Weighted SGD for ℓ_p Regression with Randomized Preconditioning

Jiyan Yang [*] Yin-Lam Chow [*] Christopher Ré [†] Michael W. Mahoney [‡]

Abstract

In recent years, stochastic gradient descent (SGD) methods and randomized linear algebra (RLA) algorithms have been applied to many large-scale problems in machine learning and data analysis. SGD methods are easy to implement and applicable to a wide range of convex optimization problems. In contrast, RLA algorithms provide much stronger worst-case performance guarantees but are applicable to a narrower class of problems. We aim to bridge the gap between these two classes of methods in solving constrained overdetermined linear regression problems—e.g., ℓ_2 and ℓ_1 regression problems.

- We propose a hybrid algorithm named PWSGD that uses RLA techniques for preconditioning and constructing an importance sampling distribution, and then performs an SGD-like iterative process with weighted sampling on the preconditioned system.
- By rewriting the ℓ_p regression problem into a stochastic optimization problem, we connect PWSGD to several existing ℓ_p solvers including RLA methods with algorithmic leveraging (RLA for short).
- We prove that PWSGD inherits faster convergence rates that only depend on the lower dimension of the linear system, while maintaining low computation complexity. Such SGD convergence rate is superior to other related SGD algorithms such as the weighted randomized Kaczmarz algorithm.
- Particularly, when solving ℓ_1 regression with size n by d, PWSGD returns an approximate solution with ϵ relative error on the objective value in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)/\epsilon^2)$ time. This complexity is *uniformly* better than that of RLA methods in terms of both ϵ and d when the problem is unconstrained. In the presence of constraints, PWSGD only has to solve a sequence of much simpler and smaller optimization problem over the same constraints. In general this is more efficient than solving the constrained subproblem required in RLA.
- For ℓ_2 regression, PWSGD returns an approximate solution with ϵ relative error on the objective value and solution vector in prediction norm in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)\log(1/\epsilon)/\epsilon)$ time. We show that when solving unconstrained ℓ_2 regression, this complexity is comparable to that of RLA and is asymptotically better over several state-of-the-art solvers in the regime where the desired accuracy ϵ, high dimension n and low dimension d satisfy $d \geq 1/\epsilon$ and $n \geq d^2/\epsilon$.

Finally, the effectiveness of such algorithms is illustrated numerically on both synthetic and real datasets, and the results are consistent with our theoretical findings and demonstrate that PWSGD converges to a medium-precision solution, e.g., $\epsilon = 10^{-3}$, more quickly than other methods.

1 Introduction

Many novel algorithms for large-scale data analysis and machine learning problems have emerged in recent years, among which stochastic gradient descent (SGD) methods and randomized linear algebra (RLA) algorithms have received much attention—both for their strong performance in practical applications and for their interesting theoretical properties [3, 18]. Here, we consider the ubiquitous ℓ_1 and ℓ_2 regression problems, and we describe a novel RLA-SGD algorithm called PWSGD. Our new algorithm combines the advantages of both RLA and SGD methods for solving constrained overdetermined ℓ_1 and ℓ_2 regression problems.

Consider the overdetermined ℓ_p regression problem

$$\min_{x \in \mathcal{Z}} f(x) = \|Ax - b\|_p, \tag{1.1}$$

where $p \in [1, \infty]$, $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ and $n \gg d$. When $\mathcal{Z} = \mathbb{R}^d$, i.e., the solution space is unconstrained, the cases $p \in \{1, 2\}$ are respectively known as the Least Absolute Deviations (LAD, or ℓ_1) and Least-squares (LS, or ℓ_2) regression problems. Classically, the unconstrained ℓ_2 regression problem can be solved by eigenvector-based methods with worst-case running time $\mathcal{O}(nd^2)$ [14]; or by iterative methods for which the running time depends on the condition number of A [2, 17, 27], while the unconstrained ℓ_1 regression problem can be formulated as a linear program [26, 6] and solved by an interior-point method [26, 25].

For these and other regression problems, SGD algorithms are widely used in practice because of their scalability and efficiency. In contrast, RLA algorithms have better

[*]Institute for Computational and Mathematical Engineering, Stanford University, Email: {jiyan, ychow}@stanford.edu

[†]Department of Computer Science, Stanford University, Email: chrismre@cs.stanford.edu

[‡]International Computer Science Institute and Department of Statistics, University of California, Berkeley, mmahoney@stat.berkeley.edu

theoretical guarantees but (thus far) have been less flexible, e.g., in the presence of constraints. For example, they may use an interior point method for solving a constrained subproblem, and this may be less efficient than SGD. (Without constraints, RLA methods can be used to construct subproblems to be solved exactly, or they can be used to construct preconditioners for the original problem; see [38] for details and implementations of these RLA methods to compute low, medium, and high precision solutions on up to terabyte-sized input data.) In this paper, we combine these two algorithmic approaches to develop a method that takes advantage of the strengths of both of these approaches.

This conference paper presents our main results; and more details on our proofs, our empirical evaluations, and connections between our results and stochastic optimization and coreset methods may be found in the longer technical report version of this paper [36].

1.1 Overview of our main algorithm.

Our main algorithm pwSGD is a hybrid method for solving constrained overdetermined ℓ_1 and ℓ_2 regression problems. It consists of two main steps. First, apply RLA techniques for *preconditioning* and construct an importance sampling distribution. Second, apply an SGD-like iterative phase with *weighted* sampling on the preconditioned system. Such an algorithm preserves the simplicity of SGD and the high quality theoretical guarantees of RLA. We show that, with a proper choice of preconditioner, pwSGD runs in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)/\epsilon^2)$ time to return an approximate solution with ϵ relative error in the objective for constrained ℓ_1 regression or in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)\log(1/\epsilon)/\epsilon)$ time to return an approximate solution with ϵ relative error in the solution vector in prediction norm for constrained ℓ_2 regression. Furthermore, for unconstrained ℓ_2 regression, pwSGD runs in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + d^3 \log(1/\epsilon)/\epsilon)$ time to return an approximate solution with ϵ relative error in the objective.

To provide a quick overview of how pwSGD compares to existing algorithms, in Tables 1 and 2, we summarize the complexity required to compute a solution $\hat{x}$ with relative error $(f(\hat{x}) - f(x^*))/f(x^*) = \epsilon$, of several solvers for unconstrained ℓ_1 and ℓ_2 regression. In Table 1, RLA with algorithmic leveraging (RLA for short) [8, 37] is a popular method for obtaining a low-precision solution, and randomized IPCPM is an iterative method for finding a higher-precision solution [20] for unconstrained ℓ_1 regression. Clearly, pwSGD has a uniformly better complexity than that of RLA methods in terms of both d and ϵ, no matter which underlying preconditioning method is used. This makes pwSGD a more suitable candidate for getting a medium-precision, e.g., $\epsilon = 10^{-3}$, solution.

In Table 2, all the methods require constructing a sketch first. Among them, "low-precision" solvers refer to "sketching + direct solver" type algorithms; see [12, 9] for projection-based examples and [9, 11] for sampling-based examples. "High-precision" solvers refer to "sketching + preconditioning + iterative solver" type algorithms; see [1, 21] for examples. One can show that, when $d \geq 1/\epsilon$ and $n \geq d^2/\epsilon$, pwSGD is asymptotically better than all the solvers shown in Table 2. Moreover, although high-precision solvers are more efficient when a high-precision solution is desired, usually they are designed for unconstrained problems, whereas pwSGD also works for constrained problems. See Section 3.3 for a more detailed discussion.

We remark that, compared to general SGD algorithms, our RLA-SGD hybrid algorithm pwSGD works for problems in a narrower range, i.e., ℓ_p regression, but inherits the strong theoretical guarantees of RLA. When solving ℓ_2 regression, for which traditional RLA methods are well designed, pwSGD has a comparable complexity. On the other hand, when solving ℓ_1 regression, due to the efficiency of SGD updates, pwSGD has a strong advantage over traditional RLA methods. See Sections 3.3 and 3.4 for more detailed discussions.

Finally, in Section 4, empirically we show that pwSGD performs favorably compared to other competing methods, as it converges to a medium-precision solution more quickly. More details on this can be found in the technical report version of this paper [36].

1.2 Connection to related algorithms.

As an interesting point of potentially-independent interest, a connection between ℓ_p regression and stochastic optimization will allow us to unify our main algorithm pwSGD and some existing ℓ_p regression solvers under the same framework. In Figure 1, we present the basic structure of this framework, which provides a view of pwSGD from another perspective. To be more specific, we reformulate (deterministic) overdetermined ℓ_p regression problems of the form (1.1) into a stochastic optimization problem of the form (1.2).

Proposition 1.1. *Let $U \in \mathbb{R}^{n\times d}$ be a basis of the range space of A in the form of $U = AF$, where $F \in \mathbb{R}^{d\times d}$. The constrained overdetermined ℓ_p regression problem* (1.1) *is equivalent to*

$$\min_{y\in\mathcal{Y}} \|Uy - b\|_p^p = \min_{y\in\mathcal{Y}} \mathbb{E}_{\xi\sim P}\left[H(y,\xi)\right], \tag{1.2}$$

where ξ is a random variable over $\{1,\ldots,n\}$ with distribution $P = \{p_i\}_{i=1}^n$, y is the decision variable in $\mathcal{Y}$ and $H(y,\xi) = |U_\xi y - b_\xi|^p/p_\xi$. The constraint set of y is given by $\mathcal{Y} = \{y \in \mathbb{R}^d | y = F^{-1}x, x \in \mathcal{Z}\}$.

As suggested in Figure 1, to solve this stochastic optimization problem, typically one needs to answer the following three questions.

solver	complexity (general)	complexity (sparse)
RLA with algorithmic leveraging	$time(R) + \mathcal{O}(\text{nnz}(A)\log n + \bar{\kappa}_1^{\frac{3}{2}} d^{\frac{9}{2}}/\epsilon^3)$	$\mathcal{O}(\text{nnz}(A)\log n + d^{\frac{69}{8}} \log^{\frac{25}{8}} d/\epsilon^{\frac{5}{2}})$
randomized IPCPM	$time(R) + nd^2 + \mathcal{O}((nd + \text{poly}(d))\log(\bar{\kappa}_1 d/\epsilon))$	$\mathcal{O}(nd\log(d/\epsilon))$
PWSGD	$time(R) + \mathcal{O}(\text{nnz}(A)\log n + d^3\bar{\kappa}_1/\epsilon^2)$	$\mathcal{O}(\text{nnz}(A)\log n + d^{\frac{13}{2}} \log^{\frac{5}{2}} d/\epsilon^2)$

Table 1: Summary of complexity of several unconstrained ℓ_1 solvers that use randomized linear algebra. The target is to find a solution $\hat{x}$ with accuracy $(f(\hat{x}) - f(x^*))/f(x^*) \leq \epsilon$, where $f(x) = \|Ax - b\|_1$. In the above, $time(R)$ denotes the time needed to compute a matrix R such that AR^{-1} is well-conditioned with condition number $\bar{\kappa}_1$ (see Definition 1). The general complexity bound and the one using sparse reciprocal exponential transform [35] as the underlying sketching method are presented. Here, we assume $n \gg d$ such that $n > d^3 \log^3 d$ and the underlying ℓ_1 regression solver in RLA with algorithmic leveraging algorithm takes $\mathcal{O}(n^{\frac{5}{4}} d^3)$ time to return a solution [26]. The complexity of each algorithm is computed by setting the failure probability to be a constant.

solver	complexity (SRHT)	complexity (CW)
low-precision solvers (projection)	$\mathcal{O}\left(nd\log(d/\epsilon) + d^3\log(nd)/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A) + d^4/\epsilon^2\right)$
low-precision solvers (sampling)	$\mathcal{O}\left(nd\log n + d^3\log d + d^3\log d/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A)\log n + d^4 + d^3\log d/\epsilon\right)$
high-precision solvers	$\mathcal{O}\left(nd\log d + d^3\log d + nd\log(1/\epsilon)\right)$	$\mathcal{O}\left(\text{nnz}(A) + d^4 + nd\log(1/\epsilon)\right)$
PWSGD	$\mathcal{O}\left(nd\log n + d^3\log d + d^3\log(1/\epsilon)/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A)\log n + d^4 + d^3\log(1/\epsilon)/\epsilon\right)$

Table 2: Summary of complexity of several unconstrained ℓ_2 solvers that use randomized linear algebra. The target is to find a solution $\hat{x}$ with accuracy $(f(\hat{x}) - f(x^*))/f(x^*) \leq \epsilon$, where $f(x) = \|Ax - b\|_2$. Two sketching methods, namely, SRHT [12, 33] and CW [9] are considered. Here, the complexity of each algorithm is computed by setting the failure probability to be a constant.

- ($C1$): How to sample: SAA (Sampling Average Approximation, i.e., draw samples in a batch mode and deal with the subproblem) or SA (Stochastic Approximation, i.e., draw a mini-batch of samples in an online fashion and update the weight after extracting useful information)?
- ($C2$): Which probability distribution P (uniform distribution or not) and which basis U (preconditioning or not) to use?
- ($C3$): Which solver to use (e.g., how to solve the subproblem in SAA or how to update the weight in SA)?

Some combinations of these choices may lead to existing solvers; see Figure 1 (as well as [36]) for more details. In the following, we briefly outline several connections. Using the SAA approach with a naive choice of $U = A$ and uniform distribution P leads to the vanilla subsampling algorithm. Importantly, such a simple algorithm might fail (ungracefully) for worst-case input. On the other hand, RLA methods (in particular, those that exploit algorithmic averaging; see Appendix B and also [11, 38]) inherit strong theoretical guarantees because the underlying sampling distribution P captures most of the important information of the original system; moreover, such a carefully constructed leverage-based distribution is defined based on a well-conditioned basis U, e.g., an orthogonal matrix for $p = 2$.

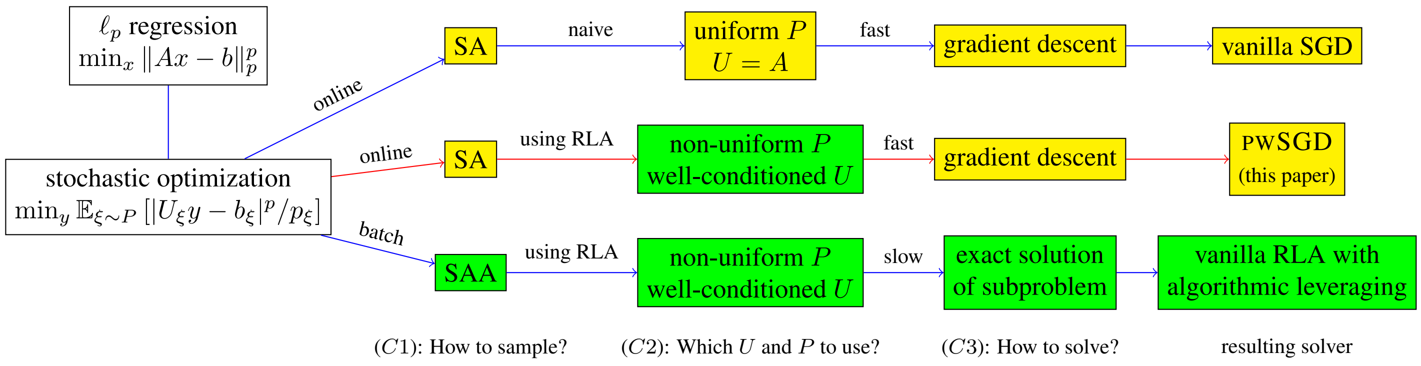
A natural question arises: can we leverage the algorithmic benefits of RLA preconditioning to improve the performance of SGD-type algorithms? One immediate idea is to develop an SGD-like algorithm that uses the same choice of U and P as in RLA methods. Indeed, this simple idea leads to our main algorithm PWSGD, which is an online algorithm ($C1$) that uses a non-uniform sampling distribution ($C2$) and performs a gradient descent update ($C3$) on a preconditioned system ($C2$), as Figure 1 suggests.

Indeed, for least-squares problems (unconstrained ℓ_2 regression), PWSGD is highly related to the weighted randomized Kaczmarz (RK) algorithm [32, 22] in the way that both algorithms perform SGD updates with non-uniform sampling distribution P. An important difference is that PWSGD runs on a well-conditioned basis U while randomized RK doesn't involve preconditioning. In Section 3.5 we show that this preconditioning step dramatically reduces the number of iteration required for PWSGD to converge to a (fixed) desired accuracy.

1.3 Main contributions.

Now we are ready to state our main contributions.

- We reformulate the deterministic ℓ_p regression problem (1.1) into a stochastic optimization problem (1.2) and make connections to existing solvers including RLA methods with algorithmic leveraging and weighted randomized Kaczmarz algorithm (Sections 1.2 and 3.5).
- We develop a hybrid algorithm for solving constrained overdetermined ℓ_1 and ℓ_2 regression called PWSGD, which is an SGD algorithm with preconditioning and a non-uniform sampling distribution constructed using RLA techniques. We present several choices of the preconditioner and their tradeoffs. We show that with a suitable preconditioner, the convergence rate of the SGD phase only depends on the low dimension d, and is independent of the high dimension n (Sections 3.1 and 3.2).
- We prove that PWSGD returns an approximate so-



ℓ_p regression $\min_x \|Ax - b\|_p^p$
stochastic optimization $\min_y \mathbb{E}_{\xi\sim P}\left[|U_\xi y - b_\xi|^p / p_\xi\right]$
online
online
batch
SA
SA
SAA
naive
using RLA
using RLA
uniform P $U = A$
non-uniform P well-conditioned U
non-uniform P well-conditioned U
fast
fast
slow
gradient descent
gradient descent
exact solution of subproblem
vanilla SGD
PWSGD (this paper)
vanilla RLA with algorithmic leveraging
($C1$): How to sample?
($C2$): Which U and P to use?
($C3$): How to solve?
resulting solver

Figure 1: An overview of our framework for solving ℓ_p regression via stochastic optimization. To construct a solver, three choices have to be made. For ($C1$), the answer can be either SAA (Sampling Average Approximation, i.e., sample a batch of points and deal with the subproblem) or SA (Stochastic Approximation, i.e., sample a mini-batch in an online fashion and update the weight vector after extracting useful information). In ($C2$), the answer is determined by P, which denotes the underlying sampling distribution (uniform or nonuniform), and U, which denotes the basis with which to work (original or preconditioned system). Finally, for ($C3$), the answer determines how we solve the subproblem (in SAA) or what information we extract and how we update the weight (in SA).

lution with ϵ relative error in the objective value in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)/\epsilon^2)$ time for ℓ_1 regression. This complexity is *uniformly* better than that of RLA methods in terms of both ϵ and d when the problem is unconstrained. In the presence of constraints, PWSGD only has to solve a sequence of much simpler and smaller optimization problems over the same constraints, which in general can be more efficient than solving the constrained subproblem required in RLA (Sections 3.3 and 3.4).

- We prove that PWSGD returns an approximate solution with ϵ relative error in the objective value and the solution vector measured in prediction norm in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)\log(1/\epsilon)/\epsilon)$ time for ℓ_2 regression. We show that for unconstrained ℓ_2 regression, this complexity is asymptotically better than several state-of-the-art solvers in the regime where $d \geq 1/\epsilon$ and $n \geq d^2/\epsilon$ (Sections 3.3 and 3.4).
- Empirically, we show that when solving ℓ_1 and ℓ_2 regression problems, PWSGD inherits faster convergence rates and performs favorably in the sense that it obtains a medium-precision much faster than other competing SGD-like solvers do. Also, theories regarding several choices of preconditioners are numerically verified (Section 4).

1.4 Other prior related work.

Numerous RLA algorithms have been proposed to solve ℓ_p regression problems [38]. RLA theories show that to achieve a relative-error bound (with either sampling-based or projection-based methods), the required sketch size only depends on d, independent of n, and the running time also depends on the time to implement the random projection at the first step. Regarding the performance of unconstrained regression problems, in [10] the authors provide an algorithm that constructs a well-conditioned basis by ellipsoid rounding and a subspace-preserving sampling matrix for ℓ_p regression problems in $\mathcal{O}(nd^5 \log n)$ time; the algorithms in [9, 19, 23] solve the problem via sparse random projections in nearly input-sparsity time, i.e., $\mathcal{O}(\log n \cdot \mathrm{nnz}(A))$ time, plus lower-order terms. Finally, the algorithms in [1, 21] use RLA to compute a preconditioner and call iterative solvers such as LSQR to solve the preconditioned problem.

In contrast, SGD algorithms update the solution vector in an iterative fashion and are simple to implement and scalable to large datasets [5, 29, 4]. Moreover, these methods can be easily extended for problems with general convex loss functions and constraints, such as Pegasos [28] for regularized SVM and stochastic coordinate descent (SCD) for ℓ_1 regularization [30]. Several techniques, such as SAGE [15], AdaGrad [13], and SVRG [16], have recently been proposed to accelerate the convergence rate of SGD, and [24] also show that SGD is favorable for parallel/distributed computation. More recently, several works, e.g., [39, 22], regarding SGD with weighted sampling, have been proposed, in which the authors show that the performance of SGD can be improved by using a nonuniform sampling distribution.

2 Preliminaries

For any matrix $A \in \mathbb{R}^{n\times d}$, we use A_i and A^j to denote the i-th row and j-th column of A, respectively. We assume A has full rank, i.e., $\mathrm{rank}(A) = d$. Also denote by $\kappa(A)$ the usual condition number of A, by $\mathrm{nnz}(A)$ the number of nonzero elements in A, and by $\text{poly}(d)$ a low-degree polynomial in d. Throughout this section, the definitions are applied to general $p \in [1, \infty)$. We denote by $|\cdot|_p$ the element-wise ℓ_p norm of a matrix: $|A|_p = \left(\sum_{i=1}^{n} \sum_{j=1}^{d} |A_{ij}|^p\right)^{1/p}$. In particular, when $p = 2$, $|\cdot|_2$ is equivalent to the Frobenius norm.

The following two notions of well-conditioned bases and leverage scores are crucial to our methods. The first notion was originally introduced by [7] and stated more precisely in [10], and it is used to justify the well-posedness of a ℓ_p regression problem. The second notion was introduced by [10].

DEFINITION 1. (WELL-CONDITIONED BASIS) *A matrix $A \in \mathbb{R}^{n\times d}$ is (α, β, p)-conditioned if $|A|_p \le \alpha$ and for all $x \in \mathbb{R}^d$, $\beta\|Ax\|_p \ge \|x\|_q$, where $1/p + 1/q = 1$. Define $\bar{\kappa}_p(A)$ as the minimum value of $\alpha\beta$ such that A is (α, β, p)-conditioned. We say that a basis U for* $\text{range}(A)$ *is a well-conditioned basis if $\bar{\kappa}_p = \bar{\kappa}_p(U)$ is a low-degree polynomial in d, independent of n.*

The notion of leverage scores captures how important each row in the dataset is, and is used in the construction of the sampling probability.

DEFINITION 2. (ℓ_p LEVERAGE SCORES) *Given a matrix $A \in \mathbb{R}^{n\times d}$, suppose U is an (α, β, p) well-conditioned basis for* $\text{range}(A)$. *Then the i-th leverage score λ_i of A is defined as $\lambda_i = \|U_i\|_p^p$ for $i = 1, \ldots, n$.*

3 Our Main Algorithm

In this section, we will state our main algorithm PWSGD (Algorithm 1) for solving the constrained overdetermined ℓ_1 and ℓ_2 regression problems. We now summarize the main steps of this algorithm as follows.

First, we compute a well-conditioned basis U (Definition 1) for the range space of A implicitly via a conditioning method; see Table 4 and Table 5 in Appendix A for a summary of recently proposed randomized conditioning methods. We refer this as the "implicit" method, i.e., it focuses on computing $R \in \mathbb{R}^{d\times d}$ such that $U = AR^{-1}$. A typical way of obtaining R is via the QR decomposition of SA where SA is a sketch of A; see Appendix A for more details.

Second, we either exactly compute or quickly approximate the leverage scores (Definition 2), i.e., the row norms of U as $\{\lambda_i\}_{i=1}^n$. To compute $\{\lambda_i\}_{i=1}^n$ exactly, we have to form the matrix U explicitly, which takes time $\mathcal{O}(nd^2)$. Alternatively, in order to further reduce the running time, we can estimate the row norms of U without computing the product between A and R^{-1}; see Appendix A for more details. We assume that $\{\lambda_i\}_{i=1}^n$ satisfy

$$(3.3) \qquad (1-\gamma)\|U_i\|_p^p \le \lambda_i \le (1+\gamma)\|U_i\|_p^p,$$

where γ is the approximation factor of estimation. When the leverage scores are exact, the approximation factor $\gamma = 0$. From that, we can define a distribution P over $\{1, \ldots, n\}$ based on $\{\lambda_i\}_{i=1}^n$ as follows:

$$(3.4) \qquad p_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}.$$

Third, in each iteration a new sample corresponding to a row of A is drawn according to distribution P and we apply an SGD process to solve the following equivalent problem with a specific choice of $F \in \mathbb{R}^{d\times d}$:

$$(3.5) \qquad \min_{y\in\mathcal{Y}} h(y) = \|AFy - b\|_p^p = \mathbb{E}_{\xi\sim P}\left[|A_\xi Fy - b_\xi|^p / p_\xi\right].$$

Here the matrix F is called the preconditioner for the linear system being solved; see Section 3.2 for several choices of F. Below, we show that with a suitable choice of F, the convergence rate of the SGD phase can be improved significantly. Indeed, we can perform the update rule in the original domain (with solution vector x instead of y), i.e., (3.8). Notice that if $\mathcal{Z} = \mathbb{R}^d$ and $F = I$, then the update rule can be simplified as

$$(3.6) \qquad x_{t+1} = x_t - \eta c_t A_{\xi_t}.$$

If $\mathcal{Z} = \mathbb{R}^d$ and $F = R^{-1}$, then the update rule becomes

$$(3.7) \qquad x_{t+1} = x_t - \eta c_t H^{-1} A_{\xi_t},$$

where $H = (R^\top R)^{-1}$. In the presence of constraints, (3.8) only needs to solve an optimization problem with a quadratic objective over the same constraints whose size is independent of n.

Finally, the output is the averaged value over all iterates, i.e., $\bar{x} = \frac{1}{T}\sum_{t=1}^{\top} x_t$, for ℓ_1 regression, or the last iterate, i.e., x_T, for ℓ_2 regression.

3.1 Main results for ℓ_1 and ℓ_2 regression problems.

The quality-of-approximation of Algorithm 1 is presented in Proposition 3.1 and Proposition 3.2 for ℓ_1 and ℓ_2 regression, respectively, in which we give the expected number of iterations that PWSGD needs for convergence within small tolerance. We show that PWSGD inherits a convergence rate of $\mathcal{O}\left(1/\sqrt{T}\right)$ for ℓ_1 regression and $\mathcal{O}(\log T/T)$ for ℓ_2 regression and the constant term only depends on the lower dimension d when $F = R^{-1}$. Worth mentioning is that for ℓ_2 regression, our bound on the solution vector is measured in prediction norm, i.e., $\|Ax\|_2$. For completeness, we present the non-asymptotic convergence results of PWSGD in Proposition A.1 and Proposition A.2 in Appendix A. All the proofs can be found in the technical report version of this paper [36]. The analysis of these results is based on the usual convergence properties of SGD methods [36].

Algorithm 1 PWSGD— preconditioned weighted SGD for over-determined ℓ_1 and ℓ_2 regression

```
1: Input: $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$ with $\mathrm{rank}(A) = d$, $x_0 \in \mathcal{Z}$, $\eta$ and $T$.
2: Output: An approximate solution vector to problem $\min_{x\in\mathcal{Z}} \|Ax - b\|_p^p$ for $p = 1$ or 2.
3: Compute $R \in \mathbb{R}^{d\times d}$ such that $U = AR^{-1}$ is an $(\alpha, \beta)$ well-conditioned basis $U$ for the range of $A$.
4: Compute or estimate $\|U_i\|_p^p$ with leverage scores $\lambda_i$, for $i \in [n]$, that satisfies (3.3).
5: Let $p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$, for $i \in [n]$.
6: Construct the preconditioner $F \in \mathbb{R}^{d\times d}$ based on $R$; see Section 3.2 for details.
7: for $t = 0, \dots, T$ do
8:     Pick $\xi_t$ from $[n]$ based on distribution $\{p_i\}_{i=1}^n$.
9:     $c_t = \begin{cases} \mathrm{sgn}\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 1; \\ 2\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 2. \end{cases}$
10:    Update $x$ by
       (3.8) $x_{t+1} = \begin{cases} x_t - \eta c_t H^{-1} A_{\xi_t} & \text{if } \mathcal{Z} = \mathbb{R}^d; \\ \arg\min_{x\in\mathcal{Z}} \ \eta c_t A_{\xi_t} x + \frac{1}{2}\|x_t - x\|_H^2 & \text{o.w.} \end{cases}$
       where $H = \left(FF^\top\right)^{-1}$.
11: end for
12: Return $\bar{x}$ for $p = 1$ or $x_T$ for $p = 2$.
```

In the following results, R is the matrix computed in step 3 in Algorithm 1, $\{\lambda_i\}_{i\in[n]}$, are the leverage scores computed in step 4, F is the preconditioner chosen in step 6 in Algorithm 1, and $H = \left(FF^\top\right)^{-1}$. Denote by $\bar{\kappa}_p(U)$ the condition number of the well-conditioned basis $U = AR^{-1}$ and γ the approximation factor of the leverage scores λ_i, $i \in [n]$, that satisfies (3.3). For any vector $x \in \mathbb{R}^d$, denote by $\|x\|_H^2 = x^\top Hx$ the ellipsoidal norm of x induced by matrix $H = H^\top \succ 0$. For any non-singular matrix A, denote $\kappa(A) = \|A\|_2\|A^{-1}\|_2$ and $\hat{\kappa}(A) = |A|_1|A^{-1}|_1$. The exact form of the step-sizes used can be found in the proofs.

PROPOSITION 3.1. *For $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $f(x) = \|Ax - b\|_1$ and suppose $f(x^*) > 0$. Then there exists a step-size η such that after*

$$T = d\bar{\kappa}_1^2(U)\hat{\kappa}^2(RF)\frac{c_1^2 c_2 c_3^2}{\epsilon^2}$$

iterations, Algorithm 1 with $p = 1$ returns a solution vector estimate $\bar{x}$ that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[f(\bar{x})\right] - f(x^*)}{f(x^*)} \le \epsilon.$$

Here, the expectation is taken over all the samples $\xi_1, \dots, \xi_T$ and x^* is the optimal solution to the problem $\min_{x\in\mathcal{Z}} f(x)$. The constants in T are given by $c_1 = \frac{1+\gamma}{1-\gamma}$, $c_2 = \frac{\|x^*-x_0\|_H^2}{\|x^*\|_H^2}$ and $c_3 = \|Ax^*\|_1/f(x^*)$.

PROPOSITION 3.2. *For $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $f(x) = \|Ax - b\|_2$ and suppose $f(x^*) > 0$. Then there exists a step-size η such that after*

$$T = c_1\bar{\kappa}_2^2(U)\kappa^2(RF) \cdot \log\left(\frac{2c_2\kappa(U)\kappa^2(RF)}{\epsilon}\right) \cdot \left(1 + \frac{\kappa^2(U)\kappa^2(RF)}{c_3\epsilon}\right)$$

iterations, Algorithm 1 with $p = 2$ returns a solution vector estimate x_T that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[\|A(x_T - x^*)\|_2^2\right]}{\|Ax^*\|_2^2} \le \epsilon.$$

Furthermore, when $\mathcal{Z} = \mathbb{R}^d$ and $F = R^{-1}$, there exists a step-size η such that after

$$T = c_1\bar{\kappa}_2^2(U) \cdot \log\left(\frac{c_2\kappa(U)}{\epsilon}\right) \cdot \left(1 + \frac{2\kappa^2(U)}{\epsilon}\right)$$

iterations, Algorithm 1 with $p = 2$ returns a solution vector estimate x_T that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[f(x_T)\right] - f(x^*)}{f(x^*)} \le \epsilon.$$

Here, the expectation is taken over all the samples $\xi_1, \dots, \xi_T$, and x^* is the optimal solution to the problem $\min_{x\in\mathcal{Z}} f(x)$. The constants in T are given by $c_1 = \frac{1+\gamma}{1-\gamma}$, $c_2 = \frac{\|x^*-x_0\|_H^2}{\|x^*\|_H^2}$, $c_3 = \|Ax^*\|_2^2/f(x^*)^2$.

The above results indicate two important properties of PWSGD. First recall that the condition number $\bar{\kappa}_p(U)$ of the well-conditioned basis U is a polynomial of d that is independent of n. Thus with a preconditioner $F = R^{-1}$ and an appropriate step-size in PWSGD, the number of iterations T required to achieve an arbitrarily low relative error only depends on the *low dimension* d of the input matrix A. Second, PWSGD is *robust* to leverage score approximations, i.e., the expected convergence rate will only be affected by a small distortion factor even when the approximation has low accuracy, such as $\gamma = 0.5$.

Remark. For constrained ℓ_2 regression, the bound is on the solution vector measured in prediction norm. By the triangular inequality, this directly implies $(\mathbb{E}\left[f(x_T)\right] - f(x^*))/f(x^*) \le \sqrt{c_3\epsilon}$.

Remark. With decaying step-sizes, it is possible to improve the dependence on ϵ from $\log(1/\epsilon)/\epsilon$ to $1/\epsilon$ for ℓ_2 regression.

3.2 The choice of the preconditioner F.

As we can see, the preconditioner F plays an important role in our algorithm. It converts the original regression problem in (1.1) to the stochastic optimization problem in (3.5). From Proposition 3.1 and Proposition 3.2, clearly, different choices of F will lead to different convergence rates in the SGD phase (reflected in $\kappa(RF)$[1]) and additional computational costs (reflected in H in (3.8)).

When $F = R^{-1}$, the effect of $\kappa_2(RF)$ on T vanishes. In this case, H is also a good approximation to the Hessian $A^\top A$. This is because usually R is the R-factor in the QR decomposition of SA, where SA is a "sketch" of A satisfying (A.1) that shares similar properties with A. Together we have $H = R^\top R = (SA)^\top(SA) \approx A^\top A$. This implies (3.7) is close to the Newton-type update. However, as a tradeoff, since H^{-1} is a $d \times d$ dense matrix, an additional $\mathcal{O}(d^2)$ cost per iteration is required to perform SGD update (3.8).

On the other hand, when $F = I$, no matrix-vector multiplication is needed in updating x. However, based on the discussion above, one should expect $\kappa(R) = \kappa(SA)$ to be close to $\kappa(A)$. Then the term $\kappa(RF) = \kappa(R)$ can be large if A is poorly conditioned, which might lead to undesirable performance in SGD phase.

Besides the obvious choices of F such as R^{-1} and I, one can also choose F to be a diagonal preconditioner D that scales R to have unit column norms. According to [34], the condition number after preconditioning $\kappa(RD)$ is always upper bounded by the original condition number $\kappa(R)$, while the additional cost per iteration to perform SGD updates with diagonal preconditioner is only $\mathcal{O}(d)$. In Section 4 we will illustrate the tradeoffs among these three choices of preconditioners empirically.

3.3 Complexities.

Here, we discuss the complexity of pwSGD with $F = R^{-1}$. The running time of Algorithm 1 consists of three parts. First, for computing a matrix R such that $U = AR^{-1}$ is well-conditioned, Appendix A provides a brief overview of various recently proposed preconditioning methods for computing R for both ℓ_1 and ℓ_2 norms; see also Table 4 and Table 5 for their running time $time(R)$ and preconditioning quality $\bar{\kappa}_p(U)$. Particularly, there are several available sparse preconditioning methods that run in $\mathcal{O}(\mathrm{nnz}(A))$ plus lower order terms in d time [9, 19, 23, 38, 35] . Second, to estimate the leverage scores, i.e., the row norms of AR^{-1}, [11, 8] proposed several algorithms for approximating the ℓ_1 and ℓ_2 leverage scores without forming matrix U. For a target constant approximation quality, e.g., $\gamma = 0.5$ and $c_1 = \frac{1+\gamma}{1-\gamma} = 3$, the running time of these algorithms is $\mathcal{O}(\log n \cdot \mathrm{nnz}(A))$. Third, Proposition 3.1 and Proposition 3.2 provide upper bounds for the expected algorithmic complexity of our proposed SGD algorithm when a target accuracy is fixed. Combining these, we have the following results.

[1]It is also reflected in $\hat{\kappa}(RF)$; however, it depends on $\kappa(RF)$ because one can show $m_1\kappa(RF) \leq \hat{\kappa}(RF) \leq m_2\kappa(RF)$, where m_1, m_2 are constants derived using matrix norm equivalences.

PROPOSITION 3.3. *Suppose the preconditioner in step 3 of Algorithm 1, is chosen from Table 4 or Table 5, with constant probability, one of the following events holds for* pwSGD *with $F = R^{-1}$. To return a solution $\tilde{x}$ with relative error ϵ on the objective,*

- *It runs in $time(R) + \mathcal{O}(\log n \cdot \mathrm{nnz}(A) + d^3\bar{\kappa}_1(U)/\epsilon^2)$ for unconstrained ℓ_1 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \mathrm{nnz}(A) + time_{update} \cdot d\bar{\kappa}_1(U)/\epsilon^2)$ for constrained ℓ_1 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \mathrm{nnz}(A) + d^3 \log(1/\epsilon)/\epsilon)$ for unconstrained ℓ_2 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \mathrm{nnz}(A) + time_{update} \cdot d\log(1/\epsilon)/\epsilon^2)$ for constrained ℓ_2 regression.*

In the above, $time(R)$ denotes the time for computing the matrix R and $time_{update}$ denotes the time for solving the optimization problem in (3.8).

Notice that, since $time_{update}$ only depends on d, an immediate conclusion is that by using sparse preconditioning methods, to find an ϵ-approximate solution, pwSGD runs in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)/\epsilon^2)$ time for ℓ_1 regression and in $\mathcal{O}(\log n \cdot \mathrm{nnz}(A) + \mathrm{poly}(d)\log(1/\epsilon)/\epsilon)$ time for ℓ_2 regression (in terms of solution vector in prediction norm for constrained problems or objective value for unconstrained problems).

Also, as can be seen in Proposition 3.3, for the complexity for ℓ_1 regression, the tradeoffs between $time(R)$ and $\bar{\kappa}_1(U)$ in choosing preconditioners from Table 4 are reflected here. On the other hand, for ℓ_2 regression, as all the preconditioning methods in Table 4 provide similar preconditioning quality, i.e., $\kappa(U) = \mathcal{O}(1)$, $time(R)$ becomes the key factor for choosing a preconditioning method. In Table 3, we summarize the complexity of pwSGD using various sketching methods for solving unconstrained ℓ_1 and ℓ_2 regression problems. The results are obtained by a direct combination of Table 2 and Tables 4 and 5.

Finally, we remind readers that Tables 1 and 2 summarize the complexities of several related algorithms for unconstrained ℓ_1 and ℓ_2 regression. As we can see, pwSGD is more suitable for finding a medium-precision, e.g., $\epsilon = 10^{-3}$, solution. In particular, it has a dependency uniformly better than RLA methods for ℓ_1 regression. Moreover, unlike the high-precision solvers, pwSGD also works for constrained problems, in which case one only needs to solve an optimization problem with quadratic objective over the same constraints at each iteration of pwSGD.

type	sketch	complexity
ℓ_1	Sparse	$\mathcal{O}(\mathrm{nnz}(A)\log n + d^{\frac{13}{2}} \log^{\frac{5}{2}} d/\epsilon^2)$
ℓ_1	FT	$\mathcal{O}(nd\log n + d^{\frac{17}{2}} \log^{\frac{9}{2}} d/\epsilon^2)$
ℓ_1	DT	$\mathcal{O}(nd^2\log n + d^{\frac{11}{2}} \log^{\frac{3}{2}} d/\epsilon^2)$
ℓ_2	Sparse	$\mathcal{O}\left(\mathrm{nnz}(A)\log n + d^4 + d^3\log(1/\epsilon)/\epsilon\right)$
ℓ_2	SRHT	$\mathcal{O}\left(nd\log n + d^3\log d + d^3\log(1/\epsilon)/\epsilon\right)$
ℓ_2	Gaussian	$\mathcal{O}\left(nd^2 + d^3\log(1/\epsilon)/\epsilon\right)$

Table 3: Summary of complexity of PWSGD with different sketching methods for computing the preconditioner when solving unconstrained ℓ_1 and ℓ_2 regression problems. The target is to return a solution $\tilde{x}$ with relative error ϵ on the objective. Here, the complexity of each algorithm is calculated by setting the failure probability to be a constant.

3.4 Complexity comparison between PWSGD and RLA.

As we pointed out in Section 1.2, PWSGD and RLA methods with algorithmic leveraging (Appendix B) (RLA for short) are closely related, as they can be viewed as methods using SA and SAA to solve the stochastic optimization problem (1.2). Omitting the time for computing the basis U and sampling distribution P, the comparison of complexity boils down to comparing $time_{sub}(s,d)$ (for RLA) and $time_{update} \cdot T$ (for PWSGD), where $time_{sub}(s,d)$ is the time needed to solve the same constrained regression problem with size s by d and $time_{update}$ denotes the time needed for to solve the optimization problem in (3.8). According to the theory, for the same target accuracy, the required s (sampling size) and T (number of iterations) are equal asymptotically, up to logarithmic factors; see [10, 37, 12] and Section B for expression of s. When the problem is unconstrained, due to the efficiency of SGD, $time_{update} = \mathcal{O}(d^2)$ as indicated in (3.8). For ℓ_2 regression, due to the efficiency of the direct solver, $time_{sub}(s,d) = \mathcal{O}(sd^2)$. This explains why PWSGD and RLA (low-precision solvers (sampling)) have similar complexities as shown in Table 2. On the other hand, for unconstrained ℓ_1 regression, a typical ℓ_1 regression solver requires time $time_{sub}(s,d) > sd^2$. For example, if an interior point method is used [26], $time_{sub}(s,d)$ is not even linear in s. This explains the advantage of PWSGD over RLA as shown in Table 1. We also note that in the presence of constraints, PWSGD may still be more efficient for solving ℓ_1 regression because, roughly speaking, $time_{sub}(s,d)/s > time_{update}$.

3.5 Connection to weighted randomized Kaczmarz algorithm.

As mentioned in Section 1, our algorithm PWSGD for least-squares regression is related to the weighted randomized Kaczmarz (RK) algorithm [32, 22]. To be more specific, weighted RK algorithm can be viewed as an SGD algorithm with constant step-size that exploits a sampling distribution based on row norms of A, i.e., $p_i = \|A_i\|_2^2/\|A\|_F^2$. In PWSGD, if the preconditioner $F = R^{-1}$ is used and the leverage scores are computed exactly, the resulting algorithm is equivalent to applying the weighted randomized Kaczmarz algorithm on a well-conditioned basis $U = AR^{-1}$ since leverage scores are defined as the row norms of U.

Since the matrix A itself can be a basis for its range space, setting $U = A$ and $F = R = I$ in Proposition 3.2 indicates that weighted RK algorithm inherits a convergence rate that depends on condition number $\kappa(A)$ times the scaled condition number $\bar{\kappa}_2(A)$. Notice that in PWSGD, the preconditioning step implicitly computes a basis U such that both $\kappa(U)$ and $\bar{\kappa}(U)$ are low. One should expect the SGD phase in PWSGD inherits a faster convergence rate, as verified numerically in Section 4.

With proof techniques as for Proposition 3.1, one can also show that for ℓ_1 regression, when using SGD with weighted sampling distribution proportional to the row norms, the convergence rate depends on ℓ_1 condition number (Definition 1) of the linear system. Thus preconditioning becomes essential to accelerate the convergence.

4 Experiments

In this section, we provide empirical evaluations of our main algorithm PWSGD on a real dataset `Year`[2] with size 5×10^5 by 90. (Again, more details can be found in the technical report version of this paper [36].) We present the time-accuracy tradeoffs among various methods in solving unconstrained ℓ_1 and ℓ_2 regression problems. For PWSGD, we implement it with three different choices of the preconditioner F. Herein, throughout the experiments, by PWSGD-full, PWSGD-diag, PWSGD-noco, we respectively mean PWSGD with preconditioner $F = R^{-1}, D, I$; see Section 3.2 for details. Note that, for PWSGD, we use the methods from [9] for preconditioning. Also, we exactly compute the row norms of AR^{-1} and use them as the leverage scores. As comparisons, we also implemented standard SGD, weighted randomized Kaczmarz (RK) [32], Adagrad [13] and RLA methods with algorithmic leveraging (RLA for short) described in Appendix B for comparisons. For AdaGrad, we use diagonal scaling and mirror descent update rule.

As for implementation details, in all SGD-like algorithms, step-size tuning is done by grid-searching where at each trial the algorithm is run with a candidate step-size for $3n$ iterations. Then the step-size that yields the lowest error is used. The initial solution vector estimate is set as zero. The algorithms are then run in the following manner. Each epoch contains 1000 iterations. At the beginning of each epoch, we sample 1000 indices according to

[2]`https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD`

the underlying distribution without replacement and update the weight using the 1000 row samples from the data matrix. The time/accuracy pair at every 1000 iterations is recorded. For RLA, we choose s from a wide range of values and record the corresponding time/accuracy pairs. Finally, the plots are generated by averaging the results over 20 independent trials. The results are presented in Figure 2.

We note that in [36] we also include a set of empirical results on the behavior of the convergence rate of PWSGD and several related SGD-like algorithms on various synthetic datasets, in which we demonstrate the superior convergence rate of PWSGD and its stable performance on problems with increasing condition number.

In our algorithm PWSGD, a faster convergence comes with the additional cost of preconditioning. For example, the preconditioning phase of PWSGD takes approximately 3 seconds. Nevertheless, with a faster convergence rate in a well-conditioned basis, PWSGD-full still outperforms other methods in converging to a higher-precision solution at a given time span. As PWSGD-diag balances convergence rate and computational cost, it outperforms PWSGD-full at the early stage and yields comparable results to AdaGrad. As expected, due to the poor conditioning, SGD, weighted-RK and PWSGD suffer from slow convergence rates. As for RLA methods, they have the same first step as in PWSGD, i.e., preconditioning and constructing the sampling distribution. For ℓ_1 regression, to obtain a fairly high-precision solution, the sampling size has to be fairly large, which might drastically increase the computation time for solving the sampled subproblem. This explains the advantage of PWSGD-full over RLA methods in Figure 2. It is worth mentioning that, although for ℓ_2 regression our theory provides relative error bound on the solution vector measured in the prediction norm, here we also see that PWSGD-full and PWSGD-diag display promising performance in approximating the solution vector measure in ℓ_2 norm. Also, although there are no theoretical results to support the solution vector convergence on ℓ_1 regression problems with PWSGD, the sequence generated by PWSGD converges to the optimal point favorably.

Finally, notice that RLA uses a high performance direct solver to solve the mid-size subsampled problem for ℓ_2 regression. In this case PWSGD methods do not show significant advantages over RLA in terms of speed. For this reason we have not included RLA results in Figures 2(a) and (b). Nevertheless, PWSGD methods may still be favorable over straightforward RLA in terms of speed and feasibility when the size of the dataset becomes increasingly larger, e.g., 10^7 by 500 (or even much larger [38]).

References

[1] H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM J. on Scientific Computing*, 32(3):1217–1236, 2010.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, 1994.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Computational Statistics (COMPSTAT)*, 2010.

[4] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems (NIPS)*, 2008.

[5] L. Bottou and Y. Le Cun. Large scale online learning. In *Neural Information Processing Systems (NIPS)*, 2004.

[6] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[7] K. L. Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Symposium on Discrete Algorithms (SODA)*, 2005.

[8] K. L. Clarkson, P. Drineas, M. Magdon-Ismail, M. W. Mahoney, X. Meng, and D. P. Woodruff. The Fast Cauchy Transform and faster robust linear regression. In *Symposium on Discrete Algorithms (SODA)*, 2013.

[9] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing (STOC)*, 2013.

[10] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM J. on Computing*, 38(5):2060–2078, 2009.

[11] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Machine Learning Research*, 13:3441–3472, 2012.

[12] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

[13] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Machine Learning Research*, 12:2121–2159, 2011.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.

[15] C. Hu, J. T. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In *Neural Information Processing Systems (NIPS)*, 2009.

[16] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Neural Information Processing Systems (NIPS)*, 2013.

[17] C. T. Kelley. *Iterative Methods for Solving Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[18] M. W. Mahoney. *Randomized algorithms for matrices and data*. Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011.

[19] X. Meng and M. W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on the Theory of Computing (STOC)*, 2013.

[20] X. Meng and M. W. Mahoney. Robust regression on MapReduce. In *International Conference on Machine Learning*

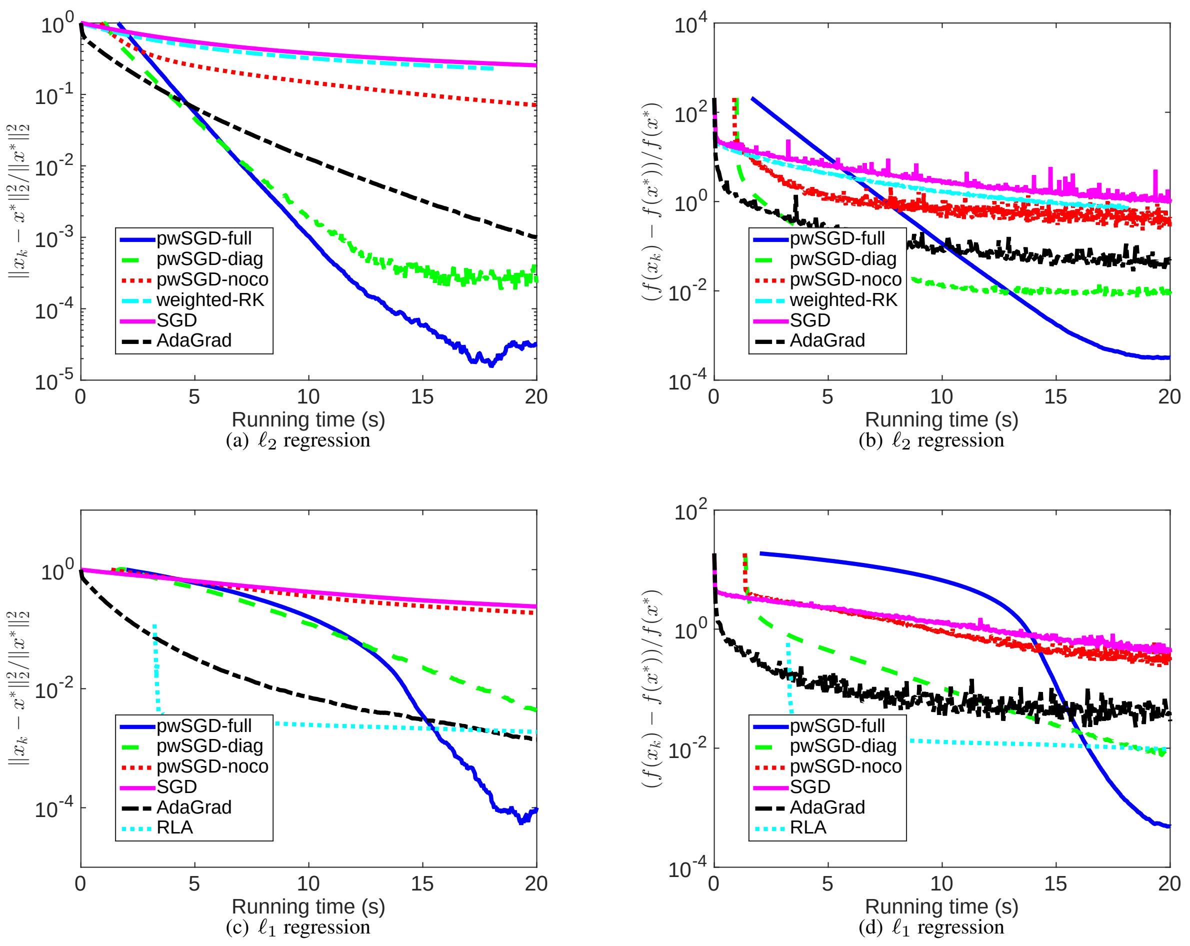


Figure 2: Time-accuracy tradeoffs of several algorithms including PWSGD with three different choices of preconditioners on `year` dataset. Both ℓ_1 and ℓ_2 regressions are tested and the relative error on both the objective value, i.e., $|f(\hat{x}) - f(x^*)|/f(x^*)$, and the solution vector, i.e., $\|\hat{x} - x^*\|_2^2/\|x^*\|_2^2$, are measured.

(ICML), 2013.

[21] X. Meng, M. A. Saunders, and M. W. Mahoney. LSRN: A parallel iterative solver for strongly over- or under-determined systems. *SIAM J. on Scientific Computing*, 36(2):C95–C118, 2014.

[22] D. Needell, R. Ward, and N. Srebro. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Neural Information Processing Systems (NIPS)*, 2014.

[23] J. Nelson and N. L. Huy. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[24] F. Niu, B. Recht, C. Ré, and J. S Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Neural Information Processing Systems (NIPS)*, 2011.

[25] S. Portnoy. On computation of regression quantiles: Making the Laplacian tortoise faster. *Lecture Notes-Monograph Series, Vol. 31, L_1-Statistical Procedures and Related Topics*, pages 187–200, 1997.

[26] S. Portnoy and R. Koenker. The Gaussian hare and the Laplacian tortoise: Computability of squared-error versus absolute-error estimators, with discussion. *Statistical Science*, 12(4):279–300, 1997.

[27] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub–gradient solver for SVM. In *International Conference on Machine Learning (ICML)*, 2007.

[29] S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *International Conference on Machine Learning (ICML)*, 2008.

[30] S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1 regularized loss minimization. In *International Conference on Machine Learning (ICML)*, 2009.

[31] C. Sohler and D. P. Woodruff. Subspace embedding for the ℓ_1-norm with applications. In *Symposium on Theory of Computing (STOC)*, 2011.

[32] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15(2), 2009.

[33] J. A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Adv. Adapt. Data Anal.*, 3(1-2):115–126, 2011.

[34] A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14(1):14–23, 1969.

[35] D. P. Woodruff and Q. Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. *Conference on Learning Theory (COLT)*, 2013.

[36] J. Yang, Y. Chow, C. Ré, and M. W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. *CoRR*, abs/1502.03571, 2015.

[37] J. Yang, X. Meng, and M. W. Mahoney. Quantile regression for large-scale applications. *SIAM J. Scientific Computing*, 36(5):S78–S110, 2014.

[38] J. Yang, X. Meng, and M. W. Mahoney. Implementing randomized matrix algorithms in parallel and distributed environments. *CoRR*, abs/1502.03032, 2015. Accepted for publication in *Proceedings of the IEEE*.

[39] P. Zhao and T. Zhang. Stochastic optimization with importance sampling. In *International Conference on Machine Learning (ICML)*, 2015.

A Supplementary Details of Algorithm 1

As we discussed, we need to compute a well-conditioned basis implicitly and estimate its row norms, i.e., AR^{-1} and $\{\lambda_i\}_{i=1}^n$ in step 3 in Algorithm 1.

A summary of various preconditioning methods can be found in [37, 38]. Here, we describe the QR-type method. A high-level summary for these methods is given as follows:

- Given a matrix $A \in \mathbb{R}^{n\times d}$ with full rank, we first construct a sketch $\Pi A \in \mathbb{R}^{\text{poly}(d)\times d}$ for A satisfying
(A.1)
$$\sigma_S \cdot \|Ax\|_p \le \|SAx\|_p \le \kappa_S \sigma_S \cdot \|Ax\|_p, \quad \forall x \in \mathbb{R}^d,$$
where κ_S is the distortion factor independent of n.

- Next, we compute the QR factorization of SA whose size only depends on d. The resulting R satisfies that AR^{-1} is a well-conditioned basis with condition number $\bar\kappa(AR^{-1})$ depending on d, κ_S, and thus on d solely.

Various ways of computing a sketching matrix S satisfying (A.1) have been proposed recently. It is worth mentioning that sketching algorithms that run in nearly input-sparsity time, i.e., in time proportional to $\mathcal{O}(\text{nnz}(A))$ plus lower order terms in d, to obtain such a sketch matrix for $p = 1$ and $p = 2$ are available via random projections composed of sparse matrices; see [9, 19, 23] for details. See Table 4 for a short summary of preconditioning methods using various sketching matrices. Note that the running time here denotes the total running for computing the matrix R. Again, below $\bar\kappa_p(U)$ is the condition number of $U = AR^{-1}$ as defined in Definition 1 and $\kappa(U)$ is the standard condition number of U.

Next, given the implicit representation of U by R, to compute the leverage scores $\|U_i\|_p^p$ exactly, one has to compute U which takes $\mathcal{O}(nd^2)$ time. Instead of forming U explicitly and "reading off" the row norms for computing the leverage scores, one can estimate the row norms of U up to a small factor by post-multiplying a random projection matrix; see [8, 11] for the cases when $p = 1, 2$ respectively. The above process can be done in $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time.

Finally, we present two additional results regarding the non-asymptotic convergence rate of pwSGD on ℓ_1 and ℓ_2 regression, respectively. Notation is similar to the one used in Proposition 3.1 and Proposition 3.2.

PROPOSITION A.1. *For $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $f(x) = \|Ax - b\|_1$. Algorithm 1 with $p = 1$ returns a solution vector estimate $\bar x$ that satisfies the following expected error bound*

(A.2)
$$\mathbb{E}\left[f(\bar x)\right] - f(x^*) \le \frac{1}{2\eta T}\|x^* - x_1\|_H^2 + \frac{\eta}{2}\left(c_1\alpha\|RF\|_1\right)^2.$$

Hereby, the expectation is taken over all the samples $\xi_1, \dots, \xi_T$ and x^* is an optimal solution to the problem $\min_{x\in\mathcal{Z}} f(x)$. The constant in the error bound is given by $c_1 = \frac{1+\gamma}{1-\gamma}$.

PROPOSITION A.2. *For $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $f(x) = \|Ax - b\|_2^2$. Algorithm 1 with $p = 2$ returns a solution vector estimate x_T that satisfies the following expected error bound*

$$\begin{aligned}
&\mathbb{E}\left[\|x_t - x^*\|_H^2\right] \\
&\quad\le \left(\frac{1 - 4\eta\left(1 - 2\eta c_1\alpha^2\|RF\|_2^2\right)}{\beta^2\|(RF)^{-1}\|_2^2}\right)^T \|x_0 - x^*\|_H^2 \\
&\quad+ \frac{2c_1\eta\bar\kappa_2^2(U)\kappa^2(RF)h(y^*)}{1 - 2c_1\eta\alpha^2\|RF\|_2^2}.
\end{aligned} \tag{A.3}$$

Hereby, $H = (F^{-1})^\top F^{-1}$ is the weights of the ellipsoidal norm and the expectation is taken over all the samples $\xi_1, \dots, \xi_T$ and x^* is an optimal solutions to the problem $\min_{x\in\mathcal{Z}} f(x)$. The constant in the error bound is given by $c_1 = \frac{1+\gamma}{1-\gamma}$.

B RLA Methods with Algorithmic Leveraging

In this section, we present the RLA sampling algorithm with algorithmic leveraging for solving ℓ_p regression problems mentioned in Section 1.2. The main idea in this class of algorithms is to sample rows based on the leverage scores of the input matrix A and solve the sample average

name	running time	$\bar{\kappa}_p(U)$
Dense Cauchy [31]	$\mathcal{O}(nd^2 \log d)$	$\mathcal{O}(d^{5/2} \log^{3/2} d)$
Fast Cauchy [8]	$\mathcal{O}(nd \log d + d^3 \log d)$	$\mathcal{O}(d^{11/2} \log^{9/2} d)$
Sparse Cauchy [19]	$\mathcal{O}(\text{nnz}(A) + d^7 \log^5 d)$	$\mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$

Table 4: Summary of running time and condition number, for several different ℓ_1 conditioning methods. The failure probability of each method is set to be a constant.

name	running time	$\kappa_p(U)$	$\bar{\kappa}_p(U)$
Gaussian	$\mathcal{O}(nd^2)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
SRHT [33]	$\mathcal{O}(nd \log n + d^3 \log d)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
Sparse ℓ_2 Embedding [9]	$\mathcal{O}(\text{nnz}(A) + d^4)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$

Table 5: Summary of running time and condition number, for several different ℓ_2 conditioning methods. The failure probability of each method is set to be a constant.

approximation (SAA) of the ℓ_p regression problem. This method is formally stated in Algorithm 2.

The following theorem (from [10]) states that if the sampling size s is large enough, the resulting approximation solution $\hat{x}$ produces a $\left(\frac{1+\epsilon}{1-\epsilon}\right)$-approximation to the original solution vector. The following theorem also shows that in a *fixed budget setup*, i.e., when the desired accuracy and confidence interval are fixed, the required sampling size only depends on the lower dimension d since α and β are independent of n.

THEOREM B.1. *Given input matrix $A \in \mathbb{R}^{n\times d}$ and vector $b \in \mathbb{R}^n$, let α, β be the condition numbers of the well-conditioned basis U and γ be the quality of approximation to the leverage scores satisfying* (3.3). *Then when $\epsilon < 1/2$ and the sampling size satisfies the following condition*

(B.4)
$$s \geq \frac{1+\gamma}{1-\gamma} \frac{(32\alpha\beta)^p}{p^2\epsilon^2} \left((d+1) \log\left(\frac{12}{\epsilon}\right) + \log\left(\frac{2}{\delta}\right) \right),$$

Algorithm 2 returns a solution vector $\hat{x}$ that satisfies the following inequality with probability at least $1-\delta$,

(B.5)
$$\|A\hat{x} - b\|_p \leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \|Ax^* - b\|_p,$$

where $x^* \in \mathcal{Z}$ is an optimal solution to the original problem $\min_{x\in\mathcal{Z}} \|Ax - b\|_p$.

Remark. As can be seen, the sampling size is $s = \mathcal{O}(\text{poly}(d)\log(1/\epsilon)/\epsilon^2$ for a target accuracy ϵ. When solving unconstrained ℓ_2 regression, however, it can be shown that a sampling size $s = \mathcal{O}(\text{poly}(d)\log(1/\epsilon)/\epsilon)$ is sufficient to compute an ϵ-approximate solution; see [9] for details.

Algorithm 2 RLA methods with algorithmic leveraging for constrained ℓ_p regression

```
1: Input: $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$ with rank$(\bar{A}) = k$, $\mathcal{Z}$ and $s > 0$.
2: Output: An approximate solution $\hat{x} \in \mathbb{R}^d$ to problem minimize$_{x\in\mathcal{Z}} \|Ax - b\|_p^p$.
3: Compute $R \in \mathbb{R}^{k\times(d+1)}$ such that $\bar{A} = UR$ and $U$ is an $(\alpha, \beta)$ well-conditioned basis $U$ for the range space of $\bar{A}$.
4: Compute or estimate $\|U_i\|_p^p$ by $\lambda_i$ satisfying (3.3) with $\gamma$, for $i \in [n]$.
5: Let $p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$, for $i \in [n]$.
6: Let $S \in \mathbb{R}^{s\times n}$ be a zero matrix.
7: for $i = 1, \ldots, s$ do
8:     Pick $\xi_t$ from $[n]$ based on distribution $\{p_i\}_{i=1}^n$.
9:     Set $S_{i,\xi_t} = \left(\frac{1}{p_{\xi_t}}\right)^{\frac{1}{p}}$.
10: end for
11: Return $\hat{x} = \arg\min_{x\in\mathcal{Z}} \|SAx - Sb\|_p$.
```