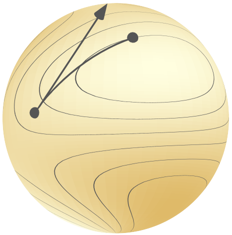


Manopt.jl

Optimization on Riemannian Manifolds

Ronny Bergmann
@ronnybergmann_

Norwegian University of Science and Technology, Trondheim, Norway.

Lightning Talk
JuliaCon 2022, online and everywhere

July 27nd, 2022

Norwegian University of Science and Technology

NTNU

Optimization

(Constrained) Optimization aims to find for a function $f\colon \mathbb{R}^m \to \mathbb{R}$ a point

$$\operatorname*{arg\,min}_{x\in\mathbb{R}^m} f(x)$$

Challenges:

- constrained to some $\mathcal{C} \subset \mathbb{R}^m$, e. g. unit vectors
- symmetries / invariances

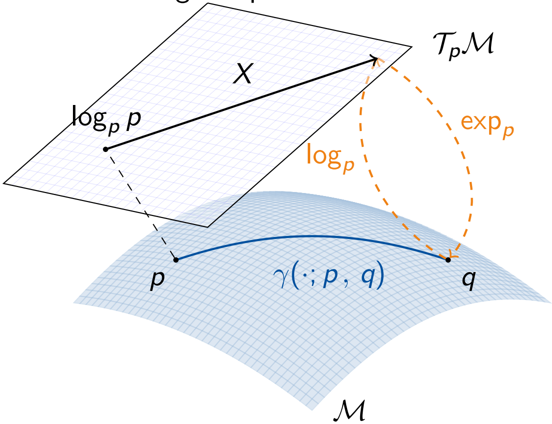
Geometric Optimization aims to find

$$\operatorname*{arg\,min}_{p\in\mathcal{M}} F(p)$$

where F is defined on a Riemannian manifold $\mathcal{M}$, e. g. the sphere $\mathbb{S}^d \subset \mathbb{R}^{d+1}$.
$\Rightarrow$ the problem is unconstrained (again).

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, and Sepulchre 2008]



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 $(\cdot, \cdot)_p$

ManifoldsBase.jl & Manifolds.jl

`ManifoldsBase.jl` is an interface for Riemannian manifolds `M`

- `inner(M, p, X, Y)` $(X, Y)_p$
- `exp(M, p, X)` and `log(M, p, q)`,
- more general:
  `retract(M, p, X, m)`,
  where `m` is a retraction method
- embeddings as decorator
- ☺ mutating variants, e. g.
  `exp!(M, q, p, X)`
  works in place of `q`

🔗 juliamanifolds.github.io/ManifoldsBase.jl/
🔗 juliamanifolds.github.io/Manifolds.jl/

`Manifolds.jl` is a Library of manifolds

- Circle, (unit) Sphere & Torus
- Fixed Rank Matrices
- (Symplectic) Stiefel & Grassmann
- Hyperbolic space & Rotations
- Symmetric positive definite matrices
- ...and many more

as well as generically

- power & product manifold
- tangent & vector bundles
- Lie groups, connections, metrics,...

[Axen, Baran, RB, and Rzecki 2021]

▶ JuliaCon 2020 youtu.be/md-FnDGCh9M

Manopt.jl – Structure

`Manopt.jl` depends only on `ManifoldsBase.jl` and consists of

- a `Problem P` to specify static properties:
  the manifold $\mathcal{M}$, the cost F, its (Riemannian) gradient grad F, …
- some `Options O` to specify a solver and containing dynamic data:
  the current iterate, the current gradient, a stopping criterion, …
- implement
  1. `initialize_solver!(P, O)` to initialise a solver run
  2. `step_solver!(P, O, i)` to perform the ith step

⇒ call `solve(P,O)` to run the solver or use a high-level interface

Furthermore one can

- specify a `Stepsize s`, that is for example a `Linesearch l`
- include a `StoppingCriterion sc`, a functor `sc(P, O, i)` returning true/false

☺ `sc1 | sc2` and `sc1 & sc2` to build more advanced criteria

- `DebugOptions` and `RecordOptions` decorate Options with print/record

Manopt.jl – Available Solvers

Currently the following solvers are available

- Gradient Descent
  CG, Stochastic, Momentum, Alternating, Average, Nesterov, ...
- Quasi-Newton
  (L-)BFGS, DFP, Broyden, SR1, ...
- Nelder-Mead, Particle Swarm
- Subgradient Method
- Trust Regions
- Chambolle-Pock (PDHG)
- Douglas-Rachford
- Cyclic Proximal Point

The Manopt Family.

manoptjl.org [RB 2022]

manopt.org [Boumal, Mishra, Absil, and Sepulchre 2014]

pymanopt.org [Townsend, Koep, and Weichwald 2016]

Example – A Riemannian Center of Mass

The mean of N data points $x_1, \ldots, x_N \in \mathbb{R}^n$ is

$$x^* = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \Leftrightarrow \quad x^* = \operatorname*{arg\,min}_{x \in \mathbb{R}^m} \frac{1}{2N}\sum_{i=1}^{N} \lVert x - x_i \rVert_2^2$$

$\Rightarrow$ the minimizer of sum of squared distances

For $p_1, \ldots, p_N \in \mathcal{M}$: [Karcher 1977]
Riemannian center(s) of mass are

$$\operatorname*{arg\,min}_{p \in \mathcal{M}} \frac{1}{2N}\sum_{i=1}^{N} d_{\mathcal{M}}^2(p, p_i),$$

- (in general) neither closed form nor unique
- For $F(p) = \frac{1}{2} d_{\mathcal{M}}^2(p, p_i)$ we have $\operatorname{grad} F(p) = -\log_p p_i$

$\Rightarrow$ use gradient descent

```
using LinearAlgebra
using Manopt, Manifolds
M = Sphere(2)
N = 100
pts = [randn(3) for _ in 1:N]
pts ./= norm.(pts)
F(M, p) = sum(
  pi -> distance(M,pi,p)^2/2N,
  pts,
)
gF(M, p) = sum(
  pi -> grad_distance(M,pi,p)/N,
  pts,
)
# compute a center of mass
# in place of m
m = copy(M, pts[1])
gradient_descent!(M, F, gF, m)
```

References

- Absil, P.-A., R. Mahony, and R. Sepulchre (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press. DOI: 10.1515/9781400830244.
- Axen, S. D., M. Baran, RB, and K. Rzecki (2021). *Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds*. arXiv: 2106.08777.
- RB (2022). "Manopt.jl: Optimization on Manifolds in Julia". In: *Journal of Open Source Software* 7.70, p. 3866. DOI: 10.21105/joss.03866.
- Boumal, N., B. Mishra, P.-A. Absil, and R. Sepulchre (2014). "Manopt, a Matlab toolbox for optimization on manifolds". In: *The Journal of Machine Learning Research* 15, pp. 1455–1459. URL: https://www.jmlr.org/papers/v15/boumal14a.html.
- Karcher, H. (1977). "Riemannian center of mass and mollifier smoothing". In: *Communications on Pure and Applied Mathematics* 30.5, pp. 509–541. DOI: 10.1002/cpa.3160300502.
- Townsend, J., N. Koep, and S. Weichwald (2016). "Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation". In: *Journal of Machine Learning Research* 17.137, pp. 1–5. URL: http://jmlr.org/papers/v17/16-177.html.

ronnybergmann.net/talks/2022-JuliaCon-Manoptjl.pdf