Introduction to Riemannian Optimization

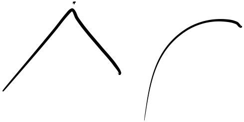
An introduction to optimization on Riemannian matrix manifolds

Benyamin Ghojogh

Winter 2023

Riemannian Manifold

Riemannian manifold vs Euclidean space



- **Vector space**: a vector space (also called a linear space) is a set whose elements, often called **vectors**, can be added together and multiplied (scaled) by numbers called scalars.
- **Euclidean space**: A Euclidean space is a vector space, but with a **Euclidean distance metric** defined over it.
- **Smooth manifold**:
  - In simple words, it is a **differentiable curvy** hyper-surface.
  - In mathematical definition, it needs the concepts of topological space, chart, and homeomorphism.
- **Riemannian manifold** $\mathcal{M}$:
  - In simple words, it is a real smooth manifold $\mathcal{M}$ with a **Riemannian distance metric** (distance on the **curvy** hyper-surface) defined over it.
  - In mathematical definition, it is a real smooth manifold $\mathcal{M}$ equipped with a positive-definite inner product on the tangent space at each point.

Euclidean optimization

- In **Euclidean optimization**, the cost function is a function from the Euclidean space to a scalar:

$$\boxed{f : \mathbb{R}^d \mapsto \mathbb{R},} \quad \boxed{f : \boldsymbol{x} \mapsto f(\boldsymbol{x}).}$$

The optimization problem is:

$$\boxed{\underset{\boldsymbol{x} \in \mathbb{R}^d}{\text{minimize}} \quad f(\boldsymbol{x}),} \tag{1}$$

$tr(X^T A X)$

or equivalently:

$$\boxed{\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in \mathbb{R}^d. \end{aligned}} \tag{2}$$

If the optimization problem is constrained:

$$\boxed{\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in \mathcal{S}, \end{aligned}} \tag{3}$$

$\min_X f(X)$ s.t. $X \in M_{stiefle}$

$\min_X f(X)$ s.t. $X^T X = I$

where $\mathcal{S}$ is the feasibility set.

$M_{SPD} \leftarrow X \in S_{++} \leftarrow X \succ 0$

Riemannian optimization

- So far in the course, we covered optimization methods in the **Euclidean space**. The Euclidean optimization methods can be slightly revised to have optimization on (possibly **curvy**) **Riemannian manifolds**.
- In **Riemannian optimization** [1, 2] optimizes a cost function while the variable lies on a Riemannian manifold $\mathcal{M}$.
- The optimization variable in the Riemannian optimization is usually matrix rather than vector; hence, Riemannian optimization is also called **optimization on matrix manifolds**:

$$\underbrace{f : \mathcal{M} \to \mathbb{R},}\quad \underbrace{f : \boldsymbol{X} \mapsto f(\boldsymbol{X}).}$$

- The optimization problem is:

$$\boxed{\underset{\boldsymbol{X}\in\mathcal{M}}{\text{minimize}} \quad f(\boldsymbol{X}),} \tag{4}$$

or equivalently:

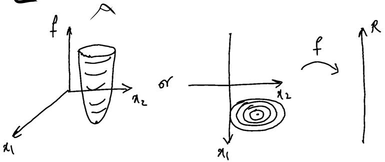
$$\boxed{\begin{aligned} &\underset{\boldsymbol{X}}{\text{minimize}} && f(\boldsymbol{X}) \\ &\text{subject to} && \boldsymbol{X} \in \mathcal{M}. \end{aligned}} \tag{5}$$

- A good technique: If the optimization problem is constrained, we may define the constraint as the matrix manifold of that constraint (such as the Stiefel (orthogonal matrix) manifold for $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{I}$) and use Eq. (5) to solve it.

Euclidean optimization vs. Riemannian optimization

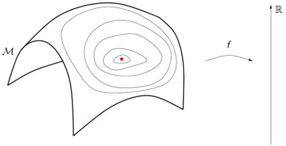
- Euclidean optimization: $f : \mathbb{R}^d \to \mathbb{R}$, $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathbb{R}^d. \end{aligned}$$

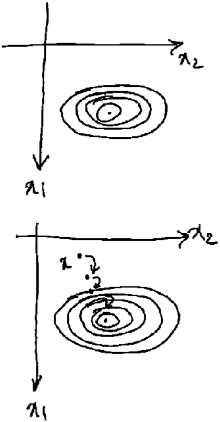


- Riemannian optimization: $f : \mathcal{M} \to \mathbb{R}$, $f : \boldsymbol{X} \mapsto f(\boldsymbol{X})$.

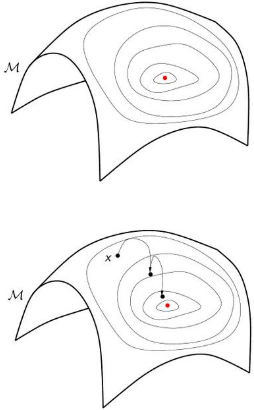
$$\begin{aligned} \underset{\boldsymbol{X}}{\text{minimize}} \quad & f(\boldsymbol{X}) \\ \text{subject to} \quad & \boldsymbol{X} \in \mathcal{M}. \end{aligned}$$



Euclidean optimization vs. Riemannian optimization



Euclidean
optimization



Riemannian
optimization

Topology and Smooth Manifold Concepts

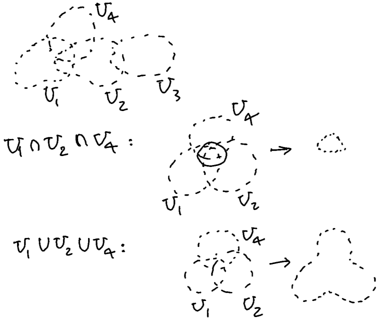
Topology and topological space

Definition (Topology and topological space [3, 4])

Let $\mathcal{X}$ be a set. A **topology** on $\mathcal{X}$ is a collection $\mathcal{T}$ of subsets $\mathcal{X}$, called open sets, satisfying:

- $\varnothing, \mathcal{X} \in \mathcal{T}$
- If $U_1, \ldots, U_k \in \mathcal{T}$, then $\bigcap_{j=1}^{k} U_j \in \mathcal{T}$. In other words, finite intersections of open sets are open.
- If $U_\alpha \in \mathcal{T}, \forall \alpha \in A$ (where A is the index set of topology), then $\bigcup_{\alpha \in A} U_\alpha \in \mathcal{T}$. In other words, arbitrary unions of open sets are open.

The pair $(\mathcal{X}, \mathcal{T})$ is called a **topological space** associated with the topology $\mathcal{T}$.



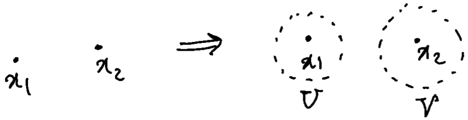
Hausdorff space

Definition (Hausdorff space [3, 4])

A topological space $(\mathcal{X}, \mathcal{T})$ is **Hausdorff** if and only if for $x_1, x_2 \in X$, $x_1 \neq x_2$, we have:

$$\exists \text{ open sets } U, V \text{ such that } x_1 \in U,\ x_2 \in V,\ U \cap V = \varnothing. \tag{6}$$

In other words, the points of a Hausdorff topological space are **separable** and **distinguishable**.



Homeomorphism and Diffeomorphism

Definition (Homeomorphism and Diffeomorphism)

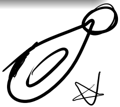
- **Homeomorphism**: A transformation from one topology to another topology without tearing up the topology. It is studied in **algebraic topology**.

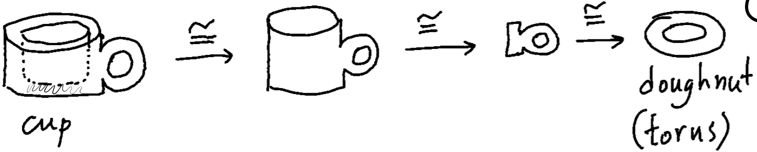
  The two topologies before and after a homeomorphism transformation are called **homeomorphic** to each other.

  The homeomorphic symbol is usually denoted by $\cong$. $\cong$

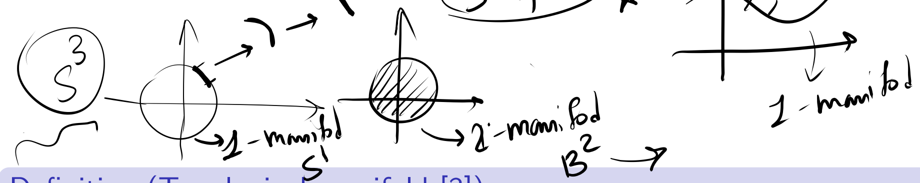
- **Diffeomorphism**: A homeomorphism transformation which is smooth and differentiable.

Example; A cup and doughnut (torus) are homeomorphic:





Topological manifold

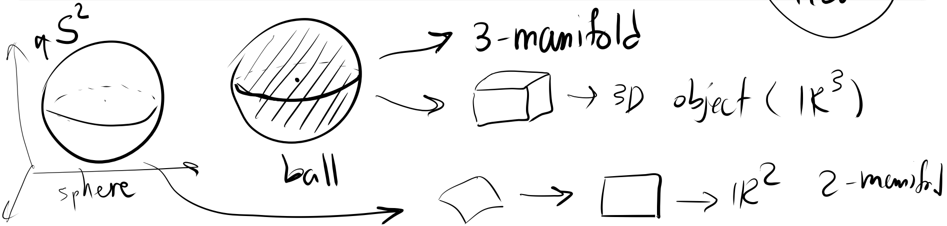


Definition (Topological manifold [3])

A topological space $(\mathcal{X}, \mathcal{T})$ is a **topological manifold** of dimension d, for $d \in \mathbb{Z}_{\geq 0}$, also called a **topological d-manifold**, if all the following conditions hold:

- $(\mathcal{X}, \mathcal{T})$ is **Hausdorff**.
- $(\mathcal{X}, \mathcal{T})$ has a **countable basis**.
- $(\mathcal{X}, \mathcal{T})$ is locally **homeomorphic** to d-dimensional Euclidean space, $\mathbb{R}^d$.

Ricci flow

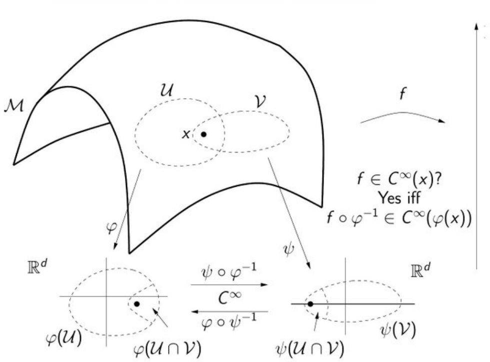


Chart

Definition (Chart [3])

Consider a topological manifold $\mathcal{M} := (\mathcal{X}, \mathcal{T})$. It is locally homeomorphic to $\mathbb{R}^d$, meaning that for all $x \in X$, there exists an open set U containing x and a homeomorphism $\phi : U \to \phi(U)$ where $\phi(U)$ is an open subset of $\mathbb{R}^d$. Such mapping is denoted by $\phi : U \xrightarrow{\cong} \phi(U)$ and the tuple (U, ϕ) is called a **coordinate chart**, or a **chart** in short, for $\mathcal{M}$.

C^1
C^2
$\vdots$
C^∞

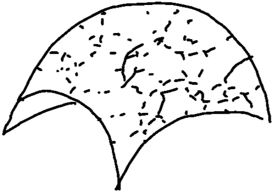


Smooth atlas

Definition (Smooth atlas [5])

A smooth **atlas** $\mathcal{A}$ for a topological d-manifold $\mathcal{M}$ is a collection of charts (U_α, ϕ_α) for $\mathcal{M}$ such that:

- They cover $\mathcal{M}$, i.e., $\bigcup_{\alpha \in A} U_\alpha = \mathcal{M}$.
- Any two charts in this collection are smoothly compatible (n.b. two charts (U, ϕ) and (V, ψ) are smoothly compatible if the mapping $\psi \circ \phi^{-1}$ is a diffeomorphism).



Definition (Maximal atlas [5])

A smooth atlas $\mathcal{A}$ for a topological d-manifold $\mathcal{M}$ is **maximal** if it is not contained in any other smooth atlas for $\mathcal{M}$.

Smooth manifold and Riemannian manifold

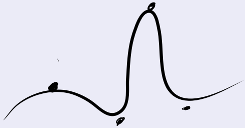
Definition (Smooth manifold [5])

A **smooth manifold** $\mathcal{M}$ of dimension d, also called a **smooth** d**-manifold**, is a topological d-manifold together with a choice of maximal smooth atlas $\mathcal{A}$ on $\mathcal{M}$.

Definition (Riemannian manifold)

Riemannian manifold $\mathcal{M}$ is a smooth manifold which also has a metric (inner product) g. Knowing the metric can determine the whole Riemannian manifold because using the metric, we can calculate:

- **inner product** on manifold
- **distance** on manifold
- **geodesic** (shortest curvy line) on manifold
- **curvature** on every point of manifold

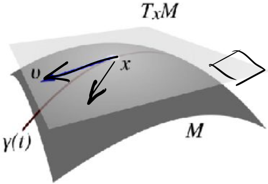


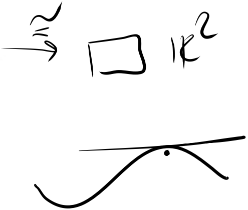
inner product   distance   geodesic   curvature

Riemannian Manifold Concepts

Riemannian concepts: tangent space, metric, norm

- **Tangent space** $T_{\mathbf{x}}\mathcal{M}$: The space of tangent vectors on the manifold $\mathcal{M}$ at the point $\boldsymbol{x}$.





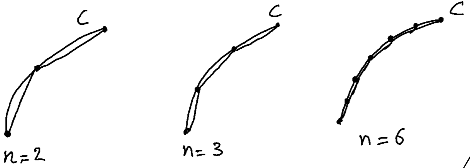
- **Riemannian metric** g:

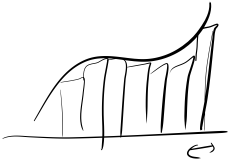
$$g_{\mathbf{x}}(\xi, \eta) : T_{\mathbf{x}}\mathcal{M} \times T_{\mathbf{x}}\mathcal{M} \to \mathbb{R}.$$

- **Norm**:

$$\|\xi\|_{\mathbf{x}} = \sqrt{g_{\mathbf{x}}(\xi, \xi)}.$$

Riemannian concepts: length of curve





Length of curve:

$$\ell(\boldsymbol{x}(t)) \approx \sum_{t=0}^{n} \big\|\boldsymbol{x}(t) - \boldsymbol{x}(t+1)\big\|$$

$$\begin{aligned}
\ell(\boldsymbol{x}(t)) &= \lim_{n\to\infty} \sum_{t=0}^{n} \big\|\boldsymbol{x}(t) - \boldsymbol{x}(t+1)\big\| \\
&= \lim_{n\to\infty} \sum_{t=0}^{n} \Big\|\frac{\boldsymbol{x}(t) - \boldsymbol{x}(t+1)}{\Delta t}\Big\|\Delta t \\
&\overset{(a)}{=} \int_0^1 \Big\|\frac{d\boldsymbol{x}(t)}{dt}\Big\| dt = \int_0^1 \|\dot{\boldsymbol{x}}(t)\| dt,
\end{aligned}$$

where (a) is because it is a Riemann's sum. In general, the length between points a and b is:

$$\boxed{\ell(\boldsymbol{x}(t)) = \int_a^b \|\dot{\boldsymbol{x}}(t)\| dt.}$$

Riemannian concepts: geodesic, gradient, Hessian

- **Geodesic**: locally minimizing curves between two points on the manifold
- **Riemannian gradient**: the direction of steepest descent of cost function (maximum growth of cost function) on the manifold

$$\nabla f(\boldsymbol{x}) = g_{\boldsymbol{x}}(\nabla f, \xi) = D_{\xi} f(\boldsymbol{x}) = \frac{d}{dt} f(\delta(t)), \quad \dot{\delta}(t) = \xi.$$

- **Riemannian Hessian**: the derivative of one of directions (ξ) in the tangent space (the derivative of derivative)

$$\boxed{\boldsymbol{B}\, f(\boldsymbol{x})\, \xi = \partial_{\xi} \nabla f(\boldsymbol{x}),}$$

where $\boldsymbol{B}$ denotes the Hessian matrix and ∂_{ξ} is the affine connection.

Riemannian concepts: logarithm and exponential maps

- **Logarithm map**:
  - In Euclidean space, **subtraction** is:

$$\boxed{\Delta = \boldsymbol{x}_n - \boldsymbol{x}_m,} \qquad \text{point} \times \text{point} \to \text{vector}.$$

  - The generalization of subtraction in the Riemannian space is **logarithm map**:

$$\mathrm{Log}_{\boldsymbol{x}_m}(\boldsymbol{x}_n) = \Delta, \qquad \text{point on } \mathcal{M} \times \text{point on } \mathcal{M} \to \text{tangent vector } T_{\boldsymbol{x}}\mathcal{M}.$$

$$\boxed{\mathrm{Log}_{\boldsymbol{x}_m}(\boldsymbol{x}_n) = \xi,} \qquad \xi \in T_{\boldsymbol{x}_m}\mathcal{M}.$$

- **Exponential map**:
  - In Euclidean space, **addition** is:

$$\boxed{\boldsymbol{x}_n = \boldsymbol{x}_m + \Delta,} \qquad \text{point} \times \text{vector} \to \text{point}.$$

  - The generalization of addition in the Riemannian space is **exponential map**:

$$\mathrm{Exp}_{\boldsymbol{x}_m}(\Delta) = \boldsymbol{x}_n, \qquad \text{point on } \mathcal{M} \times \text{tangent vector } T_{\boldsymbol{x}}\mathcal{M} \to \text{point on } \mathcal{M}.$$

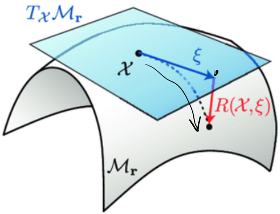
$$\boxed{\mathrm{Exp}_{\boldsymbol{x}_m}(\xi) = \boldsymbol{x}_n,} \qquad \xi \in T_{\boldsymbol{x}_m}\mathcal{M}.$$

Riemannian concepts: retraction

- The exponential map $\text{Exp}_{\boldsymbol{x}}(\xi)$ is hard to compute, because it is moving from point $\boldsymbol{x}$ on the manifold along the direction $\xi \in T_{\boldsymbol{x}}\mathcal{M}$.
- We can approximate/replace the exponential map by **retraction**.
- **Retraction** is a mapping from the tangent space to a point on manifold:

$$\text{Ret}_{\boldsymbol{x}}(\xi) : \text{point } \boldsymbol{x} \text{ on } \mathcal{M} \times \text{tangent vector } \xi \in T_{\boldsymbol{x}}\mathcal{M} \rightarrow \text{point on } \mathcal{M}.$$

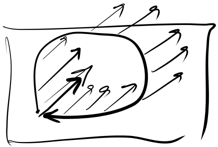
- You can see it as **projection** of a point in the tangent space onto the manifold.

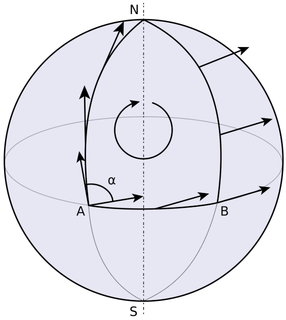


Credit of image: [6]

Riemannian concepts: parallel transport, Riemannian curvature

- **Parallel transport**: move/transport a tangent vector on the manifold in a way that it stays parallel with respect to the connection.
- Assume we do parallel transport on a tangent vector on the manifold and return back to the starting point. If the starting and ending tangent vectors do not match exactly, it means that the manifold has a curvature. This is the idea of **Riemannian curvature**.





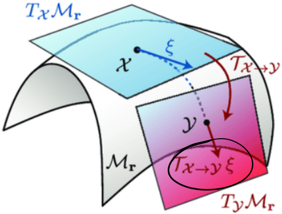
Credit of image: Wikipedia

Riemannian concepts: vector transport

- Parallel transport is hard to compute. We can approximate/replace the parallel transport by **vector transport**.
- **Vector transport** is a mapping from a tangent space to another tangent space on manifold:

$$\mathcal{T}_{\mathbf{x}_1,\mathbf{x}_2}(\xi) : T_{\mathbf{x}_1}\mathcal{M} \to T_{\mathbf{x}_2}\mathcal{M},$$

  where $\xi \in T_{\mathbf{x}_1}\mathcal{M}$ and $\mathcal{T}_{\mathbf{x}_1,\mathbf{x}_2}(\xi) \in T_{\mathbf{x}_2}\mathcal{M}$.
- You can see it as moving a tangent vector in a tangent space to the corresponding tangent vector in another tangent space.



Credit of image: [6]

First-order Riemannian Optimization

Riemannian Stochastic Gradient Descent

$x^{(k)} + \underbrace{(-\lambda \nabla f(x^{k}))}_{D}$

- Stochastic gradient descent in **Euclidean space**:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \lambda \nabla f(\boldsymbol{x}^{(k)}),}$$

where k is the iteration index and λ is the learning rate.

- Therefore:

$$\boxed{\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} := -\lambda \nabla f(\boldsymbol{x}^{(k)}),}$$

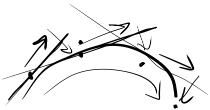
- Stochastic gradient descent in **Riemannian space** (2013) [7]:

addition

$$\boldsymbol{x}^{(k+1)} := \boxed{\mathrm{Exp}_{\boldsymbol{x}^{(k)}}\big(-\lambda \nabla f(\boldsymbol{x}^{(k)})\big),} \tag{7}$$

where ~~subtraction~~ in Euclidean space is generalized to the exponential map in Riemannian space.

- For simplicity, we can replace the exponential map with retraction:



$$\boxed{\boldsymbol{x}^{(k+1)} := \mathrm{Ret}_{\boldsymbol{x}^{(k)}}\big(-\lambda \nabla f(\boldsymbol{x}^{(k)})\big),} \tag{8}$$

Second-order Riemannian Optimization

Riemannian Newton's method

- Iterative optimization updates solution iteratively:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x},} \tag{9}$$

- Newton's method uses Hessian $\nabla^2 f(\boldsymbol{x})$ in its updating step:

$$\boxed{\Delta\boldsymbol{x} := -\nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}).} \tag{10}$$

- In the literature, this equation is sometimes restated to:

$$\boxed{\nabla^2 f(\boldsymbol{x})\, \Delta\boldsymbol{x} := -\nabla f(\boldsymbol{x}).} \tag{11}$$

- Recall **Riemannian Hessian**: the derivative of one of directions (ξ) in the tangent space (the derivative of derivative)

$$\boxed{\boldsymbol{B}\, f(\boldsymbol{x})\, \xi = \partial_{\xi} \nabla f(\boldsymbol{x}),}$$

  where $\boldsymbol{B}$ denotes the Hessian matrix and ∂_{ξ} is the affine connection.

- **Riemannian Newton's method** (compare Eqs. (11) and (12)):

$$\boxed{\boldsymbol{B}\, f(\boldsymbol{x})\, \xi := -\nabla f(\boldsymbol{x}).} \tag{12}$$

Quasi-Newton's method: Limited-memory BFGS (LBFGS)

- The quasi-Newton's method, including BFGS, approximate the inverse Hessian matrix by a dense ($d \times d$) matrix. For large d, storing this matrix is very memory-consuming.
- Hence, **Limited-memory BFGS (LBFGS)** [8, 9] was proposed, by Nocedal et al. in 1980's, which uses much less memory than BFGS.
- The LBFGS algorithm can be implemented as shown in the following algorithm [10] which is based on the algorithm in Nocedal's book [11, Chapter 6].

Initialize the solution $\boldsymbol{x}^{(0)}$
$\boldsymbol{H}^{(0)} := \frac{1}{\|\nabla f(\boldsymbol{x}^{(0)})\|_2}\boldsymbol{I}$
for $k = 0, 1, \dots$ *(until convergence)* **do**
  $\boldsymbol{p}^{(k)} \leftarrow \text{GetDirection}(-\nabla f(\boldsymbol{x}^{(k)}), k, 1)$
  $\eta^{(k)} \leftarrow$ Line-search with Wolfe conditions
  $\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)}\boldsymbol{p}^{(k)}$
  $\boldsymbol{s}^{(k)} := \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} = \eta^{(k)}\boldsymbol{p}^{(k)}$
  $\boldsymbol{y}^{(k)} := \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)})$
  $\gamma^{(k+1)} := \frac{\boldsymbol{s}^{(k)\top}\boldsymbol{y}^{(k)}}{\boldsymbol{y}^{(k)\top}\boldsymbol{y}^{(k)}}$
  $\boldsymbol{H}^{(k+1)} := \gamma^{(k+1)}\boldsymbol{I}$
  Store $\boldsymbol{y}^{(k)}$, $\boldsymbol{s}^{(k)}$, and $\boldsymbol{H}^{(k+1)}$
return $\boldsymbol{x}^{(k+1)}$

// recursive function:
Function GetDirection($\boldsymbol{p}, k, n\_recursion$)
if $k > 0$ **then**
  // do up to m recursions:
  if $n\_recursion > m$ **then**
    return $\boldsymbol{p}$
  $\rho^{(k-1)} := \frac{1}{\boldsymbol{y}^{(k-1)\top}\boldsymbol{s}^{(k-1)}}$
  $\tilde{\boldsymbol{p}} := \boldsymbol{p} - \rho^{(k-1)}(\boldsymbol{s}^{(k-1)\top}\boldsymbol{p})\boldsymbol{y}^{(k-1)}$
  $\widehat{\boldsymbol{p}} := \text{GetDirection}(\tilde{\boldsymbol{p}}, k-1, n\_recursion + 1)$
  return $\widehat{\boldsymbol{p}} - \rho^{(k-1)}(\boldsymbol{y}^{(k-1)\top}\widehat{\boldsymbol{p}})\boldsymbol{s}^{(k-1)} + \rho^{(k-1)}(\boldsymbol{s}^{(k-1)\top}\boldsymbol{s}^{(k-1)})\boldsymbol{p}$
else
  return $\boldsymbol{H}^{(0)}\boldsymbol{p}$

Riemannian LBFGS

Euclidean LBFGS (1980-1989) [8, 9], [11, Chapter 6]:

Initialize the solution $\boldsymbol{x}^{(0)}$

$\boldsymbol{H}^{(0)} := \frac{1}{\|\nabla f(\boldsymbol{x}^{(0)})\|_2}\boldsymbol{I}$

for $k = 0, 1, \dots$ *(until convergence)* **do**

- $\boldsymbol{p}^{(k)} \leftarrow \text{GetDirection}(-\nabla f(\boldsymbol{x}^{(k)}), k, 1)$
- $\eta^{(k)} \leftarrow$ Line-search with Wolfe conditions
- $\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)}\boldsymbol{p}^{(k)}$
- $\boldsymbol{s}^{(k)} := \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} = \eta^{(k)}\boldsymbol{p}^{(k)}$
- $\boldsymbol{y}^{(k)} := \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)})$
- $\gamma^{(k+1)} := \frac{\boldsymbol{s}^{(k)\top}\boldsymbol{y}^{(k)}}{\boldsymbol{y}^{(k)\top}\boldsymbol{y}^{(k)}}$
- $\boldsymbol{H}^{(k+1)} := \gamma^{(k+1)}\boldsymbol{I}$
- Store $\boldsymbol{y}^{(k)}$, $\boldsymbol{s}^{(k)}$, and $\boldsymbol{H}^{(k+1)}$

return $\boldsymbol{x}^{(k+1)}$

// recursive function:

Function GetDirection($\boldsymbol{p}, k, n\_\text{recursion}$)

if $k > 0$ **then**

- // do up to m recursions:
- **if** $n\_\text{recursion} > m$ **then**
  - **return** $\boldsymbol{p}$
- $\rho^{(k-1)} := \frac{1}{\boldsymbol{y}^{(k-1)\top}\boldsymbol{s}^{(k-1)}}$
- $\tilde{\boldsymbol{p}} := \boldsymbol{p} - \rho^{(k-1)}(\boldsymbol{s}^{(k-1)\top}\boldsymbol{p})\boldsymbol{y}^{(k-1)}$
- $\widehat{\boldsymbol{p}} := \text{GetDirection}(\tilde{\boldsymbol{p}}, k-1, n\_\text{recursion}+1)$
- **return** $\widehat{\boldsymbol{p}} - \rho^{(k-1)}(\boldsymbol{y}^{(k-1)\top}\widehat{\boldsymbol{p}})\boldsymbol{s}^{(k-1)} + \rho^{(k-1)}(\boldsymbol{s}^{(k-1)\top}\boldsymbol{s}^{(k-1)})\boldsymbol{p}$

else

- **return** $\boldsymbol{H}^{(0)}\boldsymbol{p}$

Riemannian LBFGS (2020) [10]:

Given: Riemannian manifold $\mathcal{M}$ with Riemannian metric g; vector transport $\mathcal{T}$ on $\mathcal{M}$; retraction Ret;
initial value x_0; a smooth function f
Set initial $H_{\text{diag}} = 1/\sqrt{g_{x_0}(\nabla f(x_0), \nabla f(x_0))}$
for $t = 0, 1, \dots$ **do**
  Obtain the descent direction $\xi_t \leftarrow \text{DESC}(-\nabla f(x_t), t)$
  Use line-search to find α such that it satisfies Wolfe conditions
  Calculate $x_{t+1} = \text{Ret}_{x_t}(\alpha\xi_t)$
  Define $s_{t+1} = \mathcal{T}_{x_t,x_{t+1}}(\alpha\xi_t)$
  Define $y_{t+1} = \nabla f(x_{t+1}) - \mathcal{T}_{x_t,x_{t+1}}(\nabla f(x_t))$
  Update $H_{\text{diag}} = g_{x_{t+1}}(s_{t+1}, y_{t+1})/g_{x_{t+1}}(y_{t+1}, y_{t+1})$
  Store y_{t+1}; s_{t+1}; $g_{x_{t+1}}(s_{t+1}, y_{t+1})$; $g_{x_{t+1}}(s_{t+1}, s_{t+1})$; H_{diag}
end for
return x_{t+1}

function DESC(p, t) //obtaining the descent direction by unrolling the BFGS method
if $t > 0$ **then**
  $\tilde{p} = p - \frac{g_{x_t}(s_t, p)}{g_{x_t}(y_t, s_t)} y_t$
  $\hat{p} = \mathcal{T}_{x_{t-1},x_t}\text{DESC}(\mathcal{T}^*_{x_{t-1},x_t}\tilde{p}, t-1)$
    // $\mathcal{T}^*_{x,y}$ is the adjoint of $\mathcal{T}_{x,y}$ [35] (defined by
    // $g_y(v, \mathcal{T}_{x,y}u) = g_x(u, \mathcal{T}^*_{x,y}v)\ \forall u \in T_x\mathcal{M}, v \in T_y\mathcal{M}$)
  return $\hat{p} - \frac{g_{x_t}(y_t, \hat{p})}{g_{x_t}(y_t, s_t)} s_t + \frac{g_{x_t}(s_t, s_t)}{g_{x_t}(y_t, s_t)} p$
else
  return $H_{\text{diag}}\, p$
end if
end function

Important Riemannian Matrix Manifolds

Important Riemannian Matrix Manifolds

$X^\top O$

$X^\top A$

- **Stiefel manifold** $\mathcal{S}t(p, d)$ is defined as the set of orthogonal matrices as:

$$\mathcal{M} = \mathcal{S}t(p, d) := \{\boldsymbol{X} \in \mathbb{R}^{d \times p} \,|\, \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}\}. \tag{13}$$

- The **quotient** of a vector space V by a subspace N is a vector space obtained by collapsing N to zero. The obtained space is called a **quotient space** and is denoted by V/N (read "V mod N" or "V by N").
- The **Grassmannian (Grassmann) manifold** $\mathcal{G}(p, d)$ can be seen as the quotient space of the Stiefel manifold $\mathcal{S}t(p, d)$ [1]:

$$\mathcal{M} = \mathcal{G}(p, d) := \mathcal{S}t(p, d)/\mathcal{S}t(p, p). \tag{14}$$

- The Grassmannian manifold $\mathcal{G}(p, d)$ is a space of all p-dimensional linear **subspaces** of the d-dimensional vector space. So, every element of this manifold can be the linear column-space of a projection matrix $\boldsymbol{X} \in \mathbb{R}^{d \times p}$ from a d-dimensional input space to a p-dimensional subspace, where $p \leq d$.
- Therefore, Grassmannian manifold can be used for **linear projection** in many machine learning methods, such as PCA, FDA, etc.

Important Riemannian Matrix Manifolds

S_+ : positive semi-definite

S_{++} : positive definite

- **Symmetric Positive Definite (SPD)** manifold $\mathbb{S}_{++}$ is defined as the set of SPD matrices as:

d-dim SPD $\longrightarrow$

$$\mathcal{M} = \mathbb{S}_{++} := \{\boldsymbol{X} \in \mathbb{R}^{d \times d} \,|\, \boldsymbol{X} \succ \mathbf{0}\}, \tag{15}$$

  where $\boldsymbol{X}$ is a **symmetric** matrix and all the eigenvalues of $\boldsymbol{X}$ are **positive** (neither negative nor zero).
- Examples:
  - Covariance matrix: $\boldsymbol{\Sigma}$
  - The weight matrix in quadratic functions: $\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x}$
  - The weight matrix in the generalized Mahalanobis distance: $(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{W} (\boldsymbol{x}_1 - \boldsymbol{x}_2)$

Toolboxes, Papers, and References

Important toolboxes for Riemannian optimization

- **Manopt** [12] (Matlab):
  https://github.com/NicolasBoumal/manopt
- **PyManopt** [13] (Python):
  https://github.com/pymanopt/pymanopt
- **StochMan** [14] (Python - stochastic manifolds):
  https://github.com/MachineLearningLifeScience/stochman
- **GeomStats** [15] (Python - machine learning):
  https://github.com/geomstats/geomstats
- **Geoopt** [16] (PyTorch):
  https://github.com/geoopt/geoopt
- **ROPTLIB** [17] (C++):
  https://github.com/whuang08/ROPTLIB
- **MixEst** [18] (Matlab - Riemannian LBFGS, mixture models using Riemannian optimization):
  https://github.com/utvisionlab/mixest

Important papers and books

- Papers with Codes page:
  `https://paperswithcode.com/task/riemannian-optimization`
- The books of **John M. Lee** on topology and manifolds:
  - "Introduction to Topological Manifolds" [3]
  - "Introduction to Smooth Manifolds" [5]
- Two very good books on Riemannian optimization:
  - "Optimization algorithms on matrix manifolds" by **Pierre-Antoine Absil** et al: [1]
  - "An introduction to optimization on smooth manifolds" by **Nicolas Boumal**: [2]
- Some papers:
  - A brief introduction to manifold optimization: (2020) [19]
  - Riemannian BFGS (RBFGS): (2010) [20]
  - Proving convergence of RBFGS: (2012, 2015) [21, 22]
  - Analyzing properties of RBFGS: (2013) [23]
  - As vector transport is computationally expensive in RBFGS, cautious RBFGS was proposed (2016) [24] which ignores the curvature condition in the Wolfe conditions (1969) [25] and only checks the Armijo condition (1966) [26]. Since the curvature condition guarantees that the approximation of Hessian remains positive definite, it compensates by checking a cautious condition (2001) [27] before updating the approximation of Hessian. This cautious RBFGS has been used in the Manopt optimization toolbox (2014) [12].
  - RLBFGS and SPD manifolds: (2015, 2016, 2020) [28, 29, 10].
  - Some other direct extensions of Euclidean BFGS to Riemannian spaces: (2007) [30, Chapter 7]
  - Vector-transport free RLBFGS: (2021) [31]

Important scholars in the field

Some important scientists in the field of Riemannian optimization (not limited to the following):

- **Pierre-Antoine Absil**, UCLouvain, Belgium (Author of book [1], proposer of Manopt toolbox)
- **Rodolphe Sepulchre**, KU Leuven, Belgium (Coauthor of Absil in book [1])
- **Robert Mahony**, Australian National University, Australia (Coauthor of Absil in book [1])
- **Nicolas Boumal**, EPFL, Switzerland (Author of book [2], proposer of Manopt toolbox)
- **Silvere Bonnabel**, Mines Paris PSL, France (proposed Riemannian stochastic gradient descent [7])
- **Ring Wolfgang**, Karl-Franzens-Universitat Graz, Austria (proof of convergence of RBFGS)
- **Bart Vandereycken**, University of Geneva, Switzerland (proposer of low-rank matrix completion by Riemannian optimization [32])
- **Suvrit Sra**, MIT, USA (optimization and Riemannian optimization)
- **Reshad Hosseini**, University of Tehran, Iran (Mixest toolbox, SPD manifolds, mixture models using Riemannian optimization [10])
- **Mehrtash T. Harandi**, Monash University, Australia (machine learning using Riemannian optimization)
- **Soren Hauberg**, Technical University of Denmark, Denmark (StochMan toolbox, machine learning using Riemannian optimization)
- I also thank my friend, **Reza Godaz** (see our paper together [31]), who introduced this field to me.

Acknowledgement

- The slides of this slide deck are inspired by the teachings of Prof. **Reshad Hosseini** at the University of Tehran. He also gave a virtual talk about Riemannian optimization, entitled "Manifold optimization in data analytics", at the "Sharif Optimization and Application laboratory" in Department of Mathematics at Sharif University of Technology.
- Some slides of this slide deck are inspired by the presentation of Prof. **Soren Hauberg** at the Asian Conference on Machine Learning (ACML) 2021, entitled "Differential Geometry in Generative Modeling".
- Some of the concepts on topology and smooth manifolds in this lecture are inspired by the teachings of Prof. **Spiro Karigiannis** at the Department of Pure Mathematics in the University of Waterloo (his course "Smooth manifolds").
- I thank my friend, **Reza Godaz**, and Prof. **Reshad Hosseini** (see our paper together [31]), who introduced this field to me.
- Our tutorial [33] also has briefly introduced what Riemannian optimization is.

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] N. Boumal, *An introduction to optimization on smooth manifolds*. Available online, 2020.

[3] J. M. Lee, *Introduction to topological manifolds*. 1 Springer Science & Business Media, 2010.

[4] J. L. Kelley, *General topology*. Courier Dover Publications, 2017.

[5] J. M. Lee, *Introduction to Smooth Manifolds*. 2 Springer Science & Business Media, 2013.

[6] D. Kressner, M. Steinlechner, and B. Vandereycken, "Low-rank tensor completion by Riemannian optimization," *BIT Numerical Mathematics*, vol. 54, pp. 447–468, 2014.

[7] S. Bonnabel, "Stochastic gradient descent on Riemannian manifolds," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2217–2229, 2013.

[8] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[9] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

References (cont.)

[10] R. Hosseini and S. Sra, "An alternative to EM for Gaussian mixture models: batch and stochastic Riemannian optimization," *Mathematical Programming*, vol. 181, no. 1, pp. 187–223, 2020.

[11] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2 ed., 2006.

[12] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1455–1459, 2014.

[13] J. Townsend, N. Koep, and S. Weichwald, "Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation," *arXiv preprint arXiv:1603.03236*, 2016.

[14] N. S. Detlefsen, A. Pouplin, C. W. Feldager, C. Geng, D. Kalatzis, H. Hauschultz, M. GonzÃ¡lez-Duque, F. Warburg, M. Miani, and S. Hauberg, "Stochman," *GitHub. Note: https://github.com/MachineLearningLifeScience/stochman/*, 2021.

[15] N. Miolane, N. Guigui, A. Le Brigant, J. Mathe, B. Hou, Y. Thanwerdas, S. Heyder, O. Peltre, N. Koep, H. Zaatiti, *et al.*, "Geomstats: a Python package for Riemannian geometry in machine learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 9203–9211, 2020.

[16] M. Kochurov, R. Karimov, and S. Kozlukov, "Geoopt: Riemannian optimization in PyTorch," *arXiv preprint arXiv:2005.02819*, 2020.

References (cont.)

[17] W. Huang, P. Absil, K. Gallivan, and P. Hand, "ROPTLIB: Riemannian manifold optimization library," 2017.

[18] R. Hosseini and M. Mash'al, "Mixest: An estimation toolbox for mixture models," *arXiv preprint arXiv:1507.06065*, 2015.

[19] 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*, vol. 8, no. 2, pp. 199–248, 2020.

[20] C. Qi, K. A. Gallivan, and P.-A. Absil, "Riemannian BFGS algorithm with applications," in *Recent advances in optimization and its applications in engineering*, pp. 183–192, Springer, 2010.

[21] W. Ring and B. Wirth, "Optimization methods on Riemannian manifolds and their application to shape space," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 596–627, 2012.

[22] W. Huang, K. A. Gallivan, and P.-A. Absil, "A Broyden class of quasi-Newton methods for Riemannian optimization," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1660–1685, 2015.

[23] M. Seibert, M. Kleinsteuber, and K. Hüper, "Properties of the BFGS method on Riemannian manifolds," *Mathematical System Theory C Festschrift in Honor of Uwe Helmke on the Occasion of his Sixtieth Birthday*, pp. 395–412, 2013.

References (cont.)

[24] W. Huang, P.-A. Absil, and K. A. Gallivan, "A Riemannian BFGS method for nonconvex optimization problems," in *Numerical Mathematics and Advanced Applications ENUMATH 2015*, pp. 627–634, Springer, 2016.

[25] P. Wolfe, "Convergence conditions for ascent methods," *SIAM Review*, vol. 11, no. 2, pp. 226–235, 1969.

[26] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[27] D.-H. Li and M. Fukushima, "On the global convergence of the BFGS method for nonconvex unconstrained optimization problems," *SIAM Journal on Optimization*, vol. 11, no. 4, pp. 1054–1064, 2001.

[28] S. Sra and R. Hosseini, "Conic geometric optimization on the manifold of positive definite matrices," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 713–739, 2015.

[29] S. Sra and R. Hosseini, "Geometric optimization in machine learning," in *Algorithmic Advances in Riemannian Geometry and Applications*, pp. 73–91, Springer, 2016.

[30] H. Ji, *Optimization approaches on smooth manifolds*.
PhD thesis, Australian National University, 2007.

[31] R. Godaz, B. Ghojogh, R. Hosseini, R. Monsefi, F. Karray, and M. Crowley, "Vector transport free Riemannian LBFGS for optimization on symmetric positive definite matrix manifolds," in *Asian Conference on Machine Learning*, pp. 1–16, PMLR, 2021.

References (cont.)

[32] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[33] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "KKT conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey," *arXiv preprint arXiv:2110.01858*, 2021.