

Manopt.jl

Optimization on Riemannian Manifolds in Julia

Ronny Bergmann

Julia for numerical problems in quantum and solid-state physics

CECAM workshop, Lausanne,

November 27, 2024.

Optimization on Manifolds

$$\arg \min_{p \in \mathcal{M}} f(p)$$

- ▶ $f: \mathcal{M} \rightarrow \mathbb{R}$ is a (smooth) function
- ▶ \mathcal{M} is a Riemannian manifold
- ➔ Riemannian optimization




This especially includes

- ▶ nonsmooth problems: f is (only) lower semicontinuous
- ➔ splitting methods $f(p) = g(p) + h(p)$, where g is smooth
- ▶ Difference of Convex problems $f(p) = g(p) - h(p)$
- ▶ constraints $p \in \mathcal{C} \subset \mathcal{M}$

The Rayleigh Quotient


When minimizing the **Rayleigh quotient** for a symmetric $A \in \mathbb{R}^{n \times n}$


$$\arg \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T A x}{\|x\|^2}$$

-  Any eigenvector x^* to the smallest EV λ is a minimizer
-  no isolated minima **and** Newton's method diverges
-  Constrain the problem to unit vectors $\|x\| = 1!$

classic constrained optimization (ALM, EPM, IP Newton, ...)

Today Utilize the geometry of the sphere

 unconstrained optimization $\arg \min_{p \in \mathbb{S}^{n-1}} p^T A p$


 adapt unconstrained optimization to **Riemannian manifolds**.


The Generalized Rayleigh Quotient

More general. Find a basis for the space of eigenvectors to $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$:


$$\arg \min_{X \in \text{St}(n,k)} \text{tr}(X^T A X), \quad \text{St}(n,k) := \{X \in \mathbb{R}^{n \times k} \mid X^T X = I\},$$

 a problem on the **Stiefel** manifold $\text{St}(n,k)$

 Invariant under rotations within a k -dim subspace.

 Find the best subspace!

$$\arg \min_{\text{span}(X) \in \text{Gr}(n,k)} \text{tr}(X^T A X), \quad \text{Gr}(n,k) := \{\text{span}(X) \mid X \in \text{St}(n,k)\},$$

 a problem on the **Grassmann** manifold $\text{Gr}(n,k) = \text{St}(n,k)/O(k)$.

A Riemannian Manifold \mathcal{M}

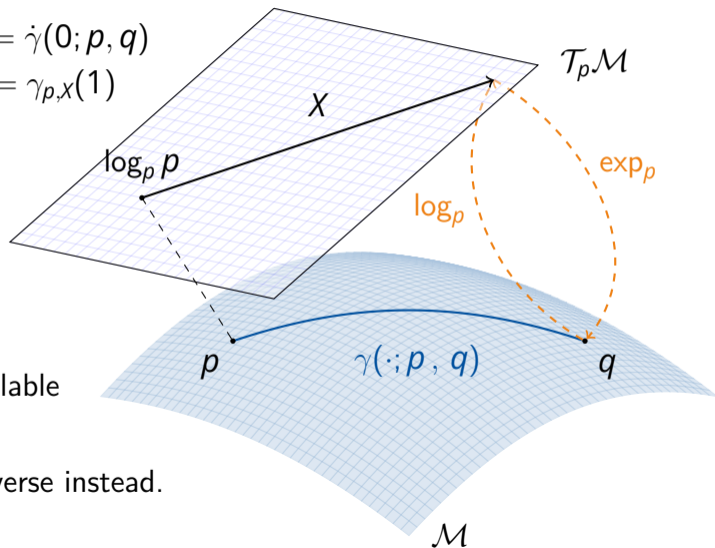
A d -dimensional Riemannian manifold can be informally defined as a set \mathcal{M} covered with a “suitable” collection of charts, that identify subsets of \mathcal{M} with open subsets of \mathbb{R}^d and a continuously varying inner product on the tangent spaces.

[Absil, Mahony, Sepulchre, 2008]

A Riemannian Manifold \mathcal{M}

Notation.

- ▶ Logarithmic map $\log_p q = \dot{\gamma}(0; p, q)$
- ▶ Exponential map $\exp_p X = \gamma_{p,X}(1)$
- ▶ Geodesic $\gamma(\cdot; p, q)$
- ▶ Tangent space $\mathcal{T}_p\mathcal{M}$
- ▶ inner product $(\cdot, \cdot)_p$



Numerics.

\exp_p and \log_p maybe not available efficiently/ in closed form

\Rightarrow use a retraction and its inverse instead.

(Geodesic) Convexity

[Sakai, 1996; Udriște, 1994]

A set $\mathcal{C} \subset \mathcal{M}$ is called (strongly geodesically) **convex** if for all $p, q \in \mathcal{C}$ the geodesic $\gamma(\cdot; p, q)$ is unique and lies in \mathcal{C} .

A function $f: \mathcal{C} \rightarrow \overline{\mathbb{R}}$ is called (geodesically) **convex** if for all $p, q \in \mathcal{C}$ the composition $f(\gamma(t; p, q)), t \in [0, 1]$, is convex.



[Axen, Baran, RB, Rzecki, 2023]

Goal. Provide an interface to implement and use Riemannian manifolds.

Interface `AbstractManifold` to model manifolds

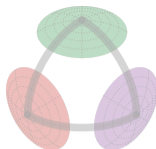
Functions like `exp(M, p, X)`, `log(M, p, X)` or `retract(M, p, X, method)`.

Decorators for implicit or explicit specification of an embedding, a metric, or a group,

Efficiency by providing in-place variants like `exp!(M, q, p, X)`

Manifolds.jl

Goal. Provide a library of Riemannian manifolds, that is efficiently implemented and well-documented



[Axen, Baran, RB, Rzecki, 2023]

Meta. generic implementations for $\mathcal{M}^{n \times m}$, $\mathcal{M}_1 \times \mathcal{M}_2$, vector- and tangent-bundles, esp. $T_p\mathcal{M}$, or Lie groups

Library. Implemented functions for

- ▶ Circle, Sphere, Torus, Hyperbolic, Projective Spaces, Hamiltonian
- ▶ (generalized, symplectic) Stiefel, Rotations
- ▶ (generalized, symplectic) Grassmann, fixed rank matrices
- ▶ Symmetric Positive Definite matrices, with fixed determinant
- ▶ (several) Multinomial, (skew-)symmetric, and symplectic matrices
- ▶ Tucker & Oblique manifold, Kendall's Shape space
- ▶ probability simplex, orthogonal and unitary matrices, ...

Concrete Manifold Examples.

Before first run] `add Manifolds` to install the package.

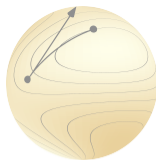
Load packages with `using Manifolds` and

- ▶ Euclidean space $M1 = \mathbb{R}^3$ and 2-sphere $M2 = \text{Sphere}(2)$
- ▶ their product manifold $M3 = M1 \times M2$
- ▶ A signal of rotations $M4 = \text{SpecialOrthogonal}(3)^{10}$
- ▶ SPDs $M5 = \text{SymmetricPositiveDefinite}(3)$ (affine invariant metric)
- ▶ a different metric $M6 = \text{MetricManifold}(M5, \text{LogCholeskyMetric}())$

Then for *any* of these

- ▶ Generate a point $p = \text{rand}(M)$ and a vector $X = \text{rand}(M; \text{vector_at}=p)$
- ▶ and for example `exp(M, p, X)`, or in-place `exp!(M, q, p, X)`

Manopt.jl



Goal. Provide optimization algorithms on Riemannian manifolds.

Features. Given a `Problem p` and a `SolverState s`, implement `initialize_solver!(p, s)` and `step_solver!(p, s, i)` \Rightarrow an algorithm in the `Manopt.jl` interface

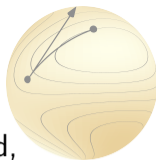
Highlevel interfaces like `gradient_descent(M, f, grad_f)` on any manifold `M` from `Manifolds.jl`.

All provide `debug` output, `recording`, `cache` & `counting` capabilities, as well as a library of `step sizes` and `stopping criteria`.

Manopt family.



List of Algorithms in Manopt.jl



Derivative Free Nelder-Mead, Particle Swarm, CMA-ES

Subgradient-based Subgradient Method, Convex Bundle Method,
Proximal Bundle Method

Gradient-based Gradient Descent, Conjugate Gradient, Stochastic,
Momentum, Nesterov, Averaged, ...
Quasi-Newton with (L-)BFGS, DFP, Broyden, SR1,...
Levenberg-Marquard

Hessian-based Trust Regions, Adaptive Regularized Cubics (ARC)

nonsmooth Chambolle-Pock, Douglas-Rachford, Cyclic Proximal Point

constrained Augmented Lagrangian, Exact Penalty, Frank-Wolfe,
Interior Point Newton

nonconvex Difference of Convex Algorithm, DCPA

Gradient Descent

For the Rayleigh quotient on \mathbb{S}^{n-1} we have for $p \in \mathbb{S}^{n-1}$

$$\text{cost } f(p) = p^T A p, \quad \text{and gradient } \nabla f(p) = 2Ap.$$

But this is not the Riemannian one. For example: $\nabla f(p) \notin T_p \mathcal{M}$.
Formally: We need the Riesz representer $Df(p)[X] = \langle \text{grad } f(p), X \rangle_p$.

Easier: Let `Manopt.jl` convert the Euclidean into a Riemannian gradient:

```
using Manopt, Manifolds
M = Sphere(2); A = Matrix(reshape(1.0:9.0, 3, 3));
f(M,p) = p'*A*p;
∇f(M,p) = 2A*p;
p0 = [1.0, 0.0, 0.0];
q = gradient_descent(M, f, ∇f, p0; objective_type=:Euclidean)
```

Works as well if you have a Hessian $\nabla^2 f$ is required.

Illustrating a few Keyword Arguments

Given a manifold M , a cost $f(M,p)$, its Riemannian gradient $\text{grad}_f(M,p)$, and a start point p_0 .

- ▶ `q = gradient_descent(M, f, grad_f, p0)` to perform gradient descent
- ▶ With Euclidean cost $f(E,p)$ and gradient $\nabla f(E, p)$, use for conversion
`q = gradient_descent(M, f, ∇f , p0; objective_type=:Euclidean)`
- ▶ print iteration number, cost and change every 10th iterate

```
q = gradient_descent(M, f, grad_f, p0;  
                    debug=[:Iteration, :Cost, :Change, 10, "\n"]  
                    )
```
- ▶ record `record=[:Iterate, :Cost, :Change]`, `return_state=true`
Access: `get_solver_result(q)` and `get_record(q)`
- ▶ modify stop: `stopping_criterion = StopAfterIteration(100)`
- ▶ cache calls `cache=(:LRU, [:Cost, :Gradient], 25)` (uses `LRUCache.jl`)
- ▶ count calls `count=[:Cost, :Gradient]` (prints with `return_state=true`)

The Riemannian Subdifferential

Let \mathcal{C} be a convex set.

The **subdifferential** of f at $p \in \mathcal{C}$ is given by [Ferreira, Oliveira, 2002; Lee, 2003; Udriște, 1994]

$$\partial_{\mathcal{M}}f(p) := \{\xi \in \mathcal{T}_p^*\mathcal{M} \mid f(q) \geq f(p) + \langle \xi, \log_p q \rangle_p \text{ for } q \in \mathcal{C}\},$$

where

- ▶ $\mathcal{T}_p^*\mathcal{M}$ is the dual space of $\mathcal{T}_p\mathcal{M}$, also called **cotangent space**
- ▶ $\langle \cdot, \cdot \rangle_p$ denotes the duality pairing on $\mathcal{T}_p^*\mathcal{M} \times \mathcal{T}_p\mathcal{M}$
- ▶ numerically we use musical isomorphisms $X = \xi^\flat \in \mathcal{T}_p\mathcal{M}$ to obtain a subset of $\mathcal{T}_p\mathcal{M}$

The Riemannian DCA in Manopt.jl

[RB, Ferreira, Santos, Souza, 2024]

To solve a problem of a difference of convex (DC) functions

$$\arg \min_{p \in \mathcal{M}} f(p), \quad f(p) = g(p) - h(p),$$

where g is convex and smooth and h is convex but not necessarily smooth:

```
q = difference_of_convex_algorithm(M, f, g, ∂h, p0; kwargs...)
```

Input: An initial point $p^{(0)} \in \text{dom}(g)$, g and $\partial_{\mathcal{M}}h$

1: **for** $k = 1, 2, \dots$ until convergence **do**

2: Take $X^{(k)} \in \partial_{\mathcal{M}}h(p^{(k)})$

3: Compute $p^{(k+1)} \in \arg \min_{p \in \mathcal{M}} g(p) - (X^{(k)}, \log_{p^{(k)}} p)_{p^{(k)}}$.

4: **end for**

➔ implement $f(M, p)$, $g(M, p)$, and $\partial h(M, p)$.

▶ efficient sub solver used if `grad_g=` is set (implement `grad_g(M, p)`)

▶ `sub_state=` to specify a solver or `sub_problem=` for closed-form solution

Summary


`Manifolds.jl` & `Manifolds.jl`

- ▶ a high-level interface to define and use Riemannian manifolds
- ▶ a library of manifolds and functions defined thereon

`Manopt.jl`

- ▶ an interface to define solvers
- ▶ a library of algorithms for optimization on manifolds
- ▶ several tools to
 - ▶ state stopping criteria, debug, record, caching
 - ▶ (re)use Euclidean gradients and Hessians \Rightarrow `ManifoldDiff.jl`

Future work.

- ▶ What is a Fenchel conjugate on Manifolds?
 Friday 10.15 in the MathICSE seminar.
- ▶ `GroupManifolds` are currently reworked \Rightarrow `LieGroups.jl`
- ▶ Solve differential equations on manifolds \Rightarrow `ManifoldDiffEq.jl`

Selected References



Absil, P.-A.; R. Mahony; R. Sepulchre (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press. DOI: [10.1515/9781400830244](https://doi.org/10.1515/9781400830244).



Axen, S. D.; M. Baran; RB; K. Rzecki (2023). "Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds". *ACM Transactions on Mathematical Software*. Accepted for publication. DOI: [10.1145/3618296](https://doi.org/10.1145/3618296). arXiv: [2106.08777](https://arxiv.org/abs/2106.08777).



RB (2022). "Manopt.jl: Optimization on Manifolds in Julia". *Journal of Open Source Software* 7.70, p. 3866. DOI: [10.21105/joss.03866](https://doi.org/10.21105/joss.03866).



RB; O. P. Ferreira; E. M. Santos; J. C. d. O. Souza (2024). "The difference of convex algorithm on Hadamard manifolds". *Journal of Optimization Theory and Applications*. DOI: [10.1007/s10957-024-02392-8](https://doi.org/10.1007/s10957-024-02392-8). arXiv: [2112.05250](https://arxiv.org/abs/2112.05250).



Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press. URL: <https://www.nicolasboumal.net/book>.

Interested in Numerical Differential Geometry?

Join  [numdiffgeo.zulipchat.com!](https://numdiffgeo.zulipchat.com)



ronnybergmann.net/talks/2024-Lausanne-Manopt.pdf