Appl. Set-Valued Anal. Optim. 5 (2023), No. 2, pp. 213-232
Available online at http://asvao.biemdas.com
https://doi.org/10.23952/asvao.5.2023.2.07

FEDERATED LEARNING ON RIEMANNIAN MANIFOLDS

JIAXIANG LI[1], SHIQIAN MA[1,2,*]

[1] *Department of Mathematics, University of California, Davis, USA*
[2] *Computational Applied Mathematics and Operations Research, Rice University, Houston, USA*

Dedicated to Professor Henry Wolkowicz on the occasion of his 75th birthday

Abstract. Federated learning (FL) has found many important applications in smart-phone-APP based machine learning applications. Although many algorithms have been studied for FL, to the best of our knowledge, algorithms for FL with nonconvex constraints have not been studied. This paper studies FL over Riemannian manifolds, which finds important applications such as federated PCA and federated kPCA. We propose a Riemannian federated SVRG (`RFedSVRG`) method to solve federated optimization over Riemannian manifolds. We analyze its convergence rate under different scenarios. Numerical experiments are conducted to compare `RFedSVRG` with the Riemannian counterparts of `FedAvg` and `FedProx`. We observed from the numerical experiments that the advantages of `RFedSVRG` are significant.
Keywords. Convergence rate; Federated learning; Machine learning; Nonconvex constraints; Riemannian manifolds.
2020 Mathematics Subject Classification. 90C30, 90C60.

1. Introduction

Federated learning (FL) has drawn lots of attentions recently due to its wide applications in modern machine learning. Canonical FL aims at solving the following finite-sum problem [1, 2, 3]:

$$\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1.1}$$

where each of the f_i (or the data associated with f_i) is stored in different client/agent that could have different physical locations and different hardware. This makes the mutual connection impossible [1]. Therefore, there is a central server that can collect the information from different agents and output a consensus that minimizes the summation of the loss functions from all the clients. The aim of such a framework is to utilize the computation resources of different agents while still maintain the data privacy by not sharing data among all the local agents. Thus the communication is always between the central server and local servers. This setting is commonly observed in modern smart-phone-APP based machine learning applications [1]. We emphasize that we always consider the heterogeneous data scenario where the functions f_i's might be

[*] Corresponding author.

Email addresses: jxjli@ucdavis.edu (J. Li), sqma@rice.edu (S. Ma).

Received September 17, 2022; Accepted January 13, 2023.

different and have different optimal solutions. This problem is inherently hard to solve because each local minima will empirically diverge the update from the global optimum [4, 5].

In this paper, we consider the following FL problem over a Riemannian manifold $\mathcal{M}$:

$$\min_{x\in\mathcal{M}} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{1.2}$$

where $f_i : \mathcal{M} \to \mathbb{R}$ are smooth but not necessarily (geodesically) convex. It is noted that most FL algorithms are designed for the unconstrained setting and convex constraint setting [1, 2, 5, 6, 7, 8, 9, 10], and FL problems with nonconvex constraints such as (1.2) have not been considered. The main difficulty for solving (1.2) lies in aggregating points over a nonconvex set, which may lead to the situation where the averaging point is outside of the constraint set.

One motivating application of (1.2) is the federated kPCA problem

$$\min_{X\in\mathrm{St}(d,r)} f(X) := \frac{1}{n}\sum_{i=1}^{n} f_i(X), \quad \text{where } f_i(X) = -\frac{1}{2}\mathrm{Tr}(X^\top A_i X), \tag{1.3}$$

where $\mathrm{St}(d,r) = \{X \in \mathbb{R}^{d\times r} | X^\top X = I_r\}$ denotes the Stiefel manifold, and A_i is the covariance matrix of the data stored in i-th local agent. When $r = 1$, (1.3) reduces to classical PCA

$$\min_{\|x\|_2=1} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where } f_i(x) = -\frac{1}{2}x^\top A_i x. \tag{1.4}$$

Another motivating example is the Karcher mean problem on positive definite cone [11]:

$$\min_{X\succ 0} f(X) := \frac{1}{n}\sum_{i=1}^{n} f_i(X), \quad \text{where } f_i(X) = \|\log(X^{-1/2}A_i X^{-1/2})\|_{\mathrm{F}}^2, \tag{1.5}$$

where $A_i \succeq 0$ is the covariance matrix of the data stored on i-th local agent. This problem finds its application in diffusion tensor imaging [12, 13, 14], elasticity theory [15] and Electroencephalography (EEG) Classification [16]. This problem is an example of Riemannian optimization with geodesically convex objective, for which we are able to obtain better convergence result.

Existing FL algorithms are not applicable to (1.3)-(1.5) due to the difficulty on aggregating points on nonconvex set. This motivates us to study FL algorithms for optimization on manifold.

1.1. Main Contributions.

We focus on designing efficient federated algorithms for solving (1.2). Our main contributions are:

(1) We propose a Riemannian federated SVRG [17] algorithm (`RFedSVRG`) for solving (1.2). We prove that the convergence rate of our RFedSVRG is $\mathcal{O}(1/\varepsilon^2)$ for obtaining an ε-stationary point. This result matches that of its Euclidean counterparts [5]. To the best of our knowledge, this is the first algorithm for solving FL problems on Riemannian manifolds with convergence guarantees.

(2) The main novelty of our `RFedSVRG` algorithm is a consensus step on the tangent space of the manifold. We compare this new approach with the widely used Karcher mean approach. We show that our method achieves certain "regularization" property and performs very well in practice.

(3) We conduct extensive numerical experiments on our method for solving the PCA (1.4), kPCA (1.3) and PSD Karcher mean (1.5) problems with both synthetic and real data. The numerical results demonstrate that our `RFedSVRG` algorithm significantly outperforms the Riemannian counterparts of two widely used FL algorithms: `FedAvg` [2] and `FedProx` [4].

1.2. **Related Work. Federated optimization.** The most natural idea for FL is the FedAvg algorithm [2], which averages local gradient descent updates and yields a good empirical convergence. However in the data heterogeneous situation, FedAvg suffers from the client-drift effect that each local client will drift the solution towards the minimum of their own local loss function [5, 6, 7, 8, 9, 10]. Many ideas were studied to resolve this issue. For example, [4] proposed the FedProx algorithm, which regularizes each of the local gradient descent update to ensure that the local iterates are not far from the previous consensus point. The FedSplit [10] was proposed later to further mitigate the client-drift effect and convergence results were obtained for convex problems. FedNova [18] was also proposed to improve the performance of FedAvg, however it still suffers from a fundamental speed-accuracy conflict under objective heterogeneity [5]. Variance reduction techniques were also incorporated to FL leading to two new algorithms: federated SVRG (FSVRG) [1] and FedLin [5]. These two algorithms require transmitting the full gradient from the central server to each local client for local gradient updates, therefore require more communication between clients and the central server. Nevertheless, FedLin achieves the theoretical lower bound for strongly convex objective functions [5] with an acceptable amount of increase in the communication cost.

Decentralized optimization on manifolds. Decentralized distributed optimization on manifold has also drawn attentions in recent years [19, 20, 21]. Under this setting, each local agent solves a local problem and then the central server takes the consensus step. The consensus step is usually done by calculating the Karcher mean on the manifold [20, 22], or calculating the minimizer of the sum of the square of the Euclidean distances in the embedded submanifold case [19]. Such consensus steps usually require solving an additional problem inexactly with no exact convergence rate guarantee [22, 23].

It is worth mentioning that the PCA problem under federated learning setting has been considered in the literature [24]. The proposed method in [24] relies on the SVD of data matrices and a subspace merging technique, which is very different from our method. The aim of the algorithm in [24] is to achieve (ε, δ)-differential privacy. In contrast, we mainly consider the convergence rate of our method. Therefore our work is totally different from [24].

2. Preliminaries on Riemannian Optimization

In this part, we briefly review the basic tools we use for optimization on Riemannian manifolds [25, 26, 27, 28]. Due to the limit of space, more detailed discussions are given in supplementary material A. Suppose $\mathscr{M}$ is an m-dimensional Riemannian manifold with Riemannian metric $g : T\mathscr{M} \times T\mathscr{M} \to \mathbb{R}$. We first review the notion of the Riemannian gradients.

Definition 2.1 (Riemannian gradients)**.** For a Riemannian manifold with Riemannian metric g, the Riemannian gradient for $f \in C^\infty(\mathscr{M})$ is the unique tangent vector $\mathrm{grad} f(x) \in T_x\mathscr{M}$ such that $df(\xi) = g(\mathrm{grad} f, \xi)$, $\forall \xi \in T_x\mathscr{M}$, where df is the differential of function f defined as $df(\xi) := \xi(f)$.

For the convergence analysis, we also need the notion of exponential mapping and parallel transport. We first review the definition of exponential mapping

Definition 2.2 (Exponential mapping)**.** Given $x \in \mathscr{M}$ and $\xi \in T_x\mathscr{M}$, the exponential mapping Exp_x is defined as a mapping from $T_x\mathscr{M}$ to $\mathscr{M}$ s.t. $\mathrm{Exp}_x(\xi) := \gamma(1)$ with γ being the geodesic with $\gamma(0) = x$, $\dot{\gamma}(0) = \xi$. A natural corollary is $\mathrm{Exp}_x(t\xi) := \gamma(t)$ for $t \in [0,1]$. Another useful fact is $d(x, \mathrm{Exp}_x(\xi)) = \|\xi\|_x$ since $\gamma'(0) = \xi$ which preserves the speed.

Throughout this paper, we always assume that $\mathscr{M}$ is complete, so that Exp_x is always defined for every $\xi \in T_x\mathscr{M}$. For $\forall x, y \in \mathscr{M}$, the inverse of the exponential mapping $\mathrm{Exp}_x^{-1}(y) \in T_x\mathscr{M}$ is called the logarithm mapping, and we have $d(x,y) = \|\,\mathrm{Exp}_x^{-1}(y)\|_x$, which will be a useful fact in the convergence analysis. We now present the definition of parallel transport.

Definition 2.3 (Parallel transport)**.** Given a Riemannian manifold $(\mathscr{M}, g)$ and two points $x, y \in \mathscr{M}$, the parallel transport $P_{x\to y} : T_x\mathscr{M} \to T_y\mathscr{M}$[1] is a linear operator which keeps the inner product: $\forall \xi, \zeta \in T_x\mathscr{M}$, we have $\langle P_{x\to y}\xi, P_{x\to y}\zeta\rangle_y = \langle \xi, \zeta\rangle_x$.

Parallel transport is useful since the Lipschitz condition for the Riemannian gradient requires moving the gradients in different tangent spaces “parallel” to the same tangent space.

We now present the definition of Lipschitz smoothness and convexity on Riemannian manifolds, which will be utilized in our convergence analysis.

Definition 2.4 (L-smoothness on manifolds)**.** f is called Lipschitz smooth on manifold $\mathscr{M}$ if there exists $L \geq 0$ such that the following inequality holds for function f:

$$\|\mathrm{grad} f(y) - P_{y\to x}\mathrm{grad} f(x)\| \leq L d(x,y). \tag{2.1}$$

For complete Riemannian manifold, we have [11]:

$$f(y) \leq f(x) + \left\langle g_x, \mathrm{Exp}_x^{-1}(y)\right\rangle_x + \frac{L_g}{2} d^2(x,y),\ \forall x, y \in \mathscr{M}. \tag{2.2}$$

The definition of geodesic convexity is given below (see, e.g., [11]).

Definition 2.5 (Geodesic convex)**.** A function $f \in C^1(\mathscr{M})$ is geodesically convex if for all $x, y \in \mathscr{M}$, there exists a geodesic γ such that $\gamma(0) = x$, $\gamma(1) = y$ and

$$f(\gamma(t)) \leq (1-t) f(x) + t f(y),\ \forall t \in [0,1].$$

Or equivalently,

$$f(y) \geq f(x) + \langle \mathrm{grad} f(x), \mathrm{Exp}_x^{-1}(y)\rangle_x.$$

3. THE RFEDSVRG ALGORITHM

The most challenging task for FL on Riemannian manifolds is the consensus step. Suppose the central server receives $x^{(i)}$, $i \in S_t \subset [n]$ from each of the local clients at round t, the question is how the central server aggregates the points to output a unique consensus. In Euclidean space, the most straightforward way is to take the average $\frac{1}{k}\sum_{i\in S_t} x^{(i)}$ with $k = |S_t|$. However, this approach does not apply to the Riemannian setting due to the loss of linearity: the arithmetic average of points can be outside of the manifold. A natural choice for the consensus step on the manifold is to take the Karcher mean of the points [22]:

$$x_{t+1} \leftarrow \mathrm{argmin}_x \frac{1}{k} \sum_{i\in S_t} d^2(x, x^{(i)}), \tag{3.1}$$

[1] Notice that the existence of parallel transport depends on the curve connecting x and y, which is not a problem for complete Riemannian manifold because we always take the unique geodesic that connects x and y.

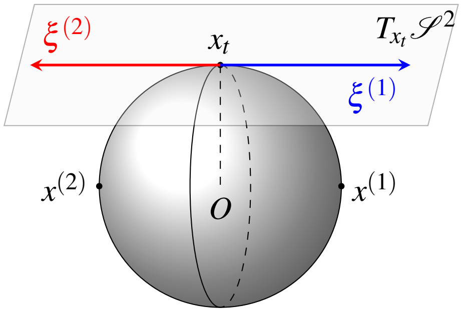


FIGURE 1. Comparison of two consensus methods on $\mathscr{S}^2$

where x_{t+1} is the next iterate point on the central server. This is a natural generalization of the arithmetic average because $d^2(x,y) = \|x-y\|^2$ in Euclidean space. However, solving (3.1) can be time consuming in practice.

We propose the following tangent space consensus step:

$$x_{t+1} \leftarrow \mathrm{Exp}_{x_t}\left(\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x^{(i)})\right), \tag{3.2}$$

where we project each of the point $x_t^{(i)}$ back to the tangent space $T_{x_t}\mathscr{M}$ and then take their average on the tangent space. The consensus step (3.2) has several advantages over the Karcher mean method (3.1). First, (3.2) is of closed-form and easy to compute. Second, (3.2) still coincides with the arithmetic mean when the manifold reduces to the Euclidean space. Third, the tangent space mean (3.2) can easily be extended to the following moving average mean:

$$\mathrm{Exp}_{x_t}\left(\frac{\beta}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x^{(i)})\right),$$

which corresponds to $(1-\beta)x_t + \frac{\beta}{k}\sum_{i\in S_t} x^{(i)}$ in the Euclidean space, while the Karcher mean cannot be easily extended in this scenario. Last, (3.2) has the following "regularization" property as the distance between two consensus points can be controlled, and the Karcher mean method (3.1) does not have this kind of property.

Lemma 3.1. *For the update defined in* (3.2)*, it holds that* $d(x_{t+1},x_t) \leq \frac{1}{k}\sum_{i\in S_t} d(x^{(i)},x_t)$.

To further illustrate this "regularization" property of the tangent space mean (3.2), we consider an (extreme) example on the unit sphere $\mathscr{S}^2$ (see Figure 1) . Here we take x_t on the north pole and two point from the local server as $x^{(1)}$ and $x^{(2)}$, also $\xi^{(i)} = \mathrm{Exp}_{x_t}^{-1}(x^{(i)}) \in T_{x_t}\mathscr{M}$. Then the tangent space mean (3.2) would yield the original point x_t, whereas the Karcher mean could yield any point on the vertical great circle, depending on the starting point in solving the optimization problem (3.1).

Our RFedSVRG algorithm is presented in Algorithm 1, which is a non-trivial manifold extension of the FSVRG algorithm [1]. For RFedSVRG, the local gradient update becomes

$$x_{\ell+1}^{(i)} \leftarrow \mathrm{Exp}_{x_\ell^{(i)}}\left[-\eta^{(i)}\left(\mathrm{grad} f_i(x_\ell^{(i)}) - P_{x_t\to x_\ell^{(i)}}(\mathrm{grad} f_i(x_t) - \mathrm{grad} f(x_t))\right)\right], \tag{3.3}$$

which matches the existing manifold SVRG work [29]. The introduction of the parallel transport $P_{x_t \to x_\ell^{(i)}}$ is necessary because we need to "transport" all the vectors to the same tangent space to conduct addition and subtraction. The algorithm utilizes the gradient information at the previous iterate $\operatorname{grad} f(x_t)$, thus avoids the "client-drift" effect and correctly converges to the global stationary points. This is confirmed by both the theory and the numerical experiments.

Algorithm 1: Riemannian FedSVRG Algorithm (RFedSVRG)

```
   input  : $n$, $k$, $T$, $\{\eta^{(i)}\}$, $\{\tau_i\}$
   output : Option 1: $\tilde{x} = x_T$; or Option 2: $\tilde{x}$ is uniformly sampled from $\{x_1, ..., x_T\}$
1  for $t = 0, ..., T-1$ do
2      Uniformly sample $S_t \subset [n]$ with $|S_t| = k$;
3      for each agent $i$ in $S_t$ do
4          Receive $x_0^{(i)} = x_t$ from the central server;
5          for $\ell = 0, ..., \tau_i - 1$ do
6              Take the local gradient step (3.3).
7          end
8          Send $\hat{x}^{(i)}$ (obtained by one of the following options) to the central server
               • Option 1: $\hat{x}^{(i)} = x_{\tau_i}^{(i)}$;
               • Option 2: $\hat{x}^{(i)}$ is uniformly sampled from $\{x_1^{(i)}, ..., x_{\tau_i}^{(i)}\}$;
9      end
10     The central server aggregates the points by the tangent space mean (3.2);
11 end
```

4. Convergence Analysis

In this section we analyze the convergence behaviour of the `RFedSVRG` algorithm (Algorithm 1). Before we proceed to the convergence results, we briefly review the necessary assumptions, which are standard assumptions for optimization on manifolds [11, 30].

Assumption 4.1 (Smoothness)**.** Suppose that f_i is L_i-smooth as defined in (2.4). It implies that f is L-smooth with $L = \sum_{i=1}^n L_i$.

Now we give the convergence rate results for Algorithm 1. Specifically, Theorem 4.1 gives the convergence rate of Algorithm 1 with $\tau_i = 1$, Theorem 4.2 gives the convergence rate of Algorithm 1 with $\tau_i > 1$, and Theorem 4.3 gives the convergence rate of Algorithm 1 when the objective function is geodescially convex.

Theorem 4.1 (Nonconvex, Algorithm 1 with $\tau_i = 1$)**.** *Suppose the problem* (1.2) *satisfies Assumption 4.1. If we run Algorithm 1 with **Option 1** in Line 8, $\eta^{(i)} \leq \frac{1}{L}$ and $\tau_i = 1$ (i.e. only one step of gradient update for each agent), then the **Option 1** of the output of Algorithm 1 satisfies:*

$$\min_{t=0,\ldots,T} \|\operatorname{grad} f(x_t)\|^2 \leq \mathcal{O}\left(\frac{L(f(x_0) - f(x^*))}{T}\right). \tag{4.1}$$

Remark 4.1. The proof of Theorem 4.1 heavily relies on the choice of $\tau_i = 1$ and the consensus step (3.2). When $\tau_i > 1$, we need to introduce multiple exponential mappings at multiple points for each iteration, which makes the convergence analysis much more challenging due to the loss of linearity. Moreover, the aggregation step makes the situation even worse. However, we are able to show the convergence of Algorithm 1 with $\tau_i > 1$ when $k = 1$. Our numerical experiments show the effectiveness of the `RFedSVRG` algorithm with both $\tau_i = 1$ and $\tau_i > 1$.

To prove the convergence of the Algorithm 1 with $\tau_i > 1$, we also need the following regularization assumption over the manifold $\mathscr{M}$ [29].

Assumption 4.2 (Regularization over manifold)**.** The manifold is complete and there exists a compact set $\mathscr{D} \subset \mathscr{M}$ (diameter bounded by D) so that all the iterates of Algorithm 1 and the optimal points are contained in $\mathscr{D}$. The sectional curvature is bounded in $[\kappa_{\min}, \kappa_{\max}]$. Moreover, we denote the following key geometrical constant that captures the impact of manifold:

$$\zeta = \begin{cases} \frac{\sqrt{|\kappa_{\min}|}D}{\tanh\left(\sqrt{|\kappa_{\min}|}D\right)}, & \text{if } \kappa_{\min} < 0, \\ 1, & \text{if } \kappa_{\min} \geq 0. \end{cases} \tag{4.2}$$

Notice that this assumption holds when the manifold is a sphere or a Stiefel manifold (since they are compact). Now we are ready to give the convergence rate result of Algorithm 1 with $\tau_i > 1$ and $k = 1$, the proof of which is inspired by [29].

Theorem 4.2 (Nonconvex, Algorithm 1 with $\tau_i > 1$ and $k = 1$)**.** *Suppose the problem (1.2) satisfies Assumptions 4.1 and 4.2. If we run Algorithm 1 with* ***Option 2*** *in Line 8, $k = 1$, $\tau_i = \tau > 1$, $\eta^{(i)} = \eta \leq \mathscr{O}(\frac{1}{nL\zeta^2})$, then the* ***Option 2*** *of the output of Algorithm 1 satisfies:*

$$\mathbb{E}\|\mathrm{grad} f(\tilde{x})\|^2 \leq \mathscr{O}\left(\frac{\rho(f(x_0) - f(x^*))}{\tau T}\right),$$

where ρ is an absolute constant specified in the proof and the expectation is taken with respect to the random index i, as well as the randomness introduced by the ***Option 2****.*

Finally, we have the convergence result when the objective function of (1.2) is geodesically convex.

Theorem 4.3 (Geodesic convex)**.** *Suppose the problem (1.2) satisfies Assumption 4.1 and 4.2. Also the functions f_i's are geodesically convex (see Definition 2.5) in $\mathscr{D}$ (as in Assumption 4.2). If we run Algorithm 1 with* ***Option 1*** *in Line 8, $\tau_i = 1$, $S_t = [n]$ (full parallel gradient), and $\eta = \eta^{(1)} = \cdots = \eta^{(n)} \leq \frac{1}{2L}$, then the* ***Option 1*** *of the output of Algorithm 1 satisfies:*

$$f(x_T) - f^* \leq \mathscr{O}\left(\frac{Ld^2(x_0, x^*)}{T}\right). \tag{4.3}$$

5. Numerical Experiments

We now demonstrate the performance of `RFedSVRG` and compare it with two natural ideas for solving (1.1): Riemannian FedAvg (`RFedAvg`) and Riemannian FedProx (`RFedProx`), which are natural extensions of FedAvg [2] and FedProx [4] to the Riemannian setting. Algorithms `RFedAvg` and `RFedProx` are descried in Algorithm 2 and Algorithm 3 in the supplementary material. We conducted our experiments on a desktop with Intel Core 9600K CPU, 32GB RAM

TABLE 1. Comparison of the two consensus methods (3.1) and (3.2). Here $h(x) := \frac{1}{k}\sum_i d^2(x^{(i)}, x)$, CPU time is in seconds and the experiments are repeated and averaged over 10 times.

Dim d	$h(x_t)$	Karcher mean (3.1)			Tangent space mean (3.2)		
		$d^2(x_{t+1}, x_t)$	$h(x_{t+1})$	Time	$d^2(x_{t+1}, x_t)$	$h(x_{t+1})$	Time
100	2.478	2.469	2.813	0.706	0.025	2.427	0.004
200	2.472	2.484	2.804	0.641	0.025	2.422	0.004
500	2.469	2.469	2.795	0.725	0.024	2.421	0.005

and NVIDIA GeForce RTX 2070 GPU. For the codes of operations on Riemannian manifolds we used the ones from the `Manopt` and `PyManopt` packages [31, 32]. Since the logarithm mapping (the inverse of the exponential mapping) on the Stiefel manifold is not easy to compute [33], we adopted the projection-like retraction [34] and the inverse of it [35] to approximate the exponential and the logarithm mappings, respectively.

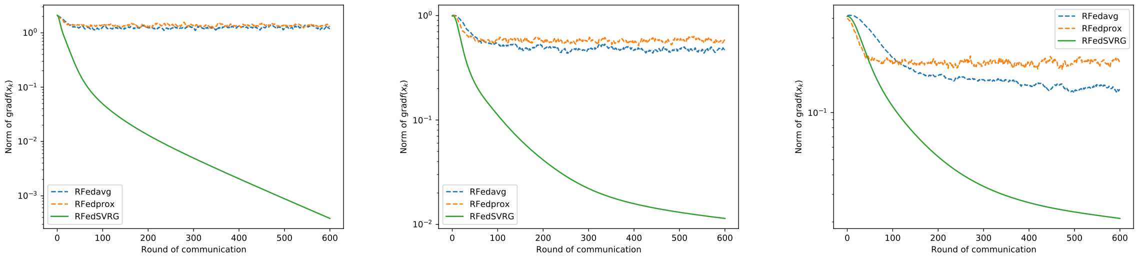
We tested the three algorithms on PCA (1.4), kPCA (1.3), and PSD Karcher mean (1.5) problems (relayed in appendix). For all problems, we measure the norm of the global Riemannian gradients. Additionally, we also measure the sum of principal angles [36] for kPCA.[2]

5.1. **Comparison of the two consensus methods (3.1) and (3.2).** We first compare the two consensus methods (3.1) and (3.2). To this end, we randomly generate x_t and $k = 100$ points $x^{(i)}$ on the unit ball $\mathscr{S}^{d-1}$ with different dimensions d. We then compare the distances $\frac{1}{k}\sum_i d^2(x_t, x^{(i)})$, $\frac{1}{k}\sum_i d^2(x_{t+1}, x^{(i)})$ and $d^2(x_t, x_{t+1})$, as well as the CPU time for computing them. Note that the smaller these distances are, the better. To calculate the Karcher mean, we run the Riemannian gradient descent method starting at x_t until the norm of the Riemannian gradient is smaller than $\varepsilon = 10^{-6}$. The results are shown in Table 1. From Table 1 we see that the tangent space mean (3.2) is indeed better than Karcher mean (3.1) in terms of both quality and CPU time.

5.2. **Experiments on synthetic data.** In this section, we report the results of the three algorithms for solving PCA (1.4) and kPCA (1.3) on synthetic data. We first generate the data $X_i \in \mathbb{R}^{d\times p}$ whose entries are drawn from standard normal distribution. We then set $A_i := X_i X_i^\top$. Notice that under this experiment setting the data in different agents are homogeneous in distribution, which provides a mild environment for comparing the behaviour of the proposed algorithms. We test highly heterogeneous real data later.

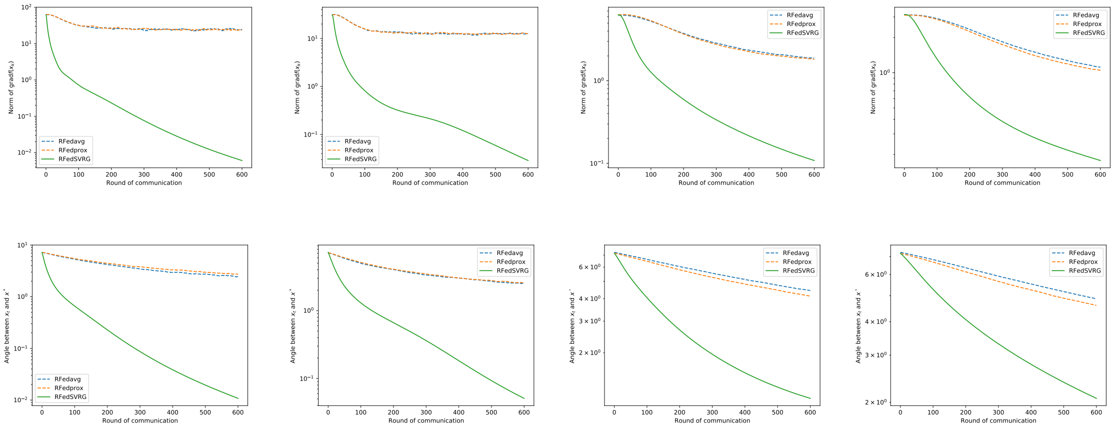
Experiments on PCA. We test the three algorithms on the standard PCA problem (1.4). The data generation process follows Section 5.2. We test our codes with different numbers of agents n and set $k = n/10$ as the number of clients we pick up for each round. We terminate the algorithms if the number of rounds of communication exceeds 600. We sample 10000 data points in $\mathbb{R}^{100}$ and partition them into n agents, each of which contains equal number of data. We test `RFedSVRG` with one iteration for each local agents, i.e. $\tau_i = 1$ and test `RFedAvg` and `RFedProx` with $\tau_i = 5$ iterations in (C.4). We use the constant stepsizes for all three algorithms, and take $\mu = n/10$ for

[2] For the loss f in (1.3), note that $f(X) = f(XQ)$ for any orthogonal matrix $Q \in \mathbb{R}^{r\times r}$. As a result, the optimal solution of $f(X)$ only represents the eigen-space corresponds to the r-largest eigenvalues. Therefore we need the principal angles to measure the angles between the subspaces.



(a) $n = 500$ (b) $n = 1000$ (c) $n = 2500$

FIGURE 2. Results for PCA (1.4). The y-axis denotes $\|\text{grad} f(x_t)\|$. For each figure, the experiments are repeated and averaged over 10 times.



(a) $(n,k) = (50,5)$ (b) $(n,k) = (100,10)$ (c) $(n,k) = (500,50)$ (d) $(n,k) = (1000,100)$

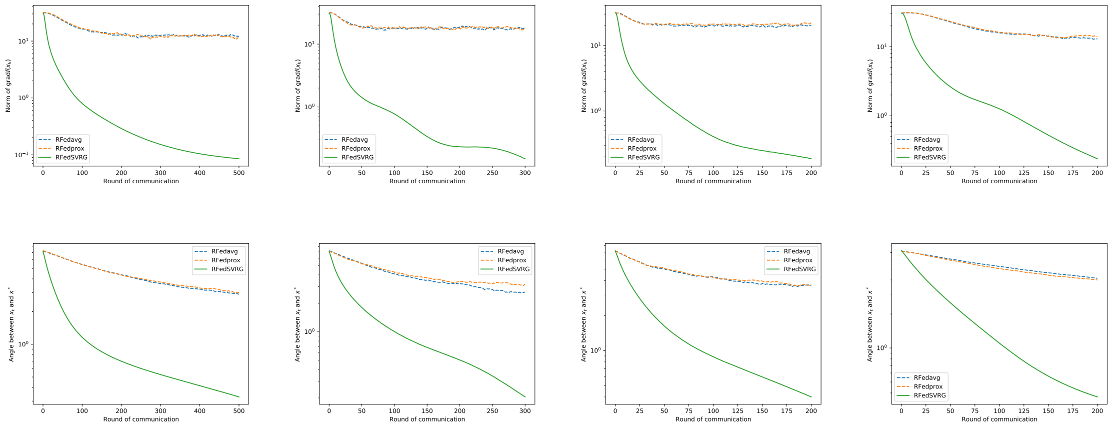
FIGURE 3. Results for kPCA. The y-axis of the figures in the first row denotes $\|\text{grad} f(x_t)\|$, and the y-axis of the figures in the second row denotes the principal angle between x_t and x^*. The experiments are repeated and averaged over 10 times.

each choice of n. The results are presented in Figure 2, from which we see that only RFedSVRG can efficiently decrease $\|\text{grad} f(x_t)\|$ to an acceptable level.

Experiments on kPCA.. We now test the three algorithms on the kPCA problem (1.3). In the first experiment we sample 10000 data points in $\mathbb{R}^{200}$ and partition them into n agents, each of which contains equal number of data. We test our codes with different number of agents n, and again set $k = n/10$. Here we take $(d,r) = (200,5)$. The results are given in Figure 3, where we see that RFedSVRG can efficiently decrease $\|\text{grad} f(x_t)\|$ and the principal angle in all tested cases.

In the second experiment we test the effect of the number of inner loops τ_i. We generate 10000 standard Gaussian vectors. We set $(d,r) = (200,5)$, $k = 10$, and $n = 100$ so that $p = 100$. We choose $\tau = [1,10,50,100]$ for the inner steps for all three algorithms. The results are presented in Figure 4. From this figure we again observe the great performance of RFedSVRG.

5.3. **Experiments for kPCA on real data.** We now show the numerical results of the three algorithms on real data. We focus on the kPCA problem (1.3) and three real data sets: the



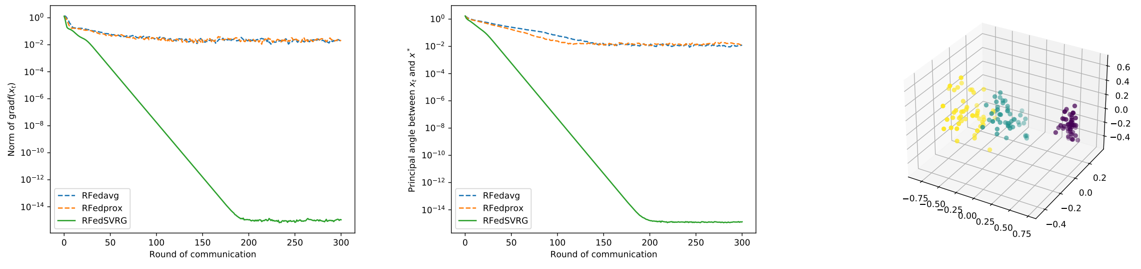
(a) $\tau = 1$ (b) $\tau = 10$ (c) $\tau = 50$ (d) $\tau = 100$

FIGURE 4. Results for kPCA (1.3) with different number of inner loops $\tau = [1, 10, 50, 100]$. The y-axis of the figures in the first row denotes $\|\text{grad} f(x_t)\|$, and the one in the second row denotes the principal angle between x_t and x^*. The experiments are repeated and averaged over 10 times.

Iris dataset [37], the wine dataset [37] and the MNIST hand-written dataset [38]. For all three datasets, we calculate the first r principal directions and the true optimal loss value directly. We can thus compute the principal angles between the iterate and the ground truth. The experiments are repeated and averaged for 10 random initializations.

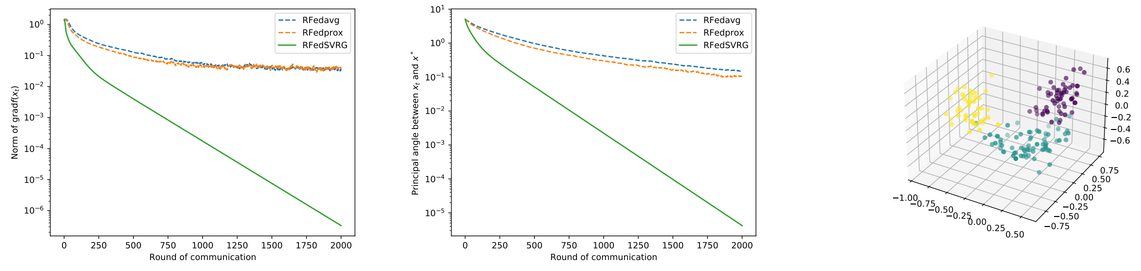
For the first two datasets, we randomly partition the datasets into 10 agents and at each iteration we take $k = 5$ agents. The Figures 5 and 6 demonstrate that RFedSVRG is able to effectively decrease the norm of Riemannian gradient and the principal angles while the other two are not as efficient. We also draw the scatter plots of the dataset toward the principal subspaces computed by RFedSVRG, which show that the algorithm indeed grasps the principal direction of the datasets.

For the MNIST hand-written dataset, the (training) dataset contains 60000 hand-written images of size 28×28, i.e. $d = 784$. This is a relatively large dataset and we test the proposed algorithms with different number of clients. The results are shown in Figure 7 where the efficiency of RFedSVRG is demonstrated again.



(a) Gradient norm (b) Principal angle (c) Scatter plot

FIGURE 5. Results for kPCA (1.3) on Iris dataset. The data is in $\mathbb{R}^4$ ($d = 4$) and we take $r = 3$. The first figure is the norm of Riemannian gradient $\|\text{grad} f(x_t)\|$ and the second is the principal angle between x_t and the true solution x^*, whereas the last figure is the scatter plot of projected data on to the subspace defined by the output of RFedSVRG.

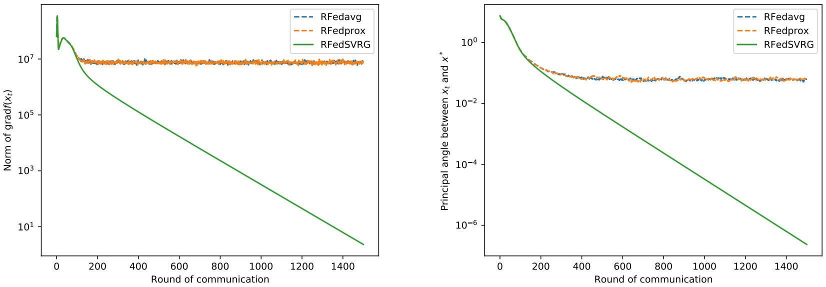


(a) Gradient norm (b) Principal angle (c) Scatter plot

FIGURE 6. Results for kPCA (1.3) with wine dataset. The data is in $\mathbb{R}^{13}$ ($d = 13$) and we take $r = 3$. The first figure is the norm of Riemannian gradient $\|\text{grad} f(x_t)\|$ and the second is the principal angle between x_t and the true solution x^*, whereas the last figure is still the scatter plot of projected data on to the subspace defined by the output of RFedSVRG.

6. CONCLUSIONS

In this paper, we studied the federated optimization over Riemannian manifolds. We proposed a Riemannian federated SVRG algorithm and analyzed its convergence rate to an ε-stationary point. To the best of our knowledge, this is the first federated algorithm over Riemannian manifolds with convergence guarantees. Numerical experiments on federated PCA and federated kPCA were conducted to demonstrate the efficiency of the proposed method. Developing algorithms with lower communication cost, better scalability and sparse solutions are some important topics for future research.



(a) Gradient norm (b) Principal angle

FIGURE 7. Results for kPCA (1.3) with MNIST dataset. The data is in $\mathbb{R}^{784}$ ($d = 784$) and we take $n = 200$ and $r = 5$. Fig (a) is the norm of Riemannian gradient grad$f(x_t)$ and Fig (b) is the principal angle between x_t and the true solution x^*. We take $k = n/10$ and $\tau = 5$ for all algorithms.

Acknowledgement

Research of Shiqian Ma was supported in part by National Science Foundation (NSF) grants DMS-2243650, CCF-2308597, UC Davis CeDAR (Center for Data Science and Artificial Intelligence Research) Innovative Data Science Seed Funding Program, and a startup fund from Rice University.

REFERENCES

[1] J. Konečný, H.B. McMahan, D. Ramage, Peter Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, arXiv preprint arXiv:1610.02527, 2016.

[2] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, pp. 1273-1282, 2017.

[3] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R.G.L. D'Oliveira, H. Eichner, Advances and open problems in federated learning, Foundations and Trends® in Machine Learning, 14 (2021), 1-210.

[4] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proceedings of Machine Learning and Systems, 2 (2020), 429-450.

[5] A. Mitra, R. Jaafar, G.J. Pappas, H. Hassani, Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients, Advances in Neural Information Processing Systems, 34 (2021), 14606-14619.

[6] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for federated learning, International Conference on Machine Learning, PMLR, pp. 5132-5143, 2020.

[7] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on non-iid data, arXiv preprint arXiv:1907.02189, 2019.

[8] G. Malinovskiy, D. Kovalev, E. Gasanov, L. Condat, P. Richtarik, From local SGD to local fixed-point methods for federated learning, International Conference on Machine Learning, PMLR, pp. 6692-6701, 2020.

[9] Z. Charles, J. Konečný, Convergence and accuracy trade-offs in federated learning and meta-learning, International Conference on Artificial Intelligence and Statistics, PMLR, pp. 2575-2583, 2021.

[10] R. Pathak, M.J. Wainwright, FedSplit: An algorithmic framework for fast federated optimization, Advances in Neural Information Processing Systems, 33 (2020), 7057-7066.

[11] H. Zhang, S. Sra, First-order methods for geodesically convex optimization, Conference on Learning Theory, PMLR, pp. 1617-1638, 2016.

[12] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, International Journal of Computer Vision, 66 (2016), 41-66.

[13] P.T. Fletcher, S. Joshi, Riemannian geometry for the statistical analysis of diffusion tensor data, Signal Processing, 87 (2007), 250-262.

[14] Q. Rentmeesters, P.A. Absil, Algorithm comparison for Karcher mean computation of rotation matrices and diffusion tensors, 19th European Signal Processing Conference, pp. 2229-2233, 2011.

[15] S.C. Cowin, G. Yang, Averaging anisotropic elastic constant data, J. Elasticity 46 (1997), 151-180.

[16] E.M. Massart, S. Chevallier, Inductive means and sequences applied to online classification of EEG, International Conference on Geometric Science of Information, pp. 763-770, 2017.

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

[18] J. Wang, Q. Liu, H. Liang, G. Joshi, H. Vincent Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, Advances in Neural Information Processing Systems, 33 (2020), 7611-7623.

[19] S. Chen, A. Garcia, M. Hong, S. Shahrampour, Decentralized Riemannian gradient descent on the Stiefel manifold, International Conference on Machine Learning, PMLR, pp. 1594-1605, 2021.

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

[21] F. Alimisis, P. Davies, B. Vandereycken, D. Alistarh, Distributed principal component analysis with limited communication, Advances in Neural Information Processing Systems, 4 (2021), 2823-2834.

[22] R. Tron, B. Afsari, R. Vidal, Riemannian consensus for manifolds with bounded curvature, IEEE Transactions on Automatic Control, 58 (2012), 921-934.

[23] S. Chen, A. Garcia, Alfredo, M. Hong, S. Shahrampour, On the local linear rate of consensus on the Stiefel manifold, arXiv preprint arXiv:2101.09346, 2021.

[24] A. Grammenos, R. Mendoza-Smith, J. Crowcroft, C. Mascolo, Federated principal component analysis, Advances in Neural Information Processing Systems, 33 (2020), 6453-6464.

[25] P.A. Absil, R. Mahony, R. Sepulchre, Optimization Algorithms on Matrix Manifolds, Optimization Algorithms on Matrix Manifolds, Princeton University Press, Princeton, 2009.

[26] J.M. Lee, Riemannian Manifolds: An Introduction to Curvature, Vol. 176, Springer, New York, 2006,

[27] L.M. Tu, An Introduction to Manifolds, Springer, New York, 2011.

[28] N. Boumal, An Introduction to Optimization on Smooth Manifolds, Cambridge University Press, Cambridge, 2022.

[29] H. Zhang, S.J. Reddi, S. Sra, Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds, Advances in Neural Information Processing Systems, 29 (2016), 4599-4607.

[30] N. Boumal, P.A. Absil, C. Cartis, Global rates of convergence for nonconvex optimization on manifolds, IMA J. Numer. Anal. 39 (2018), 1-33.

[31] N. Boumal, B. Mishra, P.A. Absil, R. Sepulchre, Manopt, a Matlab toolbox for optimization on manifolds, J. Mach. Learn. Res. 15 (2014), 1455-1459.

[32] J. Townsend, N. Koep, S. Weichwald, PyManopt: A Python toolbox for optimization on manifolds using automatic differentiation, J. Mach. Learn. Res. 17 (2016), 1-5.

[33] R. Zimmermann, K. Hüper, Computing the Riemannian logarithm on the Stiefel manifold: metrics, methods and performance, arXiv preprint arXiv:2103.12046, 2021.

[34] P.A. Absil, J. Malick, Projection-like retractions on matrix manifolds, SIAM J. Optim. 22 (2012), 135-158.

[35] T. Kaneko, S. Fiori, T. Tanaka, Empirical arithmetic averaging over the compact Stiefel manifold, IEEE Transactions on Signal Processing, 61 (2012), 883-894.

[36] A.V. Knyazev, P. Zhu, Principal angles between subspaces and their tangents, arXiv preprint arXiv:1209.0523, 2012.

[37] M. Forina, R. Leardi, C. Armanino, PARVUS: An Extendable Package of Programs for Data Exploration, Elsevier, Amsterdam, 1998.

[38] Y. LeCun, L. Bottou, Y. Bengio, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998), 2278-2324.

APPENDIX A. DETAILED PRELIMINARY RESULTS OF OPTIMIZATION ON RIEMANNIAN MANIFOLDS

Suppose that $\mathscr{M}$ is an m-dimensional differentiable manifold. The tangent space $T_x\mathscr{M}$ at $x \in \mathscr{M}$ is a linear subspace that consists of the derivatives of all differentiable curves on $\mathscr{M}$ passing through x: $T_x\mathscr{M} := \{\gamma'(0) : \gamma(0) = x, \gamma([-\delta,\delta]) \subset \mathscr{M}$ for some $\delta > 0, \gamma$ is differentiable$\}$. Notice that for every vector $\gamma'(0) \in T_x\mathscr{M}$, it can be defined in a coordinate-free sense via the operation over smooth functions: $\forall f \in C^\infty(\mathscr{M})$, $\gamma'(0)(f) := \frac{d f\circ\gamma(t)}{dt}\mid_{t=0}$. The Riemannian manifold is a smooth manifold that is equipped with an **inner product** (called Riemannian metric) on the tangent space, $g(\cdot,\cdot) = \langle\cdot,\cdot\rangle_x : T_x\mathscr{M} \times T_x\mathscr{M} \to \mathbb{R}$, that varies smoothly on $\mathscr{M}$.

We first review the notion of the differential between manifolds and the Riemannian gradients here.

Definition A.1 (Differential and Riemannian gradients)**.** Let $F : \mathscr{M} \to \mathscr{N}$ be a C^∞ map between two differential manifolds. At each point $x \in \mathscr{M}$, the differential of F is a mapping: $F_* : T_x\mathscr{M} \to T_x\mathscr{N}$ such that $\forall \xi \in T_x\mathscr{M}$, $F_*(\xi) \in T_x\mathscr{N}$ is given by $(F_*(\xi))(f) := \xi(f \circ F) \in \mathbb{R}$, $f \in C^\infty_{F(x)}(\mathscr{M})$.

If $\mathscr{N} = \mathbb{R}$, i.e., $f \in C^\infty(\mathscr{M})$, the differential f_* is also denoted as df. For a Riemannian manifold with Riemannian metric g, the Riemannian gradient for $f \in C^\infty(\mathscr{M})$ is the unique tangent vector $\mathrm{grad} f(x) \in T_x\mathscr{M}$ such that $df(\xi) = g(\mathrm{grad} f, \xi)$, $\forall \xi \in T_x\mathscr{M}$.

For the convergence analysis, we also need the notion of exponential mapping and parallel transport. To this end, we need to first recall the definition of a geodesic.

Definition A.2 (Geodesic and exponential mapping)**.** Given $x \in \mathscr{M}$ and $\xi \in T_x\mathscr{M}$, the geodesic is the curve $\gamma : I \to \mathscr{M}$, $0 \in I \subset \mathbb{R}$ is an open set, so that $\gamma(0) = x$, $\dot\gamma(0) = \xi$ and $\nabla_{\dot\gamma}\dot\gamma = 0$ where $\nabla : T_x\mathscr{M} \times T_x\mathscr{M} \to T_x\mathscr{M}$ is the Levi-Civita connection defined by metric g. In local coordinates, γ is the unique solution of the following second-order differential equations:

$$\frac{d^2\gamma^k}{dt^2} + \Gamma^k_{i,j}\frac{d\gamma^i}{dt}\frac{d\gamma^j}{dt} = 0$$

under Einstein summation convention, where $\Gamma^k_{i,j}$ are Christoffel symbols defined by metric tensor g. The exponential mapping Exp_x is defined as a mapping from $T_x\mathscr{M}$ to $\mathscr{M}$ s.t. $\mathrm{Exp}_x(\xi) := \gamma(1)$ with γ being the geodesic with $\gamma(0) = x$, $\dot\gamma(0) = \xi$. A natural corollary is $\mathrm{Exp}_x(t\xi) := \gamma(t)$ for $t \in [0,1]$. Another useful fact is $d(x, \mathrm{Exp}_x(\xi)) = \|\xi\|_x$ since $\gamma'(0) = \xi$ which preserves the speed.

APPENDIX B. PROOFS

In this section, we provide the proofs of lemmas and theorems mentioned in the main results. We first finish the proof of Lemma 3.1:

Proof of Lemma 3.1. By Cauchy-Schwarz inequality we have

$$\begin{aligned} d(x_{t+1}, x_t) &= \|\mathrm{Exp}^{-1}_{x_t}(x_{t+1})\| \\ &= \|\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}^{-1}_{x_t}(x^{(i)})\| \le \frac{1}{k}\sum_{i\in S_t}\|\mathrm{Exp}^{-1}_{x_t}(x^{(i)})\| = \frac{1}{k}\sum_{i\in S_t} d(x_t, x^{(i)}). \end{aligned}$$

□

Now we turn to the proof of Theorem 4.1. We would utilize the following lemma:

Lemma B.1. *Under the same settings as Theorem 4.1, we have*

$$f(x_{t+1})-f(x_t)\leq -\eta_t^{(i)}\|\mathrm{grad} f(x_t)\|^2+\frac{(\eta_t^{(i)})^2L}{2}\|\mathrm{grad} f(x_t)\|^2.$$

Proof of Lemma B.1. From the update we know that

$$x_{\ell+1}^{(i)}\leftarrow \mathrm{Exp}_{x_\ell^{(i)}}\left[-\eta_t^{(i)}\left(\mathrm{grad} f_i(x_\ell^{(i)})-P_{x_t\to x_\ell^{(i)}}(\mathrm{grad} f_i(x_t)-\mathrm{grad} f(x_t))\right)\right]$$

i.e.

$$\mathrm{Exp}_{x_\ell^{(i)}}^{-1}(x_{\ell+1}^{(i)})\leftarrow -\eta_t^{(i)}\left(\mathrm{grad} f_i(x_\ell^{(i)})-P_{x_t\to x_\ell^{(i)}}(\mathrm{grad} f_i(x_t)-\mathrm{grad} f(x_t))\right).$$

When $\tau_i=1$, $x_0^{(i)}=x_t$, we have

$$\mathrm{Exp}_{x_t}^{-1}(x_1^{(i)})\leftarrow -\eta_t^{(i)}\left(\mathrm{grad} f_i(x_t)-P_{x_t\to x_1^{(i)}}(\mathrm{grad} f_i(x_t)-\mathrm{grad} f(x_t))\right)=-\eta_t^{(i)}\mathrm{grad} f(x_t)$$

Using Lipschitz smooth of f_i again and the tangent space mean (3.2), we have

$$\begin{aligned}f(x_{t+1})-f(x_t)\leq&\langle \mathrm{Exp}_{x_t}^{-1}(x_{t+1}),\mathrm{grad} f(x_t)\rangle+\frac{L}{2}d^2(x_{t+1},x_t)\\ =&\langle\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x_1^{(i)}),\mathrm{grad} f(x_t)\rangle+\frac{L}{2}\|\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x_1^{(i)})\|^2\\ =&-\eta_t^{(i)}\|\mathrm{grad} f(x_t)\|^2+\frac{(\eta_t^{(i)})^2L}{2}\|\mathrm{grad} f(x_t)\|^2,\end{aligned}$$

where we used the tangent space mean (3.2) for the first equality. □

Now we are ready to present the proof of Theorem 4.1.

Proof of Theorem 4.1. By taking $\eta^{(i)}\leq\frac{1}{L}$, we find from Lemma B.1 that

$$f(x_{t+1})-f(x_t)\leq-\frac{1}{2L}\|\mathrm{grad} f(x_t)\|^2.$$

Summing this inequality over $t=0,1,\dots,T$, we obtain

$$\frac{1}{2L}\sum_{t=0}^{T}\|\mathrm{grad} f(x_t)\|^2\leq f(x_0)-f(x_{T+1})\leq f(x_0)-f(x^*),$$

which yields (4.1) immediately. □

Before we present the proof of Theorem 4.2, we need the following lemma, which is adopted from [29].

Lemma B.2 (Lemma 2 in [29]). *Consider Algorithm 1 with* ***Option*** *2. Suppose we run randomly chosen local agent i at the t-th outer iteration. If we run the local agent i for τ_i local gradient steps (3.3) with initial point x_t, then it holds:*

$$\mathbb{E}\|\mathrm{grad} f(x_\ell^{(i)})\|^2\leq\frac{R_\ell-R_{\ell+1}}{\delta_\ell},\ \ell=0,\dots,\tau_i-1, \tag{B.1}$$

where the expectation is taken with respect to the randomly selected index i, $R_\ell := \mathbb{E}[f(x_\ell^{(i)}) + c_\ell \| \operatorname{Exp}_{x_t}^{-1}(x_\ell^{(i)})\|^2]$, $c_\ell = c_{\ell+1}(1+\beta\eta+2\zeta L^2\eta^2) + L^3\eta^2$ *and* $\delta_\ell = \eta - \frac{c_{\ell+1}\eta}{\beta} - L\eta^2 - 2c_{\ell+1}\zeta\eta^2$. *Here* β *is a free constant to be determined and we take* $c_{\tau_i} = 0$ *in the recursive definition.*

Now we turn to the proof of Theorem 4.2:

Proof of Theorem 4.2. Since $k = 1$, without loss of generality, we denote i as the agent that we choose at the t-th iteration. Moreover, we denote $\eta = \eta^{(i)}$ because there is only one agent.

From (B.1), we note that if we set $\eta < \frac{1}{L+2c_{\ell+1}\zeta}(1 - \frac{c_{\ell+1}}{\beta})$, then we have $\delta^{(i)} := \min_{\ell=0,\dots,\tau_i} \delta_\ell > 0$. In this case, summing (B.1) over $\ell = 0, 1, \dots, \tau_i - 1$ yields

$$\frac{1}{\tau_i} \sum_{\ell=0,\dots,\tau_i-1} \mathbb{E}\|\operatorname{grad} f(x_\ell^{(i)})\|^2 \le \frac{R_0 - R_{\tau_i}}{\tau_i \delta^{(i)}} \le \mathbb{E}\left(\frac{f(x_t) - f(x_{\tau_i}^{(i)})}{\tau_i \delta^{(i)}}\right), \tag{B.2}$$

since $R_0 = f(x_t)$ and $R_{\tau_i} = \mathbb{E}[f(x_{\tau_i}^{(i)}) + c_\ell \| \operatorname{Exp}_{x_t}^{-1}(x_{\tau_i}^{(i)})\|^2] \ge \mathbb{E}[f(x_{\tau_i}^{(i)})]$. Now we take $\beta = L\zeta^{1/2}/n^{1/3}$ and $\eta = 1/(10Ln^{2/3}\zeta^{1/2})^3$. From the recurrence $c_\ell = c_{\ell+1}(1+\beta\eta+2\zeta L^2\eta^2) + L^3\eta^2$ and $c_{\tau_i} = 0$ we have

$$c_0 = \frac{L}{100n^{4/3}\zeta} \frac{(1+\theta)^{\tau_i} - 1}{\theta},$$

where

$$\theta = \eta\beta + 2\zeta\eta^2L^2 = \frac{1}{10n} + \frac{1}{50n^{4/3}} \in \left(\frac{1}{10n}, \frac{3}{10n}\right)$$

is a parameter. If we take $\tau_i = \lfloor 10n/3 \rfloor$ such that $(1+\theta)^{\tau_i} < (1+\frac{3}{10n})^{\tau_i} < e$, then

$$c_0 \le \frac{L}{10n^{1/3}\zeta}(e-1),$$

and $\delta^{(i)}$ is bounded by

$$\begin{aligned} \delta^{(i)} &\ge \left(\eta - \frac{c_0\eta}{\beta} - \eta^2 L - 2c_0\zeta\eta^2\right) \\ &\ge \eta\left(1 - \frac{e-1}{10\zeta^{3/2}} - \frac{1}{10n^{2/3}\zeta^{1/2}} - \frac{e-1}{50n\zeta^{1/2}}\right) \\ &\ge \frac{\eta}{2} = \frac{1}{20Ln^{2/3}\zeta^{1/2}}, \end{aligned}$$

where the last inequality is by $\zeta, n \ge 1$. Note that this lower bound of $\delta^{(i)}$ is independent from the choice of local agent i.

Now summing (B.2) over $t = 0, \dots, T-1$ with $\delta^{(i)} \ge \frac{\eta}{2}$, we obtain

$$\frac{1}{T} \sum_{t=0,\dots,T-1} \frac{1}{\tau_i} \sum_{\ell=0,\dots,\tau_i-1} \mathbb{E}\|\operatorname{grad} f(x_\ell^{(i)})\|^2 \le \frac{2\Delta}{\tau\eta T}, \tag{B.3}$$

where $\Delta = f(x_0) - f^*$.

Now using the **Option 2** of the output of Algorithm 1, we obtain

$$\mathbb{E}\|\operatorname{grad} f(\tilde{x})\|^2 \le \frac{\Delta\rho}{\tau T},$$

[3] It is straightforward to verify that $\eta < \frac{1}{L+2c_{\ell+1}\zeta}(1 - \frac{c_{\ell+1}}{\beta})$ with this choice of η for $\ell = 0, \dots, \tau_i$.

where $\rho = \frac{\eta}{2} = \frac{1}{20Ln^{2/3}\zeta^{1/2}}$. □

Before we present the proof of Theorem 4.3, we need the following lemma [11].

Lemma B.3 (Corollary 8 in [11])**.** *Suppose that the sectional curvature of $\mathscr{M}$ is lower bounded by $\kappa_{\min}$ and we update $x_{t+1} \leftarrow \mathrm{Exp}_{x_t}(-\eta_t g_t)$. Suppose also that the update sequence $\{x_t\} \subset \mathscr{D}$ where $\mathscr{D}$ is a compact set with diameter D, then for any $x \in \mathscr{M}$ it holds:*

$$\langle -g_t, \mathrm{Exp}_{x_t}^{-1}(x)\rangle \le \frac{1}{2\eta_t}(d^2(x_t,x) - d^2(x_{t+1},x)) + \frac{\zeta\eta_t}{2}\|g_t\|^2. \tag{B.4}$$

where ζ is given in (4.2).

We now present the proof of Theorem 4.3.

Proof of Theorem 4.3. From Lemma B.3 we get

$$\langle\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x^{(i)}), \mathrm{Exp}_{x_t}^{-1}(x)\rangle \le \frac{1}{2}(d^2(x_t,x) - d^2(x_{t+1},x)) + \frac{\zeta}{2}\|\frac{1}{k}\sum_{i\in S_t}\mathrm{Exp}_{x_t}^{-1}(x^{(i)})\|^2, \tag{B.5}$$

which is equivalent to (since we assume $S_t = [n]$ and $\eta^{(i)} = \eta$):

$$-\eta\langle\frac{1}{n}\sum_{i=1,\ldots,n}\mathrm{grad} f_i(x_t), \mathrm{Exp}_{x_t}^{-1}(x)\rangle \le \frac{1}{2}(d^2(x_t,x) - d^2(x_{t+1},x)) + \frac{\zeta}{2}\|\frac{1}{n}\sum_{i=1,\ldots,n}\mathrm{Exp}_{x_t}^{-1}(x^{(i)})\|^2. \tag{B.6}$$

Now use the geodesic convexity of f_i and (B.6), we have (denote $\Delta_t := f(x_t) - f(x^*)$ and $\Delta_t^i := f_i(x_t) - f_i(x^*)$)

$$\Delta_t^i \le -\langle \mathrm{grad} f_i(x_t), \mathrm{Exp}_{x_t}^{-1}(x^*)\rangle.$$

Summing this inequality over $i = 1,\ldots,n$, we have

$$\begin{aligned}\Delta_t \le& -\langle\frac{1}{n}\sum_{i=1,\ldots,n}\mathrm{grad} f_i(x_t), \mathrm{Exp}_{x_t}^{-1}(x^*)\rangle\\ \le&\frac{1}{2\eta}(d^2(x_t,x^*) - d^2(x_{t+1},x^*)) + \frac{\zeta}{2\eta}\|\frac{1}{n}\sum_{i=1,\ldots,n}\mathrm{Exp}_{x_t}^{-1}(x^{(i)})\|^2 \qquad (B.7)\\ \le&\frac{1}{2\eta}(d^2(x_t,x^*) - d^2(x_{t+1},x^*)) + \frac{\zeta\eta}{2n}\|\mathrm{grad} f(x_t)\|^2.\end{aligned}$$

Again from Lemma B.1, we obtain

$$\Delta_{t+1} - \Delta_t \le (-\eta_t^{(i)} + \frac{(\eta_t^{(i)})^2 L}{2})\|\mathrm{grad} f(x_t)\|^2. \tag{B.8}$$

Now multiply (B.8) by ζ and add it to (B.7), we have

$$\zeta\Delta_{t+1} - (\zeta-1)\Delta_t \le \zeta\left(\frac{\eta}{2n} - \eta + \frac{\eta^2 L}{2}\right)\|\mathrm{grad} f(x_t)\|^2 + \frac{1}{2\eta}(d^2(x_t,x^*) - d^2(x_{t+1},x^*)). \tag{B.9}$$

Now take $\eta \le \frac{1}{2L}$, we know that $\frac{\eta}{2n} - \eta + \frac{\eta^2 L}{2} \le 0$. Thus

$$\zeta\Delta_{t+1} - (\zeta-1)\Delta_t \le \frac{1}{2\eta}(d^2(x_t,x^*) - d^2(x_{t+1},x^*)). \tag{B.10}$$

Summing this up over t from 0 to $T-1$, we obtain

$$\zeta\Delta_T + \sum_{t=0}^{T-1}\Delta_t \leq (\zeta-1)\Delta_1 + \frac{d^2(x_0,x^*)}{2\eta}. \tag{B.11}$$

Also by (B.8) we know $\Delta_{t+1} \leq \Delta_t$. Thus

$$\Delta_T \leq \frac{\zeta D^2}{2\eta(\zeta+T-2)}. \tag{B.12}$$

□

Appendix C. RFedAvg and RFedProx Algorithms

FedAvg [2] and FedProx [4] are two widely used algorithms for FL problems in Euclidean space. At each iteration, FedAvg minimizes the local loss f_i for fixed steps using gradient descents:

$$x_{\ell+1}^{(i)} \leftarrow x_\ell^{(i)} - \eta^{(i)}\nabla f_i(x_{\ell+1}^{(i)}), \tag{C.1}$$

while FedProx solves a local proximal point subproblem:

$$x^{(i)} \leftarrow \operatorname{argmin}_x f_i(x) + \frac{\mu}{2}\|x-x_t\|^2. \tag{C.2}$$

For RFedAvg, which is the Riemannian counterpart of FedAvg, (C.1) is replaced by

$$x_{\ell+1}^{(i)} \leftarrow \operatorname{Exp}_{x_\ell^{(i)}}\left(-\eta^{(i)}\operatorname{grad} f_i(x_\ell^{(i)})\right).$$

For RFedProx, which is the Riemannian counterpart of FedProx, (C.2) is replaced by

$$x_{t+1}^{(i)} \leftarrow \operatorname{argmin}_{x\in\mathcal{M}} f_i(x) + \frac{\mu}{2}d^2(x,x_t), \tag{C.3}$$

where $d(x,y)$ is the geodesic distance between x and y. In the implementation of RFedProx, (C.3) is solved by Riemannian gradient descent:

$$x_{\ell+1}^{(i)} \leftarrow \operatorname{Exp}_{x_\ell^{(i)}}(-\eta^{(i)}\operatorname{grad} h_i(x_\ell^{(i)})),\ \ell = 0,\dots,\tau_i - 1. \tag{C.4}$$

RFedAvg and RFedProx are described in Algorithms 2 and 3, respectively.

Appendix D. Experiments on PSD Karcher Mean

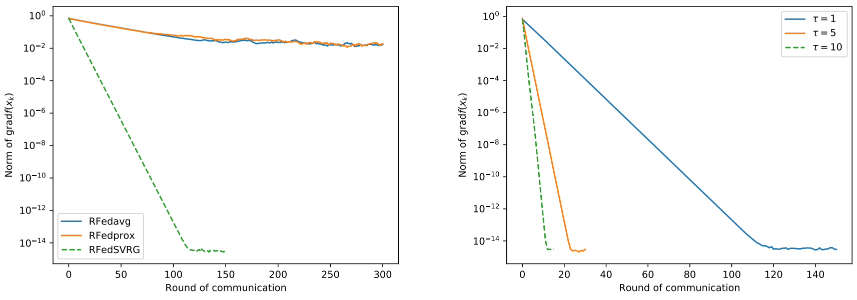
The PSD Karcher mean (1.5) is an example of how Riemannian gradient method with tangent space steps could be utilized to solve Karcher mean (3.1) type problems directly. In this experiment we take $d = 20$, $n = 10$ and $k = 5$. We test the proposed RFedSVRG with RFedAvg and RFedProx, as well as with different τ. The results are in Figure 8. The convergence curves show a linear rate of convergence, largely due to the fact that (1.5) is geodesic (strongly) convex [11], which also show great potential for better theoretical convergence analysis.

Algorithm 2: Riemannian FedAvg algorithm

```
input  : $n$, $k$, $T$, $\{\eta^{(i)}\}$, $\{\tau_i\}$
output : $x_T$
for $t = 0, ..., T-1$ do
    Uniformly sample $S_t \subset [n]$ with $|S_t| = k$;
    for each agent $i$ in $S_t$ do
        Receive $x_t$ from the central server;
        for $\ell = 0, ..., \tau_i - 1$ do
            $x_{\ell+1}^{(i)} \leftarrow \mathrm{Exp}_{x_\ell^{(i)}}\left(-\eta^{(i)} \mathrm{grad} f_i(x_\ell^{(i)})\right)$;
        end
        Send the obtained $x_{\tau_i}^{(i)}$ to the central server;
    end
    The central server aggregates the points by the tangent space mean (3.2);
end
```

Algorithm 3: Riemannian FedProx Algorithm

```
input  : $n$, $k$, $T$, $\mu$, $\gamma$
output : $x_T$
for $t = 0, ..., T-1$ do
    Uniformly sample $S_t \subset [n]$ with $|S_t| = k$;
    for each agent $i$ in $S_t$ do
        Receive $x_t$ from the central server;
        Obtain $x^{(i)} \leftarrow \mathrm{argmin}_{x \in \mathcal{M}} f_i(x) + \frac{\mu}{2} d^2(x, x_t)$ upto a $\gamma$ approximate solution;
        Send the obtained $x^{(i)}$ to the central server;
    end
    The central server aggregates the points by the tangent space mean (3.2);
end
```



(a) Gradient norm (b) Changing τ

FIGURE 8. Results for PSD Karcher mean problem (1.5). The left figure is the test of different algorithms, and the right figure is the test of `RFedSVRG` with different τ.