Federated Learning on Riemannian Manifolds

Jiaxiang Li
Department of Mathematics
University of California, Davis
Davis, CA 95616
jxjli@ucdavis.edu

Shiqian Ma*
Department of Mathematics
University of California, Davis
Davis, CA 95616
sqma@ucdavis.edu

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.

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 [16, 22, 12]:

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

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 [16]. 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 [16]. We emphasize that we always consider the heterogeneous data scenario where the functions f_i's might be 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 [19, 23].

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

$$\min_{x\in\mathcal{M}} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{2}$$

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 [16, 22, 14, 20, 21, 7, 24, 23], and FL problems with nonconvex constraints such as (2) have not been considered.

*`https://www.math.ucdavis.edu/~sqma`

Preprint. Under review.

The main difficulty for solving (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 (2) is the federated kPCA problem

$$\min_{X\in \mathrm{St}(d,r)} f(X) := \frac{1}{n}\sum_{i=1}^{n} f_i(X), \text{ where } f_i(X) = -\frac{1}{2}\operatorname{tr}(X^\top A_i X), \tag{3}$$

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 the i-th local agent. When $r = 1$, (3) reduces to classical PCA

$$\min_{\|x\|_2=1} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \text{ where } f_i(x) = -\frac{1}{2}x^\top A_i x. \tag{4}$$

Existing FL algorithms are not applicable to (3) and (4) due to the difficulty on aggregating points on nonconvex set.

1.1 Main Contributions

We focus on designing efficient federated algorithms for solving (2). Our main contributions are:

1. We propose a Riemannian federated SVRG algorithm (`RFedSVRG`) for solving (2). We prove that the convergence rate of our RFedSVRG algorithm is $\mathcal{O}(1/\epsilon^2)$ for obtaining an ϵ-stationary point. This result matches that of its Euclidean counterparts [23]. To the best of our knowledge, this is the first algorithm for solving FL problems over 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 (4) and kPCA (3) 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` [22] and `FedProx` [19].

1.2 Related Work

Federated optimization. The most natural idea for FL is the `FedAvg` algorithm [22], 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 [14, 20, 21, 7, 24, 23]. Many ideas were studied to resolve this issue. For example, [19] 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` [24] was proposed later to further mitigate the client-drift effect and convergence results were obtained for convex problems. `FedNova` [29] was also proposed to improve the performance of `FedAvg`, however it still suffers from a fundamental speed-accuracy conflict under objective heterogeneity [23]. Variance reduction techniques were also incorporated to FL leading to two new algorithms: federated SVRG (`FSVRG`) [16] and `FedLin` [23]. 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 [23] 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 [8, 25, 3]. 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 [27, 25], or calculating the minimizer of the sum of the square of the Euclidean distances in the embedded submanifold case [8]. Such consensus steps usually require solving an additional problem inexactly with no exact convergence rate guarantee [27, 9].

It is worth mentioning that the PCA problem under federated learning setting has been considered in the literature [11]. The proposed method in [11] 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 [11] 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 [11].

2 Preliminaries on Riemannian Optimization

In this part, we briefly review the basic tools we use for optimization on Riemannian manifolds [1, 18, 28, 5]. Due to the limit of space, more detailed discussions are given in supplementary material A. Suppose $\mathcal{M}$ is an m-dimensional Riemannian manifold with Riemannian metric $g : T\mathcal{M} \times T\mathcal{M} \to \mathbb{R}$. We first review the notion of the Riemannian gradients.

Definition 1 (Riemannian gradients)**.** *For a Riemannian manifold with Riemannian metric g, the Riemannian gradient for $f \in C^\infty(\mathcal{M})$ is the unique tangent vector $\operatorname{grad} f(x) \in T_x\mathcal{M}$ such that $df(\xi) = g(\operatorname{grad} f, \xi)$, $\forall \xi \in T_x\mathcal{M}$, 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 (Exponential mapping)**.** *Given $x \in \mathcal{M}$ and $\xi \in T_x\mathcal{M}$, the exponential mapping Exp_x is defined as a mapping from $T_x\mathcal{M}$ to $\mathcal{M}$ s.t. $\operatorname{Exp}_x(\xi) := \gamma(1)$ with γ being the geodesic with $\gamma(0) = x$, $\dot\gamma(0) = \xi$. A natural corollary is $\operatorname{Exp}_x(t\xi) := \gamma(t)$ for $t \in [0, 1]$. Another useful fact is $d(x, \operatorname{Exp}_x(\xi)) = \|\xi\|_x$ since $\gamma'(0) = \xi$ which preserves the speed.*

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

Definition 3 (Parallel transport)**.** *Given a Riemannian manifold $(\mathcal{M}, g)$ and two points $x, y \in \mathcal{M}$, the parallel transport $P_{x\to y} : T_x\mathcal{M} \to T_y\mathcal{M}$[2] is a linear operator which keeps the inner product: $\forall \xi, \zeta \in T_x\mathcal{M}$, 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 4 (L-smoothness on manifolds)**.** *f is called Lipschitz smooth on manifold $\mathcal{M}$ if there exists $L \geq 0$ such that the following inequality holds for function f:*

$$\|\operatorname{grad} f(y) - P_{y\to x} \operatorname{grad} f(x)\| \leq L d(x, y). \tag{5}$$

For complete Riemannian manifold, we have [31]:

$$f(y) \leq f(x) + \left\langle g_x, \operatorname{Exp}_x^{-1}(y)\right\rangle_x + \frac{L_g}{2} d^2(x, y),\ \forall x, y \in \mathcal{M}. \tag{6}$$

The definition of geodesic convexity is given below (see, e.g., [31]).

Definition 5 (Geodesic convex)**.** *A function $f \in C^1(\mathcal{M})$ is geodesically convex if for all $x, y \in \mathcal{M}$, there exists a geodesic γ such that $\gamma(0) = x$, $\gamma(1) = y$ and*

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

Or equivalently,

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

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

[2]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.

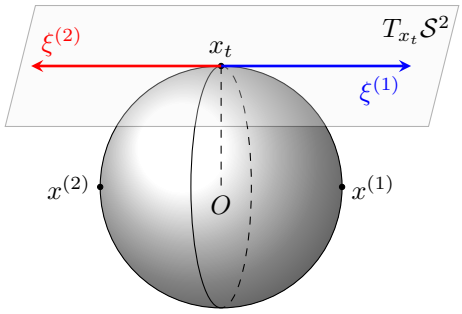


Figure 1: Comparison of two consensus methods on $\mathcal{S}^2$

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 [27]:

$$x_{t+1} \leftarrow \operatorname*{argmin}_{x} \frac{1}{k}\sum_{i\in S_t} d^2(x, x^{(i)}), \tag{7}$$

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 (7) can be time consuming in practice.

We propose the following tangent space consensus step:

$$x_{t+1} \leftarrow \operatorname{Exp}_{x_t}\left(\frac{1}{k}\sum_{i\in S_t} \operatorname{Exp}_{x_t}^{-1}(x^{(i)})\right), \tag{8}$$

where we project each of the point $x_t^{(i)}$ back to the tangent space $T_{x_t}\mathcal{M}$ and then take their average on the tangent space. The consensus step (8) has several advantages over the Karcher mean method (7). First, (8) is of closed-form and easy to compute. Second, (8) still coincides with the arithmetic mean when the manifold reduces to the Euclidean space. Third, the tangent space mean (8) can easily be extended to the following moving average mean:

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

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, (8) has the following "regularization" property as the distance between two consensus points can be controlled, and the Karcher mean method (7) does not have this kind of property.

Lemma 6. *For the update defined in* (8)*, it holds that*

$$d(x_{t+1}, x_t) \le \frac{1}{k}\sum_{i\in S_t} d(x^{(i)}, x_t).$$

To further illustrate this "regularization" property of the tangent space mean (8), we consider an (extreme) example on the unit sphere $\mathcal{S}^2$ (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 $\xi^{(i)} = \operatorname{Exp}_{x_t}^{-1}(x^{(i)}) \in T_{x_t}\mathcal{M}$. Then the tangent space mean (8) 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 (7).

Our `RFedSVRG` algorithm is presented in Algorithm 1, which is a non-trivial manifold extension of the FSVRG algorithm [16]. For `RFedSVRG`, the local gradient update becomes

$$x_{\ell+1}^{(i)} \leftarrow \operatorname{Exp}_{x_\ell^{(i)}}\left[-\eta^{(i)}\left(\operatorname{grad} f_i(x_\ell^{(i)}) - P_{x_t \to x_\ell^{(i)}}(\operatorname{grad} f_i(x_t) - \operatorname{grad} f(x_t))\right)\right], \tag{9}$$

which matches the existing manifold SVRG work [30]. 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 $\{x_1, ..., x_T\}$

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 $x_0^{(i)} = x_t$ from the central server;
    for $\ell = 0, ..., \tau_i - 1$ **do**
      Take the local gradient step (9).
    end
    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 $\{x_1^{(i)}, ..., x_{\tau_i}^{(i)}\}$;
  end
  The central server aggregates the points by the tangent space mean (8);
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 [31, 6].

Assumption 1 (Smoothness)**.** *Suppose f_i is L_i-smooth as defined in* (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 7 gives the convergence rate of Algorithm 1 with $\tau_i = 1$, Theorem 8 gives the convergence rate of Algorithm 1 with $\tau_i > 1$, and Theorem 9 gives the convergence rate of Algorithm 1 when the objective function is geodescially convex.

Theorem 7 (Nonconvex, Algorithm 1 with $\tau_i = 1$)**.** *Suppose the problem* (2) *satisfies Assumption 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:*

$$\min_{t=0,...,T} \| \operatorname{grad} f(x_t) \|^2 \leq \mathcal{O}\left(\frac{L(f(x_0) - f(x^*))}{T} \right). \tag{10}$$

Remark. Our proof of Theorem 7 relies heavily on the choice of $\tau_i = 1$ and the consensus step (8). 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 Algorithm 1 with $\tau_i > 1$, we also need the following regularization assumption over the manifold $\mathcal{M}$ [30].

Assumption 2 (Regularization over manifold)**.** *The manifold is complete and there exists a compact set $\mathcal{D} \subset \mathcal{M}$ (diameter bounded by D) so that all the iterates of Algorithm 1 and the optimal points*

are contained in $\mathcal{D}$. The sectional curvature is bounded in $[\kappa_{\min}, \kappa_{\max}]$. Moreover, we denote the following key geometrical constant that captures the impact of manifold:

$$\zeta = \begin{cases} \frac{\sqrt{|\kappa_{\min}|}D}{\tanh\left(\sqrt{|\kappa_{\min}|}D\right)}, & \text{if } \kappa_{\min} < 0 \\ 1, & \text{if } \kappa_{\min} \geq 0. \end{cases} \tag{11}$$

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 [30].

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

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

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 (2) is geodesically convex.

Theorem 9 (Geodesic convex)**.** *Suppose the problem* (2) *satisfies Assumption 1 and 2. Also the functions f_i's are geodesically convex (see Definition 5) in $\mathcal{D}$ (as in Assumption 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:*

$$f(x_T) - f^* \leq \mathcal{O}\left(\frac{Ld^2(x_0, x^*)}{T}\right). \tag{12}$$

5 Numerical experiments

We now show the performance of RFedSVRG and compare it with two natural ideas for solving (1): Riemannian FedAvg (`RFedAvg`) and Riemannian FedProx (`RFedProx`), which are natural extensions of FedAvg [22] and FedProx [19] 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 and NVIDIA GeForce RTX 2070 GPU. For the codes of operations on Riemannian manifolds we used the ones from the `Manopt` and `PyManopt` packages [4, 26]. Since the logarithm mapping (the inverse of the exponential mapping) on the Stiefel manifold is not easy to compute [32], we adopted the projection-like retraction [2] and the inverse of it [13] to approximate the exponential and the logarithm mappings, respectively.

We tested the three algorithms on PCA (4) and kPCA (3) problems. For both problems, we measure the norm of the global Riemannian gradients. Additionally, we also measure the sum of principal angles [15] for kPCA. [3]

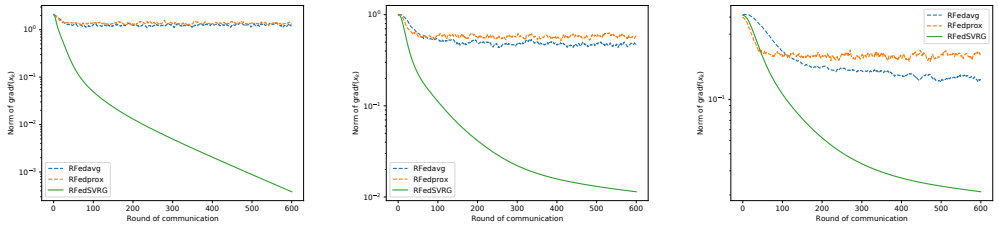
5.1 Comparison of the two consensus methods (7) and (8)

We first compare the two consensus methods (7) and (8). To this end, we randomly generate x_t and $k = 100$ points $x^{(i)}$ on the unit ball $\mathcal{S}^{d-1}$ 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 $\epsilon = 10^{-6}$. The results are shown in Table 1. From Table 1 we see that the tangent space mean (8) is indeed better than Karcher mean (7) in terms of both quality and CPU time.

[3] For the loss f in (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.

Table 1: Comparison of the two consensus methods (7) and (8). 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 (7)			Tangent space mean (8)		
		$d^2(x_{t+1}, x_t)$	$h(x_{t+1})$	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



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

Figure 2: Results for PCA (4). The y-axis denotes $\|\operatorname{grad} f(x_t)\|$. For each figure, the experiments are repeated and averaged over 10 times.

5.2 Experiments for PCA and kPCA on synthetic data

In this section, we report the results of the three algorithms for solving PCA (4) and kPCA (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 behavior of the proposed algorithms. We test highly heterogeneous real data later.

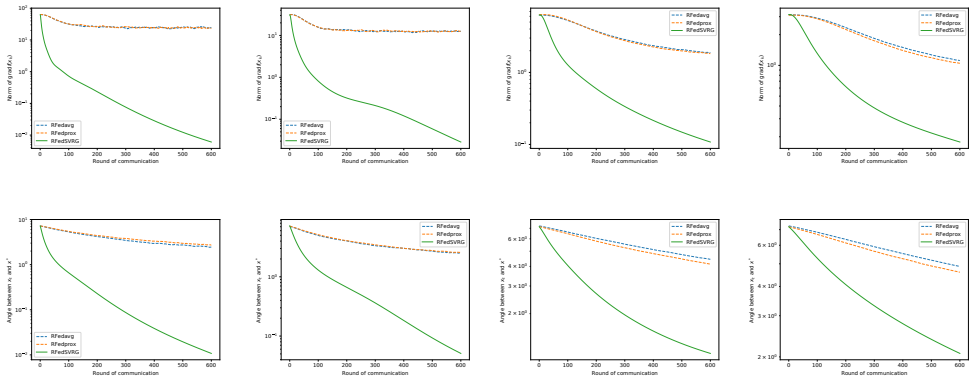
Experiments on PCA. We first test the three algorithms on the standard PCA problem (4). We test our codes with different numbers of agents n and set $k = n/10$ as the number of clients we pick up for each round. We terminate the algorithms if the number of rounds of communication exceeds 600. We sample 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 (28). We use the constant stepsizes for all three algorithms, and take $\mu = n/10$ for each choice of n. The results are presented in Figure 2, from which we see that only `RFedSVRG` can efficiently decrease $\|\operatorname{grad} f(x_t)\|$ to an acceptable level.

Experiments on kPCA. We now test the three algorithms on the kPCA problem (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 $\|\operatorname{grad} f(x_t)\|$ 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`.

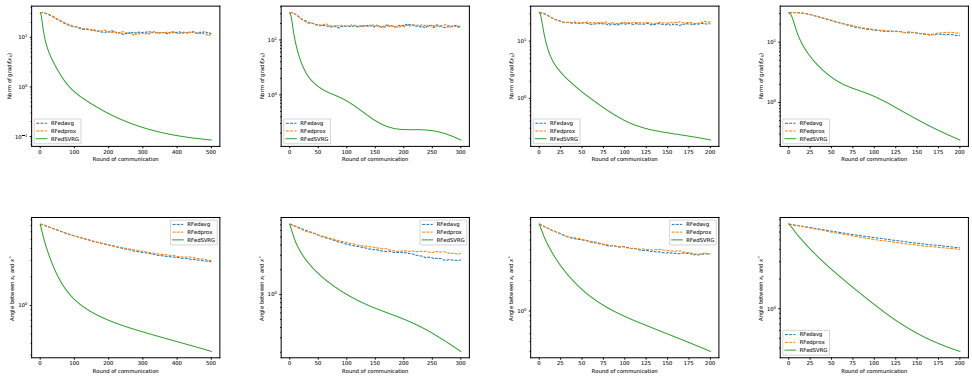
5.3 Experiments for kPCA on real data

We now show the numerical results of the three algorithms on real data. We focus on the kPCA problem (3) here. We test the three algorithms on three real data sets: the Iris dataset [10], the wine dataset [10] and the MNIST hand-written dataset [17]. 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



(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 $\|\operatorname{grad} f(x_t)\|$, 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.



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

Figure 4: Results for kPCA (3) with different number of inner loops $\tau = [1, 10, 50, 100]$. The y-axis of the figures in the first row denotes $\|\operatorname{grad} f(x_t)\|$, 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.

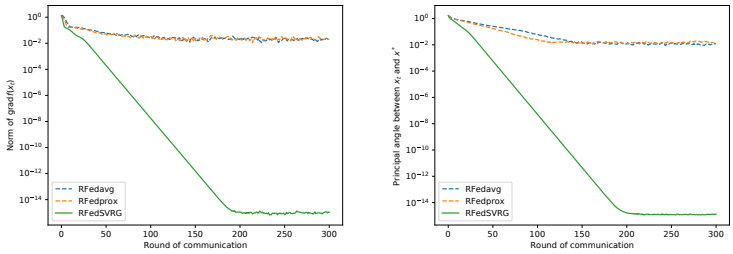
between the iterate and the ground truth. The experiments are repeated and averaged for 10 random initializations.

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 show that `RFedSVRG` is able to effectively decrease the norm of Riemannian gradient and the principal angles while the other two are not as efficient.

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. The comparison of the two rows of Figure 7 concludes that `RFedSVRG` shows better efficiency even with a larger number of clients n.

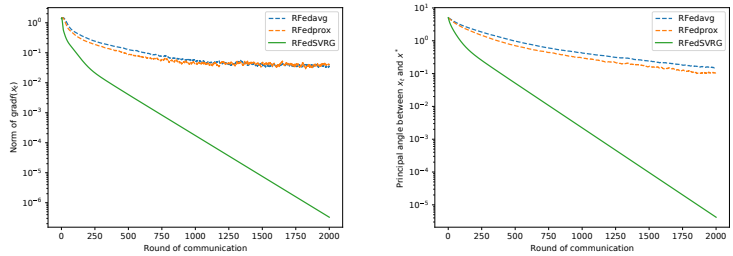
6 Conclusions

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.



(a) Gradient norm (b) Principal angle

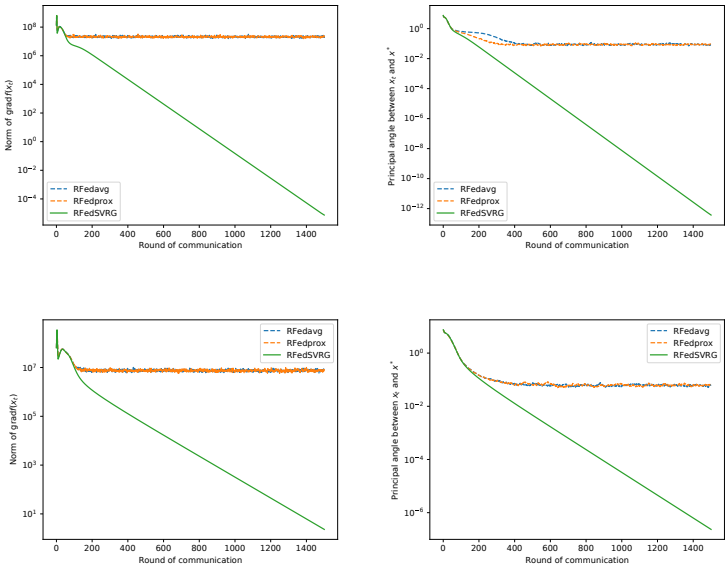
Figure 5: Results for kPCA (3) on Iris dataset. The data is in $\mathbb{R}^4$ ($d = 4$) and we take $r = 2$. The first figure is the norm of Riemannian gradient $\| \operatorname{grad} f(x_t)\|$ and the second is the principal angle between x_t and the true solution x^*.



(a) Gradient norm (b) Principal angle

Figure 6: Results for kPCA (3) with wine dataset. The data is in $\mathbb{R}^{13}$ ($d = 13$) and we take $r = 5$. The first figure is the norm of Riemannian gradient $\| \operatorname{grad} f(x_t)\|$ and the second is the principal angle between x_t and the true solution x^*.

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 (3) with MNIST dataset. The data is in $\mathbb{R}^{784}$ ($d = 784$) and we take $r = 5$. The first column is the norm of Riemannian gradient $\mathrm{grad}\, f(x_t)$ and the second is the principal angle between x_t and the true solution x^*. The two rows corresponds to $n = 100$ and $n = 200$. We take $k = n/10$ and $\tau = 5$ for all algorithms.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] P-A Absil and Jérôme Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[3] Foivos Alimisis, Peter Davies, Bart Vandereycken, and Dan Alistarh. Distributed principal component analysis with limited communication. *Advances in Neural Information Processing Systems*, 34, 2021.

[4] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(42):1455–1459, 2014.

[5] Nicolas Boumal. An introduction to optimization on smooth manifolds. To appear with Cambridge University Press, Jan 2022.

[6] Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2018.

[7] Zachary Charles and Jakub Konečnỳ. Convergence and accuracy trade-offs in federated learning and meta-learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2575–2583. PMLR, 2021.

[8] Shixiang Chen, Alfredo Garcia, Mingyi Hong, and Shahin Shahrampour. Decentralized riemannian gradient descent on the stiefel manifold. In *International Conference on Machine Learning*, pages 1594–1605. PMLR, 2021.

[9] Shixiang Chen, Alfredo Garcia, Mingyi Hong, and Shahin Shahrampour. On the local linear rate of consensus on the stiefel manifold. *arXiv preprint arXiv:2101.09346*, 2021.

[10] Michele Forina, Riccardo Leardi, Armanino C, and Sergio Lanteri. *PARVUS: An Extendable Package of Programs for Data Exploration*. Elsevier, Amsterdam, 01 1998.

[11] Andreas Grammenos, Rodrigo Mendoza Smith, Jon Crowcroft, and Cecilia Mascolo. Federated principal component analysis. *Advances in Neural Information Processing Systems*, 33:6453–6464, 2020.

[12] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[13] Tetsuya Kaneko, Simone Fiori, and Toshihisa Tanaka. Empirical arithmetic averaging over the compact stiefel manifold. *IEEE Transactions on Signal Processing*, 61(4):883–894, 2012.

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[15] Andrew V Knyazev and Peizhen Zhu. Principal angles between subspaces and their tangents. *arXiv preprint arXiv:1209.0523*, 2012.

[16] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[20] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[21] Grigory Malinovskiy, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtarik. From local sgd to local fixed-point methods for federated learning. In *International Conference on Machine Learning*, pages 6692–6701. PMLR, 2020.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[23] Aritra Mitra, Rayana Jaafar, George J Pappas, and Hamed Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems*, 34:14606–14619, 2021.

[24] Reese Pathak and Martin J Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. *Advances in Neural Information Processing Systems*, 33:7057–7066, 2020.

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

[26] J. Townsend, N. Koep, and S. Weichwald. PyManopt: a Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

[27] Roberto Tron, Bijan Afsari, and René Vidal. Riemannian consensus for manifolds with bounded curvature. *IEEE Transactions on Automatic Control*, 58(4):921–934, 2012.

[28] Loring W Tu. *An Introduction to Manifolds*. Springer Science & Universitext, 2011.

[29] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

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

[31] Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pages 1617–1638. PMLR, 2016.

[32] Ralf Zimmermann and Knut Hüper. Computing the riemannian logarithm on the stiefel manifold: metrics, methods and performance. *arXiv preprint arXiv:2103.12046*, 2021.

A Detailed Preliminary Results of Optimization on Riemannian Manifolds

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

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

Definition 10 (Differential and Riemannian gradients)**.** *Let $F : \mathcal{M} \to \mathcal{N}$ be a C^∞ map between two differential manifolds. At each point $x \in \mathcal{M}$, the differential of F is a mapping: $F_* : T_x\mathcal{M} \to T_x\mathcal{N}$ such that $\forall \xi \in T_x\mathcal{M}$, $F_*(\xi) \in T_x\mathcal{N}$ is given by $(F_*(\xi))(f) := \xi(f \circ F) \in \mathbb{R},\ f \in C^\infty_{F(x)}(\mathcal{M})$.*

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

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 11 (Geodesic and exponential mapping)**.** *Given $x \in \mathcal{M}$ and $\xi \in T_x\mathcal{M}$, the geodesic is the curve $\gamma : I \to \mathcal{M}$, $0 \in I \subset \mathbb{R}$ is an open set, so that $\gamma(0) = x$, $\dot\gamma(0) = \xi$ and $\nabla_{\dot\gamma}\dot\gamma = 0$ where $\nabla : T_x\mathcal{M} \times T_x\mathcal{M} \to T_x\mathcal{M}$ 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 $T_x\mathcal{M}$ to $\mathcal{M}$ s.t. $\operatorname{Exp}_x(\xi) := \gamma(1)$ with γ being the geodesic with $\gamma(0) = x$, $\dot\gamma(0) = \xi$. A natural corollary is $\operatorname{Exp}_x(t\xi) := \gamma(t)$ for $t \in [0, 1]$. Another useful fact is $d(x, \operatorname{Exp}_x(\xi)) = \|\xi\|_x$ since $\gamma'(0) = \xi$ which preserves the speed.

B Proofs

In this section we provide the proofs of lemmas and theorems mentioned in the main paper. We first finish the proof of Lemma 6:

Proof. [Proof of Lemma 6] By Cauchy-Schwarz inequality we have

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

□

Now we turn to the proof of Theorem 7. We would utilize the following lemma:

Lemma 12. *Under the same settings as Theorem 7, we have*

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

Proof. [Proof of Lemma 12] From the update we know that

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

i.e.

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

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

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

Using Lipschitz smooth of f_i again and the tangent space mean (8), we have

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

where we used the tangent space mean (8) for the first equality. □

Now we are ready to present the proof of Theorem 7.

Proof. [Proof of Theorem 7] By taking $\eta^{(i)} \leq \frac{1}{L}$, from Lemma 12 we have

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

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

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

which yields (10) immediately. □

Before we present the proof of Theorem 8, we need the following lemma, which is adopted from [30].

Lemma 13 (Lemma 2 in [30])**.** *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 (9) with initial point x_t, then it holds:*

$$\mathbb{E} \| \operatorname{grad} f(x_\ell^{(i)}) \|^2 \leq \frac{R_\ell - R_{\ell+1}}{\delta_\ell}, \ \ell = 0, ..., \tau_i - 1, \tag{13}$$

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 8:

Proof. [Proof of Theorem 8] 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 (13), we note that if we set $\eta < \frac{1}{L + 2c_{\ell+1}\zeta}(1 - \frac{c_{\ell+1}}{\beta})$, then we have $\delta^{(i)} := \min_{\ell=0,\ldots,\tau_i} \delta_\ell > 0$. In this case, summing (13) over $\ell = 0, 1, ..., \tau_i - 1$ yields

$$\frac{1}{\tau_i} \sum_{\ell=0,\ldots,\tau_i-1} \mathbb{E} \| \operatorname{grad} f(x_\ell^{(i)}) \|^2 \leq \frac{R_0 - R_{\tau_i}}{\tau_i \delta^{(i)}} \leq \mathbb{E} \left(\frac{f(x_t) - f(x_{\tau_i}^{(i)})}{\tau_i \delta^{(i)}} \right), \tag{14}$$

since $R_0 = f(x_t)$ and $R_{\tau_i} = \mathbb{E}[f(x_{\tau_i}^{(i)}) + c_\ell \| \operatorname{Exp}_{x_t}^{-1}(x_{\tau_i}^{(i)}) \|^2] \geq \mathbb{E}[f(x_{\tau_i}^{(i)})]$. Now we take $\beta = L\zeta^{1/2}/n^{1/3}$ and $\eta = 1/(10Ln^{2/3}\zeta^{1/2})$[4]. 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

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

[4]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 $\ell = 0, ..., \tau_i$.

where

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

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

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

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

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

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

Now summing (14) over $t = 0, ..., T-1$ with $\delta^{(i)} \geq \frac{\eta}{2}$ we get

$$\frac{1}{T}\sum_{t=0,\ldots,T-1}\frac{1}{\tau_i}\sum_{\ell=0,\ldots,\tau_i-1}\mathbb{E}\|\operatorname{grad} f(x_\ell^{(i)})\|^2 \leq \frac{2\Delta}{\tau\eta T}, \tag{15}$$

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

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

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

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

Before we present the proof of Theorem 9, we need the following lemma [31].

Lemma 14 (Corollary 8 in [31])**.** *Suppose the sectional curvature of $\mathcal{M}$ is lower bounded by $\kappa_{\min}$ and we update $x_{t+1} \leftarrow \operatorname{Exp}_{x_t}(-\eta_t g_t)$. Suppose also that the update sequence $\{x_t\} \subset \mathcal{D}$ where $\mathcal{D}$ is a compact set with diameter D, then for any $x \in \mathcal{M}$ it holds:*

$$\langle -g_t, \operatorname{Exp}_{x_t}^{-1}(x)\rangle \leq \frac{1}{2\eta_t}(d^2(x_t, x) - d^2(x_{t+1}, x)) + \frac{\zeta\eta_t}{2}\|g_t\|^2. \tag{16}$$

where ζ is given in (11)*.*

We now present the proof of Theorem 9.

Proof. [Proof of Theorem 9] From Lemma 14 we get

$$\langle\frac{1}{k}\sum_{i\in S_t}\operatorname{Exp}_{x_t}^{-1}(x^{(i)}), \operatorname{Exp}_{x_t}^{-1}(x)\rangle \leq \frac{1}{2}(d^2(x_t, x) - d^2(x_{t+1}, x)) + \frac{\zeta}{2}\|\frac{1}{k}\sum_{i\in S_t}\operatorname{Exp}_{x_t}^{-1}(x^{(i)})\|^2, \tag{17}$$

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

$$-\eta\langle\frac{1}{n}\sum_{i=1,\ldots,n}\operatorname{grad} f_i(x_t), \operatorname{Exp}_{x_t}^{-1}(x)\rangle \leq \frac{1}{2}(d^2(x_t, x) - d^2(x_{t+1}, x)) + \frac{\zeta}{2}\|\frac{1}{n}\sum_{i=1,\ldots,n}\operatorname{Exp}_{x_t}^{-1}(x^{(i)})\|^2. \tag{18}$$

Now use the geodesic convexity of f_i and (18), we have (denote $\Delta_t := f(x_t) - f(x^*)$ and $\Delta_t^i := f_i(x_t) - f_i(x^*)$)

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

Summing this inequality over $i = 1, ..., n$, we get

$$\begin{aligned}\Delta_t \leq & -\langle \frac{1}{n} \sum_{i=1,\dots,n} \operatorname{grad} f_i(x_t), \operatorname{Exp}_{x_t}^{-1}(x^*)\rangle \\ \leq & \frac{1}{2\eta}(d^2(x_t, x^*) - d^2(x_{t+1}, x^*)) + \frac{\zeta}{2\eta}\|\frac{1}{n}\sum_{i=1,\dots,n} \operatorname{Exp}_{x_t}^{-1}(x^{(i)})\|^2 \\ \leq & \frac{1}{2\eta}(d^2(x_t, x^*) - d^2(x_{t+1}, x^*)) + \frac{\zeta\eta}{2n}\|\operatorname{grad} f(x_t)\|^2.\end{aligned} \tag{19}$$

Again from Lemma 12 we get

$$\Delta_{t+1} - \Delta_t \leq (-\eta_t^{(i)} + \frac{(\eta_t^{(i)})^2 L}{2})\|\operatorname{grad} f(x_t)\|^2. \tag{20}$$

Now multiply (20) by ζ and add it to (19), we get

$$\zeta\Delta_{t+1} - (\zeta-1)\Delta_t \leq \zeta\left(\frac{\eta}{2n} - \eta + \frac{\eta^2 L}{2}\right)\|\operatorname{grad} f(x_t)\|^2 + \frac{1}{2\eta}(d^2(x_t, x^*) - d^2(x_{t+1}, x^*)). \tag{21}$$

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

$$\zeta\Delta_{t+1} - (\zeta-1)\Delta_t \leq \frac{1}{2\eta}(d^2(x_t, x^*) - d^2(x_{t+1}, x^*)). \tag{22}$$

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

$$\zeta\Delta_T + \sum_{t=0}^{T-1}\Delta_t \leq (\zeta-1)\Delta_1 + \frac{d^2(x_0, x^*)}{2\eta}. \tag{23}$$

Also by (20) we know $\Delta_{t+1} \leq \Delta_t$, thus

$$\Delta_T \leq \frac{\zeta D^2}{2\eta(\zeta + T - 2)}. \tag{24}$$

□

C `RFedAvg` and `RFedProx` algorithms

`FedAvg` [22] and `FedProx` [19] 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:

$$x_{\ell+1}^{(i)} \leftarrow x_\ell^{(i)} - \eta^{(i)}\nabla f_i(x_{\ell+1}^{(i)}), \tag{25}$$

while `FedProx` solves a local proximal point subproblem:

$$x^{(i)} \leftarrow \operatorname*{argmin}_x f_i(x) + \frac{\mu}{2}\|x - x_t\|^2. \tag{26}$$

For `RFedAvg`, which is the Riemannian counterpart of `FedAvg`, (25) 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`, (26) is replaced by

$$x_{t+1}^{(i)} \leftarrow \operatorname*{argmin}_{x\in\mathcal{M}} f_i(x) + \frac{\mu}{2}d^2(x, x_t), \tag{27}$$

where $d(x, y)$ is the geodesic distance between x and y. In the implementation of `RFedProx`, (27) is solved by Riemannian gradient descent:

$$x_{\ell+1}^{(i)} \leftarrow \operatorname{Exp}_{x_\ell^{(i)}}(-\eta^{(i)}\operatorname{grad} h_i(x_\ell^{(i)})),\ \ell = 0, ..., \tau_i - 1. \tag{28}$$

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

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**
      $x_{\ell+1}^{(i)} \leftarrow \mathrm{Exp}_{x_\ell^{(i)}} \left(-\eta^{(i)} \operatorname{grad} f_i(x_\ell^{(i)})\right)$;
    end
    Send the obtained $x_{\tau_i}^{(i)}$ to the central server;
  end
  The central server aggregates the points by the tangent space mean (8);
end

Algorithm 3: Riemannian FedProx Algorithm

input : n, k, T, μ, γ
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 γ approximate solution;
    Send the obtained $x^{(i)}$ to the central server;
  end
  The central server aggregates the points by the tangent space mean (8);
end