OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Stochastic Variance-Reduced Newton: Accelerating Finite-Sum Minimization with Large Batches

Michał Dereziński DEREZIN@UMICH.EDU
University of Michigan

Abstract

Stochastic variance reduction has proven effective at accelerating first-order algorithms for solving convex finite-sum optimization tasks such as empirical risk minimization. Yet, the benefits of variance reduction for first-order methods tend to vanish in the large-batch setting, i.e., when stochastic gradients are computed from very large mini-batches to leverage parallelization of modern computing architectures. On the other hand, incorporating second-order information via Newton-type methods has proven successful in improving the performance of large-batch algorithms. In this work, we show that, in the presence of second-order information, variance reduction in the gradient can provide significant convergence acceleration even when using extremely large-batch gradient estimates. To demonstrate this, we study a finite-sum minimization algorithm we call Stochastic Variance-Reduced Newton (SVRN). We show that SVRN provably accelerates existing stochastic Newton-type methods (such as Subsampled Newton), while retaining their parallelizable large-batch operations: The number of passes over the data is reduced from $O(\alpha \log(1/\epsilon))$ to $O\big(\frac{\log(1/\epsilon)}{\log(n)}\big)$, i.e., by a factor of $O(\alpha \log(n))$, where n is the number of sum components and α is the approximation factor in the Hessian estimate. Surprisingly, this acceleration gets more significant the larger the data size n, and can be achieved with a unit step size.

1. Introduction

Consider a convex finite-sum minimization task:

$$\text{find} \quad \mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \quad \text{for} \quad f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \psi_i(\mathbf{x}). \tag{1}$$

This optimization task naturally arises in machine learning through empirical risk minimization, where $\mathbf{x}$ is the model parameter vector and each function $\psi_i(\mathbf{x})$ corresponds to the loss incurred by the model on the i-th element in a training data set (e.g., square loss for regression, or logistic loss for classification). Many other optimization tasks, such as solving semi-definite programs and portfolio optimization, can be cast in this general form. Our goal is to find an ϵ-approximate solution, i.e., $\tilde{\mathbf{x}}$ such that $f(\tilde{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon$.

Naturally, one can use classical iterative optimization methods for this task (such as gradient descent and Newton's method), which use first/second-order information of function f to construct a sequence $\mathbf{x}_0, \mathbf{x}_1, ...$ that converges to $\mathbf{x}^*$. However, this does not leverage the finite-sum structure of the problem. Thus, extensive literature has been dedicated to efficiently solving finite-sum minimization tasks using stochastic optimization methods, which use first/second-order information of randomly sampled component functions ψ_i, that can often be computed much faster than the entire function f. Among first-order methods, variance-reduction techniques such as SAG [41], SDCA

[43], SVRG [24], SAGA [10], Katyusha [2] and others [3, 19, 25], have proven particularly effective. One of the most popular variants of this approach is Stochastic Variance-Reduced Gradient (SVRG), which achieves variance reduction by combining frequent stochastic gradient queries with occasional full batch gradient queries, to optimize the overall cost of finding an ϵ-approximate solution, where the cost is measured by the total number of queries to the components $\nabla\psi_i(\mathbf{x})$. Yet, the benefits of variance reduction in first-order methods diminish dramatically as we start using stochastic gradients with large mini-batches, which is a common practice on modern computing architectures.

Many stochastic second-order methods have also been proposed for solving finite-sum minimization, including Subsampled Newton [5, 6, 17, 40], Newton Sketch [12, 36, 37], and others [26, 32, 33, 44]. These approaches are generally less sensitive to hyperparameters such as the step size, and they typically query larger random batches of component gradients/Hessians at a time, as compared to stochastic first-order methods. The larger queries make these methods less sequential, allowing for more effective vectorization and parallelization.

A number of works have explored whether second-order information can be used to accelerate stochastic variance-reduced methods, resulting in several algorithms such as Preconditioned SVRG [20], SVRG2 [22] and others [21, 28]. However, due to their use of small-batch stochastic gradients, these are still primarily stochastic first-order methods, highly sequential and with a problem-dependent step size. Comparatively little work has been done on using variance reduction to accelerate stochastic large-batch Newton-type methods for convex finite-sum minimization (see discussion in Section 2.1 and Appendix A). To that end, we ask:

> *Can variance reduction accelerate local convergence of large-batch Newton-type methods in convex finite-sum minimization?*

We show that the answer to this question is positive. We demonstrate this with a method we call Stochastic Variance-Reduced Newton (SVRN), which retains the positive characteristics of second-order methods, including easily parallelizable large-batch gradient queries as well as minimal hyperparameter tuning (e.g., accepting a unit step size). We prove that, when the number of components ψ_i is sufficiently large, SVRN achieves a better parallel complexity than SVRG, and a better sequential complexity than the corresponding Stochastic Newton method (see Table 1).

2. Main result

In this section, we present our main result, which is the parallel and sequential complexity analysis of the local convergence of SVRN. A practical implementation of the method is given in Algorithm 1, whereas the detailed convergence analysis is given in Appendix B.

We now present the assumptions needed for our main result, starting with μ-strong convexity of f and λ-smoothness of each ψ_i. These are standard for establishing linear convergence rate of SVRG. Our result also requires Hessian regularity assumptions (Definition 4), which are standard for Newton's method and only affect the size of the local convergence neighborhood.

Assumption 1 *We assume that $f(x) = \frac{1}{n}\sum_{i=1}^{n}\psi_i(x)$ has continuous first and second derivatives, as well as a bounded condition number $\kappa = \lambda/\mu$, where μ is the strong convexity of f and λ is the maximum smoothness of ψ_i's.*

To highlight the parallelizability of SVRN due to large mini-batches, as well as the effect of variance reduction on its performance, we will consider two standard complexity measures:

1. **Parallel complexity:** Number of times the algorithm computes a gradient at an iterate, either over the full batch or a mini-batch. This corresponds to the standard PRAM model.

2. **Sequential complexity:** Number of passes over the data. One pass corresponds to n queries of component gradients $\nabla\psi_i(\mathbf{x})$, possibly at different iterates. This is a standard measure of complexity for stochastic first-order methods.

Our goal is to design a highly parallelizable algorithm, so in our analysis, we will first optimize over parallel complexity, and then over sequential complexity. Note that for any algorithm using only full gradients (such as the standard versions of gradient descent or Subsampled Newton), the two notions of complexity are exactly equivalent. For example, in gradient descent (GD), both parallel and sequential complexity is $O(\kappa\log(1/\epsilon))$. By introducing stochastic gradients and variance reduction, as in SVRG, we can improve upon the sequential complexity of GD, while preserving (but not improving) its parallel complexity. Specifically, when $n \gg \kappa$, then SVRG with optimal mini-batch size takes $O(\log(1/\epsilon))$ sequential time to find an ϵ-approximate solution, however it still needs $O(\kappa\log(1/\epsilon))$ parallel time (Table 1).

We can avoid the dependence of parallel complexity on the condition number κ by using second-order information. In particular, suppose that in each iteration we compute the full gradient $\nabla f(\mathbf{x})$ and are given access to a (typically stochastic) Hessian estimate $\tilde{\mathbf{H}}$ such that for some $1 \leq \alpha \ll \kappa$,

$$\text{(Hessian } \alpha\text{-approximation)} \qquad \frac{1}{\sqrt{\alpha}}\nabla^2 f(\mathbf{x}) \preceq \tilde{\mathbf{H}} \preceq \sqrt{\alpha}\,\nabla^2 f(\mathbf{x}), \tag{2}$$

Then, a standard Stochastic Newton (SN) update, given below in (3), can achieve parallel and sequential complexity of $O(\alpha\log(1/\epsilon))$ locally in the neighborhood of the optimum. It is thus natural to ask whether we can use stochastic gradients and variance reduction to accelerate the local sequential complexity of this method, while preserving its parallel complexity. Our main result shows not only that this is possible, but also, remarkably, that this acceleration gets more significant the larger the data size n. See also Theorem 5 for algorithmic details and convergence analysis.

Theorem 1 *Suppose that Assumption 1 holds and: (a) f has a Lipschitz Hessian, or (b) f is self-concordant. Moreover, let $n \gg \kappa \gg \alpha$. There is an algorithm (SVRN, see Algorithm 1) and an open neighborhood U such that, given any $\mathbf{x} \in U$ with a corresponding Hessian α-approximation as in (2), the cost of returning $\tilde{\mathbf{x}}$ such that $f(\tilde{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon \cdot (f(\mathbf{x}) - f(\mathbf{x}^*))$ is as follows:*

$$\text{Parallel complexity} = O\big(\alpha\log(1/\epsilon)\big) \quad \textit{and} \quad \text{Sequential complexity} = O\Big(\frac{\log(1/\epsilon)}{\log(n)}\Big).$$

Remark 2 *We can construct an $\alpha = O(1)$ Hessian approximation at a one-time cost of $O(\kappa\log(d))$ component Hessian queries [40], but it is often possible to use far fewer queries, or other estimation methods [5, 6, 12, 17, 37]. All of these are covered by our condition (2), possibly with a larger problem-dependent α. We illustrate this in Appendix C, showing that a version of SVRN with a sketched Hessian estimate [36] and leverage score sampled gradients [15] improves on the state-of-the-art complexity for reaching a high-precision solution to least squares regression.*

Remark 3 *In Section 3, we propose a globally convergent algorithm based on SVRN (SVRN-HA; see Algorithm 1), and we show empirically that it substantially accelerates Subsampled Newton.*

	Variance-reduced	Second-order	Parallel time	Sequential time	Speed-up
Gradient Descent	X	X	$O(\kappa \log(1/\epsilon))$	$O(\kappa \log(1/\epsilon))$	X
Accelerated GD	X	X	$O(\sqrt{\kappa} \log(1/\epsilon))$	$O(\sqrt{\kappa} \log(1/\epsilon))$	X
Stochastic Newton	X	✓	$O(\alpha \log(1/\epsilon))$	$O(\alpha \log(1/\epsilon))$	X
Mini-batch SVRG	✓	X	$O(\kappa \log(1/\epsilon))$	$O(\log(1/\epsilon))$	$O(\kappa)$
Mini-batch Katyusha	✓	X	$O(\sqrt{\kappa} \log(1/\epsilon))$	$O(\log(1/\epsilon))$	$O(\sqrt{\kappa})$
SVRN (this work)	✓	✓	$O(\alpha \log(1/\epsilon))$	$O(\frac{\log(1/\epsilon)}{\log(n)})$	$O(\alpha \log(n))$

Table 1: Comparison of local convergence behavior for SVRN and related stochastic methods in the big data regime, i.e., $n \gg \kappa$, along with full-batch Gradient Descent (GD), and Accelerated GD. Time complexities are obtained by first optimizing parallel time, and then optimizing sequential time. For second-order, we use a Hessian α-approximation (2) where $\alpha \ll \kappa$.

2.1. Discussion

In this section, we compare the local convergence complexity of SVRN to standard stochastic first-order and second-order algorithms, with further discussion of related works [20, 26–28, 38, 46, 49, 50] relegated to Appendix A. In this comparison, we focus on what we call the big data regime, i.e., $n \gg \kappa$, which is of primary interest in the literature on Subsampled Newton methods.

Comparison to SVRG and Katyusha. As we can see in Table 1, first-order algorithms, including variance-reduced methods such as SVRG [24], and its accelerated variants like Katyusha [2], suffer from a dependence on the condition number κ in their parallel complexity. Namely, they require either $O(\kappa \log(1/\epsilon))$ or $O(\sqrt{\kappa} \log(1/\epsilon))$ batch gradient queries, compared to $O(\alpha \log(1/\epsilon))$ for SVRN and Stochastic Newton, where α is the Hessian approximation factor, which is often much smaller than $\sqrt{\kappa}$. This is because, unlike SVRN, these methods do not scale well to large mini-batches, making them less parallelizable.

Another difference between SVRN and SVRG or Katyusha is that, when the Hessian approximation is sufficiently accurate ($\alpha \leq 2$), then SVRN accepts a unit step size, which leads to optimal convergence rate without any tuning. On the other hand, the optimal step size for SVRG depends on the strong convexity and smoothness constants μ and λ, and thus, requires tuning.

Comparison to Stochastic Newton. We next compare SVRN with Stochastic Newton methods such as Subsampled Newton and Newton Sketch. Here, the most standard proto-algorithm considered in the literature is the following update:

$$\tilde{\mathbf{x}}_{s+1} = \tilde{\mathbf{x}}_s - \eta_s \tilde{\mathbf{H}}^{-1} \nabla f(\tilde{\mathbf{x}}_s). \tag{3}$$

As mentioned earlier, this update uses only full gradients, so both its parallel and sequential complexity is $O(\alpha \log(1/\epsilon))$ (see Stochastic Newton in Table 1). On the other hand, SVRN provides a direct acceleration of the sequential complexity without sacrificing any parallel complexity. Remarkably, if we wanted to recover SVRN's sequential complexity of $O(\frac{\log(1/\epsilon)}{\log(n)})$ by improving the Hessian approximation in either Subsampled Newton or Newton Sketch, the approximation cost would be as large as computing the exact Hessian (i.e., Newton's method), which is highly undesirable.

Algorithm 1: SVRN with Hessian Averaging (SVRN-HA)

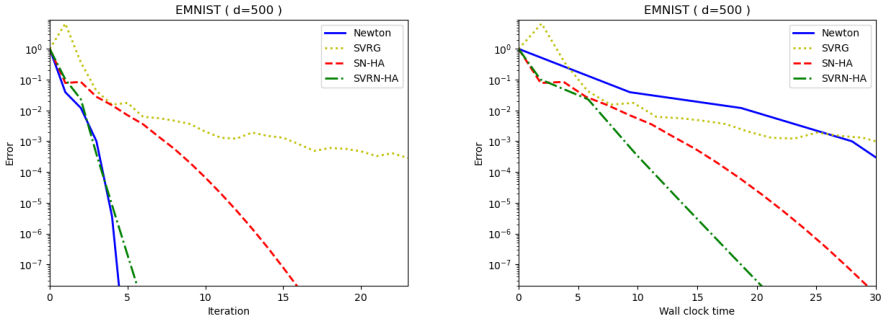
Input: iterate $\tilde{\mathbf{x}}_0$, gradient batch size m, Hessian sample size k, and local iterations $t_{\max}$
Initialize step size $\eta_{-1} = 0$ and Hessian estimate $\tilde{\mathbf{H}}_{-1} = \mathbf{0}$
for $s = 0, 1, 2, \ldots$ **do**
  Compute the subsampled Hessian: $\widehat{\mathbf{H}}_s = \frac{1}{k}\sum_{i=1}^{k} \nabla^2 \psi_i(\tilde{\mathbf{x}}_s)$, for $\psi_1, ..., \psi_k \sim \mathcal{D}$
  Compute the Hessian average: $\tilde{\mathbf{H}}_s = \frac{s}{s+1}\tilde{\mathbf{H}}_{s-1} + \frac{1}{s+1}\widehat{\mathbf{H}}_s$
  Compute the full gradient: $\tilde{\mathbf{g}}_s = \nabla f(\tilde{\mathbf{x}}_s)$
  if $\eta_{s-1} < 1$ **then**
    Compute the descent direction $\tilde{\mathbf{v}}_s$ by solving: $\tilde{\mathbf{H}}_s \tilde{\mathbf{v}}_s = -\tilde{\mathbf{g}}_s$ *global phase — Stochastic Newton —*
  else
    Initialize $\mathbf{x}_0 = \tilde{\mathbf{x}}_s$ *local phase — Begin SVRN —*
    for $t = 0, \ldots, t_{\max} - 1$ **do**
      Compute $\widehat{\mathbf{g}}_t(\mathbf{x}_t)$ and $\widehat{\mathbf{g}}_t(\tilde{\mathbf{x}}_s)$, for $\widehat{\mathbf{g}}_t(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} \nabla \psi_i(\mathbf{x})$, $\psi_1, ..., \psi_m \sim \mathcal{D}$
      Compute variance-reduced gradient $\bar{\mathbf{g}}_t = \widehat{\mathbf{g}}_t(\mathbf{x}_t) - \widehat{\mathbf{g}}_t(\tilde{\mathbf{x}}_s) + \tilde{\mathbf{g}}_s$
      Compute the descent direction $\mathbf{v}_t$ by solving: $\tilde{\mathbf{H}}_s \mathbf{v}_t = -\bar{\mathbf{g}}_t$
      Update $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t$
    end
    Compute the descent direction: $\tilde{\mathbf{v}}_s = \mathbf{x}_{t_{\max}} - \tilde{\mathbf{x}}_s$ *— End SVRN —*
  end
  Compute η_s for iterate $\tilde{\mathbf{x}}_s$ and direction $\tilde{\mathbf{v}}_s$ using the Armijo condition
  Update $\tilde{\mathbf{x}}_{s+1} = \tilde{\mathbf{x}}_s + \eta_s \tilde{\mathbf{v}}_s$
end

3. Experiments

We next demonstrate numerically that SVRN can be effectively used to accelerate stochastic Newton methods in practice. We also show how variance reduction can be incorporated into a globally convergent Subsampled Newton method in a way that is robust to hyperparameters and preserves its scalability thanks to large-batch operations. To do this, we consider a practical implementation of SVRN (see Algorithm 1, called SVRN-HA). In the initial (global) phase, this algorithm is a variant of Subsampled Newton that maintains a Hessian estimate by aggregating sampled component Hessians from all previous iterations (a.k.a. Hessian Averaging) [34]. At each iteration, we check whether the last line search returned a unit step size. If yes, then we enter local phase by running SVRN with local iterations $t_{\max} = \lfloor \log_2(n/d) \rfloor$ and gradient batch size $m = \lfloor n/\log_2(n/d) \rfloor$ (i.e., one data pass), where n is the data size and d is the dimension. We use Hessian sample size $k = 4d$.

Next, we present numerical experiments for solving a logistic loss minimization task on the EMNIST dataset [9] (with $n \approx 500$k data points) transformed using a random features map with dimension $d = 1000$. Further results on the CIFAR-10 dataset, as well as on synthetic data matrices, are presented in Appendix E. In Figure 1, we compared SVRN-HA to three baselines which are most directly comparable: (1) the classical Newton's method; (2) SVRG with the step size and number of inner iterations tuned for best wall-clock time; and (3) Subsampled Newton with Hessian Averaging (SN-HA), i.e., the Newton-type method we use in the global phase of Algorithm 1 (without the SVRN phase). All of the convergence plots are averaged over multiple runs.

From Figure 1(a), we conclude that as soon as SVRN-HA exits the initial phase of the optimization, it accelerates dramatically, to the point where it nearly matches the rate of classical Newton. This acceleration corresponds to the improvement in sequential complexity from $O(\alpha \log(1/\epsilon))$ for Stochastic Newton to $O(\frac{\log(1/\epsilon)}{\log(n)})$ for SVRN. Finally, the convergence of SVRG is initially quite fast, but over time, it stabilizes at a slower rate, indicating that the Hessian information plays a significant role in the performance of SVRN-HA.



(*a*) Convergence (*b*) Runtime

Figure 1: Convergence and runtime comparison of SVRN-HA on the EMNIST dataset against three baselines: classical Newton, SVRG (after parameter tuning), and Subsampled Newton with Hessian Averaging (SN-HA), i.e., the global phase of Algorithm 1, ran without switching to SVRN. Further results on the CIFAR-10 dataset are in Appendix E.

In Figure 1(*b*), we plot the wall clock time of the algorithms. Here, SVRN-HA also performs better than all of the baselines, despite some additional per-iteration overhead. We expect that this can be further optimized. Finally, we note that Newton's method is drastically slower than all other methods due to the high cost of solving a large linear system, and the per-iteration time of SVRG is substantially slowed by its sequential nature.

4. Conclusions

We propose Stochastic Variance-Reduced Newton (SVRN), a simple and provably effective strategy of incorporating variance reduction into popular stochastic Newton methods for solving finite-sum minimization tasks. We show that SVRN improves the local convergence complexity of Subsampled Newton (per data pass) from $O(\alpha \log(1/\epsilon))$ to $O\big(\frac{\log(1/\epsilon)}{\log(n)}\big)$, while retaining all the benefits of second-order optimization, such as a simple unit step size and easily scalable large-batch operations.

References

[1] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[2] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

[3] Zeyuan Allen-Zhu and Yang Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *International conference on machine learning*, pages 1080–1089. PMLR, 2016.

[4] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

[5] Albert S Berahas, Raghu Bollapragada, and Jorge Nocedal. An investigation of newton-sketch and subsampled newton methods. *Optimization Methods and Software*, 35(4):661–680, 2020.

[6] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2), 2018.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Kenneth L. Clarkson and David P. Woodruff. Low-rank approximation and regression in input sparsity time. *J. ACM*, 63(6):54:1–54:45, January 2017.

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[10] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[11] Michał Dereziński and Michael W Mahoney. Determinantal point processes in randomized numerical linear algebra. *Notices of the American Mathematical Society*, 68(1):34–45, 2021.

[12] Michał Dereziński, Jonathan Lacotte, Mert Pilanci, and Michael W Mahoney. Newton-LESS: Sparsification without trade-offs for the sketched newton update. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] Michał Dereziński, Zhenyu Liao, Edgar Dobriban, and Michael W Mahoney. Sparse sketches with small inversion bias. In *Proceedings of the 34th Conference on Learning Theory*, 2021.

[14] Petros Drineas and Michael W. Mahoney. RandNLA: Randomized numerical linear algebra. *Communications of the ACM*, 59:80–90, 2016.

[15] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136, 2006.

[16] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13(1): 3475–3506, December 2012.

[17] Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. *Advances in Neural Information Processing Systems*, 28:3052–3060, 2015.

[18] Gerald Folland. *Advanced calculus*. Upper Saddle River, NJ : Prentice Hall, 2002.

[19] Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, pages 728–763. PMLR, 2015.

[20] Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz. Solving ridge regression using sketched preconditioned svrg. In *International conference on machine learning*, pages 1397–1405. PMLR, 2016.

[21] Robert Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block bfgs: Squeezing more curvature out of data. In *International Conference on Machine Learning*, pages 1869–1878. PMLR, 2016.

[22] Robert Gower, Nicolas Le Roux, and Francis Bach. Tracking the gradients using the hessian: A new look at variance reducing stochastic methods. In *International Conference on Artificial Intelligence and Statistics*, pages 707–715. PMLR, 2018.

[23] Robert Jerrard. Multivariable calculus. https://www.math.toronto.edu/courses/mat237y1/20189/notes/Contents.html, 2018. Course Notes.

[24] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

[25] Jakub Konecný, Jie Liu, Peter Richtárik, and Martin Takác. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2): 242–255, 2015.

[26] Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtárik. Stochastic newton and cubic newton methods with simple local linear-quadratic rates. *arXiv preprint arXiv:1912.01597*, 2019.

[27] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. *Advances in neural information processing systems*, 28, 2015.

[28] Yanli Liu, Fei Feng, and Wotao Yin. Acceleration of svrg and katyusha x by inexact preconditioning. In *International Conference on Machine Learning*, pages 4003–4012. PMLR, 2019.

[29] Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 91–100, 2013.

[30] Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. LSRN: A parallel iterative solver for strongly over-or underdetermined systems. *SIAM Journal on Scientific Computing*, 36(2):C95–C118, 2014.

[31] Stanislav Minsker. On some extensions of bernstein's inequality for self-adjoint operators. *Statistics & Probability Letters*, 127:111–119, 2017.

[32] Aryan Mokhtari, Mark Eisen, and Alejandro Ribeiro. Iqn: An incremental quasi-newton method with local superlinear convergence rate. *SIAM Journal on Optimization*, 28(2):1670–1698, 2018.

[33] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

[34] Sen Na, Michał Dereziński, and Michael W Mahoney. Hessian averaging in stochastic newton methods achieves superlinear convergence. *Mathematical Programming*, 2022.

[35] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pages 117–126. IEEE, 2013.

[36] Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17 (1):1842–1879, 2016.

[37] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

[38] Anton Rodomanov and Dmitry Kropotov. A superlinearly-convergent proximal newton-type method for the optimization of finite sums. In *International Conference on Machine Learning*, pages 2597–2605. PMLR, 2016.

[39] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

[40] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods. *Mathematical Programming*, 174(1):293–326, 2019.

[41] Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

[42] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 143–152, Washington, DC, USA, 2006. IEEE Computer Society.

[43] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(2), 2013.

[44] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic cubic regularization for fast nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

[45] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, August 2012.

[46] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[47] David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[48] Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W Mahoney. Weighted sgd for lp regression with randomized preconditioning. *The Journal of Machine Learning Research*, 18 (1):7811–7853, 2017.

[49] Junyu Zhang, Lin Xiao, and Shuzhong Zhang. Adaptive stochastic variance reduction for subsampled newton method with cubic regularization. *INFORMS Journal on Optimization*, 4 (1):45–64, 2022.

[50] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic variance-reduced cubic regularization methods. *J. Mach. Learn. Res.*, 20(134):1–47, 2019.

Appendix A. Further related work

As mentioned in Section 1, a number of works have aimed to accelerate first-order variance reduction methods by preconditioning them with second-order information. For example, [20] proposed Preconditioned SVRG for ridge regression. The effect of this preconditioning, as in related works [28], is a reduced condition number κ of the problem. This is different from Theorem 1, which uses preconditioning to make variance reduction effective for large mini-batches and with a unit step size.

Some works have shown that, instead of preconditioning, one can use momentum to accelerate variance reduction, and also to improve its convergence rate when using mini-batches. These methods include Catalyst [27] and Katyusha [2]. However, unlike SVRN, these approaches are still limited to fairly small mini-batches, as demonstrated in Table 1.

A number of works have proposed applying techniques inspired by variance reduction to stochastic Newton-type methods in settings which are largely incomparable to ours. First, [26, 38] consider algorithms where the Hessian and gradient information is incrementally updated with either individual samples or mini-batches. However, the approximate Hessian information required by these methods is quite different than the one used in SVRN: they require the Hessian estimate to be initialized with *all* n component Hessians, possibly computed at different locations (compared to, e.g., a subsampled estimate). For example, in the case of least squares, this means computing the exact Hessian, which costs $O(nd^2)$ time and renders the task trivial (compare this to our Theorem 7, where the Hessian estimate required by SVRN can be approximated efficiently). Setting this aside, we can still compare the local convergence rate of SVRN with the Stochastic Newton method of [26, Theorem 1] using the same mini-batch size. Assuming $n \gg \kappa$ and using the setup from Theorem 1, their method achieves $O(\log(1/\epsilon))$ sequential complexity, whereas SVRN obtains the accelerated complexity of $O\big(\frac{\log(1/\epsilon)}{\log(n)}\big)$ data passes.

In the non-convex setting, variance reduction was used by [49, 50] to accelerate Subsampled Newton with cubic regularization. They use variance reduction both for the gradient and the Hessian estimates. Also, [46] incorporate variance reduction into a stochastic quasi-Newton method. However, due to the non-convex setting, these results are incomparable to ours, as we are focusing on strongly convex optimization.

Appendix B. Convergence analysis of SVRN

In this section, we present the convergence analysis for SVRN, leading to the proof of Theorem 1.

Notation. For $d \times d$ positive semidefinite matrices $\mathbf{A}$ and $\mathbf{B}$, we define $\|\mathbf{v}\|_{\mathbf{A}} = \sqrt{\mathbf{v}^\top \mathbf{A}\mathbf{v}}$, and we say that $\mathbf{A} \approx_\epsilon \mathbf{B}$, when $(1-\epsilon)\mathbf{B} \preceq \mathbf{A} \preceq (1+\epsilon)\mathbf{B}$, where $\preceq$ denotes the positive semidefinite ordering (we define analogous notation $a \approx_\epsilon b$ for non-negative scalars a, b). We use c and C to denote positive absolute constants, and let $I \sim [n]$ denote a uniformly random sample from $\{1, .., n\}$.

We will present the analysis in a slightly more general setting of expected risk minimization, i.e., where $f(\mathbf{x}) = \mathbb{E}_{\psi\sim\mathcal{D}}[\psi(\mathbf{x})]$. Here, $\mathcal{D}$ is a distribution over convex functions $\psi : \mathbb{R}^d \to \mathbb{R}$. Clearly, this setting subsumes (1), since we can let $\mathcal{D}$ be a uniformly random sample ψ_i. Thanks to this extension, our results can apply to *importance* sampling of component functions, as in Theorem 7.

Definition 4 *We define the local convergence neighborhood $U_f(\epsilon)$, parameterized by $\epsilon \in (0,1)$, as:*

1. *If f has an L-Lipschitz Hessian, then $U_f(\epsilon) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\|_{\nabla^2 f(\mathbf{x}^*)} < \epsilon\, \mu^{3/2}/L\}$;*
2. *If f is self-concordant, then we use $U_f(\epsilon) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\|_{\nabla^2 f(\mathbf{x}^*)} < \epsilon/4\}$.*

Our local convergence analysis is captured by the following theorem, which provides the rate of convergence after one outer iteration of SVRN (stated below), for a range of mini-batch sizes m.

Theorem 5 (Convergence rate of SVRN) *Suppose that Assumption 1 holds, $\alpha \geq 1$, and either: (a) f has a Lipschitz continuous Hessian, or (b) f is self-concordant. There is an absolute constant $c > 0$ such that if $\tilde{\mathbf{x}}_s \in U_f(1/c\alpha)$, and we are given the gradient $\tilde{\mathbf{g}}_s = \nabla f(\tilde{\mathbf{x}}_s)$ as well as a Hessian α-approximation, i.e., $\tilde{\mathbf{H}}$ such that $\frac{1}{\sqrt{\alpha}}\nabla^2 f(\tilde{\mathbf{x}}_s) \preceq \tilde{\mathbf{H}} \preceq \sqrt{\alpha}\,\nabla^2 f(\tilde{\mathbf{x}}_s)$, then, letting $\mathbf{x}_0 = \tilde{\mathbf{x}}_s$ and:*

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\tilde{\mathbf{H}}^{-1}\bigg(\frac{1}{m}\sum_{i=1}^m \nabla\psi_i(\mathbf{x}_t) - \nabla\psi_i(\tilde{\mathbf{x}}_s) + \tilde{\mathbf{g}}_s\bigg), \qquad \psi_1, ..., \psi_m \sim \mathcal{D},$$

after t iterations with mini-batch size $m \geq c\alpha^2\kappa\log(t/\delta)$ and step size $\eta = \min\{\sqrt{2/\alpha}, 1\}$, the iterate $\tilde{\mathbf{x}}_{s+1} = \mathbf{x}_t$ (i.e., one outer iteration of SVRN) with probability $1-\delta$ satisfies:

$$\frac{f(\tilde{\mathbf{x}}_{s+1}) - f(\mathbf{x}^*)}{f(\tilde{\mathbf{x}}_s) - f(\mathbf{x}^*)} \leq \Big(1 - \frac{1}{2\alpha}\Big)^t + c\alpha^2\log(t/\delta)\frac{\kappa}{m}.$$

The proof of Theorem 5, which is given in Appendix D, relies on a new high-probability bound for the error of the variance-reduced gradient estimates in the large mini-batch regime (Lemma 10).

Discussion. For simplicity, let us fix the number of inner iterations $t_{\max} = n/m$, so that a single outer iteration of SVRN always takes $O(1)$ passes over the data. Then, we can define the linear convergence rate after one outer iteration as a function of mini-batch size m:

$$\rho_m := \Big(1 - \frac{1}{2\alpha}\Big)^{n/m} + \tilde{O}(\kappa/m).$$

Let us assume the big data regime, i.e., $n \gg \kappa$. If we only use full-batch gradients ($m = n$), then the first term in the rate dominates, and we have $\rho_m \approx 1 - \frac{1}{2\alpha}$, which is similar to what we would get using standard Stochastic Newton (3). As we decrease m (and change $t_{\max}$ accordingly), the first term in ρ_m decreases, whereas the second term increases. As a result, the overall rate rapidly improves, reaching its optimal value of $\rho_m = \tilde{O}(\kappa/n)$ for $m \approx \frac{n}{\alpha\log(n/\kappa)}$.

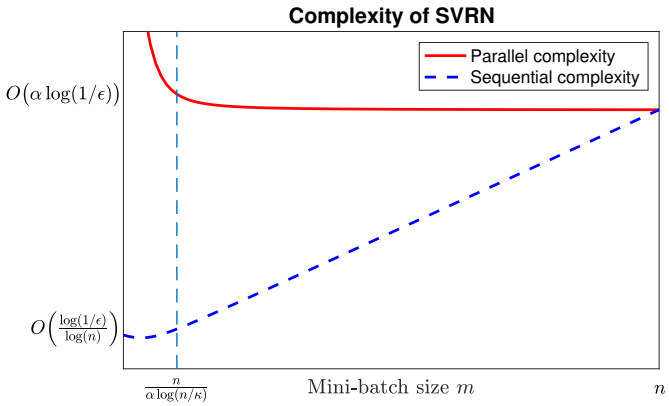


Figure 2: Illustration of the local convergence complexity analysis for SVRN, as a function of the mini-batch size m, with the number of inner iterations set to $t_{\max} = n/m$. As we decrease the mini-batch size from n (standard Stochastic Newton; SN) downto $m \approx \frac{n}{\alpha \log(n/\kappa)}$ (optimal SVRN), the sequential complexity (number of passes over the data) improves by $O(\alpha \log(n))$, while the parallel complexity (number of batch gradient queries) remains optimal.

Complexity analysis. The complexity analysis given in Theorem 1 follows directly from the above discussion, since the sequential complexity (number of data passes needed to improve by factor ϵ) is given by $O\big(\frac{\log(1/\epsilon)}{\log(1/\rho_m)}\big)$, whereas the parallel complexity (number of batch gradient queries) is $O\big(t_{\max} \cdot \frac{\log(1/\epsilon)}{\log(1/\rho_m)}\big)$. In Figure 2, we illustrate how these quantities change as a function of m. In particular, we observe that the batch gradients essentially stay flat at $O(\alpha \log(1/\epsilon))$ as we decrease m, until reaching $\frac{n}{\alpha \log(n/\kappa)}$. On the other hand, the data pass complexity decreases linearly with m, until it reaches the optimal value of $O\big(\frac{\log(1/\epsilon)}{\log(n/\kappa)}\big)$, which, for sufficiently large n, recovers Theorem 1.

B.1. Global convergence analysis of SVRN-HA

In the following result, we establish global convergence of SVRN-HA (Algorithm 1), by showing that the global phase of this method will reach the local neighborhood, and the Hessian estimate will get progressively more accurate, eventually reaching the desired approximation accuracy.

Theorem 6 *Let f be as in Theorem 5. For any neighborhood U around the optimum, Algorithm 1 will almost surely reach a point where: (a) $\tilde{\mathbf{x}}_s$ belongs to the neighborhood U, and (b) the Hessian estimate $\tilde{\mathbf{H}}_s$ satisfies the condition in Theorem 5. At this point, the line search will return $\eta_s = 1$.*

Proof This follows from global convergence analysis of Hessian averaging [34]. They show in Lemma 3.5 that if we were to run the global phase of SVRN-HA exclusively, then for any $\epsilon, \delta \in (0,1)$ there is $T := T(\epsilon, \delta)$ such that with probability $1-\delta$ for all $s \geq T$ we have $\tilde{\mathbf{x}}_s \in U_f(\epsilon)$, $\tilde{\mathbf{H}}_s \approx_\epsilon \nabla^2 f(\tilde{\mathbf{x}}_s)$, and $\eta_s = 1$. This means that, for any ϵ, the probability that the above event does not happen with any $T < \infty$ is less than any $\delta > 0$, so it must be 0. This implies that SVRN-HA will eventually switch to the local phase (i.e., to SVRN). Note that it is possible that the switch will occur before the local neighborhood and Hessian approximation conditions are met. But if this causes

SVRN to produce a poor descent direction, it will be caught by the line search (resulting in $\eta_s < 1$) and the method will simply revert back to the global phase. Eventually, the global phase will ensure that both conditions are met, and we can rely on Theorem 5 for the local convergence analysis. ■

Appendix C. Accelerating SVRN with sketching and importance sampling

When the minimization task possesses additional structure, then we can combine SVRN with Hessian and gradient estimation techniques other than uniform subsampling. For example, one such family of techniques, called randomized sketching [11, 14, 47], is applicable when the Hessian can be represented by a decomposition $\nabla^2 f(\mathbf{x}) = \mathbf{A}_f(\mathbf{x})^\top \mathbf{A}_f(\mathbf{x}) + \mathbf{C}$, where $\mathbf{A}_f(\mathbf{x})$ is a tall $n \times d$ matrix and $\mathbf{C}$ is a fixed $d \times d$ matrix. This setting applies for many empirical risk minimization tasks, including linear and logistic regression, among others.

Sketching can be used to construct an estimate of the Hessian by applying a randomized linear transformation to $\mathbf{A}_f(\mathbf{x})$, represented by a $k \times n$ random matrix $\mathbf{S}$, where $k = \tilde{O}(d)$ is much smaller than n. Using standard sketching techniques, such as Subsampled Randomized Hadamard Transforms [SRHT, 1], Sparse Johnson-Lindenstrauss Transforms [SJLT, 8, 29, 35] and Leverage Score Sparsified embeddings [LESS, 13], we can construct a Hessian estimate that satisfies the requirements of Theorem 1 at the cost of $\tilde{O}(nd + d^3)$, which corresponds to a nearly-constant number of data passes and $d \times d$ matrix multiplies. In particular, this eliminates the dependence of Hessian estimation on the condition number.

Another way of making SVRN more efficient is to use importance sampling in the stochastic gradient mini-batches. Importance sampling can be introduced to any finite-sum minimization task (1) by specifying an n-dimensional probability vector $p = (p_1, ..., p_n)$, such that $\sum_i p_i = 1$, and sampling the component gradient $\psi_{I_i}(\mathbf{x})$ so that the index I_i is drawn according to p. With the right choice of importance sampling, we can substantially reduce the smoothness parameter λ, and thereby, the condition number κ of the problem (see Appendix D.4).

The above techniques can be used to accelerate SVRN, for instance, in the important task of solving least squares regression. Here, given an $n \times d$ matrix $\mathbf{A}$ with rows $\mathbf{a}_i^\top$ and an n-dimensional vector $\mathbf{y}$, the objective being minimized is the following quadratic function:

$$f(\mathbf{x}) = \frac{1}{2n}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(\mathbf{a}_i^\top \mathbf{x} - y_i)^2.$$

One of the popular methods for solving the least squares task, known as the Iterative Hessian Sketch [IHS, 36], is exactly the Stochastic Newton update (3), where the Hessian estimate $\tilde{\mathbf{H}}$ is constructed via sketching. In this context, SVRN can be viewed as an accelerated version of IHS. To fully leverage the structure of the least squares problem, we use a popular importance sampling technique called leverage score sampling [15, 16], where the importance probabilities are (approximately) proportional to $p_i \propto \mathbf{a}_i^\top (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{a}_i$. Through an adaptation of our main result, we show that a version of SVRN for least squares, with sketched Hessian and leverage score sampled gradients, improves on the state-of-the-art complexity for reaching a high-precision solution to a preconditioned least squares task from $O(nd \log(1/\epsilon))$ [4, 30, 39] to $O\big(nd \frac{\log(1/\epsilon)}{\log(n/d)}\big)$. See Appendix D.4 for proof and further discussion.

Theorem 7 (Fast least squares solver) *Given $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$, after $O(nd\log n + d^3\log d)$ preprocessing cost to find the sketched Hessian estimate $\tilde{\mathbf{H}}$ and an approximate leverage score distribution, SVRN finds $\tilde{\mathbf{x}}$ so that*

$$f(\tilde{\mathbf{x}}) \leq (1+\epsilon)f(\mathbf{x}^*) \quad \text{in} \quad O\Big(nd\,\tfrac{\log(1/\epsilon)}{\log(n/d)}\Big) \quad \text{time.}$$

Crucially, the SVRN-based least squares solver only requires a preconditioner $\tilde{\mathbf{H}}$ that is a constant factor approximation of the Hessian, i.e., $\alpha = O(1)$. Interestingly, our approach of transforming the problem via leverage score sampling appears to be connected to a weighted and preconditioned SGD algorithm of [48] for solving a more general class of ℓ_p-regression problems. We expect that Theorem 7 can be similarly extended beyond least squares regression.

Appendix D. Proofs for local convergence analysis of SVRN

In this section, we provide the proofs of our main technical results, i.e., local convergence analysis for SVRN. First, we prove the result for the general case (Theorem 5), then we prove the result for least squares (Theorem 7).

D.1. Preliminaries

First, let us recall the formal definitions of the standard Hessian regularity assumptions used in Theorem 5. For all our results, it is sufficient that the function f satisfies either one of these assumptions.

Assumption 2 *Function $f : \mathbb{R}^d \to \mathbb{R}$ has Lipschitz continuous Hessian with constant L, i.e., $\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{x}')\| \leq L\,\|\mathbf{x}-\mathbf{x}'\|$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$.*

Assumption 3 *Function $f : \mathbb{R}^d \to \mathbb{R}$ is self-concordant, i.e., for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, the function $\phi(t) = f(\mathbf{x}+t\mathbf{x}')$ satisfies: $|\phi'''(t)| \leq 2(\phi''(t))^{3/2}$.*

In the proof, we use the following version of Bernstein's concentration inequality for random vectors [Corollary 4.1 in 31].

Lemma 8 *Let $\mathbf{v}_1, ..., \mathbf{v}_m \in \mathbb{R}^d$ be independent random vectors such that $\mathbb{E}[\mathbf{v}_i] = \mathbf{0}$ and $\|\mathbf{v}_i\| \leq R$ almost surely. Denote $\sigma^2 := \sum_{i=1}^m \mathbb{E}\,\|\mathbf{v}_i\|^2$. Then, for all $t^2 \geq \sigma^2 + tR/3$, we have*

$$\Pr\Big\{\Big\|\sum_{i=1}^m \mathbf{v}_i\Big\| > t\Big\} \leq 28\exp\Big(-\frac{t^2/2}{\sigma^2 + tR/3}\Big).$$

We also use the following lemma to convert from convergence in the norm, $\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}$, to convergence in excess loss, $f(\mathbf{x}) - f(\mathbf{x}^*)$, in the neighborhood around the optimum $\mathbf{x}^*$. The proof of this lemma, given in Appendix D.3, uses Quadratic Taylor's Theorem.

Lemma 9 *If f satisfies Assumption 1 and either Assumption 2 or 3, then for any $\epsilon \in [0,1]$ and $\mathbf{x} \in U_f(\epsilon_l)$, we have:*

$$\nabla^2 f(\mathbf{x}) \approx_{\epsilon_l} \nabla^2 f(\mathbf{x}^*) \qquad \text{and} \qquad f(\mathbf{x}) - f(\mathbf{x}^*) \approx_{\epsilon_l} \frac{1}{2}\|\mathbf{x}-\mathbf{x}^*\|^2_{\nabla^2 f(\mathbf{x}^*)}.$$

D.2. Proof of Theorem 5

To simplify the notation, we will drop the subscript s, so that $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_s$ and $\tilde{\mathbf{g}} = \tilde{\mathbf{g}}_s$. Also, let us define $\widehat{\mathbf{g}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m \nabla\psi_i(\mathbf{x})$. We use $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$, $\mathbf{g}_t = \mathbf{g}(\mathbf{x}_t)$, $\mathbf{H}_t = \nabla^2 f(\mathbf{x}_t)$, $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$, $\widehat{\mathbf{g}}_t = \widehat{\mathbf{g}}(\mathbf{x}_t)$, and $\bar{\mathbf{g}}_t = \widehat{\mathbf{g}}_t - \widehat{\mathbf{g}}(\tilde{\mathbf{x}}) + \tilde{\mathbf{g}}$ as shorthands. Also, we will use $\Delta_t = \mathbf{x}_t - \mathbf{x}^*$. We start by splitting up the error bound into two terms: the first one is an error term that would arise if we were using the exact gradient $\mathbf{g}_t$ instead of the gradient estimate $\bar{\mathbf{g}}_t$; and the second term addresses the error coming from the noise in the gradient estimate. Initially, we use the error $\|\Delta_t\|_{\mathbf{H}}$ to analyze the convergence rate in one step of the procedure, where recall that $\|\mathbf{v}\|_{\mathbf{M}} = \sqrt{\mathbf{v}^\top\mathbf{M}\mathbf{v}}$. We then convert that to get convergence in function value via Lemma 9.

We first address the assumption that $\tilde{\mathbf{H}}$ is an α-approximation of $\mathbf{H}_t$, as defined by the condition (2). Note that, via Lemma 9, for any $\mathbf{x}_t \in U_f(\epsilon_l)$ we have that $\mathbf{H}_t \approx_{\epsilon_l} \mathbf{H}$, which for a sufficiently small ϵ_l implies that $\mathbf{H}_t$ is a 1.1-approximation of $\mathbf{H}$ in the sense of (2). This, in turn implies that $\tilde{\mathbf{H}}$ is a 1.1α-approximation of $\mathbf{H}$, because $\tilde{\mathbf{H}} \preceq \sqrt{\alpha}\mathbf{H}_t \preceq \sqrt{1.1\alpha}\mathbf{H}$ (the other direction is analogous). For the sake of simplicity, we will replace 1.1α with α and say that $\tilde{\mathbf{H}}$ is an α-approximation of $\mathbf{H}$ (this can be easily accounted for by adjusting the constants at the end).

Now, suppose that after t inner iterations, we get $\mathbf{x}_t \in U_f(\epsilon_l)$ satisfying $\|\Delta_t\|_{\mathbf{H}} \leq \|\Delta_0\|_{\mathbf{H}}$. Our decomposition of the error into two terms proceeds as follows, where we use $\tilde{\mathbf{p}}_t = \tilde{\mathbf{H}}^{-1}\bar{\mathbf{g}}_t$:

$$\begin{aligned}
\|\Delta_{t+1}\|_{\mathbf{H}} &= \|(\mathbf{x}_t - \eta\tilde{\mathbf{p}}_t) - \mathbf{x}^*\|_{\mathbf{H}} \\
&= \|\Delta_t - \eta\tilde{\mathbf{H}}^{-1}\mathbf{g}_t + \eta\tilde{\mathbf{H}}^{-1}\mathbf{g}_t - \eta\tilde{\mathbf{H}}^{-1}\bar{\mathbf{g}}_t\|_{\mathbf{H}} \\
&\leq \|\Delta_t - \eta\tilde{\mathbf{H}}^{-1}\mathbf{g}_t\|_{\mathbf{H}} + \eta\|\tilde{\mathbf{H}}^{-1}(\mathbf{g}_t - \bar{\mathbf{g}}_t)\|_{\mathbf{H}} \qquad (4)
\end{aligned}$$

To bound the second term in (4), we first observe that $\tilde{\mathbf{H}}^{-1} \preceq \sqrt{\alpha}\mathbf{H}^{-1}$, which in turn yields $\mathbf{H}^{1/2}\tilde{\mathbf{H}}^{-1}\mathbf{H}^{1/2} \preceq \sqrt{\alpha}\,\mathbf{I}$. Thus, we can write $\|\mathbf{H}^{1/2}\tilde{\mathbf{H}}^{-1}\mathbf{H}^{1/2}\| \leq \sqrt{\alpha}$ and we get:

$$\begin{aligned}
\|\tilde{\mathbf{H}}^{-1}(\mathbf{g}_t - \bar{\mathbf{g}}_t)\|_{\mathbf{H}} &= \|\mathbf{H}^{1/2}\tilde{\mathbf{H}}^{-1}\mathbf{H}^{1/2}\mathbf{H}^{-1/2}(\mathbf{g}_t - \bar{\mathbf{g}}_t)\| \\
&\leq \|\mathbf{H}^{1/2}\tilde{\mathbf{H}}^{-1}\mathbf{H}^{1/2}\| \cdot \|\mathbf{H}^{-1/2}(\mathbf{g}_t - \bar{\mathbf{g}}_t)\| \\
&\leq \sqrt{\alpha} \cdot \|\mathbf{g}_t - \bar{\mathbf{g}}_t\|_{\mathbf{H}^{-1}}
\end{aligned}$$

We now break $\|\mathbf{g}_t - \bar{\mathbf{g}}_t\|_{\mathbf{H}^{-1}}$ down into two parts, introducing $\widehat{\mathbf{g}}(\mathbf{x}^*)$ and separating $\widehat{\mathbf{g}}(\tilde{\mathbf{x}})$ from $\widehat{\mathbf{g}}_t$:

$$\begin{aligned}
\|\mathbf{g}_t - \bar{\mathbf{g}}_t\|_{\mathbf{H}^{-1}} &= \|\mathbf{g}_t - (\widehat{\mathbf{g}}_t - \widehat{\mathbf{g}}(\tilde{\mathbf{x}}) + \tilde{\mathbf{g}})\|_{\mathbf{H}^{-1}} \\
&\leq \|\mathbf{g}_t - (\widehat{\mathbf{g}}_t - \widehat{\mathbf{g}}(\mathbf{x}^*))\|_{\mathbf{H}^{-1}} + \|\tilde{\mathbf{g}} - (\widehat{\mathbf{g}}(\tilde{\mathbf{x}}) - \widehat{\mathbf{g}}(\mathbf{x}^*))\|_{\mathbf{H}^{-1}}.
\end{aligned}$$

We bound the above two terms using the following lemma, which gives a new high-probability error bound for the variance reduced gradient estimates in the large mini-batch regime which, unlike results from prior work that hold in expectation, crucially relies on the iterate being in the local neighborhood $U_f(1)$ around the optimum $\mathbf{x}^*$.

Lemma 10 *There is an absolute constant $C > 0$ such that for any $\mathbf{x} \in U_f(1)$, letting $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$, the gradient estimate $\widehat{\mathbf{g}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m \nabla\psi_i(\mathbf{x})$ using $m \geq \kappa\log(1/\delta)$ samples, with probability $1-\delta$ satisfies:*

$$\big\|\widehat{\mathbf{g}}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x}^*) - \nabla f(\mathbf{x})\big\|^2_{\mathbf{H}^{-1}} \leq C\log(1/\delta)\frac{\kappa}{m}\|\mathbf{x} - \mathbf{x}^*\|^2_{\mathbf{H}}.$$

Proof We will apply Bernstein's concentration inequality for random vectors (Lemma 8) to $\mathbf{v}_i = \nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*) - \nabla f(\mathbf{x})$. First, observe that $\mathbb{E}\,\nabla\psi_i(\mathbf{x}) = \nabla f(\mathbf{x})$ and $\mathbb{E}\,\nabla\psi_i(\mathbf{x}^*) = \nabla f(\mathbf{x}^*) = \mathbf{0}$, so in particular, $\mathbb{E}[\mathbf{v}_i] = \mathbf{0}$.

In the next step, we will use the fact that for any λ-smooth function g, we have $\|\nabla g(\mathbf{x})\|^2 \leq 2\lambda \cdot (g(\mathbf{x}) - \min_{\mathbf{x}'} g(\mathbf{x}'))$, which follows because:

$$\begin{aligned}\min_{\mathbf{x}'} g(\mathbf{x}') &\leq g\big(\mathbf{x} - \tfrac{1}{\lambda}\nabla g(\mathbf{x})\big)\\ &\leq g(\mathbf{x}) - \frac{1}{\lambda}\|\nabla g(\mathbf{x})\|^2 + \frac{\lambda}{2}\,\frac{1}{\lambda^2}\|\nabla g(\mathbf{x})\|^2\\ &= g(\mathbf{x}) - \frac{1}{2\lambda}\|\nabla g(\mathbf{x})\|^2.\end{aligned}$$

We will use this fact once on f, and also a second time, on the function $g(\mathbf{x}) = \psi_i(\mathbf{x}) - \psi_i(\mathbf{x}^*) - (\mathbf{x} - \mathbf{x}^*)^\top\nabla\psi(\mathbf{x}^*)$, which is λ-smooth because ψ_i is λ-smooth, observing that $\nabla g(\mathbf{x}) = \nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)$ and that $\min_{\mathbf{x}'} g(\mathbf{x}') = g(\mathbf{x}^*) = 0$. Thus, we have

$$\begin{aligned}\|\mathbf{v}_i\|^2 &\leq 2\|\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)\|^2 + 2\|\nabla f(\mathbf{x})\|^2\\ &\leq 4\lambda\cdot\big(\psi_i(\mathbf{x}) - \psi_i(\mathbf{x}^*) - (\mathbf{x}-\mathbf{x}^*)^\top\nabla\psi(\mathbf{x}^*)\big) + 4\lambda\cdot\big(f(\mathbf{x}) - f(\mathbf{x}^*)\big)\\ &\leq 2\lambda^2\|\mathbf{x}-\mathbf{x}^*\|^2 + 2\lambda^2\|\mathbf{x}-\mathbf{x}^*\|^2 = 4\lambda^2\|\mathbf{x}-\mathbf{x}^*\|^2,\end{aligned}$$

where in the last step we used again that ψ_i and f are λ-smooth. To bound the expectation $\mathbb{E}\,\|\mathbf{v}_i\|^2$, we use the intermediate inequality from the above derivation, obtaining:

$$\begin{aligned}\mathbb{E}\,\|\mathbf{v}_i\|^2 &= \mathbb{E}\big[\|\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)\|^2\big] - 2\,\mathbb{E}\big[\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)\big]^\top\nabla f(\mathbf{x}) + \|\nabla f(\mathbf{x})\|^2\\ &= \mathbb{E}\big[\|\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)\|^2\big] - \|\nabla f(\mathbf{x})\|^2\\ &\leq \mathbb{E}\big[\|\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\mathbf{x}^*)\|^2\big]\\ &\leq \mathbb{E}\Big[2\lambda\cdot\big(\psi_i(\mathbf{x}) - \psi_i(\mathbf{x}^*) - (\mathbf{x}-\mathbf{x}^*)^\top\nabla\psi_i(\mathbf{x}^*)\big)\Big]\\ &= 2\lambda\cdot\big(f(\mathbf{x}) - f(\mathbf{x}^*) - (\mathbf{x}-\mathbf{x}^*)\nabla f(\mathbf{x}^*)\big)\\ &= 2\lambda\cdot\big(f(\mathbf{x}) - f(\mathbf{x}^*)\big).\end{aligned}$$

We now use the assumption that $\mathbf{x}\in U_f(1)$, which implies via Lemma 9 that $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 2\cdot\frac{1}{2}\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2 = \|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2$. Thus, we can use Lemma 8 with $R = 2\lambda\|\mathbf{x}-\mathbf{x}^*\|$ and $\sigma^2 = 2m\lambda\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2$, as well as μ-strong convexity of f, obtaining that, for some absolute constant C, with probability $1-\delta$, we have:

$$\begin{aligned}\|\widehat{\mathbf{g}}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x}^*) - \nabla f(\mathbf{x})\|_{\mathbf{H}^{-1}}^2 &\leq \frac{1}{\mu}\Big\|\frac{1}{m}\sum_{i=1}^m \mathbf{v}_i\Big\|^2\\ &\leq \frac{C}{\mu}\Big(\frac{\sigma^2\log(1/\delta)}{m^2} + \frac{R^2\log^2(1/\delta)}{m^2}\Big)\\ &\leq \frac{C}{\mu}\left(\frac{2\lambda\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2\log(1/\delta)}{m} + \frac{4\lambda^2\|\mathbf{x}-\mathbf{x}^*\|^2\log^2(1/\delta)}{m^2}\right)\\ &\leq 4C\left(\frac{\kappa\log(1/\delta)}{m} + \frac{\kappa^2\log^2(1/\delta)}{m^2}\right)\cdot\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2\\ &\leq 8C\log(1/\delta)\cdot\frac{\kappa}{m}\,\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2,\end{aligned}$$

where in the last step we used that $m \geq \kappa \log(1/\delta)$. ■

Letting $\epsilon_g = \sqrt{2C\log(t/\delta)\kappa/m}$, Lemma 10 implies that with probability $1-\delta/t^2$,

$$\|\mathbf{g}_t - (\widehat{\mathbf{g}}_t - \widehat{\mathbf{g}}(\tilde{\mathbf{x}}) + \tilde{\mathbf{g}})\|_{\mathbf{H}^{-1}} \leq \epsilon_g\big(\|\Delta_t\|_{\mathbf{H}} + \|\Delta_0\|_{\mathbf{H}}\big) \leq 2\epsilon_g\|\Delta_0\|_{\mathbf{H}}.$$

Finally, we return to the first term in (4), i.e., $\|\Delta_t - \eta\tilde{\mathbf{H}}^{-1}\mathbf{g}_t\|_{\mathbf{H}}$. To control this term we introduce the following lemma which is potentially of independent interest to the local convergence analysis of Newton-type methods.

Lemma 11 *Suppose that f satisfies Assumption 1 and either one of the Assumptions 2 or 3, and take any $\mathbf{x} \in U_f(\epsilon_l)$ (see Definition 4) for $\epsilon_l \leq 1/c\alpha$ for a sufficiently large absolute constant $c > 0$. Let $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$ and consider a pd matrix $\tilde{\mathbf{H}}$ that satisfies $\frac{1}{\sqrt{\alpha}}\mathbf{H} \preceq \tilde{\mathbf{H}} \preceq \sqrt{\alpha}\,\mathbf{H}$. Then, for $\eta := \min\{\sqrt{2/\alpha}, 1\}$, we have:*

$$\|\mathbf{x} - \eta\tilde{\mathbf{H}}^{-1}\nabla f(\mathbf{x}) - \mathbf{x}^*\|_{\mathbf{H}} \leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{H}}.$$

Proof Let $\Delta_0 := \mathbf{x} - \mathbf{x}^*$ and $\Delta_1 := \mathbf{x} - \eta\tilde{\mathbf{H}}^{-1}\nabla f(\mathbf{x}) - \mathbf{x}^*$. Using that $\nabla f(\mathbf{x}^*) = \mathbf{0}$, we have:

$$\begin{aligned}
\Delta_1 &= \Delta_0 - \eta\tilde{\mathbf{H}}^{-1}\nabla f(\mathbf{x}) \\
&= \Delta_0 - \eta\tilde{\mathbf{H}}^{-1}(\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^*)) \\
&= \Delta_0 - \eta\tilde{\mathbf{H}}^{-1}\int_0^1 \nabla^2 f(\mathbf{x}^* + \theta\Delta_0)\Delta_0 d\theta \\
&= (\mathbf{I} - \eta\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}})\Delta_0,
\end{aligned}$$

where we defined $\bar{\mathbf{H}} := \int_0^1 \nabla^2 f(\mathbf{x}^* + \theta\Delta_0)d\theta$. It follows that we can bound the norm of Δ_1 using a norm defined by the matrix $\bar{\mathbf{H}}$:

$$\begin{aligned}
\|\Delta_1\|_{\bar{\mathbf{H}}} &= \|\bar{\mathbf{H}}^{1/2}(\mathbf{I} - \eta\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}})\Delta_0\| \\
&= \|(\mathbf{I} - \eta\mathbf{H}^{1/2}\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}}^{1/2})\bar{\mathbf{H}}^{1/2}\Delta_0\| \\
&\leq \|\mathbf{I} - \eta\bar{\mathbf{H}}^{1/2}\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}}^{1/2}\| \cdot \|\Delta_0\|_{\bar{\mathbf{H}}}.
\end{aligned}$$

Observe that for any $\theta \in [0,1]$, the vector $\mathbf{x}^* + \theta\Delta_0$ belongs to $U_f(\epsilon_l)$, which via Lemma 9 implies that

$$\nabla^2 f(\mathbf{x}^* + \theta\Delta_0) \approx_{\epsilon_l} \mathbf{H} \quad \forall\theta \in [0,1].$$

where $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$. In particular, this means that $\bar{\mathbf{H}} \approx_{\epsilon_l} \mathbf{H}$, which, combined with the α-approximation property of $\tilde{\mathbf{H}}$, allows us to write the following:

$$\tilde{\mathbf{H}}^{-1} \preceq \sqrt{\alpha}\mathbf{H}^{-1} \preceq \sqrt{\alpha}(1+\epsilon_l)\bar{\mathbf{H}}^{-1} \quad \text{and} \quad \tilde{\mathbf{H}}^{-1} \succeq \frac{1}{\sqrt{\alpha}}\mathbf{H}^{-1} \succeq \frac{1-\epsilon_l}{\sqrt{\alpha}}\bar{\mathbf{H}}^{-1}.$$

Putting these inequalities together, we obtain that:

$$\eta\frac{1-\epsilon_l}{\sqrt{\alpha}}\mathbf{I} \preceq \eta\bar{\mathbf{H}}^{1/2}\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}}^{1/2} \preceq \eta\sqrt{\alpha}(1+\epsilon_l)\mathbf{I}.$$

Now, using the fact that $\eta = \min\{\sqrt{2/\alpha}, 1\}$, we conclude that:

$$\begin{aligned}\|\mathbf{I} - \eta\bar{\mathbf{H}}^{1/2}\tilde{\mathbf{H}}^{-1}\bar{\mathbf{H}}^{1/2}\| &\leq \max\Big\{\eta\sqrt{\alpha}(1+\epsilon_l) - 1, 1 - \eta\frac{1-\epsilon_l}{\sqrt{\alpha}}\Big\}\\ &\leq \max\Big\{\sqrt{2}(1+\epsilon_l) - 1, 1 - \frac{\sqrt{2}(1-\epsilon_l)}{\alpha}, 1 - \frac{1-\epsilon_l}{\sqrt{2}}\Big\}\\ &\leq \max\Big\{1 - \frac{1}{1.8},\ 1 - \frac{1}{\alpha}\Big\} \leq 1 - \frac{1}{1.8\alpha},\end{aligned}$$

where we used that $\epsilon_l \leq 1/c$ for a sufficiently large constant $c > 0$ such that $\max\{\sqrt{2}(1+\epsilon_l) - 1, 1 - \frac{1-\epsilon_l}{\sqrt{2}}\} \leq 1 - \frac{1}{1.8}$. Now, we analyze convergence in the norm induced by $\mathbf{H}$, instead of $\bar{\mathbf{H}}$, by relying again on the fact that $\bar{\mathbf{H}} \approx_{\epsilon_l} \mathbf{H}$, obtaining:

$$\begin{aligned}\|\Delta_1\|_{\mathbf{H}} &\leq \frac{1}{\sqrt{1-\epsilon_l}}\|\Delta_1\|_{\bar{\mathbf{H}}} \leq \frac{1}{\sqrt{1-\epsilon_l}}\Big(1 - \frac{1}{1.8\alpha}\Big)\|\Delta_0\|_{\bar{\mathbf{H}}}\\ &\leq \sqrt{\frac{1+\epsilon_l}{1-\epsilon_l}}\Big(1 - \frac{1}{1.8\alpha}\Big)\|\Delta_0\|_{\mathbf{H}} \leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\Delta_0\|_{\mathbf{H}},\end{aligned}$$

where the last step requires $\epsilon_l = 1/c\alpha$ for sufficiently large absolute constant $c > 0$. ■

Using Lemma 11 to bound the first term in (4), we obtain that:

$$\|\Delta_t - \eta\tilde{\mathbf{H}}_t^{-1}\mathbf{g}_t\|_{\mathbf{H}} \leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\Delta_t\|_{\mathbf{H}}.$$

Putting everything together, we obtain the following bound for the error of the update that uses the stochastic variance-reduced gradient estimate:

$$\begin{aligned}\|\Delta_{t+1}\|_{\mathbf{H}} &= \|\Delta_t - \eta\tilde{\mathbf{p}}_t\|_{\mathbf{H}}\\ &\leq \|\Delta_t - \eta\tilde{\mathbf{H}}^{-1}\mathbf{g}_t\|_{\mathbf{H}} + \eta\|\tilde{\mathbf{H}}^{-1}(\mathbf{g}_t - \bar{\mathbf{g}}_t)\|_{\mathbf{H}}\\ &\leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\Delta_t\|_{\mathbf{H}} + 2\eta\sqrt{\alpha}\epsilon_g\|\Delta_0\|_{\mathbf{H}}\\ &\leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\Delta_t\|_{\mathbf{H}} + 3\epsilon_g\|\Delta_0\|_{\mathbf{H}}.\end{aligned}$$

Note that, as long as $3\epsilon_g \leq \frac{1}{2\alpha}$ (which can be ensured by our assumption on m), this implies that $\|\Delta_{t+1}\|_{\mathbf{H}} \leq \|\Delta_0\|_{\mathbf{H}}$ and so $\mathbf{x}_{t+1} \in U_f(\epsilon_l)$. Thus, our analysis can be applied recursively at each inner iteration. To expand the error recursion, observe that if we apply a union bound over the high-probability events in Lemma 10 for each inner iteration t using failure probability $\delta_t = \delta/t^2$, then they hold for all t with probability at least $1 - \sum_{t=1}^{\infty}\delta/t^2 \geq 1 - \delta\pi^2/6$. We obtain:

$$\begin{aligned}\|\Delta_t\|_{\mathbf{H}} &\leq \Big(1 - \frac{1}{1.9\alpha}\Big)\|\Delta_{t-1}\|_{\mathbf{H}} + 3\epsilon_g\|\Delta_0\|_{\mathbf{H}}\\ &\leq \Big(1 - \frac{1}{1.9\alpha}\Big)^t\|\Delta_0\|_{\mathbf{H}} + \Bigg(\sum_{i=0}^{t-1}\Big(1 - \frac{1}{1.9\alpha}\Big)^i\Bigg)\cdot 3\epsilon_g\|\Delta_0\|_{\mathbf{H}}\\ &\leq \Bigg(\Big(1 - \frac{1}{1.9\alpha}\Big)^t + 9\alpha\epsilon_g\Bigg)\cdot\|\Delta_0\|_{\mathbf{H}}.\end{aligned}$$

Applying Lemma 9, we convert this to convergence in function value:

$$\begin{aligned} f(\mathbf{x}_t) - f(\mathbf{x}^*) &\leq \frac{1}{1-\epsilon_l}\frac{1}{2}\|\Delta_t\|_{\mathbf{H}}^2 \\ &\leq \frac{1}{1-\epsilon_l}\frac{1}{2}\left(\left(1-\frac{1}{1.9\alpha}\right)^t + 9\alpha\epsilon_g\right)^2\|\Delta_0\|_{\mathbf{H}}^2 \\ &\leq \frac{1+\epsilon_l}{1-\epsilon_l}\left(\left(1-\frac{1}{1.9\alpha}\right)^{2t} + 9^2\alpha^2\epsilon_g^2\right)\cdot(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \\ &\leq \left(\left(1-\frac{1}{2\alpha}\right)^t + C'\alpha^2\frac{\kappa\log(t/\delta)}{m}\right)\cdot(f(\mathbf{x}_0) - f(\mathbf{x}^*)), \end{aligned}$$

where C' is an absolute constant, and we again used that $\epsilon_l \leq 1/c\alpha$ for a sufficiently large c, thus concluding the proof.

D.3. Proof of Lemma 9

First, we show that the Hessian at $\mathbf{x} \in U_f(\epsilon_l)$ is an ϵ_l-approximation of the Hessian at the optimum $\mathbf{x}^*$. This is broken down into two cases, depending on which of the two Assumptions 2 and 3 are satisfied.

Case 1: Assumption 2 (Lipschitz Hessian). Using the shorthand $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$ and the fact that strong convexity (Assumption 1) implies that $\nabla^2 f(\mathbf{x}) \succeq \mu\mathbf{I}$, we have:

$$\|\mathbf{H}^{-1/2}(\nabla^2 f(\mathbf{x}) - \mathbf{H})\mathbf{H}^{-1/2}\| \leq \frac{1}{\mu}\|\nabla^2 f(\mathbf{x}) - \mathbf{H}\| \leq \frac{L}{\mu}\|\mathbf{x}-\mathbf{x}^*\| \leq \frac{L}{\mu^{3/2}}\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}} \leq \epsilon_l,$$

which implies that $\nabla^2 f(\mathbf{x}) \approx_{\epsilon_l} \mathbf{H}$.

Case 2: Assumption 3 (Self-concordance). The fact that $\nabla^2 f(\mathbf{x}) \approx_{\epsilon_l} \mathbf{H}$ follows from the following property of self-concordant functions [7, Chapter 9.5], which holds when $\|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}} < 1$:

$$(1 - \|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}})^2 \cdot \mathbf{H} \preceq \nabla^2 f(\mathbf{x}) \preceq (1 - \|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}})^{-2} \cdot \mathbf{H},$$

where we again let $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$.

We next use a version of Quadratic Taylor's Theorem, as given below. See Theorem 3 in Chapter 2.6 of [23] and Chapter 2.7 in [18].

Lemma 12 *Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ has continuous first and second derivatives. Then, for any $\mathbf{a}$ and $\mathbf{v}$, there exists $\theta \in (0,1)$ such that:*

$$f(\mathbf{a}+\mathbf{v}) = f(\mathbf{a}) + \nabla f(\mathbf{a})^\top \mathbf{v} + \frac{1}{2}\mathbf{v}^\top \nabla^2 f(\mathbf{a}+\theta\mathbf{v})\mathbf{v}.$$

Applying Talyor's theorem with $\mathbf{a} = \mathbf{x}^*$ and $\mathbf{v} = \mathbf{x} - \mathbf{x}^*$, there is a $\mathbf{z} = \mathbf{x}^* + \theta(\mathbf{x}-\mathbf{x}^*)$ such that:

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \frac{1}{2}\|\mathbf{x}-\mathbf{x}^*\|_{\nabla^2 f(\mathbf{z})}^2,$$

where we use that $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Since we assumed that $\mathbf{x} \in U_f(\epsilon_l)$, and naturally also $\mathbf{x}^* \in U_f(\epsilon_l)$, this means that $\mathbf{z} \in U_f(\epsilon_l)$, given that $U_f(\epsilon_l)$ is convex. Thus, using that $\nabla^2 f(\mathbf{z}) \approx_{\epsilon_l} \mathbf{H}$, we have $\|\mathbf{x}-\mathbf{x}^*\|_{\nabla^2 f(\mathbf{z})}^2 \approx_{\epsilon_l} \|\mathbf{x}-\mathbf{x}^*\|_{\mathbf{H}}^2$.

D.4. Proof of Theorem 7

In this section, we discuss how the convergence analysis of SVRN can be adapted to using leverage score sampling when solving a least squares task (proving Theorem 7).

Consider an expected risk minimization problem $f(\mathbf{x}) = \mathbb{E}[\psi(\mathbf{x})]$, where $\psi = \frac{1}{np_I}\psi_I$ and I is an index from $\{1, ..., n\}$, sampled according to some importance sampling distribution p. More specifically, consider a least squares task, where the components are given by $\psi_i(\mathbf{x}) = \frac{1}{2}(\mathbf{a}_i^\top \mathbf{x} - y_i)^2$. Then, the overall minimization task becomes:

$$\mathbb{E}[\psi(\mathbf{x})] = \mathbb{E}\big[\frac{1}{np_I}\psi_I(\mathbf{x})\big] = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{a}_i^\top \mathbf{x} - y_i)^2. \tag{5}$$

Moreover, we have $f(\mathbf{x}) - f(\mathbf{x}^*) = \frac{1}{2n}\|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 = \frac{1}{2}\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{H}}^2$, where $\mathbf{H} = \nabla^2 f(\mathbf{x}) = \frac{1}{n}\mathbf{A}^\top\mathbf{A}$. Also,

$$\nabla\psi_i(\mathbf{x}) = (\mathbf{a}_i^\top \mathbf{x} - y_i)\mathbf{a}_i, \qquad \nabla^2\psi_i(\mathbf{x}) = \mathbf{a}_i\mathbf{a}_i^\top.$$

Naturally, since the Hessian is the same everywhere for this task, the local convergence neighborhood U_f is simply the entire Euclidean space $\mathbb{R}^d$. Let us first recall our definition of the condition number for this task. Assumption 1 states that each ψ_i is λ-smooth, i.e., $\|\nabla^2\psi_i(\mathbf{x})\| = \|\mathbf{a}_i\|^2 \leq \lambda$ and f is μ-strongly convex, i.e., $\lambda_{\min}(\mathbf{H}) = \frac{1}{n}\sigma_{\min}^2(\mathbf{A}) \geq \mu$, and the condition number of the problem is defined as $\kappa = \lambda/\mu \geq \max_i\{n\|\mathbf{a}_i\|^2\}/\sigma_{\min}^2(\mathbf{A})$. Can we reduce the condition number of this problem by importance sampling?

Consider the following naive strategy which can be applied directly with our convergence result. Here, we let the importance sampling probabilities be $p_i \propto \|\mathbf{a}_i\|^2$, so that the smoothness of the new reweighted problem will be $\tilde{\lambda} = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{a}_i\|^2$. In other words, it will be the average smoothness of the original problem, instead of the worst-case smoothness. Such importance sampling strategy can theoretically be applied to a general finite-sum minimization task with some potential gain, however we may need different sampling probabilities at each step. For least squares, the resulting condition number is $\tilde{\kappa} = \tilde{\lambda}/\mu = (\sum_i\|\mathbf{a}_i\|^2)/\sigma_{\min}^2(\mathbf{A})$. This is still worse than what we claimed for least squares, but it is still potentially much better than κ.

Next, we will show that by slightly adapting our convergence analysis, we can use leverage score sampling to further improve the convergence of SVRN for the least squares task. Recall that the ith leverage score of $\mathbf{A}$ is defined as $\ell_i = \|\mathbf{a}_i\|^2_{(\mathbf{A}^\top\mathbf{A})^{-1}} = \frac{1}{n}\|\mathbf{a}_i\|^2_{\mathbf{H}^{-1}}$, and the laverage scores satisfy $\sum_{i=1}^{n}\ell_i = d$. This result will require showing a specialized version of Lemma 10, which bounds the error in the variance-reduced subsampled gradient. In this case we show a bound in expectation, instead of with high probability.

Lemma 13 *Suppose that f defines a least squares task (5) and the sampling probabilities satisfy $p_i \geq \|\mathbf{a}_i\|^2_{(\mathbf{A}^\top\mathbf{A})^{-1}}/(Cd)$. Then, $\widehat{\mathbf{g}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{np_{I_i}}\nabla\psi_{I_i}(\mathbf{x})$, where $I_1, ..., I_m \sim p$, satisfies:*

$$\mathbb{E}\,\|\widehat{\mathbf{g}}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x}^*) - \nabla f(\mathbf{x})\|^2_{\mathbf{H}^{-1}} \leq C\frac{d}{m}\cdot\|\mathbf{x} - \mathbf{x}^*\|^2_{\mathbf{H}}.$$

Proof We define $\mathbf{v}_i = \frac{1}{np_{I_i}}(\nabla\psi_{I_i}(\mathbf{x}) - \nabla\psi_{I_i}(\mathbf{x}^*)) - \nabla f(\mathbf{x})$. Note that $\mathbb{E}[\mathbf{v}_i] = \mathbf{0}$, so we have:

$$\begin{aligned}
\mathbb{E}\,\|\widehat{\mathbf{g}}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x}^*) - \nabla f(\mathbf{x})\|^2_{\mathbf{H}^{-1}} &= \mathbb{E}\,\Big\|\frac{1}{m}\sum_{i=1}^m \mathbf{v}_i\Big\|^2_{\mathbf{H}^{-1}} = \frac{1}{m}\mathbb{E}\,\|\mathbf{v}_1\|^2_{\mathbf{H}^{-1}} \\
&\leq \frac{1}{m}\mathbb{E}\,\frac{1}{n^2p^2_{I_1}}\|\nabla\psi_{I_1}(\mathbf{x}) - \nabla\psi_{I_1}(\mathbf{x}^*)\|^2_{\mathbf{H}^{-1}} \\
&= \frac{1}{m}\mathbb{E}\,\frac{\|\mathbf{a}_{I_1}\|^2_{\mathbf{H}^{-1}}}{n^2p^2_{I_1}}\big(\mathbf{a}_{I_1}^\top(\mathbf{x}-\mathbf{x}^*)\big)^2 \\
&\leq \frac{1}{m}\,Cd\cdot\mathbb{E}\,\frac{(\mathbf{a}_{I_1}^\top(\mathbf{x}-\mathbf{x}^*))^2}{np_{I_1}} = C\cdot\frac{d}{m}\|\mathbf{x}-\mathbf{x}^*\|^2_{\mathbf{H}},
\end{aligned}$$

where we used that $\|\mathbf{a}_i\|^2_{\mathbf{H}^{-1}} = n\,\|\mathbf{a}_i\|^2_{(\mathbf{A}^\top\mathbf{A})^{-1}} \leq Cndp_i$. ■

Since the above bound is obtained in expectation, to insert it into our high probability analysis, we apply Markov's inequality. Namely, it holds with probability $1-\delta$ that:

$$\|\widehat{\mathbf{g}}(\mathbf{x}) - \widehat{\mathbf{g}}(\mathbf{x}^*) - \nabla f(\mathbf{x})\|^2_{\mathbf{H}^{-1}} \leq \frac{Cd}{\delta m}\|\mathbf{x}-\mathbf{x}^*\|^2_{\mathbf{H}}.$$

Compared to Lemma 10, the dependence on the condition number κ is completely eliminated in this result. Letting $m = n/\log(n/d)$ and the number of local iterations of SVRN to be $t = O(\log(n/d))$, we can apply the union bound argument from the proof of Theorem 5 by letting $\delta = 1/(Ct)$, so that with probability $1-1/C$, one stage of leverage score sampled SVRN satisfies:

$$f(\tilde{\mathbf{x}}_{s+1}) - f(\mathbf{x}^*) \leq \rho\cdot\big(f(\tilde{\mathbf{x}}_s) - f(\mathbf{x}^*)\big) \qquad \text{for} \qquad \rho = O\Big(\frac{d\log^2(n/d)}{n}\Big).$$

Alternatively, our main convergence analysis can be adapted (for least squares) to convergence in expectation, obtaining that $\mathbb{E}[f(\tilde{\mathbf{x}}_{s+1}) - f(\mathbf{x}^*)] \leq \tilde{\rho}\cdot\mathbb{E}[f(\tilde{\mathbf{x}}_s) - f(\mathbf{x}^*)]$ for $\tilde{\rho} = O(d\log(n/d)/n)$.

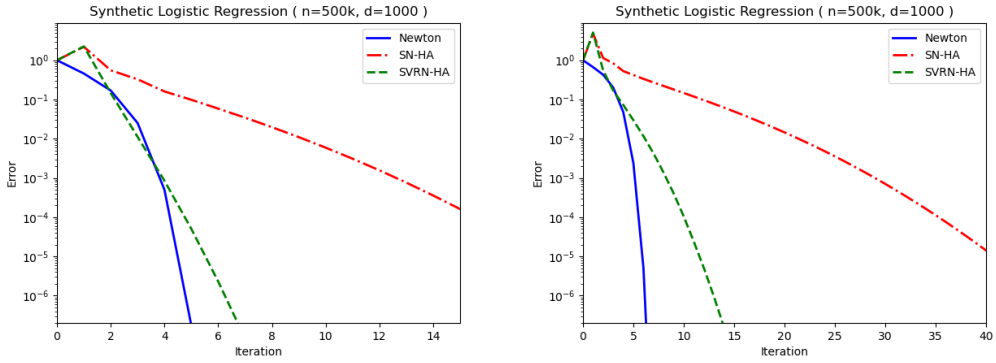
The time complexity stated in Theorem 7 comes from the fact that constructing a preconditioning matrix $\tilde{\mathbf{H}}$ that is an α-approximation of $\mathbf{H}$ with $\alpha = O(1)$, together with approximating the leverage scores, takes $O(nd\log n + d^3\log d)$ [16], whereas one stage of SVRN takes $O(nd + d^2\log(n/d))$. Here, the preconditioning matrix can be formed by applying a $k\times n$ sketching transformation $\mathbf{S}$ to the data matrix $\mathbf{A}$, and then computing the Hessian estimate $\frac{1}{n}\mathbf{A}^\top\mathbf{S}^\top\mathbf{S}\mathbf{A} \approx \mathbf{H}$. For example, if we use the Subsampled Randomized Hadamard Transform [SRHT, 1], then it suffices to use $k = O(d\log d)$. Finally, the initial iterate $\tilde{\mathbf{x}}_0$ can be constructed using the same sketching transformation via the so-called sketch-and-solve technique [42]:

$$\tilde{\mathbf{x}}_0 = \operatorname*{argmin}_{\mathbf{x}} \|\mathbf{S}\mathbf{A}\mathbf{x} - \mathbf{S}\mathbf{y}\|^2.$$

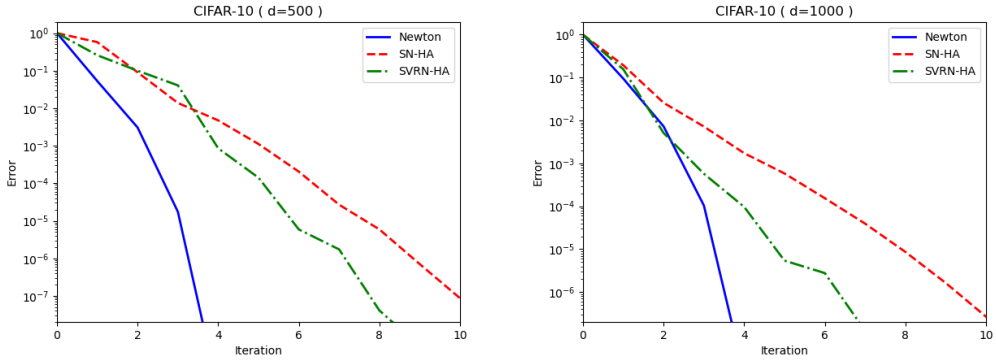
With $k = O(d\log d)$, this initial iterate will satisfy $f(\tilde{\mathbf{x}}_0) \leq O(1)\cdot f(\mathbf{x}^*)$, so the number of iterations of SVRN needed to obtain $f(\tilde{\mathbf{x}}_s) \leq (1+\epsilon)f(\mathbf{x}^*)$ is only $s = O\big(\frac{\log(1/\epsilon)}{\log(n/d)}\big)$.

We note that another way to implement SVRN with approximate leverage score sampling is to first precondition the entire least squares problem with a Randomized Hadamard Transform (i.e., SRHT without the subsampling):

$$\tilde{\mathbf{A}} = \mathbf{H}\mathbf{D}\mathbf{A} \qquad \text{and} \qquad \tilde{\mathbf{y}} = \mathbf{H}\mathbf{D}\mathbf{y}, \tag{6}$$



(*a*) Synthetic LR, $\kappa_{\mathbf{A}}^2 = 1$ (*b*) Synthetic LR, $\kappa_{\mathbf{A}}^2 = 10$



(*c*) CIFAR-10, $d = 500$ (*d*) CIFAR-10, $d = 1000$

Figure 3: Convergence comparison of SVRN-HA against SN-HA and Newton for a synthetic logistic regression task as we vary the condition number of the data matrix, and for the CIFAR-10 dataset.

where $\mathbf{H}$ is a Hadamard matrix scaled by $1/\sqrt{n}$ and $\mathbf{D}$ is a diagonal matrix with random sign entries. This is a popular technique in Randomized Numerical Linear Algebra [11, 14, 47]. The cost of this transformation is $O(nd \log n)$, thanks to fast Fourier transform techniques, and the resulting least squares task is equivalent to the original one, because $\|\tilde{\mathbf{A}}\mathbf{x} - \tilde{\mathbf{y}}\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$ for all $\mathbf{x}$. Moreover, with high probability, all of the leverage scores of $\tilde{\mathbf{A}}$ are nearly uniform, so, after this preconditioning, we can simply implement SVRN with uniform gradient subsampling and still enjoy the condition-number-free convergence rate from Theorem 7. This strategy is as effficient as direct leverage score sampling when $\mathbf{A}$ is a dense matrix, but it is less effective when we want to exploit data sparsity.

Appendix E. Further numerical experiments

In this section we provide additional details regarding our experimental setup in Section 3, as well as some further results on logistic regression with several datasets.[1]

1. The code is available at https://github.com/svrnewton/svrn.

The optimization task used in Section 3 involves training a regularized logistic regression model. For an $n \times d$ data matrix $\mathbf{A}$ with rows $\mathbf{a}_i^\top$, an n-dimensional target vector $\mathbf{y}$ (with ± 1 entries y_i) and a regularization parameter γ, our task is to minimize:

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \mathrm{e}^{-y_i \mathbf{a}_i^\top \mathbf{x}}) + \frac{\gamma}{2}\|\mathbf{x}\|^2. \tag{7}$$

As a dataset, in Section 3, we used the Extended MNIST dataset of handwritten digits [EMNIST, 9] with $n = 500$k datapoints. Here, we also include results on the CIFAR-10 image dataset with $n = 50$k datapoints. Both datasets are preprocessed in the same way: Each image is transformed by a random features map that approximates a Gaussian kernel having width 0.002, and we partitioned the classes into two labels 1 and -1. We considered two feature dimensions: $d = 500$ and $d = 1000$, and we used the regularization parameter $\gamma = 10^{-8}$. To measure the error in the convergence plots, we use $\|\mathbf{x}_t - \mathbf{x}^*\|_{\mathbf{H}}^2 / \|\mathbf{x}_0 - \mathbf{x}^*\|_{\mathbf{H}}^2$, where $\mathbf{H} = \nabla^2 f(\mathbf{x}^*)$.

We next present further results, studying the convergence properties of SVRN on synthetic datasets with varying properties, for the logistic regression task as in (7). To construct our synthetic data matrices, we first generate an $n \times d$ Gaussian matrix $\mathbf{G}$, and let $\mathbf{G} = \mathbf{UDV}$ be the reduced SVD of that matrix (we used $n = 500$k and $d = 1000$). Then, we replace diagonal matrix $\mathbf{D}$ with a matrix $\tilde{\mathbf{D}}$ that has singular values spread linearly from 1 to $\kappa_{\mathbf{A}}$. We then let $\mathbf{A} = \mathbf{U}\tilde{\mathbf{D}}\mathbf{V}$ be our data matrix. To generate the vector $\mathbf{y}$, we first draw a random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, 1/d \cdot \mathbf{I}_d)$, and then we let $\mathbf{y} = \mathrm{sign}(\mathbf{Ax})$.

Logistic regression with varying condition number. To supplement the EMNIST logistic regression experiments in Section 3, we present convergence of SVRN-HA on the CIFAR-10 dataset, as well as for the synthetic logistic regression task while varying the squared condition number $\kappa_{\mathbf{A}}^2$ of the data matrix. Note that, while $\kappa_{\mathbf{A}}^2$ is not the same as the condition number of the finite-sum minimization problem, it is correlated with it, by affecting the convexity and smoothness of f. From Figure 3, we observe that SVRN-HA outperforms SN-HA for both values of the data condition number. However, the convergence of both algorithms gets noticeably slower after increasing $\kappa_{\mathbf{A}}^2$, while it does not have as much of an effect on the Newton's method. Given that the increased condition number affects both methods similarly, we expect that the degradation in performance is primarily due to worse Hessian approximations, rather than increased variance in the gradient estimates. This may be because we are primarily affecting the global convexity of f, as opposed to the smoothness of individual components ψ_i. See our high-coherence least squares experiments for a discussion of how the smoothness of component functions affects the performances of SVRN and SN very differently.