AMSI VACATIONRESEARCH SCHOLARSHIPS 2022–23

SET YOUR SIGHTS ON RESEARCH THIS SUMMER

Randomization Techniques for Large-Scale Optimization

Xiaoyu Li

Supervised by Prof. Guoyin Li

University of New South Wales

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Contents

1 Prelude

1.1 Abstract

The interest in developing efficient numerical methods for large-scale optimization problems has recently grown remarkably due to their applications in diverse practical problems, including machine learning, statistics, and data science. Recent works have shown that the introduction of randomization techniques, such as random shuffling in optimization algorithms, would result in more efficient and accurate results. However, the mathematical justification of this is still lacking, which serves as an obstruction to understanding the randomization techniques and developing new and efficient numerical methods. The purpose of this report is to summarize the current state-of-the-art of randomization techniques used for large-scale optimization and to perform some insight through methods with numerical experiments.

1.2 Acknowledgements

I would like to thank Prof. Guoyin Li for introducing me to the fascinating world of optimization and providing consistent guidance to me. I would also like to thank the Australian Mathematical Sciences Institute and the School of Mathematics and Statistics at the University of New South Wales for this invaluable opportunity.

1.3 Statement of Authorship

This project aims to summarize the current development of randomization techniques for large-scale optimization and conduct some numerical experiments to validate their performance. While the report provides a review of the state-of-the-art of this this fast developing area and do not contain new original results due to the time limitation, these study will provide a stepping stone for my up coming honours projects, and will be further exploited.

2 Introduction

The interest in large-scale optimization methods has recently grown remarkably due to their applications in diverse practical problems, including machine learning, statistics, and data

science. The typical numerical optimization algorithm to deal with large-scale optimization problems is **Stochastic Gradient Descent (SGD)**, proposed by Robbins and Monro (1951). It is a simple but very efficient iterative method often used to solve finite-sum optimization problems of the form

$$\min_{x\in\mathbb{R}^d}\left[f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\right]. \tag{1}$$

Here, for each $i = 1, \ldots, n$, each $f_i : \mathbb{R}^d \to \mathbb{R}$ is differentiable and smooth, and the number of functions n is usually large. There are many problems in machine learning where this model fits, such as using a neural network for classification, where we have the following loss function:

$$\min_{w} \frac{1}{n}\sum_{i=1}^{n} \phi_i(h(w; i)), \tag{2}$$

where $\{(x_i, y_i)\}_{i=1}^n$ is the given training data, $h(\cdot, i), i = 1, \ldots, n$ and and ϕ_i are the corresponding classifier and loss function for each data (x_i, y_i), and w is the trainable parameter of the classifier. Note that, in many typical applications, the number of training data n is usually a larger number with several millions.

The reason why SGD is a popular algorithm to solve such large-scale problems is due to its inherent advantages, such as scalability, low computational complexity, and mild memory requirements (Bottou et al., 2018). Many recent works have studied the sampling-without-replacement variant of SGD, known as **Random Reshuffling (RR)**, and have shown that it often outperforms SGD (Gürbüzbalaban et al., 2021; Mishchenko et al., 2020; Safran and Shamir, 2020; Haochen and Sra, 2019). Compared to SGD, RR uses sampling without replacement to leverage the finite-sum structure of (1), so it usually converges faster than SGD emperically. However, it also introduces conditionally unbiased gradients in iterations within each epoch. De Sa (2020) provides an example of a learning task and algorithm for which SGD performs better than RR and counterexamples to the Operator Inequality of Noncommutative Arithmetic and Geometric Means, a longstanding conjecture proposed by (Recht and Ré, 2012) that has a close relationship with the performance of RR.

This report will introduce the basic concepts of optimization and gradient descent algorithms, investigate recent advancements in RR and other randomization techniques, and present numerical experiments demonstrating the performance of RR compared to SGD and its other

variants.

3 Preliminaries

3.1 Gradient Descent

Gradient Descent (GD) is a common numerical method to solve unconstrained minimization problems that have a differentiable objective function. The idea of GD is to choose an initial starting point $x_0 \in \mathbb{R}^d$ and then iteratively move to the next point in the direction opposite to the gradient by the stepsize η_k, which is the direction that locally decreases f the fastest.

Algorithm 1: Gradient Descent

Data: Stepsize $\eta > 0$, starting point $x_0 \in \mathbb{R}^d$
$x := x_0$; **while** *some stopping criteria not satisfied* **do**
  $x := x - \eta \nabla f(x)$
end

The advantages of GD are simplicity, computational efficiency, and low memory storage. However, when we deal with large-scale problems where f takes the form as in (1) with the number of component functions N is very large in our problem setting, computing the gradient of the objective function can be very expansive.

3.2 Stochastic Gradient Descent

To address the drawback of GD, instead of computing the gradient of the objective function, **Stochastic Gradient Descent (SGD)** approximates it by computing the gradient for a randomly selected component function. A typical SGD can be formulated as follows.

Algorithm 2: Stochastic Gradient Descent

Data: Stepsize $\eta > 0$, starting point $x_0 \in \mathbb{R}^d$, number of epochs $k > 0$
$x := x_0$; **for** $t = 1, 2, \ldots, k$ **do**
  for $j = 1, 2, \ldots, n$ **do**
    Sample $i \in 1, 2, \ldots, n$ uniformly; $x := x - \eta \nabla f_i(x)$;
  end
  $x_t := x$
end

Since SGD only computes the gradient of one component function, it is computationally more efficient than GD, especially for machine learning problems with large datasets. We also note that SGD has other advantages, such as better generalization performance (Amir et al., 2021; Zhou et al., 2020), faster convergence (Vaswani et al., 2019), and handling non-convex problems (Jin et al., 2021). However, one potential drawback of SGD is that it does not use the finite-sum structure of the objective function in (1).

4 Randomization Techniques

4.1 SGD Variants

The standard SGD is often called SGD with replacement sampling, which means that at each time, we randomly select a component function from all samples and then replace each sample after it has been selected, so that it is possible for the same sample to be selected multiple times. However, in practice, SGD without replacement sampling is more widely used for training a machine learning model. We will see that these heuristics are simple but powerful and can improve the performance of SGD.

We first introduce the **Incremental Gradient method (IGM)**, which is actually not a random but a deterministic algorithm. It performs incremental gradient steps through each component function in a fixed order.

Algorithm 3: Incremental Gradient Method

Data: Stepsize $\eta > 0$, starting point $x_0 \in \mathbb{R}^d$, number of epochs $k > 0$
$x := x_0$;
for $t = 1, 2, \ldots, k$ **do**
  for $j = 1, 2, \ldots, n$ **do**
    $x := x - \eta \nabla f_j(x)$;
  end
  $x_t := x$
end

Another variant of SGD is called **Shuffling Once (SO)**, which shuffles the order of component functions before performing iterations, and then uses the same permutation for all k

epochs. We define

$$\Gamma = \{\sigma : \sigma \text{ is a permutation of } \{1, 2, ..., n\}\}.$$

Algorithm 4: Shuffling Once

Data: Stepsize $\eta > 0$, starting point $x_0 \in \mathbb{R}^d$, number of epochs $k > 0$
$x := x_0$;
Sample a permutation σ from Γ uniformly;
for $t = 1, 2, \ldots, k$ **do**
  for $j = 1, 2, \ldots, n$ **do**
    $x := x - \eta \nabla f_{\sigma(j)}(x)$;
  end
  $x_t := x$
end

Finally, we introduce **Random Reshuffling (RR)**. Instead of sampling a permutation before performing iterations, it shuffles the component functions for each epoch, which means we have different order of gradient steps in each epoch.

Algorithm 5: Random Reshuffling

Data: Stepsize $\eta > 0$, starting point $x_0 \in \mathbb{R}^d$, number of epochs $k > 0$
$x := x_0$;
for $t = 1, 2, \ldots, k$ **do**
  Sample a permutation σ from Γ uniformly;
  for $j = 1, 2, \ldots, n$ **do**
    $x := x - \eta \nabla f_{\sigma(j)}(x)$;
  end
  $x_t := x$
end

4.2 Theoritical Lower and Upper Bounds

Before jumping into the main topic, we first briefly describe the roles played by the upper and lower bounds of the optimization algorithm. An upper bound of an algorithm often refers to the theoretical guarantees on the convergence rate of the algorithm to a minimum, which typically depends on the characteristics of the objective function and the hyperparameters used in the algorithm. On the other hand, a lower bound describes the theoretical limitations of the algorithm in terms of the convergence rate and performance guarantees. It usually provides a

lower bound on the distance from the current iterated point to the optimizer or the number of iterations required to reach a certain accuracy, which can help determine the best performance of the algorithm. In this section, we introduce some state-of-the-art results from recent papers. Most recent works restrict the objective function to a quadratic finite-sum function on $\mathbb{R}$, which facilitates the analysis of randomization techniques. The main reference is *How good is sgd with random shuffling?* (Safran and Shamir, 2020).

The lower bounds constructed by Safran and Shamir (2020) are built on the following assumptions:

Assumption 4.1 *f is a λ-strongly convex quadratic finite sum function on $\mathbb{R}$. Each f_i is quadratic and convex and of the form $f_i(x) = a_i x^2 + b_i x$, λ-smooth, and is G-Lipschitz for any x such that $\|x - x^*\| \leq 1$ around x^* is the unique minimizer of f. The algorithms are initialized at a start point x_0 such that $\|x_0 - x^*\| \leq 1$. The chosen stepsize η is constant.*

For this assumption, here are a few comments: we note that λ is not only the gradient Lipschitz constant but also the strong convexity parameter, which leads to a constant condition number. The distance from the starting point to the optimizer is assumed to be at most 1, since the current upper bounds for SGD on strongly convex functions often do not explicitly rely on the the initial distance, for both SGD and RR.

Now we begin to discuss lower bounds for these randomization techniques. To begin with, we will first consider the **Incremental Gradient Method (IGM)**, which involves cycling through component functions in a fixed deterministic order.

Theorem 4.2 (Safran and Shamir, 2020) For any $k \geq 1, n > 1$, and positive numbers G, λ such that $G \geq 6\lambda$, there exists a function f on $\mathbb{R}$ and an initialization point $x_0 \in \mathbb{R}^d$ satisfying Assumption 4.1, such that if we run IGM for k epochs with any step size $\eta > 0$, then

$$f(x^k) - \inf_x f(x) \geq c \cdot \min\left\{\lambda, \frac{G^2}{\lambda k^2}\right\}. \tag{3}$$

It is known that the existing optimal bound (both upper and lower bound) of SGD (Shamir, 2011) is

$$\Theta\left(\frac{G^2}{\lambda n k}\right).$$

Hence IGM can beat SGD only when $k > n$. However, when dealing with large-scale problems, n is often too large to allow for a larger value of k.

Next, we turn to the case of SO, where the components functions are shuffled before cycling through those functions, and we use that same order for all epochs.

Theorem 4.3 (Safran and Shamir, 2020) For any $k \geq 1, n > 1$, and positive G, λ such that $G \geq 6\lambda$, there exists a function f on $\mathbb{R}$ and an initialization point $x_0 \in \mathbb{R}$ satisfying Assumption 1, such that if we run SO for k epochs with any step size $\eta > 0$, then

$$\mathbb{E}\left[f(x^k) - \inf_x f(x)\right] \geq c \cdot \min\left\{\lambda, \frac{G^2}{\lambda n k^2}\right\}. \tag{4}$$

We note that the lower bound of SO implies an interesting result in comparing SGD and IGM, SO can achieve a lower distance from the value of the function at the iterated point to the optimal value after performing the same number of epochs.

Finally, we will see that the lower bound of RR can be much less than all other methods metioned above.

Theorem 4.4 (Safran and Shamir, 2020) For any $k \geq 1, n > 1$, and positive G, λ such that $G \geq 6\lambda$, there exists a function f on $\mathbb{R}$ and an initialization point x_0 satisfying Assumption 1, such that if we run RR for k epochs with any step size $\eta \geq 0$, then

$$\mathbb{E}\left[f(x^k) - \inf_x f(x)\right] \geq c \cdot \min\left\{\lambda, \frac{G^2}{\lambda}\left(\frac{1}{(nk)^2} + \frac{1}{nk^3}\right)\right\}. \tag{5}$$

We observed that although there are two terms here, but both of them are higher order term of the lower bounds other methods. This suggests that RR can be better than SGD, IGM, and SO in terms of the lower bounds analysis, which is consistent with empirical evidence.

There are also some results give tight upper bounds for SO and RR but also with stronge assumptions.

Assumption 4.5 *The objective function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ is λ-strongly convex on $\mathbb{R}$. In addition, each component function $f_i(x) = \frac{a_i}{2}x^2 - b_i x$ is convex, L-smooth, and satisfies $|f_i'(x^*)| \leq G$ where x^* is the global minimizer of f over $\mathbb{R}^d$.*

For SO, the following theorem provides some upper bound analysis for the function value convergence rate.

Theorem 4.6 (Safran and Shamir, 2020) Suppose that $f(x) \coloneqq \sum_{i=1}^{n} f_i(x)$ satisfies Assumption 4.5, and assume that $\frac{L}{\lambda} \leq \frac{nk}{\log(n^{0.5}k)}$. Then SO with a fixed stepsize of $\eta = \frac{\log\left(n^{0.5}k\right)}{\lambda nk}$ satisfies

$$\mathbb{E}\left[f\left(x_k\right) - \inf_x f(x)\right] \leq \tilde{O}\left(\frac{\lambda}{nk^2}\left(x_0 - x^*\right)^2 + \frac{G^2L^2}{\lambda^3 nk^2}\right)$$

where the expectation is taken over randomly sampling a permutation $\sigma \in \Gamma$ uniformly, and $\tilde{O}$ hides a constant and factors poly-logarithmic in n and k.

For RR, the following theorem gives its upper bound:

Theorem 4.7 (Safran and Shamir, 2020) Suppose that $f(x) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ satisfies Assumption 4.5, and assume that $\frac{L}{\lambda} \leq \frac{k}{2\log(nk)}$. Then RR with a fixed step size of $\eta = \frac{\log(nk)}{\lambda nk}$ satisfies

$$\mathbb{E}\left[f\left(x_k\right) - \inf_x f(x)\right] \leq \tilde{O}\left(\frac{\lambda}{n^2k^2}\left(x_0 - x^*\right)^2 + \frac{G^2L^2}{\lambda^3}\left(\frac{1}{n^2k^2} + \frac{1}{nk^3}\right)\right),$$

where the expectation is taken over randomly sampling a permutation $\sigma \in \Gamma$ uniformly, and $\tilde{O}$ hides a constant and factors poly-logarithmic in n and k.

Verifying that these upper bounds of SO and RR are consistent with our lower bounds of SO and RR in Theorems 4.3 and 4.4 in terms of their dependence on the variables n and k is straightforward. Although the assumption of univariate quadratics is limiting, Theorems 4.6 and 4.7 here demonstrate the possible accuracy of lower bounds and to clarify how *RR* can lead to a quicker convergence in a simple scenario.

Algorithm	Lower bound	Upper bound
RR	$1/(nk)^2 + 1/nk^3$ (Thm 4.4)	$1/(nk)^2 + 1/nk^3$ (1d quadratic case, Thm 4.7)
SO	$1/nk^2$ (Thm 4.3)	$1/nk^2$ (1d quadratic, Thm 4.6)
IGM	$1/k^2$ (Thm 4.2)	$1/k^2$ (Gurbuzbalaban et al., 2019)
SGD	$1/nk$	$1/nk$

Table 1: Summary of lower and upper bounds

4.3 Comparing RR with SGD

In the previous section, we compare the expected optimization error $\mathbb{E}[f(x_k) - \inf f(x)]$ between SGD and RR. In this section, we introduce an concept called shuffling variance propsed by Mishchenko et al. (2020), which can be used to compare the error $\mathbb{E}[\|x_k - x^*\|]$, where $x^* = \arg\min f(x)$, between SGD and RR.

Here, the following analysis is built on assumption:

Assumption 4.8 *The objective function $f = \sum_{i=1}^{n} f_i$ is convex, and f and each component function f_i are L-smooth. Further, there exists $f^* \in \mathbb{R}$ such that $f \geq f^*$. We also assume the existence of a minimizer $x^* \in \mathbb{R}^d$.*

Now we definte the Bregman divergence and shuffling variance.

Definition 4.9 (Bregman Divergence) (Mishchenko et al., 2020) For any i, the quantity

$$D_{f_i}(x, y) \coloneqq f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y \rangle$$

is said to be the **Bregman Divergence** between x and y associated with f_i.

Definition 4.10 (Shuffling Variance) (Mishchenko et al., 2020) Given a stepsize $\eta > 0$, and a random permutation σ of $\{1, 2, \ldots, N\}$. The **shuffling variance** is given by

$$\sigma^2_{\text{Shuffle}} \coloneqq \max_{i=2,\ldots,N} \left[\frac{1}{\eta} \mathbb{E}\left[D_{f_{\sigma(i)}}(x_i^*, x^*) \right] \right] \tag{6}$$

where $x_i^* = x^* - \eta \sum_{j=0}^{i-1} \nabla f_j(x^*)$.

The shuffling variance $\sigma^2_{\text{Shuffle}}$ depends on component functions $f_1, f_2, \ldots, f_n$, and also on the stepsize η, in a more complicated way than in SGD. In fact, the shuffling variance is analogous to the conventional definition of variance employed in the analysis of SGD. The following theorem shows the relation between them.

Theorem 4.11 (Mishchenko et al., 2020) Suppose that each f_i is λ-strongly convex and L-smooth. Then

$$\frac{\eta \lambda n}{8} \sigma_*^2 \leq \sigma^2_{\text{Shuffle}} \leq \frac{\eta \lambda n}{4} \sigma_*^2 \tag{7}$$

where $\sigma_*^2 = \frac{1}{N}\sum_{i=1}^{N} \|\nabla f_i(x_*)\|^2$.

Now we are going to give the main result proved by Mishchenko et al. (2020)

Theorem 4.12 (Mishchenko et al., 2020) Suppose that each f_i is λ-strongly convex and that Assumption 4.8 holds. Then for RR or SO run with a constant stepsize $\eta \leq \frac{1}{L}$, the iterates generated by either algorithms satisfy

$$\mathbb{E}\left[\|x_k - x^*\|^2\right] \leq (1-\eta\lambda)^{nk}\|x_0 - x^*\|^2 + \frac{2\eta\sigma_{\text{Shuffle}}^2}{\lambda}. \tag{8}$$

Several previous works have shown (Needell et al., 2014; Stich, 2019) that for any $\eta < \frac{1}{2L}$ the iterates of SGD satisfy

$$\mathbb{E}\left[\|x_k - x^*\|^2\right] \leq (1-\eta\lambda)^{nk}\|x_0 - x^*\|^2 + \frac{2\eta\sigma_*^2}{\lambda}. \tag{9}$$

By inequalities 8 and 9, the question reduces to comparing the value of σ_*^2 and $\sigma_{\text{Shuffle}}^2$. According to Theorem 4.11, it depends on both the number of component functions n and the stepsize η. Once the stepsize is sufficiently small, $\sigma_{\text{Shuffle}}^2$ becomes smaller than σ_*^2, but this might not be true in general. If we partition n component functions into n/τ groups, i.e., use minibatches of size τ, then σ_*^2 decreases as $O(1/\tau)$ and $\sigma_{\text{Shuffle}}^2$ as $O(1/\tau^2)$, so RR can become faster even without decreasing the stepsize.

5 Numerical Experiments

Finally, we conduct some numerical experiments to validate the performance of these randomization techniques. We implement these algorithms based on the literature and test them via the least square optimization problem with the form

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|^2 := \sum_{i=1}^{m}(a_i^T x - b_i)^2 \tag{10}$$

where $A \in \mathbb{R}^{m\times n}$ and a_i is the i-th row of the matrix A, to solve the linear system on the following two types of dataset for our test:

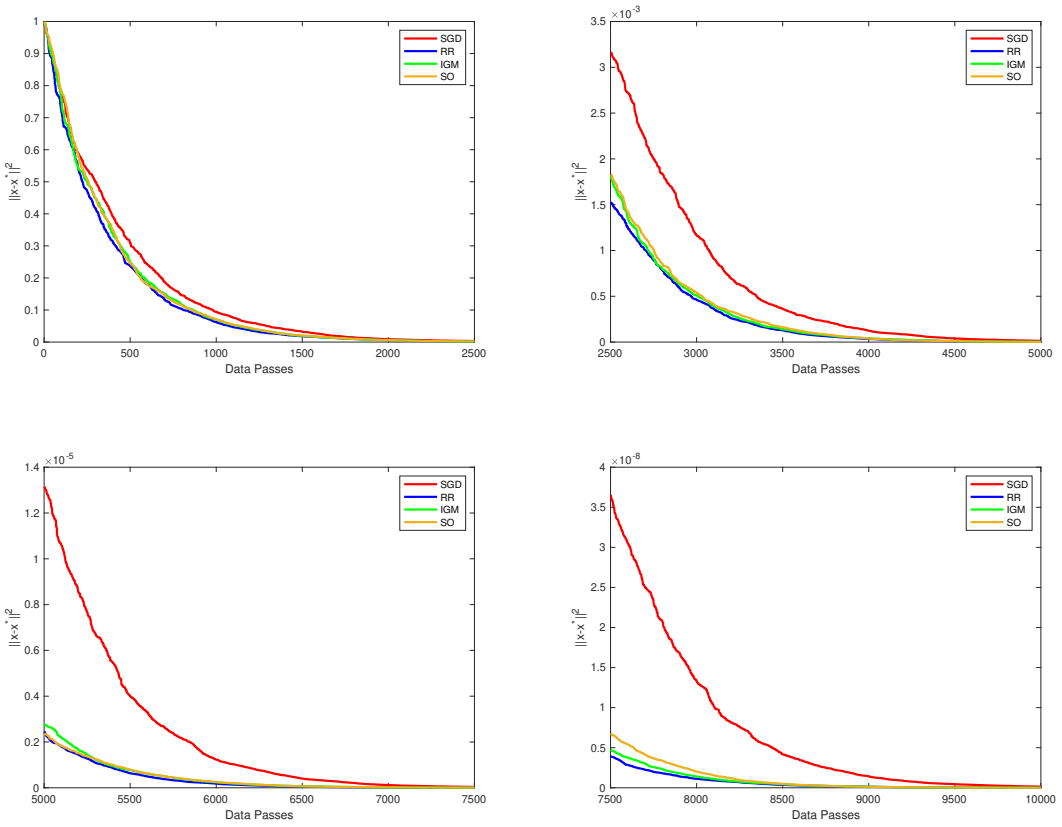

Figure 1: Experiment results on the synthetic data

Synthetic data. The synthetic data used in this experiment are generated using the MATLAB function randn. We used the MATLAB function randn(m,n) to generate the matrix A, and then modified its singular values using singular value decomposition to get an ill-conditioned matrix. The exact solution x^* was generated as follows:

$$x^* = \frac{A^\top w}{\|A^\top w\|_2} \tag{11}$$

where w was generated using the MATLAB function randn(m, 1), and $\|\cdot\|_2$ is the usual Euclidean distance. The right-side vector $b \in \mathbb{R}^m$ was set to $b = Ax^*$.

Real data. The real-world data used are accessible through the SuiteSparse Matrix Collection (Kolodziej et al., 2019). Each dataset in the SuiteSparse Matrix Collection comprises a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, but we only used the matrices A in our experiments

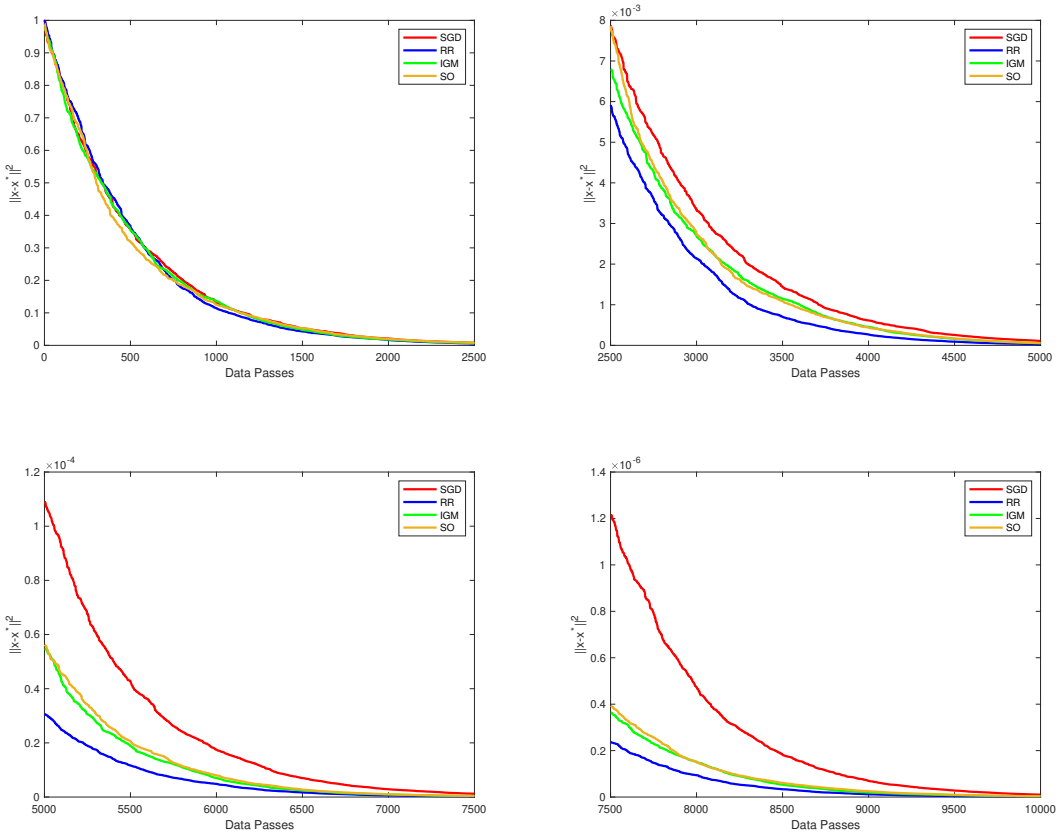

Figure 2: Experiment results on the real data

and disregarded the vector b. As in the synthetic data experiments, we created the solution using Equation 11 to guarantee the consistence of the linear system, and then set $b = Ax^*$.

We run each algorithm ten times and record the average, regardless of whether it is synthetic data or real data, we can see that adding randomization techiniques can significantly improve the effect of SGD, making the algorithm can approach the optimizer with fewer steps, where random shuffling has the best effect, consistent with the theoretical results.

We also record the averaged runtime of each algorithms. In terms of runtime, IGM has no randomness, it performs gradient steps in a fixed order each time, so it takes less time, also SO only component shuffle the functions once and takes less time. However, SGD randomly picks the component function once at each perform gradient step, while RR shuffles once before each epoch, so they both take longer, but in general, this is comparable.

Algorithm	Runtime(sec)
SGD	0.3078
IG	0.2158
SO	0.2092
RR	0.3133

(a) synthetic data

Algorithm	Runtime(sec)
SGD	0.5776
IG	0.4429
SO	0.4450
RR	0.6835

(b) real data

Figure 3: Runtime of algorithms

6 Conclusion and Future Study

Random Reshuffling is a simple but powerful technique. Simplicity means it is easily implemented in practice, and its power leads significantly improved performance when optimizing large-scale problems. By randomly shuffling the data component functions in SGD, we can reduce the correlations among successive gradients, which in turn leads to faster convergence and improved accuracy. However, despite its apparent benefits, the theoretical properties of this approach are not well-understood. There are several open questions that require further investigation to achieve a deeper understanding of the efficacy and limitations of Random Reshuffling.

One important area of inquiry is the development of tighter upper bounds for Random Reshuffling without relying on overly strong assumptions. While existing theoretical analyses have demonstrated the effectiveness of this approach, the assumptions made in these studies are quite stringent. Finding ways to improve the performance of Random Reshuffling while reducing the number and severity of assumptions would be a significant advancement in the field.

Another promising avenue for future research is the exploration of alternative data permutations that may be more effective than random reshuffling in achieving faster convergence rates or other performance metrics. For example, Chen et al. (2021) proposes a new algorithm called GraB that aims to find better provable data permutations than RR to enhance the performance of SGD. A thorough analysis of various data permutation techniques, and their theoretical properties, would provide valuable insights into the underlying mechanisms that contribute to faster convergence and better accuracy in SGD.

In summary, while Random Reshuffling is a simple but powerful technique for improving the

performance of SGD on large-scale optimization problems, its theoretical analysis is still incomplete. To fully realize the potential of this approach, further research is required to investigate its limitations and improve its performance on various types of optimization problems.

References

I. Amir, T. Koren, and R. Livni. Sgd generalizes better than gd (and regularization doesn't help). In *Conference on Learning Theory*, pages 63–92. PMLR, 2021.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

G. Chen, D. Li, and H. Li. Grab: Finding provably better data permutations than random reshuffling. In *International Conference on Machine Learning*, pages 1737–1747. PMLR, 2021.

C. M. De Sa. Random reshuffling is not always better. *Advances in Neural Information Processing Systems*, 33:5957–5967, 2020.

M. Gurbuzbalaban, A. Ozdaglar, and P. A. Parrilo. Convergence rate of incremental gradient and incremental newton methods. *SIAM Journal on Optimization*, 29(4):2542–2565, 2019.

M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, 186:49–84, 2021.

J. Haochen and S. Sra. Random shuffling beats sgd after finite epochs. In *International Conference on Machine Learning*, pages 2624–2633. PMLR, 2019.

C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29, 2021.

S. P. Kolodziej, M. Aznaveh, M. Bullock, J. David, T. A. Davis, M. Henderson, Y. Hu, and R. Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.

K. Mishchenko, A. Khaled, and P. Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33:17309–17320, 2020.

D. Needell, R. Ward, and N. Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Advances in neural information processing systems*, 27, 2014.

B. Recht and C. Ré. Toward a noncommutative arithmetic-geometric mean inequality: Conjectures, case-studies, and consequences. In *Conference on Learning Theory*, pages 11–1. JMLR Workshop and Conference Proceedings, 2012.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

I. Safran and O. Shamir. How good is sgd with random shuffling? In *Conference on Learning Theory*, pages 3250–3284. PMLR, 2020.

O. Shamir. Making gradient descent optimal for strongly convex stochastic optimization. *CoRR, abs/1109.5647*, 2011.

S. U. Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

S. Vaswani, F. Bach, and M. Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR, 2019.

P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296, 2020.

A Appendix

Definition A.1 (Strong Convexity) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. If there exists a value $\lambda > 0$ such that*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\lambda}{2}\|y - x\|^2$$

*for all $x, y \in \mathbb{R}^d$, we say that f is λ-**strongly convex**.*

Definition A.2 (Lipschitz Continuity) *Let f be a function from $\mathbb{R}^d$ to $\mathbb{R}$. If there exists a value $L > 0$ such that*

$$|f(x) - f(y)| \leq L\|x - y\|$$

*for all $x, y \in \mathbb{R}^d$, we say that f is L-**Lipschitz**.*

Definition A.3 (Smoothness) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We say that f is L-**smooth** if f has L-Lipschitz gradient, i.e.,*

$$|\nabla f(x) - \nabla f(y)| \leq L\|x - y\|$$

for all $x, y \in \mathbb{R}^d$.