OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Rieoptax : Riemannian Optimization in JAX

Saiteja Utpala SAITEJAUTPALA@GMAIL.COM
Independent

Andi Han ANDI.HAN@SYDNEY.EDU.AU
University of Sydney

Pratik Jawanpuria PRATIK.JAWANPURIA@MICROSOFT.COM
Microsoft India

Bamdev Mishra BAMDEVM@MICROSOFT.COM
Microsoft India

Abstract

We present Rieoptax, an open source Python library for Riemannian optimization in JAX. We show that many differential geometric primitives, such as Riemannian exponential and logarithm maps, are usually faster in Rieoptax than existing frameworks in Python, both on CPU and GPU. We support a range of basic and advanced stochastic optimization solvers like Riemannian stochastic gradient, stochastic variance reduction, and adaptive gradient methods. A distinguishing feature of the proposed toolbox is that we also support differentially private optimization on Riemannian manifolds.

1. Introduction

Riemannian geometry is a generalization of the Euclidean geometry [57, 76] to general Riemannian manifolds. It includes several nonlinear spaces such as the set of positive definite matrices [20, 108], Grassmann manifold of subspaces [5, 17, 38], Stiefel manifold of orthogonal matrices [5, 29, 38], Kendall shape spaces [68, 69, 81], hyperbolic spaces [27, 110, 111], and special Euclidean and orthogonal group [44, 100, 104], to name a few.

Optimization with manifold based constraints has become increasingly popular and has been employed in various applications such as low rank matrix completion [23], learning taxonomy embeddings [87, 88], neural networks [45, 62–64, 86, 92], density estimation [51, 59], optimal transport [9, 31, 54, 84, 101], shape analysis [61, 106], and topological dimension reduction [65], among others.

In addition, privacy preserving machine learning [2, 30, 34–36, 83, 105] has become crucial in real applications, which has been generalized to manifold-constrained problems very recently [52, 94, 112]. Nevertheless, such a feature is absent in existing Riemannian optimization libraries [18, 24, 72, 80, 82, 103, 109].

In this work, we introduce Rieoptax (**Rie**mannian **Opt**imization in **Jax**), an open source Python library for Riemannian optimization in JAX [25, 42]. The proposed library is mainly driven by the needs of efficient implementation of manifold-valued operations and optimization solvers, readily compatible with GPU and even TPU processors as well as the needs of privacy-supported Riemannian optimization. To the best of our knowledge, Rieoptax is the first library to provide privacy guarantees within the Riemannian optimization framework.

1.1. Background on Riemannian optimization, privacy, and JAX

Riemannian optimization. Riemannian optimization [5, 22, 43] considers the following problem

$$\min_{w \in \mathcal{M}} f(w), \tag{1}$$

where $f : \mathcal{M} \to \mathbb{R}$, and $\mathcal{M}$ denotes a Riemannian manifold. Instead of considering (1) as a constrained problem, Riemannian optimization [5, 22] views it as an unconstrained problem on the manifold space. Riemannian (stochastic) gradient descent [21, 114] generalizes the Euclidean gradient descent with intrinsic updates on manifold, i.e., $w_{t+1} = \mathrm{Exp}_{w_t}(-\eta_t \operatorname{grad} f(w_t))$, where $\operatorname{grad} f(w_t)$ is the Riemannian (stochastic) gradient, $\mathrm{Exp}_w(\cdot)$ is the Riemannian exponential map at w and η_t is the step size. Recent years have witnessed significant advancements for Riemannian optimization where more advanced solvers are generalized from the Euclidean space to Riemannian manifolds. These include variance reduction methods [49, 50, 66, 98, 116, 117], adaptive gradient methods [16, 67], accelerated gradient methods [7, 8, 53, 78, 115], quasi-Newton methods [60, 91], zeroth-order methods [77] and second order methods, such as trust region methods [4] and cubic regularized Newton's methods [6].

Differential privacy on Riemannian manifolds. Differential privacy (DP) provides a rigorous treatment for data privacy by precisely quantifying the deviation in the model's output distribution under modification of a small number of data points [34–37]. Provable guarantees of DP coupled with properties like immunity to arbitrary post-processing and graceful composability have made it a de-facto standard of privacy with steadfast adoption in the real applications [3, 11, 33, 39, 85]. Further, it has been shown empirically that DP models resist various kinds of leakage attacks that can cause privacy violations [14, 28, 93, 97, 102, 118].

Recently, there is a surge of interest on differential privacy over Riemannian manifolds, which has been explored in the context of Fréchet mean computation [94, 112] and more generally for empirical risk minimization problems on Riemannian manifolds [52]. [52] proposed differentially private Riemannian (stochastic) gradient descent methods by perturbing the Riemannian gradient with noise from the tangent Gaussian distribution before taking step: $\zeta = \operatorname{grad} f(w) + \epsilon, \epsilon \sim \mathcal{N}_w(0, \sigma^2)$. More recently, [10] has proposed efficient sampling procedures from the tangent Gaussian distribution for large scale and stochastic optimization scenarios.

JAX and its ecosystem. JAX [25, 42] is recently introduced machine learning framework which support automatic differentiation capabilities [15] via `grad()`. Further some of the distinguishing features of JAX are just-in-time (JIT) compilation using the accelerated linear algebra (XLA) compiler [48] via `jit()`, automatic vectorization (batch-level parallelism) support with `vmap()`, and strong support for parallel computation via `pmap()`. All the above transformations can be composed arbitrarily because JAX follows the functional programming paradigm and implements these as pure functions.

Given that JAX has many interesting features, its ecosystem has been constantly expanding in the last couple of years. Examples include neural network modules (Flax [56], Haiku [58], Equinox [71], Jraph [46], Equivariant-MLP [40]), reinforcement learning agents (Rlax [13]), Euclidean optimization algorithms (Optax [13]), federated learning (Fedjax [95]), optimal transport toolboxes (Ott [32]), sampling algorithms (Blackjax [73]), differential equation solvers (Diffrax [70]), rigid body simulators (Brax [41]), and differentiable physics (Jax-md [99]), among others.

1.2. Rieoptax

We believe that the proposed framework for Riemannian optimization in JAX is a timely contribution that brings several benefits of JAX and new features (such as privacy support) to the manifold optimization community discussed below.

- **Automatic and efficient vectorization with `vmap()`.** Functions that are written for inputs of size 1 can be converted to functions that take batch of inputs by wrapping it with `vmap()`. For example, the function `def dist(point_a, point_b)` for computing distance between a single `point_a` and a single `point_b` can be converted to function that computes distance between a batch of `point_a` and/or a batch `point_b` by wrapping `dist` with `vmap()` without modifying the `dist()` function. This is useful in many cases, e.g., Fréchet mean computation $\min_{w \in \mathcal{M}} \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(w) := \frac{1}{n} \sum_{i=1}^{n} \text{dist}^2(w, z_i) \right\}$. Furthermore, vectorization with `vmap()` is usually faster or on par with manual vectorization [25].
- **Per-example gradient clipping.** A key process in differentially private optimization is per-example gradient clipping $\frac{1}{n} \sum_{i=1}^{n} \text{clip}_\tau(\text{grad} f_i(w))$, where clip_τ ensures norm is atmost τ. Here, the order of operations is important: the gradients are first clipped and then averaged. Popular libraries including Autograd [79], Pytorch [89] and Tensorflow [1] are heavily optimized to directly compute the mean gradient $\frac{1}{n} \sum_{i=1}^{n} \text{grad} f_i(w)$ and hence do not expose per-example gradients i.e., $\text{grad} f_i(w)$. Hence, one has to resort to ad-hoc techniques [47, 75, 96] or come up with algorithmic modifications [26] which inherently have speed versus performance trade-off. JAX, however, offers native support for handling such scenarios and JAX-based differentially private Euclidean optimization methods have been shown to be much faster than their non-JAX counterparts [107]. We observe that JAX offer similar benefits for differentially private Riemannian optimization as well.
- **Single Source Multiple Devices (SSMD) paradigm.** JAX follows the SSMD paradigm, and therefore, the code written for CPUs can be run on GPU/TPUs without any additional modification.

Rieoptax is available at https://github.com/SaitejaUtpala/Rieoptax/.

2. Design and Implementation overview

The package currently implements several commonly used geometries, optimization algorithms and differentially private mechanisms on manifolds. More geometries and advanced solvers will be added in the future.

2.1. Core

- `rieoptax.core.ManifoldArray`: lightweight wrapper of the `jax` device array with `manifold` attribute and used to model array constrained to manifold. It is registered as `Pytree` to ensure compatibility jax primitives like `grad()` and `vmap()`.
- `rieoptax.core.rgrad`: Riemannian gradient operator.

2.2. Geometries

Geometry module contains manifolds equipped with different Riemannian metrics. Each Geometry contains Riemannian inner product `inp()`, induced norm `norm()`, Riemannian exponential `exp()`, logarithm maps `log()`, induced Riemannian distance `dist()`, parallel transport `pt()`, and transformation from the Euclidean gradient to Riemannian gradient `egrad_to_rgrad()`.

Manifolds include symmetric positive definite (SPD) matrices $\mathrm{SPD}(m) := \{\mathbf{X} \in \mathbb{R}^{m\times m} : \mathbf{X} = \mathbf{X}^\top, \mathbf{X} \succ 0\}$, hyperbolic space, Grassmann manifold $\mathcal{G}(m,r) := \{[\mathbf{X}] : \mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{X}^\top\mathbf{X} = \mathbf{I}\}$ where $[\mathbf{X}] := \{\mathbf{XO} : O \in O(r)\}$, $O(r)$ denotes the orthogonal group and hypersphere $\mathcal{S}(d) := \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^\top\mathbf{x} = 1\}$. We use $T_x\mathcal{M}$ to represent the tangent space at x and $\langle u, v\rangle_x$ to represent the Riemannian inner product. For more detailed treatment on these geometries, we refer to [5, 22, 111].

- `rieoptax.geometry.spd.SPDAffineInvariant`: SPD matrices with the affine-invariant metric [90]: $\mathrm{SPD}(m)$ with $\langle \mathbf{U}, \mathbf{V}\rangle_{\mathbf{X}} = \mathrm{tr}(\mathbf{X}^{-1}\mathbf{U}\mathbf{X}^{-1}\mathbf{V})$ for $\mathbf{U}, \mathbf{V} \in T_{\mathbf{X}}\mathrm{SPD}(m)$.
- `rieoptax.geometry.spd.SPDLogEuclidean`: SPD matrices with the Log-Euclidean metric [12]: $\mathrm{SPD}(m)$ with $\langle \mathbf{U}, \mathbf{V}\rangle_{\mathbf{X}} = \mathrm{tr}\big(\mathrm{D}_{\mathbf{U}}\mathrm{logm}(\mathbf{X})\mathrm{D}_{\mathbf{V}}\mathrm{logm}(\mathbf{X})\big)$ where $\mathrm{D}_{\mathbf{U}}\mathrm{logm}(\mathbf{X})$ is the directional derivative of matrix logarithm at $\mathbf{X}$ along $\mathbf{U}$.
- `rieoptax.geometry.hyperbolic.PoincareBall`: the Poincare-ball model of Hyperbolic space with Poincare metric [111], i.e., $\mathbb{D}(d) := \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^\top\mathbf{x} < 1\}$ with $\langle \mathbf{u}, \mathbf{v}\rangle_{\mathbf{x}} = 4\mathbf{u}^\top\mathbf{v}/(1 - \mathbf{x}^\top\mathbf{x})^2$ for $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\mathbb{D}(d)$.
- `rieoptax.geometry.hyperbolic.LorentzHyperboloid`: the Lorentz Hyperboloid model of Hyperbolic space [111], i.e., $\mathbb{H}(d) = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{x}, \mathbf{x}\rangle_{\mathcal{L}} = -1\}$ with $\langle \mathbf{u}, \mathbf{v}\rangle_{\mathbf{x}} = \langle \mathbf{u}, \mathbf{v}\rangle_{\mathcal{L}}$ for $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\mathbb{H}(d)$, where $\langle \mathbf{u}, \mathbf{v}\rangle_{\mathcal{L}} := -u_0v_0 + u_1v_1 + \cdots u_{d-1}v_{d-1}$.
- `rieoptax.geometry.grassmann.GrassmannCanonicalMetric`: the Grassmann manifold with the canonical metric [38], i.e., $\mathcal{G}(m,r)$ with $\langle \mathbf{U}, \mathbf{V}\rangle_{\mathbf{X}} = \mathrm{tr}\big(\mathbf{U}^T\mathbf{V}\big)$ for $\mathbf{U}, \mathbf{V} \in T_{\mathbf{X}}\mathcal{G}(m,r)$.
- `rieoptax.geometry.hypersphere.HypersphereCanonicalMetric`: the hypersphere manifold which canonical metric [5, 22], i.e., $\mathcal{S}(d)$ with $\langle \mathbf{u}, \mathbf{v}\rangle_{\mathbf{x}} = \mathbf{u}^\top\mathbf{v}$ for $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\mathcal{S}(d)$.

2.3. Optimizers

Optimizers module contains Riemannian optimization algorithms. Design of optimizers follows Optax [13], which implements every optimizer by chaining of few common transformations. Where every optimizer

- `riepotax.optimizers.first_order.rsgd`: Riemannian stochastic gradient descent [21].
- `riepotax.optimizers.first_order.rsvrg`: Riemannian stochastic variance reduced gradient descent [116].
- `riepotax.optimizers.first_order.rsrg`: Riemannian stochastic recursive gradient descent [66].

- `riepotax.optimizers.first_order.rasa`: Riemannian adaptive stochastic gradient algorithm [67].
- `riepotax.optimizers.zeroth_order.zo_rgd`: zeroth-order Riemannian gradient descent [77].

2.4. Privacy mechanism

Mechanism module contains differential private mechanisms on Riemannian manifolds.

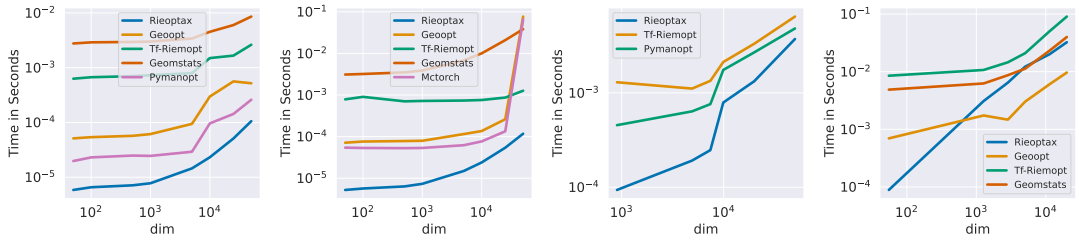
- `rieoptax.mechanism.output_perturbation.RieLaplaceMechanism`: the Riemannian Laplace mechanism [94] which is used for privatizing Fréchet mean computation.
- `rieoptax.mechanism.output_perturbation.LogEuclideanMechanism`: the Log-Euclidean mechanism [112] which is used for differentially private Fréchet mean on SPD matrices with log-Euclidean metric.
- `rieoptax.mechanism.gradient_perturbation.DPRGDMechanism`: noise calibration for differentially private Riemannian gradient descent [52] based on moments accountant [2] in the `autodp` library [113].
- `rieoptax.mechanism.gradient_perturbation.DPRSGDMechanism`: noise calibration for Differentially private Riemannian stochastic gradient descent [52] based on moments accountant [2] in `autodp` library [113].

3. Benchmarking Rieoptax

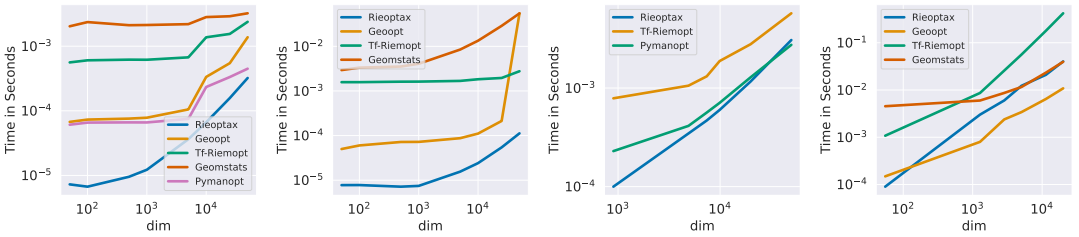
In this section, we benchmark the proposed Rieoptax against existing Riemannian optimization libraries in Python. These include Pytorch [89] based Mctorch [80] and Geoopt [72], Tensorflow [103] based Tensorflow-Riemopt (Tf-Riemopt) [103], Numpy [55] based Pymanopt [109], and Tensorflow based Geomstats [82]. While Geomstats supports Numpy, Pytorch, and Tensorflow as backend, currently only the Tensorflow backend provides support for GPUs. Other non-Python based libraries include Manopt [24] in Matlab and Manopt.jl [18] in Julia [19].

We benchmark the Riemannian exponential (Exp) and logarithm (Log) maps with the proposed Rieoptax against the aforementioned Python libraries whenever available with `64bitfloat` precision. For CPU benchmarking, we use the AMD EPYC 7B1 processor with 2 cores and 16GB RAM. For GPU benchmarking, we use CUDA version 11.0 on 16GB Tesla P100.

- **Hypersphere:** hypersphere $\mathcal{S}(d)$ is supported in Geoopt, Tf-Riemopt, Geomstats, McTorch, and Pyamanopt. McTorch does not support the Exp and Log maps. On GPU, Geomstats raises an error. We benchmark all expect McTorch and Geomstats for dimensions $d \in \{50, 100, 500, 1\,000, 5\,000, 10\,000, 25\,000, 50\,000\}$.
- **Lorentz hyperboloid model:** the Lorentz hyperboloid model $\mathbb{H}(d)$ is supported in Geoopt, Tf-Riemopt, Geomstats, and Mctorch. While the Exp map is available in Mctorch, it does not implement the Log map. We benchmark for dimensions $d \in \{50, 100, 500, 1\,000, 5\,000, 10\,000, 25\,000, 50\,000\}$.

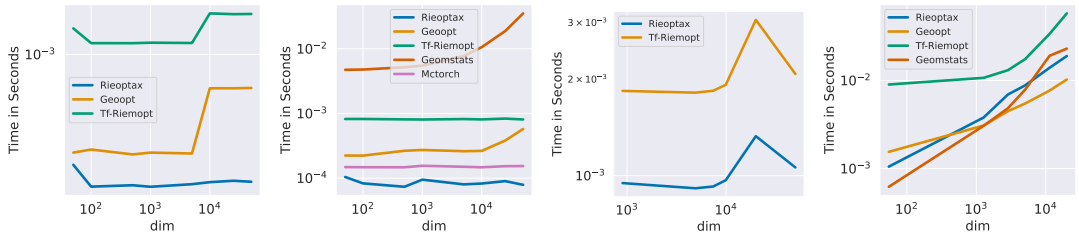


(*a*) Hypersphere Exp (*b*) Lorentz Exp (*c*) Grassmann Exp (*d*) SPD Exp

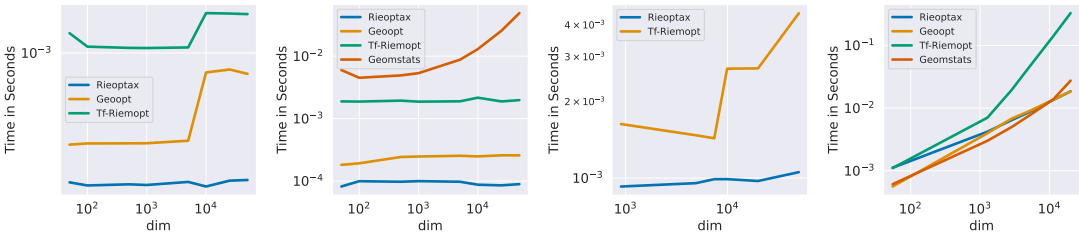


(*e*) Hypersphere Log (*f*) Lorentz Log (*g*) Grassmann Log (*h*) SPD Log

Figure 1: Benchmarking of geometric primitives on CPU.



(*a*) Hypersphere Exp (*b*) Lorentz Exp (*c*) Grassmann Exp (*d*) SPD Exp



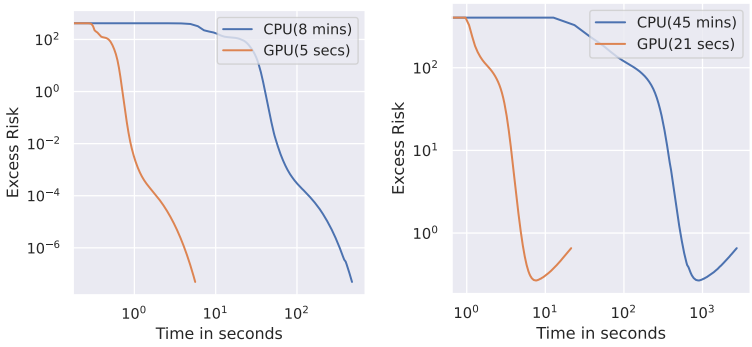
(*e*) Hypersphere Log (*f*) Lorentz Log (*g*) Grassmann Log (*h*) SPD Log

Figure 2: Benchmarking of geometric primitives on GPU.

- **Grassmann:** Grassmann manifold $\mathcal{G}(m, r)$ is supported in Tf-Riemopt, Pymanopt, McTorch, and Geomstats. McTorch does not support the Exp and Log maps. Geomstats represents Grassmann elements in projector matrices form $\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{m \times m}$ instead of $\mathbf{X} \in \mathbb{R}^{m \times r}$, which is prohibitively expensive. We, therefore, exclude these three libraries from benchmarking. We benchmark for matrix sizes $(m, r) \in \{(100, 10), (500, 10), (750, 10), (1\,000, 10), (2\,000, 10), (5\,000, 10)\}$.

- **SPD with affine-invariant metric:** SPD manifold $\mathrm{SPD}(m)$ with the affine-invariant metric is supported in Geoopt, Tf-Riemopt, Geomstats, and McTorch. McTorch, however, does not support the Exp and Log maps. We benchmark all except McTorch for matrix sizes $m \in \{10, 50, 75, 100, 150, 200\}$.

Figures 1 and 2 present the timing results with CPU- and GPU-based computations, respectively. Overall, we observe that Rieoptax offers significant time improvements, especially on GPUs. For the SPDAffineInvariant case, Rieoptax is slightly slower than Geoopt because `eigh` which provides eigen decomposition is slightly slower in JAX compared to Pytorch. Given that JAX is a relatively new framework, we believe it would be faster even in this case in the near future.



(*a*) Non-private PCA. (*b*) $(0.1, 10^{-6})$-private PCA.

Figure 3: Timing of Rieoptax for PCA on the TinyImagetNet dataset as optimization on Grassmann manifold $\mathcal{G}(12\,288, 5)$ on both CPU and GPU. The GPU implementation achieves a significant speedup than CPU on both non-private and private PCA problems.

4. An example on PCA

In this section, we consider the problem of principal components analysis (PCA) by viewing it as an optimization problem on the Grassmann manifold [5, 22], i.e.,

$$\min_{\mathbf{U} \in \mathcal{G}(m,r)} \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{z}_i - \mathbf{U}\mathbf{U}^T \mathbf{z}_i \right\|_2^2, \tag{2}$$

where $\mathbf{z}_i \in \mathbb{R}^n$ denote the data points. The Rieoptax implementation for solving the problem (2) is shown in Listing 3.

We provide timing of Rieoptax on TinyImageNet [74] which has a training set of 10^5 images in dimensions of $3 \times 64 \times 64$ on both CPU and GPU. We take $5\,000$ images and vectorize each image to produce a sample matrix of size $(n, d) = (5\,000, 12\,288)$. We compute the top $r = 5$ principal components, which leads to an optimization problem on $\mathcal{G}(12\,288, 5)$.

For non-private PCA, we run the full Riemannian gradient descent method for 400 epochs. For private PCA, we run the differentially private Riemannian gradient descent method [52] for 200 epochs with a privacy configuration of $\epsilon = 0.1, \delta = 10^{-6}$ and gradient clipping parameter of 0.1.

```python
from functools import partial
from jax import jit, vmap
from jax.numpy.linalg import norm
from rieoptax.core import rgrad, ManifoldArray
from rieoptax.geometry.grassmann import GrassmannCanonical
from rieoptax.optimizer.first_order import rsgd, dp_rsgd
from rieoptax.mechanism.gradient_perturbation import DP_RGD_Mechanism
from rieoptax.optimizer.update import apply_updates

def fit(params, data, optimizer, epochs, private=False):

    @jit
    def step(params, opt_state, data):
        def cost(params, data):
            def _cost(params, data):
                diff = data-params.value @ (params.value.T @ data)
                return norm(diff)**2
            return vmap(_cost, in_axes=(None,0))(params, data).mean()

        rgrad_fn = rgrad(cost)
        if private:
            data = data[:, None]
        #per-example gradient
        rgrad_fn = vmap(rgrad_fn, in_axes=(None, 0))
        #calculates Riemannian gradients
        rgrads = rgrad_fn(params, data)
        updates, opt_state = optimizer.update(rgrads, opt_state, params)
        #update using Riemannian Exp
        params = apply_updates(params, updates)
        return params, opt_state, loss_value

    opt_state = optimizer.init(params)
    for i in range(epochs):
        params, opt_state, loss_value = step(params, opt_state, data)

#initialization
U_init = ManifoldArray(value=init, manifold=GrassmannCanonical())
# non private PCA
lr, epochs = (3e-3, 400)
optimizer = rsgd(lr)
non_private_U = fit(U_init, Z, optimizer, epochs)
#(eps, delta) differentially private PCA
eps, delta, clip_norm, epochs = (1.0, 1e-6, 0.1, 200)
sigma = DP_RGD_Mechanism(eps, delta, clip_norm, n)
private_optimizer = dp_rsgd(lr, sigma, clip_norm)
private_U = fit(U_init, Z, optimizer, epochs, private=True)
```

Listing 1: The Rieoptax code for the private and non-private PCA problem. `dp_rsgd` and `rsgd` are the private and non-private optimizers, respectively.

For both the private and non-private algorithms, we choose the same initialization and a learning rate of 3×10^{-3}. Figures 3(a) and 3(b) show excess risk against runtime (in seconds) for non-private and private PCA, respectively.

5. Conclusion and future roadmap

In this work, we present a Python library for (privacy-supported) Riemannian optimization, Rieoptax, and illustrate its efficacy on both CPU and GPU architectures. Our roadmap includes adding support for more manifold geometries, optimization algorithms, and a collection of example codes showcasing the usage of Rieoptax in various applications, especially with differential privacy.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In *USENIX Conference on Operating Systems Design and Implementation*, 2016.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[3] John M Abowd. The US Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

[4] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

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

[6] Naman Agarwal, Nicolas Boumal, Brian Bullins, and Coralia Cartis. Adaptive regularization with cubics on manifolds. *Mathematical Programming*, 188(1):85–134, 2021.

[7] Kwangjun Ahn and Suvrit Sra. From Nesterov's estimate sequence to Riemannian acceleration. In *Conference on Learning Theory*, pages 84–118. PMLR, 2020.

[8] Foivos Alimisis, Antonio Orvieto, Gary Bécigneul, and Aurelien Lucchi. A continuous-time perspective for modeling acceleration in Riemannian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 1297–1307. PMLR, 2020.

[9] Jason Altschuler, Sinho Chewi, Patrik R Gerber, and Austin Stromme. Averaging on the Bures-Wasserstein manifold: dimension-free convergence of gradient descent. *Advances in Neural Information Processing Systems*, 34:22132–22145, 2021.

[10] Anonymous. Improved differentially private riemannian optimization. *Submitted to Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=paguBNtqiO. Under review.

[11] Differential Privacy Team, Apple. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.

[12] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM journal on matrix analysis and applications*, 29(1):328–347, 2007.

[13] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Luyu Wang, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

[14] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. *arXiv preprint arXiv:2201.04845*, 2022.

[15] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[16] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019.

[17] Thomas Bendokat, Ralf Zimmermann, and P-A Absil. A Grassmann manifold handbook: Basic geometry and computational aspects. *arXiv preprint arXiv:2011.13699*, 2020.

[18] Ronny Bergmann. Manopt. jl: Optimization on manifolds in julia. *Journal of Open Source Software*, 7(70):3866, 2022.

[19] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[20] Rajendra Bhatia. Positive definite matrices. In *Positive Definite Matrices*. Princeton university press, 2009.

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

[22] Nicolas Boumal. An introduction to optimization on smooth manifolds. To appear with Cambridge University Press, Jun 2022. URL https://www.nicolasboumal.net/book.

[23] Nicolas Boumal and Pierre-antoine Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. *Advances in neural information processing systems*, 24, 2011.

[24] Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1): 1455–1459, 2014.

[25] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[26] Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Hanwen Shen, and Uthaipon Tantipongpipat. Fast and memory efficient differentially private-sgd via jl projections. *Advances in Neural Information Processing Systems*, 34:19680–19691, 2021.

[27] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31(59-115):2, 1997.

[28] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA, 2019. USENIX Association. ISBN 9781939133069.

[29] Rudrasis Chakraborty and Baba C Vemuri. Statistics on the Stiefel manifold: theory and applications. *The Annals of Statistics*, 47(1):415–438, 2019.

[30] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[31] Sinho Chewi, Tyler Maunu, Philippe Rigollet, and Austin J Stromme. Gradient descent algorithms for Bures-Wasserstein barycenters. In *Conference on Learning Theory*, pages 1276–1304. PMLR, 2020.

[32] Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal transport tools (OTT): A jax toolbox for all things Wasserstein. *arXiv preprint arXiv:2201.12324*, 2022.

[33] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, 30, 2017.

[34] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[35] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[36] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[37] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[38] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[39] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[40] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning*, pages 3318–3328. PMLR, 2021.

[41] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

[42] Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9), 2018.

[43] Daniel Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

[44] Jean Gallier and Jocelyn Quaintance. *Differential geometry and Lie groups: a computational perspective*, volume 12. Springer Nature, 2020.

[45] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018.

[46] Jonathan Godwin*, Thomas Keck*, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Veličković, and Alvaro Sanchez-Gonzalez. Jraph: A library for graph neural networks in jax., 2020. URL http://github.com/deepmind/jraph.

[47] Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

[48] Google. Xla : Compiling machine learning for peak performance, 2020.

[49] Andi Han and Junbin Gao. Improved variance reduction methods for Riemannian non-convex optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[50] Andi Han and Junbin Gao. Riemannian stochastic recursive momentum method for non-convex optimization. In *International Joint Conference on Artificial Intelligence*, pages 2505–2511, 8 2021.

[51] Andi Han, Bamdev Mishra, Pratik Kumar Jawanpuria, and Junbin Gao. On Riemannian optimization over positive definite matrices with the Bures-Wasserstein geometry. *Advances in Neural Information Processing Systems*, 34:8940–8953, 2021.

[52] Andi Han, Bamdev Mishra, Pratik Jawanpuria, and Junbin Gao. Differentially private Riemannian optimization. *arXiv preprint arXiv:2205.09494*, 2022.

[53] Andi Han, Bamdev Mishra, Pratik Jawanpuria, and Junbin Gao. Riemannian accelerated gradient methods via extrapolation. *arXiv preprint arXiv:2208.06619*, 2022.

[54] Andi Han, Bamdev Mishra, Pratik Jawanpuria, and Junbin Gao. Riemannian block SPD coupling manifold and its application to optimal transport. *arXiv preprint arXiv:2201.12933*, 2022.

[55] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[56] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

[57] Sigurdur Helgason. *Differential geometry, Lie groups, and symmetric spaces*. Academic press, 1979.

[58] Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

[59] Reshad Hosseini and Suvrit Sra. An alternative to EM for Gaussian mixture models: batch and stochastic Riemannian optimization. *Mathematical programming*, 181(1):187–223, 2020.

[60] Wen Huang, Kyle A Gallivan, and P-A Absil. A Broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[61] Wen Huang, Kyle A Gallivan, Anuj Srivastava, and Pierre-Antoine Absil. Riemannian optimization for registration of curves in elastic shape analysis. *Journal of Mathematical Imaging and Vision*, 54(3):320–343, 2016.

[62] Zhiwu Huang and Luc Van Gool. A Riemannian network for SPD matrix learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[63] Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep learning on Lie groups for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6099–6108, 2017.

[64] Zhiwu Huang, Jiqing Wu, and Luc Van Gool. Building deep networks on Grassmann manifolds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[65] Oleg Kachan. Persistent homology-based projection pursuit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 856–857, 2020.

[66] Hiroyuki Kasai, Hiroyuki Sato, and Bamdev Mishra. Riemannian stochastic recursive gradient algorithm. In *International Conference on Machine Learning*, pages 2516–2524. PMLR, 2018.

[67] Hiroyuki Kasai, Pratik Jawanpuria, and Bamdev Mishra. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In *International Conference on Machine Learning*, pages 3262–3271. PMLR, 2019.

[68] David G Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London mathematical society*, 16(2):81–121, 1984.

[69] David G Kendall. A survey of the statistical theory of shape. *Statistical Science*, 4(2):87–99, 1989.

[70] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[71] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[72] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in PyTorch. *arXiv preprint arXiv:2005.02819*, 2020.

[73] Junpeng Lao and Rémi Louf. Blackjax: A sampling library for JAX, 2020. URL http://github.com/blackjax-devs/blackjax.

[74] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[75] Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021(1), 2021.

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

[77] Jiaxiang Li, Krishnakumar Balasubramanian, and Shiqian Ma. Stochastic zeroth-order Riemannian derivative estimation and optimization. *Mathematics of Operations Research*, 2022.

[78] Yuanyuan Liu, Fanhua Shang, James Cheng, Hong Cheng, and Licheng Jiao. Accelerated first-order methods for geodesically convex optimization on Riemannian manifolds. *Advances in Neural Information Processing Systems*, 30, 2017.

[79] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML workshop*, 2015.

[80] Mayank Meghwanshi, Pratik Jawanpuria, Anoop Kunchukuttan, Hiroyuki Kasai, and Bamdev Mishra. McTorch, a manifold optimization library for deep learning. *arXiv preprint arXiv:1810.01811*, 2018.

[81] Nina Miolane, Susan Holmes, and Xavier Pennec. Template shape estimation: correcting an asymptotic bias. *SIAM Journal on Imaging Sciences*, 10(2):808–844, 2017.

[82] Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, et al. Geomstats: a Python package for Riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

[83] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[84] Bamdev Mishra, NTV Satyadev, Hiroyuki Kasai, and Pratik Jawanpuria. Manifold optimization for non-linear optimal transport problems. *arXiv preprint arXiv:2103.00902*, 2021.

[85] Joe Near. Differential privacy at scale: Uber and Berkeley collaboration. In *Enigma 2018 (Enigma 2018)*, 2018.

[86] Xuan Son Nguyen, Luc Brun, Olivier Lézoray, and Sébastien Bougleux. A neural network based on spd manifold learning for skeleton-based hand gesture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12036–12045, 2019.

[87] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[88] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

[89] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[90] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

[91] Chunhong Qi, Kyle A Gallivan, and P-A Absil. Riemannian BFGS algorithm with applications. In *Recent Advances in Optimization and its Applications in Engineering*, pages 183–192. Springer, 2010.

[92] Guodong Qi, Huimin Yu, Zhaohui Lu, and Shuzhao Li. Transductive few-shot classification on the oblique manifold. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8412–8422, 2021.

[93] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[94] Matthew Reimherr, Karthik Bharath, and Carlos Soto. Differential privacy over Riemannian manifolds. *Advances in Neural Information Processing Systems*, 34:12292–12303, 2021.

[95] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. FedJAX: Federated learning simulation with JAX. *arXiv preprint arXiv:2108.02117*, 2021.

[96] Gaspar Rochette, Andre Manoel, and Eric W Tramel. Efficient per-example gradient computations in convolutional neural networks. *arXiv preprint arXiv:1912.06015*, 2019.

[97] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019.

[98] Hiroyuki Sato, Hiroyuki Kasai, and Bamdev Mishra. Riemannian stochastic variance reduced gradient algorithm with retraction and vector transport. *SIAM Journal on Optimization*, 29 (2):1444–1472, 2019.

[99] Samuel S. Schoenholz and Ekin D. Cubuk. Jax m.d. a framework for differentiable physics. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[100] Jon M Selig. *Geometric fundamentals of robotics*, volume 128. Springer, 2005.

[101] Dai Shi, Junbin Gao, Xia Hong, ST Boris Choy, and Zhiyong Wang. Coupling matrix manifolds assisted optimization for optimal transport problems. *Machine Learning*, 110(3):533–558, 2021.

[102] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[103] Oleg Smirnov. TensorFlow RiemOpt: a library for optimization on Riemannian manifolds. *arXiv preprint arXiv:2105.13921*, 2021.

[104] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[105] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

[106] Anuj Srivastava, Eric Klassen, Shantanu H Joshi, and Ian H Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE transactions on pattern analysis and machine intelligence*, 33(7):1415–1428, 2010.

[107] Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private SGD via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34:26409–26421, 2021.

[108] Yann Thanwerdas and Xavier Pennec. O(n)-invariant Riemannian metrics on SPD matrices. *arXiv preprint arXiv:2109.05768*, 2021.

[109] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

[110] Abraham Albert Ungar. *Analytic hyperbolic geometry and Albert Einstein's special theory of relativity*. World Scientific, 2008.

[111] Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

[112] Saiteja Utpala, Praneeth Vepakomma, and Nina Miolane. Differentially private Fréchet mean on the manifold of symmetric positive definite (SPD) matrices. *arXiv preprint arXiv:2208.04245*, 2022.

[113] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

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

[115] Hongyi Zhang and Suvrit Sra. An estimate sequence for geodesically convex optimization. In *Conference On Learning Theory*, pages 1703–1723. PMLR, 2018.

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

[117] Pan Zhou, Xiao-Tong Yuan, and Jiashi Feng. Faster first-order methods for stochastic nonconvex optimization on Riemannian manifolds. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 138–147. PMLR, 2019.

[118] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.