Least-squares regressions via randomized Hessians

Nabil Kahale NKAHALE@ESCP.EU
ESCP Business School, Paris, France

Abstract

We consider the least-squares regression problem with n points in dimension d. We analyze a novel approach, based on randomizing the Hessian matrix, to approximately solve this problem. The new algorithm is a variant of the averaged stochastic gradient descent method (SGD) with constant step-size. However, its updating rule relies on the entire response vector, and its convergence properties do not depend on the residuals. Without strong convexity assumptions, it is proven that the algorithm achieves a convergence rate for function values of $O(1/k)$ after k iterations, where the constant behind the O notation does not depend explicitly on the smallest eigenvalue of the Hessian matrix. The algorithm has a preprocessing cost proportional to the input size, and the running time of each iteration is proportional to the dimension. In the strongly-convex case, a restart version of the algorithm yields a convergence rate of $O(k^{-l})$ in $O(ld(n+k))$ time for arbitrary $l \geq 2$, where the constant behind the O notation depends on l and on the smallest eigenvalue of the Hessian matrix. Our theoretical results are illustrated with numerical experiments.

1. Introduction

The recent availability of massive volumes of data fosters the need to design computationally efficient algorithms for optimization in high dimensions. In large-scale machine learning, stochastic gradient descent algorithms are among the most effective optimization methods [2]. For general smooth convex functions, averaged SGD achieves the rate of convergence of $O(1/\sqrt{k})$ after k iterations [14]. For strongly-convex functions, i.e. when the smallest eigenvalue of the Hessian matrix is bounded away from 0, the convergence rate after k iterations is $O(1/k)$ [14]. Variance-reduced SGD algorithms that optimize the sum of n convex functions are described in [11, 20, 22], and related accelerated methods are analysed in [13, 15, 21, 23]. These methods enjoy linear convergence (a convergence rate that decreases exponentially with the number of iterations) in the strongly-convex case. For general smooth convex functions, the stochastic average gradient method (SAG) of Schmidt et al. [20] yields a convergence rate of $O(\sqrt{n}/k)$ after k iterations.

This paper focuses on the least-squares regression, which often arises in scientific computing and data analysis, and is widely used for inference and prediction. Many of the modern machine learning techniques such as the logistic and ridge regressions, the lasso method and neural networks can be considered as extensions of the least-squares regression technique. Given a non-zero $n \times d$ matrix X and an n-dimensional column vector Y, the least-squares problem consists of minimizing the function $g(\theta) = (2n)^{-1}||X\theta - Y||^2$, where θ ranges over all d-dimensional column vectors. An exact solution to this problem can be found in $O(nd^2)$ time [9]. Rokhlin and Tygert [19] describe a randomized algorithm based on a preconditioning matrix that computes in $O(d^3 + nd\ln(d/\epsilon))$ time a vector θ that minimizes $g(\theta)$ to relative precision ϵ. In the strongly-convex case, variants of SGD described in [8, 11, 20, 22] yield linear convergence rates for least-squares regressions. In typical high-dimensional machine learning problems, however, explanatory variables are highly

correlated, and so the smallest eigenvalue of the Hessian matrix is very close or equal to 0. For non-strongly-convex linear regressions, Bach and Moulines [1] show a convergence rate of $O(1/k)$ after k iterations for an averaged SGD algorithm with constant step-size in an on-line setting. Extensions and a detailed asymptotic analysis of the performance of this algorithm are given in [4]. Pilanci and Wainwright [17] provide algorithms for constrained least-squares through a random projection on a lower dimensional space. Dieuleveut et al. [6] study an averaged accelerated regularized SGD algorithm for least-squares regressions. Mini-batching and tail-averaging SGD algorithms for least-squares regressions are analyzed in [10]. Pillaud-Vivien et al. [18] show that, for hard linear regression problems, multiple passes of SGD lead to statistically optimal predictions while a single pass does not.

This paper studies a novel algorithm based on randomizing the Hessian matrix with averaging (RHA) that minimizes $g(\theta)$ at an arbitrary precision. Our approach is a variant of SGD, and each iteration of the algorithm takes $O(d)$ time. The algorithm has $O(nd)$ pre-processing cost and achieves a convergence rate of $O(1/k)$ after k iterations in the non-strongly-convex case. It enjoys the following properties:

1. Our method takes $O(nd + \bar{L}||\theta^* - \theta_0||^2 d^{3/2}/\epsilon)$ time to minimize $g(\theta)$ with expected error ϵ, where $\bar{L} := \mathrm{tr}(X^T X)/n$ is the average squared norm of a line of X, θ_0 is the starting point, and θ^* is any d-dimensional column vector that minimizes g. Thus, the time required by our algorithm to minimize $g(\theta)$ with a given expected error does not depend explicitly on the smallest eigenvalue of the Hessian matrix or on the residuals. Note that, for general matrices, the time to read the input is of order nd. For $\epsilon > 0$, the averaged SGD achieves precision ϵ in $O(nd + (\sigma^2 d^2 + R^2||\theta^* - \theta_0||^2 d)/\epsilon)$ time [1], including the time to read the input, where σ and R are real numbers satisfying certain conditions given in [1]. The smallest valid choice for σ^2 ranges between a weighted average of the squared residuals and the maximum squared residual. Similarly, the smallest valid choice for R^2 ranges between $\bar{L}$ and the maximum squared norm of a line of X. Depending on the problem instance, the upper bound on the time required by our algorithm to achieve precision ϵ can be higher or lower than that of the averaged SGD, even if all lines of X have the same norm.

2. When θ^* is drawn uniformly at random from $S(\theta_0, \rho)$, where $S(\theta_0, \rho) = \{\theta \in \mathbb{R}^{d\times 1} : ||\theta - \theta_0|| = \rho\}$ for $\rho > 0$, the time required by our algorithm to minimize $g(\theta)$ with expected error ϵ is $O(nd + \bar{L}\rho^2 d/\epsilon)$. Given a target precision ϵ, this average–case bound on the running time of our algorithm is never higher (up to absolute constants) than the aforementioned worst–case bound on the running time of the averaged SGD. We are not aware of an average–case analysis of the averaged SGD in the above sense. It is well known that the average–case running time of certain algorithms, such as the quicksort sorting algorithm [3], is lower than their worst–case running time. Pedregosa and Scieur [16] analyse the average–case behavior of gradient–based optimization algorithms for random quadratic problems.

3. When the Hessian matrix is invertible, for any $l \geq 2$, a restart version of our algorithm achieves a rate of convergence of $O(k^{-l})$ after k iterations. The restart algorithm can be simulated without any knowledge on the smallest eigenvalue of the Hessian matrix. The constant behind the O-notation depends on l and on the smallest eigenvalue of the Hessian matrix.

Our algorithm uses a non-uniform sampling scheme with sampling probabilities determined by the squared norm of each row vector. A similar sampling scheme has been applied by Frieze et al. [7] in the context of low-rank approximations of a matrix, by Strohmer and Vershynin [24] to approximately solve linear systems via an iterative algorithm, and by Défossez and Bach [4] to design an averaged SGD for least-squares regressions. However, while the updating rule of the conventional SGD, of Strohmer and Vershynin [24] and of Défossez and Bach [4] uses a single random coordinate of Y, our updating rule uses a vector that depends on the entire vector Y. Strohmer and Vershynin [24] establish a linear convergence rate for their method in the strongly-convex case. Défossez and Bach [4] analyse the asymptotic properties (as the number of iterations goes to infinity) of their algorithm, but do not study its performance after a finite number of iterations. Kahalé [12] uses a recursion similar to our updating rule to approximately simulate high-dimensional Gaussian vectors with a given covariance matrix.

Section 2 describes our method and its properties. Section 3 gives numerical experiments. Omitted proofs will appear in the final version of the paper. The running time refers to the number of arithmetic operations.

2. The algorithm description and properties

For $1 \leq i \leq n$, let e_i be the n-dimensional column vector whose i-th component is 1 and remaining components are 0, and let

$$p_i = \frac{||X^T e_i||^2}{\mathrm{tr}(X^T X)}. \tag{1}$$

Note that the numerator in (1) is the sum of the squared entries of the i-th line of X, while the denominator is the sum of squared entries of X. Thus the p_i's sum up to 1. Let $(i(k) : k \geq 0)$ be independent integral random variables on $\{1, \ldots, n\}$ such that, for $1 \leq j \leq n$,

$$\Pr(i(k) = j) = p_j.$$

Given an initial d-dimensional column vector θ_0 and a real number $\alpha \in (0, 1]$, define the sequence of d-dimensional column vectors $(\theta_k : k \geq 0)$ via the recursion

$$\theta_{k+1} = \theta_k - \alpha((u_k^T(\theta_k - \theta_0))u_k - c) \tag{2}$$

for $k \geq 0$, where $u_k = ||X^T e_{i(k)}||^{-1}(X^T e_{i(k)})$ and $c = \mathrm{tr}(X^T X)^{-1} X^T (y - X\theta_0)$. For $k \geq 1$, let

$$\bar{\theta}_k = \frac{\theta_0 + \cdots + \theta_{k-1}}{k}.$$

Theorem 1 shows that $E(g(\bar{\theta}_k))$ converges to the minimum of the function g at a rate of $O(1/k)$.

Theorem 1 *For $k \geq 1$, we have*

$$E(g(\bar{\theta}_k)) - g(\theta^*) \leq \frac{5\alpha^{-1} + 4\alpha d}{2} \bar{L} \frac{||\theta^* - \theta_0||^2}{k}.$$

If $\alpha = 1/\sqrt{d}$, Theorem 1 implies that

$$E(g(\bar{\theta}_k)) - g(\theta^*) \leq \frac{9\sqrt{d}}{2} \bar{L} \frac{||\theta^* - \theta_0||^2}{k}. \tag{3}$$

As $X^T e_i$ is the i-th column of X^T, for $1 \leq i \leq n$, the total time to calculate c and the p_i's is $O(nd)$. After an initial preprocessing cost of $O(n)$, the random variable $i(k)$ can be simulated in constant time using the alias method [5, Section III.4]. Thus the cost of each iteration is $O(d)$. Setting $\alpha = 1/\sqrt{d}$, the total time for our algorithm to minimize $g(\theta)$ with expected error ϵ is $O(nd + \bar{L}||\theta^* - \theta_0||^2 d^{3/2}/\epsilon)$.

When $\alpha = 1$ and θ^* is drawn uniformly at random from $S(\theta_0, \rho)$, Theorem 2 below gives a bound on the average convergence rate for $E(g(\bar{\theta}_k))$ that improves upon (3) by a factor of order $\sqrt{d}$.

Theorem 2 *Assume that θ^* is drawn uniformly at random from $S(\theta_0, \rho)$. Set $\alpha = 1$. Then, for $k \geq 1$, we have*

$$E(g(\bar{\theta}_k)) - g(\theta^*) \leq 5\bar{L}\frac{\rho^2}{k}.$$

In particular, if $\alpha = 1$ and θ^* is drawn uniformly at random from $S(0, \rho)$, the total time for our algorithm to minimize $g(\theta)$ with expected error ϵ is $O(nd + \bar{L}\rho^2 d/\epsilon)$.

2.1. The restart algorithm

Theorems 1 and 2 are valid even when $X^T X$ is not invertible. Assume now that $X^T X$ is invertible and denote by μ the smallest eigenvalue of $n^{-1}X^T X$. By the well-known equality

$$g(\theta) - g(\theta^*) \geq \frac{\mu}{2}||\theta - \theta^*||^2,$$

Theorem 1 implies that

$$E(g(\bar{\theta}_k)) - g(\theta^*) \leq (5\alpha^{-1} + 4\alpha d)\frac{\bar{L}}{\mu k}(g(\theta_0) - g(\theta^*)).$$

Given $k \geq 1$, let T_k be the random operator that maps any d-dimensional column vector θ_0 to $\bar{\theta}_k$. For $l \geq 1$, denote by T_k^l the random operator on the set of d-dimensional column vectors obtained by composing l times the operator T_k. It can be shown by induction on l that, for any d-dimensional column vector θ,

$$E(g(T_k^l(\theta))) - g(\theta^*) \leq ((5\alpha^{-1} + 4\alpha d)\frac{\bar{L}}{\mu k})^l (g(\theta) - g(\theta^*)).$$

$T_k^l(\theta)$ can be calculated from θ in $O(ld(n+k))$ time. This implies that $E(g(T_k^l(\theta_0))) - g(\theta^*)$ is of order k^{-l} as k goes to infinity, for any fixed $l \geq 2$. Observe that $T_k^l(\theta)$ can be simulated without the explicit knowledge of μ.

3. Numerical experiments

Our numerical experiments are conducted on the sonar[1], madelon[1] and sido0[2] binary data sets, whose characteristics are summarized in Table 1. The variables were centered, a constant variable was added to each data set, and all variables were normalized. We have implemented the following methods using the null vector as starting point:

1. http://archive.ics.uci.edu
2. http://www.causality.inf.ethz.ch

Table 1: Data sets used in the simulations

Data set	Variables	Data Points
sonar	60	208
madelon	500	2000
sido0	4932	12678

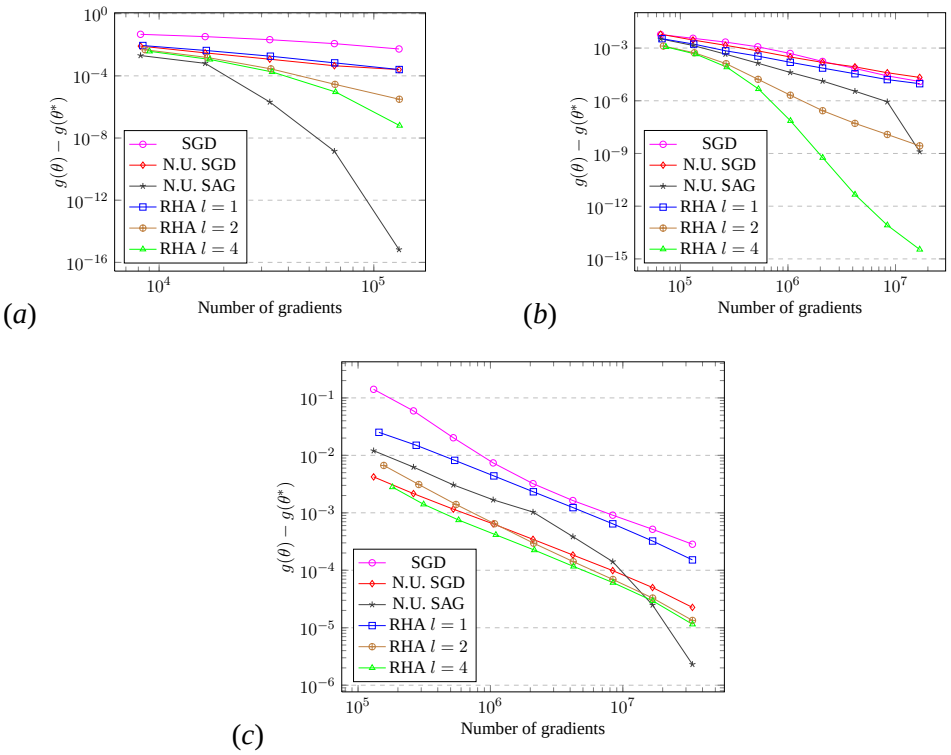

Figure 1: Convergence on the sonar, madelon and sido0 datasets

- the averaged SGD algorithm based on [1].
- the averaged SGD algorithm with non-uniform probabilities based on [4].
- the SAG algorithm with non-uniform probabilities, where the lines are sampled according to the p_i's, and the output is the vector among the final iterate and the average of iterates that minimizes g.
- the RHA method with level l that calculates $T_k^l(0)$, for $l \in \{1, 2, 4\}$. We set $\alpha = 1$ in our experiments because smaller values of α did not increase the performance of our method for moderate values of k.

The results are reported in Figure 1. The running time is measured by the number of gradient computations, that is k for the averaged SGD and SAG methods and $l(n+k)$ for the RHA algorithm, where k is the number of iterations.

References

[1] Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n). In *Advances in neural information processing systems*, pages 773–781, 2013.

[2] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[4] Alexandre Défossez and Francis Bach. Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *Artificial Intelligence and Statistics*, pages 205–213, 2015.

[5] L Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New-York, 1986.

[6] Aymeric Dieuleveut, Nicolas Flammarion, and Francis Bach. Harder, better, faster, stronger convergence rates for least-squares regression. *The Journal of Machine Learning Research*, 18(101):1–51, 2017.

[7] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[8] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548, 2015.

[9] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU Press, Baltimore, fourth edition, 2013.

[10] Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18(223):1–42, 2018.

[11] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[12] Nabil Kahalé. Efficient simulation of high dimensional Gaussian vectors. *Mathematics of Operations Research*, 44(1):58–73, 2019.

[13] Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, 171(1-2):167–215, 2018.

[14] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4): 1574–1609, 2009.

[15] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[16] F. Pedregosa and D. Scieur. Average-case acceleration through spectral density estimation. In *International Conference on Machine Learning*, 2020.

[17] 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 (53):1–38, 2016.

[18] Loucas Pillaud-Vivien, Alessandro Rudi, and Francis Bach. Statistical optimality of stochastic gradient descent on hard learning problems through multiple passes. In *Advances in Neural Information Processing Systems*, pages 8114–8124, 2018.

[19] 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.

[20] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[21] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. *Mathematical Programming*, pages 1–37, 2018.

[22] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

[23] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, pages 64–72, 2014.

[24] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009.