



NTNU

Norwegian University of Science and Technology

The Difference of Convex Algorithm on Riemannian Manifolds

Ronny Bergmann

joint work with

O. P. Ferreira, E. M. Santos, and J. C. O. Souza.

Manifolds and Geometric Integration Colloquia, Ilsetra,

March 1, 2023

Introduction

Task. We aim to solve

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

where

- ▶ \mathcal{M} is a **Riemannian manifold**, for this talk even Hadamard
- ▶ $f: \mathcal{M} \rightarrow \mathbb{R}$ is (today) a **difference of convex** function, i. e. of the form

$$f(p) = g(p) - h(p)$$

- ▶ $g, h: \mathcal{M} \rightarrow \overline{\mathbb{R}}$ are **convex, lsc., and proper**

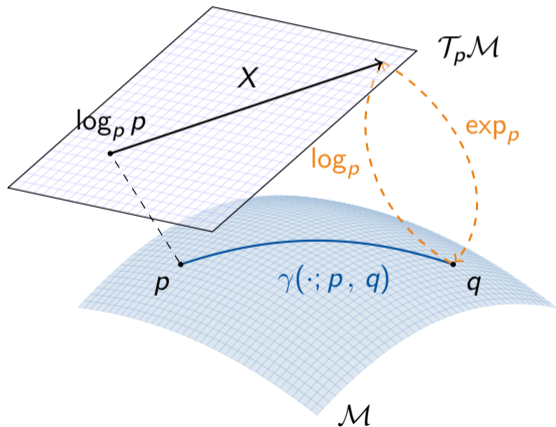
A Riemannian Manifold \mathcal{M}

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 $(\cdot, \cdot)_p$



(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.

The Riemannian Subdifferential

The **subdifferential** of f at $p \in \mathcal{C}$ is given by

[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 $\mathcal{T}_p \mathcal{M}$,
- ▶ $\langle \cdot, \cdot \rangle_p$ denotes the duality pairing on $\mathcal{T}_p^* \mathcal{M} \times \mathcal{T}_p \mathcal{M}$

Fenchel Duality on a Hadamard Manifold

[Silva Louzeiro, RB, and Herzog 2022]

Definition

Let $f: \mathcal{M} \rightarrow \overline{\mathbb{R}}$. The Fenchel conjugate of f is the function $f^*: \mathcal{T}^*\mathcal{M} \rightarrow \overline{\mathbb{R}}$ defined by

$$f^*(p, X) := \sup_{q \in \mathcal{M}} \left\{ \langle X, \log_p q \rangle - f(q) \right\}, \quad (p, X) \in \mathcal{T}^*\mathcal{M}.$$

[RB, Ferreira, Santos, and J. C. O. Souza 2023]

Theorem.

$$\inf_{(q, X) \in \mathcal{T}^*\mathcal{M}} \left\{ h^*(q, X) - g^*(q, X) \right\} = \inf_{p \in \mathcal{M}} \{g(p) - h(p)\}.$$

The Euclidean DCA

Idea 1. At x_k , approximate the the second DC component $h(x)$ by its affine minorization $h_k(x) := h(x^k) + \langle x - x_k, y_k \rangle$ for some $y_k \in \partial h(x^k)$.

\Rightarrow minimize $g(x) - h_k(x) = g(x) + h(x_k) - \langle x - x_k, y_k \rangle$ instead.

Idea 2. Using duality theory finding a new $y_k \in \partial h(x_k)$ is equivalent to

$$\arg \min_{y \in \mathbb{R}^n} \left\{ h^*(y) - g^*(y_{k-1}) - \langle y - y_{k-1}, x_k \rangle \right\}$$

Idea 3. Formulates the second idea in terms of proximal maps \Rightarrow PPA

\Rightarrow Yields a PPA, on manifolds:

[Almeida, Neto, Oliveira, and J. C. d. O. Souza 2020; J. C. d. O. Souza and Oliveira 2015]

All yield equivalent algorithms on manifolds, here we focus on the first idea

Derivation of the Riemannian DCA

We consider the linearization of h at some point p_k .

For an $\xi \in \partial h(p_k)$ we consider

$$h_k(p) = h(p_k) + \langle \xi, \log_{p_k} p \rangle_{p_k}$$

We use the **musical isomorphisms** to identify $X = \xi^\sharp \in T_{p_k}\mathcal{M}$.

Since X is a subgradient we have that h_k locally minorizes h , i.e.

$$h_k(q) \leq h(q) \text{ locally around } p_k$$

\Rightarrow Use as **upper bound** for $-h(p)$ in f .

Note. While on \mathbb{R}^n the function is linear, this is not necessarily **convex** on manifolds, not even Hadamard ones.

The Riemannian DC Algorithm

[RB, Ferreira, Santos, and J. C. O. Souza 2023]

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

- 1: Set $k = 0$.
- 2: **while** not converged **do**
- 3: Take $X_k \in \partial_{\mathcal{M}}h(p_k)$
- 4: Compute the next iterate p^{k+1} as

$$p_{k+1} \in \arg \min_{p \in \mathcal{M}} \left(g(p) - (X_k, \log_{p_k} p)_{p_k} \right).$$

- 5: Set $k \leftarrow k + 1$
- 6: **end while**

Note. In general the subproblem can not be solved in closed form, but even approximately (a few steps gradient descent) yields a good candidate.

Convergence

[RB, Ferreira, Santos, and J. C. O. Souza 2023]

Proposition. Let $\{p_k\}_{k \in \mathbb{N}}$ be generated by the Riemannian DCA and g be σ -strongly (geodesically) convex. Then, the following inequality holds

$$f(p_{k+1}) \leq f(p_k) - \frac{\sigma}{2} d^2(p_k, p_{k+1}).$$

Moreover, if $p_{k+1} = p_k$, then p_k is a critical point of f .

Proposition. Let $\{p_k\}_{k \in \mathbb{N}}$ be generated by the Riemannian DCA. Then,

$$\sum_{k=0}^{+\infty} d^2(p_k, p_{k+1}) < \infty.$$

In particular, $\lim_{k \rightarrow \infty} d(p_k, p_{k+1}) = 0$.

Theorem. Let $\{p_k\}_{k \in \mathbb{N}}$ and $\{X_k\}_{k \in \mathbb{N}}$ be generated by the Riemannian DCA. If \bar{p} is a cluster point of $\{p_k\}_{k \in \mathbb{N}}$, then $\bar{p} \in \text{dom}(g)$ and there exists a cluster point \bar{X} of $\{X_k\}_{k \in \mathbb{N}}$ s. t. $\bar{X} \in \partial g(\bar{p}) \cap \partial h(\bar{p})$.
 \Rightarrow Every cluster point of $\{p_k\}_{k \in \mathbb{N}}$, if any, is a critical point of f .

ManifoldsBase.jl & Manifolds.jl

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

- ▶ `inner(M, p, X, Y)` for $(X, Y)_p$
- ▶ `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

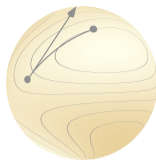
- ▶ Circle, (unit) Sphere & Torus
- ▶ Fixed Rank Matrices
- ▶ (Symplectic) Stiefel & Grassmann
- ▶ Hyperbolic space & Rotations
- ▶ Symmetric positive definite matrices
- ▶ ...and many more

as well as generically

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



Manopt.jl: Optimisation on Manifolds in Julia






Goal. Optimisation algorithms on [Riemannian manifolds](#), based on [ManifoldsBase.jl](#) \Rightarrow works with any manifold from [Manifolds.jl](#).

Features.

- ▶ generic algorithm framework:
With `Problem p` and a `SolverState s`
 - ▶ `initialize_solver!(p, s)`
 - ▶ `step_solver!(p, s, i)`: *i*th step
- ⊕ run algorithm: call `solve(p, s)`
- ▶ generic debug and recording
- ▶ step sizes and stopping criteria.

Manopt Family.

-  manoptjl.org [RB 2022]
-  manopt.org [Boumal, Mishra, Absil, and Sepulchre 2014]
-  pymanopt.org [Townsend, Koep, and Weichwald 2016]

Algorithms.

- ▶ Nelder-Mead, Particle Swarm
- ▶ Subgradient Method
- ▶ Gradient Descent
CG, Stochastic, Momentum, ...
- ▶ Quasi-Newton
BFGS, DFP, Broyden, SR1, ...
- ▶ Trust Regions
- ▶ Chambolle-Pock
- ▶ Douglas-Rachford, CPPA
- ▶ ALM, EPM, Frank-Wolfe, ...
- ▶ Difference of Convex
DCA, DCPA

Implementation

The Algorithm is implemented¹ in Julia using `Manopt.jl` which uses manifolds from `Manifolds.jl` A solver call just looks like

```
x_star = difference_of_convex_algorithm(M, f, g, ∂h, p0)
```

where

- ▶ provide `∂h(M, X, p)!` to reuse the memory `X` for the `∂h`
- ▶ a sub problem is automatically generated
- ▶ an efficient version of the cost and gradient is provided
- ▶ you can specify the sub-solver to using `sub_state=` to also set up the specific parameters of your favourite algorithm

Rosenbrock and First Order Methods

Problem. We consider the classical Rosenbrock example

$$\arg \min_{x \in \mathbb{R}^2} a(x_1^2 - x_2)^2 + (x_1 - b)^2,$$

where $a, b > 0$ are positive numbers, classically $b = 1$ and $a \gg b$
We will use $a = 2 \cdot 10^5$ (to see an effect).

Known Minimizer $x^* = \begin{pmatrix} b \\ b^2 \end{pmatrix}$ with cost $f(x^*) = 0$.

Goal. Compare first-order methods, i.e. using the (Euclidean) gradient

$$\nabla f(x) = \begin{pmatrix} 4a(x_1^2 - x_2) \\ -2a(x_1^2 - x_2) \end{pmatrix} + \begin{pmatrix} 2(x_1 - b) \\ 0 \end{pmatrix}$$

A New Metric on \mathbb{R}^2

In our Riemannian framework, we can introduce a new metric on \mathbb{R}^2 as

$$G_p := \begin{pmatrix} 1 + 4p_1^2 & -2p_1 \\ -2p_1 & 1 \end{pmatrix}, \text{ with inverse } G_p^{-1} = \begin{pmatrix} 1 & 2p_1 \\ 2p_1 & 1 + 4p_1^2 \end{pmatrix}.$$

We obtain $(X, Y)_p = X^T G_p Y$

The exponential and logarithmic map are given as

$$\exp_p(X) = \begin{pmatrix} p_1 + X_1 \\ p_2 + X_2 + X_1^2 \end{pmatrix}, \quad \log_p(q) = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 - (q_1 - p_1)^2 \end{pmatrix}.$$

The Riemannian Gradient w.r.t. the New Metric

Let $f: \mathcal{M} \rightarrow \mathbb{R}$. Since we just changed the metric, its Riemannian gradient $\text{grad } f: \mathcal{M} \rightarrow T\mathcal{M}$ can be computed by

$$\text{grad } f(p) = G_p^{-1} \nabla f(p).$$

Denoting the two components of the Euclidean gradient by $\nabla f(p) = \begin{pmatrix} f'_1(p) \\ f'_2(p) \end{pmatrix}$ we can derive that given two points $p, q \in \mathcal{M}$ we have

$$\left\langle \text{grad } f(q), \log_q p \right\rangle_q = (p_1 - q_1) f'_1(q) + (p_2 - q_2 - (p_1 - q_1)^2) f'_2(q)$$

This is [automatically](#) done in `Manopt.jl`.

The Experiment Setup

Algorithms. We now compare

1. The Euclidean gradient descent algorithm on \mathbb{R}^2 ,
2. The Riemannian gradient descent algorithm on \mathcal{M} ,
3. The Difference of Convex Algorithm on \mathcal{M} ,
using Riemannian gradient descent as a sub-solver

For the third we split f into $f(x) = g(x) - h(x)$ with

$$g(x) = a(x_1^2 - x_2)^2 + 2(x_1 - b)^2 \quad \text{and} \quad h(x) = (x_1 - b)^2.$$

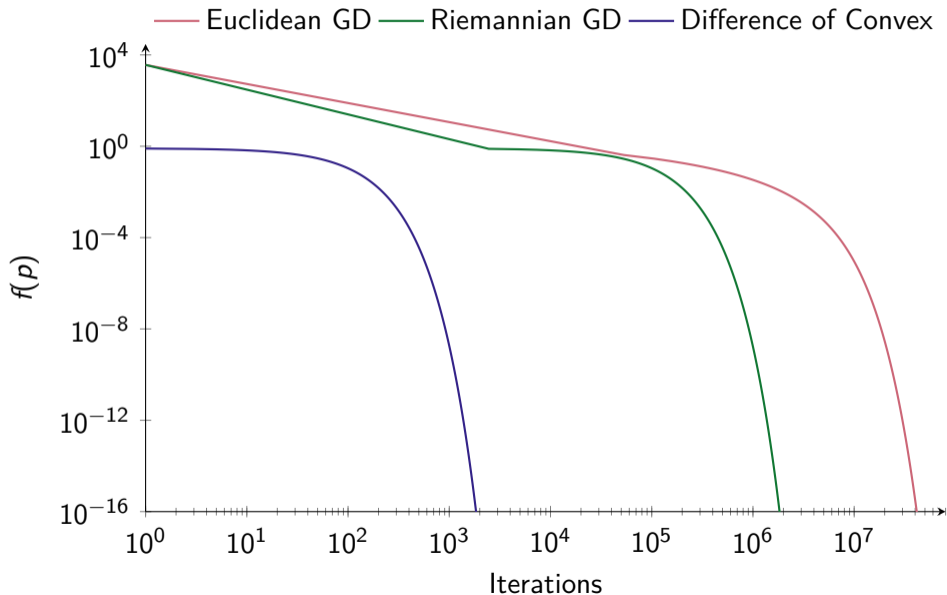
and use

$$p_0 = \frac{1}{10} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \text{with cost } f(p_0) \approx 7220.81.$$

Stopping Criterion. Change in iterates $< 10^{-16}$.

For the sub solver a gradient norm $< 10^{-16}$.

The Results



The Results

Algorithm	Runtime	# Iterations
Euclidean GD	305.567 sec.	53 073 227
Riemannian GD	18.894 sec.	2 454 017
Difference of Convex Algorithm	7.704 sec.	2 459

References

-  Almeida, Y. T., J. X. d. C. Neto, P. R. Oliveira, and J. C. d. O. Souza (Feb. 2020). “A modified proximal point method for DC functions on Hadamard manifolds”. In: *Computational Optimization and Applications* 76.3, pp. 649–673. DOI: 10.1007/s10589-020-00173-3.
-  RB, O. P. Ferreira, E. M. Santos, and J. C. O. Souza (2023). *The difference of convex algorithm on Hadamard manifolds*. arXiv: 2112.05250.
-  Silva Louzeiro, M., RB, and R. Herzog (June 2022). “Fenchel Duality and a Separation Theorem on Hadamard Manifolds”. In: *SIAM Journal on Optimization* 32.2, pp. 854–873. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/21M1400699. arXiv: 2102.11155.
-  Souza, J. C. d. O. and P. R. Oliveira (Feb. 2015). “A proximal point algorithm for DC functions on Hadamard manifolds”. In: *Journal of Global Optimization* 63.4, pp. 797–810. DOI: 10.1007/s10898-015-0282-7.

 ronnybergmann.net/talks/2023-MaGIC-Difference-of-Convex.pdf