Third-order Smoothness Helps: Faster Stochastic Optimization Algorithms for Finding Local Minima

Yaodong Yu\textsuperscript{1}, Pan Xu\textsuperscript{1} and Quanquan Gu\textsuperscript{1}

\textsuperscript{1}University of Virginia \hspace{1cm} \textsuperscript{1}University of California, Los Angeles

\section*{Preliminaries}

\subsection*{Geometric Distribution}
A random integer $X$ follows a geometric distribution with parameter $p$, denoted as $\text{Geom}(p)$, if it satisfies that
\[
P(X = k) = p^k(1 - p), \quad \forall k = 0, 1, \ldots.
\]

\subsection*{Smoothness}
(First-order smoothness) A differentiable function $f$ is $L_1$-smooth, if
\[
\|\nabla f(x) - \nabla f(y)\| \leq L_1 \|x - y\|, \quad \text{for all} \ x, y \in \mathbb{R}^d.
\]

\subsection*{Hessian Lipschitz (Second-order Smoothness)}
A twice-differentiable function $f$ is $L_2$-Hessian Lipschitz, if
\[
\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_2 \|x - y\|, \quad \text{for all} \ x, y \in \mathbb{R}^d.
\]

\subsection*{Third-order Derivative}
The third derivative of function $f: \mathbb{R}^d \to \mathbb{R}$ is a three-way tensor $\nabla^3 f(x) \in \mathbb{R}^{d \times d \times d}$ which is defined as
\[
\nabla^3 f(x)_{ijk} = \frac{\partial}{\partial x_i \partial x_j \partial x_k} f(x),
\]
for $i, j, k = 1, \ldots, d$ and $x \in \mathbb{R}^d$.

\subsection*{Third-order Derivative Lipschitz}
(Third-order Smoothness) A thrice-differentiable function $f$ has $L_3$-Lipschitz third-derivative, if
\[
\|\nabla^3 f(x) - \nabla^3 f(y)\| \leq L_3 \|x - y\|, \quad \text{for all} \ x, y \in \mathbb{R}^d.
\]

\section*{Stochastic Nonconvex Optimization}

\subsection*{Optimization problem:}
\[
\min_{x \in \mathbb{R}^d} f(x) = \mathbb{E}_{\xi \sim D}[F(x; \xi)]
\]

\begin{itemize}
\item $F(x; \xi)$: $\mathbb{R}^d \to \mathbb{R}$ is a stochastic function
\item $\xi$ is a random variable sampled from a fixed distribution $D$
\item $f(x)$ is nonconvex
\end{itemize}

\section*{Why approximate local minimum?}
A local minimum is adequate and can be as good as a global minimum in terms of generalization performance.

\subsection*{Numerical Experiments}

\begin{itemize}
\item \textbf{Baseline Algorithms}
(1) stochastic gradient descent (SGD); (2) SGD with momentum (SGD-m); (3) noisy stochastic gradient descent (NSGD); (4) Stochastically Controlled Stochastic Gradient (SCSG); (5) NEG-curvevature-Originiated-from-Noise (Neon); (6) NEG-curvevature-Originiated-from-Noise 2 (Neon2).
\item \textbf{Optimization Problems}
\textbullet Matrix Sensing
\[
\min_{u \in \mathbb{R}^d} \ell(U) = \frac{1}{2m} \sum_{i=1}^{m} \|A_i - UU^Tb_i\|_F^2
\]
\textbullet Deep Autoencoder
\[
\min_{\theta \in \mathbb{R}^d} \ell(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \|x_i - f(x_i; \theta)\|_2^2
\]
\end{itemize}

\section*{Exploiting Third-order Smoothness}

Suppose $\|\nabla f(x)\| \leq \epsilon$ and $x$ is not an $(\epsilon, \epsilon_H)$-second-order stationary point, then there must exist a unit vector $\nu$ such that
\[
\nu^T \nabla^2 f(x) \nu \leq -\frac{\epsilon_H}{2}.
\]

\subsection*{Without Third-order Smoothness}
\[
\hat{y} = \arg \min_{y(x, u)} f(y), \quad u = x - \hat{\alpha} \nu, \quad w = x + \hat{\alpha} \nu,
\]
the step size $\hat{\alpha}$ can be set as $\hat{\alpha} = O(\epsilon_H/L_2)$, the negative curvature descent step is guaranteed to attain the following function value decrease
\[
f(\hat{y}) - f(x) = -O(\epsilon_H^2/L_2^2).
\]

\subsection*{With Third-order Smoothness}
\[
\tilde{y} = \arg \min_{y(x, u)} f(y), \quad u = x - \tilde{\alpha} \nu, \quad w = x + \tilde{\alpha} \nu,
\]
the step size $\tilde{\alpha}$ can be set as $\tilde{\alpha} = O(\sqrt{\epsilon_H}/L_2)$ which is larger than previous step size $\hat{\alpha}$, the negative curvature descent step is guaranteed to attain the following function value decrease
\[
f(\tilde{y}) - f(x) = -O(\epsilon_H^2/L_2^3).
\]

\section*{Theoretical Analysis}

\subsection*{Negative Curvature Descent Step}
If the input $x$ of the negative curvature algorithm (with larger step size) satisfies $\lambda_{min}(\nabla^2 f(x)) < -\epsilon_H$, then with probability $1 - \delta$, the algorithm will return $\hat{y}$ such that $E_f[f(x) - f(\hat{y})] \geq \epsilon^3_H/\delta L_2$, where $E_f$ denotes the expectation over the Rademacher random variable $\xi$. Furthermore, if we choose $\delta \leq \epsilon_H/(\epsilon_H + 8L_2)$, it holds that
\[
E[f(\hat{y}) - f(x)] \leq \frac{\epsilon^3_H}{8L_2},
\]
where $E$ is over all randomness of the algorithm, and the total runtime complexity is $\tilde{O}(\mathcal{L}/\epsilon^3_H)$.

\subsection*{Total Runtime Complexity Analysis}
Let $f(x) = \mathbb{E}_{\xi \sim D}[F(x; \xi)]$, suppose the third derivative of $f(x)$ is $L_3$-Lipschitz and each stochastic function $F(x; \xi)$ is $L_3$-smooth and $L_2$-Hessian Lipschitz continuous. Suppose that the stochastic gradient $\nabla F(x; \xi)$ satisfies the gradient sub-Gaussian condition with parameter $\sigma$. Set batch size $B = O(\sigma^2/\epsilon_H^2)$ and $\epsilon_H \geq 2/13$. If our algorithm FLASH adopts online algorithms, such as Oja’s algorithm, to compute the negative curvature, then FLASH finds an $(\epsilon, \epsilon_H)$-second-order stationary point with probability at least $1/3$ in runtime
\[
\tilde{O}\left(\frac{L_2\sigma^2/\epsilon_H^2}{\epsilon_H} + \frac{L_3\sigma^2/\epsilon_H^2}{\epsilon^2_H} + \frac{L_1^2L_2\Delta_1}{\epsilon_H}\right).
\]