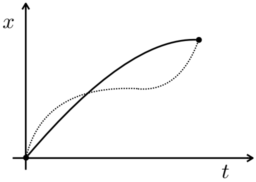
$$\dot{\gamma}_t = e^{\alpha_t}$$

Bregman Lagrangian

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

Variational problem over curves:

$$\min_X \int \mathcal{L}(X_t, \dot{X}_t, t)\, dt$$



Optimal curve is characterized by **Euler-Lagrange** equation:

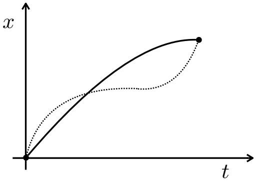
$$\frac{d}{dt} \left\{ \frac{\partial \mathcal{L}}{\partial \dot{x}}(X_t, \dot{X}_t, t) \right\} = \frac{\partial \mathcal{L}}{\partial x}(X_t, \dot{X}_t, t)$$

Bregman Lagrangian

$$\mathcal{L}(x, \dot{x}, t) = e^{\gamma_t + \alpha_t} \left(D_h(x + e^{-\alpha_t}\dot{x}, x) - e^{\beta_t} f(x) \right)$$

Variational problem over curves:

$$\min_X \int \mathcal{L}(X_t, \dot{X}_t, t)\, dt$$



Optimal curve is characterized by **Euler-Lagrange** equation:

$$\frac{d}{dt}\left\{ \frac{\partial \mathcal{L}}{\partial \dot{x}}(X_t, \dot{X}_t, t) \right\} = \frac{\partial \mathcal{L}}{\partial x}(X_t, \dot{X}_t, t)$$

E-L equation for Bregman Lagrangian under ideal scaling:

$$\ddot{X}_t + (e^{\alpha_t} - \dot{\alpha}_t)\dot{X}_t + e^{2\alpha_t + \beta_t}\left[\nabla^2 h(X_t + e^{-\alpha_t}\dot{X}_t)\right]^{-1} \nabla f(X_t) = 0$$

General convergence rate

Theorem

Theorem Under ideal scaling, the E-L equation has convergence rate

$$f(X_t) - f(x^*) \leq O(e^{-\beta_t})$$

General convergence rate

Theorem

Theorem Under ideal scaling, the E-L equation has convergence rate

$$f(X_t) - f(x^*) \leq O(e^{-\beta_t})$$

Proof. Exhibit a Lyapunov function for the dynamics:

$$\mathcal{E}_t = D_h\left(x^*,\, X_t + e^{-\alpha_t}\dot{X}_t\right) + e^{\beta_t}(f(X_t) - f(x^*))$$
$$\dot{\mathcal{E}}_t = -e^{\alpha_t+\beta_t} D_f(x^*, X_t) + (\dot{\beta}_t - e^{\alpha_t})e^{\beta_t}(f(X_t) - f(x^*)) \;\leq\; 0$$

□

Note: Only requires convexity and differentiability of f, h

Polynomial convergence rate

For $p > 0$, choose parameters:

$$\alpha_t = \log p - \log t$$
$$\beta_t = p \log t + \log C$$
$$\gamma_t = p \log t$$

E-L equation has $O(e^{-\beta_t}) = O(1/t^p)$ convergence rate:

$$\ddot{X}_t + \frac{p+1}{t}\dot{X}_t + Cp^2 t^{p-2}\Big[\nabla^2 h\Big(X_t + \frac{t}{p}\dot{X}_t\Big)\Big]^{-1}\nabla f(X_t) = 0$$

Polynomial convergence rate

For $p > 0$, choose parameters:

$$\alpha_t = \log p - \log t$$
$$\beta_t = p \log t + \log C$$
$$\gamma_t = p \log t$$

E-L equation has $O(e^{-\beta_t}) = O(1/t^p)$ convergence rate:

$$\ddot{X}_t + \frac{p+1}{t}\dot{X}_t + Cp^2 t^{p-2}\left[\nabla^2 h\Big(X_t + \frac{t}{p}\dot{X}_t\Big)\right]^{-1}\nabla f(X_t) = 0$$

For $p = 2$:

- Recover result of Krichene et al with $O(1/t^2)$ convergence rate
- In Euclidean case, recover ODE of Su et al:
  $\ddot{X}_t + \frac{3}{t}\dot{X}_t + \nabla f(X_t) = 0$

Time dilation property (reparameterizing time)

($p = 2$: accelerated gradient descent)

$$O\left(\frac{1}{t^2}\right): \quad \ddot{X}_t + \frac{3}{t}\dot{X}_t + 4C\Big[\nabla^2 h\Big(X_t + \frac{t}{2}\dot{X}_t\Big)\Big]^{-1}\nabla f(X_t) = 0$$

$$\Big\downarrow \quad \text{speed up time: } Y_t = X_{t^{3/2}}$$

$$O\left(\frac{1}{t^3}\right): \quad \ddot{Y}_t + \frac{4}{t}\dot{Y}_t + 9Ct\Big[\nabla^2 h\Big(Y_t + \frac{t}{3}\dot{Y}_t\Big)\Big]^{-1}\nabla f(Y_t) = 0$$

($p = 3$: accelerated cubic-regularized Newton's method)

Time dilation property (reparameterizing time)

($p = 2$: accelerated gradient descent)

$$O\left(\frac{1}{t^2}\right): \quad \ddot{X}_t + \frac{3}{t}\dot{X}_t + 4C\left[\nabla^2 h\Big(X_t + \frac{t}{2}\dot{X}_t\Big)\right]^{-1}\nabla f(X_t) = 0$$

$$\Big\downarrow \quad \text{speed up time: } Y_t = X_{t^{3/2}}$$

$$O\left(\frac{1}{t^3}\right): \quad \ddot{Y}_t + \frac{4}{t}\dot{Y}_t + 9Ct\left[\nabla^2 h\Big(Y_t + \frac{t}{3}\dot{Y}_t\Big)\right]^{-1}\nabla f(Y_t) = 0$$

($p = 3$: accelerated cubic-regularized Newton's method)

- All accelerated methods are traveling the same curve in space-time at different speeds
- Gradient methods don't have this property
  - From gradient flow to rescaled gradient flow: Replace $\frac{1}{2}\|\cdot\|^2$ by $\frac{1}{p}\|\cdot\|^p$

Time dilation for general Bregman Lagrangian

$$
\begin{array}{c}
O(e^{-\beta_t}) : \quad \text{E-L for Lagrangian } \mathcal{L}_{\alpha,\beta,\gamma} \\
\Big\downarrow \quad \text{speed up time: } Y_t = X_{\tau(t)} \\
O(e^{-\beta_{\tau(t)}}) : \quad \text{E-L for Lagrangian } \mathcal{L}_{\tilde{\alpha},\tilde{\beta},\tilde{\gamma}}
\end{array}
$$

where

$$
\begin{aligned}
\tilde{\alpha}_t &= \alpha_{\tau(t)} + \log \dot{\tau}(t) \\
\tilde{\beta}_t &= \beta_{\tau(t)} \\
\tilde{\gamma}_t &= \gamma_{\tau(t)}
\end{aligned}
$$

Time dilation for general Bregman Lagrangian

$$O(e^{-\beta_t}) : \quad \text{E-L for Lagrangian } \mathcal{L}_{\alpha,\beta,\gamma}$$

$$\Big\downarrow \quad \text{speed up time: } Y_t = X_{\tau(t)}$$

$$O(e^{-\beta_{\tau(t)}}) : \quad \text{E-L for Lagrangian } \mathcal{L}_{\tilde{\alpha},\tilde{\beta},\tilde{\gamma}}$$

where

$$\tilde{\alpha}_t = \alpha_{\tau(t)} + \log \dot{\tau}(t)$$
$$\tilde{\beta}_t = \beta_{\tau(t)}$$
$$\tilde{\gamma}_t = \gamma_{\tau(t)}$$

Question: How to discretize E-L while preserving the convergence rate?

Discretizing the dynamics (naive approach)

Write E-L as a system of first-order equations:

$$Z_t = X_t + \frac{t}{p}\dot{X}_t$$

$$\frac{d}{dt}\nabla h(Z_t) = -Cpt^{p-1}\nabla f(X_t)$$

Discretizing the dynamics (naive approach)

Write E-L as a system of first-order equations:

$$Z_t = X_t + \frac{t}{p}\dot{X}_t$$
$$\frac{d}{dt}\nabla h(Z_t) = -Cpt^{p-1}\nabla f(X_t)$$

Euler discretization with time step $\delta > 0$ (i.e., set $x_k = X_t$, $x_{k+1} = X_{t+\delta}$):

$$x_{k+1} = \frac{p}{k+p}z_k + \frac{k}{k+p}x_k$$
$$z_k = \arg\min_z \left\{ Cpk^{(p-1)}\langle \nabla f(x_k), z\rangle + \frac{1}{\epsilon}D_h(z, z_{k-1}) \right\}$$

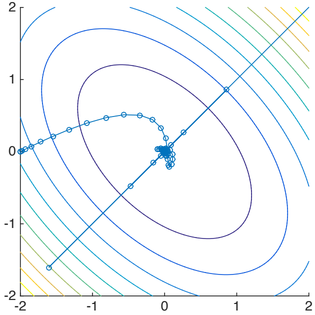
with step size $\epsilon = \delta^p$, and $k^{(p-1)} = k(k+1)\cdots(k+p-2)$ is the rising factorial

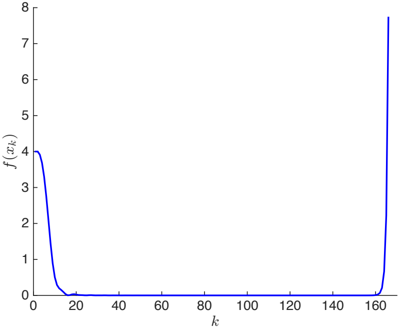
Naive discretization doesn't work

$$x_{k+1} = \frac{p}{k+p} z_k + \frac{k}{k+p} x_k$$

$$z_k = \arg\min_z \left\{ Cpk^{(p-1)} \langle \nabla f(x_k), z \rangle + \frac{1}{\epsilon} D_h(z, z_{k-1}) \right\}$$

Cannot obtain a convergence guarantee, and empirically unstable





Modified discretization

Introduce an auxiliary sequence y_k:

$$x_{k+1} = \frac{p}{k+p} z_k + \frac{k}{k+p} y_k$$

$$z_k = \arg\min_z \left\{ Cpk^{(p-1)} \langle \nabla f(y_k), z \rangle + \frac{1}{\epsilon} D_h(z, z_{k-1}) \right\}$$

Sufficient condition: $\langle \nabla f(y_k), x_k - y_k \rangle \geq M \epsilon^{\frac{1}{p-1}} \|\nabla f(y_k)\|_*^{\frac{p}{p-1}}$

Modified discretization

Introduce an auxiliary sequence y_k:

$$x_{k+1} = \frac{p}{k+p} z_k + \frac{k}{k+p} y_k$$

$$z_k = \arg\min_z \left\{ Cpk^{(p-1)} \langle \nabla f(y_k), z \rangle + \frac{1}{\epsilon} D_h(z, z_{k-1}) \right\}$$

Sufficient condition: $\langle \nabla f(y_k), x_k - y_k \rangle \geq M \epsilon^{\frac{1}{p-1}} \|\nabla f(y_k)\|_*^{\frac{p}{p-1}}$

Assume h is uniformly convex: $D_h(y, x) \geq \frac{1}{p} \|y - x\|^p$

Theorem

Theorem

$$f(y_k) - f(x^*) \leq O\left(\frac{1}{\epsilon k^p}\right)$$

Note: Matching convergence rates $1/(\epsilon k^p) = 1/(\delta k)^p = 1/t^p$

Proof using generalization of Nesterov's estimate sequence technique

Accelerated higher-order gradient method

$$x_{k+1} = \frac{p}{k+p} z_k + \frac{k}{k+p} y_k$$

$$y_k = \arg\min_y \left\{ f_{p-1}(y; x_k) + \frac{2}{\epsilon p} \|y - x_k\|^p \right\} \quad \leftarrow O\left(\frac{1}{\epsilon k^{p-1}}\right)$$

$$z_k = \arg\min_z \left\{ Cpk^{(p-1)} \langle \nabla f(y_k), z \rangle + \frac{1}{\epsilon} D_h(z, z_{k-1}) \right\}$$

If $\nabla^{p-1} f$ is $(1/\epsilon)$-Lipschitz and h is uniformly convex of order p, then:

$$f(y_k) - f(x^*) \leq O\left(\frac{1}{\epsilon k^p}\right) \quad \leftarrow \text{ accelerated rate}$$

$p = 2$: Accelerated gradient/mirror descent

$p = 3$: Accelerated cubic-regularized Newton's method (Nesterov '08)

$p \geq 2$: Accelerated higher-order method

Recap: Gradient vs. accelerated methods

How to design dynamics for minimizing a convex function f?

Rescaled gradient flow

$$\dot{X}_t = -\nabla f(X_t) \,/\, \|\nabla f(X_t)\|_*^{\frac{p-2}{p-1}}$$

$$O\left(\frac{1}{t^{p-1}}\right)$$

Higher-order gradient method

$O\left(\frac{1}{\epsilon k^{p-1}}\right)$ when $\nabla^{p-1} f$ is $\frac{1}{\epsilon}$-Lipschitz

matching rate with $\epsilon = \delta^{p-1} \Leftrightarrow \delta = \epsilon^{\frac{1}{p-1}}$

Recap: Gradient vs. accelerated methods

How to design dynamics for minimizing a convex function f?

Rescaled gradient flow

$$\dot{X}_t = -\nabla f(X_t) \,/\, \|\nabla f(X_t)\|_*^{\frac{p-2}{p-1}}$$

$$O\left(\frac{1}{t^{p-1}}\right)$$

Polynomial Euler-Lagrange equation

$$\ddot{X}_t + \frac{p+1}{t}\dot{X}_t + t^{p-2}\left[\nabla^2 h\left(X_t + \frac{t}{p}\dot{X}_t\right)\right]^{-1}\nabla f(X_t) =$$

$$O\left(\frac{1}{t^{p}}\right)$$

Higher-order gradient method

$O\left(\frac{1}{\epsilon k^{p-1}}\right)$ when $\nabla^{p-1} f$ is $\frac{1}{\epsilon}$-Lipschitz

matching rate with $\epsilon = \delta^{p-1} \Leftrightarrow \delta = \epsilon^{\frac{1}{p-1}}$

Accelerated higher-order method

$O\left(\frac{1}{\epsilon k^{p}}\right)$ when $\nabla^{p-1} f$ is $\frac{1}{\epsilon}$-Lipschitz

matching rate with $\epsilon = \delta^{p} \Leftrightarrow \delta = \epsilon^{\frac{1}{p}}$

Towards A Symplectic Perspective

- If initialized close enough, diminishing gradient flow will relax to an optimum quickly

$$\frac{\mathrm{d}q}{\mathrm{d}t} = -\frac{C}{t}\nabla f(q)$$

Towards A Symplectic Perspective

- We can construct physical systems that will rapidly evolve into the neighborhood of the optimum, but the inertia can slow relaxation once we get there

$$L(q, v, f(q), t)$$

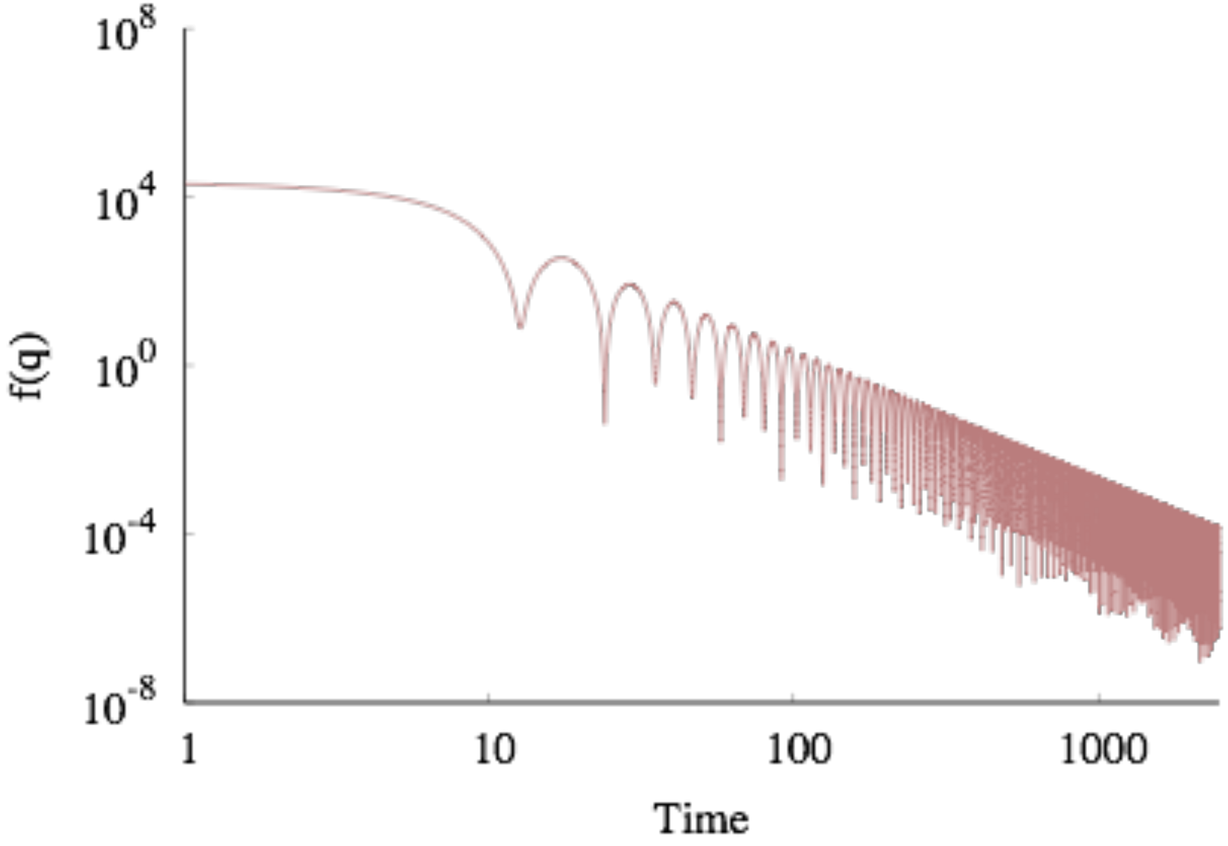
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial v} = \frac{\partial L}{\partial q}$$

$$\Rightarrow$$

$$\frac{\mathrm{d}q}{\mathrm{d}t} = v$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = g(q, v, t)$$

Towards A Symplectic Perspective



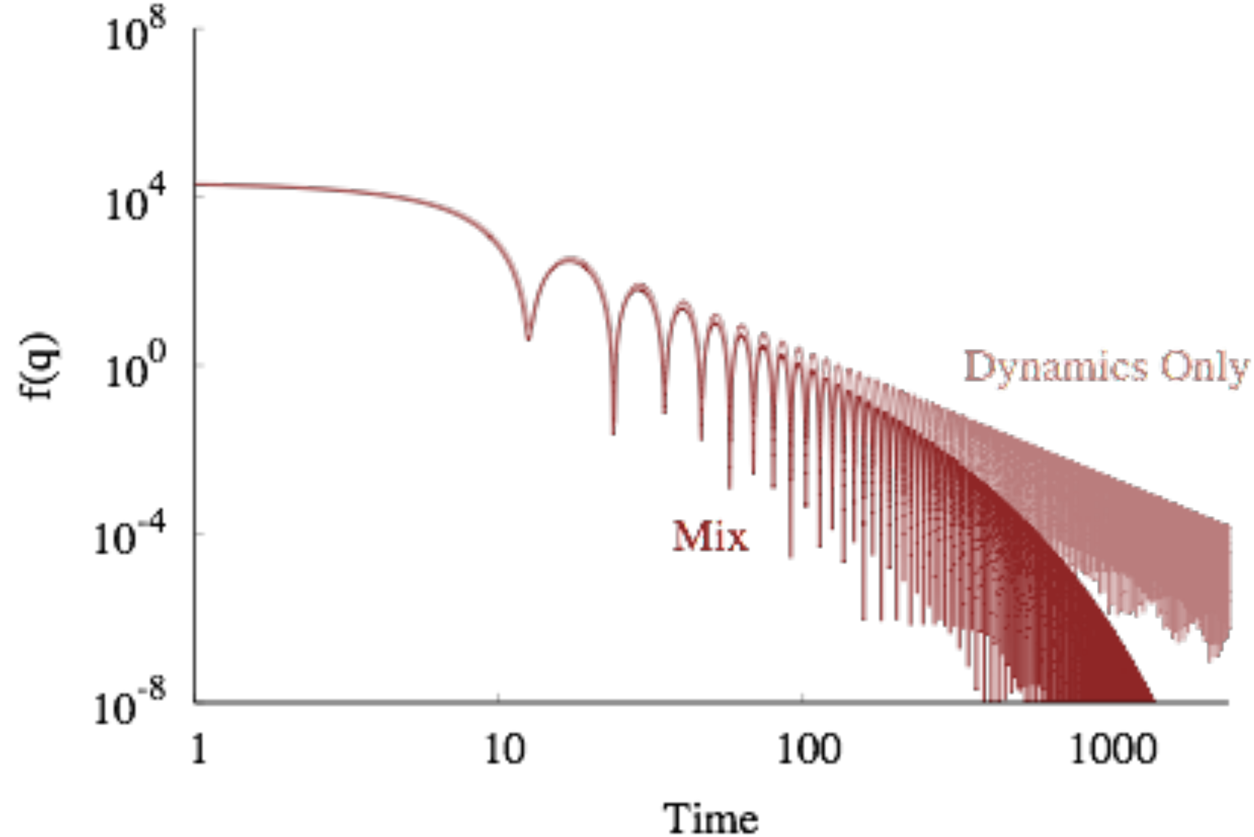
Towards A Symplectic Perspective

- Can a mixture of these flows yield rapid convergence to the optimum in both regimes?

$$\frac{\mathrm{d}q}{\mathrm{d}t} = v \quad - \delta \, \frac{C}{t} \nabla f(q)$$

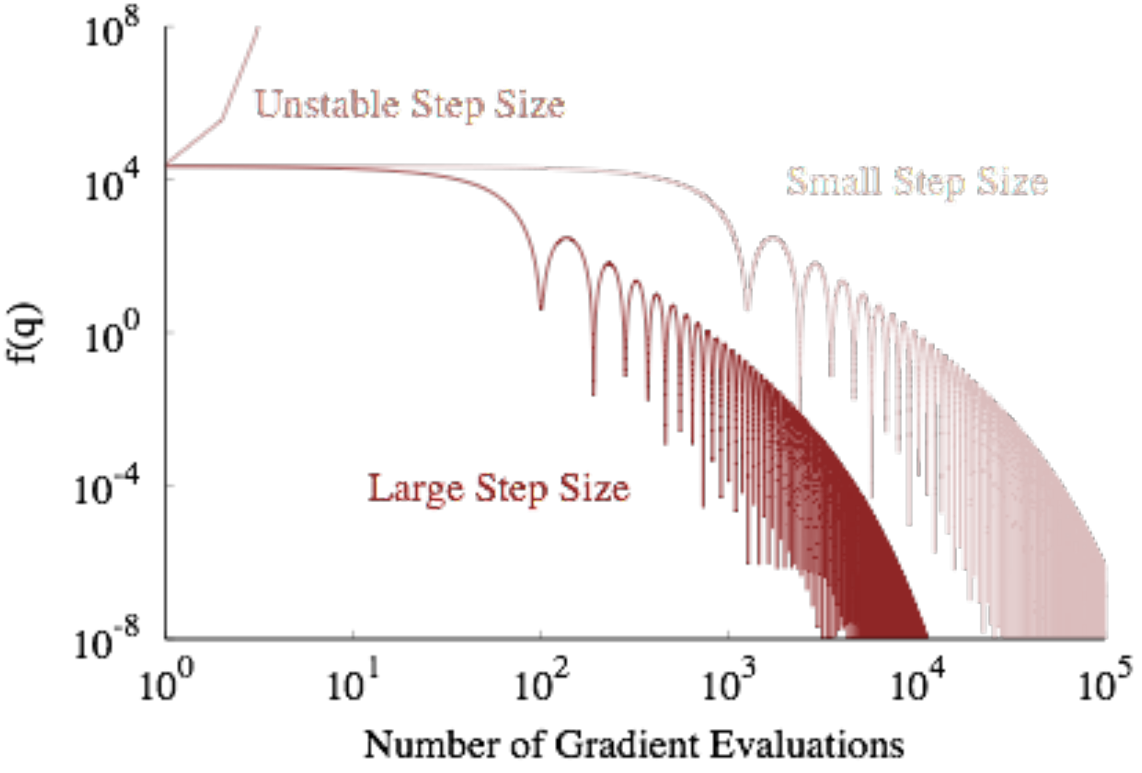
$$\frac{\mathrm{d}v}{\mathrm{d}t} = g(q, v, t)$$

Towards A Symplectic Perspective



Towards A Symplectic Perspective

- Speed also depends on the discretization



Towards A Symplectic Perspective

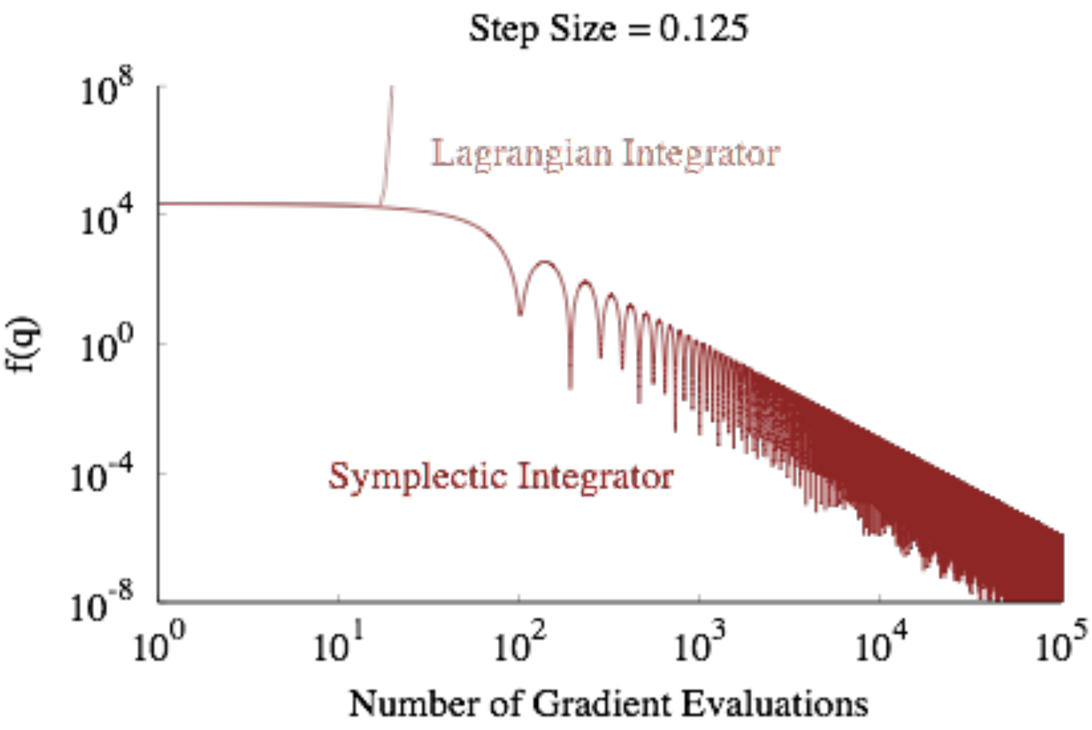
- Discretization of the Lagrangian dynamics, however, is fragile and requires small step sizes.
- We can build more robust solutions by taking a Legendre transform and considering a *Hamiltonian* formalism:

$$L(q, v, t) \to H(q, p, t, \mathcal{E})$$

$$\left(\frac{\mathrm{d}q}{\mathrm{d}t}, \frac{\mathrm{d}v}{\mathrm{d}t} \right) \to \left(\frac{\mathrm{d}q}{\mathrm{d}\tau}, \frac{\mathrm{d}p}{\mathrm{d}\tau}, \frac{\mathrm{d}t}{\mathrm{d}\tau}, \frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\tau} \right)$$

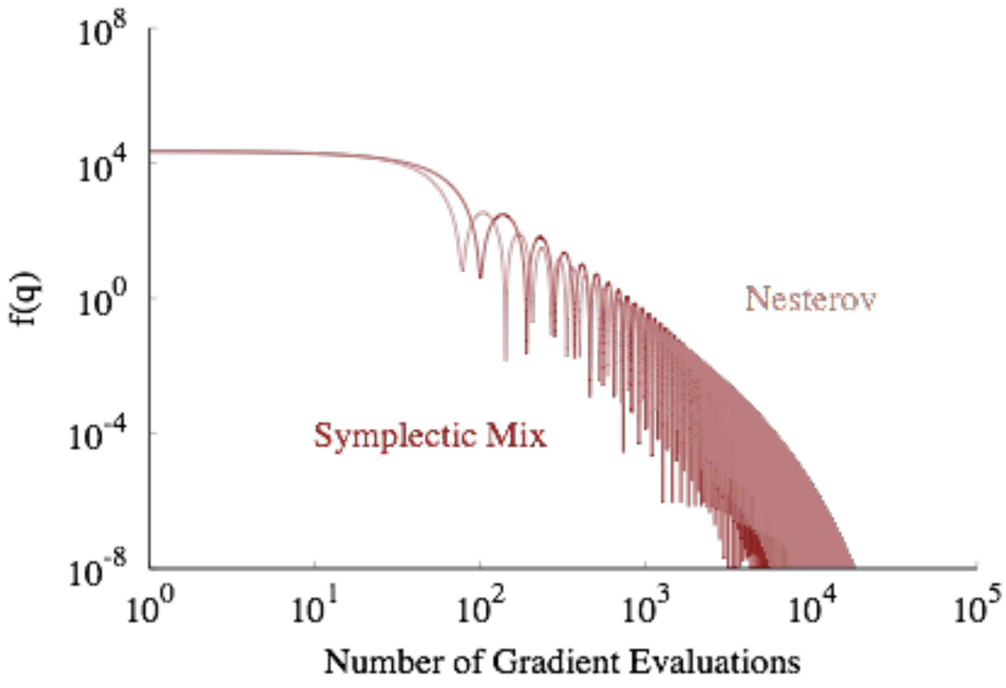
Towards A Symplectic Perspective

- The Hamiltonian perspective admits *symplectic integrators* which are accurate and stable even for large step sizes



Towards A Symplectic Perspective

- Exploiting this stability yields algorithms with state-of-the-art performance, and perhaps even more:



Part II

Avoiding Saddlepoints, Efficiently

with Chi Jin, Rong Ge, Praneeth Netrapalli and Sham Kakade

Gradient Descent

To minimize a function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$, **gradient descent** (GD)

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t).$$

Function $f(\cdot)$ is ℓ**-smooth (or gradient Lipschitz)**

$$\forall \mathbf{x}_1, \mathbf{x}_2, \ \|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\| \leq \ell \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

Point $\mathbf{x}$ is an ϵ**-first-order stationary point** if $\|\nabla f(\mathbf{x})\| \leq \epsilon$.

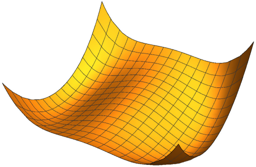
GD Converges to First-order Stationary Point (Nesterov, 1998)

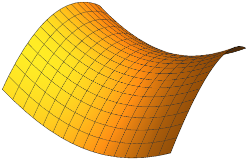
For ℓ-smooth function, gradient descent with learning rate $\eta = 1/\ell$ finds an ϵ-first-order stationary point in $\ell(f(\mathbf{x}_0) - f^\star)/\epsilon^2$ iterations.

Iterations required is dimension free, thus scalable for high dimensional problem.

Saddle Points and Perturbed Gradient Descent

However, first-order stationary points can be **local min/max** or **saddle points**.





Perturbed Gradient Descent (PGD)

1. **for** $t = 0, 1, \ldots$ **do**
2. **if** perturbation condition holds **then**
3. $\mathbf{x}_t \leftarrow \mathbf{x}_t + \xi_t, \qquad \xi_t$ uniformly $\sim \mathbb{B}_0(r)$
4. $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)$

Question: how fast can perturbed gradient descent escape saddle points?

Main Result

Function $f(\cdot)$ is ρ-**Hessian Lipschitz** if

$$\forall \mathbf{x}_1, \mathbf{x}_2, \ \|\nabla^2 f(\mathbf{x}_1) - \nabla^2 f(\mathbf{x}_2)\| \leq \rho \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

Point $\mathbf{x}$ is an ϵ-**second-order stationary point** if (Nesterov and Polyak, 2006)

$$\|\nabla f(\mathbf{x})\| \leq \epsilon, \qquad \text{and} \qquad \lambda_{\min}(\nabla^2 f(\mathbf{x})) \geq -\sqrt{\rho\epsilon}$$

PGD Converges to Second-order Stationary Point

For ℓ-gradient Lipschitz and ρ-Hessian Lipschitz function, perturbed gradient descent with learning rate $\eta = O(1/\ell)$ finds an ϵ-second-order stationary point in $\tilde{O}(\ell(f(\mathbf{x}_0) - f^\star)/\epsilon^2)$ iterations, with high probability.

Stronger guarantees within same iteration as (Nesterov 1998) up to log factors.

Answer: almostly as fast as finding first-order stationary points.

Compare with Earlier Works

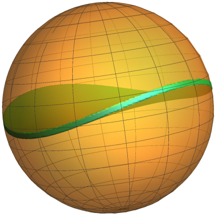
Standard approaches check Hessian info to escape saddle points.

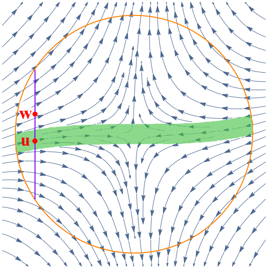
Algorithm	**Iterations**	**Oracle**
Ge et al. (2015)	$O(\text{poly}(d/\epsilon))$	Gradient
Levy (2016)	$O(d^3 \cdot \text{poly}(1/\epsilon))$	Gradient
This Work	$O(\log^4(d)/\epsilon^2)$	Gradient
Agarwal et al. (2016)	$O(\log(d)/\epsilon^{1.75})$	Hessian-vector product
Carmon et al. (2016)	$O(\log(d)/\epsilon^{1.75})$	Hessian-vector product
Carmon and Duchi (2016)	$O(\log(d)/\epsilon^2)$	Hessian-vector product
Nesterov and Polyak (2006)	$O(1/\epsilon^{1.5})$	Hessian
Curtis et al. (2014)	$O(1/\epsilon^{1.5})$	Hessian

For simplicity, we only highlight dependencies on dimension d and ϵ.

Geometry and Dynamics around Saddle Points

Key step: PGD will decrease function value over multiple steps even when "around saddle point".





Stuck region (green) forms a non-flat "thin pancake" shape, which is so "thin" that random perturbation has extremely small chance to hit it.

Take Away: a bit perturbation is all you need to escape saddle points efficiently

Part III

Stochastically-Controlled Stochastic Gradient

with Lihua Lei

Setup

Task: minimizing a composite objective:

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

Setup

Task: minimizing a composite objective:

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

Assumption: $\exists L < \infty, \mu \geq 0$, s.t.

$$\frac{\mu}{2}\|x - y\|^2 \leq f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y \rangle \leq \frac{L}{2}\|x - y\|^2$$

Setup

Task: minimizing a composite objective:

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

Assumption: $\exists L < \infty, \mu \geq 0$, s.t.

$$\frac{\mu}{2}\|x-y\|^2 \leq f_i(x) - f_i(y) - \langle \nabla f_i(y), x-y \rangle \leq \frac{L}{2}\|x-y\|^2$$

- $\mu = 0$: non-strongly convex case;
- $\mu > 0$: strongly convex case; $\kappa \triangleq L/\mu$.

Computation Complexity

- Accessing $(f_i(x), \nabla f_i(x))$ incurs one unit of cost;
- Given $\epsilon > 0$, let $T(\epsilon)$ be the minimum cost to achieve

$$\mathbb{E}\left(f(x_{T(\epsilon)}) - f(x^*)\right) \leq \epsilon;$$

- Worst-case analysis: bound $T(\epsilon)$ almost surely, e.g.,

$$T(\epsilon) = O\left((n+\kappa)\log\frac{1}{\epsilon}\right) \qquad \text{(SVRG)}.$$

SVRG Algorithm

SVRG: (within an epoch)

1: $\mathcal{I} \leftarrow [n]$

2: $g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

3: $m \leftarrow n$

4: Generate $N \sim U([m])$

5: **for** $k = 1, 2, \cdots, N$ **do**

6: Randomly pick $i \in [n]$

7: $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

8: $x \leftarrow x - \eta \nu$

9: **end for**

Analysis

	General Convex	Strongly Convex
Nesterov's AGD	$\frac{n}{\sqrt{\epsilon}}$	$n\sqrt{\kappa}\log\frac{1}{\epsilon}$
SGD	$\frac{1}{\epsilon^2}$	$\frac{\kappa}{\epsilon}\log\frac{1}{\epsilon}$
SVRG	-	$(n+\kappa)\log\frac{1}{\epsilon}$
Katyusha	$\frac{n}{\sqrt{\epsilon}}$	$(n+\sqrt{n\kappa})\log\frac{1}{\epsilon}$

- All above results are from worst-case analysis;
- SGD is the only method with complexity free of n; however, the stepsize η depends on the unknown $\|x_0 - x^*\|^2$ and the total number of epochs T.

An algorithm is *tuning-friendly* if:

- the stepsize η is the only parameter to tune;
- η is a constant which only depends on L and μ.

	General Convex	Strongly Convex	Tuning-friendly
SGD	$\frac{1}{\epsilon^2}$	$\frac{\kappa}{\epsilon}\log\frac{1}{\epsilon}$	No
SCSG	$\frac{1}{\epsilon^2}\wedge\frac{n}{\epsilon}$	$\left(\frac{1}{\epsilon}\wedge n+\kappa\right)\log\frac{1}{\epsilon}$	Yes
SCSG+	$\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\wedge n\right)+\frac{\log n}{n\epsilon^2}$	$\frac{1}{\epsilon}+\frac{\kappa}{\epsilon^{\alpha}}(\alpha\ll 1)$	Yes
SCSG+	$\frac{1}{\epsilon}\sqrt{\log\left(\frac{1}{\epsilon}\wedge n\right)}+\frac{\sqrt{\log n}}{\sqrt{n}\epsilon^{\frac{3}{2}}}$	$\sqrt{\frac{\kappa}{\epsilon}}+\kappa$	No

SCSG/SCSG+: Algorithm

SVRG: (within an epoch)

$\mathcal{I} \leftarrow [n]$

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$m \leftarrow n$

Generate $N \sim U([m])$

for $k = 1, 2, \cdots, N$ **do**

  Randomly pick $i \in [n]$

  $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

  $x \leftarrow x - \eta\nu$

end for

SCSG/SCSG+: Algorithm

SVRG: (within an epoch)

$\mathcal{I} \leftarrow [n]$

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$m \leftarrow n$

Generate $N \sim U([m])$

for $k = 1, 2, \cdots, N$ **do**

    Randomly pick $i \in [n]$

    $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

    $x \leftarrow x - \eta\nu$

end for

SCSG(+): (within an epoch)

SCSG/SCSG+: Algorithm

SVRG: (within an epoch)

$\mathcal{I} \leftarrow [n]$

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$m \leftarrow n$

Generate $N \sim U([m])$

for $k = 1, 2, \cdots, N$ **do**

  Randomly pick $i \in [n]$

  $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

  $x \leftarrow x - \eta\nu$

end for

SCSG(+): (within an epoch)

Randomly pick $\mathcal{I}$ with size B

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

SCSG/SCSG+: Algorithm

SVRG: (within an epoch)

$\mathcal{I} \leftarrow [n]$

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$m \leftarrow n$

Generate $N \sim U([m])$

for $k = 1, 2, \cdots, N$ **do**

  Randomly pick $i \in [n]$

  $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

  $x \leftarrow x - \eta\nu$

end for

SCSG(+): (within an epoch)

Randomly pick $\mathcal{I}$ with size B

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$\gamma \leftarrow 1 - 1/B$

Generate $N \sim \mathrm{Geo}(\gamma)$

SCSG/SCSG+: Algorithm

SVRG: (within an epoch)

$\mathcal{I} \leftarrow [n]$

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$m \leftarrow n$

Generate $N \sim U([m])$

for $k = 1, 2, \cdots, N$ **do**

    Randomly pick $i \in [n]$

    $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

    $x \leftarrow x - \eta\nu$

end for

SCSG(+): (within an epoch)

Randomly pick $\mathcal{I}$ with size B

$g \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_i'(x_0)$

$\gamma \leftarrow 1 - 1/B$

Generate $N \sim \mathrm{Geo}(\gamma)$

for $k = 1, 2, \cdots, N$ **do**

    Randomly pick $i \in \mathcal{I}$

    $\nu \leftarrow f_i'(x) - f_i'(x_0) + g$

    $x \leftarrow x - \eta\nu$

end for

SCSG/SCSG+: Algorithm

In epoch j,

- SCSG fixes $B_j \equiv B(\epsilon)$;
- Explicit forms of $B(\epsilon)$ are given in both non-strongly convex cases and strongly convex cases;
- SCSG+ uses an geometrically increasing sequence

$$B_j = \lceil B_0 b^j \wedge n \rceil$$

Conclusion

- SCSG/SCSG+ saves computation costs on average by avoiding calculating the full gradient;
- SCSG/SCSG+ also saves communication costs in the distributed system by avoiding sampling a gradient from the whole dataset;
- SCSG/SCSG+ are able to achieve an approximate optimum with potentially less than a single pass through the data;
- The average computation cost of SCSG+ beats the oracle lower bounds from worst-case analysis.