Faster Constrained Linear Regression via Two-step Preconditioning

Di Wang Jinhui Xu

Department of Computer Science and Engineering,
State University of New York at Buffalo,
Buffalo, New York 14260-2500, USA
Email: {dwang45,jinhui}@buffalo.edu

Abstract

In this paper, we study the large scale constrained linear regression problem and propose a two-step preconditioning method, which is based on some recent developments on random projection, sketching techniques and convex optimization methods. Combining the method with (accelerated) mini-batch SGD, we can achieve an approximate solution with a time complexity lower than that of the state-of-the-art techniques for the low precision case. Our idea can also be extended to the high precision case, which gives an alternative implementation to the Iterative Hessian Sketch (IHS) method with significantly improved time complexity. Experiments on benchmark and synthetic datasets suggest that our methods indeed outperform existing ones considerably in both the low and high precision cases.

Keywords: linear regression, gradient descent, random projection

1. Introduction

Linear regression with convex constraints is a fundamental problem in Machine Learning, Statistics and Signal Processing, since many other problems, such as SVM, LASSO, signal recovery [1], can be all formulated as constrained linear regression problems. Thus, the problem has received a great deal of attentions from both the Machine Learning and Theoretical Computer Science communities. The problem can be formally defined as follows,

$$\min_{x\in\mathcal{W}} f(x) = \|Ax - b\|_2^2,$$

where A is a matrix in $\mathbb{R}^{n\times d}$ with $d < n < e^d$, $\mathcal{W}$ is a closed convex set and $b \in \mathbb{R}^n$ is the response vector. The goal is to find an $x \in \mathcal{W}$ such that $f(x) \leq (1+\epsilon)\min_{x\in\mathcal{W}} f(x)$ or $f(x) - \min_{x\in\mathcal{W}} f(x) \leq \epsilon$, where ϵ is the approximation error.

Roughly speaking, there are two types of methods to solve the problem. The first type of techniques is based on Stochastic Gradient Descent (SGD). Recent developments on first-order stochastic methods, such as Stochastic Dual Coordinate Ascent (SDCA) [2], Stochastic Variance Reduced Gradient (SVRG) [3] and Katyusha [4], have made significant improvements on the convergence speed of large scale optimization problems in both theory and practice [5], which provide the potential for us to obtain faster solution to our problem.

The second type of techniques is based on randomized linear algebra. Among them, random projection and sketching are commonly used theoretical tools in many optimization problems as preconditioner, dimension reduction or sampling techniques to reduce the time complexity. This includes low rank approximation [6], SVM [7], column subset selection [8] and l_p regression (for $p \in [1, 2]$) [9; 10]. Thus, it is very tempting to combine these two types of techniques to develop faster methods with theoretical or statistical guarantee for more constrained optimization problems. Recently, quite a number of works have successfully combined the two types of techniques. For example, [11] proposed faster methods for Ridge Regression and Empirical Risk Minimization, respectively, by using SVRG, Stochastic Gradient Descent (SGD) and low rank approximation. [12] achieved guarantee for Empirical Risk Minimization by using random projection in dual problem.

In this paper, we revisit the preconditioning methods for solving large-scale constrained linear regression problem, and propose a new method called **two-step preconditioning**. Combining this method with some recent developments on large scale convex optimization problems, we are able to achieve faster algorithms for both the low ($\epsilon \approx 10^{-1} \sim 10^{-4}$) and high ($\epsilon \leq 10^{-10}$) precision cases. Specifically, our main contributions can be summarized as follows.

1. For the low precision case, we first propose a novel algorithm called HDpwBatchSGD (*i.e.,* Algorithm 2) by combining the method of two step preconditioning with mini-batch SGD. Mini-batch SGD is a popular way for improving the efficiency of SGD. It uses several samples, instead of one, in each iteration and runs the gradient descent updating on all these samples (simultaneously). Ideally, we would hope for a factor of r speed-up on the convergence if using a batch of size r. However, this is not always possible for general case. Actually in some cases, there is even no speed-up at all when a large-size batch is used [19; 20; 21]. A unique feature of our method is its optimal speeding-up with respect to the batch size, *i.e.* the iteration complexity will decrease by a factor of b if we increase the

Method	Complexity for Unconstrained	Complexity for General Constraint Set	Precision
[13]	$O\left(nd\log(\frac{d}{\epsilon}) + d^3\log n\log d + \frac{d^3\log n}{\epsilon}\right)$	$O\left(nd\log n + \text{poly}(d, \frac{1}{\epsilon^2})\right)$	Low
pwSGD[14]	$O\left(nd\log n + d^3\log n\log d + \frac{d^3\log(\frac{1}{\epsilon})}{\epsilon}\right)$	$O\left(nd\log n + d^3\log n\log d + \frac{\text{poly}(d)\log(\frac{1}{\epsilon})}{\epsilon}\right)$	Low
HDpwBatchSGD	$O\left(nd\log n + \frac{d^2\log n}{\epsilon^2} + \frac{d^3\log n}{r\epsilon^2}\right)$	$O\left(nd\log n + \frac{d^2\log n}{\epsilon^2} + \frac{\text{poly}(d)\log n}{r\epsilon^2}\right)$	Low
HDpwBatchAccSGD	$O\left(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{d^3\log n}{r\epsilon} + rd^2\log\frac{1}{\epsilon}\right)$	$O\left(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{\text{poly}(d)\log n}{r\epsilon} + r\text{poly}(d)\log\frac{1}{\epsilon}\right)$	Low
[15; 16]	$O\left(nd\log\frac{d}{\epsilon} + d^3\log d\right)$	—	High
IHS [17]	$O\left(nd\log d\log\frac{1}{\epsilon} + d^3\log\frac{1}{\epsilon}\right)$	$O\left((nd\log d + \text{poly}(d))\log\frac{1}{\epsilon}\right)$	High
Preconditioning+SVRG (pwSVRG) (Alogrithm 7)	$O\left(nd\log n + (nd + d^3)\log\frac{1}{\epsilon}\right)$	$O\left(nd\log n + (nd + \text{poly}(d))\log\frac{1}{\epsilon}\right)$	High
pwGradient	$O\left(nd\log n + (nd + d^3)\log\frac{1}{\epsilon}\right)$	$O\left(nd\log n + (nd + \text{poly}(d))\log\frac{1}{\epsilon}\right)$	High

Table 1: Summary of the time complexity of several linear regression methods for finding x_t such that $\|Ax_t - b\|_2^2 - \|Ax^* - b\|_2^2 \leq \epsilon\|Ax^* - b\|_2^2$. For sketching based methods, we use the Subsampled Randomized Hadamard Transform (SRHT) [18] as the sketch matrix. All methods run in sequential environment. r is an input in our method. '—' means not applicable.

batch size by a factor of b.

We also use the two-step preconditioning method and Multi-epoch Stochastic Accelerated mini-batch SGD proposed in [22] to obtain another slightly different algorithm called HDpwBatchAccSGD (*i.e.,* Algorithm 3 and 4), which has the time complexity lower than that of the state-of-the-art technique [14] and HDpwBatchSGD.

2. The optimality on speeding-up in HDpwBatchSGD and HDpwBatchAccSGD further inspires us to think about how it will perform if using the whole gradient, *i.e.* projected Gradient Descent, which leads to another algorithm called pwGradient (*i.e.,* Algorithm 6). We find that it actually allows us to have an alternative implementation of the Iterative Hessian Sketch (IHS) method [17], which is arguably the state-of-the-art technique for the high precision case. Particularly, we are able to show that one step of sketching is sufficient for IHS, instead of a sequence of sketchings used in the current form of IHS. This enables us to considerably improve the time complexity of IHS.

3. Finally, we implement our algorithms and test them on both large synthetic and real benchmark datasets. Numerical results confirm our theoretical analysis of HDpwBatchSGD, HDpwBatchAccSGD and pwGradient. Also, our methods outperform existing ones in both low and high precision cases.

This paper is a substantially extended version of our previous work appeared in AAAI'18 [23]. The following are the main added contents. Firstly, we add detailed algorithms of HDpwBatchAccSGD and pwSVRG, which have not been discussed in [23] (see Algorithm 3, 4 and 7). Secondly, we provide the proofs for all theorems and lemmas. Thirdly, we expand the previous work by validating our results with additional synthetic and real world datasets. More specifically, for the low precision case, we add experimental results for HDpwBatchAccSGD and show its superiority in the low precision case, compared with other existing methods and HDpwBatchSGD. For the high precision case, we provide comparisons with more large scale real datasets and show that our method is faster. We also conduct experimental studies on the relative error with different sketch size.

The rest of the paper is organized as follows. Section 2 introduces some related work. Section 3 gives some background on random projection and stochastic gradient descent. Section 4 describes our proposed algorithms for the low precision case. Section 5 presents our algorithm for the high precision case. Finally, we experimentally study our methods in Section 6, and conclude them in Section 7.

2. Related Work

There is a vast number of papers studying the large scale constrained linear regression problem from different perspectives, such as [24; 25]. We mainly focus on those results that have theoretical time complexity guarantees (note that the time complexity cannot depend on the condition number of A, such as [26]), due to their similar natures to ours. We summarize these methods in **Table 1**.

For the low precision case, [13] directly uses sketching with a very large sketch size of $\text{poly}(\frac{1}{\epsilon^2})$, which is difficult to determine the optimal sketch size in practice (later, we show that our proposed method avoids this issue). The state-of-the-art technique is probably the one in [14], which presents an algorithm for solving the general l_p regression problem and shares with ours the first step of preconditioning. Their method then applies the weighted sampling technique, based on the leverage score and SGD, while ours first conducts a further step of

preconditioning, using uniform sampling in each iteration, and then applies mini-batch SGD. Although their paper mentioned the mini-batch version of their algorithm, there is no theoretical guarantee on the quality and convergence rate, while our methods provide both and runs faster in practice. Also we have to note that, in the case of $O(\frac{\log(\frac{1}{\epsilon})}{\epsilon}) \geq \log n$, the time complexity of HDpwBatchAccSGD is less than it in pwSGD theoretically when we set the batchsize $r = O(\sqrt{\frac{\text{poly}(d)\log n}{\epsilon \log(1/\epsilon)}})$. [27] also uses mini-batch SGD to solve the linear regression problem. Their method is based on importance sampling, while ours uses the much simpler uniform sampling; furthermore, their convergence rate heavily depends on the condition number and batch partition of A, which means that there is no fixed theoretical guarantee for all instances.

For the high precision case, unlike the approach in [15], our method can be extended to the constrained case. Compared with IHS [17], ours uses only one step of sketching and thus has a lower time complexity when $\epsilon \leq \frac{1}{n}$. Although a similar time complexity can be achieved by using the two-step preconditioning method and SVRG (see Algorithm 7), our method pwGradient performs better in practice. We notice that [28] has recently studied the large scale linear regression with constraints via (Accelerated) Gradient Projection and IHS. But there is no guarantee on the time complexity, and it is also unclear how to choose the best parameters. For these reasons, we do not compare their results with ours here.

3. Preliminaries

Let A be a matrix in $\mathbb{R}^{n\times d}$ with $e^d > n > d$ and $d = \text{rank}(A)$ (note that our proposed methods can be easily extended to the case of $d > \text{rank}(A)$), and A_i and A^j be its i-th row (*i.e.,* $A_i \in \mathbb{R}^{1\times d}$) and j-th column, respectively. Let $\|A\|_2$ and $\|A\|_F$ be the spectral norm and Frobenius norm of A, respectively, and $\sigma_{\min}(A)$ be the minimal singular value of A.

We first give the formal definition of the problem to be studied throughout the paper.

Large Scaled Constrained Linear Regression. Let $A \in \mathbb{R}^{n\times d}$ be a dataset of size n represented as a matrix, where each data record has d dimensions. The dataset is also associated with a response vector $b \in \mathbb{R}^n$. The objective is to solve the following optimization problem

$$\min_{x\in\mathcal{W}} f(x) = \|Ax - b\|_2^2,$$

where $\mathcal{W}$ is some (un)bounded constraint set, such as ℓ_1 or ℓ_2 norm ball. For a given approximation error ϵ, the goal is to find an $x \in \mathcal{W}$ such that $f(x) \leq (1+\epsilon)\min_{x\in\mathcal{W}} f(x)$ or $f(x) - \min_{x\in\mathcal{W}} f(x) \leq \epsilon$ with as less time complexity as possible, where the time complexity should depend only on n, d and ϵ.

Then, we give several definitions and lemmas that will be used throughout this paper.

3.1. Randomized Linear Algebra

We first give the definition of $(\alpha, \beta, 2)$-conditioned matrix. Note that it is a special case of (α, β, p)-well conditioned basis in [25].

Definition 1 (((α,β,2)-conditioned) [14])**.** *A matrix $U \in \mathbb{R}^{n\times d}$ is called (α,β,2)-conditioned if $\|U\|_F \leq \alpha$ and for all $x \in \mathbb{R}^{d\times 1}$, $\beta\|Ux\|_2 \geq \|x\|_2$,* i.e., *$\sigma_{min}(U) \geq \frac{1}{\beta}$.*

Note that if U is an orthogonal matrix, it is $(\sqrt{d},1,2)$-conditioned. Thus we can view an $(\alpha,\beta,2)$-conditioned matrix as a generalized orthogonal matrix.

The purpose of introducing the concept of $(\alpha,\beta,2)$-conditioned is for obtaining in much less time a matrix which can approximate the orthogonal basis of matrix A. Clearly, if we directly calculate the orthogonal basis of A, it will take $O(nd^2)$ time. However, we can get an $\left(O\left(\sqrt{d}\right), O(1), 2\right)$-conditioned matrix U from A (called $\left(O\left(\sqrt{d}\right), O(1), 2\right)$-conditioned basis of A) in $o(nd^2)$ time, through **Algorithm 1** by using the sketch matrix. That is, we first calculate SA and perform the QR-decomposition of SA. Note that in practice we can just set $O(1)$ as a small constant and return R instead of AR^{-1}, since computing AR^{-1} needs $O(nd^2)$ time.

Definition 2 (Sketch Matrix)**.** *A (random) matrix $S \in \mathbb{R}^{s\times n}$ is called a sketch matrix for A, if for all $x \in \mathbb{R}^d$ with high probability*

$$(1 - O(1))\|Ax\|_2 \leq \|SAx\|_2 \leq (1 + O(1))\|Ax\|_2,$$

where $O(1)$ is a sufficiently small constant.

Algorithm 1 Constructing $(O(\sqrt{d}),O(1),2)$-conditioned basis of A

1: Construct a sketch matrix $S \in \mathbb{R}^{s\times n}$ with $n > s > d$ (see Table 2 for details on constructing the matrix) that satisfies the following condition with high probability, $\forall x \in \mathbb{R}^d$,

$$(1 - O(1))\|Ax\|_2 \leq \|SAx\|_2 \leq (1 + O(1))\|Ax\|_2,$$

2: [Q,R]=QR-decomposition(SA), where $Q \in \mathbb{R}^{s\times d}$ is an orthogonal matrix. Then AR^{-1} is an $O\left(\sqrt{d}\right), O(1), 2\right)$-conditioned basis of A.

3: **return** AR^{-1} or R

Next, we give the definition of Randomized Hadamard Transform [18], which is the tool to be used in the second step of our preconditioning.

Definition 3. *(Randomized Hadamard Transform) $M = HD \in \mathbb{R}^{n\times n}$ is called a Randomized Hadamard Transform, where n is assumed to be 2^s for some integer s, if $D \in \mathbb{R}^{n\times n}$ is a diagonal Rademacher matrix (that is, each D_{ii} is drawn independently from $\{1,-1\}$ with probability 1/2), and $H \in \mathbb{R}^{n\times n}$ is an $n \times n$ Walsh-Hadamard Matrix scaled by a factor of $1/\sqrt{n}$,* i.e.,

$$H = \frac{1}{\sqrt{n}}H_n, H_n = \begin{pmatrix} H_{\frac{n}{2}} & H_{\frac{n}{2}} \\ H_{\frac{n}{2}} & -H_{\frac{n}{2}} \end{pmatrix}, H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Randomized Hadamard Transform has two important features. One is that it takes only $O(n \log n)$ time to multiply a vector, and the other is that it can "spread out" orthogonal matrices.

Since an (α,β,2)-conditioned matrix can be viewed as an approximate orthogonal matrix, an immediate question is whether an (α,β,2)-conditioned matrix can also achieve the same result. We answer this question by the following theorem, which is interesting in its own right.

Theorem 1. *Let HD be a Randomized Hadamard Transform, and $U \in \mathbb{R}^{n\times d}$ be an (α,β,2)-conditioned matrix. Then, the following holds for any constant $c > 1$:*

$$Pr\left\{\max_{i=1,2,\ldots,n} \|(HDU)_i\|_2 \geq \left(1 + \sqrt{8\log(cn)}\right)\frac{\alpha}{\sqrt{n}}\right\} \leq \frac{1}{c}. \quad (1)$$

In order to proof Theorem 1, we need the following lemma on the tail bound of Lipschitz convex function on Rademacher random vector.

Lemma 2. *(Lipschitz Tail Bound [29]) Let f be a convex function on vectors having L-Lipschitz property, and ϵ be a Rademacher vector. Then for any $t \geq 0$, the following inequality holds*

$$Pr\{f(\epsilon) \geq \mathbb{E}f(\epsilon) + Lt\} \leq e^{-\frac{t^2}{8}}.$$

Proof. Consider a fixed row index $j \in \{1, 2, \ldots, n\}$. Let

$$f(x) = \|e_j^T H\text{diag}(x)U\|_2 = \|x^T \text{diag}(e_j^T H)U\|_2.$$

Then $f(x)$ is convex and

$$\|f(x) - f(y)\| \leq \|x - y\|_2 \|\text{diag}(e^T H)\|_2 \|U\|_2 \leq \frac{\alpha}{\sqrt{n}}\|x - y\|_2.$$

Since each entry of H is either $\frac{-1}{\sqrt{n}}$ or $\frac{1}{\sqrt{n}}$. Thus $f(x)$ is $\frac{\alpha}{\sqrt{n}}$-Lipschitz. By the fact that $f(\epsilon)$ is a Rademacher function, we have

$$\mathbb{E}f(\epsilon) \leq [\mathbb{E}f^2(\epsilon)]^{\frac{1}{2}} = (\mathbb{E}\|\epsilon^T \text{diag}(e_j^T H)U\|_2^2)^{\frac{1}{2}} = \|\text{diag}(e_j^T H)U\|_F$$
$$\leq \|\text{diag}(e_j^T H)\|_2 \|U\|_F \leq \frac{\alpha}{\sqrt{n}}.$$

Then by Lemma 2, taking $t = \sqrt{8\log(cn)}$ and the union for all arrow indices, we have the theorem. □

By Theorem 1, we can make each row of HDU have no more than one value with high probability. Since $\alpha = O(\sqrt{d})$, the norm of each row will small when $n \gg d$ after preconditioning by the Ramdomized Hadamard Transform. Also since H, D are orthogonal, we have $\|HDUy - HDb\|_2 = \|Uy - b\|_2$ for any $y \in \mathbb{R}^d$.

3.2. SGD and Mini-batch SGD for Strongly Smooth Convex Functions

Consider the following general case of a convex optimization problem: $\min_{x\in\mathcal{W}} F(x) = \mathbb{E}_{i\sim D} f_i(x)$, where i is drawn from the distribution of $D = \{p_i\}_{i=1}^n$ and $\mathcal{W}$ is a closed convex set. We assume the following.

Assumption 1. *$F(\cdot)$ is L-Lipschitz. That is, for any $x, y \in \mathcal{W}$,*

$$\|\nabla F(x) - \nabla F(y)\|_2 \leq L\|x - y\|_2.$$

Assumption 2. *$F(x)$ has strong convexity parameter μ,* i.e.,

$$\langle x - y, \nabla F(x) - \nabla F(y)\rangle \geq \mu\|x - y\|_2^2, \forall x, y \in \mathcal{W}.$$

3.2.1. SGD

Now let the Stochastic Gradient Descent (SGD) update in the $(k + 1)$-th iteration be

$$x_{k+1} = \arg\min_{x\in\mathcal{W}} \eta_k\langle\nabla f_{i_k}(x_k), x - x_k\rangle + \frac{1}{2}\|x - x_k\|_2^2 \quad (2)$$
$$= P_{\mathcal{W}}(x_k - \eta_k \nabla f_{i_k}(x_k))$$

where i_k is drawn from the distribution D, x_0 is the initial number, and $P_{\mathcal{W}}$ is the projection operator. If we denote

$$x^* = \arg\min_{x\in\mathcal{W}} F(x), \sigma^2 = \sup_{x\in\mathcal{W}} \mathbb{E}_{i\sim D}\|\nabla f_i(x) - \nabla F(x)\|_2^2,$$

then we have the following theorem, given in [30].

Theorem 3. *If Assumption 1 holds, after T iterations of the SGD iterations of (2) with fixed step-size*

$$\eta = \min\left(\frac{1}{2L}, \sqrt{\frac{D_{\mathcal{W}}^2}{2T\sigma^2}}\right), \quad (3)$$

where $D_{\mathcal{W}} = \sqrt{\max_{x\in\mathcal{W}} \frac{1}{2}\|x\|_2^2 - \min_{x\in\mathcal{W}} \frac{1}{2}\|x\|_2^2}$. Then the inequality $\mathbb{E}F(x_T^{avg}) - F(x^*) \leq \frac{3\sqrt{2}D_{\mathcal{W}}\sigma}{\sqrt{T}}$ is true, where $x_T^{avg} = \frac{\sum_{i=1}^T x_i}{T}$. Which means after

$$T = \Theta\left(\frac{D_{\mathcal{W}}^2\sigma^2}{\epsilon^2}\right) \quad (4)$$

iterations, we have $\mathbb{E}F(x_T^{avg}) - F(x^*) \leq \epsilon$.

3.2.2. Mini-batch SGD

Instead of sampling one term in each iteration, mini-batch SGD samples several terms in each iteration and takes the average. Below we consider the uniform sampling version,

$$\min_{x\in\mathcal{W}} F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x). \quad (5)$$

Note that for a mini-batch of size r, let τ denote the sampled indices and

$$g_\tau = \frac{\sum_{i\in\tau} \nabla f_i(x)}{r},$$

where each index in τ is i.i.d uniformly sampled. Then, we have $\sigma_{batch}^2 = \sup_{x\in\mathcal{W}} \mathbb{E}_\tau\|g_\tau - \nabla F(x)\|_2^2 \leq \frac{\sigma^2}{r}$. This means that the variance can be reduced by a factor of r if we use a sample of size r.

Remark 1. *Note that our mini-batch sampling strategy is different from the one in [27], which is to partition all the indices into $\lceil\frac{n}{r}\rceil$ groups and samples only within one group in each iteration.*

4. Two-step Preconditioning Mini-batch SGD

4.1. Main Idea

The main idea of our algorithm is to use two steps of preconditioning to reform the problem in the following way,

$$\min_{x\in\mathcal{W}} \|Ax-b\|_2^2 = \min_{y\in\mathcal{W}'} \|Uy-b\|_2^2 \tag{6}$$

$$= \|HDUy - HDb\|_2^2 = \frac{1}{n}\sum_{i=1}^{n} n\|(HDU)_i y - (HDb)_i\|_2^2, \tag{7}$$

where $\mathcal{W}'$ in the first equality is the convex set corresponding to $\mathcal{W}$ and the second equality is due to the fact that matrix HD is orthogonal. Below we discuss the idea behind these reformulations.

The first step of the preconditioning (6) is to obtain U, an $\left(O\left(\sqrt{d}\right), O(1), 2\right)$-conditioned basis of A (*i.e.,* $U = AR^{-1}$; see Algorithm 1), which means that the function in problem (6) is an $O(d)$-smooth (actually it is $O(1)$-smooth, see Table 2) and $O(1)$-strongly convex function. The reason for using matrix U is as the follows: If we directly use the (Stochastic) Gradient Descent methods to the problem, the number of needed iterations will depend on the condition number of A, *i.e.,* κ_A, which could be quite large and thus make the algorithm slow. Hence, one way to avoid this is to reformulate the problem by using other regular matrix, instead of A. The most direct way is to use the orthogonal matrix of A, *i.e.,* U_A. That is, we can reformulate the problem as follows

$$\min_{x\in\mathcal{W}} \|Ax-b\|_2^2 = \min_{y\in\mathcal{W}''} \|U_A y - b\|_2^2.$$

Let $A = U_A R_A$, where $U_A \in \mathbb{R}^{n\times d}, R_A \in \mathbb{R}^{d\times d}$. Then, we know that the optimal $y^* = R_A x^*$. Thus, it is sufficient to approximate y^*. Since the condition number of U_A is $O(1)$, it is better to solve the latter optimization problem. However, since getting U_A, R_A needs $O(nd^2)$ time, it is impractical in the large scale setting. Fortunately, as we discussed before, we can get an $(O(\sqrt{d}),O(1),2)$-conditioned basis of A and U which approximate U_A in $o(nd^2)$ time with condition number $O(1)$. Thus, we will use U instead of U_A, which is the main idea of (6).

The second step of the preconditioning (7) is to use Randomized Hadamard Transform to 'spread out' the row norm of U (by **Theorem 1**). The reason for spreading out the row norm is the following: we now want to use SGD to solve (6). However, it is well known that the non-uniform sampling w.r.t to the ℓ_2-norm of the rows of U is better than uniformly sampling [14] and the time for computing the norm of rows is $O(nd^2)$. We use HD as a precondition matrix to make the norm of each row in HDU almost the same. Thus, uniformly sampling will be almost the same as the non-uniform sampling w.r.t the norm of rows with time complexity $O(nd\log n)$.

After applying the two-step preconditioning, we use mini-batch SGD with uniform sampling for each iteration. We can show $x^* = R^{-1}y^*$, where $y^* = \arg\min_{y\in\mathcal{W}'} \|HDUy - HDb\|_2^2$, .

Algorithm 2 HDpwBatchSGD(A,b,x_0,T,r,η,s)

Input: x_0 is the initial point, r is the batch size, η is the fixed step size, s is the sketch size, and T is the iteration number.

1: Compute $R \in \mathbb{R}^{d\times d}$ which makes AR^{-1} an $(O(\sqrt{d}), O(1), 2)$-conditioned basis of A as in Algorithm 1 by using a sketch matrix S with size $s \times n$.
2: Compute HDA and HDb, where HD is a Randomized Hadamard Transform.
3: **for** $t \leftarrow 1, \dots T$ **do**
4: Randomly sample an indices set τ_t of size r, where each index in τ_t is i.i.d uniformly sampled.
5: Denote c_{τ_t} as

$$c_{\tau_t} = \frac{2n}{r}\sum_{j\in\tau_t} (HDA)_j^T \cdot [(HDA)_j x_{t-1} - (HDb)_j]$$
$$= \frac{2n}{r}(HDA)_{\tau_t}^T \cdot [(HDA)_{\tau_t} x_{t-1} - (HDb)_{\tau_t}]$$

6: Let

$$x_t = \arg\min_{x\in\mathcal{W}} \frac{1}{2}\|R(x_{t-1}-x)\|_2^2 + \eta\langle c_{\tau_t}, x\rangle$$
$$= \mathcal{P}_{\mathcal{W}}\left(x_{t-1} - \eta R^{-1}(R^{-1})^T c_{\tau_t}\right)$$

7: **end for**
8: **return** $x_T^{\text{avg}} = \frac{\sum_{i=1}^T x_i}{T}$

4.2. Method of HDpwBatchSGD

The main steps of our algorithm are given in the following **Algorithm 2**.

Note that the way of updating x_t in Algorithm 2 is equivalent to the updating procedure of y_t for the reformulated problem (7) (*i.e.,* set $y_0 = R^{-1}x_0$, then use mini-batch SGD and let $x_t = R^{-1}y_t$). However, there are several benefits if we update x_t in Step 6 directly.

1. Directly updating the term y_t by solving (7) needs additional $O(nd^2)$ time since we have to compute $AR^{-1} = U$, while updating x_{t+1} can avoid that, *i.e.*, it is sufficient to just compute R.

2. In practice, the domain set $\mathcal{W}$ of x is much more regular than the domain set of y, *i.e.,* $\mathcal{W}'$ in (7). Thus, it is easier to solve the optimization problem in Step 6.

In the above algorithm, Step 1 is the same as the first step of pwSGD in [14]. But the later steps are quite different. Particularly, our algorithm does not need to estimate the approximate leverage score of U for computing the sampling probability in each iteration. It uses the much simpler uniform sampling, instead of the weighted sampling. By doing so, we need to compute HDA and HDb, which takes only $O(nd\log n)$ time, is much faster than the $O(nd^2)$ time required for exactly computing the leverage score of U, and costs approximately the same time (*i.e.,* $O(\text{nnz}(A)\log n)$) for computing the approximate leverage score. The output is also different as ours is the

average of $\{x_i\}_{i=1}^T$. Also, we note that in the experiment section, [14] uses the exact leverage score instead of its approximation.

By Theorem 1 and 3, we can get an upper bound on σ^2 and our main result (note that $\sup_{x\in\mathcal{W}} \|Ax-b\|_2^2$ in the result is determined by the structure of the original problem and thus is assumed to be a constant here).

Theorem 4. *Let A be a matrix in $\mathbb{R}^{n\times d}$, r be the batch size and b be a vector in $\mathbb{R}^d$. Let $f(x)$ denote $\|Ax-b\|_2^2$. Then with some fixed step size η in (3), we have*

$$\mathbb{E} f(x_T^{avg}) - f(x^*) \le \frac{3\sqrt{2} D_{\mathcal{W}} \sigma}{\sqrt{rT}}, \tag{8}$$

where $\sigma^2 = O(d\log(n)\sup_{x\in\mathcal{W}}\|Ax-b\|_2^2)$ with high probability. That is, after $T = \Theta(\frac{d\log n}{r\epsilon^2})$ iterations, the output of Algorithm 2 ensures $\mathbb{E}[f(x_T^{avg}) - f(x^*)] \le \epsilon$ with high probability (at least 0.9).

Before we give the proof of Theorem 4, we first provide the following lemma.

Lemma 5. *After two steps of preconditioning as in (6), (7), the following holds with high probability (approximately 0.9) for the stochastic optimization problem: $\min g(y) = E_{i\sim\mathcal{D}'} g_i(y)$, where $g_i(y) = n\|(HDU)_i y - (HDb)_i\|_2^2$, $g(y) = \|HDUy - HDb\|_2^2$, $i\sim\mathcal{D}'$ is uniformly sampled from $\{1,2,\ldots,n\}$, and U is an $(\alpha,\beta,2)$-conditioned basis of A.*

$$\mu \ge \frac{2}{\beta^2}, \tag{9}$$

$$\sup\|(HDU)_i\|_2^2 \le \alpha^2\left(1+\sqrt{8\log 10n}\right)^2, \tag{10}$$

$$\sigma^2 \le 4\alpha^2\left(1+\sqrt{8\log(10n)}\right)^2 \sup_{y\in\mathcal{W}'} g(y) \tag{11}$$

$$= 4\alpha^2\left(1+\sqrt{8\log(10n)}\right)^2 \sup_{x\in\mathcal{W}} \|Ax-b\|_2^2 \tag{12}$$

where the constant 10 comes from Theorem 3 with $c = 10$.

Proof. We know $\mu = 2\sigma_{min}^2(HDU) = 2\sigma_{min}^2(U) \ge \frac{2}{\beta^2}$ which is due to U is an $(\alpha,\beta,2)$-conditioned basis of A. By **Theorem 1**, we know that with probability at least 0.9, the norm of each row of HDU is smaller than $\frac{\alpha}{\sqrt{n}}\left(1+\sqrt{8\log 10n}\right)$. Hence, we have

$$\sup_{\tau_i}\|(HDU)_{\tau_i}\|_2^2 \le \frac{\alpha^2}{n}\left(1+\sqrt{8\log 10n}\right)^2.$$

For $\sigma^2 = \sup_{y\in\mathcal{W}'}\mathbb{E}_{i\sim D}\|\nabla g_i(y) - \nabla g(y)\|_2^2$, we have the following with probability at least 0.9,

$$\begin{aligned}
\sigma^2 &= \sup_{y\in\mathcal{W}'} \mathbb{E}_{j\sim\mathcal{D}}\|2n(HDU)_j^T\cdot\left((HDU)_j y - (HDb)_j\right)\|_2^2 \\
&\quad - \|2(HDU)^T\cdot((HDU)y - HDb)\|_2^2 \\
&= 4n\sum_{j=1}^n \|((HDU)_j^T\cdot\left((HDU)_j y^* - (HDb)_j\right)\|_2^2 \\
&\quad - 4\|(HDU)^T\cdot((HDU)y - HDb)\|_2^2 \\
&\le 4n\sup\|(HDU)_j\|_2^2\cdot\|HDUy^* - HDUb\|_2^2 \\
&\quad - \sigma_{min}(HDU)^2\cdot\|((HDU)y - HDb)\|_2^2 \\
&\le 4n\frac{\alpha^2}{n}\left(1+\sqrt{8\log(10n)}\right)^2 \sup_{y\in\mathcal{W}'} g(y).
\end{aligned}$$

Where the last inequality comes from Theorem 1 and $\sigma_{min}(HDU) = O(1)$. □

Proof of Theorem 4. Consider $\{y_i\}_{i=1}^T$ updated by Lemma 5 with $y_0 = Rx_0$. We first show by mathematical induction that $y_i = Rx_i$ and $y^* = Rx^*$ for all i. Clearly, by the definition of y_i, this is true for $i = 0$. Assume that it is true for $i = k$. In the $(k+1)$-th iteration, we assume that the i-th sample in the k-th iteration is obtained by using SGD. Then, we have (denote $m = \frac{n}{r}$)

$$\begin{aligned}
y_{k+1} &= \arg\min_{y\in\mathcal{W}'} \eta\langle\nabla g_{\tau_k}(y_k), y\rangle + \frac{1}{2}\|y - y_k\|_2^2 \\
&= \arg\min_{y\in\mathcal{W}'} 2m\eta\sum_{j\in\tau_k}((HDU)_j R^{-1}x_k - (HDb)_j)(HDA)_j^T R^{-1} y \\
&\quad + \frac{1}{2}\|y - Rx_t\|_2^2.
\end{aligned}$$

From Steps 5 and 6, we know that

$$\begin{aligned}
x_{k+1} = \arg\min_{x\in\mathcal{W}} 2m\eta\sum_{j\in\tau_k}\left((HDA)_j x_k - (HDb)_j\right)\cdot(HDA)_j x \\
+ \frac{1}{2}\|Rx - Rx_t\|_2^2,
\end{aligned}$$

where $\mathcal{W} = R^{-1}\mathcal{W}'$. From above, we know that $x_{k+1} = R^{-1}y_{k+1}$. This means that $y_i = Rx_i$ is true for all i. Next, by using the variance in the mini-batch SGD, we know the $\sigma_{batch}^2 = \frac{\sigma^2}{r}$, where σ^2 is as in Lemma 5. Then by Theorem 3 we get $\mathbb{E}g(y_T^{\text{avg}}) - g(y^*) \le \frac{3\sqrt{2}D_{\mathcal{W}'}\sigma}{\sqrt{rT}}$, replacing $y_T = Rx_T, y_T^{avg} = Rx_T^{avg}$ and $D_{\mathcal{W}'} = D_{\mathcal{W}}$ (by the definition), we get the proof. □

The time complexity of our algorithm can be easily obtained as

$$\text{time}(R) + O\left(nd\log n + \text{time}_{\text{update}}\times\frac{d\log n}{r\epsilon^2}\right),$$

where time(R) is the time for computing R in Step 1. Different sketch matrices and their time complexities for getting R are shown in **Table 2**. Step 2 takes $O(nd\log n)$ time. $\text{time}_{\text{update}}$ is the time for updating x_{t+1} in Steps 5 and 6. Step 5 takes $O(rd)$ time, while Step 6 takes poly(d) time since it is just a quadratic optimization problem in d dimensions. Thus, if we use SRHT

as the sketching matrix S, the overall time complexity of our algorithm is

$$O\left(nd\log n + d^3\log d\log n + (\text{poly}(d) + rd)\frac{d\log n}{r\epsilon^2}\right). \quad (13)$$

Moreover, in the unconstrained case, the time complexity will be $O\left(nd\log n + \frac{d^2\log n}{\epsilon^2} + \frac{d^3\log n}{r\epsilon^2}\right)$. Comparing with the state-of-the-art result, *i.e.,* pwSGD [14], which has the time complexity $O\left(nd\log n + d^3\log d\log n + \frac{\text{poly}(d)\log(\frac{1}{\epsilon})}{\epsilon}\right)$, ours is much faster in the case of $O(\frac{\log n\text{poly}(d)}{\epsilon}) \le r$ and $\text{poly}(d) \ge O(\frac{\log n}{\epsilon})$. However, as we will see, our method always outperform the batch version of pwSGD with some batchsize in practice, see experiment section for details.

Table 2: Time complexity for computing R in Algorithm 1 with different sketch matrices [14].

Sketch Matrix	Time Complexity	$\kappa(AR^{-1})$
Gaussian Matrix	$O\left(nd^2\right)$	$O(1)$
SRHT [18]	$O\left(nd\log d + d^3\log d\log n\right)$	$O(1)$
CountSketch [31]	$O\left(\text{nnz}(A) + d^4\right)$	$O(1)$
Sparse l_2 Embedding [32]	$O\left(\text{nnz}(A)\log d + d^3\log d\right)$	$O(1)$

4.3. Further Reducing the Time Complexity

Although Theorem 4 has the advantage of optimality on the batchsize, it does not make use of the properties of $O(1)$-strongly convexity and condition number $\frac{L}{\mu} = O(1)$ of the problem after the two-step preconditioning Eq (7). Actually, we can apply a different first-order method to achieve an ϵ-error in $\Theta(\frac{d\log n}{r\epsilon} + \log(\frac{1}{\epsilon}))$ iterations, instead of $\Theta(\frac{d\log n}{r\epsilon^2})$ iterations as in Theorem 4. The preconditioning steps are the same, and the optimization method is the multi-epoch stochastic accelerated gradient descent, which was proposed in [33; 22] (see **Algorithm 3**). In stead of using Theorem 3, we will use the following lemma whose proof was given in [22].

Lemma 6 ([22])**.** *If Assumptions 1 and 2 hold and $\epsilon < V_0$, then after $O\left(\sqrt{\frac{L}{\mu}}\log(\frac{V_0}{\epsilon}) + \frac{\sigma^2}{\mu\epsilon}\right)$ iterations of stochastic accelerated gradient descent with $O\left(\log(\frac{V_0}{\epsilon})\right)$ epochs, the output of multi-epoch stochastic accelerated gradient descent p_S in Algorithm 3 satisfies $\mathbb{E}F(p_S) - F(x^*) \le \epsilon$, where V_0 is a given bound such that $F(x_0) - F(x^*) \le V_0$.*

Thus, we can use a two-step preconditioning and multi-epoch stochastic accelerated mini-batch gradient descent to obtain an algorithm (called HDpwBatchAccSGD) similar to **Algorithm 2**, as well as the following theorem, see Algorithms 3 and 4 for details. We have the following Theorem, whose proof is similar to it for Theorem 4.

Algorithm 3 Multi-epoch Stochastic Accelerated Gradient Descent [30]

Input: $p_0 \in \mathcal{W}$ is the initial point, and a bound V_0 such that $F(x_0) - F(x^*) \le V_0$ is given, S is the epoch number.

1: **for** $s = 0, 1, \cdots S$ **do**

2: Run N_s iterations of Stochastic Accelerated Gradient method with $x_0 = p_{s-1}, \alpha_t = \frac{2}{t+1}, q_t = \alpha_t$, and $\eta_t = \eta_s t$, where

$$N_s = \max\left\{4\sqrt{\frac{2L}{\mu}}, \frac{64\sigma^2}{3\mu V_0 2^{-s}}\right\},$$

$$\eta_s = \min\left\{\frac{1}{4L}, \sqrt{\frac{3V_0 2^{-(s-1)}}{2\mu\sigma^2 N_s(N_s+1)^2}}\right\}$$

3: Set $p_s = \hat{x}_{N_s}$, where $\hat{x}_{N_s}$ is get from step 1.

4: **end for**

5: **return** p_s

Theorem 7. *Let A be a matrix in $\mathbb{R}^{n\times d}$, r be the batch size and b be a vector in $\mathbb{R}^d$. Let $f(x)$ denote $\|Ax - b\|_2^2$, and $\epsilon < V_0$ be a fixed number. Then, after $O(\log(\frac{V_0}{\epsilon}) + \frac{d\log n}{r\epsilon})$ iterations of stochastic accelerated gradient descent with $S = O(\log(\frac{V_0}{\epsilon}))$ epochs of HDpwBatchAccSGD, the output p_S satisfies the following inequality with high probability*

$$\mathbb{E}F(p_S) - F(x^*) \le \epsilon.$$

Moreover, if using SRHT as the sketching matrix, the total time complexity is

$$O(nd\log n + d^3\log d\log n + \frac{d^2\log n}{\epsilon} + \frac{poly(d)\log n}{r\epsilon} + rpoly(d)\log\frac{1}{\epsilon}). \quad (14)$$

Also, if the batchsize r satisfies $r^2 = O\left(\frac{poly(d)\log n}{\epsilon\log(1/\epsilon)}\right)$, the time complexity becomes

$$O\left(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{poly(d)\sqrt{\log n\log(\frac{1}{\epsilon})}}{\sqrt{\epsilon}}\right),$$

which is less than (13), and less than pwSGD if $O\left(\frac{\log(\frac{1}{\epsilon})}{\epsilon}\right) \ge \log n$.

5. High Precision Case: Improved Iterative Hessian Sketch

Now, we go back to our results on time complexity in (13) and (14). One benefit of these results is that the ϵ term in the time complexity is independent of n and depends only on $\text{poly}(d)$ and $\log n$. Thus, if directly using the Variance Reduced methods developed in recent years (such as [3]) to the constrained linear regression problem, we can obtain a time complexity of $O\left((n+\kappa)\text{poly}(d)\log\frac{1}{\epsilon}\right)$, where the ϵ term is $\log(\frac{1}{\epsilon})$

Algorithm 4 HDpwBatchAccSGD(A,b,x_0,r,s, V_0,S)

Input: $\tilde{x}_0 = \hat{X}_0 = x_0$ are the initial points, r is the batch size, η is the fixed step size, s is the sketch size, S is the number of epochs and bound V_0 satisfies $F(x_0) \leq V_0$.

1: Compute $R \in \mathbb{R}^{d\times d}$ which makes AR^{-1} an $O(\sqrt{d}), O(1), 2)$-conditioned basis of A as in Algorithm 1 by using a sketch matrix S with size $s \times n$.
2: Compute HDA and HDb, where HD is a Randomized Hadamard Transform.
3: Run Multi-epoch Stochastic Accelerated Gradient Descent (Algorithm 3) with x_0, V_0, $L = O(1)$,$\mu = O(1)$ and with S epochs, where the step of Stochastic Accelerated Gradient Descent is as following:
4: Randomly sample an indices set τ_t of size r, where each index in τ_t is i.i.d uniformly sampled.
5: Updata as the followings:

$$c_{\tau_t} = 2\frac{n}{r}(HDA)_{\tau_t}^T \cdot [(HDA)_{\tau_t}x_{t-1} - (HDb)_{\tau_t}],$$
$$\tilde{x}_t = (1-q_t)\hat{x}_{t-1} + q_t x_{t-1},$$
$$x_t = \arg\min_{x\in\mathcal{W}} \left\{ \eta_t[\langle c_{\tau_t}, x\rangle + \frac{\mu}{2}\|R(\tilde{x}_t - x)\|_2^2] + \frac{1}{2}\|R(x - x_{t-1})\|_2^2 \right\}$$
$$\hat{x}_t = (1-\alpha_t)\hat{x}_{t-1} + \alpha_t x_t$$

return p_s

instead of poly($\frac{1}{\epsilon}$) and κ is the condition number of A. Comparing with the ϵ term in these methods, we know that HDpwBatchSGD and HDpwBatchAccSGD are more suitable for the **low precision and large scale** case.

Recently, [17] introduced the Iterative Hessian Sketch (IHS) method to solve the large scale constrained linear regression problem (see Algorithm 5). IHS is capable of achieving high precision, but needs a sequence of sketch matrices $\{S^t\}_{t=1}^T$ (which seems to be unavoidable due to their analysis) to ensure the linear convergence with high probability. Ideally, if we could use just one sketch matrix, it would greatly reduce the running time. The need of using a less number of sketch matrices do arise in applications. For example, it is possible that matrix A is too large in size (*e.g.,* in large scale image (like CT image) reconstruction problems) and may not be able to fit into memory or even need to be constructed on the fly due to its prohibited size [28]; in such scenarios, reducing the number of sketch matrices could significantly lower the number of I/O operations and thus considerably speed up the computation. In this section, we show that by adopting our two-step preconditioning strategy, it is indeed possible to use only one sketch matrix in IHS to achieve the desired linear convergence with high probability, see pwGradient (**Algorithm 6**).

Our pwGradient method uses the first step of preconditioning (*i.e.,* Step 1 of Algorithm 2) and then performs (projected) gradient decent (GD) method directly, instead of the mini-batch SGD in Algorithm 2 (note that we do not need the second step of preconditioning since the matrix HD is an orthogonal matrix). Since the condition number after preconditioning is $O(1)$ (see **Table 2**), by the convergence rate of GD, we know that only $O(\log\frac{1}{\epsilon})$ iterations are needed to attain an ϵ-solution. Formally, we have the following theorem.

Algorithm 5 IHS(A,b,x_0,s, T) [1]

Input: x_0 is the initial point, T is the iteration number and s is the sketch size.

1: **for** $t = 0, 1, \cdots, T-1$ **do**
2: Generate an independent sketch matrix $S^{t+1} \in \mathbb{R}^{s\times n}$ as in Algorithm 1. Compute the matrix $M = S^{t+1}A$.
3: Perform the updating

$$x_{t+1} = \arg\min_{x\in\mathcal{W}} \frac{1}{2}\|M(x - x_t)\|_2^2 + \langle A^T(Ax_t - b), x\rangle$$
$$= \mathcal{P}_{\mathcal{W}}\left(x_t - M^{-1}(M^{-1})^T A^T(Ax_t - b)\right)$$

4: **end for**
5: **return** x_T

Algorithm 6 pwGradient(A,b,x_0,s, η, T)

Input: x_0 is the initial point, s is the sketch size, T is the iteration number and η is the step size.

1: Compute $R \in \mathbb{R}^{d\times d}$ which makes AR^{-1} an $O(\sqrt{d}), O(1), 2)$-conditioned basis of A as in Algorithm 1 by using a sketch matrix S with size $s \times n$.
2: **for** $t = 0, 1, \cdots, T-1$ **do**
3: Perform the updating

$$x_{t+1} = \arg\min_{x\in\mathcal{W}} \frac{1}{2}\|R(x - x_t)\|_2^2 + \eta\langle 2A^T(Ax_t - b), x\rangle$$
$$= \mathcal{P}_{\mathcal{W}}\left(x_t - 2\eta R^{-1}(R^{-1})^T A^T(Ax_t - b)\right).$$

4: **end for**
5: **return** x_T

Theorem 8. *Let $f(x) = \|Ax - b\|_2^2$. Then, for some step size $\eta = O(1)$ in pwGradient, the following holds,*

$$f(x_t) - f(x^*) \leq (1 - O(1))^t (f(x_0) - f(x^*)) \tag{15}$$

with high probability.

One major advantage of our method is that the time complexity is much lower than that of IHS, since it needs only one step of sketching. For example, in the unconstrained case, if SRHT is used as the sketch matrix, the complexity of our method becomes

$$O\left(nd\log n + d^3\log d\log n + (nd + d^3)\log\frac{1}{\epsilon}\right), \tag{16}$$

while the time complexity of IHS with SHRT sketch matrix is $O\left(nd\log d\log\frac{1}{\epsilon} + d^3\log\frac{1}{\epsilon}\right)$. Thus, our method is always better than IHS if ϵ satisfies the condition of $\epsilon \leq \frac{1}{n}$, *i.e.,* when the error

is small as in the high precision large scale case. Also, it is notable that if we use other sketch matrics, such as CountSketch, the time complexity will be

$$O\left(nnz(A) + d^4 + (nd + d^3)\log\frac{1}{\epsilon}\right),$$

which is always less than that in IHS. We will verify this in the experiment section.

Proof. Similar to the proof of Theorem 4, we can show that the updating step is just performing the projected gradient descent operations on $\|AR^{-1}y - b\|_2^2$ and thus $x_{t+1} = R^{-1}y_{t+1}$. Since the condition number of $U = AR^{-1}$ is $O(1)$, by the convergence rate of gradient descent on strongly convex functions [34] and with step size $\eta = O(1)$, we know that

$$\begin{aligned} f(x_t) - f(x^*) &= \|Uy_t - b\|_2^2 - \|Uy^* - b\|_2^2 \\ &\leq \frac{2\sigma_{\max}^2(U)}{2}(1 - O(1))^{2k}\|y_0 - y^*\|_2^2. \end{aligned}$$

By the strongly convexity property, we know that

$$\frac{2\sigma_{\min}^2(U)}{2}\|y_0 - y^*\|_2^2 \leq \|Uy_0 - b\|_2^2 - \|Uy^* - b\|_2^2 = f(x_0) - f(x^*).$$

Also, by $\kappa(U) = O(1)$ (see Table 2). Thus, we have the theorem. □

Below we reveal the relationship between IHS and pwGradient. Particularly, we will show that when $\eta = \frac{1}{2}$, the updating in pwGradient with sketching matrix S has the same form as that in IHS with $\{S^t\}_{t=0}^{T-1} = S$. This is due to the fact that if we let QR be the QR-decomposition of $S^{t+1}A = SA$, we have

$$(QR)^{-1}\left((QR)^{-1}\right)^T = (R)^{-1}(R^{-1})^T,$$

which is due to the fact that matrix Q is an orthogonal matrix.

Although they look alike, there are still some differences.

1. The ideas behind them are quite different. IHS is based on sketching the Hessian in each iteration and uses the second order methods of the optimization problem, while ours is based on preconditioning the original problem and uses the first-order methods. Thus, we need an additional step size η, while IHS does not require it. As we can see from the above discussion and experiments, $\eta = \frac{1}{2}$ is sufficient for our algorithm pwGradient to achieve good performance.

2. One advantage of IHS is that it can explore the underlying geometric structure of the constraint set $\mathcal{W} \subseteq \mathbb{R}^d$. Since it uses the sketching idea to solve the optimization problem

$$\min_{x\in\mathcal{W}} \frac{1}{2n}\|A(x - x_t)\|_2^2 - x^T A^T(b - Ax_t)$$

in each iteration, that is, to solve the sketched optimization problem

$$\min_{x\in\mathcal{W}} \frac{1}{2s}\|S^t A(x - x_t)\|_2^2 - x^T A^T(b - Ax_t).$$

By using the sharpness result of the sketch size in [1; 17], the authors show that the sketch size of S depends only on the Gaussian Width of $\mathcal{W}$ (see [17] for details). However, since the sketch matrix in our paper is to construct a well-conditioned matrix, the size will not be depending on the Gaussian width. Thus, the sketch size s in our method is theoretically different from that of IHS. It is possible that IHS could have a smaller size s. However, as it will be shown in experiments, our method is faster even if IHS has a smaller sketch size.

6. Numerical Experiments

In this section we present some experimental results on our proposed methods. We will mainly focus on the iteration complexity and running time. Experiments confirm that our proposed algorithms are indeed faster than those existing ones. The algorithms are implemented using CountSketch as the sketch matrix $S \in \mathbb{R}^{s\times n}$ in the step for computing R^{-1} due to its fast constructing time. The Matlab code of CountSketch can be found in [35]. Below is a rephrase of our methods.

- **HDpwBatch**, *i.e.* ***Algorithm 2***. We use the optimal step size as described in Theorem 3 (note that we assume that the step size is already known in advance).
- **HDpwBatchAcc**, *i.e.* ***Algorithm 3 and 4***. We use the way of choosing step size in tfocs[1].
- **pwGradient**, *i.e.* ***Algorithm 6***. As we discussed in previous section, setting $\eta = \frac{1}{2}$ as the step size is enough.

6.1. Baseline of Experiments

Table 3: Summary of Datasets used in the experiments.

Dataset	Rows	Columns	$\kappa(A)$	Sketch Size
Syn1	10^5	20	10^8	1000
Syn2	10^5	20	1000	1000
Syn3	10^6	40	10^5	4000
Syn4	10^6	10	10^5	20000
Syn5	5×10^6	50	10^5	20000
IJCNN	91701	22	6.9	2000
Year	5×10^5	90	3000	20000
Covertype(short)	5.8×10^4	8	27.2	10000
Buzz	5×10^5	77	10^8	20000
HT_Sensor	9×10^5	12	277	20000
SUSY	5×10^6	19	84	60000
Gas_methane	4.1×10^6	19	7600	40000
Gas_CO	4.1×10^6	19	8100	40000
HEPMASS	7×10^6	29	6.77	60000

[1] http://cvxr.com/tfocs/

Algorithm 7 pwSVRG(A,b,x_0,s, η, T, m, r)

Input: x_0 is the initial point, r is the batch size, s is the sketch size, T is the outer iteration number, m is the inner iteration number and η is the step size.

1: Compute $R \in \mathbb{R}^{d\times d}$ which makes AR^{-1} an $O(\sqrt{d}), O(1), 2)$-conditioned basis of A as in Algorithm 1 by using a sketch matrix S with size $s \times n$.
2: Compute HDA and HDb, where HD is a Randomized Hadamard Transform.
3: Let $x_0^0 = x_0$.
4: **for** $t = 0, 1, \cdots, T-1$ **do**
5: Let $g_t = 2A^T\|Ax_0^{t+1} - b\|_2$.
6: **for** $s = 0, 1, \cdots, m-1$ **do**
7: Randomly sample an indices set τ_s of size r, where each index in τ_s is i.i.d uniformly sampled.
8: Let

$$c_{\tau_s}^t = \frac{2n}{r}(HDA)_{\tau_s}^T \cdot [(HDA)_{\tau_s}x_s^t - (HDb)_{\tau_s}]$$

$$c_{\tau_0}^t = \frac{2n}{r}(HDA)_{\tau_s}^T \cdot [(HDA)_{\tau_s}x_0^t - (HDb)_{\tau_s}].$$

9: Perform the updating

$$x_{s+1}^t = \mathcal{P}_{\mathcal{W}}\left(x_s - \eta R^{-1}(R^{-1})^T(c_{\tau_s}^t - c_{\tau_0}^t + g_t)\right).$$

10: **end for**
11: Set $x_0^{t+1} = x_m^t$
12: **end for**
13: **return** x_0^T

In both the low and high precision cases, we select some widely recognized algorithms with guaranteed time complexities for comparisons. We measure the performance of methods by the wall-clock time or iteration number. For each experiment, the initial value x_0 is set to be the zero vector and we test every method 10 times and take the average as the final results.

For the low precision case, we choose **pwSGD** [14], which has the best known time complexity, and use the optimal step size of their method. We need to note that the theoretical guarantee of pwSGD is for SGD with batchsize $r = 1$. Here we will use the the mini-batch version of pwSGD, which has also been studied in [14] in the experiments. Thus, we call the method as **pwBatchSGD**. We choose **SGD** and **Adagrad** as the standard first order method for comparisons (the code for SGD and Adagrad can be found in [36]), and use the batch version of both methods.

For the high precision case, besides **IHS**, we also use a method called **pwSVRG** for comparison, which uses our two-step preconditioning first and then performs SVRG with different batch sizes (the related method can be found in [15]), see Algorithm 7. We use pwSVRG, instead of SVRG, due to the fact that the condition number of the considered datasets are very high, which means that directly using SVRG or related methods could lead to rather poor performance; thus we do not use them for comparison (although [28] used SAGA for comparison, it was done after normalizing the datasets).

The y-axis of each plot is the relative error $\frac{\|Ax_t-b\|_2^2-\|Ax^*-b\|_2^2}{\|Ax^*-b\|_2^2}$ in the low precision case and the log relative error $\log(\frac{\|Ax_t-b\|_2^2-\|Ax^*-b\|_2^2}{\|Ax^*-b\|_2^2})$ in the high precision case. Table 3 is a summary of the datasets and the recommended sketch size for our methods. The real world datasets come from [37] and [38]. For IHS, we first run it with $\frac{3}{4}$ and $\frac{1}{2}$ of the recommended sketch size and then select the best one. The sketch size of IHS can be theoretically smaller than ours, as discussed in the previous section. We note that selecting an appropriate sketch size is very important, as it will be shown in the discussion section that the methods may perform poorly if the size is too small and are costly if it is too large.

The synthetic datasets are generated as follows. We first generate a Gaussian vector x^* as the response vector and let $b = Ax^* + e$, where e is a Gaussian noise with standard variance of 0.1. The design matrix A is of the form $A = U\Sigma V^T$, where $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{d\times d}$ are random orthogonal matrices and $\Sigma \in \mathbb{R}^{d\times d}$ controls the condition number $\kappa(A)$.

We consider both the constrained and unconstrained cases with ℓ_1 and ℓ_2 norm ball as the constraints. For the constrained case, we first generate an optimal solution for the unconstrained case, and then set it as the radius of the balls. Note that such a setting also appears in existing work [14] and [28].

We run all the numerical experiments on a Macbook Pro with 2.3 GHz Intel Core i5 and 8 GB RAM, MATLAB version R2018a.

6.2. Results For Low Precision Case

6.2.1. Synthetic datasets

Figure 1 shows the iteration complexity for achieving a given relative error ϵ with different batch sizes in HDpwBatchSGD. As we can see from Figure 1, the iteration number decreases as the batch size increasing. Specifically, we can see that the iteration number decreases by a factor of 2 if the batch size increases by the same factor, which supports our analysis of HDpwBatchSGD, *i.e.* Theorem 4.

Figure 2, 3, 4 show the comparisons with other existing low precision methods on the synthetic datasets with or without the ℓ_1 and ℓ_2 norm balls as the constraints. The figures indicate that with the same batch size r, the HDpwBatchAccSGD method has the shortest running time among all the methods. Also, when the batch size becomes larger, the HDpwBatchSGD method takes less time to achieve the target relative error. Compared with the state-of-art result pwSGD, our methods are much faster. All these advantages of our method have already been predicted in Theorem 4 and 8.

6.2.2. Real world datasets

For the low precision case, we examine our methods on the IJCNN, Year and Covertype datasets. Note that for the Covertype dataset, we use the first 8 columns as the matrix A and the 9-th column as the response vector b. The results are shown in Figure 5, 6 and 7, respectively. From the figures, we can see that most of the findings from the synthetic datasets still hold

for the real datasets. We notice that there are some cases where pwbatchSGD can outperform our method (such as in Figure 5). This is due to the fact that when the batchsize is too small, the time complexity of our method is larger, as mentioned in the previous section.

To summarize, we list the time when the relative error is less than 10^{-3} for different low precision methods on different datasets. See Table 4 for details.

6.3. Results For High Precision Case

6.3.1. Synthetic datasets

For the high precision case, we test our methods on Syn3, Syn4 and Syn5, and plot the results in Figure 8, 9 and 10, respectively. From the result of the unconstrained case (Figure 8), we can see that our method pwGradient outperforms IHS, and is almost $2\times$ faster, which supports the theoretical analysis given in Theorem 8.

6.3.2. Real world datasets

We also evaluate our high precision methods on Year, Buzz, HT_Sensor, SUSY, Gas_Sensor and HEPMASS datasets. Figure 11, 12 and 13 are the results for Year, Buzz, and HT_Sensor, respectively. Figure 14, 15 and 16 show the result of other datasets. From the results, we can see that all findings from the synthetic datasets are still true for the real world datasets. We also note that for the datasets Gas_Sensor_methane, Gas_Sensor_CO and HEPMASS, the method pwSVRG does not perform well in the case constrained by the ℓ_1 and ℓ_2 norm balls.

To summarize, we list the time when the relative error is less than 10^{-10} for different high precision methods on different datasets. See Table 5 for details.

From the experimental results, we can see that they all support our theoretical claims.

7. Conclusion and Discussion

In this paper, we studied the large scale constrained linear regression problem, and presented new methods for both the low and high precision cases, using some recent developments in random projection, sketching and optimization. For the low precision case, our proposed methods have lower time complexity than the state-of-the-art technique. For the high precision case, our method considerably improves the time complexity of the Iterative Hessian Sketch method. Experiments on synthetic and benchmark datasets confirm that our methods indeed run much faster than the existing ones.

There are still some open problem left and we leave them as future research.

1. From a theoretical point of view, one open problem is to determine whether there exists even faster algorithms for the problem. Also, it would be interesting to establish lower bounds on the time complexity for achieving a solution with ϵ relative error for both low and high precision cases.

2. Our methods need to perform a convex set projection operation in each iteration. However, as shown in [39], projection onto some convex set could be quite costly in many scenarios. To resolve such an issue, a possible approach is to combine our two-step preconditioning method with the Frank-wolfe method. However, it is known that there are still cases where even Frank-wolfe methods are not applicable. To deal with this problem, recently [40; 41] considered the first order methods which use fewer or only one projection step under some mild assumptions on the constrained set. Thus, another possible approach is to combine our methods with these techniques. Also, it would be interesting to consider the case where the constraint set is non-convex, such as ℓ_0-norm ball.

3. From a practical point of view, although our methods perform well, there are still some issues for further improvement. Firstly, in the low precision case, we use the optimal stepsize in our algorithms, which is often unknown in practice. Thus, it is important to find a way to choose an appropriate stepsize. Secondly, both HDpwBatchSGD and HDpwBatchAccSGD need to choose a proper batchsize for optimal performance. It is still unknown how to choose the best batchsize. Thirdly, our methods need a systematic way to choose the sketch size. To determine such a value, we tested HDpwBatch, HDpwBatchAcc and pwGradient on the Year and Syn3 datasets with different sketch sizes for the unconstrained case (see Figure 17 and 18). From the experiments, we can see that when the sketch size is too small, these methods perform poorly, which is due to the failure of constructing a well-conditioned basis (Definition 1). However, if the sketch size is too large, the method will be very costly (especially for the high precision case). Thus, a better strategy is needed to determine the proper sketch size.

8. Acknowledgments

This research was supported in part by National Science Foundation (NSF) through grants IIS-1422591, CCF-1422324, and CCF-1716400. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] M. Pilanci, M. J. Wainwright, Randomized sketches of convex programs with sharp guarantees, IEEE Transactions on Information Theory 61 (9) (2015) 5096–5115.

[2] S. Shalev-Shwartz, T. Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, Journal of Machine Learning Research 14 (Feb) (2013) 567–599.

[3] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Advances in Neural Information Processing Systems, 2013, pp. 315–323.

[4] Z. Allen-Zhu, Katyusha: The first direct acceleration of stochastic gradient methods, in: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2017, pp. 1200–1205.

Dataset	Constraint	HDpwBatchAcc	HDpwBatch $r = 4000$	HDpwBatch $r = 2000$	pwBatch $r = 4000$	Adagrad $r = 4000$
Syn1	Unconstrained	**2.21**	2.43	3.47	2.45	–
	ℓ_1 norm	2.06	2.57	1.83	4.05	–
	ℓ_2 norm	**0.83**	1.06	2.36	5.52	–
Syn2	Unconstrained	**1.13**	1.21	1.32	1.54	–
	ℓ_1 norm	**3.05**	3.13	4.47	5.82	–
	ℓ_2 norm	**0.86**	1.02	2.01	3.51	–
Syn3	Unconstrained	**38.7**	49.1	56.3	75.8	–
	ℓ_1 norm	**19.8**	19.9	24.7	45.4	–
	ℓ_2 norm	**15.31**	15.38	24.7	37.6	–
IJCNN	Unconstrained	1.17	1.15	1.46	1.26	–
	ℓ_1 norm	**2.63**	4.15	4.17	5.44	–
	ℓ_2 norm	4.56	–	4.50	–	–
Year	Unconstrained	33.5	49.3	–	47.1	–
	ℓ_1 norm	**24.2**	25.3	28.8	30.3	–
	ℓ_2 norm	–	–	–	40.0	–
Covertype(short)	Unconstrained	6.18	7.43	–	7.46	–
	ℓ_1 norm	**9.03**	9.55	9.53	–	–
	ℓ_2 norm	**3.95**	4.13	5.08	–	–

Table 4: Comparison of different low precision methods by using the time (/s) when the relative error is less than 10^{-3}.

Dataset	Constraint	pwGradient	IHT	pwSVRG $r = 2000$	pwSVRG $r = 4000$	pwSVRG $r = 1000$
Syn3	Unconstrained	**0.93**	1.92	7.40	7.44	7.42
	ℓ_1 norm	**1.61**	1.65	10.1	25.6	9.73
	ℓ_2 norm	**2.42**	2.43	–	–	–
Syn4	Unconstrained	**4.22**	11.6	–	–	–
	ℓ_1 norm	**3.26**	3.57	23.8	27.6	20.4
	ℓ_2 norm	**2.45**	4.34	–	29.4	24.2
Syn5	Unconstrained	**7.23**	16.9	42.5	46.1	36.3
	ℓ_1 norm	**9.72**	10.2	33.4	36.8	29.6
	ℓ_2 norm	8.46	8.41	–	43.2	27.2
Year	Unconstrained	**0.71**	1.32	7.21	9.11	8.06
	ℓ_1 norm	**1.13**	1.21	9.71	11.6	8.63
	ℓ_2 norm	**1.04**	1.08	10.8	11.3	8.53
Buzz	Unconstrained	**0.56**	0.95	6.34	6.81	5.39
	ℓ_1 norm	**0.974**	0.976	7.80	9.36	6.34
	ℓ_2 norm	**1.35**	0.20	12.5	16.6	15.2
HT Sensor	Unconstrained	0.26	0.24	1.25	1.37	1.14
	ℓ_1 norm	**0.27**	0.38	1.61	1.92	1.70
	ℓ_2 norm	**0.42**	0.46	1.37	1.82	1.96
SUSY	Unconstrained	**0.91**	1.67	12.4	16.2	14.8
	ℓ_1 norm	**0.19**	0.24	11.3	11.3	11.2
	ℓ_2 norm	**2.16**	2.71	11.7	12.7	12.7
Gas Sensor methane	Unconstrained	**0.16**	0.30	17. 1	17.0	18.3
	ℓ_1 norm	**0.30**	0.33	–	–	–
	ℓ_2 norm	**3.85**	5.02	–	–	–
Gas Sensor CO	Unconstrained	**0.15**	0.27	14.0	14.0	14.0
	ℓ_1 norm	2.95	2.47	–	–	–
	ℓ_2 norm	**3.22**	4.27	–	–	–
HEPMASS	Unconstrained	**2.67**	2.54	32.3	23.4	27.1
	ℓ_1 norm	**4.38**	5.07	–	–	–
	ℓ_2 norm	**4.17**	5.08	–	–	–

Table 5: Comparison of different high precision methods by using the time (/s) when the relative error is less than 10^{-10}.

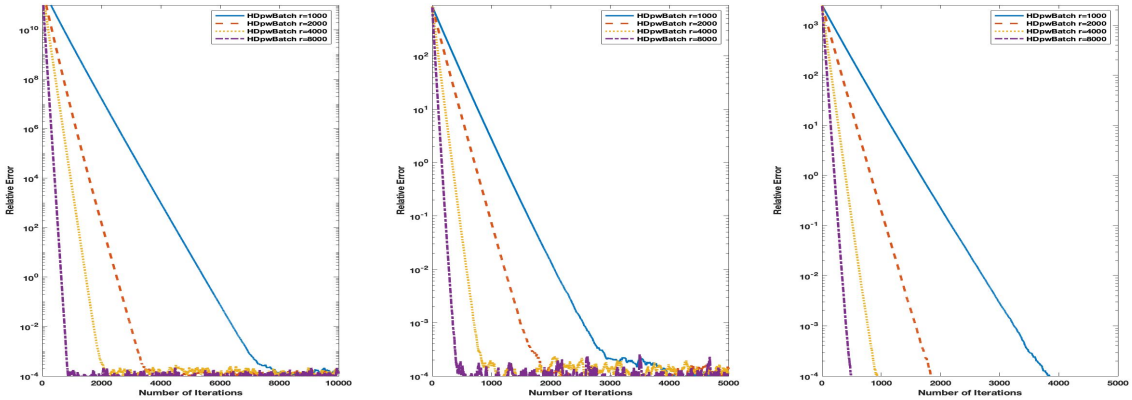

Figure 1: Iteration number of HDpwBatchSGD with different batch size r on (from left to right) datasets Syn1, Syn2 and Syn3 (unconstrained case).

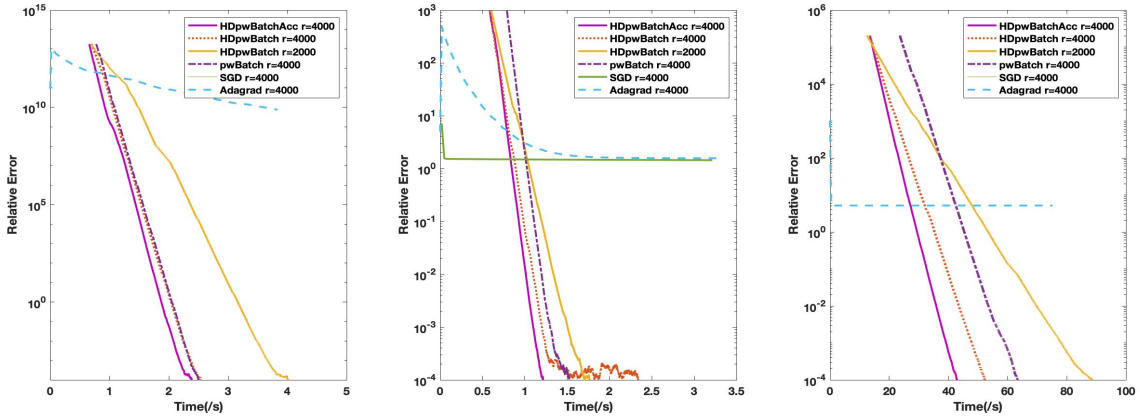

Figure 2: Experimental results on synthetic datasets for the unconstrained low precision case. From left to right is for Syn1, Syn2 and Syn3, respectively.

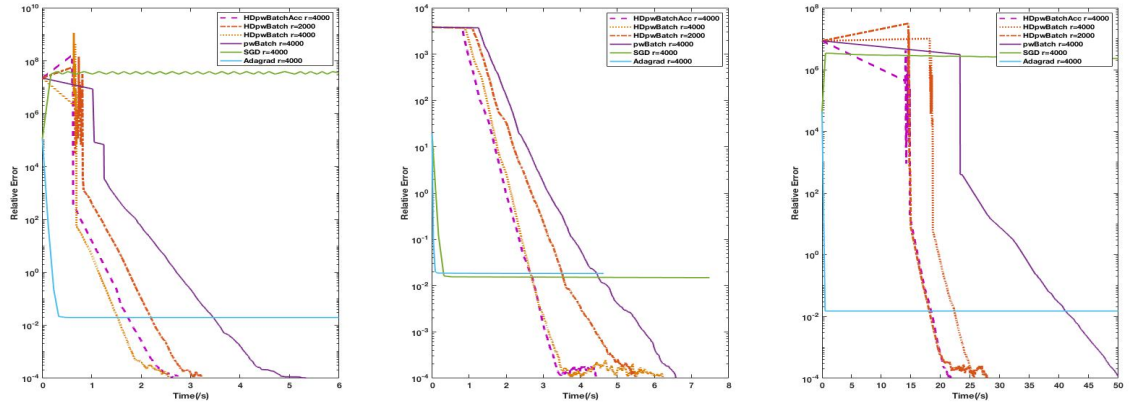

Figure 3: Experimental results on synthetic datasets for low precision case with ℓ_1 norm constraint. From left to right is for Syn1, Syn2 and Syn3, respectively.

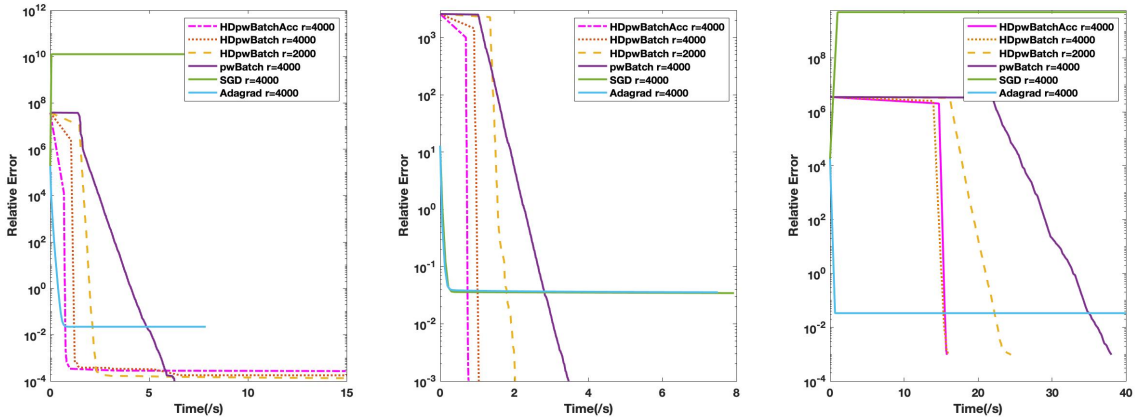

Figure 4: Experimental results on synthetic datasets for low precision case with ℓ_2 norm constraint. From left to right is for Syn1, Syn2 and Syn3, respectively.

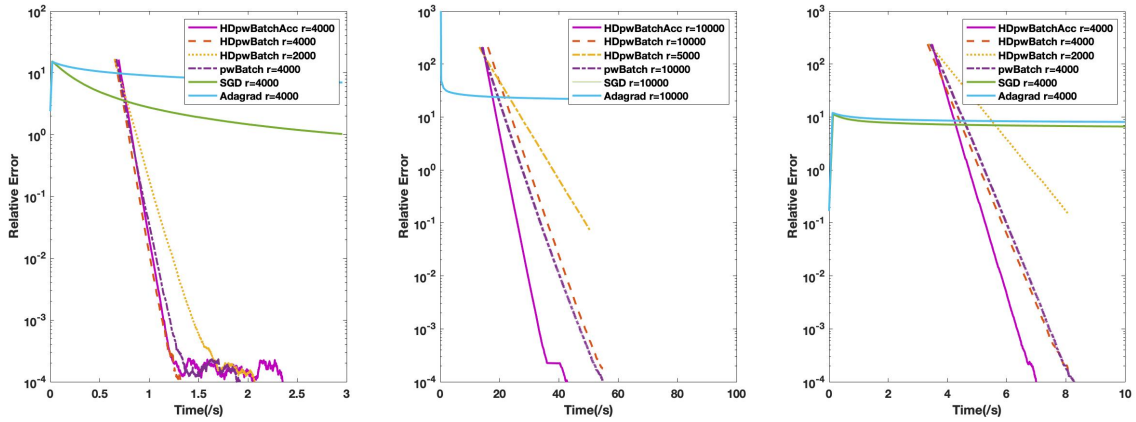

Figure 5: Experimental results on real datasets for the unconstrained low precision case. From left to right is for IJCNN, Year and Covertype(short), respectively.

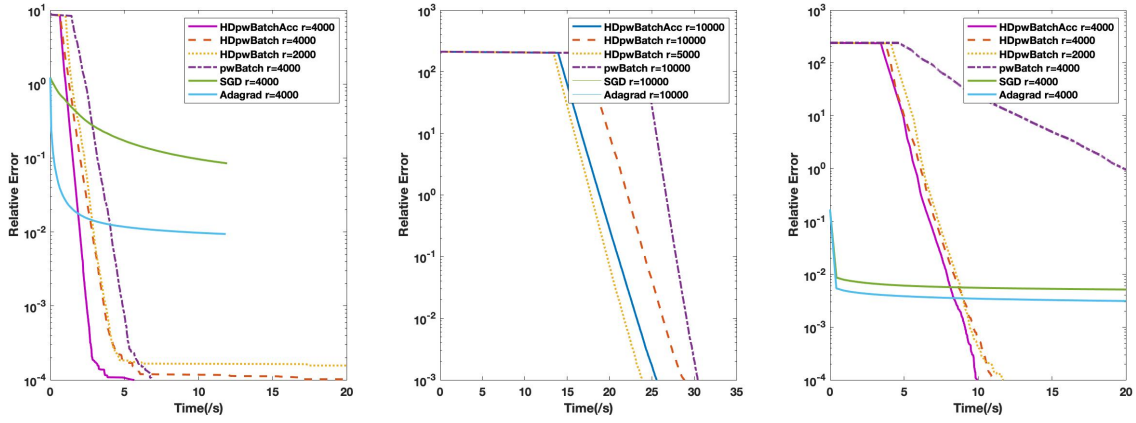

Figure 6: Experimental results on real datasets for low precision case with ℓ_1 norm constraint. From left to right is for IJCNN, Year and Covertype(short), respectively.

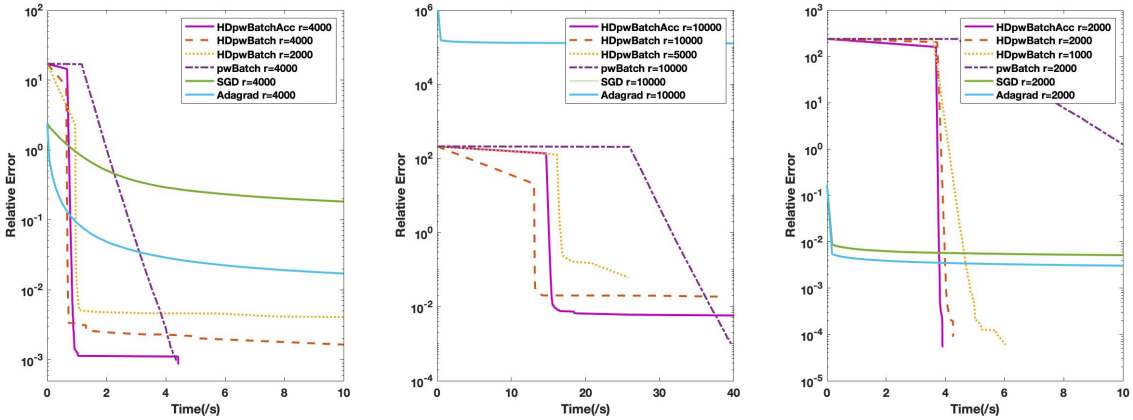

Figure 7: Experimental results on synthetic datasets for low precision case with ℓ_2 norm constraint. From left to right is for IJCNN, Year and Covertype(short), respectively.

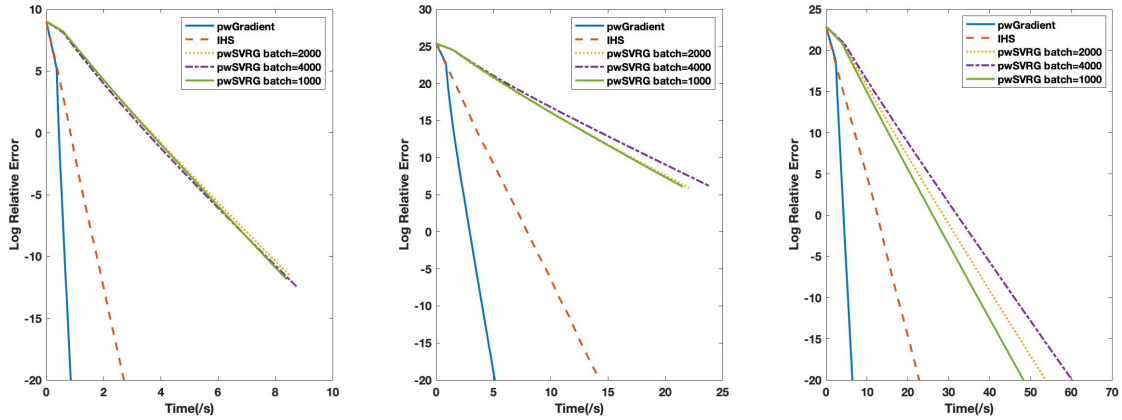

Figure 8: Experimental results on synthetic datasets for the unconstrained high precision case. From left to right is for Syn3, Syn4 and Syn5, respectively.

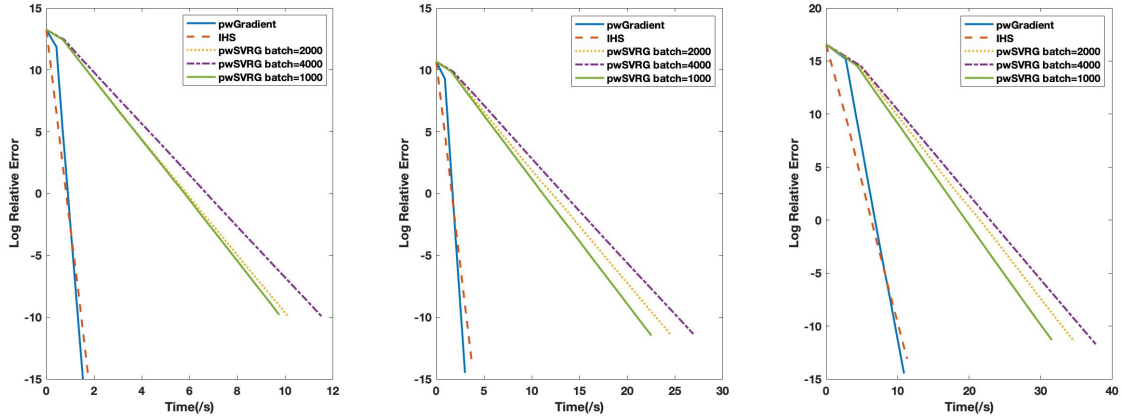

Figure 9: Experimental results on synthetic datasets for high precision case with ℓ_1 norm constraint. From left to right is for Syn3, Syn4 and Syn5, respectively.

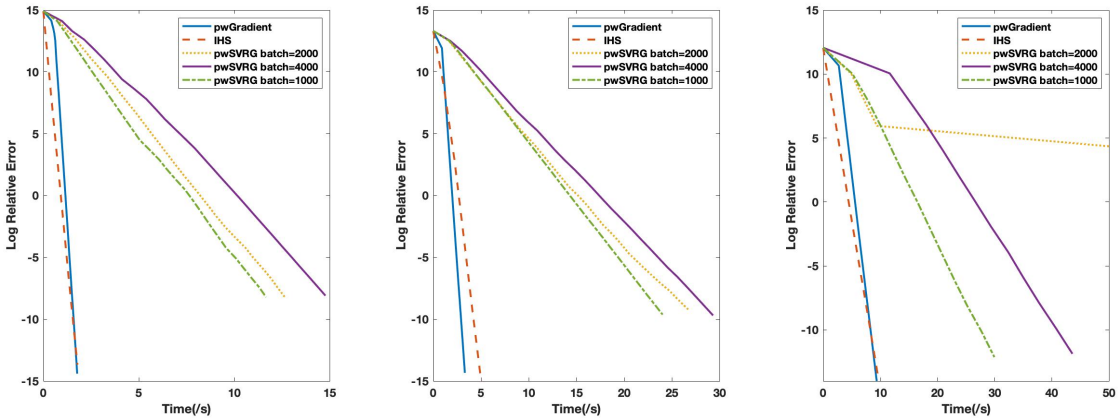

Figure 10: Experimental results on synthetic datasets for high precision case with ℓ_2 norm constraint. From left to right is for Syn3, Syn4 and Syn5, respectively.

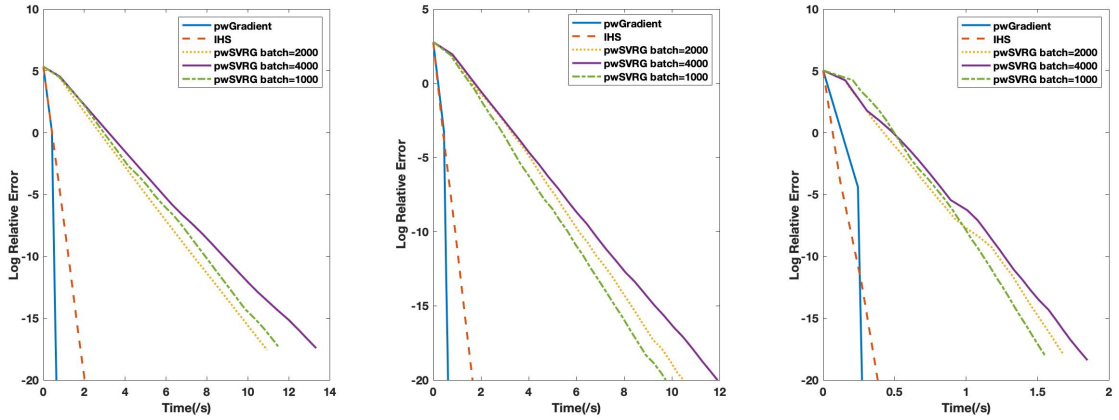

Figure 11: Experimental results on real datasets for the unconstrained high precision case. From left to right is for Year, Buzz and HT_Sensor dataset, respectively.

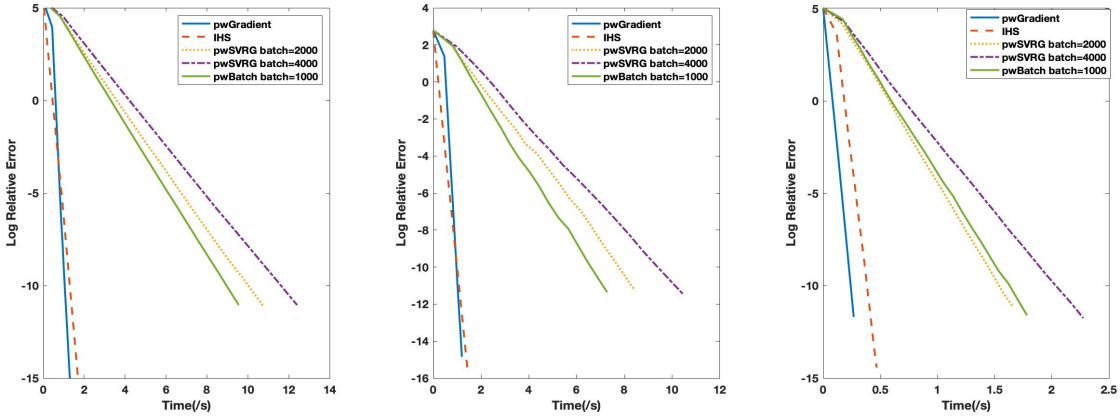

Figure 12: Experimental results on real datasets for high precision case with ℓ_1 norm constraint. From left to right is for Year, Buzz and HT_Sensor dataset, respectively.

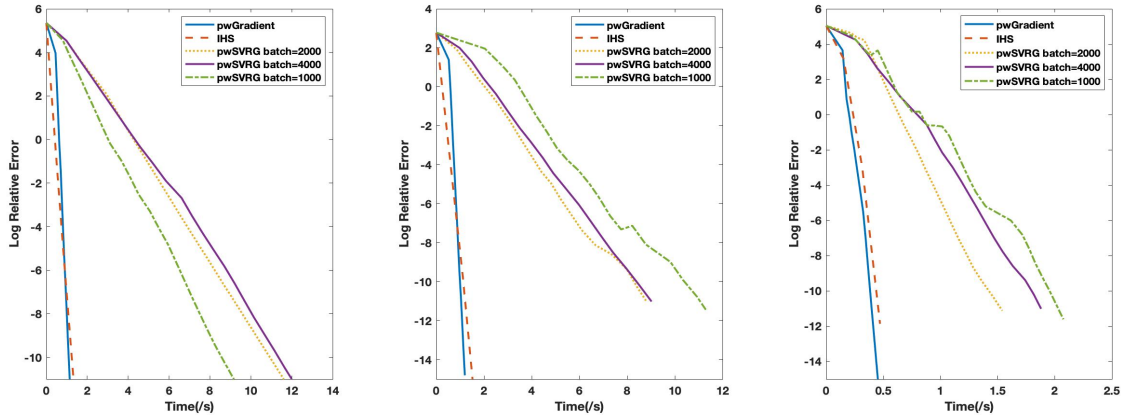

Figure 13: Experimental results on synthetic datasets for high precision case with ℓ_2 norm constraint. From left to right is for Year, Buzz and HT_Sensor dataset, respectively.

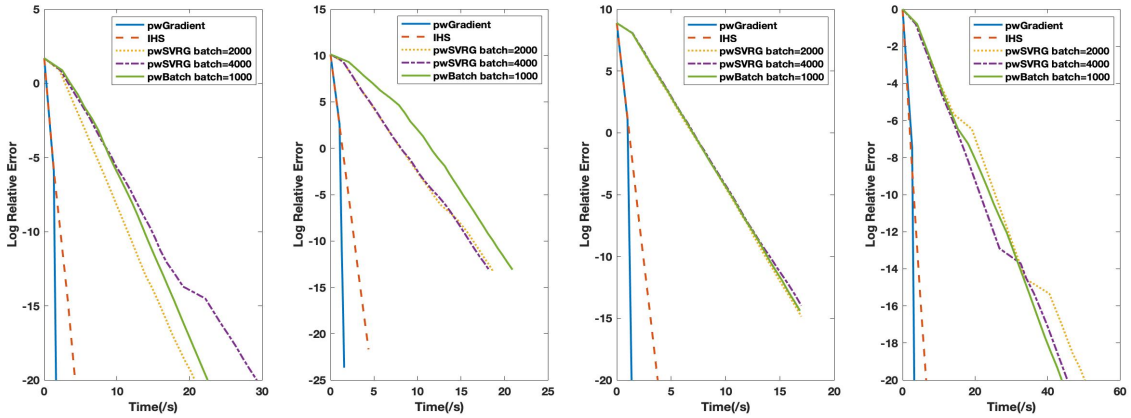

Figure 14: Experimental results on real datasets for the unconstrained high precision case. From left to right is for SUSY, Gas_Sensor_methane, Gas_Sensor_CO and HEPMASS, respectively.

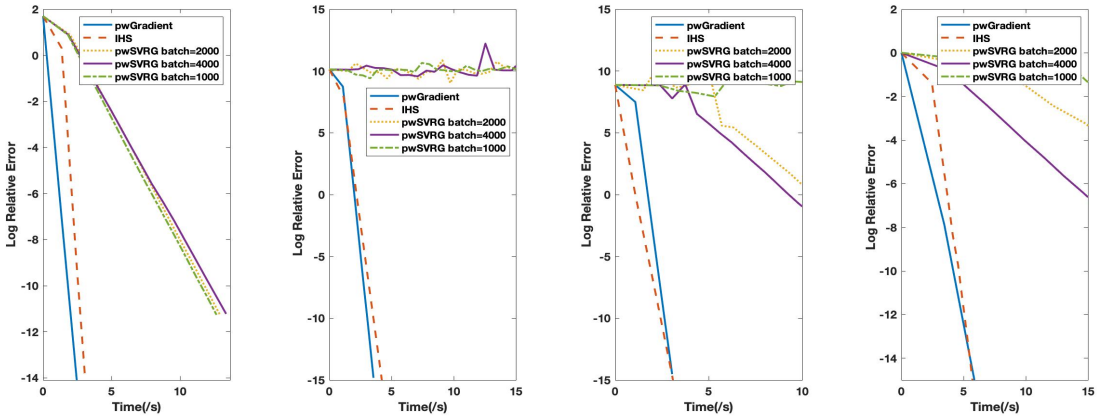

Figure 15: Experimental results on real datasets for high precision case with ℓ_1 norm constraint. From left to right is for SUSY, Gas_Sensor_methane, Gas_Sensor_CO and HEPMASS, respectively.

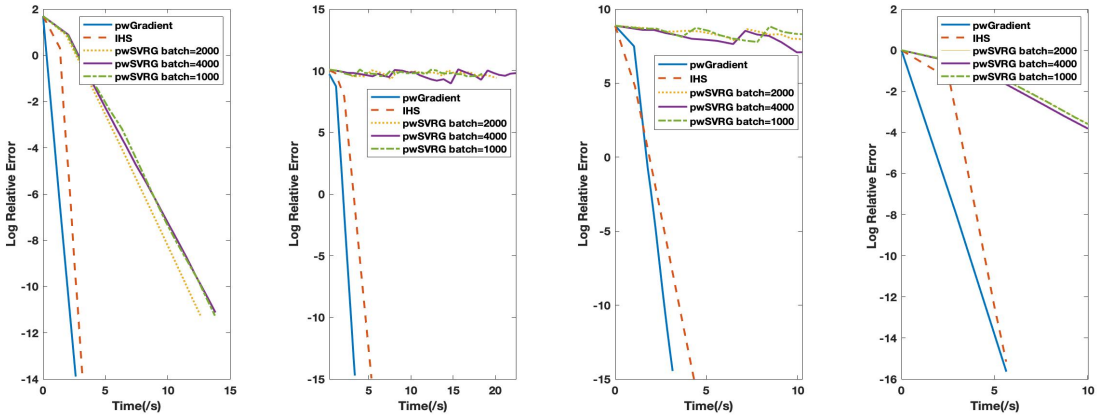

Figure 16: Experimental results on synthetic datasets for high precision case with ℓ_2 norm constraint. From left to right is for SUSY, Gas_Sensor_methane, Gas_Sensor_CO and HEPMASS, respectively.

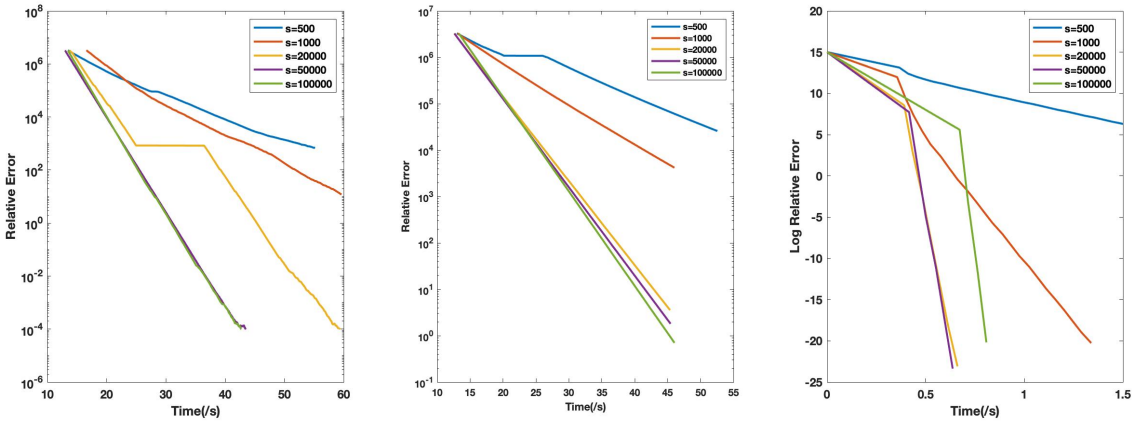

Figure 17: Experimental results on Syn3 dataset for methods with differnent sketch size. From left to right is for HDpwBatchSGD, HDpwBatchAccSGD and pwGradient respectively.

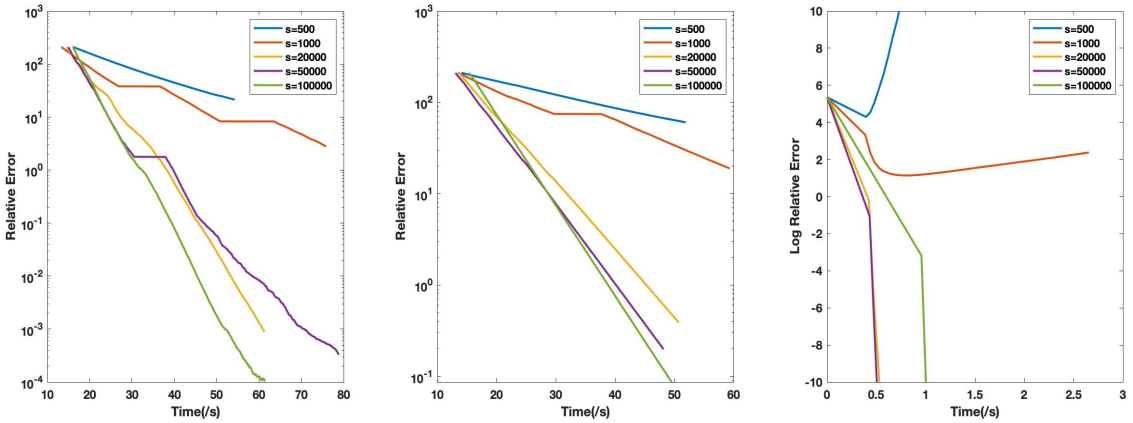

Figure 18: Experimental results on Year dataset for methods with differnent sketch size. From left to right is for HDpwBatchSGD, HDpwBatchAccSGD and pwGradient respectively.

[5] D. Wang, M. Ye, J. Xu, Differentially private empirical risk minimization revisited: Faster and more general, in: Advances in Neural Information Processing Systems, 2017, pp. 2722–2731.

[6] C. Musco, C. Musco, Randomized block krylov methods for stronger and faster approximate singular value decomposition, in: Advances in Neural Information Processing Systems, 2015, pp. 1396–1404.

[7] S. Paul, C. Boutsidis, M. Magdon-Ismail, P. Drineas, Random projections for support vector machines., in: AISTATS, Vol. 3, 2013, p. 4.

[8] C. Boutsidis, P. Drineas, M. Magdon-Ismail, Near-optimal column-based matrix reconstruction, SIAM Journal on Computing 43 (2) (2014) 687–717.

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

[10] D. Durfee, K. A. Lai, S. Sawlani, ℓ_1 regression using lewis weights preconditioning and stochastic gradient descent, in: S. Bubeck, V. Perchet, P. Rigollet (Eds.), Proceedings of the 31st Conference On Learning Theory, Vol. 75 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1626–1656.

[11] A. Gonen, F. Orabona, S. Shalev-Shwartz, Solving ridge regression using sketched preconditioned svrg, in: International Conference on Machine Learning, 2016, pp. 1397–1405.

[12] L. Zhang, M. Mahdavi, R. Jin, T. Yang, S. Zhu, Recovering the optimal solution by dual random projection., in: COLT, 2013, pp. 135–157.

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

[14] J. Yang, Y.-L. Chow, C. Ré, M. W. Mahoney, Weighted sgd for ℓ_p regression with randomized preconditioning, in: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2016, pp. 558–569.

[15] V. Rokhlin, M. Tygert, A fast randomized algorithm for overdetermined linear least-squares regression, Proceedings of the National Academy of Sciences 105 (36) (2008) 13212–13217.

[16] H. Avron, P. Maymounkov, S. Toledo, Blendenpik: Supercharging lapack's least-squares solver, SIAM Journal on Scientific Computing 32 (3) (2010) 1217–1236.

[17] M. Pilanci, M. J. Wainwright, Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares, Journal of Machine Learning Research 17 (53) (2016) 1–38.

[18] J. A. Tropp, Improved analysis of the subsampled randomized hadamard transform, Advances in Adaptive Data Analysis 3 (2011) 115–126.

[19] M. Takác, A. S. Bijral, P. Richtárik, N. Srebro, Mini-batch primal and dual methods for svms., in: ICML (3), 2013, pp. 1022–1030.

[20] R. H. Byrd, G. M. Chin, J. Nocedal, Y. Wu, Sample size selection in optimization methods for machine learning, Mathematical programming 134 (1) (2012) 127–155.

[21] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction using mini-batches, Journal of Machine Learning Research 13 (Jan) (2012) 165–202.

[22] S. Ghadimi, G. Lan, Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, ii: shrinking procedures and optimal algorithms, SIAM Journal on Optimization 23 (4) (2013) 2061–2089.

[23] D. Wang, J. Xu, Large scale constrained linear regression revisited: Faster algorithms via preconditioning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, 2018.

[24] G. Raskutti, M. W. Mahoney, A statistical perspective on randomized sketching for ordinary least-squares, The Journal of Machine Learning Research 17 (1) (2016) 7508–7538.

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

[26] N. Agarwal, S. Kakade, R. Kidambi, Y. T. Lee, P. Netrapalli, A. Sidford, Leverage score sampling for faster accelerated regression and erm, arXiv preprint arXiv:1711.08426.

[27] D. Needell, R. Ward, Batched stochastic gradient descent with weighted sampling, in: International Conference Approximation Theory, Springer, 2016, pp. 279–306.

[28] J. Tang, M. Golbabaee, M. E. Davies, Gradient projection iterative sketch for large-scale constrained least-squares, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, Vol. 70 of Proceedings of Machine Learning Research, PMLR, International Convention Centre, Sydney, Australia, 2017, pp. 3377–3386.

[29] M. Ledoux, On talagrand's deviation inequalities for product measures, ESAIM: Probability and statistics 1 (1997) 63–87.

[30] G. Lan, An optimal method for stochastic composite optimization, Mathematical Programming 133 (1) (2012) 365–397.

[31] K. L. Clarkson, D. P. Woodruff, Low-rank approximation and regression in input sparsity time, Journal of the ACM (JACM) 63 (6) (2017) 54.

[32] M. B. Cohen, Nearly tight oblivious subspace embeddings by trace inequalities, in: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, SIAM, 2016, pp. 278–287.

[33] S. Ghadimi, G. Lan, Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework, SIAM Journal on Optimization 22 (4) (2012) 1469–1492.

[34] Y. Nesterov, Introductory lectures on convex optimization: A basic course, Vol. 87, Springer Science & Business Media, 2013.

[35] S. Wang, A practical guide to randomized matrix computations with matlab implementations, arXiv preprint arXiv:1505.07570.

[36] H. Kasai, Sgdlibrary: A matlab library for stochastic optimization algorithms, The Journal of Machine Learning Research 18 (1) (2017) 7942–7946.

[37] J. Vanschoren, J. N. van Rijn, B. Bischl, L. Torgo, Openml: Networked science in machine learning, SIGKDD Explorations 15 (2) (2013) 49–60.

[38] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017). URL `http://archive.ics.uci.edu/ml`

[39] M. Jaggi, Revisiting frank-wolfe: Projection-free sparse convex optimization., in: ICML (1), 2013, pp. 427–435.

[40] L. Zhang, T. Yang, R. Jin, X. He, O (logt) projections for stochastic optimization of smooth and strongly convex functions, in: International Conference on Machine Learning, 2013, pp. 1121–1129.

[41] T. Yang, Q. Lin, L. Zhang, A richer theory of convex constrained optimization with reduced projections and improved rates, arXiv preprint arXiv:1608.03487.