Weighted SGD for ℓ_p Regression with Randomized Preconditioning [*]

Jiyan Yang JIYAN@STANFORD.EDU
Institute for Computational and Mathematical Engineering
Stanford University
Stanford, CA 94305, USA

Yin-Lam Chow YCHOW@STANFORD.EDU
Institute for Computational and Mathematical Engineering
Stanford University
Stanford, CA 94305, USA

Christopher Ré CHRISMRE@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Michael W. Mahoney MMAHONEY@STAT.BERKELEY.EDU
International Computer Science Institute and Department of Statistics
University of California, Berkeley
Berkeley, CA 94720, USA

Editor: Zhihua Zhang

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 performance guarantees but are applicable to a narrower class of problems. We aim to bridge the gap between these two 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 a deterministic ℓ_p regression problem as 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 rates are superior to other related SGD algorithm 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 in 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 in the objective value and the solution vector measured in prediction norm in $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d) \log(1/\epsilon)/\epsilon)$ time. We show that for 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 $n \geq d^2/\epsilon$.

We also provide lower bounds on the coreset complexity for more general regression problems, indicating that still new ideas will be needed to extend similar RLA preconditioning ideas to weighted SGD algorithms for more general regression problems. 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.

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 (Bottou, 2010; Mahoney, 2011). Here, we consider the ubiquitous ℓ_1 and ℓ_2 regression problems, and we describe a novel RLA-SGD algorithm called PWSGD (preconditioned weighted SGD). 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

$$\min_{x \in \mathcal{Z}} f(x) = \|Ax - b\|_p, \tag{1}$$

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 $\mathcal{O}(nd^2)$ (Golub and Van Loan, 1996); or by iterative methods for which the running time depends on the condition number of A (Barrett et al., 1994; Kelley, 1995; Saad, 2003), while the unconstrained ℓ_1 regression problem can be formulated as a linear program (Portnoy and Koenker, 1997; Chen et al., 2001) and solved by an interior-point method (Portnoy and Koenker, 1997; Portnoy, 1997).

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 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 Yang et al. (2016b) 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.

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. In particular, we prove that after preconditioning, the number of iterations required to converge to a target accuracy is fully predictable and only depends on the low dimension d, i.e., it is independent of the high dimension n. 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; and 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, i.e., $(f(\hat{x}) - f(x^*))/f(x^*) = \epsilon$, of several solvers for unconstrained ℓ_1 and ℓ_2 regression. In Table 1, RLA with algorithmic leveraging (RLA for short) (Clarkson et al., 2013; Yang et al., 2014) is a popular method for obtaining a low-precision solution and randomized IPCPM is an iterative method for finding a higher-precision solution (Meng and Mahoney, 2013b) 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 Drineas et al. (2011); Clarkson and Woodruff (2013) for projection-based examples and Clarkson and Woodruff (2013); Drineas et al. (2012) for sampling-based examples. "High-precision" solvers refer to "sketching + preconditioning + iterative solver" type algorithms; see Avron et al. (2010); Meng et al. (2014) 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.

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 update,* PWSGD *has a strong advantage over traditional RLA methods.* See Sections 4.3 and 4.4 for more detailed discussions.

Finally, in Section 5, empirically we show that PWSGD performs favorably compared to other competing methods, as it converges to a medium-precision solution more quickly.

solver	complexity (general)	complexity (sparse)
RLA with algorithmic leveraging	$time(R) + \mathcal{O}(\mathrm{nnz}(A)\log n + \bar{\kappa}_1^{\frac{5}{4}} d^{\frac{17}{4}} / \epsilon^{\frac{5}{2}})$	$\mathcal{O}(\mathrm{nnz}(A)\log n + d^{\frac{69}{8}} \log^{\frac{25}{8}} d / \epsilon^{\frac{5}{2}})$
randomized IPCPM	$time(R) + nd^2 + \mathcal{O}((nd + \mathrm{poly}(d))\log(\bar{\kappa}_1 d/\epsilon))$	$nd^2 + \mathcal{O}((nd + \mathrm{poly}(d))\log(d/\epsilon))$
PWSGD	$time(R) + \mathcal{O}(\mathrm{nnz}(A)\log n + d^3 \bar{\kappa}_1 / \epsilon^2)$	$\mathcal{O}(\mathrm{nnz}(A)\log n + d^{\frac{13}{2}} \log^{\frac{5}{2}} d / \epsilon^2)$

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 $(f(\hat{x}) - f(x^*))/f(x^*) \le \epsilon$, 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 $\bar{\kappa}_1$ (Definition 1). The general complexity bound and the one using sparse reciprocal exponential transform (Woodruff and Zhang, 2013) as the underlying sketching method are presented. Here, we assume $n \gg d$ such that $n > d^3 \log d$ 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 (Portnoy and Koenker, 1997). 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)	$\mathcal{O}\left(nd\log(d/\epsilon) + d^3 \log n(\log d + 1/\epsilon)\right)$	$\mathcal{O}\left(\mathrm{nnz}(A) + d^4/\epsilon^2\right)$
low-precision solvers (sampling)	$\mathcal{O}\left(nd\log n + d^3 \log n \log d + d^3 \log d/\epsilon\right)$	$\mathcal{O}\left(\mathrm{nnz}(A)\log n + d^4 + d^3 \log d/\epsilon\right)$
high-precision solvers	$\mathcal{O}\left(nd\log n + d^3 \log n \log d + nd\log(1/\epsilon)\right)$	$\mathcal{O}\left(\mathrm{nnz}(A) + d^4 + nd\log(1/\epsilon)\right)$
PWSGD	$\mathcal{O}\left(nd\log n + d^3 \log n \log d + d^3 \log(1/\epsilon)/\epsilon\right)$	$\mathcal{O}\left(\mathrm{nnz}(A)\log n + d^4 + d^3 \log(1/\epsilon)/\epsilon\right)$

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 $(f(\hat{x}) - f(x^*))/f(x^*) \le \epsilon$, where $f(x) = \|Ax - b\|_2$. Two sketching methods, namely, SRHT (Drineas et al., 2011; Tropp, 2011) and CW (Clarkson and Woodruff, 2013) are considered. Here, we assume $d \le n \le e^d$. The complexity of each algorithm is computed by setting the failure probability to be a constant.

1.2 Connection to related algorithms

As a side 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 (in Proposition 4 formally) reformulate the deterministic ℓ_p regression problem in (1) as a stochastic optimization problem, i.e.,

$$\min_{y \in \mathcal{Y}} \|Uy - b\|_p^p = \min_{y \in \mathcal{Y}} \mathbb{E}_{\xi \sim P} \left[|U_\xi y - b_\xi|^p / p_\xi \right],$$

where U is a basis for the range space of A and ξ is a random variable over $\{1, \ldots, n\}$ with distribution $P = \{p_i\}_{i=1}^n$. As suggested in Figure 1, to solve this stochastic optimization problem, typically one needs to answer the following three questions.

- *(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?*
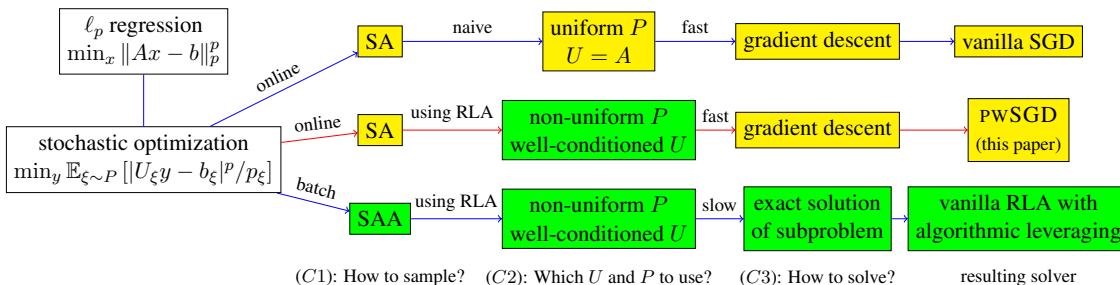
Figure 1: An overview of our framework for solving ℓ_p regression via stochastic optimization. To construct a solver, three choices have to be made. For ($C1$), the answer can be either SAA (Sampling Average Approximation, i.e., sample a batch of points and deal with the subproblem) or SA (Stochastic 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).

- *($C3$): 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 and Section 3 for more details. A natural question arises: is there a combination of these choices that leverages the algorithmic benefits of RLA preconditioning to improve the performance of SGD-type algorithms? Recall that RLA methods (in particular, those that exploit algorithmic averaging; see Appendix B and also Drineas et al. (2012); Yang et al. (2016b)) 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 $p = 2$. One immediate idea is to develop an SGD-like algorithm that uses the same choice of U and P as in RLA methods. 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 (Strohmer and Vershynin, 2009; Needell et al., 2014) in the way that both algorithms are SGD algorithm with non-uniform P but PWSGD runs on a well-conditioned basis U while randomized RK doesn't involve preconditioning. In Section 4.5 we show that this preconditioning step dramatically reduces the number of iterations 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) into a stochastic optimization problem (4) and make connections to existing solvers including RLA methods with algorithmic leveraging and weighted randomized Kaczmarz algorithm (Sections 3 and 4.5).

- 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, convergence rate of the SGD phase only depends on the low dimension d, and is independent of the high dimension n (Sections 4.1 and 4.2).

- We prove that PWSGD returns an approximate solution 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 4.3 and 4.4).

- We prove that PWSGD returns a 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 4.3 and 4.4).

- 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 5).

- We show connections between RLA algorithms and coreset methods of empirical optimization problems under the framework of Feldman and Langberg (2011). We show that they are equivalent for ℓ_p regression and provide lower bounds on the coreset complexity for some more general regression problems. We also discuss the difficulties in extending similarly RLA preconditioning ideas to general SGD algorithms (Section 6).

1.4 Other prior related work

Numerous RLA algorithms have been proposed to solve ℓ_p regression problems (Yang et al., 2016b). RLA theories show that to achieve a relative-error bound, the required sampling 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 Dasgupta et al. (2009) 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; a sampling algorithm based on Lewis weights for ℓ_p regression have been proposed by Cohen and Peng (2015); the algorithms in Sohler and Woodruff (2011) and Clarkson et al. (2013) use the "slow" and "fast" Cauchy Transform to compute the low-distortion ℓ_1 embedding matrix and solve the over-constrained ℓ_1 regression problem in $\mathcal{O}(nd^{1.376+})$ and $\mathcal{O}(nd \log n)$ time, respectively; the algorithms in Drineas et al. (2012) estimate the leverage scores up to a small factor and solve the ℓ_2 regression problem in $\mathcal{O}(nd \log n)$ time respectively; and the algorithms in Clarkson and Woodruff (2013); Meng and Mahoney (2013a); Nelson and Nguyen (2013), 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, and a tighter analysis is provided by Cohen (2016). As for iterative algorithms, the algorithms in Avron et al. (2010); Meng et al. (2014) use randomized linear algebra 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 (Bottou and Le Cun, 2004; Shalev-Shwartz and Srebro, 2008; Bottou and Bousquet, 2008). Moreover, these methods can be easily extended for problems with general convex loss functions and constraints, such as Pegasos (Shalev-Shwartz et al., 2007) for regularized SVM and stochastic coordinate descent (SCD) for ℓ_1 regularization (Shalev-Shwartz and Tewari, 2009). Several techniques, such as SAGE (Hu et al., 2009), AdaGrad (Duchi et al., 2011), SVRG (Johnson and Zhang, 2013), have recently been proposed to accelerate the convergence rate of SGD, and Niu et al. (2011) also show that SGD is favorable for parallel/distributed computation. More recently, several works, e.g., Zhao and Zhang (2015); Needell et al. (2014) regarding SGD with weighted sampling are proposed, in which the authors show that the performance of SGD can be improved by using a nonuniform sampling distribution.

In addition, as we point out in Section 4.2, PWSGD has a close relationship to second-order methods. It can be viewed as an algorithm with approximate Hessians obtained by sketching and stochastic gradients. This is related to the iterative Hessian sketching algorithm for solving constrained least squares problems proposed by Pilanci and Wainwright (2014) which is essentially a Newton-type algorithm with iterative sketched Hessians and batch gradients. Moreover, the idea of using approximate Hessians and stochastic gradients have been discussed in several recent papers. For example, (Moritz et al., 2016; Byrd et al., 2016; Curtis, 2016) exploit the idea of approximating Hessian with L-BFGS type updates and (variance-reduced) stochastic updates.

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 $\mathrm{poly}(d)$ a low-degree polynomial in d. We also use $[n]$ to denote the set of indices $1, \ldots, n$.

Throughout this subsection, 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 on well-conditioned bases and leverage scores are crucial to our methods. The first notion is originally introduced by Clarkson (2005) and stated more precisely in Dasgupta et al. (2009), and it is used to justify the well-posedness of a ℓ_p regression problem. These notions were introduced by Dasgupta et al. (2009).

Definition 1 ((α, β, p)-conditioning and well-conditioned basis) *A matrix $A \in \mathbb{R}^{n \times d}$ is (α, β, p)-conditioned if $|A|_p \leq \alpha$ and for all $x \in \mathbb{R}^d$, $\beta \|Ax\|_p \geq \|x\|_q$, 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 $\mathrm{range}(A)$ 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 $A \in \mathbb{R}^{n \times d}$, suppose U is an (α, β, p) well-conditioned basis for* range(A). *Then the i-th leverage score λ_i of A is defined as $\lambda_i = \|U_i\|_p^p$ for $i = 1, \ldots, n$.*

2.1 Preconditioning

Here, we briefly review the preconditioning methods that will be used in our main algorithms. A detailed summary of various preconditioning methods can be found in Yang et al. (2014, 2016b). The procedure for computing a preconditioner can be summarized in the following two steps.

- Given a matrix $A \in \mathbb{R}^{n \times d}$ with full rank, we first construct a sketch $SA \in \mathbb{R}^{s \times d}$ for A satisfying

$$\sigma_S \cdot \|Ax\|_p \leq \|SAx\|_p \leq \kappa_S \sigma_S \cdot \|Ax\|_p, \quad \forall x \in \mathbb{R}^d, \tag{2}$$

  where κ_S is the distortion factor independent of n.

- Next, we compute the QR factorization of SA whose size only depends on d. Return R^{-1}.

The following lemma guarantees that the preconditioner satisfies that AR^{-1} is well-conditioned since κ_S and s depend on d only, independent of n.

Lemma 3 *Let R be the matrix returned by the above preconditioning procedure, then we have*

$$\bar{\kappa}_p(AR^{-1}) \leq \kappa_S d^{\max\{\frac{1}{2}, \frac{1}{p}\}} s^{\left|\frac{1}{p} - \frac{1}{2}\right|}. \tag{3}$$

Various ways of computing a sketching matrix S satisfying (2) are proposed recently. It is worth mentioning that sketching algorithms that run in nearly input-sparsity time, i.e., in time proportional to $\mathcal{O}(\mathrm{nnz}(A))$ to obtain such a sketch matrix for $p = 1$ and $p = 2$ are available via random projections composed of sparse matrices; see Clarkson and Woodruff (2013); Meng and Mahoney (2013a); Woodruff and Zhang (2013); Nelson and Nguyen (2013) for details. In Tables 5 and 6 in Appendix A we provide a short summary of these sketching methods and the resulting running time and condition number.

3. A connection to stochastic optimization

In this section, we describe our framework for viewing deterministic ℓ_p regression problems from the perspective of stochastic optimization. This framework will recover both RLA and SGD methods in a natural manner; and by combining these two approaches in a particular way we will obtain our main algorithm.

We reformulate overdetermined ℓ_p regression problems of the form (1) into a stochastic optimization problem of the form (4) [1], which reformulates a deterministic regression problem into a stochastic optimization problem. Note that the result holds for general $p \in [1, \infty)$.

Proposition 4 *Let $U \in \mathbb{R}^{n \times d}$ be a basis of the range space of A in the form $U = AF$, where $F \in \mathbb{R}^{d \times d}$. The constrained overdetermined ℓ_p regression problem (1) is equivalent to*

$$\min_{y \in \mathcal{Y}} \|Uy - b\|_p^p = \min_{y \in \mathcal{Y}} \mathbb{E}_{\xi \sim P} \left[H(y, \xi) \right], \tag{4}$$

---

1. Technically, this result is straightforward; but this reformulation allows us to introduce randomness—parameterized by a probability distribution P—into the deterministic problem (1) in order to develop randomized algorithms for it.

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 $\mathcal{Y} = \{y \in \mathbb{R}^d | y = F^{-1}x, x \in \mathcal{Z}\}$.

With Proposition 4, as suggested in Figure 1, one can solve the overdetermined ℓ_p regression problem (1) by applying either SAA or SA, i.e., $(C1)$ on the stochastic optimization problem (4). In addition to the choice of SA versus SAA, one also has to choose U and P, i.e., $(C2)$, and determine the underlying solver, i.e., $(C3)$.

Assume that if SAA is used, then for $(C3)$ we solve the subproblem exactly, i.e., we compute a high-precision solution of the subproblem; this leads to a class of *randomized linear algebra* (RLA) algorithms for solving ℓ_p regression. Alternatively, if we assume that SA is used, then we extract the first-order information, i.e., sub-gradient of the sample, and update the weight in a gradient descent fashion; this leads to a family of *stochastic gradient descent* (SGD) algorithms for solving ℓ_p regression.

For $(C2)$, we need to choose a basis U that converts (1) into an equivalent problem represented by U and choose a distribution P for which the algorithm samples a row at every iteration accordingly. In general, different choices of U and P lead to different algorithms. In the following two subsections, we will discuss their effects on SAA and SA and make connections between existing solvers and our new solution methods. For simplicity, we assume there are no constraints, i.e., $\mathcal{Z} = \mathbb{R}^d$ (although much of this framework generalizes to nontrivial constraints).

3.1 Using RLA (SAA) to solve ℓ_p regression

In this subsection, we briefly discuss the algorithms induced by applying SAA to (4) with different choices of basis U and distribution P in Proposition 4.

We first show that the choice of the basis U has no effect on the resulting sampling algorithm. Let $S \in \mathbb{R}^{s \times n}$ be the equivalent sampling matrix in the sampling algorithm. That is,

$$S_{ij} = \begin{cases} 1/p_j & \text{if the } j\text{-th row is sampled in the } i\text{-th iteration} \\ 0 & \text{otherwise.} \end{cases}$$

Then the subproblem can be cast as $\min_{y \in \mathcal{Y}} \|SUy - b\|_p^p$, which is equivalent to $\min_{x \in \mathcal{Z}} \|SAx - b\|_p^p$. Therefore, with a given distribution P, applying SAA to the stochastic optimization problem associated with any basis U is equivalent to applying SAA to the original problem with matrix A.

Next, we discuss the effect of the choice of P, i.e., the sampling distribution in SAA, on the required sampling size.

Naive choice of P One choice of P is a uniform distribution, i.e., $p_i = 1/n$ for $i = 1,\ldots,n$. The resulting SAA algorithm becomes uniformly sampling s rows from the original n rows and solving the subproblem induced by the selected rows. If all the rows are equally "important", such an algorithm can be expected to work. However, consider the following toy example for which uniform sampling gives undesirable answers with high probability. Suppose the first row of the matrix contains the only nonzero element in the first column of the design matrix A. Since the only measurement of x_1 lies in the first row, in order to recover the optimal value, namely x_1^*, the first row in matrix A is crucial. However, when a uniform sampling scheme is used, the sampling size required in order to sample the first row is $\Omega(n)$. This implies that RLA with uniform sampling will fail with high probability unless the sampling size $s = \Omega(n)$.

Smarter choice of P In the above example, it is not hard to show that the leverage score of the first row is 1, i.e., it is much larger than the average value of the leverage scores. This inspires us to put more weights on "important" rows, i.e., rows with higher leverage scores. An immediate solution is to define P based on the leverage scores as follows:

$$p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j},$$

where λ_i is the i-th leverage score of A (which depends on whether one is working with ℓ_1, ℓ_2, or more general ℓ_p regression). Applying SAA with this distribution and solving the subproblem exactly recovers the recently proposed RLA methods with algorithmic leveraging for solving overdetermined ℓ_p regression problems; see Mahoney (2011); Dasgupta et al. (2009); Clarkson et al. (2013); Yang et al. (2014); Clarkson and Woodruff (2013); Meng and Mahoney (2013a); Ma et al. (2014) for details. (In RLA, this is simply solving the subproblem of the original problem, but in statistical learning theory, this has the interpretation of Empirirical Risk Minimization.) This algorithm is formally stated in Algorithm 3 in Appendix B. We also include its approximation-of-quality results from (Dasgupta et al., 2009) in Appendix B, which state that the resulting approximate solution $\hat{x}$ produces a $(1 + \epsilon)$-approximation to the objective if the sampling size s is large enough. (Note, in particular, that "large enough" here means that when the desired accuracy and failure probability are fixed, the required sampling size only depends on the lower dimension d, independent of n.)

3.2 Using SGD (SA) to solve ℓ_p regression

Applying SA to (4) and updating the weight vector using first-order information results in a SGD algorithm. It is not hard to show that, given $U = AF$ and $P = \{p_i\}_{i=1}^n$, the update rule is as follows. Suppose the ξ_t-th row is sampled; then the weight vector x_t is updated by

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

where $H = \left(FF^\top\right)^{-1} \in \mathbb{R}^{d \times d}$, η is the step size, and c_t is a constant that depends on x_t and ξ_t.

Next, we discuss how different choices of U and P affect the convergence rates of the resulting SGD algorithms. For simplicity, we restrict our discussions to unconstrained ℓ_1 regressions.

Naive choice of U **and** P Consider the following choices of U and P that lead to undesirable convergence rates. Let $U = A$. If we apply the SGD with some distribution $P = \{p_i\}_{i=1}^n$, some simple arguments in the SGD convergence rate analysis lead to a relative approximation error of

$$\frac{f(\hat{x}) - f(x^*)}{f(\hat{x})} = \mathcal{O}\left(\frac{\|x^*\|_2 \max_{1 \le i \le n}\|A_i\|_1/p_i}{\|Ax^* - b\|_1}\right), \tag{5}$$

where $f(x) = \|Ax - b\|_1$ and x^* is the optimal solution. When $\{p_i\}_{i=1}^n$ is the uniform distribution, (5) becomes $\mathcal{O}\left(n\frac{\|x^*\|_2 \cdot M}{\|Ax^* - b\|_1}\right)$, where $M = \max_{1 \le i \le n} \|A_i\|_1$ is the maximum ℓ_1 row norm of A. Alternatively, if one chooses p_i to be proportional to the row norms of A, i.e., $p_i = \frac{\|A_i\|_1}{\|A\|_1}$, then (5) becomes $\mathcal{O}\left(\frac{\|x^*\|_2 \cdot \|A\|_1}{\|Ax^* - b\|_1}\right)$. Consider the following scenario. Given A and b, we continue to append samples (z, c) satisfying $z^\top x^* = c$ and $\|z\|_2 \le M$ to A and b, respectively. This process will keep

x^*, M and $\|Ax^* - b\|_1$ unchanged. However, the value of n and $\|A\|_1$ will increase. Thus, in this case, the expected time for convergence of SGD with these naive sampling distributions might blow up as the size of the matrix grows.

Smarter choice of U and P To avoid this problem, we need to precondition the linear regression problem. If we work with a well-conditioned basis U for the range space of A and choose the sampling probabilities proportional to the row norms of U, i.e., leverage scores of A, then the resulting convergence rate on the relative error of the objective becomes $\mathcal{O}\left(\frac{\|y^*\|_2 \cdot \|U\|_1}{\|Uy^* - b\|_1}\right)$, where y^* is an optimal solution to the transformed problem. By Definition 1, if U is a well-conditioned basis, then one obtains $\|U\|_1 \leq \alpha$ and $\|y^*\|_\infty \leq \beta \|Uy^*\|_1$. Since the condition number $\alpha\beta$ of a well-conditioned basis depends only on d and since $\|Uy^* - b\|_1 / \|Uy^*\|_1$ is a constant, it implies that the resulting SGD inherits a convergence rate in a relative scale that depends on d and is independent of n.

The idea of using a preconditioner and a sampling distribution according to the leverage scores leads to our main algorithm.

4. 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 our main 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 Tables 5 and 6 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, we can estimate the row norms of U without computing the product between A and R^{-1}, in order to further reduce the running time; see Appendix A for more details. We assume that $\{\lambda_i\}_{i=1}^n$ satisfy

$$(1 - \gamma)\|U_i\|_p^p \leq \lambda_i \leq (1 + \gamma)\|U_i\|_p^p, \tag{6}$$

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:

$$p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}. \tag{7}$$

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 $F \in \mathbb{R}^{d \times d}$:

$$\min_{y \in \mathcal{Y}} h(y) = \|AFy - b\|_p^p = \mathbb{E}_{\xi \sim P}\left[|A_\xi Fy - b_\xi|^p / p_\xi\right]. \tag{8}$$

Here the matrix F is called the preconditioner for the linear system being solved; see Section 4.2 for several choices of F. Below, we show that with a suitable choice of F, the convergence rate of the

11

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 $\text{rank}(A) = d$, $x_0 \in \mathcal{Z}$, η 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 a well-conditioned basis U as described in Section 2.1.
4: Compute or estimate $\|U_i\|_p^p$ with leverage scores λ_i, for $i \in [n]$, that satisfies (6).
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 4.2 for details.
7: **for** $t = 0, \ldots, T$ **do**
8:     Pick ξ_t from $[n]$ based on distribution $\{p_i\}_{i=1}^n$.
9:

$$c_t = \begin{cases} \text{sgn}\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 1; \\ 2\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 2. \end{cases}$$

10:     Update x by

$$x_{t+1} = \begin{cases} x_t - \eta c_t H^{-1} A_{\xi_t} & \text{if } \mathcal{Z} = \mathbb{R}^d; \\ \underset{x \in \mathcal{Z}}{\arg\min} \ \eta c_t A_{\xi_t} x + \frac{1}{2}\|x_t - x\|_H^2 & \text{otherwise.} \end{cases} \tag{11}$$

       where $H = \left(FF^\top\right)^{-1}$.
11: **end for**
12: **Return** $\bar{x}$ for $p = 1$ or x_T for $p = 2$.

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., (11). Notice that if $\mathcal{Z} = \mathbb{R}^d$ and $F = I$, then the update rule can be simplified as

$$x_{t+1} = x_t - \eta c_t A_{\xi_t}. \tag{9}$$

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

$$x_{t+1} = x_t - \eta c_t H^{-1} A_{\xi_t}, \tag{10}$$

where $H = (R^\top R)^{-1}$. In the presence of constraints, (11) 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., $\bar{x} = \frac{1}{T}\sum_{t=1}^T x_t$, for ℓ_1 regression, or the last iterate, i.e., x_T, for ℓ_2 regression.

4.1 Main results for ℓ_1 and ℓ_2 regression problems

The quality-of-approximation of Algorithm 1 is presented in Proposition 5 and Proposition 6 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 $\mathcal{O}\left(\log T/T\right)$ 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 analysis of PWSGD in Proposition 14 and Proposition 15 in Appendix A. All the proofs can be found in Appendix C. The analysis of these results is based on the convergence properties of SGD; see Appendix D for technical details.

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 (6). For any vector $x \in \mathbb{R}^d$, denote by $\|x\|_H^2 = x^\top H x$ 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 [2].

Proposition 5 *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

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

Here, the expectation is taken over all the samples $\xi_1, \ldots, \xi_T$ 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 6 *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*

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

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

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

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

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

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

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

---

2. The exact expression of the optimal stepsize contains unknown quantities such as x^*. In fact, this is also the case for many SGD-type algorithms. In practice, standard techniques for searching stepsizes can be used. In our experiments, we evaluate our algorithm using theoretically optimal stepsizes as well as stepsizes after grid searching.

Here, the expectation is taken over all the samples $\xi_1, \ldots, \xi_T$, 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 [3] $\bar{\kappa}_p(U)$ 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 $(\mathbb{E}[f(x_T)] - f(x^*))/f(x^*) \leq \sqrt{c_3 \epsilon}$.

Remark. Our approach can also be applied to other type of linear regression problems such as ridge regressions in which SGD can be invoked in a standard way. In this case, the "condition number" of SGD is lower than κ due to the regularization term. The randomized preconditioning methods discussed in Section 2.1 can be used but it is an "overkill". More sophisticated preconditioning methods can be devised, e.g., based on ridge leverage scores (Cohen et al., 2015b).

4.2 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) to the stochastic optimization problem in (8). From Proposition 5 and Proposition 6, clearly, different choices of F will lead to different convergence rates in the SGD phase (reflected in $\kappa(RF)$ [4]) and additional computational costs (reflected in H in (11)).

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 (2) that shares similar properties with A. Together we have $H = R^\top R = (SA)^\top(SA) \approx A^\top A$. This implies (10) 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 (11).

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 van der Sluis (1969), 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 5 we will illustrate the tradeoffs among these three choices of preconditioners empirically.

---

3. One can show that $\bar{\kappa}_2$ is a scaled version of the standard condition number κ. $\bar{\kappa}_1$ is also related to κ with $\bar{\kappa}_1 \geq \kappa/\sqrt{nd}$. This implies that in general $\bar{\kappa}_1$ can be large without preconditioning, e.g., the `buzz` dataset used in our experiments.

4. 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.

4.3 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 5 and Table 6 for their running time $time(R)$ and preconditioning quality $\bar{\kappa}_p(U)$. Particularly, there are several available sparse preconditioning methods that run in $\mathcal{O}(\text{nnz}(A))$ plus lower order terms in d time (Clarkson and Woodruff, 2013; Meng and Mahoney, 2013a; Nelson and Nguyen, 2013; Yang et al., 2016b; Woodruff and Zhang, 2013). Second, to estimate the leverage scores, i.e., the row norms of AR^{-1}, Drineas et al. (2012); Clarkson et al. (2013) 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 $\mathcal{O}(\log n \cdot \text{nnz}(A))$. Third, Proposition 5 and Proposition 6 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.

Proposition 7 *Suppose the preconditioner in step 3 of Algorithm 1, is chosen from Table 5 or Table 6, 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 $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + d^3 \bar{\kappa}_1(U)/\epsilon^2)$ for unconstrained ℓ_1 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + time_{update} \cdot d\bar{\kappa}_1(U)/\epsilon^2)$ for constrained ℓ_1 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + d^3 \log(1/\epsilon)/\epsilon)$ for unconstrained ℓ_2 regression.*
- *It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + time_{update} \cdot d \log(1/\epsilon)/\epsilon^2)$ 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 (11).

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 $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d)/\epsilon^2)$ time for ℓ_1 regression and in $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d) \log(1/\epsilon)/\epsilon)$ 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 7, for the complexity for ℓ_1 regression, the tradeoffs in choosing preconditioners from Table 5 are reflected here. On the other hand, for ℓ_2 regression, as all the preconditioning methods in Table 5 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 Tables 2, 5 and 6. We remark that, with decaying step-sizes, it is possible to improve the dependence on ϵ from $\log(1/\epsilon)/\epsilon$ to $1/\epsilon$ (Rakhlin et al., 2012).

Finally, we remind readers that Table 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 each iteration of PWSGD only needs to solve an optimization problem with quadratic objective over the same constraints.

type	sketch	complexity
ℓ_1	Dense Cauchy (Sohler and Woodruff, 2011)	$\mathcal{O}(nd^2 \log n + d^3 \log d + d^{\frac{11}{2}} \log^{\frac{3}{2}} d/\epsilon^2)$
ℓ_1	Fast Cauchy (Clarkson et al., 2013)	$\mathcal{O}(nd \log n + d^3 \log^5 d + d^{\frac{17}{2}} \log^{\frac{9}{2}} d/\epsilon^2)$
ℓ_1	Sparse Cauchy (Meng and Mahoney, 2013a)	$\mathcal{O}(\text{nnz}(A) \log n + d^7 \log^5 d + d^{\frac{19}{2}} \log^{\frac{11}{2}} d/\epsilon^2)$
ℓ_1	Reciprocal Exponential (Woodruff and Zhang, 2013)	$\mathcal{O}(\text{nnz}(A) \log n + d^3 \log d + d^{\frac{13}{2}} \log^{\frac{5}{2}} d/\epsilon^2)$
ℓ_1	Lewis Weights (Cohen and Peng, 2015)	$\mathcal{O}(\text{nnz}(A) \log n + d^3 \log d + d^{\frac{9}{2}} \log^{\frac{1}{2}} d/\epsilon^2)$
ℓ_2	Gaussian Transform	$\mathcal{O}(nd^2 + d^3 \log(1/\epsilon)/\epsilon)$
ℓ_2	SRHT (Tropp, 2011)	$\mathcal{O}(nd \log n + d^3 \log n \log d + d^3 \log(1/\epsilon)/\epsilon)$
ℓ_2	Sparse ℓ_2 embedding (Cohen, 2016)	$\mathcal{O}(\text{nnz}(A) \log n + d^3 \log d + d^3 \log(1/\epsilon)/\epsilon)$
ℓ_2	Refinement Sampling (Cohen et al., 2015a)	$\mathcal{O}(\text{nnz}(A) \log(n/d) \log d + d^3 \log(n/d) \log d + d^3 \log(1/\epsilon)/\epsilon)$

Table 3: Summary of complexity of PWSGD with different sketching methods for computing the precondi-
tioner 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.

4.4 Complexity comparison between PWSGD and RLA

As we pointed out in Section 3, PWSGD and RLA methods with algorithmic leveraging (Ap-
pendix B) (RLA for short) are closely related as they can be viewed as methods using SA and
SAA to solve the stochastic optimization problem (4). Omitting the time for computing 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 (11). According to the theory, for the same target accu-
racy, the required s (sampling size) and T (number of iterations) are the same asymptotically, up
to logarithmic factors; see Dasgupta et al. (2009); Yang et al. (2014); Drineas et al. (2011) 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 (11). 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 (Portnoy and Koenker, 1997), $time_{sub}(s, d)$ is not even linear in s. This explains
the advantage PWSGD has 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,
$time_{sub}(s, d)/s > time_{update}$.

4.5 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 (Strohmer and Vershynin, 2009; Needell et al.,
2014). To be more specific, weighted RK algorithm can be viewed as an SGD algorithm with con-
stant 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 Karczmarz 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 6 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 5.

5. Experiments

In this section, we provide empirical evaluations of our main algorithm PWSGD. We evaluate its convergence rate and overall running time on both synthetic and real datasets. 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 4.2 for details. Note that, for PWSGD, we use the methods from Clarkson and Woodruff (2013) for preconditioning. Also, we exactly compute the row norms of AR^{-1} and use them as the leverage scores. In each experiment, the initial solution vector estimate is set as zero. The above algorithms are then run in the following manner. Each epoch contains $\lceil n/10 \rceil$ iterations. At the beginning of each epoch, we sample $\lceil n/10 \rceil$ indices according to the underlying distribution without replacement and update the weight using the $\lceil n/10 \rceil$ row samples from the data matrix. Finally, the plots are generated by averaging the results over 20 independent trials.

5.1 Empirical evaluations on synthetic datasets

Theoretically the major advantage of PWSGD is the fast convergence rate. To evaluate its performance, we compare the convergence rate of relative error, i.e., $|\hat{f} - f^*|/f^*$, with other competing algorithms including vanilla SGD and fully weighted randomized Kaczmarz (weighted-RK) algorithm (Needell et al. (2014); Strohmer and Vershynin (2009)) for solving least-squares problem (unconstrained ℓ_2 regression). For each of these methods, given a target relative error $\epsilon = 0.1$ on the objective, i.e., $(\hat{f} - f^*)/f^* = 0.1$, we use the optimal step-size suggested in the theory. In particular, for PWSGD, we are showing the convergence rate of the SGD phase after preconditioning. We stop the algorithm when the relative error reaches ϵ. In this task, we use synthetic datasets for better control over the properties on input matrices A and b. Each dataset has size 1000 by 10 and is generated in one of the following two ways.

Synthetic 1 The design matrix A has skewed row norms and skewed leverage scores. That is, 5 rows have leverage scores and row norms significantly larger than the rest[5].

Synthetic 2 The design matrix A is of the form $A = U\Sigma V^\top$ where $U \in \mathbb{R}^{1000 \times 10}$ and $V \in \mathbb{R}^{10 \times 10}$ are random orthonormal matrices and $\Sigma \in \mathbb{R}^{10 \times 10}$ is a diagonal matrix that controls $\kappa(A)$.

In both cases, the true solution x^* is a standard Gaussian vector and the response vector b is set to be Ax^* corrupted by some Gaussian noise with standard deviation 0.1.

In Figure 2, we present the results on two Synthetic 1 datasets with condition number around 1 and 5. From the plots we can clearly see that among the methods we used, PWSGD-full

---

5. Note that, in general, there is no correlation between row norms and leverage scores unless the matrix has nearly orthonormal columns. For construction details of Synthetic 1, see the construction of NG matrices Section 5.3 in Yang et al. (2016b).

and PWSGD-diag exhibit superior speed in terms of achieving the target accuracy. The relative ordering within PWSGD with three different preconditioners is consistent with the theory according to our discussions in Section 4.2. Since the datasets considered here are well-conditioned, the preconditioning phases in PWSGD-diag and PWSGD-full have similar effects and both methods perform well. However as suggested in Corollary 6, as the condition number increases (in comparison of the results in Figure 2(a) versus Figure 2(b)), other methods show degradations in convergence. Furthermore Figure 2(a) shows that the weighted-RK algorithm outperforms standard SGD. This is due to the fact that A in this dataset is well-conditioned but with non-uniform row norms.

We further investigate the relation between the condition number of A and convergence rate. As suggested in Proposition 6, for weighted SGD algorithm, the number of iterations required to solve an ℓ_2 regression problem is proportional to $\bar{\kappa}_2^2(A)\kappa^2(A) = \|(A^\top A)^{-1}\|_2^2\|A\|_2^2\|A\|_F^2 \leq \bar{\kappa}_2^4(A)$. To verify this hypothesis, we generate a sequence of A matrices using `Synthetic 2` dataset with increasing $\bar{\kappa}_2^4(A)$ values such that U and V in the sequence are constants.[6] This construction ensures that all other properties such as leverage scores and coherence (the largest leverage score) remain unchanged. Similar to Figure 2, we present the experimental results (number of iterations required for different methods versus $\bar{\kappa}_2^4(A)$) for the `Synthetic 2` dataset in Figure 3.

As shown in Figure 3, the required number of iterations of all the methods except for PWSGD-full scales linearly in $\bar{\kappa}_2^4(A)$. This phenomenon matches the result predicted in theory. A significant advantage of PWSGD-full over other methods is its robust convergence rate against variations in $\bar{\kappa}_2^4(A)$. This is mainly because its SGD phase operates on a well-conditioned basis after preconditioning and the preconditioning quality of PWSGD-full depends only on the low-dimension of A; thus increasing $\bar{\kappa}_2^4(A)$ has little effect on changing its convergence rate. Also, while the diagonal preconditioner in PWSGD-diag reduces the condition number, i.e., $\kappa(RD) \leq \kappa(R)$, its convergence rate still suffers from the increase of $\bar{\kappa}_2^4(A)$.

5.2 Time-accuracy tradeoffs

Next, we present the time-accuracy tradeoffs among these methods on the following two datasets described in Table 4.

name	#rows	# columns	$\kappa(A)$
`Year`[7]	5×10^5	90	2×10^3
`Buzz`[8]	5×10^5	77	10^8

Table 4: Summary of the two real datasets we evaluate in the experiments.

Here we test the performance of various methods in solving unconstrained ℓ_1 and ℓ_2 regression problems. Although there are no theoretical results to support the solution vector convergence on ℓ_1 regression problems with PWSGD, we still evaluate relative error in the solution vector. To further examine the performance of PWSGD methods, we also include AdaGrad, SVRG, and RLA methods with algorithmic leveraging (RLA for short) mentioned in Section 3 and Appendix B for comparisons. For AdaGrad, we use diagonal scaling and mirror descent update rule. For SVRG, we compute the full gradient every $\lceil n/2 \rceil$ iterations. As for implementation details, in all SGD-like

---

6. In `Synthetic 2`, U and V are fixed. Σ is of the form $\text{diag}(\sigma_1, \ldots, \sigma_d)$ where $\sigma_i = 1 + (i-1)q$ for $i \in [d]$. We solve for q such that $\sum_{i=1}^d \sigma_i^2 = \bar{\kappa}_2^2(U)$ for any desired value $\bar{\kappa}_2^2(U)$.

7. https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD

8. https://archive.ics.uci.edu/ml/datasets/Buzz+in+social+media+

algorithms, step-size tuning is done by grid-searching where at each trial the algorithm is run with a candidate step-size for enough iterations. Then the step-size that yields the lowest error within 10 seconds is used. The time/accuracy pair at every 2000 iterations is recorded. For RLA, we choose s from a wide range of values and record the corresponding time/accuracy pairs. The results on the two datasets are presented in Figures 4 and 5, respectively.

As we can see in Figures 4 and 5, in our algorithm PWSGD, a faster convergence comes with the additional cost of preconditioning. For example, the preconditioning phase of PWSGD takes approximately 0.5 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, SVRG, and PWSGD-noco 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 4. 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 measured in ℓ_2 norm.

We also notice that on `Buzz` (Figure 5), all the methods except for PWSGD-full and PWSGD-diag have a hard time converging to a solution with low solution error. This is due to the fact that `Buzz` has a high condition number. The advantage of applying a preconditioner is manifested.

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 Figure 4(a) and 4(b). Nevertheless, PWSGD methods may still be favorable over RLA in terms of speed and feasibility when the size of the dataset becomes increasingly larger, e.g., 10^7 by 500.
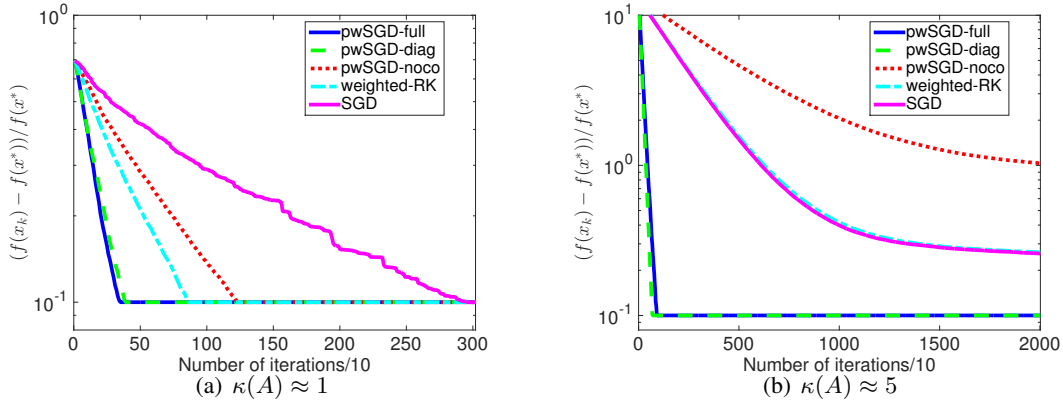
Figure 2: Convergence rate comparison of several SGD-type algorithms including PWSGD with three different choices of preconditioners for solving ℓ_2 regression on Synthetic 1 datasets with condition number around 1 and 5, respectively. For each method, the optimal step-size is set according to the theory with target accuracy $|f(\hat{x}) - f(x^*)|/f(x^*) = 0.1$. The y-axis is showing the relative error on the objective, i.e., $|f(\hat{x}) - f(x^*)|/f(x^*)$.
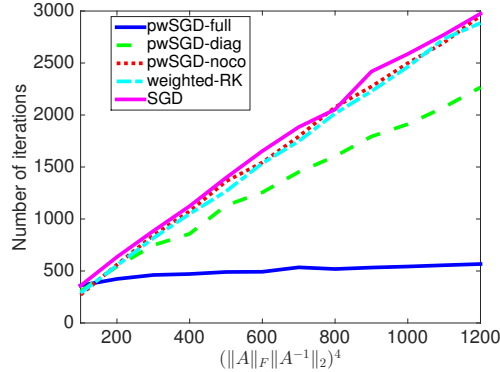

Figure 3: Convergence rate comparison of several SGD-type algorithms including PWSGD with three different choices of preconditioners for solving ℓ_2 regression on Synthetic 2 datasets with increasing condition number. For each method, the optimal step-size is set according to the theory with target accuracy $|f(\hat{x}) - f(x^*)|/f(x^*) = 0.1$. The y-axis is showing the minimum number of iterations for each method to find a solution with the target accuracy.

5.3 Empirical evaluations with sparse ℓ_2 regression

Finally, we evaluate our algorithm on a constrained problem — sparse ℓ_2 regression, which is a special case of (1). The problem formulation is as follows. Given a matrix $A \in \mathbb{R}^{n \times d}$ and a vector
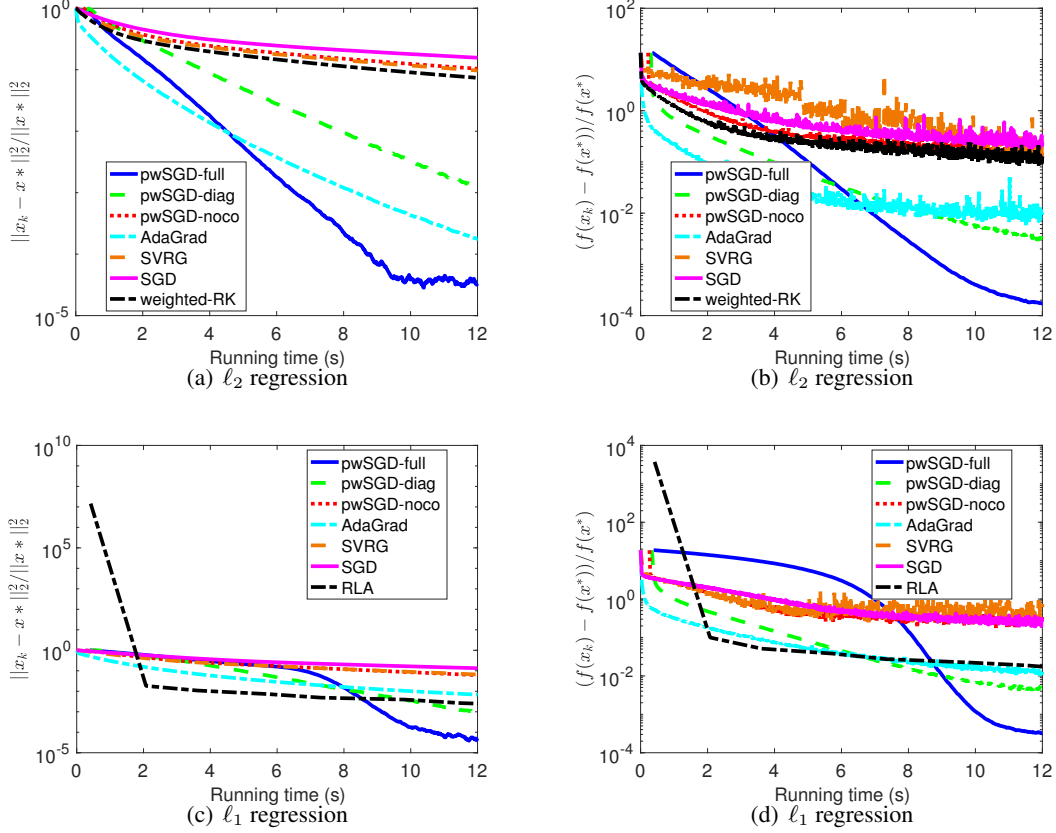
Figure 4: Time-accuracy tradeoffs of several algorithms including PWSGD with three different choices of preconditioners on year dataset. Both ℓ_1 and ℓ_2 regressions are tested and the relative error on both the objective value, i.e., $|f(\hat{x}) - f(x^*)|/f(x^*)$, and the solution vector, i.e., $\|\hat{x} - x^*\|_2^2/\|x^*\|_2^2$, are measured.

$b \in \mathbb{R}^n$, we want to solve the following constrained problem

$$\min_{\|x\|_1 \leq R} \|Ax - b\|_2, \tag{12}$$

where R controls the size of the ℓ_1-ball constraint.

When using PWSGD, according to (11) in Algorithm 1, at each iteration, a sparse ℓ_2 regression problem with size d by d needs to be solved. Here, to use the samples more efficiently, we use a mini-batch version of PWSGD. That is, in Step 8-10 of Algorithm 1, rather than picking only one row from A to compute the noisy gradient, we select m rows and average the scaled version of them. Doing this allows us to reduce the variance of the noisy gradient. In our experiments, we set $m = 200$.

In this task, the observation model is generated in the following manner, $b = Ax^* + e$ where $A \in \mathbb{R}^{n \times d}$ has independent standard normal entries, x^* has s nonzero entries and noise vector $e \in \mathbb{R}^n$ has independent standard normal entries. We evaluate both the optimization error $\|\hat{x} -$
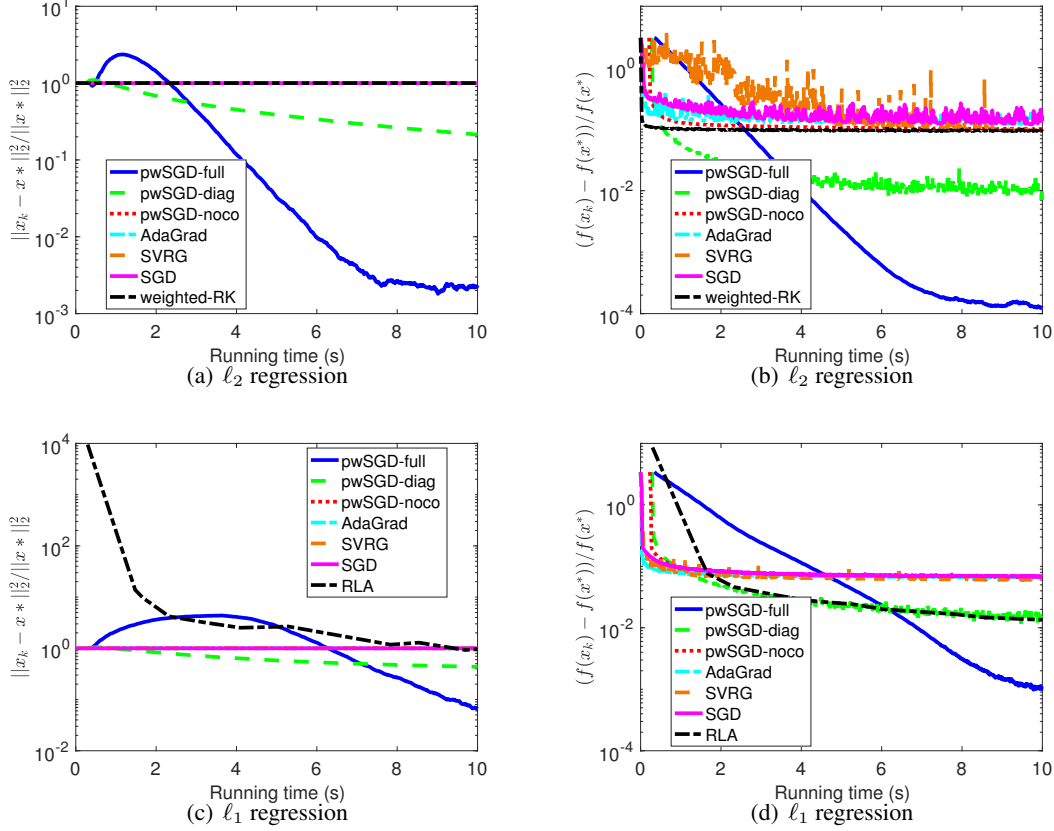
Figure 5: Time-accuracy tradeoffs of several algorithms including PWSGD with three different choices of preconditioners on buzz dataset. Both ℓ_1 and ℓ_2 regressions are tested and the relative error on both the objective value, i.e., $|f(\hat{x}) - f(x^*)|/f(x^*)$, and the solution vector, i.e., $\|\hat{x} - x^*\|_2^2/\|x^*\|_2^2$, are measured.

$x^{LS}\|_2$ and statistical error $\|\hat{x} - x^*\|_2$ of PWSGD-full with several choices of stepsize η where x^{LS} the optimal solution of problem (12). It is known that the least squares error of x^{LS} is $\|x^{LS} - x^*\|_2 \approx \sqrt{s \log(ed/s)/n}$ (Hastie et al., 2015). The statistical error can be bounded using the triangle inequality as shown below,

$$\|\hat{x} - x^*\|_2 \le \|\hat{x} - x^{LS}\|_2 + \|x^{LS} - x^*\|_2.$$

Therefore, the statistical error $\|\hat{x} - x^*\|_2$ is dominated by the least squares error $\|x^{LS} - x^*\|_2$ when the optimization error $\|\hat{x} - x^{LS}\|_2$ is small.

In Figure 6, we show the results on a data instance with $n = 1e4$, $d = 400$ and $s = 30$. Here R is set to be $R = \|x^*\|_1$ for the experimental purpose. First, we briefly describe the effect of stepsize η. When a constant stepsize is used, typically, a smaller η allows the algorithm to converge to a more accurate solution with a slower convergence rate. This is verified by Figure 6(a) in which the performance of PWSGD-full with larger η's saturates earlier at a coarser level while $\eta = 0.001$ allows the algorithm to achieve a finer solution. Nevertheless, as discussed above, the statical error

is typically dominated by the least squares error. For our choice of (n, d, s), one can show that the least squares error $\|x^{LS} - x^*\|_2^2 \approx 0.01$. Therefore, the statistical error shown in Figure 6(b) is around 0.01 when the optimization error is small enough.
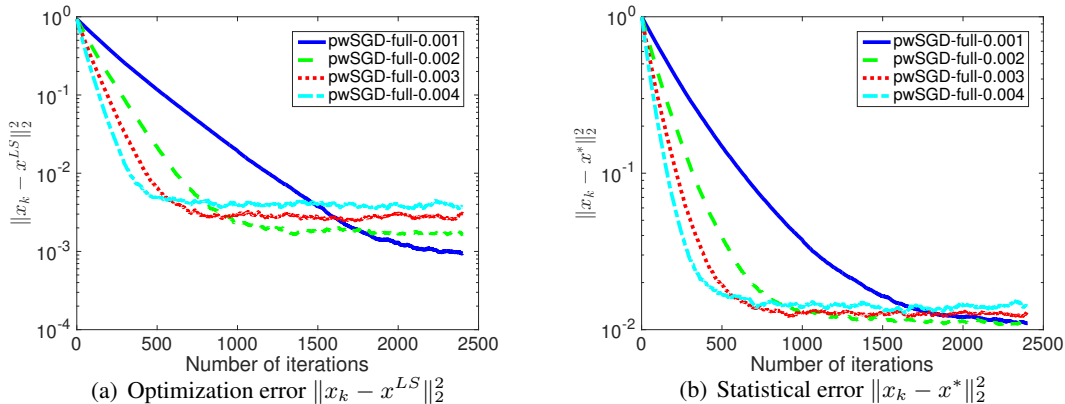

(a) Optimization error $\|x_k - x^{LS}\|_2^2$　　　　　(b) Statistical error $\|x_k - x^*\|_2^2$

Figure 6: Performance of PWSGD-full on a synthetic sparse ℓ_2 regression problem with difference choices of stepsize η. Both optimization error and statistical error are shown.

6. Connection with Coreset Methods

After viewing RLA and SGD from the stochastic optimization perspective and using that to develop our main algorithm, a natural question arises: can we do this for other types of problems? To do so, we need to define "leverage scores" for them, since they play a crucial role in this stochastic framework. Here, we first describe the coreset framework of Feldman and Langberg (2011). Then we show that—on ℓ_p regression problems—two key notions (leverage scores from RLA and sensitivities from coresets) correspond. Finally we will show what amounts to a negative result (i.e., a lower bound) for other problems. Note here, in this section, we work on constrained ℓ_p regression (1) with $p \in [1, \infty)$ and we use $\bar{A}$ to denote the augmented linear system $\begin{pmatrix} A & b \end{pmatrix}$.

6.1 Short summary of coreset methods

In Feldman and Langberg (2011), the authors propose a framework for computing a coreset of $\mathcal{F}$ to a given optimization problem of the form,

$$\text{cost}(\mathcal{F}, x) = \min_{x \in \mathcal{X}} \sum_{f \in \mathcal{F}} f(x),$$

where $\mathcal{F}$ is a set of functions from a set $\mathcal{X}$ to $[0, \infty)$. By Proposition 4, it is not hard to see, the ℓ_p regression problem (1) can be written as

$$\min_{x \in \mathcal{C}} \sum_{i=1}^{n} f_i(x),$$

23

Algorithm 2 Compute ϵ-coreset

  1: **Input:** A class of functions $\mathcal{F}$, sampling size s.
  2: **Output:** An ϵ-coreset to $\mathcal{F}$.
  3: Initialize $\mathcal{D}$ as an empty set.
  4: Compute the sensitivity $m(f)$ for each function $f \in \mathcal{F}$.
  5: $M(\mathcal{F}) \leftarrow \sum_{f \in \mathcal{F}} m(f)$.
  6: **for** $f \in \mathcal{F}$ **do**
  7:     Compute probabilities $p(f) = \frac{m(f)}{M(\mathcal{F})}$.
  8: **end for**
  9: **for** $i = 1, \ldots, s$ **do**
10:     Pick f from $\mathcal{F}$ with probability $p(f)$.
11:     Add $f/(s \cdot p(f))$ to $\mathcal{D}$.
12: **end for**
13: Return $\mathcal{D}$.

---

where $f_i(x) = |\bar{A}_i x|^p$ and $\mathcal{C} = \{x \in \mathbb{R}^{d+1} | x_{d+1} = -1\}$, in which case one can define a set of functions $\mathcal{F} = \{f_i\}_{i=1}^n$.

Central to the coreset method of Feldman and Langberg (2011) is the following notion of sensitivity, which is used to construct importance sampling probabilities, as shown in Algorithm 2, and the dimension of the given class of function, which is based as Definition 6.1 in Feldman and Langberg (2011). They are defined as below.

Definition 8 *Given a set of function $\mathcal{F} = \{f_i\}_{i=1}^n$, the* sensitivity $m(f)$ *of each function is defined as* $m(f) = \lfloor \sup_{x \in \mathcal{X}} n \cdot \frac{f(x)}{\text{cost}(\mathcal{F},x)} \rfloor + 1$, *and the* total sensitivity $M(\mathcal{F})$ *of the set of functions is defined as* $M(\mathcal{F}) = \sum_{f \in \mathcal{F}} m(f)$.

Definition 9 *The dimension of $\mathcal{F}$ is defined as the smallest integer d such that for any $G \subset \mathcal{F}$,*

$$|\{\mathbf{Range}(G, x, r) \mid x \in \mathcal{X}, r \geq 0\}| \leq |G|^d,$$

where $\mathbf{Range}(G, x, r) = \{g \in G \mid g(x) \leq r\}$.

The algorithm proposed in Feldman and Langberg (2011) is summarized in Algorithm 2 below, and the corresponding result of quality of approximation is presented in Theorem 10.

Theorem 10 *Given a set of functions $\mathcal{F}$ from $\mathcal{X}$ to $[0, \infty]$, if $s \geq \frac{cM(\mathcal{F})}{\epsilon^2}(\dim(\mathcal{F}') + \log\left(\frac{1}{\delta}\right))$, then with probability at least $1 - \delta$, Algorithm 2 returns an ϵ-coreset for $\mathcal{F}$. That is,*

$$(1 - \epsilon) \sum_{f \in \mathcal{F}} f(x) \leq \sum_{f \in \mathcal{D}} f(x) \leq (1 + \epsilon) \sum_{f \in \mathcal{F}} f(x),$$

where $\mathcal{F}' = \{f/s(f) \mid f \in \mathcal{F}\}$ is a rescaled version of $\mathcal{F}$.

6.2 Connections between RLA and coreset methods

In the following, we present two results on the connection between RLA with algorithmic leveraging, i.e., with sampling based on exact or approximate leverage scores, and coreset methods. These results originally appeared in Varadarajan and Xiao (2012). We include them here and give different proofs.

The first result shows that the sensitivities are upper bounded by a constant factor times the ℓ_p leverage scores. With this connection between leverage scores and sensitivities, it is not hard to see that applying Algorithm 2 to ℓ_p regression is exactly the same as applying Algorithm 3 (RLA sampling algorithm described in Appendix B).

Proposition 11 *Given $\bar{A} \in \mathbb{R}^{n \times (d+1)}$, let $f_i(x) = |\bar{A}_i x|^p$, for $i \in [n]$. Let λ_i be the i-th leverage score of $\bar{A}$. Then, the i-th sensitivity*

$$m(f_i) \leq n\beta^p \lambda_i + 1,$$

for $i \in [n]$ and the total sensitivity

$$M(\mathcal{F}) \leq n((\alpha\beta)^p + 1).$$

The second result is that, for the ℓ_p regression problem, the dimension of the class of functions $\dim(\mathcal{F}')$ is the same as the dimension of the subspace being considered, which is $\mathcal{O}(d)$. To be more specific, since all the $f \in \mathcal{F}'$ here are of the form $f(x) = |a^T x|^p$ for some vector $a \in \mathbb{R}^d$, we consider a broader class of functions, namely $\mathcal{A} = \{|a^T x|^p \mid a \in \mathbb{R}^d\}$, and compute its dimension.

Proposition 12 *Let $\mathcal{A} = \{|a^T x|^p \mid a \in \mathbb{R}^d\}$. We have*

$$\dim(\mathcal{A}) \leq d + 1.$$

With these results, in combination with Theorem 10, we can see that to compute a coreset $\mathcal{D}$, which leads to a $\left(\frac{1+\epsilon}{1-\epsilon}\right)$-approximate solution the ℓ_p regression using coreset method of (Feldman and Langberg, 2011), the required sampling complexity is the same (up to constants) as that of RLA sampling algorithm, as indicated by Theorem 16 (assuming $\gamma = 0$) in Appendix B.

6.3 Limitation of our approach

From the above, we see that for ℓ_p regression, a small coreset whose size only depends on d exists, and by solving it we can get a $(1 + \epsilon)$-approximation solution. This results in the same sampling algorithm as in RLA. Also, the sensitivities defined in the framework can be used as a distribution when one converts a deterministic problem into a stochastic optimization problem. We want to see whether we can extend this scheme to other problems. Indeed, beyond ℓ_p regression, the coreset methods work for any kind of convex loss function (Feldman and Langberg, 2011). However, since it depends on the total sensitivity, the size of the coreset is not necessarily small. For RLA, this translates into requiring a very large sample size to construct a good subproblem. For example, for hinge loss, we have the following example showing that the size of the coreset has an exponential dependency on d.

Proposition 13 *Define $f_i(x) = f(x, a_i) = (x^T a_i)^+$, where $x, a_i \in \mathbb{R}^d$ for $i \in [n]$. There exists a set of vectors $\{a_i\}_{i=1}^d$ such that the total sensitivity of $\mathcal{F} = \{f_i\}_{i=1}^n$ is approximately 2^d.*

This result indicates that new ideas will be needed to extend similarly RLA preconditioning ideas to weighted SGD algorithms for other types of convex optimization problems. This should not be surprising, since RLA methods have been developed for randomized *linear* algebra problems, but it suggests several directions for follow-up work.

7. Conclusion

In this paper, we propose a novel RLA-SGD hybrid algorithm called PWSGD. We show that after a preconditioning step and constructing a non-uniform sampling distribution using RLA techniques, its SGD phase inherits fast convergence rates that only depend on the lower dimension of the input matrix. For ℓ_1 regression, PWSGD displays strong advantages over RLA methods in terms of the overall complexity. For ℓ_2 regression, it has a complexity comparable to that of several state-of-the-art solvers. Empirically we show that PWSGD is preferable when a medium-precision solution is desired. Finally, we provide lower bounds on the coreset complexity for more general regression problems, which point to specific directions for future work to extend our main results.

References

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

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.

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

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

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

R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM J. on Optimization*, 26(2):1008–1031, 2016.

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

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

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

K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S. Teng. Approximating center points with iterated radon points. In *Symposium on Computational Geometry*, 1993.

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.

M. B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Symposium on Discrete Algorithms (SODA)*, 2016.

M. B. Cohen and R. Peng. ℓ_p row sampling by Lewis weights. In *Symposium on the Theory of Computing (STOC)*, 2015.

M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford. Uniform sampling for matrix approximation. In *Conference on Innovations in Theoretical Computer Science (ITCS)*, 2015a.

M. B. Cohen, C. Musco, and C. Musco. Ridge leverage scores for low-rank approximation. *CoRR*, abs/1511.07263, 2015b.

F. Curtis. A self-correcting variable-metric algorithm for stochastic optimization. In *International Conference on Machine Learning (ICML)*, 2016.

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.

P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numer. Math.*, 117(2):219–249, 2011.

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.

J. C. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Conference on Learning Theory (COLT)*, 2010.

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.

D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Symposium on Theory of Computing (STOC)*, 2011.

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

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, 2015.

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

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

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

P. Ma, M. W. Mahoney, and B. Yu. A statistical perspective on algorithmic leveraging. In *International Conference on Machine Learning (ICML)*, 2014.

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

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)*, 2013a.

X. Meng and M. W. Mahoney. Robust regression on MapReduce. In *International Conference on Machine Learning (ICML)*, 2013b.

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.

P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

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

J. Nelson and H. L. Nguyen. OSNAP: faster numerical linear algebra algorithms via sparser subspace embeddings. In *Symposium on Foundations of Computer Science (FOCS)*, 2013.

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.

M. Pilanci and M. J. Wainwright. Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares. *ArXiv e-prints*, 2014.

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.

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.

A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *International Conference on Machine Learning (ICML)*, 2012.

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

N. Sauer. On the density of families of sets. *J. Combinatorial Theory, Series A*, 13(1):145–147.

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

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

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

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

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

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

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

K. Varadarajan and X. Xiao. On the sensitivity of shape fitting problems. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2012.

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

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

J. Yang, Y. Chow, C. Ré, and M. W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. In *Symposium on Discrete Algorithms (SODA)*, 2016a.

J. Yang, X. Meng, and M. W. Mahoney. Implementing randomized matrix algorithms in parallel and distributed environments. *Proceedings of the IEEE*, 104(1):58–92, 2016b.

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

Appendix 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 Steps 3 and 4 in Algorithm 1.

In Section 2.1 we have summarized the major steps for computing the preconditioner using sketching. Below in Table 5 we provide a short summary of preconditioning methods using various sketches along with the resulting running time and condition number. Note that the running time here denotes the total running for computing the matrix R which is the sketching time plus the time

for QR factorization of the sketch. 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.

name	running time	$\bar{\kappa}_1(U)$
Dense Cauchy Transform (Sohler and Woodruff, 2011)	$\mathcal{O}(nd^2 \log d + d^3 \log d)$	$\mathcal{O}(d^{5/2} \log^{3/2} d)$
Fast Cauchy Transform (Clarkson et al., 2013)	$\mathcal{O}(nd \log d + d^3 \log d)$	$\mathcal{O}(d^{11/2} \log^{9/2} d)$
Sparse Cauchy Transform (Meng and Mahoney, 2013a)	$\mathcal{O}(\text{nnz}(A) + d^7 \log^5 d)$	$\mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$
Reciprocal Exponential Transform (Woodruff and Zhang, 2013)	$\mathcal{O}(\text{nnz}(A) + d^3 \log d)$	$\mathcal{O}(d^{\frac{7}{2}} \log^{\frac{5}{2}} d)$
Lewis Weights (Cohen and Peng, 2015)	$\mathcal{O}(\text{nnz}(A) \log n + d^3 \log d)$	$\mathcal{O}(d^{\frac{3}{2}} \log^{\frac{1}{2}} d)$

Table 5: Summary of running time and condition number, for several different ℓ_1 conditioning methods. The failure probability of each chmethod is set to be a constant.

name	running time	$\kappa_2(U)$	$\bar{\kappa}_2(U)$
Gaussian Transform	$\mathcal{O}(nd^2)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
SRHT (Tropp, 2011)	$\mathcal{O}(nd \log n + d^3 \log n \log d)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
Sparse ℓ_2 Embedding (Clarkson and Woodruff, 2013)	$\mathcal{O}(\text{nnz}(A) + d^4)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
Sparse ℓ_2 Embedding[9] (Cohen, 2016)	$\mathcal{O}(\text{nnz}(A) \log d + d^3 \log d)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$
Refinement Sampling (Cohen et al., 2015a)	$\mathcal{O}(\text{nnz}(A) \log(n/d) \log d + d^3 \log(n/d) \log d)$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{d})$

Table 6: Summary of running time and condition number, for several different ℓ_2 conditioning methods. Here, we assume $d \le n \le e^d$. The failure probability of each method is set to be a constant.

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 Clarkson et al. (2013); Drineas et al. (2012) 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 5 and Proposition 6.

Proposition 14 *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*

$$\mathbb{E}\left[f(\bar{x})\right] - f(x^*) \le \frac{1}{2\eta T} \|x^* - x_1\|_H^2 + \frac{\eta}{2} \left(c_1 \alpha \|RF\|_1\right)^2. \tag{13}$$

Hereby, the expectation is taken over all the samples $\xi_1, \ldots, \xi_T$ 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 $c_1 = \frac{1+\gamma}{1-\gamma}$.

---

9. In Cohen (2016), the author analyzes a more general version of the original count-sketch like sparse ℓ_2 embedding (Clarkson and Woodruff, 2013). By setting the sparsity parameter differently, different running time complexities can be achieved.

Proposition 15 *For $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, define $f(x) = \|Ax - b\|_2$. Algorithm 1 with $p = 2$ returns a solution vector estimate x_T that satisfies the following expected error bound*

$$\mathbb{E}\left[\|x_t - x^*\|_H^2\right] \leq \left(\frac{1 - 4\eta\left(1 - 2\eta c_1 \alpha^2 \|RF\|_2^2\right)}{\beta^2 \|(RF)^{-1}\|_2^2}\right)^T \|x_0 - x^*\|_H^2 + \frac{2c_1 \eta \bar{\kappa}_2^2(U) \kappa^2(RF) h(y^*)}{1 - 2c_1 \eta \alpha^2 \|RF\|_2^2}. \tag{14}$$

Hereby, $H = (F^{-1})^\top F^{-1}$ is the weighs of the ellipsoidal norm and the expectation is taken over all the samples $\xi_1, \ldots, \xi_T$ 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 $c_1 = \frac{1+\gamma}{1-\gamma}$.

Appendix B. RLA Methods with Algorithmic Leveraging

In this section, we present the RLA sampling algorithms with algorithmic leveraging for solving ℓ_p regression problems mentioned in Section 3. The main idea in this class of algorithms is to sample rows based on the leverage scores of $\begin{pmatrix} A & b \end{pmatrix}$ and solve the sample average approximation of the ℓ_p regression problem. This method is formally stated in Algorithm 3.

The following theorem (from Dasgupta et al. (2009)) 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 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 16 *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 for the range space of $\begin{pmatrix} A & b \end{pmatrix}$ and γ be the quality of approximation to the leverage scores satisfying (6). Then when $\epsilon < 1/2$ and the sampling size satisfies the following condition*

$$s \geq \frac{1+\gamma}{1-\gamma} \frac{(32\alpha\beta)^p}{p^2 \epsilon^2} \left((d+1) \log\left(\frac{12}{\epsilon}\right) + \log\left(\frac{2}{\delta}\right)\right), \tag{15}$$

Algorithm 3 returns a solution vector $\hat{x}$ that satisfies the following inequality with probability at least $1 - \delta$,

$$\|A\hat{x} - b\|_p \leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \|Ax^* - b\|_p, \tag{16}$$

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

Remark. Compared to the RLA algorithm described in Section 3, the algorithm described here computes the leverage scored based on a basis for the range space of the augmented linear system $\bar{A} = \begin{pmatrix} A & b \end{pmatrix}$ rather than A. One can show similar results if the basis is computed for the range space of A.

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. For 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 Drineas et al. (2011); Clarkson and Woodruff (2013) for details.

Appendix C. Proofs

Here, we present the proofs of the theoretical results in the main text.

Algorithm 3 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 $\text{rank}(\bar{A}) = k$ where $\bar{A} = \begin{pmatrix} A & b \end{pmatrix}$, $\mathcal{Z}$ and $s > 0$.
2: **Output:** An approximate solution $\hat{x} \in \mathbb{R}^d$ to problem $\text{minimize}_{x \in \mathcal{Z}} \|Ax - b\|_p^p$.
3: Compute $R \in \mathbb{R}^{k \times (d+1)}$ such that $\bar{A} = UR$ and U is an (α, β) well-conditioned basis U for the range space of $\bar{A}$.
4: Compute or estimate $\|U_i\|_p^p$ by λ_i satisfying (6) with γ, 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 ξ_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 $\hat{x} = \arg\min_{x \in \mathcal{Z}} \|SAx - Sb\|_p$.

C.1 Proof of Proposition 7

Consider the following three events:

- $\mathcal{E}_1$: Compute a matrix R such that $U = AR^{-1}$ has condition number $\bar{\kappa}_p$, and then compute $F = R^{-1}$ and $H = (FF^\top)^{-1}$.

- $\mathcal{E}_2$: Given R^{-1}, compute $\{\lambda_i\}_{i=1}^n$ as an estimation of row norms of AR^{-1} satisfying (6) with $\gamma = 0.5$.

- $\mathcal{E}_3$: For a given basis U with condition number $\bar{\kappa}_p(U)$ and $\{\lambda_i\}_{i=1}^n$ with approximation quality γ, PWSGD returns a solution with the desired accuracy with iterations $10T$ where T is specified in Proposition 5 or Proposition 6.

Since each preconditioning method shown in Table 5 successes with constant probability, $\mathcal{E}_1$ holds with a constant probability. Also, as introduced in Appendix A, $\mathcal{E}_2$ has a constant failure probability. Finally, by Markov inequality, we know that $\mathcal{E}_3$ holds with probability at least 0.9. As setting the failure probability of $\mathcal{E}_1$ and $\mathcal{E}_2$ to be arbitrarily small will not alter the results in big-O notation, we can ensure that, with constant probability, $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ holds.

Conditioned on the fact that $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ holds, to converge to the desired solution, for ℓ_1 regression, PWSGD runs in $\mathcal{O}(d\bar{\kappa}_1(U)/\epsilon^2)$ iterations. Since all the preconditioning methods in Table 6 provide $\kappa(U) = \mathcal{O}(1)$ and $\bar{\kappa}_2(U) = \mathcal{O}(\sqrt{d})$, for unconstrained ℓ_2 regression, it runs in $\mathcal{O}(d\log(1/\epsilon)/\epsilon)$ iterations. For constrained ℓ_2 regression, since an ϵ-approximate solution in terms of the solution vector measured in the prediction norm implies a $\sqrt{\epsilon}$-approximate solution on the objective, it runs in $\mathcal{O}(d\log(1/\epsilon)/\epsilon^2)$ iterations to return an ϵ-solution in the objective value.

The overall complexity is the sum of the complexity needed in each of the above events. For $\mathcal{E}_1$, it is $time(R)$ since the time for computing F and H is $\mathcal{O}(d^3)$ which can absorbed into $time(R)$ and they only have to be computed once. For $\mathcal{E}_2$, it is $\mathcal{O}(\text{nnz}(A) \cdot \log n)$. Finally, for $\mathcal{E}_3$, when the problem is unconstrained, $time_{update} = \mathcal{O}(d^2)$; when the problem is constrained, $time_{update} = \text{poly}(d)$. Combining these, we get the complexities shown in the statement. This completes the proof.

C.2 Proof of Proposition 14

The proof of this proposition is structured as follows. First we reformulate the problem using Proposition 4. Second we show that the sequence of solution vector estimates $\{x_t\}_{t=1}^T$ in Algorithm 1 is equivalent to the solution vector estimates $\{y_t\}_{t=1}^T$ obtained by running SGD on the equivalent problem. Third, we analyze the convergence rate of $\{y_t\}_{t=1}^T$ and conclude the error bound analysis.

Problem reformulation   Suppose U is an ℓ_p well-conditioned basis for the range space of A and $A = UR$ for some nonsingular matrix R. Let P be the distribution defined based on the estimation of the corresponding leverage scores. That is, for $i \in [n]$,

$$p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}, \tag{17}$$

where λ_i is an estimation of $\|U_i\|_p^p$ satisfying

$$(1 - \gamma)\|U_i\|_p^p \leq \lambda_i \leq (1 + \gamma)\|U_i\|_p^p. \tag{18}$$

This implies

$$\frac{1 - \gamma}{1 + \gamma}\frac{\|U_i\|_p^p}{\|U\|_p^p} \leq p_i \leq \frac{1 + \gamma}{1 - \gamma}\frac{\|U_i\|_p^p}{\|U\|_p^p}. \tag{19}$$

From Proposition 4, recall that for any non-singular matrix $F \in \mathbb{R}^{(d+1)\times(d+1)}$, the constrained ℓ_p regression problem

$$\min_{x \in \mathcal{Z}} f(x) := \|Ax - b\|_p^p \tag{20}$$

can be equivalently written as the following stochastic optimization problem,

$$\min_{y \in \mathcal{Y}} h(y) = \|URFy - b\|_p^p = \mathbb{E}_{\xi \sim P}\left[|U_\xi RFy - b_\xi|^p / p_\xi\right]. \tag{21}$$

Notice that by comparing to the objective function defined in (1) where $f(x) = \|Ax - b\|_p$, we rewrite $f(x)$ into the form of the sum of subfunctions, i.e., $f(x) = \|Ax - b\|_p^p$, so that SGD can be applied.

Equivalence of sequences   By using the following linear transformation, one notices that the sequence $\{x_t\}_{t=1}^T$ obtained by (11) in Algorithm 1 has a one-to-one correspondence to the sequence $\{y_t\}_{t=1}^T$ obtained by running SGD on problem (21):

$$\begin{aligned} Fy_t &= x_t, \\ F\bar{y} &= \bar{x}, \\ Fy^* &= x^*. \end{aligned} \tag{22}$$

Thus with condition (22), immediately the objective function value has the following equivalence as well:

$$\begin{aligned} h(y_t) &= f(x_t), \\ h(\bar{y}) &= f(\bar{x}), \\ h(y^*) &= f(x^*), \end{aligned} \tag{23}$$

where $\bar{x} = \frac{1}{T}\sum_{i=1}^{T}x_t$, $\bar{y} = \frac{1}{T}\sum_{i=1}^{T}y_t$ and x^* and y^* are the optimal point to optimization problem (20) and (21) respectively.

Now we prove (22) by induction. By defining $Fy_0 = x_0$, one immediately shows that the equivalence condition holds at the base case ($t = 0$). Now by induction hypothesis, assume (22) holds for case $t = k$. Now for $t = k + 1$, we show that x_{k+1} returned by Algorithm 1 and y_{k+1} returned by the update rule of SGD satisfy (22).

For simplicity, assume that at k-th iteration, the i-th row is picked. For subfunction $h_k(y) = |U_iRFy|^p - b_i/p_i$, its (sub)gradient is

$$g_k(y) = p \cdot \mathrm{sgn}(U_iRFy - b_i) \cdot (U_iRFy - b_i)^{p-1} \cdot U_iRF/p_i, \tag{24}$$

for which the SGD update rule becomes

$$y_{k+1} = \arg\min_{y\in\mathcal{Y}} \eta\langle y - y_k, c_k U_iRF\rangle + \frac{1}{2}\|y_k - y\|_2^2, \tag{25}$$

where $c_k = p \cdot \mathrm{sgn}(U_iRFy - b_i) \cdot (U_iRFy - b_i)^{p-1}/p_i$ is the corresponding (sub)gradient. Recall the linear transformation $Fy_k = x_k$, feasible set $\mathcal{Y} = \{y \in \mathbb{R}^k | y = F^{-1}x, x \in \mathcal{Z}\}$ and input matrix $A_i = U_iR$, the update rule (25) becomes

$$x_{k+1} = \arg\min_{x\in\mathcal{Z}} \eta c_k A_i x + \frac{1}{2}\|F^{-1}(x_k - x)\|_2^2. \tag{26}$$

The equation above is exactly the update performed in (11). In particular, when $\mathcal{Z} = \mathbb{R}^d$, i.e., in the unconstrained case, (26) has a closed-form solution as shown in (11). From the above analysis on the equivalence between (25) and (26), one notices x_{k+1} and y_{k+1} satisfy the relationship defined in (22), i.e., the induction hypothesis holds at $t = k + 1$.

Therefore by induction, we just showed that condition (22), and therefore condition (23), hold for any t.

Convergence rate   Based on the equivalence condition in (23), it is sufficient to analyze the performance of sequence $\{y_t\}_{t=1}^{T}$. When $p = 1$, the objective function is non-differentiable. Thus by substituting the subgradient of an ℓ_1 objective function to the update in (25), one notices that the SA method simply reduces to stochastic subgradient descent. We now analyze the convergence rate of running stochastic subgradient descent on problem (21) with $p = 1$.

Suppose the i-th row is picked at the t-th iteration. Recall that the (sub)gradient of the sample objective $|U_iRFy - b_i|/p_i$ in (25) is expressed as

$$g_t(y) = \mathrm{sgn}(U_iRFy - b_i) \cdot U_iRF/p_i. \tag{27}$$

Hence, by inequality (19), the norm of $g_t(y)$ is upper-bounded as follows:

$$\begin{aligned}
\|g_t(y)\|_1 &= \|U_iRF \cdot \mathrm{sgn}(U_iRFy - b_i)\|_1/p_i \\
&\leq |RF|_1\|U_i\|_1\frac{1+\gamma}{1-\gamma} \cdot \frac{|U|_1}{\|U_i\|_1} \leq \alpha|RF|_1\frac{1+\gamma}{1-\gamma}.
\end{aligned} \tag{28}$$

In above, we use the property of the well-conditioned basis U. Furthermore by Proposition 17 and the equivalence condition in (23), for $H = \left(FF^\top\right)^{-1}$ we have

$$
\begin{aligned}
\mathbb{E}\left[f(\bar{x})\right] - f(x^*) &= \mathbb{E}\left[h(\bar{y})\right] - h(y^*) && (29) \\
&\leq \frac{1}{2\eta(T+1)}\|y^* - y_0\|_2^2 + \frac{\eta}{2}\left(\alpha|RF|_1\frac{1+\gamma}{1-\gamma}\right)^2 \\
&= \frac{1}{2\eta(T+1)}\|x^* - x_0\|_H^2 + \frac{\eta}{2}\left(\alpha|RF|_1\frac{1+\gamma}{1-\gamma}\right)^2, && (30)
\end{aligned}
$$

which completes the proof.

C.3 Proof of Proposition 5

By Proposition 17, when the step-size equals to

$$
\eta = \frac{\|y^* - y_0\|_2}{\alpha|RF|_1\sqrt{T+1}}\frac{1-\gamma}{1+\gamma},
$$

the expected error bound is given by

$$
\mathbb{E}\left[h(\bar{y})\right] - h(y^*) \leq \alpha|RF|_1\frac{\|y^* - y_0\|_2}{\sqrt{T+1}}\frac{1+\gamma}{1-\gamma}. \tag{31}
$$

By simple algebraic manipulations, we have that

$$
\begin{aligned}
\frac{1}{\sqrt{d}}\|y^*\|_2 &\leq \|y^*\|_\infty = \|(RF)^{-1}RFy^*\|_\infty \leq |(RF)^{-1}|_1\|RFy^*\|_\infty \\
&\leq \beta|(RF)^{-1}|_1\|URFy^*\|_1 = c_3\beta|(RF)^{-1}|_1 h(y^*), \tag{32}
\end{aligned}
$$

where $c_3 = \|URFy^*\|_1/h(y^*)$. In above, we use the property of the well-conditioned basis U.

Furthermore from inequality (31) and the equivalence condition in (23), the expected relative error bound can be upper-bounded by

$$
\begin{aligned}
\frac{\mathbb{E}\left[f(\bar{x})\right] - f(x^*)}{f(x^*)} &= \frac{\mathbb{E}\left[h(\bar{y})\right] - h(y^*)}{h(y^*)} \\
&\leq \frac{c_3\sqrt{d}\beta|(RF)^{-1}|_1}{\|y^*\|_2}\left(\alpha|RF|_1\frac{\|y^* - y_0\|_2}{\sqrt{T+1}}\frac{1+\gamma}{1-\gamma}\right) \\
&\leq |RF|_1|(RF)^{-1}|_1\frac{\|y^* - y_0\|_2}{\|y^*\|_2}\left(\frac{c_3\sqrt{d}\alpha\beta}{\sqrt{T+1}}\frac{1+\gamma}{1-\gamma}\right). \tag{33}
\end{aligned}
$$

Since the right hand side of the above inequality is a function of stopping time $T > 0$, for any arbitrarily given error bound threshold $\epsilon > 0$, by setting the right hand side to be ϵ, one obtains the following stopping condition:

$$
\frac{\sqrt{d}\alpha\beta}{\sqrt{T+1}} = \frac{\epsilon}{c_1c_3\sqrt{c_2}|RF|_1|(RF)^{-1}|_1}, \tag{34}
$$

35

where the above constants are given by

$$c_1 = \frac{1+\gamma}{1-\gamma}, \quad c_2 = \frac{\|x_0 - x^*\|_H^2}{\|x^*\|_H^2} = \frac{\|y_0 - y^*\|_2^2}{\|y^*\|_2^2}.$$

Rearranging the above terms we know that after

$$T \geq \frac{d\alpha^2\beta^2 c_1^2 c_2 c_3^2}{\epsilon^2} |RF|_1^2 |(RF)^{-1}|_1^2 \tag{35}$$

iterations, the relative expected error is upper-bounded by $\epsilon > 0$, i.e.,

$$\frac{\mathbb{E}\left[f(\bar{x})\right] - f(x^*)}{f(x^*)} \leq \epsilon. \tag{36}$$

This completes the proof.

C.4 Proof of Proposition 15

Similar to the proof of Proposition 14, the proof of this proposition is split into three parts: **Problem reformulation**, **Equivalence of sequences** and **Convergence rates**. From the proof of Proposition 14, one notices that the proofs in **Problem reformulation** and **Equivalence of sequences** hold for general p, and thus the proofs hold for the case when $p = 2$ as well. Now we proceed to the proof of the convergence rate. Again by the equivalence condition, we can show the convergence rate of solution vector estimate $\{x_t\}$ by showing the convergence rate achieved by the sequence $\{y_t\}$, i.e., the convergence rate of SGD of problem (21) for $p = 2$.

Throughout the rest of the proof, we denote

$$f(x) = \|Ax - b\|_2^2, \quad h(y) = \|AFy - b\|_2^2. \tag{37}$$

Denote by $H = \left(FF^\top\right)^{-1}$ the weighs of the ellipsoidal norm. Also recall that when the leverage scores satisfy the error condition in (6), we have the following condition

$$\frac{1-\gamma}{1+\gamma}\frac{\|U_i\|_2^2}{\|U\|_2^2} \leq p_i \leq \frac{1+\gamma}{1-\gamma}\frac{\|U_i\|_2^2}{\|U\|_2^2}. \tag{38}$$

Also, we assume that U is (α, β)-conditioned with $\bar{\kappa}_2(U) = \alpha\beta$. Based on Definition 1, we have

$$\alpha^2 = \|U\|_F^2, \tag{39}$$
$$\beta^2 = \|(U^\top U)^{-1}\|_2, \tag{40}$$

and thus

$$\bar{\kappa}_2^2(U) = \|(U^\top U)^{-1}\|_2 \cdot \|U\|_F^2 = \alpha^2\beta^2. \tag{41}$$

Before deriving the convergence rate, we compute a few constants.

$$\mu = 2\sigma_{\min}^2(AF) = \frac{2}{\left\|\left((URF)^\top URF\right)^{-1}\right\|_2^2} \geq \frac{2}{\left\|(U^\top U)^{-1}\right\|_2 \cdot \|(RF)^{-1}\|_2^2} = \frac{2}{\beta^2 \cdot \|(RF)^{-1}\|_2^2}, \tag{42}$$

and

$$\sup_i L_i = \sup_i \frac{2\|A_i F\|_2^2}{p_i} = \sup_i \frac{2\|U_i RF\|_2^2}{p_i} \le 2c_1\|U\|_F^2 \cdot \|RF\|_2^2 = 2c_1\alpha^2 \cdot \|RF\|_2^2, \qquad (43)$$

and

$$
\begin{aligned}
\sigma^2 = \mathbb{E}_{i\sim\mathcal{D}}\left[\|g_i(y^*)\|^2\right] &= 4\sum_{i=1}^{n}(A_i F y^* - b_i)^2\|A_i F\|^2/p_i \\
&= 4\sum_{i=1}^{n}(U_i RF y^* - b_i)^2\|U_i RF\|^2/p_i \\
&\le 4c_1\|RF\|_2^2\|U\|_F^2\left(\sum_{i=1}^{n}(U_i RF y^* - b_i)^2\right) \\
&= 4c_1\|U\|_F^2 \cdot \|RF\|_2^2 \cdot h(y^*) \\
&= 4c_1\alpha^2 \cdot \|RF\|_2^2 \cdot h(y^*). \qquad (44)
\end{aligned}
$$

Equipped with these constant and from Proposition 18, we have the following error bound of the solution vector estimate $\{y_t\}_{t=1}^{T}$ generated by the weighted SGD algorithm

$$
\begin{aligned}
&\mathbb{E}\left[\|x_T - x^*\|_H^2\right] \\
&= \mathbb{E}\left[\|y_T - y^*\|_2^2\right] \\
&\le \left(1 - 4\eta\sigma_{\min}^2(AF)\left(1 - \eta\sup_i\frac{2\|A_i F\|_2^2}{p_i}\right)\right)^T \|y_0 - y^*\|_2^2 \\
&\quad + \frac{2\eta\sum_{i=1}^{n}(A_i F y^* - b_i)^2\|A_i F\|_2^2/p_i}{\sigma_{\min}^2(AF)(1 - \eta\sup_i\frac{2\|A_i F\|_2^2}{p_i})} \\
&= \left(1 - 4\eta\sigma_{\min}^2(AF)\left(1 - \eta\sup_i\frac{2\|A_i F\|_2^2}{p_i}\right)\right)^T \|x_0 - x^*\|_H^2 \\
&\quad + \frac{2\eta\sum_{i=1}^{n}(A_i F y^* - b_i)^2\|A_i F\|_2^2/p_i}{\sigma_{\min}^2(AF)(1 - \eta\sup_i\frac{2\|A_i F\|_2^2}{p_i})} \\
&\le \left(\frac{1 - 4\eta\left(1 - 2\eta c_1\alpha^2\|RF\|_2^2\right)}{\beta^2\|(RF)^{-1}\|_2^2}\right)^T \|x_0 - x^*\|_H^2 + \frac{2c_1\eta\bar{\kappa}_2^2(U)\kappa^2(RF)h(y^*)}{1 - 2\eta c_1\alpha^2\|RF\|_2^2}. \qquad (45)
\end{aligned}
$$

Notice that the above equalities follow from the equivalence condition in (23). Combining the results from the above parts completes the proof of this lemma.

C.5 Proof of Proposition 6

Throughout the proof, we denote

$$f(x) = \|Ax - b\|_2^2, \quad h(y) = \|AFy - b\|_2^2. \qquad (46)$$

Denote by $H = \left(FF^\top\right)^{-1}$ the weights of the ellipsoidal norm. Also recall the following constants defined in the statement of proposition

$$c_1 = \frac{1 + \gamma}{1 - \gamma}, \; c_2 = \frac{\|y_0 - y^*\|_2^2}{\|y^*\|_2^2} = \frac{\|x_0 - x^*\|_H^2}{\|x^*\|_H^2}, \; c_3 = \frac{\|Ax^*\|_2^2}{f(x^*)}. \qquad (47)$$

Before diving into the detailed proof, we first show a useful inequality.

$$c_3 h(y^*) = c_3 f(x^*) = \|Ax^*\|_2^2 = \|URFy^*\|_2^2 \geq \mu \|y^*\|_2^2/2. \tag{48}$$

Now we show the first part. For an arbitrary target error $\epsilon > 0$, using (42), (43), (44) and setting

$$\frac{c_3 \epsilon \cdot h(y^*)}{\|AF\|_2^2} \to \epsilon \tag{49}$$

in Corollary 19 we have that when the step-size is set to be

$$\eta = \frac{1}{4} \frac{c_3 \epsilon \cdot \sigma_{\min}^2(AF) \cdot h(y^*)/\|AF\|_2^2}{\sum_{i=1}^n (A_i F y^* - b_i)^2 \|A_i F\|_2^2/p_i + c_3 \left(\epsilon \cdot h(y^*)/\|AF\|_2^2\right) \sigma_{\min}^2(AF) \sup_i \frac{\|A_i F\|_2^2}{p_i}}, \tag{50}$$

then after

$$\begin{aligned}
\log &\left(\frac{2\|y_0 - y^*\|_2^2}{c_3 \epsilon \cdot h(y^*)/(\|U\|_2^2 \|RF\|_2^2)} \right) \cdot \\
&\left(c_1 \alpha^2 \beta^2 \|RF\|_2^2 \|(RF)^{-1}\|_2^2 + \frac{c_1 \alpha^2 \beta^4 \|U\|_2^2 \|RF\|_2^4 \|(RF)^{-1}\|_2^4}{c_3 \epsilon} \right) \\
\leq \log &\left(\frac{2\|U\|_2^2 \|RF\|_2^2 \cdot \|y_0 - y^*\|_2^2}{c_3 \epsilon \cdot h(y^*)} \right) \left(c_1 \bar{\kappa}_2^2(U) \kappa^2(RF) + \frac{c_1 \bar{\kappa}_2^2(U) \kappa^2(U) \kappa^4(RF)}{c_3 \epsilon} \right) \\
\leq \log &\left(\frac{2c_2 \kappa^2(U) \kappa^2(RF))}{\epsilon} \right) \left(c_1 \bar{\kappa}_2^2(U) \kappa^2(RF) \right) \left(1 + \frac{\kappa^2(U) \kappa^2(RF)}{c_3 \epsilon} \right) \tag{51}
\end{aligned}$$

iterations, the sequence $\{y_t\}_{k=1}^T$ generated by running weighted SGD algorithm satisfies the error bound

$$\|y_T - y^*\|_2^2 \leq \frac{c_3 \epsilon \cdot h(y^*)}{\|AF\|_2^2}. \tag{52}$$

Notice that in (51), we used (48). From this, we have

$$\begin{aligned}
\|A(x_T - x^*)\|_2^2 &= \|AFF^{-1}(x_T - x^*)\|_2^2 \\
&\leq \|AF\|_2^2 \cdot \|x_T - x^*\|_H^2 \\
&= \|AF\|_2^2 \cdot \|y_T - y^*\|_2^2 \\
&= c_3 \epsilon \cdot h(y^*) \\
&= \epsilon \|Ax^*\|_2^2. \tag{53}
\end{aligned}$$

For the second part, we show the result for general choice of F. The proof is basically the same as that of the first part except that we set

$$\frac{2\epsilon h(y^*)}{\|AF\|_2^2} \to \epsilon \tag{54}$$

in Corollary 19. The resulting step-size η and number of iterations required T become

$$\eta = \frac{1}{4} \frac{2\epsilon \cdot \sigma_{\min}^2(AF) \cdot h(y^*)/\|AF\|_2^2}{\sum_{i=1}^n (A_i F y^* - b_i)^2 \|A_i F\|_2^2/p_i + \left(2\epsilon \cdot h(y^*)/\|AF\|_2^2\right) \sigma_{\min}^2(AF) \sup_i \frac{\|A_i F\|_2^2}{p_i}} \tag{55}$$

and

$$T = \log\left(\frac{c_2\kappa^2(U)\kappa^2(RF))}{\epsilon}\right)\left(c_1\bar{\kappa}_2^2(U)\kappa^2(RF)\right)\left(1 + \frac{\kappa^2(U)\kappa^2(RF)}{2\epsilon}\right). \tag{56}$$

Setting $F = R^{-1}$ recovers the value of T shown in Proposition 6. The sequence $\{y_t\}_{k=1}^T$ generated by running weighted SGD algorithm satisfies the error bound

$$\|y_T - y^*\|_2^2 \leq \frac{2\epsilon h(y^*)}{\|AF\|_2^2}. \tag{57}$$

Notice that when the problem is unconstrained, by smoothness of the objective $h(y)$, we have

$$h(y_T) - h(y^*) \leq \|AF\|_2^2 \cdot \|y_T - y^*\|_2^2 \leq 2\epsilon h(y^*). \tag{58}$$

Then by (23), we have

$$f(x_T) \leq (1 + 2\epsilon)f(x^*) \leq (1 + 2\epsilon + \epsilon^2)f(x^*). \tag{59}$$

This implies

$$\sqrt{f(x_T)} \leq (1 + \epsilon)\sqrt{f(x^*)}. \tag{60}$$

This completes the proof since $\sqrt{f(x)} = \|Ax - b\|_2$.

C.6 Proof of Theorem 10

Let G_f consist of m_f copies of g_f and $G = \bigcup_{f\in\mathcal{F}} G_f$. We may view the sampling step in Algorithm 2 as follows. Sample s items uniformly from G independently with replacement and denote the corresponding subset of samples by S. Then rescale every function in S by $M(\mathcal{F})/s$ and obtain $\mathcal{D}$.

By Theorem 4.1 in Feldman and Langberg (2011), we know that if the above intermediate set S is an $(\epsilon \cdot n/M(\mathcal{F}))-$approximation of the set G, then the resulting set $\mathcal{D}$ is a desired ϵ-coreset for $\mathcal{F}$. Indeed, S is such a set according to Theorem 6.10 in Feldman and Langberg (2011).

C.7 Proof of Proposition 11

We use A to denote $\bar{A}$ for the sake of simplicity. Also define the sensitivity at row index $i \in [n]$ as

$$s_i = n \cdot \sup_{x\in\mathcal{C}} \frac{|A_i x|^p}{\sum_{j=1}^n |A_j x|^p}. \tag{61}$$

Suppose $U \in \mathbb{R}^{n\times k}$ is an (α, β) well-conditioned basis of the range space of A satisfying $A = UR$, where $k = \text{rank}(A)$ and $R \in \mathbb{R}^{k\times(d+1)}$. Then from (61), we have that

$$\frac{s_i}{n} = \sup_{x\in\mathcal{C}} \frac{|A_i x|^p}{\|Ax\|_p^p} = \sup_{x\in\mathcal{C}} \frac{|U_i Rx|^p}{\|URx\|_p^p} = \sup_{y\in\mathcal{C}'} \frac{|U_i y|^p}{\|Uy\|_p^p} \leq \sup_{y\in\mathcal{C}'} \frac{\|U_i\|_p^p\|y\|_q^p}{\|y\|_q^p/\beta^p} = \beta^p\|U_i\|_p^p = \beta^p \cdot \lambda_i, \tag{62}$$

where $\mathcal{C}' = \{y \in \mathbb{R}^d | y = Rx, x \in \mathcal{C}\}$ is a one-to-one mapping. The first inequality follows from Hölder's inequality with $\frac{1}{p} + \frac{1}{q} = 1$ and the properties of well-conditioned bases. According to the definition of sensitivity $m(f_i) = \lfloor s_i \rfloor + 1$, the above property implies

$$m(f_i) \leq n\beta^p\lambda_i + 1. \tag{63}$$

which implies $M(\mathcal{F}) = \sum_{i=1}^n s_i \leq (n\beta^p \sum_{i=1}^n \lambda_i) + n = n((\alpha\beta)^p + 1)$, and completes the proof.

C.8 Proof of Proposition 12

According to Definition 9, we only have to show that for any arbitrary constant n and set of points $G = \{a_1, \ldots, a_n\} \subseteq \mathbb{R}^d$, the following condition holds:

$$|\{\mathbf{Range}(G, x, r)|x \in \mathcal{X}, r \geq 0\}| \leq n^{d+1},$$

where $\mathbf{Range}(G, x, r) = \{a_i | |a_i^\top x|^p \leq r\}$ is the region located in the $p-$norm ellipsoid $|a_i^\top x|^p = r$. Since the following condition holds: $\{a_i | |a_i^\top x|^p \leq r\} = \{a_i | |a_i^\top x| \leq r^{\frac{1}{p}}\}$ and the constant r is non-negative and arbitrary. Without loss of generality, we assume $p = 1$ in the above definition, i.e., $\mathbf{Range}(G, x, r) = \{a_i | |a_i^\top x| \leq r\}$.

Notice that for every x and r, $\mathbf{Range}(G, x, r)$ is a subset of G. Hence, we may view it as a binary classifier on G, denoted by $c_{x,r}$. Given $x \in \mathcal{X}$ and $r \geq 0$, for any $a_i \in G$ we have that

$$c_{x,r}(a_i) = \begin{cases} 1, & \text{if } |a_i^\top x| \leq r; \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, one immediately sees that $|\{\mathbf{Range}(G, x, r)|x \in \mathcal{X}, r \geq 0\}|$ is the shattering coefficient of $C := \{c_{x,r} | x \in \mathcal{X}, r \geq 0\}$ on n points, denoted by $s(C, n)$. To bound the shattering coefficient of C, we provide an upper bound based on its VC dimension.

We claim that the VC dimension of C is at most $d + 1$. By contradiction, suppose there exists $n + 2$ points such that any labeling on these $n + 2$ points can be shattered by C. By Radon's Theorem (Clarkson et al., 1993), we can partition these points into two disjoint subsets, namely, V and W with size n_1 and n_2 respectively, where the intersection of their convex hulls is nonempty. Let b be a point located in the intersection of the convex hulls of V and W, which in general can be written as

$$b = \sum_{i=1}^{n_1} \lambda_i v_i = \sum_{i=1}^{n_2} \sigma_i w_i, \tag{64}$$

where $\lambda_i \geq 0$, $\sigma_i \geq 0$ and $\sum_{i=1}^{n_1} \lambda_i = \sum_{i=1}^{n_2} \sigma_i = 1$.

By the above assumption, we can find vector $x \in \mathbb{R}^n$ and nonnegative constant r such that the following conditions hold:

$$-r \leq x^\top v_i \leq r, \quad i = 1, \ldots, n_1; \tag{65}$$
$$x^\top w_i > r \text{ or } x^\top w_i < -r, \quad i = 1, \ldots, n_2. \tag{66}$$

By combining the conditions in (64), (65) and (66), we further obtain both inequalities

$$-r \leq b^\top x \leq r, \tag{67}$$

and

$$b^\top x < -r \text{ or } b^\top x > r, \tag{68}$$

which is clearly paradoxical! This concludes that the VC dimension of C is less than or equal to $d + 1$. Furthermore, by Sauer's Lemma (Sauer), for $n \geq 2$ the shattering coefficient $s(C, n) = |\{\mathbf{Range}(G, x, r)|x \in \mathcal{X}, r \geq 0\}|$ is less than n^{d+1}, which completes the proof of this proposition.

C.9 Proof of Proposition 13

Without loss of generality, assume the low dimension d is even (because if d is odd, we can always add an extra arbitrary row to input matrix A and upper bound the size of the original total sensitivity set by the same analysis). Let $a_i \in [0, 1]^d$ be a vector with exactly $d/2$ elements to be 1. For each $i \in [n]$, let $B_i = \{j | a_{ij} = 1\}$, where a_{ij} denotes the j-th element of vector a_i. For fixed i, define x as follows,

$$x_j = \begin{cases} 2/d, & \text{if } j \in B_i, \\ -d, & \text{otherwise.} \end{cases} \tag{69}$$

One immediately notices from the above expression that $x^\top a_i = 1$. Thus for $j \neq i$, $a_j \neq a_i$, there exists an index $k \in [d]$ such that $a_{jk} = 1$ but $a_{ik} = 0$. Furthermore the above condition implies

$$x^\top a_j = \sum_{l=1}^{d} x_l a_{jl} = \sum_{l \in B_j, l \neq k}^{d} x_l a_{jl} + \sum_{l \neq B_j}^{d} x_l a_{jl} + x_k a_{jk} \leq (d/2 - 1)(2/d) - d < 0, \tag{70}$$

which further implies $f_j(x) = x^\top a_j = 0$; Therefore, the i-th sensitivity becomes

$$s_i = \sup_x \frac{f_i(x)}{\sum_{i=j}^{n} f_j(x)} \geq 1. \tag{71}$$

Since the above condition holds for arbitrary index $i \in [n]$, and we have $\binom{d}{d/2}$ number of vectors a_i, i.e., $n = \binom{d}{d/2}$, this concludes that the size of the total sensitivity set is at least $\binom{d}{d/2} \approx 2^d$.

Appendix D. Stochastic Gradient Descent

Consider minimizing the following objective

$$\text{minimize}_{x \in \mathcal{X}} f(x) = \mathbb{E}_{i \sim P} [f_i(x)]. \tag{72}$$

Stochastic gradient descent (SGD) exploits the following update rule

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \eta \langle x - x_t, g_{\xi_t}(x_t) \rangle + \frac{1}{2} \|x - x_t\|_2^2, \tag{73}$$

where $\xi_t \in [n]$ is an index drawn according to P, $g_{\xi_t}(x) = \nabla f_{\xi_t}(x)$ and $\mathbb{E}_{\xi_t \sim P} [f_{\xi_t}(x)] = f(x)$. When $\mathcal{X} = \mathbb{R}^d$, the update rule (73) boils down to $x_{t+1} = x_t - \eta g_{\xi_t}(x_t)$. Note here, if $f_{\xi_t}(x)$ is not differentiable, we take $g_{\xi_t}(x_t)$ to be one of its sub-gradients, i.e., $g_{\xi_t}(x_t) \in \partial f_{\xi_t}(x_t)$. In this case, SGD boils down to stochastic sub-gradient method. For simplicity, we still refer to the algorithms as SGD.

In the following, we present two results regarding the convergence rate of SGD on problem with non-strongly convex objective and strongly convex objective, respectively.

D.1 Non-strongly convex case

Here we analyze the case where the objective function $f(x)$ is not strongly convex. Also, each sub-function is not necessary differentiable. That is, $g_i(x)$ can be a sub-gradient of function f_i at x.

Proposition 17 *Assume that $\frac{1}{2}\|\cdot\|_2^2 \geq \frac{\lambda}{2}\|\cdot\|^2$ for some norm $\|\cdot\|^2$. Also assume that $\|g_t(x_t)\|_* \leq M$ for any $t > 0$ where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. The output $\bar{x} = \frac{1}{T+1}\sum_{t=1}^{T} x_t$ of SGD satisfies, for any $y \in \mathcal{X}$,*

$$\mathbb{E}\left[f(\bar{x})\right] - f(y) \leq \frac{\|y - x_0\|_2^2}{2\eta(T+1)} + \frac{\eta}{2\lambda}M^2. \tag{74}$$

In particular, when $\eta = \frac{\|y - x_0\|_2}{M}\sqrt{\frac{\lambda}{T+1}}$, we have

$$\mathbb{E}\left[f(\bar{x})\right] - f(y) \leq M\|y - x_0\|_2\sqrt{\frac{1}{(T+1)\lambda}}. \tag{75}$$

Proof From Lemma 1 in Duchi et al. (2010), at step t, we have that

$$\eta(f_t(x_t) - f_t(y)) \leq \frac{1}{2}\|y - x_t\|_2^2 - \frac{1}{2}\|y - x_{t+1}\|_2^2 + \frac{\eta^2}{2\lambda}\|g_t(x_t)\|_*^2. \tag{76}$$

Conditioned on x_t, taking the conditional expectation with respect to ξ_t on both sides, we have

$$\mathbb{E}\left[\eta(f_t(x_t) - f_t(y))|x_t\right] \leq \mathbb{E}\left[\frac{1}{2}\|y - x_t\|_2^2 - \frac{1}{2}\|y - x_{t+1}\|_2^2 + \frac{\eta^2}{2\lambda}\|g_t(x_t)\|_*^2|x_t\right]. \tag{77}$$

Noticing that $\mathbb{E}_{\xi_t \sim P}\left[f_t(x)\right] = f(x)$, we have

$$\eta f(x_t) - \eta f(y) \leq \frac{1}{2}\|y - x_t\|_2^2 + \mathbb{E}\left[-\frac{1}{2}\|y - x_{t+1}\|_2^2 + \frac{\eta^2}{2\lambda}\|g_t(x_t)\|_*^2|x_t\right]. \tag{78}$$

Then by taking the expectation over x_t and using the fact that $\|g_t(x_t)\|_* \leq M$, we have

$$\mathbb{E}\left[\eta f(x_t)\right] - \eta f(y) \leq \mathbb{E}\left[\frac{1}{2}\|y - x_t\|_2^2\right] - \mathbb{E}\left[\frac{1}{2}\|y - x_{t+1}\|_2^2\right] + \frac{\eta^2}{2\lambda}M^2. \tag{79}$$

Summing up the above equation with $t = 0, \ldots, T$ and noticing $\|y - x_{t+1}\|_2^2 \geq 0$, we have

$$\eta\sum_{t=0}^{T}\mathbb{E}\left[f(x_t)\right] - \eta(T+1)f(y) \leq \frac{1}{2}\|y - x_0\|_2^2 + \frac{\eta^2(T+1)}{2\lambda}M^2. \tag{80}$$

Finally by convexity of f, we have that

$$\mathbb{E}\left[f(\bar{x})\right] - f(y) \leq \frac{\|y - x_0\|_2^2}{2\eta(T+1)} + \frac{\eta}{2\lambda}M^2. \tag{81}$$

In particular with $\eta = \frac{\|y - y_1\|_2}{M}\sqrt{\frac{\lambda}{T+1}}$, we have

$$\mathbb{E}\left[f(\bar{x})\right] - f(y) \leq M\|y - x_0\|_2\sqrt{\frac{1}{(T+1)\lambda}}, \tag{82}$$

which completes the proof.

■

D.2 Strongly convex case

Here we analyze the case where the objective function $f(x)$ is strongly convex. We make the following two assumptions:

(A1) Function $f(x)$ is strongly convex with modulus μ. That is, for any $x, y \in \mathcal{X}$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|_2^2. \tag{83}$$

(A2) For each $i \in [n]$, the gradient of each sub-function $\nabla f_i(x)$ is Lipschitz continuous with constant L_i. That is, for any $x, y \in \mathcal{X}$,

$$\|\nabla f_i(y) - \nabla f_i(x)\|_2 \leq L_i\|y - x\|_2. \tag{84}$$

The following results also appeared in Needell et al. (2014).

Proposition 18 *Under assumption (A1), (A2), the sequence $\{x_t\}$ generated by SGD satisfies*

$$\mathbb{E}\left[\|x_T - x^*\|_2^2\right] \leq (1 - 2\eta\mu(1 - \eta \sup L_i))^T\|x_0 - x^*\|_2^2 + \frac{\eta\sigma^2}{\mu(1 - \eta \sup L_i)}, \tag{85}$$

where $\sigma^2 = \mathbb{E}_{i\sim\mathcal{D}}\left[\|\nabla f_i(x^*)\|_2^2\right]$ and x^* is the optimal solution to (72).

Proof The proof essentially follows the same lines of arguments as in Needell et al. (2014). The only difference is that, here we are working on the constrained problem where update rule (73) is equivalent to

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta g_t(x_t)). \tag{86}$$

Notice that $\Pi_{\mathcal{X}}(x)$ is a projection operator to the feasible set $\mathcal{X}$ and it is non-expansive. This further implies

$$\|x_{t+1} - x^*\|_2^2 = \|\Pi_{\mathcal{X}}(x_t - \eta g_t(x_t)) - x^*\|_2^2 \leq \|x_t - \eta g_t(x_t) - x^*\|_2^2. \tag{87}$$

The rest of the proof follows analogous arguments in Needell et al. (2014). ∎

Corollary 19 *Given a target accuracy $\epsilon > 0$, and let the step-size be $\eta = \frac{\epsilon\mu}{2\sigma^2 + 2\epsilon\mu \sup L_i}$. Then after*

$$T \geq \log\left(\frac{2\|x_0 - x^*\|^2}{\epsilon}\right)\left(\frac{\sigma^2}{\epsilon\mu^2} + \frac{\sup L_i}{\mu}\right) \tag{88}$$

iterations, we have that

$$\mathbb{E}\left[\|x_T - x^*\|_2^2\right] \leq \epsilon. \tag{89}$$

Proof The proof can be found in Needell et al. (2014). ∎