RIEMANNIAN NATURAL GRADIENT METHODS

JIANG HU*, RUICHENG AO†, ANTHONY MAN-CHO SO‡, MINGHAN YANG§, AND ZAIWEN WEN¶

Abstract. This paper studies large-scale optimization problems on Riemannian manifolds whose objective function is a finite sum of negative log-probability losses. Such problems arise in various machine learning and signal processing applications. By introducing the notion of Fisher information matrix in the manifold setting, we propose a novel Riemannian natural gradient method, which can be viewed as a natural extension of the natural gradient method from the Euclidean setting to the manifold setting. We establish the almost-sure global convergence of our proposed method under standard assumptions. Moreover, we show that if the loss function satisfies certain convexity and smoothness conditions and the input-output map satisfies a Riemannian Jacobian stability condition, then our proposed method enjoys a local linear—or, under the Lipschitz continuity of the Riemannian Jacobian of the input-output map, even quadratic—rate of convergence. We then prove that the Riemannian Jacobian stability condition will be satisfied by a two-layer fully connected neural network with batch normalization with high probability, provided that the width of the network is sufficiently large. This demonstrates the practical relevance of our convergence rate result. Numerical experiments on applications arising from machine learning demonstrate the advantages of the proposed method over state-of-the-art ones.

Key words. Manifold optimization, Riemannian Fisher information matrix, Kronecker-factored approximation, Natural gradient method

AMS subject classifications. 90C06, 90C22, 90C26, 90C56

1 Introduction

Manifold constrained learning problems are ubiquitous in machine learning, signal processing, and deep learning ; see, e.g., [6, 14, 32, 40, 17]. In this paper, we focus on manifold optimization problems of the form

$$\min_{\Theta\in\mathcal{M}} \quad \Psi(\Theta) := -\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} \log p(y|f(x,\Theta)), \tag{1.1}$$

where $\mathcal{M}$ is either an embedded submanifold of $\mathbb{R}^{m\times n}$ or a quotient manifold whose total space is an embedded submanifold of $\mathbb{R}^{m\times n}$, $\Theta\in\mathcal{M}$ is the parameter to be estimated, $\mathcal{S}$ is a collection of $|\mathcal{S}|$ data pairs (x,y) with $x\in\mathcal{X}, y\in\mathcal{Y}$, $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces, respectively, $f(\cdot,\Theta):\mathcal{X}\to\mathcal{Y}$ is a mapping from the input space to the output space, and $p(y|f(x,\Theta))$ is the conditional probability of taking y conditioning on $f(x,\Theta)$. If the conditional distribution is assumed to be Gaussian, the objective function in (1.1) reduces to the square loss. When the conditional distribution $p(y|f(x,\Theta))$ obeys the multinomial distribution, the corresponding objective function is the cross-entropy loss. As an aside, it is worth noting the equivalence between the negative log probability loss and Kullback-Leibler (KL) divergence shown in [38].

Let us take the low-rank matrix completion (LRMC) problem [14, 32] as an example and explain how it can be fitted into the form (1.1). The goal of LRMC

*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, NT, Hong Kong (hujiangopt@gmail.com).

†School of Mathematical Sciences, Peking University, China (archer_arc@pku.edu.cn).

‡Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, NT, Hong Kong (manchoso@se.cuhk.edu.hk).

§Beijing International Center for Mathematical Research, Peking University, China (yangminghan@pku.edu.cn).

¶Beijing International Center for Mathematical Research, Center for Data Science and College of Engineering, Peking University, Beijing, China (wenzw@pku.edu.cn).

is to recover a low-rank matrix from an observed matrix X of size $n \times N$. Denote by Ω the set of indices of known entries in X, the rank-p LRMC problem amounts to solving

$$\min_{U \in \mathrm{Gr}(n,p), A \in \mathbb{R}^{p \times N}} \frac{1}{2} \left\| \mathcal{P}_\Omega(UA - X) \right\|^2, \tag{1.2}$$

where $\mathrm{Gr}(n,p)$ is the Grassmann manifold consists of all p-dimensional subspaces in $\mathbb{R}^n$. The operator $\mathcal{P}_\Omega(X)$ is defined in an element-wise manner with $\mathcal{P}_\Omega(X_{ij}) = X_{ij}$ if $(i,j) \in \Omega$ and 0 otherwise. Partitioning $X = [x_1, \ldots, x_N]$ leads to the following equivalent formulation

$$\min_{U \in \mathrm{Gr}(n,p), a_i \in \mathbb{R}^p} \frac{1}{2N} \sum_{i=1}^{N} \left\| \mathcal{P}_{\Omega_{x_i}} (Ua_i - x_i) \right\|^2,$$

where $x_i \in \mathbb{R}^n$ and the j-th element of $\mathcal{P}_{\Omega_{x_i}}(v)$ is v_j if $(i,j) \in \Omega$ and 0 otherwise. Given U, we can obtain a_i by solving a least squares problem, i.e.,

$$a_i = a(U; x_i) := \arg\min_a \| \mathcal{P}_{\Omega_{x_i}} (Ua - x_i) \|^2.$$

Then, the LRMC problem can be written as

$$\min_{U \in \mathrm{Gr}(n,p)} \quad \Psi(U) := \frac{1}{2N} \sum_{i=1}^{N} \| \mathcal{P}_{\Omega_{x_i}} (Ua(U; x_i) - x_i) \|^2. \tag{1.3}$$

For the Gaussian distribution $p(y|z) = \frac{1}{\sqrt{(2\pi)^n}} \exp(-\frac{1}{2}(y-z)^\top (y-z))$, it holds that $-\log p(y|z) = \frac{1}{2}\|y-z\|^2 + \frac{n \log(2\pi)}{2}$. Hence, problem (1.3) is a special case of problem (1.1), in which $\mathcal{S} = \{(x_i, 0)\}_{i=1}^N$, $\mathcal{X} = \mathbb{R}^n$, $\mathcal{Y} = \mathbb{R}^n$, $f(x, U) = \mathcal{P}_{\Omega_x} (Ua(U; x) - x)$, $\mathcal{M} = \mathrm{Gr}(n,p)$, and $p(y|z) = \frac{1}{\sqrt{(2\pi)^n}} \exp(-\frac{1}{2}(y-z)^\top (y-z))$. Other applications that can be fitted into the form (1.1) will be introduced in Section 4.

1.1 Motivation of this work Since the calculation of the gradient of Ψ in (1.1) can be expensive when the dataset $\mathcal{S}$ is large, various approximate or stochastic methods for solving (1.1) have been proposed. On the side of first-order methods, we have the stochastic gradient method [47], stochastic variance-reduced gradient method [31], and adaptive gradient methods [19, 35] for solving (1.1) in the Euclidean setting (i.e., $\mathcal{M} = \mathbb{R}^{m \times n}$). We refer the reader to the book [37] for variants of these algorithms and a comparison of their performance. For the general manifold setting, by utilizing manifold optimization techniques [1, 26, 13], Riemannian versions of the stochastic gradient method [11], stochastic variance-reduced gradient method [52, 67, 29], and adaptive gradient methods [10] have been developed.

On the side of second-order methods, existing algorithms for solving (1.1) in the Euclidean setting (i.e., $\mathcal{M} = \mathbb{R}^{m \times n}$) can be divided into two classes. The first is based on approximate Newton or quasi-Newton techniques; see, e.g., [48, 44, 15, 60, 61, 21, 45]. The second is the natural gradient-type methods, which are based on the Fisher information matrix (FIM) [4]. When the FIM can be approximated by a Kronecker-product form, the natural gradient direction can be computed using relatively low computational cost. It is well known that second-order methods can accelerate convergence by utilizing curvature information. In particular, natural gradient-type methods can perform much better than the stochastic gradient method

[39, 63, 7, 62, 9, 42] in the Euclidean setting. The connections between natural gradient methods and second-order methods have been established in [38]. Compared with the approximate Newton/quasi-Newton-type methods, methods based on FIM are shown to be more efficient when tackling large-scale learning problems. For the general manifold setting, Riemannian stochastic quasi-Newton-type and Newton-type methods [34, 33, 65] have been proposed by utilizing the second-order manifold geometry and variance reduction techniques. However, to the best of our knowledge, there is currently no Riemannian natural gradient-type method for solving (1.1). In view of the efficiency of Euclidean natural gradient-type methods, we are motivated to develop their Riemannian analogs for solving (1.1).

1.2 Our contributions In this paper, we develop a new Riemannian natural gradient method for solving (1.1). Our main contributions are summarized as follows.

- We introduce the Riemannian FIM (RFIM) and Riemannian empirical FIM (REFIM) to approximate the Riemannian Hessian. These notions extend the corresponding ones for the Euclidean setting [4, 38] to the manifold setting. Then, we propose an adaptive regularized Riemannian natural gradient descent (RNGD) method. We show that for some representative applications, Kronecker-factorized approximations of RFIM and REFIM can be constructed, which reduce the computational cost of the Riemannian natural gradient direction. Our experiment results demonstrate that although RNGD is a second-order-type method, it has low per-iteration cost and enjoys favorable numerical performances.
- Under some mild conditions, we prove that RNGD globally converges to a stationary point of (1.1) almost surely. Moreover, if the loss function satisfies certain convexity and smoothness conditions and the input-output map f satisfies a Riemannian Jacobian stability condition, then we can establish the local linear—or, under the Lipschitz continuity of the Riemannian Jacobian of f, even quadratic—rate of convergence of the method by utilizing the notion of second-order retraction. We then show that for a two-layer neural network with batch normalization, the Riemannian Jacobian stability condition will be satisfied with high probability when the width of the network is sufficiently large.

1.3 Notation For an $m \times n$ matrix Θ, we denote its Frobenius norm by $\|\Theta\|$ and its vectorization by $\theta = \mathrm{vec}(\Theta) \in \mathbb{R}^{mn}$. For a smooth function $h : \mathbb{R}^{m\times n} \to \mathbb{R}$, we use $\nabla h(\Theta) \in \mathbb{R}^{m\times n}$ to denote its Euclidean gradient at $\Theta \in \mathbb{R}^{m\times n}$. For simplicity, we set $r = mn$. When no confusion can arise, we use $\nabla h(\theta)$ to denote the vectorization of $\nabla h(\Theta)$. We use $\nabla^2 h(\theta) \in \mathbb{R}^{r\times r}$ to denote the Euclidean Hessian of h at $\theta \in \mathbb{R}^r$. We denote the tangent space to $\mathcal{M}$ at Θ by $T_\Theta\mathcal{M}$. We write $d \in T_\theta\mathcal{M}$ to mean $\mathrm{mat}(d) \in T_\Theta\mathcal{M}$, where $d \in \mathbb{R}^r$ and $\mathrm{mat}(d)$ converts d into a m-by-n matrix. For a retraction R defined on $\mathcal{M}$, we write $R_\theta(d) := \mathrm{vec}(R_\Theta(D))$ for $D \in T_\Theta\mathcal{M}$, $\theta = \mathrm{vec}(\Theta)$, and $d = \mathrm{vec}(D)$. We shall use θ and Θ interchangeably when no confusion can arise. Basically, Θ is used when we want to utilize the manifold structure, while θ is used when we want to utilize the vector space structure of the ambient space.

1.4 Organization We begin with the preliminaries on manifold optimization and natural gradient methods in Section 2. In Section 3, we introduce the RFIM and its empirical version REFIM and derive some of their properties. Then, we present our proposed RNGD method by utilizing the RFIM and REFIM. In Section 4, we discuss practical implementations of the RNGD method when problem (1.1) enjoys certain

Kronecker-product structure. In Section 5, we study the convergence behavior of the RNGD method under various assumptions. Finally, we present numerical results in Section 6.

2 Preliminaries

2.1 Manifold optimization Consider the optimization problem

$$\min_{\Theta \in \mathcal{M}} \quad h(\Theta), \tag{2.1}$$

where $\mathcal{M}$ is either an embedded submanifold of $\mathbb{R}^{m\times n}$ or a quotient manifold whose total space is an embedded submanifold of $\mathbb{R}^{m\times n}$ and $h : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a smooth function. For every $\Theta \in \mathcal{M}$, we endow the tangent space $T_\Theta\mathcal{M}$ with a general Riemannian metric $\langle U, V\rangle_\Theta := \mathrm{vec}(U)^\top D(\theta)\mathrm{vec}(V)$, where $D(\theta) \in \mathbb{R}^{r\times r}$ is symmetric and positive definite on $T_\theta\mathcal{M}$. The design and analysis of numerical algorithms for tackling (2.1) have been extensively studied over the years; see, e.g., [1, 26, 13] and the references therein. One of the key constructs in the design of manifold optimization algorithms is the retraction operator. A smooth mapping $R : T\mathcal{M} := \cup_{\Theta\in\mathcal{M}} T_\Theta\mathcal{M} \to \mathcal{M}$ is called a retraction operator if

- $R_\Theta(0) = \Theta$,
- $\mathrm{D}R_\Theta(0)[\xi] := \frac{\mathrm{d}}{\mathrm{d}t} R_\Theta(t\xi)\,|_{t=0} = \xi$, for all $\xi \in T_\Theta\mathcal{M}$.

We call R a second-order retraction [1, Proposition 5.5.5] if $\mathcal{P}_{T_\Theta\mathcal{M}}\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2} R_\Theta(t\xi)|_{t=0}\right) = 0$ for all $\Theta \in \mathcal{M}$ and $\xi \in T_\Theta\mathcal{M}$. Some examples of second-order retraction can be found in [3, Theorem 22]. Another key concept is the Riemannian gradient. Given $\Theta \in \mathcal{M}$, the vectorization of the Riemannian gradient $\widetilde{\mathrm{grad}}\, h(\Theta) \in \mathbb{R}^{m\times n}$ of h at Θ is given by

$$\widetilde{\mathrm{grad}}\, h(\theta) = D(\theta)^{-1}\mathcal{P}_{T_\theta\mathcal{M}}(\nabla h(\theta)) \in \mathbb{R}^r,$$

where $\mathcal{P}_{T_\theta\mathcal{M}}(\cdot)$ is the orthogonal projection operator onto $T_\theta\mathcal{M}$. The retraction-based methods for solving (2.1) perform updates of the form

$$\Theta^{k+1} = R_{\Theta^k}(td^k), \tag{2.2}$$

where d^k is a descent direction in the tangent space $T_{\Theta^k}\mathcal{M}$ and $t > 0$ is the step size. The retraction operator R constrains the iterates on $\mathcal{M}$. For the case where $\mathcal{M}$ is an embedded submanifold, we always take the Euclidean metric as the Riemannian metric (i.e., $\langle U, V\rangle_\Theta = \mathrm{vec}(U)^\top\mathrm{vec}(V)$ for any $\Theta \in \mathcal{M}$) and use $\mathrm{grad}\, h(\theta) \in \mathbb{R}^r$ and $\mathrm{Hess}\, h(\theta) \in \mathbb{R}^{r\times r}$ to denote the Riemannian gradient and Riemannian Hessian of h under the Euclidean metric, respectively. For the case where $\mathcal{M}$ is a quotient manifold, we use a Riemannian metric that satisfies the horizontally invariant property in [1, Equation (3.38)], so that the Riemannian norm of a vector on $T_\theta\mathcal{M}$ does not depend on the representative element of θ in $\mathcal{M}$. We also assume that the total space has a retraction satisfying the projection property in [1, Equation (4.9)], so that the retraction R on $\mathcal{M}$ can be defined according to [1, Equation (4.10)].

2.2 Natural gradient descent method The natural gradient descent (NGD) method was originally proposed in [4] to solve (1.1) in the Euclidean setting (i.e., $\mathcal{M} = \mathbb{R}^{m\times n}$). Suppose that y follows the conditional distribution $P_{y|f(x,\Theta)}$. Consider the population loss under $P_{y|x}(\Theta) := P_{y|f(x,\Theta)}$, i.e.,

$$\Phi(\Theta) := -\mathbb{E}_{P_x}\left[\mathbb{E}_{P_{y|x}(\Theta)} \log p(y|f(x,\Theta))\right]. \tag{2.3}$$

When $P_{y|x}(\Theta)$ and P_x are replaced by their empirical counterparts defined using $\mathcal{S}$, the population loss $\Phi(\Theta)$ reduces to the empirical loss $\Psi(\Theta)$. Now, the FIM associated with Φ is defined as

$$F(\theta) := \mathbb{E}_{P_x}[\mathbb{E}_{P_{y|x}(\theta)}[\nabla \log p(y|f(x,\theta))\nabla \log p(y|f(x,\theta))^\top]] \in \mathbb{R}^{r\times r}.$$

Under certain regularity condition [20], we can interchange the order of expectation and derivative to obtain $F(\theta) = \nabla^2\Phi(\theta)$. In what follows, we assume that such a regularity condition holds. Since the distribution of x is unknown, we set P_x to be the empirical distribution defined by $\mathcal{S}$. In practice, we may only be able to get hold of an empirical counterpart of $P_{y|x}(\Theta)$. The empirical FIM (EFIM) associated with Ψ is then defined by replacing $P_{y|x}(\Theta)$ with its empirical counterpart [53], i.e.,

$$\bar{F}(\theta) := \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} \nabla \log p(y|f(x,\theta))\nabla \log p(y|f(x,\theta))^\top.$$

With the FIM, the natural gradient direction is given by

$$\tilde{\nabla}\Phi(\theta) := (F(\theta))^{-1}\nabla\Phi(\theta) \in \mathbb{R}^r.$$

It is shown in [5, Theorem 1] and [43, Proposition 1] that $\tilde{\nabla}\Phi(\theta)$ is the steepest descent direction in the sense that

$$-\frac{\tilde{\nabla}\Phi(\theta)}{\|\nabla\Phi(\theta)\|_{(F(\theta))^{-1}}} = \lim_{\epsilon\to 0}\frac{1}{\epsilon}\operatorname*{arg\,min}_{d\in\mathbb{R}^r:\mathrm{KL}(P_{x,y}(\theta+d)\|P_{x,y}(\theta))\le\epsilon^2/2}\Phi(\theta+d),$$

where $\|\nabla\Phi(\theta)\|_{(F(\theta))^{-1}} := \sqrt{\nabla\Phi(\theta)(F(\theta))^{-1}\nabla\Phi(\theta)}$.

In the k-th iteration, the iterative scheme of NGD for minimizing (2.3) is

$$\theta^{k+1} = \theta^k - t_k\tilde{\nabla}\Phi(\theta^k),$$

where $t_k > 0$ is a step size. In the case where $F(\theta)$ is computationally expensive or inaccessible, we use the EFIM instead of the FIM. The connections between NGD and second-order methods are presented in [38].

3 Riemannian natural gradient method

3.1 Fisher information matrix on manifold When the parameter to be estimated Θ lies on $\mathcal{M}$, the Euclidean natural gradient direction need not lie on the tangent space to $\mathcal{M}$ at Θ and thus cannot be used as a search direction in retraction-based methods. To overcome this difficulty, we first introduce the RFIM, which is defined as

$$F^R(\theta) := \mathbb{E}_{P_x}\left[\mathbb{E}_{P_{y|x}(\theta)}\left[\mathrm{grad}\log p(y|f(x,\theta))\mathrm{grad}\log p(y|f(x,\theta))^\top\right]\right] \in \mathbb{R}^{r\times r}, \tag{3.1}$$

where $\mathrm{grad}\log p(y|f(x,\theta))$ is the Riemannian gradient of $\log p(y|f(x,\theta))$ with respect to θ under the Euclidean metric.[1] Note that the generalization of FIM in the manifold setting has been developed in [55, 12]. The RFIM defined in (3.1) can be regarded as an extrinsic representation (i.e., an r-by-r matrix) of the said generalization. Such

[1] The RFIM should not be confused with the Riemannian Fisher information metric. For any two tangent vectors $u, v \in T_\theta\mathcal{M}$, the Riemannian Fisher information metric associated with the RFIM (3.1) is given by $u^\top F^R(\theta)v$.

an extrinstic representation relies on the Euclidean representation of the Riemannian gradient in the total space and presents a straightforward way to compute RFIM. It is easy to see that the range of $F^R(\theta)$ is included in $T_\theta\mathcal{M}$. Assuming that $F^R(\theta)$ is positive definite on $T_\theta\mathcal{M}$, we define the Riemannian natural gradient direction $d^R(\theta)$ as

$$d^R(\theta) := (F^R(\theta))^{-1}\mathrm{grad}\,\Phi(\theta) \in \mathbb{R}^r, \tag{3.2}$$

which is a vector on $T_\theta\mathcal{M}$. The following theorem justifies our definition of RFIM. It extends the corresponding results on FIM given in [5, Theorem 1] and [43, Proposition 1].

THEOREM 3.1. *Let $\mathcal{M}$ be either an embedded submanifold of $\mathbb{R}^{m\times n}$ or a quotient manifold whose total space is an embedded submanifold of $\mathbb{R}^{m\times n}$, and $\Phi : \mathcal{M} \to \mathbb{R}$ be the function given in (2.3). Given $\Theta \in \mathcal{M}$, suppose that $F^R(\theta)$ is positive definite on $T_\theta\mathcal{M}$. Then, for any second-order retraction R on $\mathcal{M}$, the steepest descent direction in the tangent space to $\mathcal{M}$ at Θ is given by $-d^R(\theta)$ in (3.2), i.e.,*

$$\frac{-d^R(\theta)}{\|\mathrm{grad}\,\Phi(\theta)\|_{(F^R(\theta))^{-1}}} = \lim_{\epsilon\to 0}\frac{1}{\epsilon}\operatorname*{arg\,min}_{d\in T_\theta\mathcal{M}:\mathbb{E}_{P_x}\left[\mathrm{KL}\left(P_{y|x}(R_\theta(d))\|P_{y|x}(\theta)\right)\right]\le\epsilon^2/2} \Phi(R_\theta(d)), \tag{3.3}$$

where $\|\mathrm{grad}\,\Phi(\theta)\|_{(F^R(\theta))^{-1}} = \sqrt{\mathrm{grad}\,\Phi(\theta)^\top (F^R(\theta))^{-1}\mathrm{grad}\,\Phi(\theta)}$.

Proof. For $\Theta \in \mathcal{M}$, from the definition

$$\mathrm{KL}(P_{y|x}(\theta)||P_{y|x}(R_\theta(td))) = \mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,\theta)) - \mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,R_\theta(td))),$$

we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(P_{y|x}(\theta)||P_{y|x}(R_\theta(td)))\mid_{t=0} &= -\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,R_\theta(td)))\mid_{t=0}\\ &= -d^\top\nabla\mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,\theta)).\end{aligned}$$

By definition of the Riemannian gradient, we obtain

$$d^\top\mathrm{grad}\,\mathrm{KL}(P_{y|x}(\theta)||P_{y|x}(R_\theta(td)))\mid_{t=0} = -d^\top\nabla\mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,\theta)), \quad \forall d\in T_\theta\mathcal{M},$$

where $\mathrm{grad}\,\mathrm{KL}(P_{y|x}(\theta)\parallel P_{y|x}(R_\theta(td)))\mid_{t=0}\in T_\theta\mathcal{M}$. Then, we have

$$\mathrm{grad}\,\mathrm{KL}(P_{y|x}(\theta)\parallel P_{y|x}(R_\theta(td)))\mid_{t=0} = -\mathrm{grad}\,\mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,\theta)).$$

Accordingly, using the Leibniz integral rule and the property of second-order retractions [1, Proposition 5.5.5], we have the second-order derivative

$$\begin{aligned}&\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathrm{KL}(P_{y|x}(\theta)||P_{y|x}(R_\theta(td)))\mid_{t=0}\\ =&\mathbb{E}_{P_{y|x}(\theta)}[d^\top\mathrm{grad}\,\log p(y|f(x,\theta))\,(\mathrm{grad}\,\log p(y|f(x,\theta)))^\top d].\end{aligned}$$

It follows that $\mathrm{grad}\,\mathbb{E}_{P_{y|x}(\theta)}\log p(y|f(x,\theta)) = 0$. By the definition of F^R, we conclude that

$$\mathbb{E}_{P_x}\mathrm{KL}(P_{y|x}(\theta)||P_{y|x}(R_\theta(d)) = \frac{1}{2}d^\top F^R(\theta)d + O(d^3), \quad \forall d\in T_\theta\mathcal{M}.$$

From the fact [43, Proposition 1] that

$$\frac{-A^{-1}\nabla h(\theta)}{\|\nabla h(\theta)\|_{A^{-1}}} = \lim_{\epsilon\to 0}\frac{1}{\epsilon}\operatorname*{arg\,min}_{d:\|d\|_A\le\epsilon} h(\theta+d),$$

where A is a positive definite matrix and $\|d\|_{A^{-1}} = \sqrt{d^\top A^{-1} d}$, we have

$$\frac{-B^{-1}\nabla(\Phi \circ R_\theta)(0)}{\|\nabla(\Phi \circ R_\theta)(0)\|_{B^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d \in T_\theta\mathcal{M}: \|d\|_B \leq \epsilon} \Phi(R_\theta(d)), \tag{3.4}$$

where $B : T_\theta\mathcal{M} \to T_\theta\mathcal{M}$ is a positive definite linear operator. Note that for all $u \in T_\theta\mathcal{M}$, it holds that

$$\nabla(\Phi \circ R_\theta)(0)[u] = \nabla\Phi(R_\theta(0))[\mathrm{D}R_\theta(0)[u]] = u^\top \operatorname{grad}\Phi(\theta).$$

This gives

$$\nabla(\Phi \circ R_\theta)(0) = \operatorname{grad}\Phi(\theta).$$

Substituting the above into (3.4) and letting $B = F^R(\theta)$, we have

$$\frac{-(F^R(\theta))^{-1}\operatorname{grad}\Phi(\theta)}{\|\operatorname{grad}\Phi(\theta)\|_{(F^R(\theta))^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d \in T_\theta\mathcal{M}: \|d\|_{F^R(\theta)} \leq \epsilon} \Phi(R_\theta(d)). \tag{3.5}$$

Therefore, (3.3) holds for any second-order retraction R. This completes the proof. □

Note that for an embedded submanifold $\mathcal{M}$ endowed with the Euclidean metric, the Riemannian Hessian [2, Equation 7] of Φ at θ along $u \in T_\theta\mathcal{M}$ is given by

$$\operatorname{Hess}\Phi(\theta)[u] = \mathcal{P}_{T_\theta\mathcal{M}}\left(\nabla^2\Phi(\theta)[u]\right) - \mathcal{P}_{T_\theta\mathcal{M}}\mathrm{D}_u(\operatorname{grad}\Phi(\theta)).$$

Since $\mathbb{E}_{P_{y|x}(\theta)}\nabla \log p(y|f(x,\theta)) = \int_y \nabla p(y|f(x,\theta))\mathrm{d}y = \nabla \int_y p(y|f(x,\theta))\mathrm{d}y = 0$, we have $\operatorname{grad}\Phi(\theta) = 0$ and $\operatorname{Hess}\Phi(\theta) = F^R(\theta)$. Due to the uniqueness of the second-order Taylor expansion, the Riemannian Newton's direction at θ does not depend on the Riemannian metric and is equal to $d^R(\theta)$ in (3.2). Hence, it is reasonable to use the Euclidean metric to define the Riemannian natural gradient direction (3.2). For a quotient manifold $\mathcal{M}$ whose total space is an embedded submanifold and whose endowed Riemannian metric is horizontally invariant, it follows from [1, Equation (3.39)] that the Riemannian gradient of Φ in the total space is the horizontal lift of the corresponding Riemannian gradient in $\mathcal{M}$. Since the total space is an embedded manifold, we see from [2, Equation 7] and our earlier argument that $F^R(\theta)$ is the Riemannian Hessian of Φ in the total space at the representative element θ. Furthermore, by [1, Proposition 5.3.3], the horizontal lift of the corresponding Riemannian Hessian in $\mathcal{M}$ at the representative element θ equals the horizontal projection of $F^R(\theta)$. Since the Riemannian gradient of Φ in the total space at a representative element θ belongs to the horizontal space at θ, we conclude that $d^R(\theta)$ in (3.2), which lies in the horizontal space at θ, is the Riemannian Newton's direction at θ. As the Riemannian natural gradient direction is independent of the choice of the Riemannian metric, we can use the Euclidean metric to define (3.2), but a horizontally invariant Riemannian metric should be introduced to compare the norms of Riemannian gradients. In summary, the Riemannian natural descent direction (3.2) behaves as the Riemannian Newton's direction whenever $\mathcal{M}$ is an embedded submanifold or a quotient manifold whose total space is an embedded submanifold.

Similar to EFIM, we can define REFIM as

$$\bar{F}^R(\theta) := \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} \operatorname{grad}\log p(y|f(x,\theta))\operatorname{grad}\log p(y|f(x,\theta))^\top. \tag{3.6}$$

3.2 Algorithmic framework To fix ideas, let us first consider the case where $\mathcal{M}$ is an embedded submanifold. In the k-th iteration, once we obtain an estimate F_k of the RFIM (3.1) associated with Φ or the REFIM (3.6) associated with Ψ at θ^k, the Riemannian natural gradient direction in the tangent space to $\mathcal{M}$ at θ^k is computed by solving the following optimization problem:

$$d^k = \underset{d \in T_{\theta^k}\mathcal{M}}{\arg\min} \quad m_k(d) := \Psi_k + \left\langle g^k, d \right\rangle + \frac{1}{2}\left\langle (F_k + \lambda_k I)d, d \right\rangle, \tag{3.7}$$

where $\langle u, v\rangle := u^\top v$ for two vectors $u, v \in \mathbb{R}^r$, $F_k d$ is the usual matrix-vector multiplication, Ψ_k and g^k are stochastic estimates of $\Psi(\theta^k)$ and $\operatorname{grad}\Psi(\theta^k)$, respectively, and $\lambda_k > 0$ is usually updated adaptively by a trust region-like strategy. In view of the finite-sum structure of Ψ (see (1.1)), the stochastic estimates Ψ_k and g_k can be obtained using, e.g., a mini-batch strategy (i.e., randomly sample a subset of $\mathcal{S}$ and sum the corresponding terms in Ψ and $\operatorname{grad}\Psi$ to get Ψ_k and g_k, respectively). Since $F_k + \lambda_k I : T_{\theta^k}\mathcal{M} \to T_{\theta^k}\mathcal{M}$ is positive definite and $g^k \in T_{\theta^k}\mathcal{M}$, the solution of (3.7) is $d^k = -(F_k + \lambda_k I)^{-1} g^k$. If the inverse of $F_k + \lambda_k I$ is costly to compute, then the truncated conjugate gradient method can be utilized [41]. We will introduce the constructions of a few computationally efficient approximation F_k in Section 4.

Once d^k is obtained, we construct a trial point

$$z^k = R_{\theta^k}(d^k). \tag{3.8}$$

To measure whether z^k leads to a sufficient decrease in the objective value, we first calculate the ratio ρ_k between the reduction of Ψ and the reduction of m_k. Since the exact evaluation of Ψ is costly, one popular way [16] is to construct estimates Ψ_k^0 and $\Psi_k^{z^k}$ of $\Psi(\theta^k)$ and $\Psi(z^k)$, respectively. Then, we compute the ratio as

$$\rho_k = \frac{\Psi_k^{z^k} - \Psi_k^0}{m_k(d^k) - \Psi_k^0}. \tag{3.9}$$

Here, we take $\Psi_k = \Psi_k^0$ in the calculation of $m_k(d^k)$. Lastly, we perform the update

$$\theta^{k+1} = \begin{cases} z^k, & \text{if } \rho_k \geq \eta_1 \text{ and } \|g^k\| \geq \frac{\eta_2}{\sigma_k}, \\ \theta^k, & \text{otherwise}, \end{cases} \tag{3.10}$$

where $\eta_1 \in (0, 1)$ and $\eta_2 > 0$ are constants and $\sigma_k > 0$ is used to control the regularization parameter λ_k. Indeed, to ensure the descent property of the original function Ψ, some assumptions on the accuracy of the estimates of $\Psi(\theta^k)$, $\Psi(z^k)$ and the model m_k are needed, and they will be introduced later in the convergence analysis. Due to the error in the estimates, the regularization parameter λ_{k+1} should not only depend on the ratio ρ_k but also on the norm of the estimated Riemannian gradient g^k. In particular, we set $\lambda_{k+1} := \sigma_{k+1}\|g^{k+1}\|$ and update σ_{k+1} as

$$\sigma_{k+1} = \begin{cases} \max\left\{\sigma_{\min}, \frac{1}{\gamma}\sigma_k\right\}, & \text{if } \rho_k \geq \eta_1 \text{ and } \|g^k\| > \frac{\eta_2}{\sigma_k}, \\ \gamma\sigma_k, & \text{otherwise}, \end{cases} \tag{3.11}$$

where $\eta_1 \in (0, 1)$, $\eta_2 > 0$ are as before and $\sigma_{\min} > 0$, $\gamma > 1$ are parameters. Our proposed RNGD method is summarized in Algorithm 1. It is worth mentioning that a trust-region method is developed in [16] to solve stochastic optimization problems. Algorithm 1 can be seen as a combination of the stochastic update rule of the trust-region

Algorithm 1: Riemannian natural gradient descent (RNGD) for solving (1.1).

Choose an initial point θ^0 and parameters $\sigma_0 > 0$, $\sigma_{\min} > 0$, $\lambda_0 = \sigma_0\|g^0\|$, $\eta_1 \in (0,1)$, $\eta_2 > 0$, and $\gamma > 1$. Set $k = 0$.
while *stopping conditions not met* **do**
  Compute the estimated Riemannian gradient g^k and the estimated Riemannian Fisher information matrix F_k.
  Compute the negative natural gradient direction d^k by solving (3.7) and compute the trial point z^k by (3.8).
  Update θ^{k+1} based on (3.10).
  Update λ_{k+1} based on (3.11).
  $k \leftarrow k + 1$.

radius in [16] and the adaptive regularization technique for manifold optimization in [27]. Compared with the trust-region subproblem in [16, Equation (2)], the subproblem (3.7) can be efficiently solved if the cost of computing the inverse of $F_k + \lambda_k I$ is low. We remark that regularized subproblems similar to (3.7) have appeared in [39, 63, 62].

Now, for the case where $\mathcal{M}$ is a quotient manifold, we have a horizontally invariant Riemannian metric $\langle U, V\rangle_\Theta := \mathrm{vec}(U)^\top D(\theta)\mathrm{vec}(V)$. The Riemannian gradient in the k-th iteration is $\tilde{g}^k = D(\theta^k)^{-1}g^k$. Thus, in Algorithm 1, we can still use g_k and F_k in (3.7) but should replace $\|g^k\|$ in λ_k, (3.10), and (3.11) with $\|\tilde{g}_k\|_{\theta^k} := \sqrt{(\tilde{g}^k)^\top D(\theta^k)\tilde{g}^k}$.

4 Practical Riemannian natural gradient descent methods

From the definitions of RFIM and REFIM in Section 3, the computational cost of solving subproblem (3.7) may be high because of the vectorization of Θ. Fortunately, analogous to [39], the Riemannian natural gradient direction can be computed with a relatively low cost if the gradient of a single sample is of low rank, i.e., for a pair of observations $(x, y) \in \mathcal{S}$ and $\psi(\Theta; x, y) := -\log p(y|f(x,\Theta))$, $\nabla\psi$ takes the form

$$\nabla\psi(\Theta; x, y) = G(x, y)A(x, y)^\top, \tag{4.1}$$

where $G(x,y) \in \mathbb{R}^{m\times q}$ and $A(x,y) \in \mathbb{R}^{n\times q}$ with $q \ll \min(m,n)$. Let us now elaborate on this observation.

Recall that the Riemannian gradient of ψ is given by

$$\mathrm{grad}\,\psi(\Theta; x, y) = \mathcal{P}_{T_\Theta\mathcal{M}}(\nabla\psi(\Theta; x, y)).$$

When $\nabla\psi$ has the form (4.1), the linearity of the projection operator implies that

$$\begin{aligned} F^R(\theta) &= \mathbb{E}_{P_{x,y}(\theta)}\left[\mathrm{grad}\,\psi(\theta; x, y)\mathrm{grad}\,\psi(\theta; x, y)^\top\right] \\ &\approx \mathcal{P}\left(\mathbb{E}_{P_{x,y}(\theta)}\left[A(x,y)A(x,y)^\top\right] \otimes \mathbb{E}_{P_{x,y}(\theta)}\left[G(x,y)G(x,y)^\top\right]\right)\mathcal{P}, \end{aligned} \tag{4.2}$$

where $P_{x,y}(\theta)$ is the joint distribution of (x, y) given θ, $\mathcal{P} \in \mathbb{R}^{r\times r}$ is the matrix representation of $\mathcal{P}_{T_\Theta\mathcal{M}}$ (note that $\mathcal{P}^\top = \mathcal{P}$ due to the symmetry of orthogonal projection operators), and the approximation is due to the assumption that $A(x,y)$ and $G(x,y)$ are approximately independent; see also [23, Theorem 1] for a use of such an assumption to derive a simplified form of the FIM. By replacing $P_{x,y}(\theta)$ with its

empirical distribution observed from $\mathcal{S}$, an approximate REFIM is given by (4.3)

$$\bar{F}^R(\theta) \approx \mathcal{P}\left(\left[\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} A(x,y)A(x,y)^\top\right] \otimes \left[\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} G(x,y)G(x,y)^\top\right]\right)\mathcal{P}. \tag{4.3}$$

When a direct inverse of $\bar{F}^R(\theta)$ is expensive to compute, the truncated conjugate gradient method can be used. In preparation for the applications, we now show how to construct computationally efficient approximations of the RFIM and REFIM on the Grassmann manifold.

4.1 RFIM and REFIM on Grassmann manifold If the matrix representation $\mathcal{P}$ of the projection operator $\mathcal{P}_{T_\Theta\mathcal{M}}$ has dimensions m-by-m or n-by-n, i.e.,

$$\operatorname{grad}\psi(\Theta;x,y) = B_1G(x,y)A(x,y)^\top \text{ or } \operatorname{grad}\psi(\Theta;x,y) = G(x,y)A(x,y)^\top B_2$$

with $B_1 \in \mathbb{R}^{m\times m}$ and $B_2 \in \mathbb{R}^{n\times n}$, then we can approximate the RFIM in (4.2) by

$$F^R(\theta) \approx \mathbb{E}_{P_{x,y}(\theta)}\left[A(x,y)A(x,y)^\top\right] \otimes \mathbb{E}_{P_{x,y}(\theta)}\left[B_1G(x,y)G(x,y)^\top B_1\right]$$

or

$$F^R(\theta) \approx \mathbb{E}_{P_{x,y}(\theta)}\left[B_2A(x,y)A(x,y)^\top B_2\right] \otimes \mathbb{E}_{P_{x,y}(\theta)}\left[G(x,y)G(x,y)^\top\right].$$

Moreover, if we replace $P_{x,y}(\theta)$ by its empirical distribution observed from $\mathcal{S}$, then we can approximate the REFIM in (4.3) by

$$\bar{F}^R(\theta) \approx \left(\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} A(x,y)A(x,y)^\top\right) \otimes \left(\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} B_1G(x,y)G(x,y)^\top B_1\right)$$

or

$$\bar{F}^R(\theta) \approx \left(\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} B_2A(x,y)A(x,y)^\top B_2\right) \otimes \left(\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} G(x,y)G(x,y)^\top\right).$$

Note that the Kronecker product form allows the inverse of $\bar{F}^R(\theta)$ to be calculated efficiently by inverting two smaller matrices [39]. A typical manifold that yields the above Kronecker product representations is the Grassmann manifold $\mathrm{Gr}(m,n)$, which consists of all n (resp., m) dimensional subspaces in $\mathbb{R}^m$ (resp., $\mathbb{R}^n$) if $m \geq n$ (resp., $m < n$). The matrix representation of the projection operator at a point Θ with $\Theta^\top\Theta = I$ is $B_1 = I_m - \Theta\Theta^\top$ ($m \geq n$) or $B_2 = I_n - \Theta^\top\Theta$ ($m < n$). In what follows, we derive the RFIMs associated with three concrete applications involving the Grassmann manifold and explain how they can be computed efficiently.

4.2 Applications

4.2.1 Low-rank matrix completion For simplicity, we derive the RFIM associated with problem (1.3) for the fully observed case, i.e., $\Omega = \{1,\dots,n\} \times \{1,\dots,N\}$. One can derive the RFIM for the partly observed case in a similar fashion. By definition, we have $f(x,U) = Ua(U;x) - x$ and $\psi(U;x,y) = -\log p(y|f(x,U)) = \frac{1}{2}\|f(x,U) - y\|^2 + \frac{n\log(2\pi)}{2}$. It follows from [14, Subsection 3.4] that the Jacobian of a along a tangent vector $H \in T_U\mathrm{Gr}(n,p)$ is given by $J_a(U;x)[H] = H^\top x$ and its

adjoint $J_a^\top(U;x)$ satisfies $J_a^\top(U;x)[v] = x^\top v$ for $v \in \mathbb{R}^p$. The Riemannian gradient of $\psi(\cdot; x, y)$ is

$$\begin{aligned}\operatorname{grad}\psi(U;x,y) =&(I-UU^\top)((Ua(U;x)-x-y)a(U;x)^\top)\\ &+(I-UU^\top)x(Ua(U;x)-x-y)^\top U.\end{aligned}$$

By assuming that the residual $Ua(U;x)-x$ is close to zero, we have $(I-UU^\top)x \approx (I-UU^\top)Ua(U;x) = 0$. This leads to the following approximate Riemannian gradient of $\psi(\cdot; x, y)$:

$$\operatorname{grad}\psi(U;x,y) \approx (I-UU^\top)((Ua(U;x)-x-y)a(U;x)^\top). \tag{4.4}$$

Plugging the above approximation into (4.2) leads to

$$\begin{aligned}F^R(u) =&\mathbb{E}_{P_x}\left[\mathbb{E}_{P_{y|x}(u)}\left[\operatorname{grad}\psi(u;x,y)\operatorname{grad}\psi(u;x,y)^\top\right]\right]\\ \approx&\mathbb{E}_{P_x}\Big[\mathbb{E}_{P_{y|x}(U)}\left[[a(U;x)a(U;x)^\top]\otimes\left[(I-UU^\top)(Ua(U;x)-x-y)\right.\right.\\ &\left.\left.(Ua(U;x)-x-y)^\top(I-UU^\top)\right]\right]\Big]\\ \approx&\left[\frac{1}{N}\sum_{i=1}^N a(U;x_i)a(U;x_i)^\top\right]\otimes(I-UU^\top),\end{aligned}$$

where $u = \operatorname{vec}(U)$ is the vectorization of U, the second line is due to (4.4), $\operatorname{vec}(uv^\top) = v\otimes u$, $(A\otimes B)^\top = A^\top\otimes B^\top$, and $(A\otimes B)(A^\top\otimes B^\top) = (AA^\top)\otimes(BB^\top)$, and the last line follows from $\mathbb{E}_{P_{y|x}(U)}\left[(Ua(U;x)-x-y)(Ua(U;x)-x-y)^\top\right] = I$ and by substituting P_x with its empirical distribution. For $H \in T_U\mathrm{Gr}(n,p)$, we have

$$\begin{aligned}\operatorname{mat}(F^R(u)[\operatorname{vec}(H)]) &\approx \left[\frac{1}{N}\sum_{i=1}^N a(U;x_i)a(U;x_i)^\top\right]\otimes(I-UU^\top)\operatorname{vec}(H)\\ &= H\left[\frac{1}{N}\sum_{i=1}^N a(U;x_i)a(U;x_i)^\top\right],\end{aligned} \tag{4.5}$$

where $\operatorname{mat}(b)$ converts the vector $b \in \mathbb{R}^{np}$ into an n-by-p matrix and the equality follows from $(I-UU^\top)H = H$. For the partly observed case, the matrix $F^R(u)$ defined in the above equation can serve as a good approximation of the exact RFIM. Note that $\frac{1}{N}\sum_{i=1}^N a(U;x_i)a(U;x_i)^\top \in \mathbb{R}^{p\times p}$ is of low dimension since the rank p is usually small. Thus, the Riemannian natural gradient direction can be calculated with a relatively low cost.

4.2.2 Low-dimension subspace learning In multi-task learning [6, 40], different tasks are assumed to share the same latent low-dimensional feature representation. Specifically, suppose that the i-th task has the training set $X_i \in \mathbb{R}^{d_i\times n}$ and the corresponding label set $y_i \in \mathbb{R}^{d_i}$ for $i = 1,\dots,N$. The multi-task feature learning problem can then be formulated as

$$\min_{U\in\mathrm{Gr}(n,p)} \Psi(U) = \frac{1}{2N}\sum_{i=1}^N \|X_iUw(U;X_i,y_i) - y_i\|^2, \tag{4.6}$$

where $w(U;X_i,y_i) = \arg\min_w \frac{1}{2}\|X_iUw - y_i\|^2 + \lambda\|w\|^2$ and $\lambda > 0$ is a regularization parameter. Suppose that $d_1 = \cdots = d_N = d$. Then, problem (4.6) has the form (1.1),

where $\mathcal{S} = \{((X_i, y_i), 0)\}_{i=1}^N$, $\mathcal{X} = \mathbb{R}^{d\times(n+1)}$, $\mathcal{Y} = \mathbb{R}^d$, $f(X, y, U) = XUw(U; X, y) - y$, and $p(z|f(X, y, U)) = \frac{1}{\sqrt{(2\pi)^d}} \exp(-\frac{1}{2}(z - f(X, y, U))^\top (z - f(X, y, U)))$. By ignoring the constant $\frac{1}{\sqrt{(2\pi)^d}}$ when computing ψ, we denote $\psi(U; X, y, z) = \frac{1}{2}\|XUw(U; X, y) - y - z\|^2$. Using the optimality of $w(U; X, y)$, we have $U^\top X^\top (XU$ $w(U; X, y) - y) + \lambda w(U; X, y) = 0$. Then, we can compute the Euclidean gradient of $\psi(\cdot; X, y, z)$ as

$$\begin{aligned}
&\nabla\psi(U; X, y, z)\\
=&X^\top (XUw(U; X, y) - y - z)w(U; X, y)^\top + J_w^\top(U)\left[U^\top X^\top (XUw(U; X, y) - y - z)\right]\\
\approx&X^\top (XUw(U; X, y) - y)w(U; X, y)^\top,
\end{aligned}$$

where $J_w(U)$ is the Jacobian of $w(U; X, y)$, $J_w^\top(U)$ denotes the adjoint of $J_w(U)$, and the approximation holds for small λ and $\|z\|$. Note that z will lie in a small neighborhood of zero with high probability if $f(X, y, U)$ is close to 0. Besides, z is always zero in the dataset $\mathcal{S}$. With the above, an approximate Riemannian gradient of $\psi(\cdot; X, y, z)$ is given by

$$\operatorname{grad}\psi(U; X, y, z) \approx (I - UU^\top)X^\top (XUw(U; X, y) - y - z)w(U; X, y)^\top. \tag{4.7}$$

Consequently, we have

$$\begin{aligned}
F^R(u) =& \mathbb{E}_{P_{(X,y)}}\left[\mathbb{E}_{P_{z|(X,y)}(u)}[\operatorname{grad}\psi(u; X, y, z)\operatorname{grad}\psi(u; X, y, z)^\top]\right]\\
\approx&\frac{1}{N}\sum_{i=1}^N (w_i \otimes ((I - UU^\top)X_i^\top))(w_i \otimes ((I - UU^\top)X_i^\top))^\top\\
=&\frac{1}{N}\sum_{i=1}^N \left[(w_i w_i^\top) \otimes ((I - UU^\top)X_i^\top X_i(I - UU^\top))\right]\\
\approx&\frac{1}{N}\left[\sum_{i=1}^N w_i w_i^\top\right] \otimes \left[\frac{1}{N}\sum_{i=1}^N (I - UU^\top)X_i^\top X_i(I - UU^\top)\right],
\end{aligned} \tag{4.8}$$

where $u = \operatorname{vec}(U)$ is the vectorization of U, $w_i := w(U; X_i, y_i)$, the second line follows from (4.7), $\mathbb{E}_{P_{z|(X,y)}(u)}[(XUw(U; X, y) - y - z)(XUw(U; X, y) - y - z)^\top] = I$, and the empirical approximation of $P_{(X,y)}$, and the last line holds under the same condition as in (4.2). Though the construction of $F^R(u)$ is for the case $d_1 = \cdots = d_N$, it can be easily extended to the case where the d_i's are not equal.

4.2.3 Fully connected network with batch normalization Consider an L-layer neural network with input $a_0 = x$. In the l-th layer, we have

$$s_l = W_l a_{l-1} + b_l,\ t_{l,i} = \frac{s_{l,i} - \mathbb{E}(s_{l,i})}{\operatorname{Var}(s_{l,i})} \times \gamma_{l,i} + \beta_{l,i},\ i = 1, \ldots, n_l,\ a_l = \varphi_l(t_l), \tag{4.9}$$

where φ_l is an element-wise activation function, $W_l \in \mathbb{R}^{n_l\times n_{l-1}}$ is the weight, $b_l \in \mathbb{R}^{n_l}$ is the bias, $s_{l,i}$ is the i-th component of $s_l \in \mathbb{R}^{n_l}$, $\gamma_{l,i}, \beta_{l,i} \in \mathbb{R}$ are two learnable parameters, $\operatorname{Var}(s_{l,i})$ is the variance of $s_{l,i}$, and $f(x, \Theta) = a_L \in \mathbb{R}^m$ is the output of the network with Θ being the collection of parameters $\{W_l, b_l, \gamma_l, \beta_l\}$. By default, the elements of $\gamma_{l,i}$ are set to 1 and the elements of $\beta_{l,i}$ are set to 0. In [28], $t_{l,i}$ is called the batch normalization of $s_{l,i}$.

Given a dataset $\mathcal{S}$, our goal is to minimize the discrepancy between the network output $f(x,\Theta)$ and the observed output y, namely,

$$\min_{\Theta} \ \Psi(\Theta) = -\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}} \log p(y|f(x,\Theta)). \tag{4.10}$$

By [17], each row of W_l lies on the Grassmann manifold $\mathrm{Gr}(1,n_{l-1})$. It follows that W_l lies on the product of Grassmann manifolds, i.e., $W_l \in \mathrm{Gr}(1,n_{l-1})\times\cdots\times\mathrm{Gr}(1,n_{l-1}) \in \mathbb{R}^{n_l\times n_{l-1}}$. The remaining parameters lie in the Euclidean space. Rather than batch normalization, layer normalization [8] and weight normalization [49] have also been widely investigated in the study of deep neural networks, where $\mathrm{vec}(W_l) \in \mathrm{Gr}(n_l\times n_{l-1},1)$ and $W_l \in \mathrm{Sp}(n_{l-1}-1)\times\cdots\times\mathrm{Sp}(n_{l-1}-1) \in \mathbb{R}^{n_l\times n_{l-1}}$ with $\mathrm{Sp}(n_{l-1}-1) := \{u\in\mathbb{R}^{n_{l-1}} : \|u\|=1\}$, respectively.

By back-propagation, the Euclidean gradient of Ψ with respect to W_l is given by

$$g_l \leftarrow \mathrm{D}a_l \odot \varphi_l'(t_l) \odot \mathrm{D}t_l, \quad \nabla\Psi(W_l) \leftarrow g_l a_{l-1}^\top, \quad \mathrm{D}a_{l-1} \leftarrow W_l^\top g_l.$$

In particular, we see that $\nabla\Psi(W_l)$ has the Kronecker product form (4.1). Moreover, note that $\Psi(w_{l,i}) = \Psi(cw_{l,i})$, $\forall c\neq 0$. Now, we compute

$$\nabla\Psi(w_{l,i})w_{l,i}^\top = \lim_{t\to 0}\frac{\Psi(w_{l,i}+tw_{l,i})-\Psi(w_{l,i})}{t} = 0.$$

By definition of the projection operator defined on the product of Grassmann manifolds, the Riemannian gradient $\mathrm{grad}\,\Psi(W_l)$ is actually the same as the Euclidean gradient $\nabla\Psi(W_l)$. Specifically, for the i-th row of $\mathrm{grad}\,\Psi(W_l)$, we have

$$[\mathrm{grad}\,\Psi(W_l)]_i = \mathrm{grad}\,\Psi(w_{l,i}) = \nabla\Psi(w_{l,i}) - \nabla\Psi(w_{l,i})w_{l,i}^\top w_{l,i} = \nabla\Psi(w_{l,i}).$$

Therefore, the RFIM coincides with the FIM. The inverse of $F^R(\theta)$ can be computed easily when the FIM has a Kronecker product form.

5 Convergence Analysis

In this section, we study the convergence behavior of the RNGD method (Algorithm 1).

5.1 Global convergence to a stationary point

To begin, let us consider the case where $\mathcal{M}$ is an embedded submanifold and extend some of the definitions used in the study of Euclidean stochastic trust-region methods (see, e.g., [16]) to this setting.

DEFINITION 5.1. *Let $\kappa_{\mathrm{ef}},\kappa_{\mathrm{eg}}>0$ be given constants. A function m_k is called a $(\kappa_{\mathrm{ef}},\kappa_{\mathrm{eg}})$-fully linear model of Ψ on $B_{\theta^k}(0,1/\sigma_k)$ if for any $y\in B_{\theta^k}(0,1/\sigma_k)$,*

$$\|\nabla(\Psi\circ R_{\theta^k})(y) - \nabla m_k(y)\| \le \frac{\kappa_{\mathrm{eg}}}{\sigma_k} \quad \textit{and} \quad |\Psi\circ R_{\theta^k}(y) - m_k(y)| \le \frac{\kappa_{\mathrm{ef}}}{\sigma_k^2}, \tag{5.1}$$

where $B_\theta(0,\rho) := \{d\in T_\theta\mathcal{M} : \|d\|\le\rho\}$.

DEFINITION 5.2. *Let $\epsilon_F,\sigma_k>0$ be given constants. The quantities Ψ_k^0 and $\Psi_k^{z^k}$ are called ϵ_F-accurate estimates of $\Psi\left(\theta^k\right)$ and $\Psi_k\left(z^k\right)$, respectively if*

$$\left|\Psi_k^0 - \Psi\left(\theta^k\right)\right| \le \frac{\epsilon_F}{\sigma_k^2} \quad \textit{and} \quad \left|\Psi_k^{z^k} - \Psi_k\left(z^k\right)\right| \le \frac{\epsilon_F}{\sigma_k^2}, \tag{5.2}$$

where z^k is defined in (3.8).

Analogous to [16, 58], the inequalities (5.1) and (5.2) can be guaranteed when $\mathcal{M}$ is compact, the number of samples is large enough, and $\nabla(\Psi \circ R)$ is Lipschitz continuous.

Next, we introduce the assumptions needed for our convergence analysis. Their Euclidean counterparts can be found in, e.g., [16, Assumptions 4.1 and 4.3].

ASSUMPTION 5.3. *Let $\theta^0 \in \mathbb{R}^r, \sigma_{\min} > 0$ be given. Let $\mathcal{L}(\theta^0)$ denote the set of iterates generated by Algorithm 1. Then, the function Ψ is bounded from below on $\mathcal{L}(\theta^0)$. Moreover, the function $\Psi \circ R$ and its gradient $\nabla(\Psi \circ R)$ are L-Lipschitz continuous on the set*

$$\mathcal{L}_{\text{enl}}(\theta^0) = \bigcup_{\theta \in \mathcal{L}(\theta^0)} B_\theta\left(0, \frac{1}{\sigma_{\min}}\right).$$

ASSUMPTION 5.4. *The RFIM or REFIM F_k satisfies $\|F_k\|_{\text{op}} \leq \kappa_{\text{fim}}$ for all $k \geq 0$, where $\|\cdot\|_{\text{op}}$ is the operator norm.*

We remark that Assumptions 5.3 and 5.4 hold for any compact $\mathcal{M}$ and smooth Ψ. With the above assumptions, we can prove the convergence of Algorithm 1 by adapting the arguments in [16]. The main difference is that our analysis makes use of the pull-back function $\Psi \circ R$ and its Euclidean gradient; see Definitions 5.1 and 5.2.

THEOREM 5.5. *Suppose that Assumptions 5.3 and 5.4 hold, m_k is a $(\kappa_{\text{ef}}, \kappa_{\text{eg}})$-fully linear model for some $\kappa_{\text{ef}}, \kappa_{\text{eg}} > 0$, and the estimates Ψ_k^0 and $\Psi_k^{z^k}$ are ϵ_F-accurate for some $\epsilon_F > 0$. Furthermore, suppose that $\eta_2 \geq \max\left\{\kappa_{\text{fim}}, \frac{16\kappa_{\text{ef}}}{1-\eta_1}\right\}$ and $\epsilon_F \leq \min\left\{\kappa_{\text{ef}}, \frac{1}{32}\eta_1\eta_2\right\}$. Then, the sequence of iterates $\{\theta^k\}$ generated by Algorithm 1 will almost surely satisfy*

$$\liminf_{k\to\infty} \left\|\operatorname{grad}\Psi(\theta^k)\right\| = 0.$$

Proof. One can prove the conclusion by following the arguments in [16, Theorem 4.16]. We here present a sketch of the proof. Define $\mathcal{F}_k$ as the σ-algebra generated by $\Psi_1^0, \Psi_1^{z^1}, \ldots, \Psi_k^0, \Psi_k^{z^k}$ and $m_1, \ldots, m_k$. Consider the random function $\Phi_k = v\Psi(\theta^k) + (1-v)/\sigma_k^2$, where $v \in (0,1)$ is fixed. The idea is to prove that there exists a constant $\tau > 0$ such that for all k,

$$\mathbb{E}\left[\Phi_{k+1} - \Phi_k \mid \mathcal{F}_{k-1}\right] \leq -\frac{\tau}{\sigma_k^2} < 0. \tag{5.3}$$

Summing (5.3) over $k \geq 1$ and taking expectations on both sides lead to $\sum_{k=1}^{\infty} 1/\sigma_k^2 < \infty$. The inequality (5.3) can be proved in the following steps. Firstly, a decrease on Ψ of order $-\mathcal{O}(1/\sigma_k^2)$ can be proved using the fully linear model approximation and the positive definiteness of $F_k + \sigma_k\|g^k\|I$ with a sufficiently large σ_k. Secondly, the trial point z^k is accepted provided that the estimates Ψ_k^0 and $\Psi_k^{z^k}$ are ϵ_F-accurate with sufficiently small ϵ_F and large σ_k. In addition, with $\eta_2 \geq \max\left\{\kappa_{\text{fim}}, \frac{16\kappa_{\text{ef}}}{1-\eta_1}\right\}$, if z^k is accepted (i.e., $\theta^{k+1} = z^k$), then a decrease of $-\mathcal{O}(1/\sigma_k^2)$ on Ψ can always be guaranteed when $\epsilon_F \leq \min\left\{\kappa_{\text{ef}}, \frac{1}{32}\eta_1\eta_2\right\}$ based on the update scheme (3.11). On the other hand, if z^k is rejected (i.e., $\theta^{k+1} = \theta^k$), then $\mathbb{E}\left[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}\right] = (1-v)(1/\gamma^2 - 1)/\sigma_k^2$. By choosing v to be sufficiently close to 1, the inequality (5.3) holds for any k.

Now, we will have $\sigma_k \to \infty$ as $k \to \infty$ with probability 1. If there exist $\epsilon > 0$ and $k_0 \geq 1$ such that $\|\operatorname{grad}\Psi(\theta^k)\| \geq \epsilon$ for all $k \geq k_0$, then the trial point will be accepted eventually because the estimates Ψ_k^0 and $\Psi_k^{z^k}$ are ϵ_F-accurate. Recall that

σ_k is decreasing in the case of accepting z^k. This means that σ_k is bounded above, which leads to a contradiction. Hence, we conclude that $\liminf_{k\to\infty} \|\mathrm{grad}\, \Psi(\theta^k)\| = 0$ will hold almost surely. ∎

REMARK 5.6. *Analogous to [16, Theorem 4.18], one can show that* $\lim_{k\to\infty} \|\mathrm{grad}\, \Psi(\theta^k)\| = 0$ *will hold almost surely by assuming the Lipschitz continuity of* $\mathrm{grad}\, \Psi$.

REMARK 5.7. *For the case where* $\mathcal{M}$ *is a quotient manifold, we modify Algorithm 1 according to the approach mentioned in the last paragraph of Section 3.2. The iterate* θ^k *and the tangent space at* θ^k *should be understood as a representative element and the horizontal space at* θ^k*, respectively. Due to the horizontal invariance of the Riemannian metric, the almost sure convergence result of* $\liminf_{k\to\infty} \|D(\theta^k)^{-1}\mathrm{grad}\, \Psi(\theta^k)\|_{\theta^k} \to 0$ *also holds.*

5.2 Convergence rate analysis of RNGD In this subsection, we study the local convergence rate of a deterministic version of the RNGD method. Let us start with some definitions. Let

$$L(z, y) := -\log p(y|z)$$

and suppose that P_x is the empirical distribution defined by $\mathcal{S}$. We define $\mathcal{S}_x := \{x : (x, y) \in \mathcal{S}\}$, $\mathcal{S}_y := \{y : (x, y) \in \mathcal{S}\}$, $F_L(x, \theta) := \mathbb{E}_{P_{y|x}(\theta)}[\nabla_z \log p(y|z) \nabla_z \log p(y|z)^\top]|_{z=f(x,\theta)}$, and write $J^R(x, \theta) := [\mathrm{grad}\, f_1(x, \theta), \ldots, \mathrm{grad}\, f_q(x, \theta)]^\top$ for the Riemannian Jacobian of $f(x, \theta) = [f_1(x, \theta), \ldots, f_q(x, \theta)]^\top$ with respect to θ. Furthermore, we write $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N$ with $N = |\mathcal{S}|$ and $u(\theta) = [f(x_1, \theta), \ldots, f(x_N, \theta)]^\top$. Let $J^R(\theta) := [J^R(x_1, \theta), \ldots, J^R(x_N, \theta)]$ and $H_L(u(\theta)) := \mathrm{blkdiag}(H_L(u(\theta)_1), \ldots, H_L(u(\theta)_N))$. For simplicity, we write $u^k := u(\theta^k)$.

5.2.1 Convergence rate Throughout this subsection, we make the following assumptions on the loss function L.

ASSUMPTION 5.8. *For any* $y \in \mathcal{S}_y$*, the loss function* $L(\cdot, y)$ *is smooth and* μ*-strongly convex and has* κ_L*-Lipschitz gradient and* κ_H*-Lipschitz Hessian, namely,*

$$\mu I \preceq \nabla^2_{zz} L(z, y) \preceq \kappa_L I, \quad \|\nabla^2_{zz} L(z, y) - \nabla^2_{zz} L(x, y)\| \le \kappa_H \|z - x\|, \quad \forall z, x \in \mathbb{R}^n.$$

In addition, the following condition holds:

$$F_L(x, \theta) = \nabla^2_{zz} L(z, y)|_{z=f(x,\theta)} := H_L(f(x, \theta)). \tag{5.4}$$

We remark that the equality (5.4) holds if $\nabla^2_{zz} L(z, y)|_{z=f(x,\theta)}$ does not depend on y, which is the case for the square loss $L(z, y) = \|z - y\|^2$ and the cross-entropy loss $L(y, z) = -\sum_j y_j \log z_j + \log(\sum_j \exp(z_j))$. We refer the reader to [38, Section 9.2] for other loss functions that satisfy (5.4). We remark that the square loss $L(z, y) = \|z - y\|^2$, which appears in both the LRMC and low-dimension subspace learning problems, satisfies Assumption 5.8.

According to the definition of RFIM in (3.1) and the chain rule, we obtain

$$F^R(\theta) = \frac{1}{|\mathcal{S}_x|} \sum_{x \in \mathcal{S}_x} J^R(x, \theta)^\top F_L(x, \theta) J^R(x, \theta).$$

Based on Assumption 5.8, we have $F^R(\theta) = J^R(\theta)^\top H_L(u(\theta)) J^R(\theta)$. Note that $F^R(\theta)$ may be singular when $J^R(\theta)$ is not of full column rank. In this case, provided that $\left(J^R(\theta^k) J^R(\theta^k)^\top\right)^{-1}$ exists, we can use the pseudo-inverse

$$F^R(\theta^k)^\dagger = J^R(\theta^k)^\top (J^R(\theta^k) J^R(\theta^k)^\top)^{-1} H_L(u^k)^{-1} (J^R(\theta^k) J^R(\theta^k)^\top)^{-1} J^R(\theta^k)$$

for computation. As mentioned at the beginning of this subsection, we focus on a deterministic version of the RNGD method, in which we adopt a fixed step size $t > 0$ and perform the update

$$d^k = (F^R(\theta^k))^\dagger J^R(\theta^k)^\top \nabla L(u^k, y), \quad \theta^{k+1} = R_{\theta^k}(-td^k)). \tag{5.5}$$

For concreteness, let us take R to be the exponential map for $\mathcal{M}$ in our subsequent development. Our convergence rate analysis of this deterministic RNGD method can be divided into two steps. The first step is to prove that the iterates $\{\theta^k\}$ always stay in a neighborhood of θ^0 if J^R satisfies certain stability condition. The second step is to establish the convergence rate of the method by utilizing the strong convexity of L. We remark that the zero acceleration property of the exponential map [1, Equation (5.24)] is essential to our analysis. As such, we can only handle the case where the retraction is the exponential map. The analysis for the case of a more general retraction is left as an open problem. Motivated by [66], we now formulate the aforementioned stability condition on J^R.

ASSUMPTION 5.9. *For any θ satisfying $\|\theta - \theta^0\| \le 4\kappa_L(\mu\sigma_0)^{-1}\|u^0 - y\|$, where $\sigma_0 := \sqrt{\lambda_{\min}(J^R(\theta^0)J^R(\theta^0)^\top)} > 0$, it holds that*

$$\|J^R(\theta) - J^R(\theta^0)\| \le \min\left\{\frac{1}{2}, \frac{\mu}{6\kappa_L}\right\}\sigma_0. \tag{5.6}$$

As will be seen in Section 5.2.2, Assumption 5.9 is satisfied by the Riemannian Jacobian that arises in a two-layer fully connected neural network with batch normalization and sufficiently large width. We are now ready to prove the following theorem.

THEOREM 5.10. *Let R be the exponential map for $\mathcal{M}$. Suppose that Assumptions 5.8 and 5.9 hold. Let $\{\theta^k\}$ be the iterates generated by (5.5).*

(a) *There exists a constant $\kappa_R > 0$ such that if $\|u^0 - y\| < \frac{\mu}{3\kappa_H}$ and $t \le \min\left\{1, \left(\frac{1}{6\|u^0-y\|} - \frac{\kappa_H}{2\mu}\right)\cdot\frac{3\mu^2\sigma_0}{8\kappa_R\kappa_L^2}\right\}$, then*

$$\|u^{k+1} - y\| \le \left(1 - \frac{t}{2}\right)\|u^k - y\|. \tag{5.7}$$

(b) *Suppose further that J^R is κ_J-Lipschitz continuous with respect to θ, i.e.,*

$$\|J^R(\theta) - J^R(\nu)\| \le \kappa_J\|\theta - \nu\|, \quad \forall \theta, \nu \in \mathbb{R}^r. \tag{5.8}$$

The rate of convergence is quadratic when $t = 1$, namely, there is a constant $\kappa_q > 0$ such that

$$\|u^{k+1} - y\| \le \kappa_q\|u^k - y\|^2. \tag{5.9}$$

Proof. (a). We proceed by induction. Assume that for $j \le k$, we have

$$\|\theta^j - \theta^0\| \le 4\kappa_L(\mu\sigma_0)^{-1}\|u^0 - y\|, \ \ \|u^j - y\| \le \left(1 - \frac{\eta}{2}\right)\|u^{j-1} - y\|.$$

By the definition of d^k in (5.5),

$$\begin{aligned}\|d^k\| &\le \|J^R(\theta^k)^\top(J^R(\theta^k)J^R(\theta^k)^\top)^{-1}\|\|H_L(\theta^k)^{-1}\|\|\nabla_u L(u^k, y) - \nabla_u L(y, y)\| \\ &\le \mu^{-1}\kappa_L\sigma_{\min}^{-1}(J^R(\theta^k))\|u^k - y\| \\ &\le 2\kappa_L(\mu\sigma_0)^{-1}\|u^k - y\|,\end{aligned} \tag{5.10}$$

where the first inequality is due to $\nabla L(y,y) = 0$ and the last inequality is from Assumption 5.9. Now, define the map $c_k : [0,1] \to \mathcal{M}$ as $c_k(s) = R_{\theta^k}(-std^k)$. Note that for the exponential map R, the geodesic distance between θ and $R_\theta(\xi)$ is equal to $\|\xi\|$ [1, Equation (7.25)], and inequality (2.2) holds with $\alpha = 1$ when we take the Euclidean metric as the Riemannian metric on $\mathcal{M}$. Thus, for any $s \in [0,1]$,

$$\begin{aligned}\|c_k(s) - \theta^0\| &\le \|c_k(s) - \theta^k\| + \sum_{j=0}^{k-1} \|\theta^{j+1} - \theta^j\| \le t \sum_{j=0}^{k} \|d^j\| \\ &\le 2\kappa_L(\mu\sigma_0)^{-1} t \sum_{j=0}^{k} \|u^j - y\|.\end{aligned}$$

Since $\|u^j - y\| \le (1 - \frac{\eta}{2})\|u^{j-1} - y\|$ for all $j \le k$, we have $\|c_k(s) - \theta^0\| \le 4\kappa_L(\mu\sigma_0)^{-1}\|u^0 - y\|$ for all $s \in (0,1]$. This gives $\|\theta^{k+1} - \theta^0\| \le 4\mu\kappa_L\sigma_0^{-1}\|u^0 - y\|$. To prove (5.7), we split $\|u^{k+1} - y\|$ into three terms, namely,

$$\begin{aligned}u^{k+1} - y =& u^{k+1} - u^k + u^k - y = \int_0^1 J^R(c_k(s)) c_k'(s) \mathrm{d}s + u^k - y \\ =& \underbrace{\int_0^1 J^R(c_k(s))(c_k'(s) - td^k)\mathrm{d}s}_{b_1} + \underbrace{t\int_0^1 (J^R(c_k(s)) - J^R(\theta^k))d^k \mathrm{d}s}_{b_2} \\ &+ \underbrace{t\int_0^1 J^R(\theta^k)d^k \mathrm{d}s + u^k - y}_{b_3}.\end{aligned} \tag{5.11}$$

For the exponential map R [1, Equation (5.24)], it holds that

$$c_k'(s) - td^k = c_k''(s)[-std^k] + \tilde{\kappa}_R s^2 t^2 \|d^k\|^2, \tag{5.12}$$

where $c_k''(s)[-std^k]$ belongs to the normal space to $\mathcal{M}$ at $c_k(s)$ and $\tilde{\kappa}_R > 0$ is the smoothness constant. Plugging (5.12) into (5.11), we have

$$\begin{aligned}\|b_1\| &\le \int_0^1 (\|J^R(\theta^0)\| + \|J^R(c_k(s)) - J^R(\theta^0)\|)\tilde{\kappa}_R s^2 t^2 \|d^k\|^2 \mathrm{d}s \\ &\le \int_0^1 2\sigma_0\kappa_R s^2 t^2 \|d^k\|^2 \mathrm{d}s = \frac{2}{3}\sigma_0\kappa_R t^2 \|d^k\|^2,\end{aligned}$$

where $\kappa_R := \tilde{\kappa}_R \cdot (1/4 + \|J^R(\theta^0)\|/(2\sigma_0))$. By (5.6) and (5.10), we have

$$\|b_2\| \le t\int_0^1 \min\left\{\frac{1}{2}, \frac{\mu}{6\kappa_L}\right\} \sigma_0 \cdot 2\kappa_L(\mu\sigma_0)^{-1}\|u^k - y\|\mathrm{d}s \le \frac{t}{3}\|u^k - y\|.$$

Now, the update (5.5) yields $J^R\left(u^k\right) d^k = H_L\left(u^k\right)^{-1} \nabla L\left(u^k, y\right)$. It follows that

$$\begin{aligned}
\|b_3\| &= \|u^k - y - tH_L\left(u^k\right)^{-1}\left(\nabla L\left(u^k, y\right) - \nabla L(y, y)\right)\| \\
&= \|H_L\left(u^k\right)^{-1}\left(H_L\left(u^k\right)\left(u^k - y\right) - t\left(\nabla L\left(u^k, y\right) - \nabla L(y, y)\right)\right)\| \\
&= \left\|H_L\left(u^k\right)^{-1}\left(H_L\left(u^k\right)\left(u^k - y\right) - t\int_0^1 H_L\left(u^k + s\left(y - u^k\right)\right)\left(u^k - y\right) \mathrm{d}s\right)\right\| \\
&= \left\|H_L\left(u^k\right)^{-1}\left[\int_0^1 \left(H_L\left(u^k\right) - tH_L\left(u^k + s\left(y - u^k\right)\right)\right) \mathrm{d}s\right]\left(u^k - y\right)\right\| \\
&\leq \int_0^1 \left(1 - t + t\mu^{-1}\kappa_H s\left\|u^k - y\right\|\right) \mathrm{d}s \cdot \left\|u^k - y\right\| \\
&= \left(1 - t + \frac{\kappa_H t}{2\mu}\left\|u^k - y\right\|\right)\left\|u^k - y\right\|,
\end{aligned}$$

where the first inequality is due to Assumption 5.8. Combining the estimates on b_1, b_2, b_3, we conclude that

(5.13)
$$\begin{aligned}
\|u^{k+1} - y\| &\leq \left(1 - \frac{2t}{3} + \frac{\kappa_H t}{2\mu}\|u^k - y\|\right)\|u^k - y\| + \frac{8}{3}\mu^{-2}\kappa_R\kappa_L^2\sigma_0^{-1}t^2\|u^k - y\|^2 \\
&\leq \left(1 - \frac{t}{2}\right)\|u^k - y\|
\end{aligned}$$

whenever $\|u^k - y\| < \frac{\mu}{3\kappa_H}$ and $t \leq \left(\frac{1}{6\|u^k - y\|} - \frac{\kappa_H}{2\mu}\right) \cdot \frac{3\mu^2\sigma_0}{8\kappa_R\kappa_L^2}$. Therefore, the inequality (5.7) holds by using the inductive hypothesis $\|u^k - y\| \leq \|u^0 - y\|$.

(b). The proof is similar to that for (a). Substituting $t = 1$ into (5.11), we obtain

$$\begin{aligned}
\|u^{k+1} - y\| &\leq \frac{\kappa_H}{2\mu}\|u^k - y\|^2 + \frac{1}{2}\kappa_J\|d^k\|^2 + \frac{8}{3}\mu^{-2}\kappa_R\kappa_L^2\sigma_0^{-1}\|u^k - y\|^2 \\
&\leq \left[\frac{\kappa_H}{2\mu} + 2\kappa_L^2(\mu\sigma_0)^{-2}\left(\kappa_J + \frac{4}{3}\sigma_0\kappa_R\right)\right]\|u^k - y\|^2,
\end{aligned}$$

where we use (5.8) to get

$$\begin{aligned}
&\left\|\int_0^1 (J^R(c_k(s)) - J^R(\theta^k))d^k \mathrm{d}s\right\| \\
\leq &\kappa_J \int_0^1 \|c_k(s) - \theta^k\|\|d^k\| \mathrm{d}s \leq \frac{1}{2}\kappa_J\|d^k\|^2 \leq 2\kappa_J\kappa_L^2(\mu\sigma_0)^{-2}\|u^k - y\|^2.
\end{aligned}$$

The verification of the neighborhood condition for θ^k is similar to that in (a). This completes the proof. ∎

5.2.2 Jacobian stability of two-layer neural network with batch normalization

Problem setting From the previous subsection, we see that the Jacobian stability condition in Assumption 5.9 plays an important role in the convergence rate analysis of the RNGD method. Let us now show that such a condition is satisfied by a two-layer neural network with batch normalization, thereby demonstrating its relevance. The difference between our setting and that of [66] lies in the use of batch normalization.

To begin, consider the input-output map f given by

$$f(x,\theta,a) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} a_j \phi\left(\frac{\theta_j^\top (x - \mathbb{E}[x])}{\sqrt{\theta_j^\top V \theta_j}}\right), \tag{5.14}$$

where $x \in \mathbb{R}^n$ is the (random) input vector, $V = \mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top]$ is the covariance matrix, $\theta = [\theta_1^\top, \theta_2^\top, \ldots, \theta_m^\top]^\top \in \mathbb{R}^{mn}$ is the weight vector of the first layer, $a_j \in \mathbb{R}$ is the output weight of hidden unit j, and ϕ is the ReLU activation function. This represents a single-output two-layer neural network with batch normalization. We fix the a_j's throughout as in [66] and apply the RNGD method with a fixed step size on θ, in which each weight vector θ_j is assumed to be normalized. For the Grassmann manifold $\mathrm{Gr}(1,n)$, we choose d with $\|d\| = 1$ as the representative element of the one-dimensional subspace $\{cd : c \neq 0\}$. With a slight abuse of notation, we write $\mathrm{Gr}(1,n) := \{d \in \mathbb{R}^n : \|d\| = 1\}$. Then, we can regard the vector θ as lying on a Cartesian product of $\mathrm{Gr}(1,n)$'s.

Jacobian stability It is well known that if θ_j is a standard Gaussian random vector, then the random vector $\theta_j/\|\theta_j\|$ is uniformly distributed on $\mathrm{Gr}(1,n)$. We draw each θ_j uniformly from $\mathrm{Gr}(1,n)$ and each a_j uniformly from $\{-1,+1\}$. As mentioned in Section 4.2.3, we have $J^R(\theta) = J(\theta)$. Thus, our goal now is to establish the stability of J. To begin, let $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N$ denote the dataset and $u(\theta) = [f(x_1,\theta,a), f(x_2,\theta,a), \ldots, f(x_N,\theta,a)]^\top$ denote the output vector. Following [18, 59, 66], we make the following assumption on $\mathcal{S}$.

ASSUMPTION 5.11. *For any $(x,y) \in \mathcal{S}$, it holds that $\|x\| = 1$ and $|y| = \mathcal{O}(1)$. For any $x_i, x_j \in \mathcal{S}_x$ with $i \neq j$, it holds that $x_i \neq \pm x_j$. In addition, the input vector x satisfies $\mathbb{E}[x] = \mathbf{0}$ and the covariance matrix $V = \mathbb{E}[xx^\top]$ is positive definite with minimum eigenvalue $\sigma_V > 0$.*

The above assumptions on the dataset $\mathcal{S}$ are mild as explained in [66, Assumption 1]. The positive-definite property of the variance V is used to ensure the well-posedness of the input-output map (5.14). If V is just positive semidefinite, one can replace it by the shift matrix $V + \sigma_V I$ in (5.14) and remove the assumption on V. Motivated by [66], we use $[x_i^\top \theta_j^0]_{k-}$ to represent the k-th smallest entry of $[x_i^\top \theta_1^0, x_i^\top \theta_2^0, \ldots, x_i^\top \theta_m^0]$ in absolute value. Since V is positive definite and $\mathrm{Gr}(1,n) = \{d \in \mathbb{R}^n : \|d\| = 1\}$ is compact, for $i = 1, \ldots, N$, the function $u \mapsto \varphi_i(u) = \frac{x_i}{\sqrt{u^\top V u}} - \frac{V u u^\top x_i}{(u^\top V u)^{3/2}}$ is L-Lipschitz on $\mathrm{Gr}(1,n)$ for some constant $L > 0$, i.e., $\|\varphi_i(u) - \varphi_i(v)\| \leq L\|u - v\|$ for any $u, v \in \mathrm{Gr}(1,n)$. To prove the desired Jacobian stability result, we need the following lemmas. They extend those in [66], which are developed for the Euclidean setting, to the Grassmann manifold setting. In what follows, we use δ_A to denote the indicator function of an event A, i.e., δ_A takes the value 1 if the event A happens and 0 otherwise.

LEMMA 5.12. *Let $\theta_j, \theta_j^0 \in \mathrm{Gr}(1,n)$, where $j = 1, \ldots, m$, be given. Suppose that for some $k \in \{1, \ldots, m\}$, we have $\left\|\theta - \theta^0\right\| \leq \sqrt{k}[x_i^\top \theta_j^0]_{k-}$ for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, m$. Then, we have*

$$\left\|J(\theta) - J(\theta^0)\right\|^2 \leq \frac{2NkM + NkL}{m}, \tag{5.15}$$

where $M = \max_{i \in \{1,\ldots,N\}} \left(\max_{u \in \mathrm{Gr}(1,n)} \left\| \frac{x_i}{\sqrt{u^\top V u}} - \frac{V u u^\top x_i}{(u^\top V u)^{3/2}} \right\|^2 \right)$.

Proof. Let $A_{i,j}$ denote the event that the signs of $x_i^\top \theta_j$ and $x_i^\top \theta_j^0$ are different. We claim that, for $i = 1, 2, \ldots, N$, there are at most $2k$ non-zero entries of $\{\delta_{A_{i,j}}\}_{j=1}^m$. Otherwise, there exists an $i \in \{1, \ldots, N\}$ such that

$$\begin{aligned}\|\theta - \theta^0\|^2 &\geq \sum_{j=1}^m |x_i^\top \theta_j - x_i^\top \theta_j^0|^2 \\ &\geq \sum_{j \in \{j:\delta_{A_{i,j}}=1\}} |x_i^\top \theta_j - x_i^\top \theta_j^0|^2 \geq \sum_{j \in \{j:\delta_{A_{i,j}}=1\}} |x_i^\top \theta_j^0|^2 > k[x_i^\top \theta_j^0]_{k-}^2,\end{aligned}$$

which contradicts our assumption. Now, the generalized Jacobian of f with respect to θ is given by

$$J(\theta) = \frac{1}{\sqrt{m}} \sum_{j=1}^m \sum_{i=1}^N a_j \left[\delta_{x_i^\top \theta_1 \geq 0} \cdot \varphi_i(\theta_1)^\top, \ldots, \delta_{x_i^\top \theta_m \geq 0} \cdot \varphi_i(\theta_m)^\top\right].$$

When $x_i^\top \theta_j$ and $x_i^\top \theta_j^0$ have the same sign, the difference $\delta_{x_i^\top \theta_j \geq 0} \cdot \frac{a_j}{\sqrt{m}} \varphi_i(\theta_j) - \delta_{x_i^\top \theta_j^0 \geq 0} \cdot \frac{a_j}{\sqrt{m}} \varphi_i(\theta_j^0)$ is either $\mathbf{0}$ or $\frac{a_j}{\sqrt{m}}(\varphi_i(\theta_j) - \varphi_i(\theta_j^0))$. Splitting $\|J(\theta) - J(\theta^0)\|^2$ into two parts according to the event $A_{i,j}$ yields

$$\begin{aligned}&\|J(\theta) - J(\theta^0)\|^2 \\ \leq &\frac{M}{m} \sum_{(x_i,y_i)\in\mathcal{S}} \sum_{j=1}^m \delta_{A_{i,j}} + \frac{L}{m} \sum_{(x_i,y_i)\in\mathcal{S}} \sum_{j=1}^m \|\theta_j - \theta_j^0\|^2 \\ \leq &\frac{2NkM}{m} + \frac{L}{m} \sum_{(x_i,y_i)\in\mathcal{S}} \|\theta - \theta^0\|^2 \\ \leq &\frac{2NkM + NkL}{m},\end{aligned}$$

where the last inequality follows from the assumption on $\|\theta - \theta^0\|$ and the fact that $|[x_i^\top \theta_j^0]_{k-}| \leq 1$ for $i = 1, \ldots, N$ and $j = 1, \ldots, m$. ∎

The next lemma gives an upper bound on the probability of the event $\{|x_i^\top \theta_j| \leq \gamma\}$ for all $\gamma > 0$, which will be used to estimate $[x_i^\top \theta_j^0]_{k-}$ in Lemma 5.14.

LEMMA 5.13. *Let v be uniformly distributed on $\mathrm{Gr}(1, n)$, $x \in \mathrm{Gr}(1, n)$ be a given unit-norm vector, and $\gamma > 0$ be a given positive number, where $n \geq 2$. Then, we have $\mathbb{P}(|x^\top v| \leq \gamma) \leq \sqrt{\pi n}\gamma$. Moreover, the dependence on n in the bound is optimal up to constant factors.*

Proof. Without loss of generality, we may assume that $x = (1, 0, \ldots, 0)$ since the Euclidean inner product and the distribution of v are invariant under orthogonal transformation. Then, we have $x^\top v = v_1$. Let $Z_1, \ldots, Z_n$ be standard Gaussian random variables. Then, the random variable $x^\top v$ has the same distribution as $B := \frac{Z_1}{\sqrt{Z_1^2 + \cdots + Z_n^2}}$. It is well known that B^2 follows the distribution $\mathrm{Beta}(\frac{1}{2}, \frac{n-1}{2})$ [30, Section 25.2]. As a result, the density function h of B can be explicitly written as

$$h(r) = \frac{\Gamma(\frac{n}{2})}{\sqrt{\pi}\Gamma(\frac{n-1}{2})}(1 - r^2)^{\frac{n-3}{2}}, \quad |r| < 1. \tag{5.16}$$

It follows directly that

$$\mathbb{P}(|x^\top v| \leq \gamma) = \mathbb{P}(|B| \leq \gamma) = \int_{-\gamma}^{\gamma} h(r)dr \leq \frac{\gamma\Gamma(\frac{n}{2})}{\sqrt{\pi}\Gamma(\frac{n-1}{2})} \leq \sqrt{\pi n}\gamma, \tag{5.17}$$

where the last step uses the classic result $\Gamma(\frac{n}{2}) \leq \pi\sqrt{n}\Gamma(\frac{n-1}{2})$ in calculus.

To see the optimality of the dependence on n in the bound, note that for $\gamma \leq \frac{1}{\sqrt{n}}$, we have

$$\mathbb{P}(|x^\top v| \leq \gamma) = \mathbb{P}(|B| \leq \gamma) = \int_{-\gamma}^{\gamma} h(r)dr \geq \frac{\gamma\Gamma(\frac{n}{2})}{2\sqrt{\pi}\Gamma(\frac{n-1}{2})} \geq \frac{5}{12\sqrt{2e}}\sqrt{n},$$

where the third step uses $(1-r^2)^{\frac{n-3}{2}} \geq 1 - \frac{n-3}{2}r^2$ and the fact that $\gamma \leq \frac{1}{\sqrt{n}}$, and the last step follows from an application of Stirling's formula; see, e.g., [56, Eq. (33)]. Hence, the dependence on n in the bound is optimal up to constant factors. ∎

Using the above lemmas, we show that Assumption 5.9 will hold with high probability.

LEMMA 5.14. *Let $\theta_j, \theta_j^0 \in \mathrm{Gr}(1,n)$, where $j = 1,\ldots,m$, be given. For any given $Q, \epsilon > 0$, if $\|\theta - \theta^0\| \leq Q$, then with probability at least $1-\epsilon$, we will have*

$$\|J(\theta) - J(\theta^0)\|^2 \leq \frac{2(\pi n)^{\frac{1}{3}} N^{\frac{5}{3}} M Q^{\frac{2}{3}}}{\epsilon^{\frac{2}{3}} m^{\frac{1}{3}}} + \frac{(\pi n)^{\frac{1}{3}} N^{\frac{5}{3}} L Q^{\frac{2}{3}}}{\epsilon^{\frac{2}{3}} m^{\frac{1}{3}}}. \tag{5.18}$$

Proof. For given integers $k \in \{1,\ldots,m\}$ and $i \in \{1,2\ldots,N\}$, we prove that with probability at least $1 - \epsilon/N$, there will be at most $k-1$ hidden units θ_j^0 such that $|x_i^\top \theta_j^0| \leq \frac{k\epsilon}{Nm\sqrt{\pi n}}$. For $\tau > 0$, let γ_τ be the positive number such that $\mathbb{P}(|g| \leq \gamma_\tau) = \tau$, where g follows the same distribution as $x_i^\top \theta_j^0$. It follows from Lemma 5.13 that $\gamma_\tau \geq \frac{1}{\sqrt{\pi n}}\tau$. Let $\tau = \frac{k\epsilon}{Nm}$. Then, we have

$$\mathbb{E}\left[\sum_{j=1}^{m} \delta_{|x_i^\top \theta_j^0| \leq \gamma_\tau}\right] = \sum_{j=1}^{m} \mathbb{P}\left[|x_i^\top \theta_j^0| \leq \gamma_\tau\right] \leq \frac{k\epsilon}{N}. \tag{5.19}$$

Applying the Markov inequality yields

$$\mathbb{P}\left[\sum_{j=1}^{m} \delta_{|x_i^\top \theta_j^0| \leq \gamma_\tau} \geq k\right] \leq \frac{\epsilon}{N}. \tag{5.20}$$

Therefore, by taking $k = \frac{Q^{\frac{2}{3}} m^{\frac{2}{3}} (\pi n)^{\frac{1}{3}} N^{\frac{2}{3}}}{\epsilon^{\frac{2}{3}}}$, the inequalities $\sqrt{k}[x_i^\top \theta^0]_{k-} \geq \frac{k^{\frac{3}{2}}\epsilon}{Nm\sqrt{\pi n}} = Q$ will hold simultaneously for $i = 1,\ldots,N$ with probability at least $1-\epsilon$. The desired conclusion then follows from Lemma 5.12. ∎

Linear convergence of RNGD With the help of Lemma 5.14, we are now ready to establish the convergence rate of the RNGD method when applied to the two-layer neural network with batch normalization.

THEOREM 5.15. *Suppose that Assumptions 5.8 and 5.11 hold. Let $\epsilon > 0$ be a given constant. Suppose that the number m of hidden units satisfies*

$$m = \Omega\left(\frac{128(L+2M)^3 \pi n N^6 \kappa_L^2}{\mu^2 \sigma_0^8 \sigma_V \epsilon^3 \min\left\{\frac{1}{2}, \frac{\mu}{6\kappa_L}\right\}^6}\right),$$

where the constants $L, M, \kappa_L, \mu, \sigma_0, \sigma_V$ are defined previously. If we draw θ_j^0 uniformly from $\mathrm{Gr}(1,n)$ and a_j uniformly from $\{-1,+1\}$ for $j = 1,2\ldots,m$, then the

Riemannian Jacobian stability condition in Assumption 5.9 will hold with probability at least $1-\epsilon$. *Furthermore, when* $m \geq \frac{16(L+2M)^3\pi n N^5\kappa_L^2}{9\sigma_0^8\kappa_H^2\epsilon^2\min\left\{\frac{1}{2},\frac{\mu}{6\kappa_L}\right\}^6}$, $\|u^0-y\| \leq \frac{\mu}{3\kappa_H}$, *and* $\eta \leq \min\left\{1, \left(\frac{1}{6\|u^0-y\|}-\frac{\kappa_H}{2\mu}\right)\cdot\frac{3\mu^2\sigma_0}{8\kappa_R\kappa_L^2}\right\}$, *with probability at least* $1-\epsilon$, *we will have*

$$\|u^{k+1}-y\| \leq \left(1-\frac{1}{2}\eta\right)\|u^k-y\|. \tag{5.21}$$

Proof. By Assumption 5.11 and the fact that a_j is drawn uniformly from $\{-1,+1\}$, we have $\mathbb{E}\left[u^0\right]=\mathbf{0}$ and

$$\begin{aligned}\mathbb{E}\left[(u_j^0)^2\right] &= \mathbb{E}\left[\frac{1}{m}\left(\sum_{j=1}^m a_j\phi\left(\frac{(\theta_j^0)^\top(x-\mathbb{E}[x])}{\sqrt{(\theta_j^0)^\top V\theta_j^0}}\right)\right)^2\right] \\ &= \mathbb{E}\left[\frac{1}{m}\sum_{j=1}^m\phi\left(\frac{(\theta_j^0)^\top x}{\sqrt{(\theta_j^0)^\top V\theta_j^0}}\right)^2\right] = \mathcal{O}\left(\frac{1}{\sigma_V}\right), \quad j=1,\ldots,N.\end{aligned}$$

This gives

$$\mathbb{E}\left[\|u^0-y\|^2\right] = \|y\|^2 + 2y^\top\mathbb{E}[u^0] + \mathbb{E}\left[\|u^0\|^2\right] = \mathcal{O}\left(\frac{N}{\sigma_V}\right). \tag{5.22}$$

Applying the Markov inequality, we see that $\|u^0-y\|^2 = \mathcal{O}\left(\frac{2N}{\epsilon\sigma_V}\right)$ will hold with probability at least $1-\frac{1}{2}\epsilon$. This, together with the result of Lemma 5.14 with $Q = 4\kappa_L(\mu\sigma_0)^{-1}\|u^0-y\|$, implies that Assumption 5.9 will hold with probability at least $1-\epsilon$ for $m = \Omega\left(\frac{128(L+2M)^3\pi n N^6\kappa_L^2}{\mu^2\sigma_0^8\sigma_V\epsilon^3\min\left\{\frac{1}{2},\frac{\mu}{6\kappa_L}\right\}^6}\right)$.

To establish the convergence rate result, observe from Theorem 5.10 that $\|\theta^k-\theta^0\| \leq 4\kappa_L(\mu\sigma_0)^{-1}\|u^0-y\|$ when $\|u^0-y\| \leq \frac{\mu}{3\kappa_H}$ and $\eta \leq \min\left\{1, \left(\frac{1}{6\|u^0-y\|}-\frac{\kappa_H}{2\mu}\right)\cdot\frac{3\mu^2\sigma_0}{8\kappa_R\kappa_L^2}\right\}$. By taking $Q = 4\kappa_L\sigma_0^{-1}/(3\kappa_H)$ in Lemma 5.14, we see that Assumption 5.9 will hold with probability at least $1-\epsilon$ if $m \geq \frac{16(L+2M)^3\pi n N^5\kappa_L^2}{9\sigma_0^8\kappa_H^2\epsilon^2\min\left\{\frac{1}{2},\frac{\mu}{6\kappa_L}\right\}^6}$. Following the proof of Theorem 5.10, we conclude that (5.21) will hold for all $k \geq 0$ with probability at least $1-\epsilon$. This completes the proof. ∎

6 Numerical results

In this section, we demonstrate the efficacy of our proposed method via numerical experiments on three problems: Low-rank matrix completion, low-dimension subspace learning, and deep learning model training. Our code is available at https://github.com/hujiangpku/RNGD.

6.1 Low-rank matrix completion

We compare our proposed RNGD method with the Riemannian stochastic gradient descent (RSGD) method [11], the Riemannian adaptive stochastic gradient algorithm (RASA) [32], the Riemannian stochastic variance-reduced gradient (RSVRG) method [52], and the Riemannian conjugate gradient (RCG) method without preconditioner [14, 46, 51]. All algorithms are initialized by the QR decomposition of a random n-by-p matrix whose entries are generated from the standard Gaussian distribution. We consider two real datasets. One is taken

from the Jester joke recommender system,[2] which contains ratings (with scores from -10.00 to $+10.00$) of 100 jokes from 24983 users. The other is the movie rating dataset MovieLens-1M,[3] which contains ratings (with stars from 1 to 5) of 3952 movies from 6040 users. In the experiments, each dataset is randomly divided into 2 sets, one for training and the other for testing. We utilize the implementations of RSGD and RSVRG given in the RSOpt package[4] and the implementation of RCG given in the Manopt package.[5] For RASA, the LR-type variant is adopted due to its efficiency. The default parameters therein are used. For RNGD, the same variance reduction technique as that in RSVRG is adopted to update both the estimated gradient and the approximate RFIM (4.5). Specifically, we compute $a_i(U)$ for all i in each outer iteration and update $a_i(U)$ if the i-th sample is used in the estimation of the gradient. We use fixed step sizes for RNGD and RSVRG. For RSGD, the step size η_k is set to $\eta_k = \frac{\eta_0}{1+\eta_0 k/10}$. As suggested in [32], the step size $\eta_k = \eta_0/\sqrt{k}$ is used for RASA. We search in the set $\{2, 1, 0.5, \ldots, 2 \times 10^{-8}, 10^{-8}, 5 \times 10^{-9}\}$ to find the best initial step size η_0 for RSGD and RASA and the best step size for RSVRG. The step size for RNGD is set to 0.05 for both datasets.

Figure 6.1 reports the mean squared error (MSE) on both the training and testing datasets, which are defined as $\|\mathcal{P}_{\Omega_{\text{train}}}(UA - X)\|^2/|\Omega_{\text{train}}|$ and $\|\mathcal{P}_{\Omega_{\text{test}}}(UA - X)\|^2/|\Omega_{\text{test}}|$, respectively, where Ω_{train} and Ω_{test} are the sets of known indices in the training and testing datasets, respectively. The label "#grad/N" means the number of epochs, which is defined as the number of cycles through the full dataset. The label "time" represents the wall-clock time. We run all algorithms with a specified number of epochs for different datasets. On the Jester dataset, we see that RNGD, RSVRG, and RCG achieve lower MSEs than the other two methods. Furthermore, RNGD converges faster than RSVRG and RCG. In the case of the MovieLens-1M dataset, RASA and RSVRG exhibit fast reductions of MSEs in the early iterations. However, RNGD returns a point with the lowest MSE.

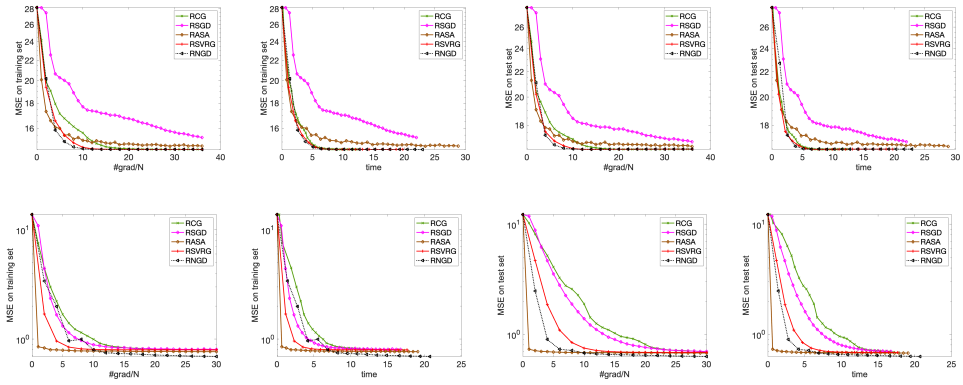

FIG. 6.1. *Numerical results for LRMC on the Jester dataset (first row) and the MovieLens-1M dataset (second row).*

[2]The dataset Jester can be downloaded from https://grouplens.org/datasets/jester
[3]The dataset MovieLens-1M can be downloaded from https://grouplens.org/datasets/movielens
[4]The code of RSOpt can be downloaded from https://github.com/hiroyuki-kasai/RSOpt
[5]The code of Manopt can be downloaded from https://github.com/NicolasBoumal/manopt

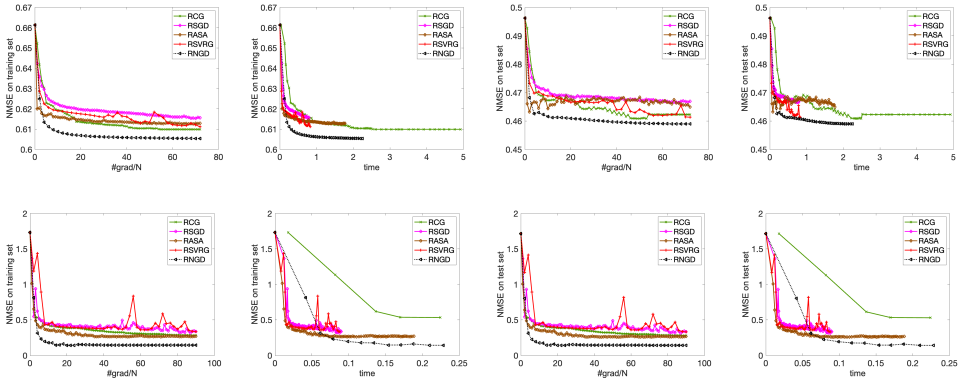

FIG. 6.2. *Numerical results for multitask learning on the School dataset (first row) and the Sarcos dataset (second row).*

6.2 Low-dimension subspace learning We compare our proposed RNGD with RCG, RSGD, RASA, and RSVRG on two real-world datasets: *School* [22] and *Sarcos* [57]. The dimension p is set to be 6 for both datasets. We choose the best step sizes for RSVRG, RASA, and RSGD from the set $\{1, 0.5, 0.2, 0.1, 0.05, 0.02, \ldots, 10^{-8}, 5 \times 10^{-9}, 2 \times 10^{-9}, 10^{-9}\}$. We use the step size 4 (resp., 1) on the *School* (resp., *Sarcos*) dataset for RNGD. All the codes are implemented within the RSOpt framework and the other parameters of the algorithms are set to the default values therein.

Figure 6.2 reports the normalized MSE (NMSE) [40] on both datasets, which is the mean of the normalized squared error of all tasks. For both datasets, RNGD returns a point with the lowest NMSE. Especially for the *Sarcos* dataset, a significant difference in the NMSE between RNGD and other methods is observed. Another noteworthy phenomenon is that RSGD and RSVRG tend to be less efficient than RCG. This demonstrates the advantage of using the Fisher information.

6.3 Deep learning model training Batch normalization and momentum-based optimizer are standard techniques to train state-of-the-art image classification models [24, 50, 54]. We evaluate the proposed method with Kronecker-factorized approximate RFIM described in Section 4, denoted by MKFAC, on VGG16BN [54] and WRN-16-4 [64] while the benchmark datasets CIFAR-10/100 [36] are used. The detailed network structures are described in [54, 64]. In VGG16BN, batch normalization layers are added before every ReLU activation layer. Additionally, we change the number of neurons in fully connected layers from 4096 to 512 and remove the middle layer of the last three in VGG due to memory allocation problems (otherwise, one has to compute the inverse of 4096^2-by-4096^2 matrices). This setting is also adopted in [17, 63].

The baseline algorithms are SGD, Adam, KFAC [39], AdamP, and SGDP [25]. The tangential projections are used to control the increase in norms of the weight parameters in AdamP and SGDP. These methods can be seen as approximate Riemannian first-order methods. We fine tune the initial learning rates of the baseline algorithms by searching in the set $\{0.5, 0.2, 0.1, 0.05, 0.02, 0.01, \ldots, 5 \times 10^{-5}, 2 \times 10^{-5}, 10^{-5}\}$. The learning rate decays in epoch 30, 60, and 90 with a decay rate 0.1, where an epoch is defined as one cycle through the full training dataset. We choose the parameters β_1, β_2 in Adam and AdamP from the set $\{0.9, 0.99, 0.999\}$. We search in the set $\{0.05, 0.1, 0.2, 0.5, 1, 2\}$ to determine the damping parameter λ used in cal-

TABLE 6.1
Classification accuracy of various networks on CIFAR-10/100 (median of five runs).

Dataset	CIFAR-10		CIFAR-100	
Model	WRN-16-4	VGG16BN	WRN-16-4	VGG16BN
SGD	93.84	92.88	74.30	71.79
SGDP	93.42	92.49	73.67	71.54
Adam	92.53	89.88	71.64	62.79
AdamP	92.55	91.43	71.23	58.88
KFAC	93.90	94.36	74.31	76.38
MKFAC	**94.06**	**94.76**	**74.55**	**77.28**

culating the natural direction $(F_k + \lambda I)^{-1} g^k$ and update the KFAC matrix in epoch 30, 60, and 90. The initial damping parameter of KFAC is set to 2 in all four tasks. We set the weight decay to 5×10^{-4} for all algorithms. Each mini-batch contains 128 samples. The maximum number of epochs is set to 100 for all algorithms. For MKFAC, we use RNGD for parameters constrained on the Grassmann manifold and SGD for the remaining parameters. Let η, η_g denote the learning rates for the Euclidean space and Grassmann manifold, respectively. For the dataset CIFAR-10, we set $\eta_g = 0.25$ and $\eta = 0.05$ with decay rates 0.2 and 0.1, respectively. The weight decay is only applied to the unconstrained weights with parameter 5×10^{-4}. The initial MKFAC damping parameters for WRN-16-4 and VGG16BN are set to 1 and 2 with decay rates 0.8 and 0.5, respectively, when the preconditioners update in epoch 30, 60, and 90. For the dataset CIFAR-100, we set $\eta_g = 0.3$ for WRN-16-4, $\eta_g = 0.15$ for VGG16BN, and $\eta = 0.05$ for both. The learning rate η_g has a decay rate 0.15 for WRN-16-4 and 0.2 for VGG16BN, while η has a decay rate 0.1 for both of them. The initial MKFAC damping parameters for VGG16BN and WRN16-4 are set to 0.5 and 1 with decay rates 0.5 and 0.8, respectively. Other settings are the same as KFAC.

Table 6.1 presents the comparison of the baseline and the proposed algorithms on CIFAR-10 and CIFAR-100 datasets. We list the best classification accuracy in 100 epochs, where the results are obtained from the median of 5 runs. The performance of our proposed MKFAC method is the best in all four tasks. Compared with the second-order type method KFAC, our MKFAC method reaches higher accuracy, though KFAC has a much better behavior than SGD on these tasks. Compared with the manifold geometry-based first-order algorithms SGDP and AdamP, we see that using second-order information can give better accuracy than using first-order information alone.

7 Conclusion In this paper, we developed a novel efficient RNGD method for tackling the problem of minimizing a sum of negative log-probability losses over a manifold. Key to our development is a new notion of FIM on manifolds, which we introduced in this paper and could be of independent interest. We established the global convergence of RNGD and the local convergence rate of a deterministic version of RNGD. Our numerical results on representative machine learning applications demonstrate the efficiency and efficacy of the proposed method.

REFERENCES

[1] P.-A. ABSIL, R. MAHONY, AND R. SEPULCHRE, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008.

[2] P.-A. ABSIL, R. MAHONY, AND J. TRUMPF, *An extrinsic look at the Riemannian Hessian*, in Geometric Science of Information, Springer, 2013, pp. 361–368.

[3] P.-A. Absil and J. Malick, *Projection-like retractions on matrix manifolds*, SIAM Journal on Optimization, 22 (2012), pp. 135–158.
[4] S.-i. Amari, *Neural learning in structured parameter spaces — natural Riemannian gradient*, in Advances in Neural Information Processing Systems, vol. 9, 1996, pp. 127–133.
[5] S.-I. Amari, *Natural gradient works efficiently in learning*, Neural Computation, 10 (1998), pp. 251–276.
[6] R. K. Ando, T. Zhang, and P. Bartlett, *A framework for learning predictive structures from multiple tasks and unlabeled data*, Journal of Machine Learning Research, 6 (2005), pp. 1817–1853.
[7] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer, *Scalable second order optimization for deep learning*, arXiv:2002.09018, (2020).
[8] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, in Advances in Neural Information Processing Systems - Deep Learning Symposium, 2016, p. arXiv preprint arXiv:1607.06450.
[9] A. Bahamou, D. Goldfarb, and Y. Ren, *A mini-block natural gradient method for deep neural networks*, arXiv:2202.04124, (2022).
[10] G. Bécigneul and O.-E. Ganea, *Riemannian adaptive optimization methods*, in International Conference on Learning Representations, 2019.
[11] S. Bonnabel, *Stochastic gradient descent on Riemannian manifolds*, IEEE Transactions on Automatic Control, 58 (2013), pp. 2217–2229.
[12] F. Bouchard, A. Breloy, A. Renaux, and G. Ginolhac, *Riemannian geometry and Cramér-Rao bound for blind separation of Gaussian sources*, in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 4717–4721.
[13] N. Boumal, *An Introduction to Optimization on Smooth Manifolds*, Available online, May, 2020.
[14] N. Boumal and P.-A. Absil, *Low-rank matrix completion via preconditioned optimization on the Grassmann manifold*, Linear Algebra and its Applications, 475 (2015), pp. 200–239.
[15] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, *A stochastic quasi-Newton method for large-scale optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1008–1031.
[16] R. Chen, M. Menickelly, and K. Scheinberg, *Stochastic optimization using a trust-region method and random models*, Mathematical Programming, 169 (2018), pp. 447–487.
[17] M. Cho and J. Lee, *Riemannian approach to batch normalization*, in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5231–5241.
[18] S. S. Du, X. Zhai, B. Poczos, and A. Singh, *Gradient descent provably optimizes over-parameterized neural networks*, in International Conference on Learning Representations, 2019.
[19] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, Journal of Machine Learning Research, 12 (2011).
[20] H. Flanders, *Differentiation under the integral sign*, The American Mathematical Monthly, 80 (1973), pp. 615–627.
[21] D. Goldfarb, Y. Ren, and A. Bahamou, *Practical quasi-Newton methods for training deep neural networks*, in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 2386–2396.
[22] H. Goldstein, *Multilevel modelling of survey data*, Journal of the Royal Statistical Society. Series D (The Statistician), 40 (1991), pp. 235–244.
[23] R. Grosse and J. Martens, *A Kronecker-factored approximate Fisher matrix for convolution layers*, in International Conference on Machine Learning, 2016, pp. 573–582.
[24] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[25] B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J.-W. Ha, *AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights*, in International Conference on Learning Representations, 2021.
[26] J. Hu, X. Liu, Z.-W. Wen, and Y.-X. Yuan, *A brief introduction to manifold optimization*, Journal of the Operations Research Society of China, 8 (2020), pp. 199–248.
[27] J. Hu, A. Milzarek, Z. Wen, and Y. Yuan, *Adaptive quadratically regularized Newton method for Riemannian optimization*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1181–1207.
[28] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International Conference on Machine Learning, 2015, pp. 448–456.

[29] B. Jiang, S. Ma, A. M.-C. So, and S. Zhang, *Vector transport-free SVRG with general retraction for Riemannian optimization: Complexity analysis and practical implementation*, arXiv:1705.09059, (2017).
[30] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions, volume 2*, vol. 289, John Wiley & Sons, 1995.
[31] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Advances in Neural Information Processing Systems, vol. 26, 2013, pp. 315–323.
[32] H. Kasai, P. Jawanpuria, and B. Mishra, *Riemannian adaptive stochastic gradient algorithms on matrix manifolds*, in International Conference on Machine Learning, 2019, pp. 3262–3271.
[33] H. Kasai and B. Mishra, *Inexact trust-region algorithms on Riemannian manifolds.*, in NeurIPS, 2018, pp. 4254–4265.
[34] H. Kasai, H. Sato, and B. Mishra, *Riemannian stochastic quasi-Newton algorithm with variance reduction and its convergence analysis*, in International Conference on Artificial Intelligence and Statistics, 2018, pp. 269–278.
[35] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, International Conference for Learning Representations, (2015).
[36] A. Krizhevsky, G. Hinton, et al., *Learning multiple layers of features from tiny images*, (2009).
[37] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), p. 436.
[38] J. Martens, *New insights and perspectives on the natural gradient method*, The Journal of Machine Learning Research, 21 (2020), pp. 5776–5851.
[39] J. Martens and R. Grosse, *Optimizing neural networks with Kronecker-factored approximate curvature*, in International Conference on Machine Learning, 2015, pp. 2408–2417.
[40] B. Mishra, H. Kasai, P. Jawanpuria, and A. Saroop, *A Riemannian gossip approach to subspace learning on Grassmann manifold*, Machine Learning, 108 (2019), pp. 1783–1803.
[41] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, second ed., 2006.
[42] L. Nurbekyan, W. Lei, and Y. Yang, *Efficient natural gradient descent methods for large-scale optimization problems*, arXiv:2202.06236, (2022).
[43] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, *Information-geometric optimization algorithms: A unifying picture via invariance principles*, Journal of Machine Learning Research, 18 (2017), pp. 1–65.
[44] M. Pilanci and M. J. Wainwright, *Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence*, SIAM Journal on Optimization, 27 (2017), pp. 205–245.
[45] Y. Ren and D. Goldfarb, *Kronecker-factored quasi-Newton methods for convolutional neural networks*, arXiv:2102.06737, (2021).
[46] W. Ring and B. Wirth, *Optimization methods on Riemannian manifolds and their application to shape space*, SIAM Journal on Optimization, 22 (2012), pp. 596–627.
[47] H. Robbins and S. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics, (1951), pp. 400–407.
[48] F. Roosta-Khorasani and M. W. Mahoney, *Sub-sampled Newton methods*, Mathematical Programming, 174 (2019), pp. 293–326.
[49] T. Salimans and D. P. Kingma, *Weight normalization: A simple reparameterization to accelerate training of deep neural networks*, in Advances in Neural Information Processing Systems, vol. 29, 2016, pp. 901–909.
[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *MobileNetV2: Inverted residuals and linear bottlenecks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[51] H. Sato, *Riemannian Optimization and Its Applications*, Springer, 2021.
[52] H. Sato, H. Kasai, and B. Mishra, *Riemannian stochastic variance reduced gradient algorithm with retraction and vector transport*, SIAM Journal on Optimization, 29 (2019), pp. 1444–1472.
[53] N. N. Schraudolph, *Fast curvature matrix-vector products for second-order gradient descent*, Neural Computation, 14 (2002), pp. 1723–1738.
[54] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv:1409.1556, (2014).
[55] S. T. Smith, *Covariance, subspace, and intrinsic Cramér-Rao bounds*, IEEE Transactions on Signal Processing, 53 (2005), pp. 1610–1630.
[56] A. M.-C. So, *Non-asymptotic performance analysis of the semidefinite relaxation detector in digital communications*. Preprint, 2010.

[57] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal, *Statistical learning for humanoid robots*, Autonomous Robots, 12 (2002), pp. 55–69.
[58] X. Wang and Y.-x. Yuan, *Stochastic trust region methods with trust region radius depending on probabilistic models*, arXiv:1904.03342, (2019).
[59] X. Wu, S. S. Du, and R. Ward, *Global convergence of adaptive gradient methods for an over-parameterized neural network*, arXiv:1902.07111, (2019).
[60] M. Yang, A. Milzarek, Z. Wen, and T. Zhang, *A stochastic extra-step quasi-Newton method for nonsmooth nonconvex optimization*, Mathematical Programming, (2021), pp. 1–47.
[61] M. Yang, D. Xu, H. Chen, Z. Wen, and M. Chen, *Enhance curvature information by structured stochastic quasi-Newton methods*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10654–10663.
[62] M. Yang, D. Xu, Q. Cui, Z. Wen, and P. Xu, *An efficient Fisher matrix approximation method for large-scale neural network optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2022).
[63] M. Yang, D. Xu, Z. Wen, M. Chen, and P. Xu, *Sketch-based empirical natural gradient methods for deep learning*, Journal of Scientific Computing, 92 (2022), pp. 1–29.
[64] S. Zagoruyko and N. Komodakis, *Wide residual networks*, arXiv:1605.07146, (2016).
[65] D. Zhang and S. D. Tajbakhsh, *Riemannian stochastic variance-reduced cubic regularized Newton method*, arXiv:2010.03785, (2020).
[66] G. Zhang, J. Martens, and R. Grosse, *Fast convergence of natural gradient descent for overparameterized neural networks*, in Advances in Neural Information Processing Systems, 2019, pp. 8082–8093.
[67] H. Zhang, S. J. Reddi, and S. Sra, *Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds*, in Advances in Neural Information Processing Systems, 2016, pp. 4592–4600.