Accelerated Distributed Riemannian Optimization Algorithms with Random Shuffling

Wenhan Xian[1] , **Heng Huang**[1]
[1]Computer Science, University of Maryland, College Park
{wxian1, heng}@umd.edu

Abstract

Riemannian optimization has attracted increasing attention recently as it is related to a variety of machine learning problems, including principal component analysis, dictionary learning, and mixture modeling. To tackle large-scale machine learning tasks, distributed learning is usually considered as an effective solution that trains a global model over multiple worker nodes collaboratively to enhance the computational power. Centralized learning and decentralized learning are two types of distributed learning. Centralized learning employs a central parameter server to coordinate the training process while in decentralized learning, each worker only communicates with its peer neighbors. In this paper, we propose accelerated distributed Riemannian stochastic gradient descent algorithms with random shuffling in the cases of both centralized and decentralized learning. We improve the stochastic first-order oracle complexity of the Riemannian SGD from $O(\epsilon^{-4})$ to $O(\sqrt{N}\epsilon^{-3})$ where N is the size of training data. We conduct experiment of the leading eigenvector problem under the condition of centralized learning and decentralized learning to validate the performance of our methods.

1 Introduction

Optimization on Riemannian manifolds has received significant attention in recent years because it is the key issue of many machine learning problems, such as principal component analysis (PCA) [Wold *et al.*, 1987], dictionary learning [Cherian and Sra, 2016; Sun *et al.*, 2016], mixture modeling [Hosseini and Sra, 2015], and deep neural networks with orthogonal constraints [Arjovsky *et al.*, 2016; Huang *et al.*, 2018]. As an example, the principal component analysis results in a Riemannian optimization problem over the Stiefel manifold. One straightforward solution to the Riemannian optimization problem is to operate the projection onto the manifold. However, the projection computation could be too expensive. An alternative solution is to directly perform the optimization process on the manifold. The standard stochastic gradient descent on the Riemannian manifold was proposed in [Bonnabel, 2013] which computes the Riemannian gradients and adopts a retraction operation to map the tangent space to the manifold. More details of related works will be introduced in the next section.

In addition, distributed learning has emerged as a promising research area in the machine learning community. In distributed learning, multiple worker nodes collaboratively train a global learning model without sharing training data. Since large-scale data are distributed to different nodes and heavy computation workloads can be done in parallel, distributed learning is capable of tackling challenging large-scale learning tasks effectively and efficiently. There are two categories of distributed learning, centralized learning, and decentralized learning. Centralized learning is the classic distributed learning scheme that uses a central parameter server to efficiently coordinate the training process. The central node aggregates the computed results from worker nodes and broadcasts the new global model. In contrast, decentralized learning does not need a central node, and communication is only required between local neighborhoods. Decentralized learning has attracted more interest recently because it avoids the high communication overhead on the central node.

In this paper, we consider the following distributed optimization problem:

$$\min_{x\in\mathcal{M}} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),\ f_i(x) = \frac{1}{K_i}\sum_{j=1}^{K_i} F_i(x;\xi_j^i) \quad (1)$$

where $\mathcal{M}$ is the Stiefel manifold $St(d,r) = \{x \in \mathbb{R}^{d\times r} : x^T x = I_r\}$. n is the number of worker nodes and K is the size of data on each node. f_i is the local objective function on the i-th worker node, which is the finite sum of loss functions over K_i local training samples. Let N be the total number of training samples over all worker nodes. In this paper f_i is supposed to be Lipschitz smooth and potentially (geodesically) nonconvex.

Due to the crucial importance of Riemannian optimization and distributed learning, we are motivated to study and accelerate the convergence rate of current distributed Riemannian stochastic gradient descent algorithms. We only consider stochastic algorithms because the full gradient is too expensive to compute for large data. Besides, the objective function is not necessarily convex because modern machine learning models are complicated and convexity cannot be guar-

anteed. Thus, in this paper we propose the centralized and decentralized distributed Riemannian stochastic gradient descent algorithms with random shuffling to improve the convergence rate for nonconvex Riemannian optimization. Random shuffling is the without-replacement version of SGD, which randomly shuffles the dataset in each epoch and loads the training data batch by batch sequentially, rather than using the vanilla with-replacement SGD. SGD with random shuffling converges faster in empirical [Bottou, 2009] and it can be computationally more practical [Bonnabel, 2013]. These reasons motivate us to study Riemannian optimization algorithms with random shuffling. In this paper, we establish a complete convergence analysis that covers both the conditions of centralization and decentralization to prove the advantages of our methods theoretically. We summarize our contributions as follows.

- We propose the centralized and decentralized distributed Riemannian stochastic gradient descent algorithms with random shuffling to accelerate the convergence rate of the standard SGD on Riemannian manifold.
- We establish the convergence analysis that covers both the conditions of centralized and decentralized learning, which guarantees the SFO complexity of $O(\sqrt{N}\epsilon^{-3})$ for nonconvex Riemannian optimization, where N is the size of training data. Our results match the best complexity of random shuffling SGD in Euclidean space.
- We conduct an experiment of the leading eigenvector problem to validate the performance of our methods.

2 Preliminary

2.1 Riemannian Optimization

As defined in problem (1), $\mathcal{M}$ denotes the Stiefel manifold. The notation $T_x\mathcal{M}$ represents the tangent space at the point x on the manifold. In first-order Riemannian optimization algorithms [Bonnabel, 2013; Zhang and Sra, 2016], the Riemannian gradient is computed, which is defined by $grad\ h(x) = \mathcal{P}_{T_x\mathcal{M}}\nabla h(x)$ where $\nabla h(x)$ is the gradient in the Euclidean space and $\mathcal{P}_{T_x\mathcal{M}}$ is the orthogonal projection onto tangent space $T_x\mathcal{M}$. To map a vector v in the tangent space onto the manifold, in some early works the exponential map $Exp_x : T_x\mathcal{M} \to \mathcal{M}$ is used, which finds a geodesic curve γ such that $\gamma(0) = x$ and $\frac{d}{dt}\gamma(0) = v$. However, in many circumstances the computation of the exponential map could be expensive. To avoid this issue, in recent works about Riemannian optimization, an approximated version called retraction map is used instead of the exponential map, which can be calculated much easier and faster. In each iteration of these algorithms, the update rule is defined by $x_{t+1} = R_{x_t}(-\eta\ grad\, h(x_t))$ where η is the learning rate and R_x is the retraction map. The retraction $R_x : T_x\mathcal{M} \to \mathcal{M}$ is a map from the tangent space to the manifold with a local rigidity condition that preserves the gradient at $x \in \mathcal{M}$. Moreover, we assume R_x is a second-order retraction such that $R_x(u) = x + u + O(\|u\|^2)$. In this paper, $\|\cdot\|$ represents the Frobenius norm if there is no special comment. We can achieve the following lemma, which ensures a bound of the gap between $(x + u)$ and $R_x(u)$.

Lemma 1. *(Appendix E in [Liu* et al.*, 2019]) Let R be a second-order retraction on Stiefel manifold $\mathcal{M}$. Then for $\forall x \in \mathcal{M}$ and $\forall u \in T_x\mathcal{M}$, we have $\|R_x(u) - (x + u)\| \leq M\|u\|^2$ for some constant M. Especially, when we use polar retraction and $\|u\| \leq 1$, we can set $M = 1$.*

Asymptotic [Liu *et al.*, 2003; Bonnabel, 2013] and non-asymptotic [Kasai *et al.*, 2018] convergence analyzes were performed for previous Riemannian stochastic gradient descent algorithms for (geodesically) nonconvex problems. In [Kasai *et al.*, 2018], the stochastic first-order oracle (SFO) of $O(\epsilon^{-4})$ was proved for the Riemannian stochastic gradient descent algorithm to achieve an ϵ-stationary point such that $\mathbb{E}\|grad\ f(x)\| \leq \epsilon$. Some subsequent works adopt variance reduction to accelerate the convergence rate of Riemannian stochastic gradient descent [Zhang *et al.*, 2016; Zhou *et al.*, 2019]. In this paper, we do not consider the variance reduction technique because the full gradient is required periodically in variance reduced algorithms, which will be expensive to compute when the size of data is large. Therefore, we only focus on the standard Riemannian stochastic gradient descent, which is more efficient and convenient to implement in practice.

2.2 Decentralized Optimization

Decentralized optimization is a class of distributed optimization that trains models in parallel across multiple worker nodes over a decentralized communication network. Different from the conventional centralized paradigm where all worker nodes need to communicate with the central node, in decentralized optimization each worker node only communicates with its neighbors, which avoids high communication cost issue on the central node, especially when the number of nodes is large. Furthermore, it has been shown that decentralized algorithms can achieve competitive convergence rates compared to their centralized counterparts [Lian *et al.*, 2017; Tang *et al.*, 2018].

In decentralized optimization, each worker node has its own training data and model parameter $x^{(i)}$. In this paper we only consider the case where each $x^{(i)}$ has the same structure. The topology of the decentralized network can be represented by a graph $\mathcal{G}$ where an edge indicates that two worker nodes will communicate with each other. The communication between local neighborhood can be represented by a doubly stochastic mixing matrix $W \in \mathbb{R}^{n\times n}$. Specifically, the sum of each row and each column in W is 1. The new model parameter is a weighted average among its neighborhood according to the weights in W. Hence, in the standard decentralized stochastic gradient descent algorithm [Lian *et al.*, 2017], the update rule on the i-th worker node can be formulated by $x_{t+1}^{(i)} = \sum_{j=1}^{n} w_{ij}(x_t^{(j)} - \eta\nabla f_j(x_t^{(j)}; \xi_t^{(j)}))$ where $\nabla f_j(x_t^{(j)}; \xi_t^{(j)})$ is the stochastic gradient. The communication between neighborhood is also called the consensus step because the gap between the model parameters on different worker nodes will be decreasing after certain number of iterations. The consensus error is defined by $\frac{1}{n}\sum_{i=1}^{n}\|x_t^{(i)} - \bar{x}_t\|^2$. Eventually the consensus error will be smaller than some threshold and the model parameters on different nodes will become closed to their mean value. As a result of [Lian *et*

al., 2017], the decentralized SGD algorithm can achieve an ϵ-stationary point such that $\|\nabla f(\bar{x}_t)\| \leq \epsilon$ with the SFO complexity of $O(\epsilon^{-4})$.

Some previous works have studied the decentralized Riemannian optimization [Shah, 2017; Chen *et al.*, 2021]. In decentralized Riemannian optimization, the original mean value $\bar{x}$ defined in the Euclidean space may not lie on the manifold. The term named induced arithmetic mean (IAM) is used in [Chen *et al.*, 2021], which is defined by

$$\bar{x} = \arg\min_{y \in \mathcal{M}} \frac{1}{n} \sum_{i=1}^{n} \|y - x^{(i)}\|^2 = \mathcal{P}_{\mathcal{M}}(\hat{x}) \tag{2}$$

where $\hat{x}$ is the original mean in the Euclidean space and $\mathcal{P}_{\mathcal{M}}$ is the orthogonal projection onto the manifold $\mathcal{M}$. According to the theoretical results in [Chen *et al.*, 2021], the decentralized Riemannian stochastic gradient descent algorithm can achieve an ϵ-stationary point such that $\|\nabla f(\bar{x}_t)\| \leq \epsilon$ with the SFO complexity of $O(\epsilon^{-4})$. Apart from decentralized Riemannian optimization algorithms, some previous works also focus on centralized distributed learning or federated learning on the Riemannian manifold [Meng *et al.*, 2019; Grammenos *et al.*, 2020; Li and Ma, 2022].

2.3 Random Shuffling

Random shuffling is the without-replacement version of SGD, which randomly permutes the order of training samples in the dataset in each epoch and loads the training data batch by batch. Conversely, according to the vanilla definition of SGD, in each iteration the samples are randomly selected with replacement. It is shown that SGD with random shuffling has many advantages over the with-replacement SGD, including faster convergence and easier implementation. In practice, random shuffling is widely used to train deep learning models, which is the default data loader of many well-known frameworks such as TensorFlow and PyTorch.

Although random shuffling SGD is prevalent in practical implementations, its theoretical convergence analysis is much more complicated than the vanilla SGD because the independence of stochastic sampling in different iterations is lost. [Haochen and Sra, 2019; Nagaraj *et al.*, 2019; Safran and Shamir, 2020; Rajput *et al.*, 2020] studied the convergence of random shuffling SGD in the strongly-convex case. [Haochen and Sra, 2019] proved a non-asymptotic convergence rate of $O((NT)^{-2} + T^{-3})$ under the assumptions of strong convexity, bounded gradient, Lipschitz smoothness and Lipschitz Hessian. [Nagaraj *et al.*, 2019] further achieved the convergence rate of $O(N^{-1}T^{-2})$ without the Lipschitz Hessian assumption. [Safran and Shamir, 2020] and [Rajput *et al.*, 2020] provided lower bounds for SGD with random shuffling.

In nonconvex optimization, [Shamir, 2016] and [Meng *et al.*, 2019] established the convergence analysis to reach a neighborhood of a stationary point for random shuffling SGD. [Nguyen *et al.*, 2021] proposed a unified convergence analysis framework for SGD with random shuffling and guaranteed a convergence rate of $O(N^{-1/3}T^{-2/3})$ in nonconvex optimization. Another previous work [Mishchenko *et al.*, 2020] proved the SFO complexity of $O(\sqrt{N}\epsilon^{-3})$ for SGD with random shuffling. Our convergence analysis on Riemannian manifold matches this result, which is the best complexity for SGD with random shuffling in nonconvex optimization in the Euclidean space.

In the analysis framework of random shuffling SGD in nonconvex optimization, the exact relation is required that $x_i^t = x_0^t - \eta \sum_{j=0}^{i-1} \nabla f(x_j^t)$ where x_j^t is the model parameter in the j-th iteration of the t-th epoch. However, on Riemannian manifold this relation is not satisfied because of the retraction, which will be a challenge in the convergence analysis of Riemannian optimization algorithms.

3 Algorithms

3.1 Centralized Algorithm

In this section we will propose and demonstrate our distributed Riemannian stochastic gradient descent algorithms with random shuffling. Before introducing our methods, we find that the single-machine Riemannian stochastic gradient descent algorithm with random shuffling has not been proposed so far. To fill this gap, we will first provide the description of the single-machine Riemannian stochastic gradient descent algorithm with random shuffling (RSGD-R) in Algorithm 1.

Algorithm 1 Riemannian Stochastic Gradient Descent with Random Shuffling

Input: initial value x_0^0, learning rate η, epoch size K.

for $t = 0, 1, \ldots, T-1$ **do**
  Randomly shuffle the training data into K batches $\pi_0^t, \pi_1^t, \ldots, \pi_{K-1}^t$.
  for $k = 0, 1, \ldots, K-1$ **do**
    Compute Riemannian stochastic gradient $grad\, f(x_k^t; \pi_k^t)$.
    Update $x_{k+1}^t = R_{x_k^t}(-(\eta/K)\, grad\, f(x_k^t; \pi_k^t))$.
  end for
  Let $x_0^{t+1} = x_K^t$.
end for

Let x_k^t represents the model parameter in the k-th iteration in the t-th epoch. Suppose η is the learning rate and there are K iterations in each epoch. At the beginning of each epoch, the training data is shuffled randomly into K batches. In this paper the batchsize is not required to be large, which could be as small as $O(1)$. In each iteration, we compute the Riemannian stochastic gradient $grad\, f(x_k^t; \pi_k^t) \in T_{x_k^t}\mathcal{M}$ based on the stochastic sample set π_k^t. We update the model parameter by $x_{k+1}^t = R_{x_k^t}(-(\eta/K)\, grad\, f(x_k^t; \pi_k^t))$ with the retraction map.

Next we will introduce our centralized Riemannian stochastic gradient descent algorithm with random shuffling (CRSGD-R), which is shown in Algorithm 2.

In the centralized distributed algorithm, each worker node shares the same global model parameter and we still use the notation x_k^t. Let η be the learning rate and K be the number of iteration in each epoch. At the beginning of each epoch, the training data on each worker node is shuffled ran-

Algorithm 2 Centralized Riemannian Stochastic Gradient Descent with Random Shuffling

Input: initial value x_0^0, learning rate η, epoch size K.

for $t = 0, 1, \ldots, T-1$ **do**
  On the i-th worker node:
  Randomly shuffle the training data into K batches $\pi_0^{t,i}, \pi_1^{t,i}, \ldots, \pi_{K-1}^{t,i}$.
  for $k = 0, 1, \ldots, K-1$ **do**
    Compute local Riemannian stochastic gradient $grad\ F_i(x_k^t; \pi_k^{t,i})$.
    Send $grad\ F_i(x_k^t; \pi_k^{t,i})$ to the server.
    Receive the updated global model parameter x_{k+1}^t from the server.
  end for
  On the central server:
  for $k = 0, 1, \ldots, K-1$ **do**
    Receive local Riemannian stochastic gradients from all worker nodes.
    Aggregate the gradient estimator $grad\ f(x_k^t; \pi_k^t) = \frac{1}{n}\sum_{i=1}^n\ grad\ F_i(x_k^t; \pi_k^{t,i})$.
    Update the global model parameter $x_{k+1}^t = R_{x_k^t}(-(\eta/K)\ grad\ f(x_k^t; \pi_k^t))$.
    Broadcast the updated global model parameter x_{k+1}^t to all worker nodes.
  end for
  Let $x_0^{t+1} = x_K^t$.
end for

domly into K batches $\pi_0^{t,i}, \pi_1^{t,i}, \ldots, \pi_{K-1}^{t,i}$ where i is the index of worker nodes. The permutations on different worker nodes are independent to each other. In each iteration, every worker node computes its local Riemannian stochastic gradient $grad\ F_i(x_k^t; \pi_k^{t,i})$ based on the stochastic sample set $\pi_k^{t,i}$. Then the local gradients are sent to the central parameter server. The central server receives these results and aggregates the global gradient estimator. Notice that the Riemannian gradient $grad\ f(x)$ is the orthogonal projection of $\nabla f(x)$ onto the tangent space $T_x\mathcal{M}$ where $\nabla f(x)$ is the regular gradient in the Euclidean space. Therefore, the gradient $\nabla f(x)$ can be written as the sum of $grad\ f(x) + Nf(x)$ where $Nf(x)$ is a vector in the normal space $N_x\mathcal{M}$. Since each worker node shares the same model parameter x_k^t and the same tangent space $T_{x_k^t}\mathcal{M}$, we have

$$grad\ f(x_k^t; \pi_k^t) = \frac{1}{n}\sum_{i=1}^n\ grad\ F_i(x_k^t; \pi_k^{t,i})$$

$$= grad\ (\frac{1}{n}\sum_{i=1}^n F_i(x_k^t; \pi_k^{t,i})) \in T_{x_k^t}\mathcal{M} \tag{3}$$

For consistency and convenience, we define $f(x_k^t; \pi_k^t) = \frac{1}{n}\sum_{i=1}^n F_i(x_k^t; \pi_k^{t,i})$, which will be used in our analysis. Then, the central parameter server updates the global model parameter with the aggregated gradient estimator and the retraction map. At the end of this iteration, the central parameter server broadcasts the new updated model parameter x_{k+1}^t to all worker nodes.

3.2 Decentralized Algorithm

Next we will demonstrate our decentralized Riemannian stochastic gradient descent algorithm with random shuffling (DRSGD-R). The description can be found in Algorithm 3. In decentralized distributed learning, the model is trained over a decentralized network without a central parameter server. In each iteration, a worker node only communicates with its neighbors. In this paper, we only consider static communication network $\mathcal{G}$ and mixing matrix W.

Algorithm 3 Decentralized Riemannian Stochastic Gradient Descent with Random Shuffling

Input: initial values $x_0^{0,1} = \ldots = x_0^{0,n}$, learning rate η, consensus stepsize α, epoch size K.

On the i-th worker node:
for $t = 0, 1, \ldots, T-1$ **do**
  Randomly shuffle the training data into K batches $\pi_0^{t,i}, \pi_1^{t,i}, \ldots, \pi_{K-1}^{t,i}$.
  for $k = 0, 1, \ldots, K-1$ **do**
    Compute local Riemannian stochastic gradient $grad\ F_i(x_k^{t,i}; \pi_k^{t,i})$.
    Communicate with neighbors and compute $y_k^{t,i} = \mathcal{P}_{T_{x_k^{t,i}}\mathcal{M}}(\sum_{j=1}^n W_{ij}x_k^{t,j})$.
    Update the model parameter $x_{k+1}^{t,i} = R_{x_k^{t,i}}(\alpha y_k^{t,i} - (\eta/K)\ grad\ F_i(x_k^{t,i}; \pi_k^{t,i}))$.
  end for
  Let $x_0^{t+1,i} = x_K^{t,i}$.
end for

In decentralized optimization, each worker node has its own model parameter. Let $x_k^{t,i}$ be the local model parameter on the i-th worker node in the k-th iteration in the t-th epoch. Let all worker nodes start with the same initial value. η and K are the learning rate and the number of iterations in each epoch. Similarly to our centralized algorithm, the training data on each worker node is randomly shuffled into K batches $\pi_0^{t,i}, \pi_1^{t,i}, \ldots, \pi_{K-1}^{t,i}$ at the beginning of each epoch. The random permutation on different worker nodes is independent of each other. In each iteration, every worker node computes its local Riemannian stochastic gradient $grad\ F_i(x_k^{t,i}; \pi_k^{t,i})$ based on the local model parameter $x_k^{t,i}$ and the set of stochastic samples $\pi_k^{t,i}$. At the same time, each worker node communicates with its neighbors and calculates the weighted average $y_k^{t,i} = \mathcal{P}_{T_{x_k^{t,i}}\mathcal{M}}(\sum_{j=1}^n W_{ij}x_k^{t,j})$ where $\mathcal{P}$ is the orthogonal projection. According to [Chen *et al.*, 2021], the term $y_k^{t,i}$ can be regarded as applying a Riemannian gradient to function $\phi(X_k^t) = \frac{1}{4}\sum_{i=1}^n\sum_{j=1}^n W_{ij}\|x_k^{t,i} - x_k^{t,j}\|^2$. Hence, it will lead to the consensus of model parameters. After that, each worker node updates its local model parameter by $x_{k+1}^{t,i} = R_{x_k^{t,i}}(\alpha y_k^{t,i} - (\eta/K)\ grad\ F_i(x_k^{t,i}; \pi_k^{t,i}))$ with the retraction map.

4 Convergence Analysis

4.1 RSGD-R and CRSGD-R

In this section, we will show the main theorems of our convergence analysis. The theoretical results indicate that our CRSGD-R and DRSGD-R algorithms can match the SFO complexities of the Euclidean counterparts. Our distributed algorithms are proved to enjoy linear speedup when comparing with the single-machine counterparts. Besides, we can also prove that our algorithms with random shuffling achieves better theoretical results than the vanilla with-replacement version of SGD. In the first part we will demonstrate the theoretical results of the single-machine RSGD-R algorithm and our centralized distributed algorithm CRSGD-R. Before introducing the main theorems, we need the following assumptions, which are commonly used in the analysis of related works.

Assumption 1. *(Lower Bound) The objective f is lower bounded, i.e.,* $\inf_x f(x) = f^* > -\infty$.

Assumption 2. *(Lipschitz Smoothness) Each component function $F_i(x;\xi)$ is L-Lipschitz smooth in Euclidean space, i.e., $\|\nabla F_i(x_1;\xi) - \nabla F_i(x_2;\xi)\| \leq L\|x_1 - x_2\|$ for all $x_1, x_2 \in \mathcal{M}$.*

Lipschitz smoothness is an important and fundamental assumption in optimization. Some Lipschitz-type inequalities are satisfied on Stiefel manifold if we have the following bounded gradient assumption.

Assumption 3. *(Bounded Gradient) There exists a constant G such that $\|\nabla F_i(x;\xi)\| \leq G$.*

The bounded gradient assumption is widely used in the analysis for Riemannian optimization algorithms [Kasai *et al.*, 2018; Chen *et al.*, 2021]. With the bounded gradient assumption, we can achieve the Lipschitz smoothness on Stiefel manifold as the following Lemma.

Lemma 2. *(Appendix C in [Chen* et al.*, 2021]) Assume Assumption 2 and 3 are satisfied. Then there exists a constant L such that for all $x, y \in \mathcal{M}$ we have*

$$f(y) \leq f(x) + \langle grad\, f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2,$$
$$\|grad\, f(x) - grad\, f(y)\| \leq L\|x - y\| \tag{4}$$

We will first demonstrate the main theorems of the single-machine algorithm RSGD-R.

Theorem 1. *Assume Assumption 1, 2 and 3 are satisfied. Let learning rate $\eta \leq \min\{\frac{1}{6L}, \frac{K}{4MG}\}$. Then for single-machine algorithm RSGD-R, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x_0^t)\|^2 \leq \frac{12(f(x_0^0) - f^*)}{\eta T} + O(\frac{\eta^2}{K})$$

Remark 1. *Let $\eta = O(\sqrt{K}\epsilon)$ and $T = O(K^{-1/2}\epsilon^{-3})$. By Theorem 1 we know the RSGD-R algorithm can achieve an ϵ-stationary point by $O(\sqrt{K}\epsilon^{-3})$ iterations. Since the batchsize is $O(1)$, the total SFO complexity is $O(\sqrt{N}\epsilon^{-3})$. This result matches the complexity of SGD with random shuffling for nonconvex optimization in the Euclidean space [Mishchenko* et al.*, 2020]. Besides, it improves the SFO complexity of $O(\epsilon^{-4})$ of Riemannian SGD [Kasai* et al.*, 2018] when $\sqrt{N} < \epsilon^{-1}$.*

Next, we will show the main theorems of our centralized distributed algorithm CRSGD-R.

Theorem 2. *Assume Assumption 1, 2 and 3 are satisfied. Let learning rate $\eta \leq \min\{\frac{1}{6L}, \frac{K}{4MG}\}$. Then for centralized distributed algorithm CRSGD-R, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x_0^t)\|^2 \leq \frac{12(f(x_0^0) - f^*)}{\eta T} + O(\frac{\eta^2}{nK})$$

Remark 2. *Let $\eta = O(\sqrt{nK}\epsilon)$ and $T = O((nK)^{-1/2}\epsilon^{-3})$. By Theorem 2 we know the CRSGD-R algorithm can achieve an ϵ-stationary point by $O(\sqrt{\frac{K}{n}}\epsilon^{-3})$ iterations. Since the batchsize is $O(1)$, the total SFO complexity is $O(\sqrt{nK}\epsilon^{-3}) = O(\sqrt{N}\epsilon^{-3})$. As these gradient oracles are computed by n worker nodes in parallel, it indicates that the centralized distributed algorithm CRSGD-R admits linear speedup with respect to the number of worker nodes.*

4.2 DRSGD-R

In the second part, we will demonstrate the main theorem of our DRSGD-R algorithm for decentralized distributed learning. First, we need to introduce an additional spectral gap assumption about the topology of the decentralized network as follows.

Assumption 4. *(Spectral Gap) The decentralized network is connected by a doubly-stochastic mixing matrix $W \in \mathbb{R}^{n\times n}$ satisfying $W\mathbf{1}_n = W^T\mathbf{1}_n = \mathbf{1}_n$ and $\lambda := \|W - J\|_2 \in [0, 1)$.*

Here J is a $n \times n$ matrix with all elements equal to $\frac{1}{n}$. W is the weight matrix of the decentralized network where $W_{ij} > 0$ if node i and node j are connected, otherwise $W_{ij} = 0$. $\|\cdot\|_2$ denotes the spectral norm of matrices (*i.e.*, largest singular value). Notice that λ is a connectivity measurement of the network graph and it is also the second largest singular value of W. We do not assume W to be symmetric and hence the communication network can be both directed and undirected graph. The spectral gap assumption is crucial in the analysis of decentralized algorithms.

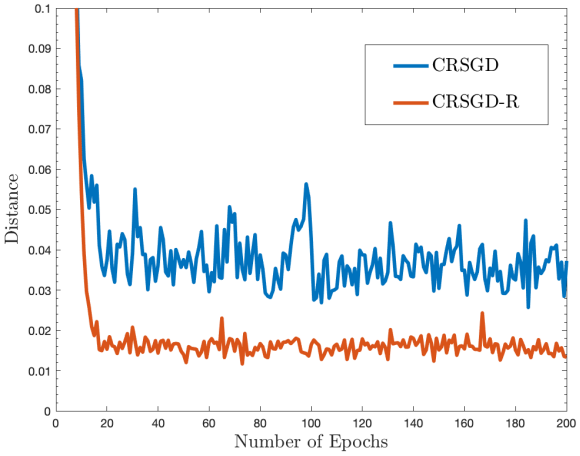
Now we will show the main conclusions of our decentralized distributed algorithm DRSGD-R.

Lemma 3. *Assume Assumption 3 and 4 are satisfied. Let δ be the smallest eigenvalue of W and $\alpha = \frac{1}{2(1-\delta)}$. Define $\rho = 1 - \frac{\alpha(1-\lambda)}{4}$ and $\nu = 1 + \frac{4}{\alpha(1-\lambda)}$. Then we have*

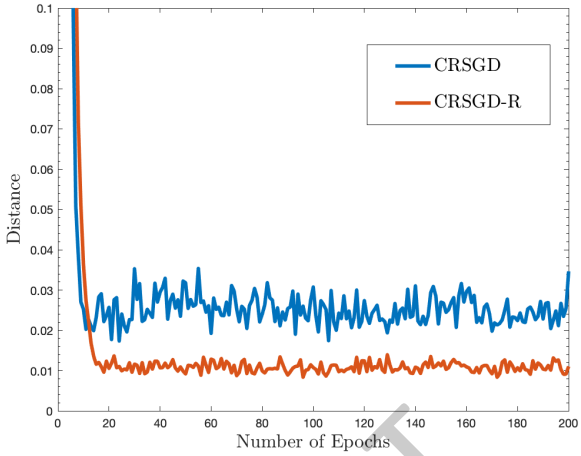
$$\frac{1}{n}\sum_{i=1}^{n}\|x_{k+1}^{t,i} - \bar{x}_{k+1}^t\|^2 \leq \frac{\nu\eta^2G^2}{(1-\rho)K^2} \tag{5}$$

Lemma 3 guarantees an upper bound for the consensus error, which is important and necessary to the convergence of decentralized distributed algorithms.

Theorem 3. *Assume Assumption 1, 2, 3 and 4 are satisfied. Let learning rate $\eta \leq \min\{\frac{1}{4L}, \frac{K}{4MG}\}$. Let δ, α, rho and ν be*



(a) n = 20



(b) n = 40

Figure 1: Experimental results of CRSGD and CRSGD-R algorithms in the leading eigenvectors problem. We plot the distance to the optimal solution with respect to the number of epochs. Figures (a) and (b) show the results of 20 and 40 worker nodes respectively.

	single-machine	centralized	decentralized
SGD	$O(\epsilon^{-4})$	$O(\epsilon^{-4})$	$O(\epsilon^{-4})$
SGD-R	$O(\sqrt{N}\epsilon^{-3})$	$O(\sqrt{N}\epsilon^{-3})$	-
RSGD	$O(\epsilon^{-4})$	$O(\epsilon^{-4})$	$O(\epsilon^{-4})$
RSGD-R	$O(\sqrt{N}\epsilon^{-3})$	$O(\sqrt{N}\epsilon^{-3})$	$O(\sqrt{N}\epsilon^{-3})$

Table 1: Comparisons of the SFO complexity in nonconvex optimization between related works. RSGD and RSGD-R are algorithms on Riemannian manifold.

the same as Lemma 3. Let constant $C_1 = 2M + \frac{(\gamma+4\sqrt{r})\nu}{1-\rho}$. *Then for decentralized distributed algorithm DRSGD-R, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{x}_0^t)\|^2 \leq \frac{4(f(\bar{x}_0^0)-f^*)}{\eta T} + \frac{8L^2\eta^2G^2}{nK} + \frac{8\nu L^2\eta^2G^2}{(1-\rho)K^2} + \frac{8C_1^2\eta^2G^4}{K^2} \tag{6}$$

Remark 3. *When the data size K is larger than the number of worker nodes, the last two terms of Eq. (6) are dominated by $O(\frac{\eta^2}{nK})$. Let $\eta = O(\sqrt{nK}\epsilon)$ and $T = O((nK)^{-1/2}\epsilon^{-3})$. By Theorem 3 we know the DRSGD-R algorithm can achieve an ϵ-stationary point by $O(\sqrt{\frac{K}{n}}\epsilon^{-3})$ iterations and hence the total SFO complexity is $O(\sqrt{N}\epsilon^{-3})$. Lemma 3 ensures a small consensus error of $O(\epsilon^2)$.*

4.3 Comparisons with Related Works

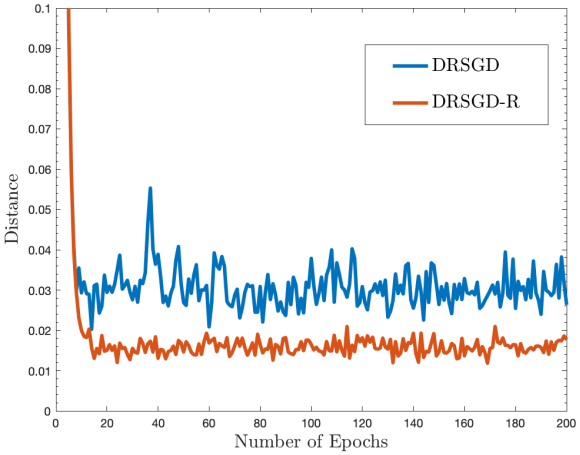
In this section we will compare our algorithms with previous works to have a better illustration of the advantages of our methods. Comparisons of the SFO complexity of related works in nonconvex optimization are shown in Table 1, including the results of single-machine, centralized distributed and decentralized distributed algorithms in the Euclidean space and on the Riemannian manifold. SGD-R represents the stochastic gradient descent algorithm with random shuffling, while SGD refers to the vanilla stochastic gradient descent with replacement.

The results of the single-machine, centralized and decentralized Euclidean SGD can be derived from [Ghadimi and Lan, 2013], [Lian *et al.*, 2015] and [Lian *et al.*, 2017] respectively. The results of SGD with random shuffling in the Euclidean space can be found in [Mishchenko *et al.*, 2020]. The result of the centralized distributed version should be easily achieved based on the more complicated case of federated learning [Mishchenko *et al.*, 2022]. To our best knowledge, we did not find an analysis for decentralized SGD with random shuffling. The results of Riemannian SGD can be derived from [Kasai *et al.*, 2018] and [Chen *et al.*, 2021], where the centralized algorithm can be regarded as a special case of decentralized algorithm with uniform weighted complete graph. The results of Riemannian SGD with random shuffling (in red) are provided by this work. According to Table 1 we can see our theoretical results match state-of-the-art results of the Euclidean counterparts and our algorithms accelerate the convergence rate of the vanilla Riemannian SGD with replacement.

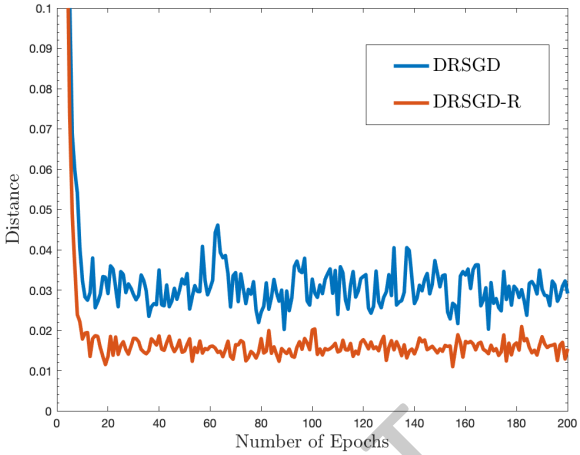
5 Numerical Experiment

We conduct the experiment of the leading eigenvector problem to validate the practical performance of our algorithms. The experiment is conducted under the conditions of centralized distributed learning and decentralized distributed learning respectively. The problem can be formulated by

$$\min_{x\in\mathcal{M}} -\frac{1}{2n}\sum_{i=1}^{n} x^T A_i^T A_i x \tag{7}$$



(a) Ring, n = 40



(b) ER, n = 40

Figure 2: Experimental results of DRSGD and DRSGD-R algorithms in the leading eigenvectors problem. We plot the distance to the optimal solution with respect to the number of epochs. Figures (a) and (b) show the results of the ring topology and ER model with 40 worker nodes.

where $\mathcal{M}$ is the Stiefel manifold $St(d, r)$, n is the number of worker nodes, $A_i \in \mathbb{R}^{m_i \times d}$ is the local data matrix on the i-th worker node and m_i is its corresponding sample size. Let $A = [A_1^T, \cdots, A_n^T]^T$ be the global data matrix. It is known that the global minimum x^* of problem 7 is formed by the first r leading eigenvectors of $A^T A = \sum_{i=1}^n A_i^T A_i$. To measure the quality of solutions, we adopt the canonical correlation as the evaluation metric, which is used to evaluate the distance between two column spaces. The definition is:

$$dist(x, y) := \min_{Q \in O(r)} \|Qu - v\| \tag{8}$$

where $O(r)$ is the orthogonal group, u and v are the orthogonal basis of the column spaces of x and y respectively. Following the experimental setup of [Chen *et al.*, 2021], we set $m_1 = \ldots = m_n = 1000$, $d = 100$ and $r = 5$. The training data A is generated by standard multi-variate Gaussian distribution. Let $A = USV^T$ be the truncated SVD. The singular values of A are modified according to a given eigengap $\Delta \in (0, 1)$ such that $S_{i,i} = S_{0,0} \times \Delta^{i/2}$. Δ is set to 0.8 in our experiment. The experiment is implemented by Python with the mpi4py package. The code is run on a high performance computing (HPC) cluster with multiple nodes equipped with Intel Xeon processors E5-2660 with 14 cores.

We first demonstrate the experimental results of our centralized distributed algorithm CRSGD-R. We choose the centralized distributed version of Riemannian SGD [Kasai *et al.*, 2018] as the baseline. The number of worker nodes is set to 20 and 40 respectively. The learning rate is set to 0.005 and the batchsize is set to 10. We plot the distance to the optimal solution $dist(x, x^*)$ with respect to the number of epochs. The results are shown in Figure 2. According to the experimental results, we can see our CRSGD-R algorithm converges faster than the CRSGD algorithm without random shuffling. The training results of our CRSGD-R algorithm is also more stable than CRSGD. These results validate the effectiveness of random shuffling in centralized distributed Riemannian optimization.

Next we will demonstrate the experimental results of our decentralized distributed algorithm DRSGD-R. We choose decentralized Riemannian SGD [Chen *et al.*, 2021] as the baseline. The decentralized networks considered in the experiment are uniform weighted ring topology and the Erdős-Rényi model $ER(n, p)$ associated with the Metropolis constant matrix and the probability $p = 0.3$. The number of worker nodes is set to 40. Similar to hyperparamter setup of the centralized case, the learning rate is set to 0.005 and the batchsize is set to 10. The stepsize of the consensus step is set to $\alpha = 1$. We also plot the distance to the optimal solution $dist(\bar{x}, x^*)$ with respect to the number of epochs. The results are shown in Figure 1.

According to the experimental results in Figure 1, we can see our DRSGD-R algorithm converges faster than the baseline algorithm DRSGD without random shuffling under different worker node numbers and different topologies. The training results of our RRSGD-R algorithm is also more stable than the baseline algorithm DRSGD. These results verify the effectiveness of random shuffling in decentralized distributed Riemannian optimization.

6 Conclusion

In this paper, we propose the distributed Riemannian stochastic gradient descent algorithms with random shuffling to accelerate the convergence rate of the standard with-replacement SGD algorithm on the Riemannian manifold. We establish the convergence analysis that covers both the conditions of centralized and decentralized learning to improve the SFO complexity of Riemannian SGD from $O(\epsilon^{-4})$ to $O(\sqrt{N}\epsilon^{-3})$ where N is the size of training data. We conduct the experiment of the leading eigenvector problem under the condition of centralized learning and decentralized learning to validate the performance of our methods.

Acknowledgements

This work was partially supported by NSF IIS 2347592, 2348169, DBI 2405416, CCF 2348306, CNS 2347617, RISE 2536663.

References

[Arjovsky *et al.*, 2016] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *International conference on machine learning*, 48:1120–1128, 2016.

[Bonnabel, 2013] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[Bottou, 2009] Léon Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. *Proceedings of the symposium on learning and data science*, 8:2624–2633, 2009.

[Chen *et al.*, 2021] Shixiang Chen, Alfredo Garcia, Mingyi Hong, and Shahin Shahrampour. Decentralized riemannian gradient descent on the stiefel manifold. *International Conference on Machine Learning*, 139:1594–1605, 2021.

[Cherian and Sra, 2016] Anoop Cherian and Suvrit Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE transactions on neural networks and learning systems*, 28(12):2859–2871, 2016.

[Ghadimi and Lan, 2013] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

[Grammenos *et al.*, 2020] Andreas Grammenos, Rodrigo Mendoza Smith, Jon Crowcroft, and Cecilia Mascolo. Federated principal component analysis. *Advances in neural information processing systems*, 33:6453–6464, 2020.

[Haochen and Sra, 2019] Jeff Haochen and Suvrit Sra. Random shuffling beats sgd after finite epochs. *International Conference on Machine Learning*, 97:2624–2633, 2019.

[Hosseini and Sra, 2015] Reshad Hosseini and Suvrit Sra. Matrix manifold optimization for gaussian mixtures. *Advances in neural information processing systems*, 28:910–918, 2015.

[Huang *et al.*, 2018] Lei Huang, Xianglong Liu, Bo Lang, Adams Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):3271–3278, 2018.

[Kasai *et al.*, 2018] Hiroyuki Kasai, Hiroyuki Sato, and Bamdev Mishra. Riemannian stochastic recursive gradient algorithm. *International conference on machine learning*, 80:2516–2524, 2018.

[Li and Ma, 2022] Jiaxiang Li and Shiqian Ma. Federated learning on riemannian manifolds. arXiv preprint arXiv:2206.05668, 2022.

[Lian *et al.*, 2015] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. *Advances in neural information processing systems*, 28:2737–2745, 2015.

[Lian *et al.*, 2017] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30:5330–5340, 2017.

[Liu *et al.*, 2003] Xiuwen Liu, Anuj Srivastava, and Kyle Gallivan. Optimal linear representations of images for object recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:229–234, 2003.

[Liu *et al.*, 2019] Huikang Liu, Anthony Man-Cho So, and Weijie Wu. Quadratic optimization with orthogonality constraint: explicit łojasiewicz exponent and linear convergence of retraction-based line-search and stochastic variance-reduced gradient methods. *Mathematical Programming*, 178(1):215–262, 2019.

[Meng *et al.*, 2019] Qi Meng, Wei Chen, Yue Wang, Zhi-Ming Ma, and Tie-Yan Liu. Convergence analysis of distributed stochastic gradient descent with shuffling. *Neurocomputing*, 337:46–57, 2019.

[Mishchenko *et al.*, 2020] Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33:17309–17320, 2020.

[Mishchenko *et al.*, 2022] Konstantin Mishchenko, Ahmed Khaled, and Peter Richtarik. Proximal and federated random reshuffling. *Proceedings of the 39th International Conference on Machine Learning*, 162:15718–15749, 2022.

[Nagaraj *et al.*, 2019] Dheeraj Nagaraj, Prateek Jain, and Praneeth Netrapalli. Sgd without replacement: Sharper rates for general smooth convex functions. *International Conference on Machine Learning*, 97:4703–4711, 2019.

[Nguyen *et al.*, 2021] Lam M Nguyen, Quoc Tran-Dinh, Dzung T Phan, Phuong Ha Nguyen, and Marten Van Dijk. A unified convergence analysis for shuffling-type gradient methods. *Journal of Machine Learning Research*, 22(207):1–44, 2021.

[Rajput *et al.*, 2020] Shashank Rajput, Anant Gupta, and Dimitris Papailiopoulos. Closing the convergence gap of sgd without replacement. *International Conference on Machine Learning*, 119:7964–7973, 2020.

[Safran and Shamir, 2020] Itay Safran and Ohad Shamir. How good is sgd with random shuffling? *Conference on Learning Theory*, 125:3250–3284, 2020.

[Shah, 2017] Suhail M Shah. Distributed optimization on riemannian manifolds for multi-agent networks. arXiv preprint arXiv:1711.11196, 2017.

[Shamir, 2016] Ohad Shamir. Without-replacement sampling for stochastic gradient methods. *Advances in neural information processing systems*, 29:46–54, 2016.

[Sun *et al.*, 2016] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere ii: Recovery by riemannian trust-region method. *IEEE Transactions on Information Theory*, 63(2):885–914, 2016.

[Tang *et al.*, 2018] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. d^2: Decentralized training over decentralized data. *Proceedings of the 35th International Conference on Machine Learning*, 80:4848–4856, 2018.

[Wold *et al.*, 1987] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2:37–52, 1987.

[Zhang and Sra, 2016] Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. *Conference on learning theory*, 49:1617–1638, 2016.

[Zhang *et al.*, 2016] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *Advances in Neural Information Processing Systems*, 29:4592–4600, 2016.

[Zhou *et al.*, 2019] Pan Zhou, Xiao-Tong Yuan, and Jiashi Feng. Faster first-order methods for stochastic non-convex optimization on riemannian manifolds. *The 22nd International Conference on Artificial Intelligence and Statistics*, 89:138–147, 2019.