The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Large Scale Constrained Linear Regression Revisited: Faster Algorithms via Preconditioning*

Di Wang
Department of Computer Science and Engineering
State University of New York at Buffalo

Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo

Abstract

In this paper, we revisit the large-scale constrained linear regression problem and propose faster methods based on some recent developments in sketching and optimization. Our algorithms combine (accelerated) mini-batch SGD with a new method called two-step preconditioning to 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.

Introduction

Since many problems in compressed sensing and machine learning can be formulated as a constrained linear regression problem, such as SVM, LASSO, signal recovery (Pilanci and Wainwright 2015), large scale linear regression with constraints now becomes one of the most popular and basic models in Machine Learning and has received a great deal of attentions from both the Machine Learning and Theoretical Computer Science communities. Formally, the problem can be 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 $e^d > n > d$ and $\mathcal{W}$ is a closed convex set. 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 $f(x) - \min_{x\in\mathcal{W}} f(x) \leq \epsilon$.

On one hand, recent developments on first-order stochastic methods, such as Stochastic Dual Coordinate Ascent (SDCA) (Shalev-Shwartz and Zhang 2013) and Stochastic Variance Reduced Gradient (SVRG) (Johnson and Zhang 2013), have made significant improvements on the convergence speed of large scale optimization problems in practice. On the other hand, random projection and sampling 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 (Musco and Musco 2015), SVM (Paul et al. 2013), column subset selection (Boutsidis, Drineas, and Magdon-Ismail 2014) and l_p regression (for $p \in [1, 2]$) (Dasgupta et al. 2009). 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, (Gonen, Orabona, and Shalev-Shwartz 2016; Gonen and Shalev-Shwartz 2015) proposed faster methods for Ridge Regression and Empirical Risk Minimization, respectively, by using SVRG, Stochastic Gradient Descent (SGD) and low rank approximation. (Zhang et al. 2013) achieved guarantee for Empirical Risk Minimization by using random projection in dual problem.

In this paper, we revisit the preconditioning method for solving large-scale constrained linear regression problem, and propose faster algorithms for both the low ($\epsilon \approx 10^{-1} \sim 10^{-4}$) and high ($\epsilon \leq 10^{-8}$) precision cases by combining it with some recent developments in sketching and optimization. Our main contributions can be summarized as follows.

- For the low precision case, we first propose a novel algorithm called HDpwBatchSGD (*i.e.,* Algorithm 2) by combining a new method called 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 method 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 (Takác et al. 2013; Byrd et al. 2012; Dekel et al. 2012). 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 batch size by a factor of b. To further improve the running time, we also use the Multi-epoch Stochastic Accelerated mini-batch SGD (Ghadimi and Lan 2013) to obtain another slightly different algorithm called HDpwAccBatchSGD, which has a time complexity lower than that of the state-of-the-art technique (Yang et al. 2016).

*This research was supported in part by NSF through grants IIS-1422591, CCF-1422324, and CCF-1716400

Method	Complexity for Unconstrained	Complexity for Constrained	Precision
(Drineas et al. 2011)	$O\left(nd\log(\frac{d}{\epsilon}) + d^3\log n\log d + \frac{d^3\log n}{\epsilon}\right)$	$O\left(nd\log n + \text{poly}(d, \frac{1}{\epsilon^2})\right)$	Low
pwSGD(Yang et al. 2016)	$O\left(nd\log n + \frac{d^3\log(\frac{1}{\epsilon})}{\epsilon}\right)$	$O\left(nd\log n + \frac{\text{poly}(d)\log(\frac{1}{\epsilon})}{\epsilon}\right)$	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
HDpwAccBatchSGD	$O\left(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{d^3\log n}{r\epsilon} + rd^2\log\frac{1}{\epsilon}\right)$	$O\left(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{\text{poly}(d)\log n}{r\epsilon} + r\text{poly}(d)\log\frac{1}{\epsilon}\right)$	Low
(Rokhlin and Tygert 2008; Avron, Maymounkov, and Toledo 2010)	$O(nd\log\frac{d}{\epsilon} + d^3\log d)$	—	High
IHS (Pilanci and Wainwright 2016)	$O\left(nd\log d\log\frac{1}{\epsilon} + d^3\log\frac{1}{\epsilon}\right)$	$O\left((nd\log d + \text{poly}(d))\log\frac{1}{\epsilon}\right)$	High
Preconditioning+SVRG	$O(nd\log d + (nd + d^3)\log\frac{1}{\epsilon})$	$O(nd\log d + (nd + \text{poly}(d))\log\frac{1}{\epsilon})$	High
pwGradient	$O(nd\log d + (nd + d^3)\log\frac{1}{\epsilon})$	$O(nd\log d + (nd + \text{poly}(d))\log\frac{1}{\epsilon})$	High

Table 1: Summary of the time complexity of several linear regression methods for finding x_t such that $||Ax_t - b||_2^2 - ||Ax^* - b||_2^2 \leq \epsilon$. For sketching based methods, we use the Subsampled Randomized Hadamard Transform (SRHT) (Tropp 2011) as the sketch matrix. All methods run in sequential environment. r is an input in our method. '—' means not applicable.

- The optimality on speeding-up further inspires us to think about how it will perform if using the whole gradient, *i.e.* projected Gradient Descent, (called pwGradient (*i.e.,* Algorithm 4)). A somewhat surprising discovery is that it actually allows us to have an alternative implementation of the Iterative Hessian Sketch (IHS) method, 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.
- Numerical experiments on large synthetic/real benchmark datasets confirm the theoretical analysis of HDpwBatchSGD and pwGradient. Our methods outperform existing ones in both low and high precision cases.

Related Work

There is a vast number of papers studying the large scale constrained linear regression problem from different perspectives (Yang, Meng, and Mahoney 2016). 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), due to their similar natures to ours. We summarize these methods in **Table 1**.

For the low precision case, (Drineas et al. 2011) directly uses sketching with a very large sketch size of 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 (Yang et al. 2016), 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 method provides both and runs faster in practice. Also we have to note that, even if we set $r = 1$ in HDpwAccBatchSGD, the time complexity is less than that in pwSGD when $\frac{1}{\epsilon} > n$. (Needell and Ward 2016) 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 (Rokhlin and Tygert 2008), our method can be extended to the constrained case. Compared with IHS (Pilanci and Wainwright 2016), ours uses only one step of sketching and thus has a lower time complexity. Although a similar time complexity can be achieved by using the preconditioning method in (Yang, Meng, and Mahoney 2016) and SVRG, ours performs better in practice. We notice that (Tang, Golbabaee, and Davies 2017) 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 it with ours here.

Preliminaries

Let A be a matrix in $\mathbb{R}^{n\times d}$ with $e^d > n > d$ and $d = \text{rank}(A)$, and denote 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. (Note that our proposed methods can be easily extended to the case of $d > \text{rank}(A)$.) 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. In this section we will give several definitions and lemmas that will be used throughout this paper. Due to space limit, we leave all the proofs to the full paper.

Randomized Linear Algebra

First we give the definition of (α, β, 2)-conditioned matrix. Note that it is a special case of (α, β, p)-well conditioned basis in (Yang, Meng, and Mahoney 2016).

Definition 1. *((α,β,2)-conditioned) (Yang et al. 2016)) 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 $(O(\sqrt{d}), O(1), 2)$-conditioned matrix U from A (called $O(\sqrt{d}), O(1), 2)$-conditioned basis of A) in $o(nd^2)$ time, through **Algorithm 1**. Note that in practice we can just set $O(1)$ as a small constant and return R instead of AR^{-1}.

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

1: Construct an oblivious subspace embedding(sketch matrix) $S \in \mathbb{R}^{s\times n}$ with $n > s > d$ 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 $O(\sqrt{d})$,$O(1)$,2)-conditioned basis of A.

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

Next, we give the definition of Randomized Hadamard Transform (Tropp 2011), which is the tool to be used in the second step of our preconditioning.

Definition 2. *(Randomized Hadamard Transform) $M = HD \in \mathbb{R}^{n\times n}$ is called a Randomized Hadamard Tranform, where n is assumed to be 2^s for some integer s, $D \in \mathbb{R}^{n\times n}$ 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 $(\alpha,\beta,2)$-conditioned matrix can be viewed as an approximate orthogonal matrix, an immediate question is whether an $(\alpha,\beta,2)$-conditioned matrix can also achieve the same result. We answer this question by the following theorem,

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

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

By the above theorem, 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 is small. Also since H, D are orthogonal, we have $\|HDUy - HDb\|_2 = \|Uy - b\|_2$ for any y.

Strongly Smooth SGD and Mini-batch SGD

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.,

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

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

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

$$= P_{\mathcal{W}}(x_k - \eta_k\nabla f_{i_k}(x_k)) \quad (3)$$

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 (Lan 2012).

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

$$\eta = \min(\frac{1}{2L}, \sqrt{\frac{D_{\mathcal{W}}^2}{2T\sigma^2}}), \quad (4)$$

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

$$T = \Theta(\frac{D_{\mathcal{W}}^2\sigma^2}{\epsilon^2}) \quad (5)$$

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

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,

$$\min_{x\in\mathcal{W}} F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x). \quad (6)$$

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 that in (Needell and Ward 2016), which is to partition all the indices into $\lceil\frac{n}{r}\rceil$ groups and samples only within one group in each iteration.*

Two-step Preconditioning Mini-batch SGD

Main Idea

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

$$\min_{x\in\mathcal{W}} f(x) = \|Ax-b\|_2^2 = \min_{y\in\mathcal{W}'} \|Uy-b\|_2^2 \tag{7}$$

$$= \|HDUy - HDb\|_2^2 = \frac{1}{n}\sum_{i=1}^{n} n\|(HDU)_i y - (HDb)_i\|_2^2. \tag{8}$$

The first step of the preconditioning (7) is to get U, an $(O(\sqrt{d}),O(1),2)$-conditioned basis of A (*i.e.,* $U = AR^{-1}$; see Algorithm 1), which means that the function in problem (7) is an $O(d)$-smooth (actually it is $O(1)$-smooth, see Table 2) and $O(1)$-strongly convex function. The second step of the preconditioning (8) is to use Randomized Hadamard Transform to 'spread out' the row norm of U by **Theorem 1**. Then, we use mini-batch SGD with uniform sampling for each iteration. We can show that $x^* = R^{-1}y^*$, where $y^* = \arg\min_{y\in\mathcal{W}'} \|HDUy - HDb\|_2^2$, and $\mathcal{W}'$ is the convex set corresponding to $\mathcal{W}$.

Algorithm

The main steps of our algorithm are given in the following **Algorithm 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: $c_{\tau_t} = \frac{2n}{r}\sum_{j\in\tau_t}(HDA)_j^T[(HDA)_j x_{t-1} - (HDb)_j] = \frac{2n}{r}(HDA)_{\tau_t}^T[(HDA)_{\tau_t}x_{t-1} - (HDb)_{\tau_t}]$
6: $x_t = \arg\min_{x\in\mathcal{W}} \frac{1}{2}\|R(x_{t-1}-x)\|_2^2 + \eta\langle c_{\tau_t}, x\rangle = \mathcal{P}_{\mathcal{W}}(x_{t-1} - \eta R^{-1}(R^{-1})^T c_{\tau_t})$
7: **end for**
8: **return** $x_T^{\text{avg}} = \frac{\sum_{i=1}^T x_i}{T}$

Note that the way of updating x_t in **Algorithm 2** is equivalent to the updating procedure of y_t for the reformed problem (8) (*i.e.,* set $y_0 = R^{-1}x_0$, then use mini-batch SGD and let $x_t = R^{-1}y_t$). There are several benefits if we update x_t directly.

- Directly updating y_t 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.
- 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 much easier to solve the optimization problem in Step 7.

In the above algorithm, Step 1 is the same as the first step of pwSGD in (Yang et al. 2016). 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, (Yang et al. 2016) uses the exact leverage score instead of its approximation. By Theorem 1 and 2, 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 3. *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 (4), we have*

$$\mathbb{E}f(x_T^{avg}) - f(x^*) \le \frac{3\sqrt{2}D_{\mathcal{W}}\sigma}{\sqrt{rT}}, \tag{9}$$

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, **Algorithm 2** *ensures the following with high probability, $\mathbb{E}|f(x_T^{avg}) - f(x^*)| \le \epsilon$.*

The time complexity of our algorithm can be easily obtained as

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

where $\text{time}(R)$ 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 $\text{poly}(d)$ 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

$$O(nd\log n + d^3\log d + (\text{poly}(d) + rd)\frac{d\log n}{r\epsilon^2}). \tag{10}$$

Further Reducing the Iteration Complexity

Theorem 3 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. 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 3. The preconditioning steps are the same, and the optimization method is the multi-epoch stochastic accelerated gradient descent, which was proposed in (Ghadimi and Lan 2012; 2013). In stead of using Theorem 2, we will use the following theorem whose proof was given in (Ghadimi and Lan 2013).

Table 2: Time complexity for computing R in step 1 of Algorithm 2, 3, 4 with different sketch matrix (Yang et al. 2016).

Sketch Matrix	Time Complexity	$\kappa(AR^{-1})$
Gaussian Matrix	$O(nd^2)$	$O(1)$
SRHT(Pilanci and Wainwright 2016)	$O(nd\log d + d^3\log d)$	$O(1)$
CountSketch	$O(\text{nnz}(A) + d^4)$	$O(1)$
Sparse l_2 Embedding	$O(\text{nnz}(A)\log d + d^3\log d)$	$O(1)$

Theorem 4. *If Assumption 1 and 2 hold and $\epsilon < V_0$, then after $O(\sqrt{\frac{L}{\mu}}\log(\frac{V_0}{\epsilon}) + \frac{\sigma^2}{\mu\epsilon})$ iterations of stochastic accelerated gradient descent with $O(\log(\frac{V_0}{\epsilon}))$ epochs, the output of multi-epoch stochastic accelerated gradient descent x_S satisfies $\mathbb{E}F(x_S) - F(x^*) \leq \epsilon$, where V_0 is a given bound such that $F(x_0) - F(x^*) \leq V_0$.*

Thus, we can use a two-step preconditioning and multi-epoch stochastic accelerated mini-batch gradient descent to obtain an algorithm (called HDpwAccBatchSGD) similar to **Algorithm 2**, as well as the following theorem. Due to the space limit, we leave the details of the algorithm to the full paper.

Theorem 5. *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 fix $\epsilon < V_0$. Then with high probability, 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 HDpwAccBatchSGD, the output x_S satisfies $\mathbb{E}F(x_S) - F(x^*) \leq \epsilon$. Moreover, if we take SRHT as the sketching matrix , the total time complexity is*

$$O(nd\log n + \frac{d^2\log n}{\epsilon} + \frac{poly(d)\log n}{r\epsilon} + rpoly(d)\log\frac{1}{\epsilon}). \quad (11)$$

Improved Iterative Hessian Sketch

Now, we go back to our results in (10) and (11). One benefit of these results is that ϵ is independent of n and depends only on poly(d) and $\log n$. If directly using the Variance Reduced methods developed in recent years (such as (Johnson and Zhang 2013)), we can get the time complexity $O((n+\kappa)\text{poly}(d)\log\frac{1}{\epsilon})$. Comparing with these methods, we know that HDpwBatchSGD and HDpwAccBatchSGD are more suitable for the **low precision and large scale** case. Recently (Pilanci and Wainwright 2016) introduced the Iterative Hessian Sketch (IHS) method to solve the large scale constrained linear regression problem (see Algorithm 3). IHS is capable of achieving high precision, but needs a sequence of sketch matrices $\{S^t\}$ (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. In this section, we show that by adopting our 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 4**)).

Our pwGradient algorithm uses the first step of preconditioning (*i.e.,* Step 1 of Algorithm 2) and then performs gradient decent (GD) operations, instead of the mini-batch SGD in Algorithm 2 (note that we do not need the second step of preconditioning since it 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 ϵ-solution.

Algorithm 3 IHS(A,b,x_0,s)(Pilanci and Wainwright 2015)

Input: x_0 is the initial point, 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 step 1 of Algorithm 1. Compute $M = S^{t+1}A$.
3: Perform the updating

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

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

Algorithm 4 pwGradient(A,b,x_0,s, η)

Input: x_0 is the initial point, s is the sketch size, 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

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

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

Theorem 6. *Let $f(x) = \|Ax-b\|_2^2$. Then, for some step size $\eta = O(1)$ in pwGradient, the following holds,*

$$f(x_t) - f(x^*) \leq (1 - O(1))^t O((f(x_0) - f(x^*))) \quad (12)$$

with high probability.

Below we will 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 is equivalent to that of IHS with $\{S^t\} = S$. Let QR be the QR-decomposition of $S^{t+1}A = SA$. Then, we have

$$(QR)^{-1}((QR)^{-1})^T = (R)^{-1}(R^{-1})^T.$$

Although they look like the same, the ideas behind them are quite different. IHS is based on sketching the Hessian and uses the second-order method of the optimization problem, while ours is based on preconditioning the original problem and uses the first-order method. Thus we need step size η, while IHS does not require it. As we can see from the above, $\eta = \frac{1}{2}$ is sufficient. One main advantage of our method is that the time complexity is much lower than that of IHS, since it needs only one step of sketching. If SRHT is used as the sketch matrix, the complexity of our method becomes $O(nd \log d + d^3 \log d + (nd + \text{poly}(d)) \log \frac{1}{\epsilon})$ (see Table 1 for comparison with IHS).

Numerical Experiments

In this section we present some experimental results of our proposed methods (for convenience, we just use the slower HDpwBatchSGD algorithm for the low precision case). We will focus on the iteration complexity and running time. Experiments confirm that our algorithms are indeed faster than those existing ones. Our algorithms are implemented using CountSketch as the sketch matrix $S \in \mathbb{R}^{s \times n}$ in the step for computing R^{-1}. The Matlab code of CountSketch can be found in (Wang 2015).

- **HDpwBatchSGD**, *i.e.* ***Algorithm 2***. We use the step size as described in Theorem 2 (note that we assume that the step size is already known in advance).
- **pwGradient**, *i.e.* ***Algorithm 4***. We set $\eta = \frac{1}{2}$ as the step size.

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
Buzz	5×10^5	77	10^8	20000
Year	5×10^5	90	3000	20000

In both the low and high precision cases, we select some widely recognized algorithms with guaranteed time complexities for comparisons. For the low precision case, we choose pwSGD (Yang et al. 2016), which has the best known time complexity, and use the optimal step size. We also choose SGD and Adagrad for comparisons. For the high precision case, we use a method called pwSVRG for comparison, which uses preconditioning first and then performs SVRG with different batch sizes (the related method can be found in (Rokhlin and Tygert 2008)). Note that since the condition number of the considered datasets are very high, directly using SVRG or related methods could lead to rather poor performance; thus we do not use them for comparison (although (Tang, Golbabaee, and Davies 2017) used SAGA for comparison, it was done after normalizing the datasets). We also compare our proposed pwGradient algorithm with IHS. The code for SGD and Adagrad can be found in (https://github.com/hiroyuki-kasai/SGDLibrary). All the methods are implemented using MATLAB. We measure the performance of methods by the wall-clock time or iteration number. For each experiment, we test every method 10 times and take the best. Also for the low precision solvers, we firstly normalize the dataset.

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 sketch size used in the experiments. The datasets Year[1] and Buzz[2] come from UCI Machine Learning Repository (Lichman 2013).

We consider both the unconstrained case and the constrained cases with ℓ_1 and ℓ_2 norm ball constraints. For the constrained case, we first generate the optimal solution for the unconstrained case, and then set it as the radius of balls.

Experiments on Synthetic Datasets

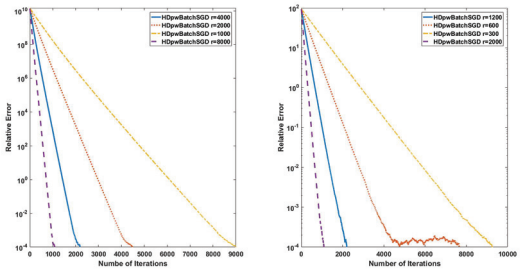

Figure 1: Iteration number of HDpwBatchSGD with different batch size r on (from left to right) datasets Syn1 and Syn2 (unconstrained case).

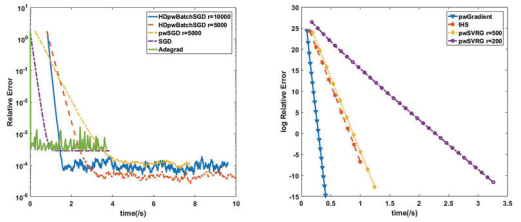

Figure 2: Experimental results on dataset Syn1 (unconstrained case); left is for the low precision solvers, right is for the high precision solvers.

We 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. In each experiment, the initial value x_0 is set to be the zero vector. We start with some numerical experiments to gain insights to the iteration complexity and

[1]https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd
[2]https://archive.ics.uci.edu/ml/datasets/Buzz+in+social+media+

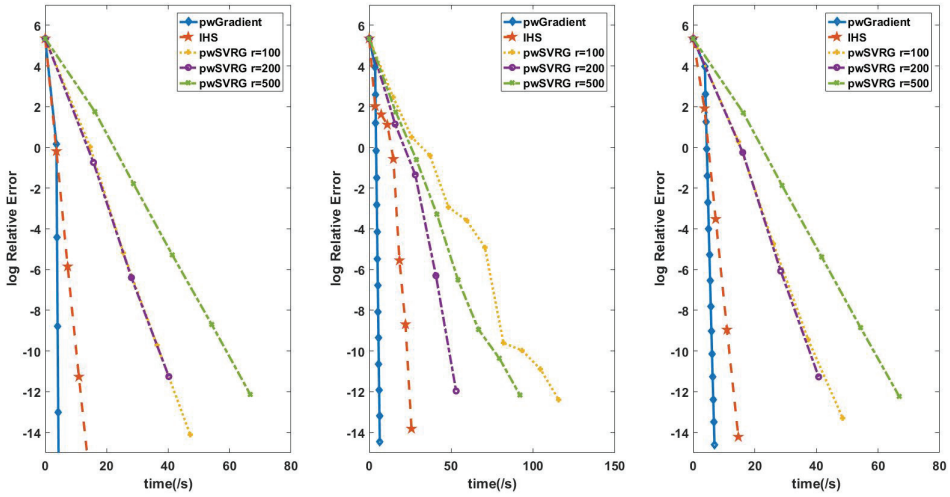

Figure 3: Experimental Results on dataset Year for the high precision solvers; left is for the unconstrained case, middle is for the ℓ_1 constrained case, and right is for the ℓ_2 constrained case.

relative error shown in **Theorem 3**, and verify them for the unconstrained case using synthetic datasets Syn1 and Syn2. Later we determine the relative errors for the low and high precision solvers, and plot the results in Figures 1 and 2.

Experiments on Real Datasets

We consider the unconstrained and the ℓ_1 and ℓ_2 constrained linear regression problems on the Buzz dataset for both the low and high precision cases and on the Year dataset for the high precision case. The results are plotted in Figures 3, 4, 5 and 6, respectively.

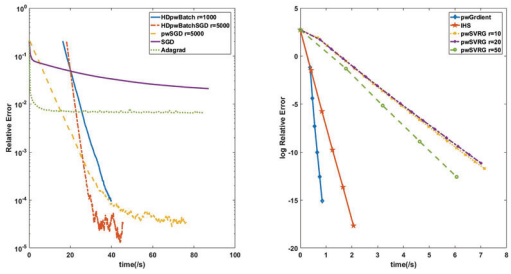

Figure 4: Experimental results on dataset Buzz (unconstrained case); left is for the low precision solvers and right is for the high precision solvers.

Results

From Figure 1, we can see that in both cases if the batch size is increased by a factor of $b = 2$, the iteration complexity approximately decreases by a factor of $b = 2$. This confirms the theoretical guarantees of our methods. From other figures, we can also see that in both the high (Figures 2,3,4, and 5) and the low precision (Figures 2,4, and 6) cases, our

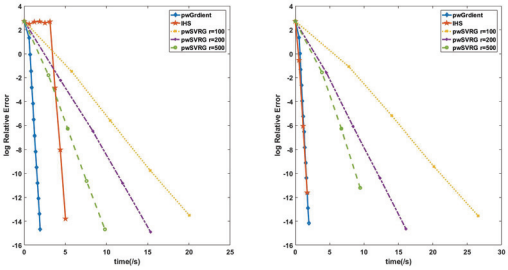

Figure 5: Experimental results on dataset Buzz for the high precision solvers (constrained case); left is for the ℓ_1 constrained case and right is for the ℓ_2 constrained case.

methods considerably outperform other existing methods. Particularly, in the low precision case, the relative error of HDpwBatchSGD decreases much faster with a large batch size (except for the ℓ_2 constrained case in Figure 6). With a large batch size, our method runs even faster than pwSGD despite a relatively long the preconditioning time (due to the second preconditioning step). This is because in practice CountSketch is faster than SRHT, especially when the dataset is sparse. This also confirms the claim of our methods.

For the high precision case, experiments indicate that pwGradient can even outperform the stochastic methods. Also, as we mentioned earlier, pwGradient needs to sketch only once. This enables it to run much faster than IHS, and still preserves the high probability of success.

Conclusion

In this paper, we studied the large scale constrained linear regression problem, and presented new methods for both

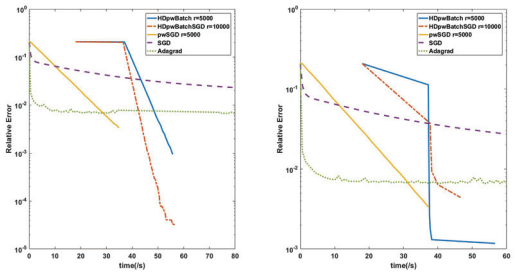

Figure 6: Experimental results on dataset Buzz for the low precision solvers (constrained case); left is for the ℓ_1 constrained case and right is for the ℓ_2 constrained case.

the low and high precision cases, using some recent developments in 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.

References

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

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

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

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

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

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

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

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

Gonen, A., and Shalev-Shwartz, S. 2015. Faster sgd using sketched conditioning. *arXiv preprint arXiv:1506.02649*.

Gonen, A.; Orabona, F.; and Shalev-Shwartz, S. 2016. Solving ridge regression using sketched preconditioned svrg. *arXiv preprint arXiv:1602.02350*.

Johnson, R., and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 315–323.

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

Lichman, M. 2013. UCI machine learning repository.

Musco, C., and Musco, C. 2015. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, 1396–1404.

Needell, D., and Ward, R. 2016. Batched stochastic gradient descent with weighted sampling. *arXiv preprint arXiv:1608.07641*.

Paul, S.; Boutsidis, C.; Magdon-Ismail, M.; and Drineas, P. 2013. Random projections for support vector machines. In *AISTATS*, volume 3, 4.

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

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

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

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

Takác, M.; Bijral, A. S.; Richtárik, P.; and Srebro, N. 2013. Mini-batch primal and dual methods for svms. In *ICML (3)*, 1022–1030.

Tang, J.; Golbabaee, M.; and Davies, M. E. 2017. Gradient projection iterative sketch for large-scale constrained least-squares. In Precup, D., and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3377–3386. International Convention Centre, Sydney, Australia: PMLR.

Tropp, J. A. 2011. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis* 3:115–126.

Wang, S. 2015. A practical guide to randomized matrix computations with matlab implementations. *arXiv preprint arXiv:1505.07570*.

Yang, J.; Chow, Y.-L.; Ré, C.; and Mahoney, M. W. 2016. Weighted sgd for ℓ_p regression with randomized preconditioning. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 558–569. SIAM.

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

Zhang, L.; Mahdavi, M.; Jin, R.; Yang, T.; and Zhu, S. 2013. Recovering the optimal solution by dual random projection. In *COLT*, 135–157.