Stabilized SVRG: Simple Variance Reduction for Nonconvex Optimization

Xiang Wang*

Joint work with Rong Ge*, Zhize Li† and Weiyao Wang*,
*Duke University †Tsinghua University
Non-convex Optimization

- **In theory**, finding global minima is NP-Hard.
- **In practice**, just run (stochastic) gradient descent.

  All local minima are global minima, all saddle points are strict. (e.g. matrix completion [GLM16], dictionary learning [SQW17], certain objectives of neural networks [GLM17].)

Goal: find **second-order stationary points** (0 gradient and psd Hessian).
Empirical Risk Minimization

- Empirical risk minimization:
  \[ \min \text{ empirical risk} = \frac{1}{n} \sum_{i=1}^{n} (\text{risk over sample } i) \]

- Problem:
  \[ \min_{x \in \mathbb{R}^d} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\} \]

- Both \( f_i(\cdot) \) and \( f(\cdot) \) can be non-convex.

- \( f_i(x) \): risk over one sample

- \( f(x) \): empirical risk
SVRG (Stochastic Variance Reduced Gradient)

- **SGD**: $x_{t+1} = x_t - \eta \nabla f_i(x_t), \ i \sim [n]$
  Converges to an $\epsilon$-first-order stationary point ($\|\nabla f(x)\| \leq \epsilon$) in $O\left(\frac{\sigma^2}{\epsilon^4}\right)$

- **SVRG** [JZ13]: in each epoch, compute the full gradient of the first point (snapshot point) and use it to reduce variance in the following iterates

- **SVRG**: $O\left(\frac{n^{2/3}}{\epsilon^2} + n\right)$ [AH16] [RHSPS16] [LL18]

\[ x_{t+1} = x_t - \eta (\nabla f_i(x_t) - \nabla f_i(\bar{x}) + \nabla f(\bar{x})) \approx \nabla f(x_t) \]
Our Results

Theorem. We design an algorithm (Stabilized SVRG) that can find an $\epsilon$-second order stationary point using

$$\tilde{O}\left(\frac{n^{2/3}}{\epsilon^2} + \frac{n}{\epsilon^{1.5}}\right)$$

stochastic gradients.

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

1. The first simple variant of SVRG with similar guarantee.
2. Stabilization technique might be applicable to other algorithms.
Neon [XRY17] and Neon2 [AL17] can transform an algorithm that finds first-order stationary point to an algorithm with second-order guarantee.

Negative Curvature Search (NC-search)
Given a point $x$, decide if $\nabla^2 f(x) \succeq -\sqrt{\epsilon} I$ or find a unit vector $v$ such that $v^\top \nabla^2 f(x)v \leq -\frac{\sqrt{\epsilon}}{2}$

Neon2+SVRG: $\widetilde{O}\left(\frac{n^{2/3}}{\epsilon^2} + \frac{n}{\epsilon^{1.5}} + \frac{n^{3/4}}{\epsilon^{1.75}}\right)$

Adding a separate NC-search makes the algorithm complicated, which is not necessary in practice.

Without NC-Search, our algorithm is simpler!
Stabilized SVRG

- Modifications to original SVRG

At the beginning of each epoch, if the gradient is small

1. add a small perturbation to the current point
2. run SVRG on a shifted function

\[ \hat{f}(x) := f(x) - \langle \nabla f(\tilde{x}), x - \tilde{x} \rangle \]

whose gradient at initial point \( \tilde{x} \) is exactly zero.
Challenge

- GD: iterates escape along $e_1$ direction
- SVRG: initial projection along $e_1$ (only $\frac{\delta}{\sqrt{d}}$) can be canceled by the variance

Minimum eigenvalue direction of $\nabla^2 f(\tilde{x})$
Stabilization

- Variance can be bounded by the distance to the snapshot point.
- Hope the iterates stay close to the initial point for long enough time.
Two Phase Analysis

- **Phase 1**
  Bounded in a ball with radius $\tilde{O}(\delta)$
  At the end of Phase 1, the projection along $e_1$ at least $\delta/2$
  Implicit negative curvature search!

- **Phase 2**
  Starting from “a good initial point”,
  $x_t - \tilde{x}$ increases exponentially along $e_1$ direction

Minimum eigenvalue direction of $\nabla^2 f(\tilde{x})$
Summary

- **Main Result:**
  We give the first simple variant of SVRG which converges to an $\varepsilon$-second-order stationary point within $\tilde{O}\left(\frac{n^{2/3}}{\varepsilon^2} + \frac{n}{\varepsilon^{1.5}}\right)$ time.

- **Future work:**
  1. Formulate the properties that are required for the stabilization idea to work.
  2. Give a reduction that produces simpler algorithms with second order guarantees.

Poster #180