Restarted Nonconvex Accelerated Gradient Descent: No More Polylogarithmic Factor in the $O(\epsilon^{-7/4})$ Complexity

Huan Li
Nankai University

Zhouchen Lin
Peking University

Non-convex Optimization

- Problem: $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$ $f(\mathbf{x})$: non-convex function

- Applications: Matrix completion, matrix factorization, robust PCA, phase retrieval, deep learning

- Demand: Fast first-order solvers for high dimensional problems in machine learning

Accelerated Gradient Descent (AGD)

- Gradient descent (GD), a fundamental algorithm in machine learning
- GD is not optimal for convex problems. AGD is faster and optimal.
- Question: Can we design AGD for non-convex problems faster than GD ?
- Assumptions:
  - Lipschitz gradient: $\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\|$
  - Lipschitz Hessian: $\|\nabla^2 f(\mathbf{y}) - \nabla^2 f(\mathbf{x})\|_2 \leq \rho\|\mathbf{y} - \mathbf{x}\|$

Previous Work

- Carmon et al. (2018) and Agarwal et al. (2017) proved the $O\left(\frac{\mathrm{polylog} d}{\epsilon^{7/4}}\log\frac{1}{\epsilon}\right)$ rate to find second-order stationary point with high probability
  - ϵ-second-order stationary point: $\|\nabla f(\mathbf{x})\| \leq \epsilon, \quad \lambda_{min}(\nabla^2 f(\mathbf{x})) \succeq -\sqrt{\epsilon\rho}$
  - Solve sequence of regularized subproblems using convex AGD
- Carmon et al. (2017) proved the $O\left(\frac{L^{1/2}\rho^{1/4}\triangle_f}{\epsilon^{7/4}}\log\frac{1}{\epsilon}\right)$ rate to find first-order stationary point
  - Also solve sequence of regularized subproblems using convex AGD
- Jin et al. (2018) proposed the first single-loop AGD with the $O\left(\frac{\mathrm{polylog} d}{\epsilon^{7/4}}\log\frac{1}{\epsilon}\right)$ rate to find second-order stationary point with high probability
  - Use negative curvature exploitation (NCE) when the function is too non-convex

Question

- Can we design much simpler AGD?
  - Without NCE and sequence of regularized subproblems to solve
- If possible, can we prove faster rate?
  - For example, remove the $O\left(\log\frac{1}{\epsilon}\right)$ factor

Our Method: Restarted AGD

- Two distinguishing features:
  - Restart
  - Specific average
- Simple algorithm structures
  - No NEC, no subproblems to solve
- Faster rate to find first-order stationary point
  - $O\left(\frac{L^{1/2}\rho^{1/4}\triangle_f}{\epsilon^{7/4}}\right)$ v.s. $O\left(\frac{L^{1/2}\rho^{1/4}\triangle_f}{\epsilon^{7/4}}\log\frac{1}{\epsilon}\right)$

    Ours SOTA
  - No more $O\left(\log\frac{1}{\epsilon}\right)$ factor

Algorithm 1 Restarted AGD

Initialize $\mathbf{x}^{-1} = \mathbf{x}^0 = \mathbf{x}_{int}$, $k = 0$.
while $k < K$ **do**
  $\mathbf{y}^k = \mathbf{x}^k + (1-\theta)(\mathbf{x}^k - \mathbf{x}^{k-1})$
  $\mathbf{x}^{k+1} = \mathbf{y}^k - \eta\nabla f(\mathbf{y}^k)$
  $k = k+1$
  if $k\sum_{t=0}^{k-1}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 > B^2$ **then**
    $\mathbf{x}^{-1} = \mathbf{x}^0 = \mathbf{x}^k$, $k = 0$
  end if
end while
$K_0 = \text{argmin}_{\lfloor\frac{K}{2}\rfloor \le k \le K-1}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$
Output $\hat{\mathbf{y}} = \frac{1}{K_0+1}\sum_{k=0}^{K_0}\mathbf{y}^k$

Extension to Second-order Stationary point

- Add perturbations when restart

$$\mathbf{x}^{-1} = \mathbf{x}^0 = \mathbf{x}^k + \xi 1_{\|\nabla f(\mathbf{y}^{k-1})\| \le \frac{B}{\eta}}, \ k = 0,$$
$$\xi \sim \mathrm{Unif}(\mathbb{B}_0(r))$$

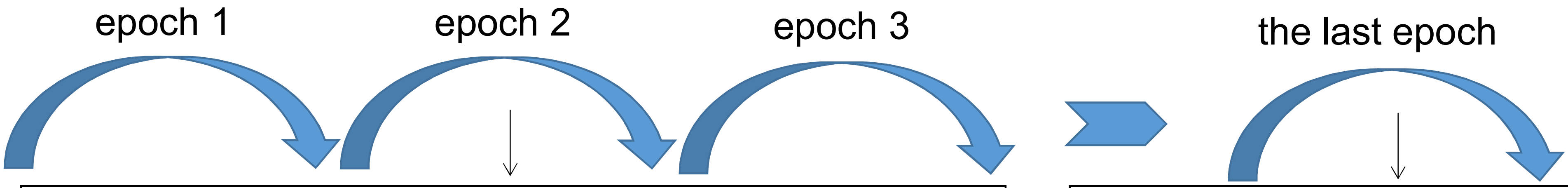
- Need $O\left(\frac{L^{1/2}\rho^{1/4}\triangle_f}{\epsilon^{7/4}}\log^6\frac{d}{\zeta\epsilon}\right)$ iterations to find ϵ-second-order stationary point with probability at least $1-\zeta$
  - The same with the rate in (Jin et al. 2018)

Proof Sketch

- One epoch:

Iterations from $k=0$ to $\mathcal{K}=\min_k\left\{k \middle| k\sum_{t=0}^{k-1}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 > B^2\right\}$ until the if condition triggers

- Approximate $f(\mathbf{x})$ by its quadratic Talor expansion at each epoch



- Less than $K=O(\epsilon^{-1/4})$ iterations at each epoch
- Descent: $f(\mathbf{x}^{\mathcal{K}})-f(\mathbf{x}^0)\le -O(\epsilon^{1.5})$
- Terminate in at most $O(\epsilon^{-1.5})$ epochs
- Total iterations: $O(\epsilon^{-1.5}\times\epsilon^{-1/4})=O(\epsilon^{-7/4})$

- Equal to K iterations
- Find the specific average $\hat{\mathbf{y}}$ in this epoch with small gradient:

$$\|\nabla f(\hat{\mathbf{y}})\|\le\epsilon$$

Thanks for your watching!