Stochastic Newton and Cubic Newton Methods with Simple Local Linear-Quadratic Rates

Dmitry Kovalev Konstantin Mishchenko Peter Richtárik

King Abdullah University of Science and Technology (KAUST)

Problem

We want to solve the **finite-sum optimization** problem

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(x). \quad (1)$$

- Problem (1) has many applications in machine learning, data science and engineering.
- We focus on the regime when n is very large. This is typically the case in big data settings (e.g., massively distributed and federated learning).

Assumptions

- Each f_i is μ-strongly convex ($\mu > 0$), i.e.,
$$\mu I_d \preceq \nabla^2 f_i(x), \ \forall x \in \mathbb{R}^d,$$
- Each f_i has H-Lipschitz Hessian, i.e.,
$$\|\nabla^2 f_i(x) - \nabla^2 f_i(y)\| \le H\|x - y\|, \ \forall x, y \in \mathbb{R}^d.$$

Our Contribution

We develop two new simple and fundamental stochastic second-order methods:

- Stochastic Newton (SN) method,
- Stochastic Cubic Newton (SCN) method.

Our methods have highly desirable properties:

- Cost of 1 iteration does not depend on n,
- (Local) convergence rate does not depend on the conditioning of the problem.

Motivation I: The Curse of 1st Order Methods

In this regime, the state of the art methods for solving (1) are variants of **stochastic gradient descent.** However, the performance of all first order methods depends heavily on the **conditioning** of the problem. Various strategies have been proposed to address this problem:

- preconditioning (e.g., data normalization, layer and batch normalization),
- momentum (e.g., Polyak and Nesterov),
- adaptive stepsizes (e.g., Adagrad, ADAM, Barzilai-Borwein, Malitsky-Mishchenko) and line search,
- minibatching and importance sampling.

Some of these methods reduce the effect of conditioning provably, and some are heuristics which often work but sometimes fail. However, first order methods are inherently incapable to remove the effect of conditioning.

Newton's Method

It will be useful to recall classical **Newton's method:**

$$x^{k+1} = x^k - \left[\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(x^k)\right]^{-1} \nabla f(x^k) \quad (2)$$
$$= \left[\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(x^k)\right]^{-1} \left[\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(x^k) x^k - \nabla f(x^k)\right].$$

Fact. If f is μ-strongly convex and its Hessian is Lipschitz continuous, then if $\|x^0 - x^*\|$ is small enough (where $x^* = \arg\min f$), then the iterates (2) of Newton's method converge to x^* quadratically: $\|x^{k+1} - x^*\| \le \frac{1}{2}\|x^k - x^*\|^2$. This means that

$$k \ge \mathcal{O}\left(\log_2 \log_2 \frac{1}{\varepsilon}\right) \Rightarrow \|x^k - x^*\| \le \varepsilon.$$

Motivation II: Issues with Existing Stochastic 2nd Order Methods

Because of what we said above, there is **a lot of effort to develop efficient stochastic 2nd order methods.**

Almost every such method has the form

$$x^{k+1} = x^k - (H^k)^{-1} g^k,$$

where $H^k \approx \nabla^2 f(x^k)$ is a stochastic approximation of the Hessian and $g^k \approx \nabla f(x^k)$ is a stochastic gradient approximation of the gradient. Most methods let

$$g^k = \frac{1}{|S_g^k|} \sum_{i \in S_g^k} \nabla f_i(x^k), \quad H^k = \frac{1}{|S_H^k|} \sum_{i \in S_H^k} \nabla^2 f_i(x^k),$$

where S_H^k and S_g^k are suitably chosen random subsets of $\{1, 2, \ldots, n\}$. However, all these methods suffer from severe issues:

- they require $\mathcal{O}(\epsilon^{-1})$ or even $\mathcal{O}(\epsilon^{-2})$ samples **in each iteration**, where ϵ is the target accuracy. The number of required samples often exceeds n in theory,
- the resulting convergence rate is **worse** than the rate of first order methods.

New Algorithm: Stochastic Newton

Algorithm 1: Stochastic Newton (SN)

Initialize: Choose $w_1^0, w_2^0, \ldots, w_n^0 \in \mathbb{R}^d$ and $\tau \in \{1, 2, \ldots, n\}$

for $k = 0, 1, \ldots$ **do**

Compute Hessian estimator: $H^k = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(w_i^k)$;

Compute gradient estimator: $g^k = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_i^k)$;

$x^{k+1} = \left[H^k\right]^{-1} \left[\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(w_i^k) w_i^k - g^k\right]$;

Choose a random set $S^k \subseteq \{1, \ldots, n\}$ of cardinality τ;

Update

$$w_i^{k+1} = \begin{cases} x^{k+1}, & i \in S^k \\ w_i^k, & i \notin S^k \end{cases}$$

end

Theorem 1 (Stochastic Newton)

The random iterates of SN (Algorithm 1) satisfy the recursion

$$\mathbb{E}_k\left[\mathcal{W}^{k+1}\right] \le \left(1 - \frac{\tau}{n} + \frac{\tau}{n}\left(\frac{H}{2\mu}\right)^2 \mathcal{W}^k\right)\mathcal{W}^k,$$

where $\mathcal{W}^k \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\left\|w_i^k - x^\star\right\|^2$. Furthermore, if $\|w_i^0 - x^\star\| \le \frac{\mu}{H}$ for $i = 1, \ldots, n$, then

$$\mathbb{E}_k\left[\mathcal{W}^{k+1}\right] \le \left(1 - \frac{3\tau}{4n}\right)\mathcal{W}^k.$$

New Algorithm: Stochastic Cubic Newton

Motivation: Newton method converges only locally. To fix it, one needs cubic regularization:

$$\Phi^k(x) \stackrel{\text{def}}{=} \left\langle \nabla f(x^k), x - x^k \right\rangle + \frac{1}{2}\|x - x^k\|^2_{\nabla^2 f(x^k)}$$
$$x^{k+1} = \operatorname*{argmin}_x \left\{\Phi^k(x) + \frac{M}{6}\|x - x^k\|^3\right\}.$$

Our algorithm

$$\Psi^k \stackrel{\text{def}}{=} \left\langle \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_i^k), x - x^k \right\rangle + \frac{1}{2n}\sum_{i=1}^{n}\|x - w_i^k\|^2_{\nabla^2 f_i(w_i^k)}$$
$$x^{k+1} = \operatorname*{argmin}_x \left\{\Psi^k(x) + \frac{M}{6n}\sum_{i=1}^{n}\|x - w_i^k\|^3\right\}.$$

Theorem 2 (Stochastic Cubic Newton)

The random iterates of SCN satisfy the recursion

$$\mathbb{E}_k\left[\mathcal{V}^{k+1}\right] \le \left(1 - \frac{\tau}{n} + \frac{\tau}{n}\left(\frac{(M+H)\sqrt{2}}{3\mu^{\frac{3}{2}}}\right)^{3/2}\sqrt{\mathcal{V}^k}\right)\mathcal{V}^k,$$

where $\mathcal{V}^k \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\left(f(w_i^k) - f(x^\star)\right)^{\frac{3}{2}}$. Furthermore, if the vectors w_i^k are close enough to x^*, then

$$\mathbb{E}_k\left[\mathcal{V}^{k+1}\right] \le \left(1 - \frac{\tau}{2n}\right)\mathcal{V}^k.$$

Experiments

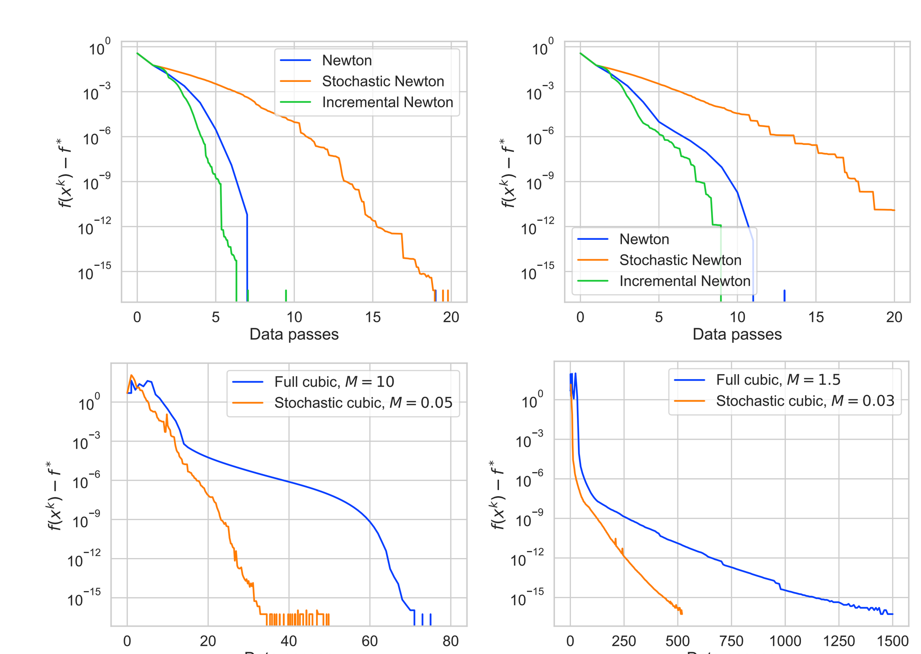

Figure 1: Logistic regression. Top: Newton methods, bottom: cubic Newton methods. Left: $\mu = \frac{1}{100n}$, right: $\mu = \frac{1}{10000n}$.