Gradient Projection Iterative Sketch for Large-Scale Constrained Least-Squares

Junqi Tang [1] Mohammad Golbabaee [1] Mike E. Davies [1]

Abstract

We propose a randomized first order optimization algorithm Gradient Projection Iterative Sketch (GPIS) and an accelerated variant for efficiently solving large scale constrained Least Squares (LS). We provide the first theoretical convergence analysis for both algorithms. An efficient implementation using a tailored line-search scheme is also proposed. We demonstrate our methods' computational efficiency compared to the classical accelerated gradient method, and the variance-reduced stochastic gradient methods through numerical experiments in various large synthetic/real data sets.

1. Introduction

We are now in an era of boosting knowledge and large data. In our daily life we have various signal processing and machine learning applications which involve the problem of tackling a huge amount of data. These applications vary from Empirical Risk Minimization (ERM) for statistical inference, to medical imaging such as the Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), channel estimation and adaptive filtering in communications, and in machine learning problems where we need to train a neural network or a classifier from a large amount of data samples or images. Many of these applications involve solving constrained optimization problems. In a large data setting a desirable algorithm should be able to simultaneously address high accuracy of the solutions, small amount of computations and high speed data storage.

Recent advances in the field of randomized algorithms have provided us with powerful tools for reducing the computation for large scale optimizations. From the latest literature we can clearly see two streams of randomized algorithms, the first stream is the stochastic gradient descent (SGD) and its variance-reduced variants (Johnson & Zhang, 2013)(Konečnỳ & Richtárik, 2013)(Defazio et al., 2014)(Allen-Zhu, 2016). The stochastic gradient techniques are based on the computationally cheap unbiased estimate of the true gradients with progressively reduced estimation variance. Although there has been several works on SGD techniques for performing constrained optimization (Xiao & Zhang, 2014)(Konečnỳ et al., 2016), to the best of our knowledge, there are no results highlighting the computational speed up one could achieve by exploiting the data structure promoted by the constraint set.

This paper follows a second line of research and uses *sketching* techniques, the crux of which is reducing the dimensionality of a large scale problem by random projections (e.g., sub-Gaussian matrices, Fast Johnson-Lindenstrauss Transforms (FJLT) (Ailon & Liberty, 2008)(Ailon & Chazelle, 2009), the Count Sketch (Clarkson & Woodruff, 2013), the Count-Gauss Sketch (Kapralov et al., 2016) or random sub-selection) so that the resulting sketched problem becomes computationally tractable. The meta-algorithms Classical Sketch (CS)(Mahoney, 2011)(Drineas et al., 2011)(Pilanci & Wainwright, 2015) and the Iterative Hessian Sketch (IHS) (Pilanci & Wainwright, 2016) have been recently introduced for solving efficiently large scale constrained LS problems which utilize the random sketching idea combined with the fact that solutions have low-dimensional structures such as sparsity in a properly-chosen dictionary, low-rank, etc.

1.1. Main Contributions

- **Novel first order solvers based on iterative sketches for constrained Least-squares**

  We propose a basic first order algorithm Gradient Projection Iterative Sketch (GPIS) based on the combination of the *Classical Sketch* (Pilanci & Wainwright, 2015) and *Iterative Hessian Sketch* (Pilanci & Wainwright, 2016) for efficiently solving the constrained Least-squares, and also an accelerated variant by applying Nesterov's acceleration scheme (Nesterov, 2007)(Nesterov, 2013a).

[1]Institute for Digital Communications, the University of Edinburgh, Edinburgh, UK. Correspondence to: Junqi Tang <J.Tang@ed.ac.uk>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

- **Theoretical analysis for both GPIS and Acc-GPIS**

  Although there exists established theories for the sketching programs in (Pilanci & Wainwright, 2015)(Pilanci & Wainwright, 2016) which describes their estimation performance under the assumption that the sketched programs are solved exactly, there is no theoretical analysis of the use of first order methods within this framework, where each of the sketched programs are only approximately solved. The paper is the first one to provide this convergence analysis.

- **Structure exploiting algorithms**

  In related theoretical works in sketching (Pilanci & Wainwright, 2015)(Pilanci & Wainwright, 2016), convex relaxation (Chandrasekaran & Jordan, 2013), and the Projected Gradient Descent (PGD) analysis (Oymak et al., 2015) with greedy step sizes when the data matrix is a Gaussian map, researchers have discovered that the constraint set is able to be exploited to accelerate computation. In this paper's convergence analysis of the proposed algorithms (which have an inner loop and an outer loop), we show explicitly how the outer loop's convergence speed is positively influenced by the constrained set. [1]

- **Sketched gradients versus stochastic gradients – quality versus quantity**

  The proposed GPIS algorithm draws a different line of research for first order randomized algorithms from the SGD and its recently introduced variance-reduced variants such as SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014) by utilizing randomized sketching techniques and deterministic iterations instead of the stochastic iterations. This approach leads to convenience in optimally choosing the step size by implementing line search because it follows the classical results and techniques in first order optimization. Although such stochastic gradient algorithms have good performance in terms of epoch counts when a small minibatch size is used, this type of measure does not consider at least three important aspects: 1) the computational cost of projection / proximal operator, 2) the modern computational devices are usually more suitable for vectorized / parallel computation, 3) the operational efforts to access new data batches each iteration (note that the large data should be stored in large memories, which are usually slow).

  It is well known that the small batch size in stochastic gradients usually leads to a greater demand on the number of iterations. In the cases where the projection / proximal operator is costly to compute, for instance, if we wish to enforce sparsity in a transformed domain, or an analytical domain (total-variation), we would need to use a large batch size in order to control computation which generally would not be favorable for stochastic gradients techniques as they usually achieves best performance when small batch size is used. In this paper we have designed experiments to show the time efficiency of the sketched gradients with Count-sketch (Clarkson & Woodruff, 2013) and an aggressive line-search scheme for near-optimal choice of step size each iteration (Nesterov, 2007) compared to a mini-batched version of the SAGA algorithm (Defazio et al., 2014) and the accelerated full gradient method (Beck & Teboulle, 2009) in large scale constrained least-square problems.

[1] Meanwhile we can show empirically that the inner loop is also being able to choose an aggressive step size with respect to the constraint. This extra step-size experiment can be found in the supplementary material.

1.2. Background

Consider a constrained Least-squares regression problem in the large data setting. We have the training data matrix $A \in \mathbb{R}^{n\times d}$ with $n > d$ and observation $y \in \mathbb{R}^n$. Meanwhile we restrict our regression parameter to a convex constrained set $\mathcal{K}$ to enforce some desired structure such as sparsity and low-rank[2]:

$$x^\star = \arg\min_{x\in\mathcal{K}} \left\{ f(x) := \|y - Ax\|_2^2 \right\}. \tag{1}$$

Then we define the error vector e as:

$$e = y - Ax^\star \tag{2}$$

A standard first order solver for (1) is the projected gradient algorithm (we denote the orthogonal projection operator onto the constrained set $\mathcal{K}$ as $\mathcal{P}_\mathcal{K}$):

$$x_{j+1} = \mathcal{P}_\mathcal{K}(x_j - \eta A^T(Ax_j - y)). \tag{3}$$

Throughout the past decade researchers proposed a basic meta-algorithm for approximately solving the Least-squares problem that we call the *Classical Sketch* (CS), see e.g. (Mahoney, 2011) (Drineas et al., 2011) (Pilanci & Wainwright, 2015), which compresses the dimension of the LS and makes it cheaper to solve. The Johnson-Lindenstrauss theory (Johnson & Lindenstrauss, 1984) (Dasgupta & Gupta, 2003) and the related topic of Compressed Sensing (Donoho, 2006)(Candes et al., 2006)(Baraniuk et al., 2008) revealed that random projections can achieve stable embeddings of high dimensional data into lower dimensions and that the number of measurements required is proportional to the intrinsic dimensionality of data (as opposed to the ambient dimension)

[2] In scenarios where we do not know the exact constraint $\mathcal{K}$, we may wish to use *regularized* least-squares instead of strict constraint. This paper focus on the constrained case and leave the extension for the proximal setting as future work.

which is manifested in the set of constraints $\mathcal{K}$. This motivates replacing the original constrained LS problem with a sketched LS (Pilanci & Wainwright, 2015):

$$\hat{x} = \arg\min_{x \in \mathcal{K}} \left\{ f_0(x) := \|Sy - SAx\|_2^2 \right\}, \qquad (4)$$

where the sketching matrix $S \in \mathbb{R}^{m \times n}, m \ll n$ is a random projection operator which satisfies:

$$E\left(\frac{S^T S}{m}\right) = I. \qquad (5)$$

When the embedding dimension m is larger than a certain factor of the true solution's intrinsic dimension (measured through a statistical tool called the Gaussian Width (Chandrasekaran et al., 2012)), the *Classical Sketch* (4) ensures a robust estimation of $x^\star$ with a noise amplification factor compared to the estimator given by solving the original LS problem (1), and it has been shown that the smaller the embedding dimension m is, the bigger the noise amplification factor will be. To get a sketching scheme for the scenarios where a high accuracy estimation is demanded, a new type of meta-algorithm *Iterative Hessian Sketch* (IHS) was introduced by Pilanci and Wainwright (Pilanci & Wainwright, 2016):

$$\begin{aligned} x^{t+1} = \arg\min_{x \in \mathcal{K}} \{ f_t(x) := \quad & \frac{1}{2m}\|S^t A(x - x^t)\|_2^2 \\ & -x^T A^T (y - Ax^t)\}. \end{aligned} \qquad (6)$$

At the tth iteration of IHS a new sketch of the data matrix $S^t A$ and a full gradient $A^T(y - Ax^t)$ at the current estimate x^t is calculated to form a new sketched least-square problem. By repeating this procedure the IHS will converge to the solution of the original problem (1) in typically a small number of iterations. The iterative sketch essentially corrects the noise amplification and enables $(1+\epsilon)$ LS accuracy in the order of $\log \frac{1}{\epsilon}$ outer loop iterations.

2. Gradient Projection Iterative Sketch

2.1. The Proposed Algorithms

Here we consider the combination of CS with the first order PGD algorithm, the *Gradient Projection Classical Sketch* (GPCS):

$$x_{i+1} = \mathcal{P}_{\mathcal{K}}(x_i - \eta (S^0 A)^T (S^0 A x_i - S^0 y)). \qquad (7)$$

Similarly we obtain the *Gradient Projection Iterative Hessian Sketch* (GPIHS) for solving IHS (6):

$$x_{i+1} = \mathcal{P}_{\mathcal{K}}(x_i - \eta((S^t A)^T (S^t A)(x_i - x^t) + mA^T(Ax^t - y)). \qquad (8)$$

Our proposed GPIS algorithm applies PGD to solve a sequence of sketched LS, starting with a CS step for a fast initialization, and then is followed by further iterations of IHS. We can observe from Algorithm 1 that sketches are constructed in the outer loop and within the inner loop we only need to access them. This property could be very useful when, for instance A is stored in a slow speed memory and it is too large to be loaded at once into the fast memory, or in large scale image reconstruction problems such as CT where due to its prohibited size A is constructed on the fly. Note that thanks to the sketching each inner iteration of GPIS is $\frac{n}{m}$ times cheaper than a full PGD iterate in terms of matrix-vector multiplication, so intuitively we can see that there is potential in Algorithm 1 to get computational gain over the standard first order solver PGD.

Algorithm 1 Gradient Projection Iterative Sketch — $\mathcal{G}(m, [\eta], [k])$

```
Initialization: $x_0^0 = 0$
Given $A \in \mathbb{R}^{n \times d}$, sketch size $m \ll n$
Prior knowledge: the true solution $x$ belongs to set $\mathcal{K}$
Run GPCS iterates (Optional):
Generate a random sketching matrix $S^0 \in \mathcal{R}^{m \times n}$
Calculate $S^0 A$, $S^0 y$
for $i = 1$ to $k_0$ do
  $x_{i+1}^0 = \mathcal{P}_{\mathcal{K}}(x_i^0 - \eta_{0,i}(S^0 A)^T (S^0 A x_i^0 - S^0 y))$
end for
$x_0^1 = x_{k_0}^0$
Run GPIHS iterates
for $t = 1$ to $N$ do
  Calculate $g = A^T(Ax_0^t - y)$
  Generate a random sketching matrix $S^t \in \mathcal{R}^{m \times n}$
  Calculate $A_s^t = S^t A$
  for $i = 1$ to $k_t$ do
    $x_{i+1}^t = \mathcal{P}_{\mathcal{K}}(x_i^t - \eta_{t,i}(A_s^{t^T} A_s^t (x_i^t - x_0^t) + mg))$
  end for
  $x_0^{t+1} = x_{k_t}^t$
end for
```

Since it is well-known that in convex optimization the standard first order method Projected/proximal gradient descent can be accelerated by Nesterov's acceleration scheme (Nesterov, 2007) (Nesterov, 2013a) (Beck & Teboulle, 2009), our Algorithm 1 has potential to be further improved by introducing Nesterov's acceleration. Here we propose Algorithm 2 – *Accelerated Gradient Projection Iterative Sketch* (Acc-GPIS) which is based on the combination of the accelerated PGD and iterative sketching.

One of the benefits of deterministically minimising the sketched cost function can bring is that the implementation of the line-search scheme can be easy and provably reliable since the underlying sketched cost function each outer loop is fixed. For example (Nesterov, 2007) provides a simple line-search scheme for gradient methods to make the step size of each iteration to be nearly optimal, with rigorous

Algorithm 2 Accelerated Gradient Projection Iterative Sketch — $\mathcal{A}(m, [\eta], [k])$

```
Initialization: $x_0^0 = 0$, $\tau_0 = 1$
Given $A \in \mathbb{R}^{n\times d}$, sketch size $m \ll n$
Prior knowledge: the true solution $x$ belongs to set $\mathcal{K}$
Run GPCS iterates (Optional):
Generate a random sketching matrix $S^0 \in \mathcal{R}^{m\times n}$
Calculate $S^0 A$, $S^0 y$
for $i = 1$ to $k_0$ do
  $x_{i+1}^0 = \mathcal{P}_{\mathcal{K}}(z_i^0 - \eta_{0,i}(S^0A)^T(S^0Az_i^0 - S^0y))$
  $\tau_i = (1 + \sqrt{1 + 4\tau_{i-1}^2})/2$
  Extrapolate $z_{i+1}^0 = x_{i+1}^0 + \frac{\tau_{i-1}-1}{\tau_i}(x_{i+1}^0 - x_i^0)$
end for
$x_0^1 = z_0^1 = x_{k_0}^0$
Run GPIHS iterates
for $t = 1$ to $N$ do
  Calculate $g = A^T(Ax_0^t - y)$
  Generate a random sketching matrix $S^t \in \mathcal{R}^{m\times n}$
  Calculate $A_s^t = S^t A$
  $\tau_0 = 1$
  for $i = 1$ to $k_t$ do
    $x_{i+1}^t = \mathcal{P}_{\mathcal{K}}(z_i^t - \eta_{t,i}(A_s^{t^T}A_s^t(z_i^t - x_0^t) + mg))$
    $\tau_i = (1 + \sqrt{1 + 4\tau_{i-1}^2})/2$
    Extrapolate $z_{i+1}^t = x_{i+1}^t + \frac{\tau_{i-1}-1}{\tau_i}(x_{i+1}^t - x_i^t)$
  end for
  $x_0^{t+1} = z_0^{t+1} = x_{k_t}^t$
end for
```

convergence theory and also a explicit bound for the number of additional gradient calls. The line-search scheme is described by Algorithm 3. On the other hand in the stochastic gradient literature there are no practical strategies for efficient line search in the case of constrained optimization. To the best of our knowledge, only the SAG paper (Schmidt et al., 2013) addresses the issue of line-search but their implementation is only for unconstrained optimization.

3. Convergence Analysis

3.1. General Theory

We start our theoretical analysis by some definitions:

Definition 1. *The Lipschitz constant L and strong convexity μ for the LS (1) are defined as the largest and smallest singular values of the Hessian matrix A^TA:*

$$\mu\|z_d\|_2^2 \le \|Az_d\|_2^2 \le L\|z_d\|_2^2, \tag{9}$$

for all $z_d \in \mathbb{R}^d$, where $0 \le \mu < L$ ($\mu = 0$ means the LS (1) is non-strongly convex).

Definition 2. *Let $\mathcal{C}$ be the smallest closed cone at $x^\star$ containing the set $\mathcal{K} - x^\star$:*

Algorithm 3 line-search scheme for GPIS and Acc-GPIS — $\mathcal{L}(x_i, f_t(x), \nabla f_t(x_i), \gamma_u, \gamma_d)$ (Nesterov, 2007)

```
Input: update $x_i$, sketched objective function $f_t(x)$, gradient vector $\nabla f_t(x_i)$, line search parameters $\gamma_u$ and $\gamma_d$, step size of previous iteration $\eta_{i-1}$.
Define composite gradient map $m_L$:
$m_L := f_t(x_i) + (x - x_i)^T\nabla f_t(x_i) + \frac{1}{2\eta}\|x - x_i\|_2^2$
$\eta = \gamma_d\eta_{i-1}$
$x = \mathcal{P}_{\mathcal{K}}(x_i - \eta\nabla f_t(x_i))$
while $f_t(x) \ge m_L$ do
  $\eta = \eta/\gamma_u$
  $x = \mathcal{P}_{\mathcal{K}}(x_i - \eta\nabla f_t(x_i))$
end while
Return $x_{i+1} = x$ and $\eta_i = \eta$
```

$$\mathcal{C} = \left\{p \in \mathbb{R}^d |\, p = c(x - x^\star), \forall c \ge 0, x \in \mathcal{K}\right\}, \tag{10}$$

$\mathbb{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$, $\mathcal{B}^d$ be the unit ball in $\mathbb{R}^d$, z be arbitrary fixed unit-norm vectors in $\mathbb{R}^n$. The factors $\alpha(\eta, S^tA)$, $\rho(S^t, A)$ and $\sigma(S^t, A)$ are defined as:

$$\alpha(\eta_t, S^tA) = \sup_{u,v\in\mathcal{B}^d} v^T(I - \eta_t A^T S^{t^T} S^t A)u, \tag{11}$$

$$\rho(S^t, A) = \frac{\sup_{v\in A\mathcal{C}\cap\mathbb{S}^{n-1}} v^T(\frac{1}{m}S^{t^T}S^t - I)z}{\inf_{v\in A\mathcal{C}\cap\mathbb{S}^{n-1}} \frac{1}{m}\|S^tv\|_2^2}, \tag{12}$$

$$\sigma(S^t, A) = \frac{\sup_{v\in range(A)\cap\mathbb{S}^{n-1}} \|S^tv\|_2^2}{\inf_{v\in range(A)\cap\mathbb{S}^{n-1}} \|S^tv\|_2^2}, \tag{13}$$

For convenience, we denote each of this terms as: $\alpha_t := \alpha(\eta_t, S^tA)$, $\rho_t := \rho(S^t, A)$ and $\sigma_t := \sigma(S^t, A)$. Our theory hangs on these three factors and we will show that they can be bounded with exponentially high probabilities for Gaussian projections.

Definition 3. *The optimal points $x_\star^t$ of the sketch programs $f_t(x)$ are defined as:*

$$x_\star^t = \arg\min_{x\in\mathcal{K}} f_t(x). \tag{14}$$

We also define a constant R for the simplicity of the theorems:

$$R = \max_t \|x_\star^t - x^\star\|_2^2. \tag{15}$$

We use the notation $\|v\|_A = \|Av\|_2$ to describe the A-norm of a vector v in our theory. After defining these properties we can derive our first theorem for GPIS when $f(x)$ is strongly convex, e.g, $\mu > 0$:

Theorem 1. *(Linear convergence of GPIS when $\mu > 0$) For fixed step sizes $\eta_t \le \frac{1}{\|S^tA\|_2^2}$, the following bounds hold: for $t = 0$ (the initialization loop by GPCS),*

$$\|x_0^1 - x^\star\|_A \le (\alpha_0)^{k_0}\sqrt{\frac{L}{\mu}}\|x_0^0 - x_\star^0\|_A + 2\rho_0\|e\|_2, \tag{16}$$

for $N \geq 1$ and $x_0^t := x_{k_{t-1}}^{t-1}$ (the consecutive loops by GPIHS),

$$\|x_0^{N+1} - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t^\star\right\} \|x_0^1 - x^\star\|_A; \tag{17}$$

where we denote:

$$\rho_t^\star = (\alpha_t)^{k_t}\left[(1+\rho_t)\sqrt{\frac{L}{\mu}}\right] + \rho_t \tag{18}$$

From Theorem 1 we can see that when we have strong convexity, aka $\mu > 0$, by choosing a appropriate step size the GPCS loop will linearly converge to a sub-optimal solution, the accuracy of which depends on the value of $2\rho_0\|e\|_2$; and the following GPIHS iterations enjoys a linear convergence towards the optimal point.

When the least-squares solution is relatively consistent ($\|e\|_2$ is small), the GPCS loop will provide excellent initial convergence speed, otherwise it is not beneficial – that's why we say that the GPCS loop is optional for our GPIS / Acc-GPIS algorithm. For regression problems on data sets, we advise not to run the GPCS iterates, but for signal/image processing applications, we would recommend it.

For the cases where the strong convexity is not guaranteed ($\mu \geq 0$) we show the $\mathcal{O}(\frac{1}{k})$ convergence rate for GPIS algorithm:

Theorem 2. *(Convergence guarantee for GPIS when $\mu \geq 0$) If we choose a fixed number (k) of inner-loops for $t = 1, ..., N$, the following bounds hold: for $t = 0$,*

$$\|x_0^1 - x^\star\|_A \leq \sqrt{\frac{\beta L \sigma_0 R}{2k_0}} + 2\rho_0\|e\|_2, \tag{19}$$

for $N \geq 1$ and $x_0^t := x_k^{t-1}$

$$\|x_0^{N+1} - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t\right\} \|x_0^1 - x^\star\|_A + \frac{\max_t \sqrt{\sigma_t}}{1 - \max_t \rho_t}\sqrt{\frac{\beta L R}{2k}}, \tag{20}$$

where $\beta = 1$ for fixed step sizes $\eta_t = \frac{1}{\|S^t A\|_2^2}$, $\beta = \gamma_u$ for a line search scheme described by Algorithm 3 with parameter $\gamma_u > 1$ and $\gamma_d = 1$.

For the Accelerated GPIS algorithm we also prove the desired $\mathcal{O}(\frac{1}{k^2})$ convergence rate:

Theorem 3. *(Convergence guarantee for Accelerated GPIS when $\mu \geq 0$) If we choose a fixed number (k) of inner-loops for $t = 1, ..., N$, the following bounds hold: for $t = 0$,*

$$\|x_0^1 - x^\star\|_A \leq \sqrt{\frac{2\beta L \sigma_0 R}{(k_0+1)^2}} + 2\rho_0\|e\|_2, \tag{21}$$

for $N \geq 1$ and $x_0^t := x_k^{t-1}$

$$\|x_0^{N+1} - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t\right\} \|x_0^1 - x^\star\|_A + \frac{\max_t \sqrt{\sigma_t}}{1 - \max_t \rho_t}\sqrt{\frac{2\beta L R}{(k+1)^2}}, \tag{22}$$

where $\beta = 1$ for fixed step sizes $\eta_t = \frac{1}{\|S^t A\|_2^2}$, $\beta = \gamma_u$ for a line search scheme described by Algorithm 3 with parameter $\gamma_u > 1$ and $\gamma_d = 1$.

We include the proofs in our supplementary material. It is well known that for the case $\mu > 0$, the accelerated gradients can potentially enjoy the improved linear rate $\mathcal{O}((1 - \sqrt{\frac{\mu}{L}}))$ but it demands the exact knowledge of the value μ (which is often unavailable in practical setups). In our implementation for the Acc-GPIS method in the experiments, we use the adaptive *gradient restart* scheme proposed by (O'Donoghue & Candes, 2015).

3.2. Explicit Bounds for Gaussian Sketches

The theorems above provide us with a framework to describe the convergence of GPIS and Acc-GPIS in terms of the constants α, ρ and σ. For Gaussian sketches, these constants find explicit bounding expressions in terms of the sketch size m and the complexity of the constraint cone $\mathcal{C}$. For this, we use the Gaussian Width argument (see, e.g. (Chandrasekaran et al., 2012)):

Definition 4. *The Gaussian width $\mathcal{W}(\Omega)$ is a statistical measure of the size of a set Ω:*

$$\mathcal{W}(\Omega) = E_g\left(\sup_{v \in \Omega} v^T g\right), \tag{23}$$

where $g \in \mathbb{R}^n$ is draw from i.i.d. normal distribution.

The value of $\mathcal{W}(\mathcal{C} \cap \mathbb{S}^{d-1})$ is an useful measure of the tightness of the structure of $x^\star$. For example, if $x^\star$ is s-sparse and we model the sparsity constraint using an l_1 ball, we will have $\mathcal{W}(\mathcal{C} \cap \mathbb{S}^{d-1}) \leq \sqrt{2slog(\frac{d}{s}) + \frac{5}{4}s}$, which means the sparser $x^\star$ is, the smaller the $\mathcal{W}(\mathcal{C} \cap \mathbb{S}^{d-1})$ will be (Chandrasekaran et al., 2012). As an illustration we now quantify the bounds in our general theorems in terms of the sketch size m and the Gaussian width of the transformed cone $\mathcal{W}(A\mathcal{C} \cap \mathbb{S}^{n-1}) \leq \sqrt{d}$, and the ambient dimension of the solution domain (d). Now we are ready to provide the explicit bounds for the factors α_t, ρ_t and σ_t for the general theorems (we denotes $b_m := \sqrt{2}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})} \approx \sqrt{m}$ (Oymak et al., 2015) and $\mathcal{W} := \mathcal{W}(A\mathcal{C} \cap \mathbb{S}^{n-1})$ for the following lemmas):

Proposition 1. *If the step-size $\eta_t = \frac{1}{L(b_m + \sqrt{d} + \theta)^2}$, sketch size m satisfies $b_m > \sqrt{d}$, and the entries of the sketching* *matrix S^t are i.i.d drawn from Normal distribution, then:*

$$\alpha_t \leq \left\{1 - \frac{\mu}{L}\frac{(b_m - \sqrt{d} - \theta)^2}{(b_m + \sqrt{d} + \theta)^2}\right\}, \tag{24}$$

with probability at least $(1 - 2e^{-\frac{\theta^2}{2}})$.

Proposition 2. *If the entries of the sketching matrix S^t are i.i.d drawn from Normal distribution, then:*

$$\rho_t \leq \frac{m}{(b_m - \mathcal{W} - \theta)^2}\left(\frac{\sqrt{2}b_m(\mathcal{W} + \theta)}{m} + |\frac{b_m^2}{m} - 1|\right), \tag{25}$$

With probability at least $(1 - e^{-\frac{\theta^2}{2}})(1 - 8e^{-\frac{\theta^2}{8}})$.

Proposition 3. *If the entries of the sketching matrix S^t are i.i.d drawn from Normal distribution, and the sketch size m satisfies $b_m > \sqrt{d}$, then:*

$$\sigma_t \leq \frac{(b_m + \sqrt{d} + \theta)^2}{(b_m - \sqrt{d} - \theta)^2} \tag{26}$$

with probability at least $(1 - 2e^{-\frac{\theta^2}{2}})$.

(We include the proofs in the supplementary material.) We would like to point out that our bound on factor ρ_t in proposition 2 has revealed that the outer-loop convergence of GPIS and Acc-GPIS relies on the Gaussian Width of the solution $x^\star$ and the choice of the sketch size m:

$$\rho_t \lesssim \frac{\sqrt{2}\frac{\mathcal{W}}{\sqrt{m}}}{(1 - \frac{\mathcal{W}}{\sqrt{m}})^2}. \tag{27}$$

We can then observe that the larger the sketch size m is with respect to $\mathcal{W}$, the faster the outer loop convergence of GPIS and Acc-GPIS can be, but on the other hand we should not choose m too large otherwise the inner-loop iteration become more costly – this trade-off means that there is always a sweet spot for the choice of m to optimize the computation.

Our theory is conservative in a sense that it does not provide guarantee for a sketch size which is below the ambient dimension d since the factors α_t and σ_t which are related to the inner loop prohibit this.

Although the Gaussian sketch provides us strong guarantees, due to computational cost of dense matrix multiplication, which is of $\mathcal{O}(mnd)$, it is not computationally attractive in practice. In the literature of randomized numerical linear algebra and matrix sketching, people usually use the random projections with fast computational structures such as the Fast Johnson-Lindenstrauss Transform (Ailon & Liberty, 2008)(Ailon & Chazelle, 2009), Count sketch (Clarkson & Woodruff, 2013) and Count-Gauss sketch(Kapralov et al., 2016), which cost $\mathcal{O}(nd\log(d))$, $\mathcal{O}(nnz(A))$ and $\mathcal{O}(nnz(A) + m^{1.5}d^3)$ respectively. These fast sketching methods provide significant speed up in practice compared to Gaussian sketch when $n \gg d$.

4. Implementation for GPIS and Acc-GPIS in Practice

In this section we describe our implementation of GPIS and Acc-GPIS algorithm in the experiments:

- **Count sketch** In this paper we choose the Count Sketch as our sketching method since it can be calculated in a streaming fashion and we observe that this sketching method provides the best computational speed in practice. A MATLAB implementation for efficiently applying the Count Sketch can be found in (Wang, 2015).

- **Line search** We implement the line-search scheme given by (Nesterov, 2007) and is described by Algorithm 3 for GPIS and Acc-GPIS in our experiments with parameters $\gamma_u = 2$, and $\gamma_d = 2$.

- **Gradient restart for Acc-GPIS** We choose a efficient restarting scheme *gradient restart* proposed by (O'Donoghue & Candes, 2015).

5. Numerical Experiments

5.1. Settings for Environments and Algorithms

We run all the numerical experiments on a DELL laptop with 2.60 GHz Intel Core i7-5600U CPU and 1.6 GB RAM, MATLAB version R2015b.

We choose two recognized algorithms to represent the the full gradients methods and the (incremental) stochastic gradient method. For the full gradient, we choose the Accelerated projected gradient descent (Beck & Teboulle, 2009) (Nesterov, 2013b) with line-search method described in Algorithm 3 and gradient restart to optimize its performance. For the stochastic gradients we choose a mini-batched version of SAGA (Defazio et al., 2014) with various batch sizes ($b = 10$, $b = 50$ and $b = 100$). We use the step size suggested by SAGA's theory which is $\frac{1}{3\hat{L}}$. The code for the minibatch SAGA implementation can be found in (https://github.com/mdeff/saga). We get the estimated value for $\hat{L}$ by averaging the largest singular value of each batch (note that we do not count this into the elapsed time and epoch counts for SAGA). The sketch size of our proposed methods for each experiments are list in Table 1. We use the l_1 projection operator provided by the SPGL1 toolbox (Van Den Berg & Friedlander, 2007) in the experiments.

Table 1. Sketch sizes (m) for GPIS and Acc-GPIS for each experiments

SYN1	SYN2	SYN3	MAGIC04	YEAR
800	800	400	475	1000

5.2. Synthetic Data Sets

We start with some numerical experiments on synthetic problems (Table 2) to gain some insights into the algorithms. We begin by focusing on l_1 norm constrained problems [3]. We generate synthetic constrained least-square problems by first generating a random matrix sized n by d, then perform SVD on such matrix and replace the singular values with a logarithmically decaying sequence. (The details of the procedure can be found in supplementary materials.) Similarly we generate a synthetic problem (Syn3) for low-rank recovery with nuclear-norm constraint. This is also called the *multiple response regression* with a generalized form of the Least-squares:

$$X^\star = \arg \min_{\|X\|_\star \leq r} |||Y - AX|||_F^2. \tag{28}$$

5.3. Real Data Sets

We first run an unconstrained least-squares regression on the Year-prediction (Million-song) data set from UCI Machine Learning Repository (Lichman, 2013) after we normalize each column of the data matrix. We use this example to demonstrate our algorithms' performance in unconstrained problems.

Then we choose Magic04 Gamma Telescope data set from (Lichman, 2013) to generate a constrained Least-square regression problem. The original number of features for Magic04 are 10 , and we normalize each columns of the original data matrix and additional irrelevant random features as the same way as the experiments in (Langford et al., 2009)(Shalev-Shwartz & Tewari, 2011) to the data sets so that the regressor $x^\star$ can be chosen to select the sparse set of relevant features by again solving (1). For this case we first precalculate the l_1-norm of the original program's solution and then set it as the radius of our l_1 constraint. The details of the real data sets can be found in Table 3.

5.4. Discussion

We measure the performance of the algorithms by the wall-clock time (simply using the tic toc function in MATLAB)

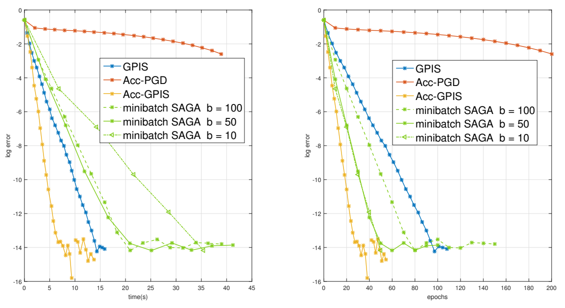

Figure 1. Experimental results on Million-song Year prediction data set (unconstrained LS regression experiment)

and the epoch counts. The y-axis of each plot is the relative error $\log(\frac{f(x)-f(x^\star)}{f(x^\star)})$. The values below 10^{-10} are reported as exact recovery of the least-square solution.

In all the experiments, our methods achieve the best performance in terms of wall-clock time. We show that in many cases the sketched gradient methods can outperform leading stochastic gradient methods. Both sketched gradients and stochastic gradients can achieve reduced complexity compared to the (accelerated) full gradient method, but since the sketched method has inner-loops with deterministic iterations, the line-search scheme of the classic gradient descent method can be directly used to make each iteration's step size be near optimal, and unlike the stochastic gradient, our methods do not need to access new mini-batches from memory each iteration, which can save operational time in practice.

SAGA performs competitively in terms of epoch counts (right hand figures) which is generally achieved using a small batch size of 10. Unfortunately the additional cost of the projection per iteration can severely impact on the wall clock time performance[4]. The experiment on Syn1 and Syn2 are similar but in Syn2 we put the constraint on a dictionary U, hence in Syn2 the projection operator has an additional cost of performing such orthogonal transform. In Syn1's wall-clock time plot we can see that SAGA with $b = 10$ has the fastest convergence among all the batch size choices, but in Syn2 it becomes the worst batch size choice for SAGA since it demands more iterations and hence more calls on the projection operator. In Syn3 we have a more expensive projection operator since our constraint is on the nuclear-norm of a matrix $X \in \mathbb{R}^{100\times 100}$, and we can observe that the real convergence speed of SAGA with $b = 10$ become much slower than any other methods in terms of

[3]In practice we would consider the l_1 regularized least squares, but in this paper we focus on the constrained case to make simple illustrations.

[4]For the unconstrained case (Million-song data set, sized 5×10^5 by 90), we also observe that, SAGA with $b = 10$ is unattractive in wall-clock time since it does not benefit from the vectorized operation of MATLAB as larger choices of batch size and takes too many iterations.

Table 2. Synthetic data set settings. (*) U denotes the dense dictionary which is a orthogonal transform. (**) s denotes sparsity or rank of the ground truth

DATA SET	SIZE	$(**)s$	$\frac{L}{\mu}$	Φ
SYN1	(100000, 100)	10	10^7	I
SYN2	(100000, 100)	10	10^7	(*)U
SYN3 (LOW RANK)	(50000, 100)	5	10^4	-

Table 3. Chosen data sets for Least-square regression, RFs: number of relevant features

DATA SET	SIZE	RFS	Φ
YEAR	(500000, 90)	90	-
MAGIC04	(19000, 10 + 40)	10	I

wall-clock time. In this scenario the full gradient method is much more competitive. However even here as the error reduces the sketched gradient methods exhibit a computational advantage.

6. Conclusions

We propose two sketched gradient algorithms GPIS and Acc-GPIS for constrained Least-square regression tasks. We provide theoretical convergence analysis of the proposed algorithms for general sketching methods and high probability concentration bounds for the Gaussian sketches. The numerical experiments demonstrates that for dense large scale overdetermined data sets our sketched gradient methods performs very well compares to the stochastic gradient method (mini-batch) SAGA and the Accelerated full gradient method in terms of wall-clock time thanks to the benefits of sketched deterministic iterations, the efficient implementation of the Count-sketch and the use of aggressive line-search methods.

Acknowledgements

JT, MG and MD would like to acknowledge the support from H2020-MSCA-ITN Machine Sensing Training Network (MacSeNet), project 642685; EPSRC Compressed Quantitative MRI grant, number EP/M019802/1; and ERC Advanced grant, project 694888, C-SENSE, respectively. MD is also supported by a Royal Society Wolfson Research Merit Award. The authors also give thanks to the anonymous reviewers for insightful comments.

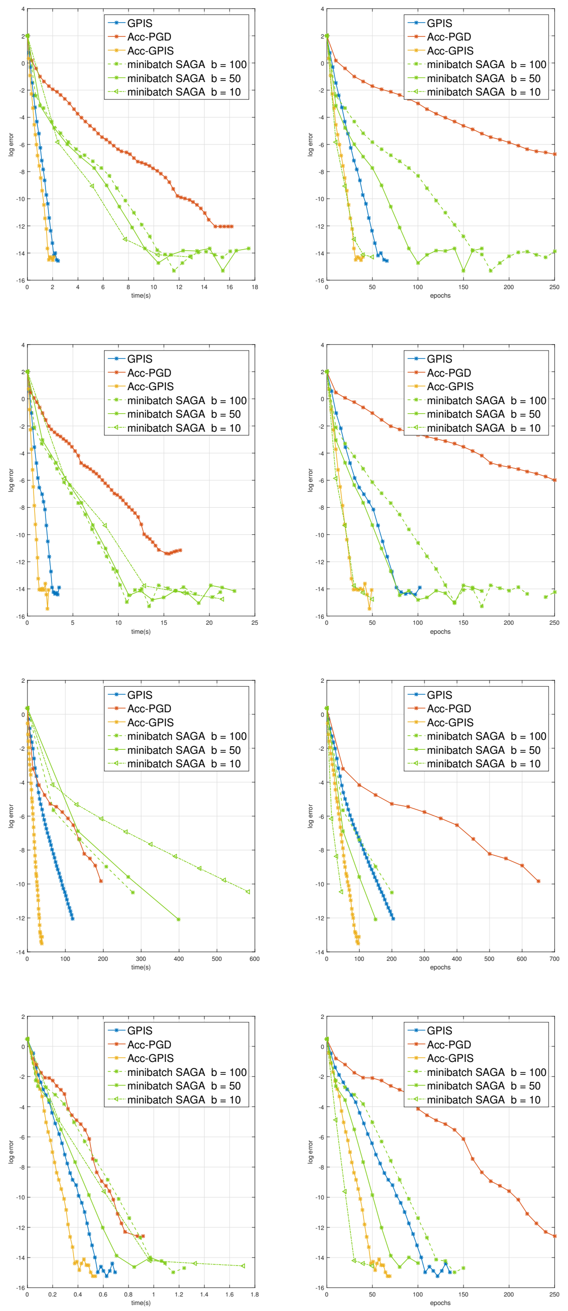

Figure 2. Experimental results on (from top to button) Syn1, Syn2, Syn3 and Magic04 data sets. The left column is for wall-clock time plots, while the right column is for epoch counts

References

Ailon, N. and Chazelle, B. The fast johnsonlindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

Ailon, N. and Liberty, E. Fast dimension reduction using rademacher series ondual bch codes. *Discrete & Computational Geometry*, 42(4):615–630, 2008.

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3): 253–263, 2008.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Candes, E., Romberg, J., and Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

Chandrasekaran, V. and Jordan, M. I. Computational and statistical tradeoffs via convex relaxation. *Proceedings of the National Academy of Sciences*, 110(13):E1181–E1190, 2013.

Chandrasekaran, V., Recht, B., Parrilo, P. A., and Willsky, A. S. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849, 2012.

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 81–90. ACM, 2013.

Dasgupta, S. and Gupta, A. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Donoho, D. L. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

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

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pp. 315–323. Curran Associates, Inc., 2013.

Johnson, W. B. and Lindenstrauss, J. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Kapralov, M., Potluru, V. K., and Woodruff, D. P. How to fake multiply by a gaussian matrix. *arXiv preprint arXiv:1606.05732*, 2016.

Konečný, J. and Richtárik, P. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

Konečný, J., Liu, J., Richtárik, P., and Takáč, M. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

Langford, J., Li, L., and Zhang, T. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Mahoney, M. W. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3 (2):123–224, 2011.

Nesterov, Y. Gradient methods for minimizing composite objective function. Technical report, UCL, 2007.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013a.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013b.

O'Donoghue, B. and Candes, E. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

Oymak, S., Recht, B., and Soltanolkotabi, M. Sharp time–data tradeoffs for linear inverse problems. *arXiv preprint arXiv:1507.04793*, 2015.

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

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

Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, pp. 1–30, 2013.

Shalev-Shwartz, S. and Tewari, A. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun):1865–1892, 2011.

Van Den Berg, E. and Friedlander, M. P. Spgl1: A solver for large-scale sparse reconstruction, 2007.

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

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.