The Difference of Convex Algorithm on Riemannian Manifolds

Ronny Bergmann

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

10th International Congress on Industrial and Applied Mathematics,
Tokyo & online
August 24, 2023

NTNU
Norwegian University of Science and Technology

Difference of Convex

We aim to solve

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

where

- $\mathcal{M}$ is a Riemannian manifold
- $f\colon \mathcal{M} \to \mathbb{R}$ is a difference of convex function, i. e. of the form

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

- $g, h\colon \mathcal{M} \to \overline{\mathbb{R}}$ are convex, lower semicontinuous, and proper

A Riemannian Manifold $\mathcal{M}$

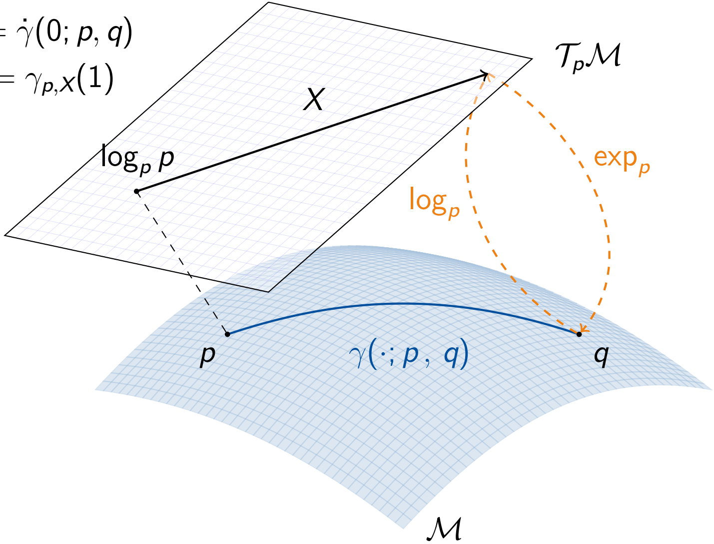
A d-dimensional Riemannian manifold can be informally defined as a set $\mathcal{M}$ covered with a "suitable" collection of charts, that identify subsets of $\mathcal{M}$ with open subsets of $\mathbb{R}^d$ and a continuously varying inner product on the tangent spaces.

[Absil, Mahony, and Sepulchre 2008]

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 $(\cdot\,,\,\cdot)_p$



(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 $F\colon \mathcal{C} \to \overline{\mathbb{R}}$ 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]

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

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 $\mathcal{T}_p^*\mathcal{M} \times \mathcal{T}_p\mathcal{M}$

The Euclidean DCA

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

$\Rightarrow$ iteratively 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

$$y_k \in \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \Bigl\{ h^*(y) - g^*(y_{k-1}) - \langle y - y_{k-1}, x_k \rangle \Bigr\}$$

Idea 3. Formulate the idea using a proximal map $\Rightarrow$ DCPPA

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

In the Euclidean case, all three models are equivalent.

Derivation of the Riemannian DCA

We consider the linearization of h at some point p_k:
With $\xi \in \partial h(p_k)$ we get

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

Using musical isomorphisms we identify $X = \xi^\sharp \in T_p\mathcal{M}$,
where we call X a subgradient. Locally h_k minorizes h, i. e.

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

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

Note. On $\mathbb{R}^n$ the function h_k is linear.
On a manifold h_k is not necessarily convex, even on a Hadamard manifold.

The Riemannian DC Algorithm

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

Input: An initial point $p^0 \in \operatorname{dom}(g)$, 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

$$p_{k+1} \in \operatorname*{arg\,min}_{p \in \mathcal{M}} \Bigl(g(p) - \bigl(X_k \,,\, \log_{p_k} p \bigr)_{p_k} \Bigr). \qquad (*)$$

5: Set $k \leftarrow k + 1$

6: **end while**

Note. In general the subproblem $(*)$ can not be solved in closed form. But an approximate solution yields a good candidate.

Convergence of the Riemannian DCA

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

Let $\{p_k\}_{k\in\mathbb{N}}$ and $\{X_k\}_{k\in\mathbb{N}}$ be the iterates and subgradients of the RDCA.

Theorem.
If $\bar{p}$ is a cluster point of $\{p_k\}_{k\in\mathbb{N}}$, then $\bar{p} \in \operatorname{dom}(g)$ 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.

Proposition. Let g be σ-strongly (geodesically) convex. Then

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

and $\sum_{k=0}^{\infty} d^2(p_k, p_{k+1}) < \infty$, so in particular $\lim_{k\to\infty} d(p_k, p_{k+1}) = 0$.

NTNU

ManifoldsBase.jl

[Axen, Baran, RB, and 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

NTNU

[Axen, Baran, RB, and Rzecki 2023]

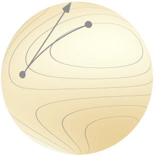
Goal. Provide a library of Riemannian manifolds, that is efficiently implemented and well-documented

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
- (generalized, symplectic) Stiefel, (generalized) Grassmann, Rotations
- symmetric positive definite matrices
- multinomial, symmetric, symplectic matrices
- Tucker & Oblique manifold, Kendall's Shape space

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

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

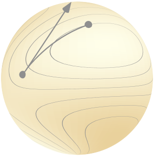
Manopt family.

manoptjl.org [RB 2022]

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

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

Manopt.jl



Algorithms.

Cost-based Nelder-Mead, Particle Swarm

Subgradient-based Subgradient Method

Gradient-based Gradient Descent, Conjugate Gradient, Stochastic,
Momentum, Nesterov, Averaged, …
Quasi-Newton: (L-)BFGS, DFP, Broyden, SR1,…

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

non-smooth Chambolle-Pock, Douglas-Rachford, Cyclic Proximal Point

constrained Augmented Lagrangian, Exact Penalty, Frank-Wolfe

non-convex Difference of Convex Algorithm, DCPPA

manoptjl.org

Implementation of the DCA

The algorithm is implemented and released in Julia using `Manopt.jl`[1].
It can be used with any manifold from `Manifolds.jl`

A solver call looks like

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

where one has to implement `f(M, p)`, `g(M, p)`, and `∂h(M, p)`.

- a sub problem is automatically generated
- an efficient version of its 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

[1] see https://manoptjl.org/stable/solvers/difference_of_convex/

Rosenbrock and First Order Methods

Problem. We consider the classical Rosenbrock example[2]

$$\operatorname*{arg\,min}_{x\in\mathbb{R}^2} a\bigl(x_1^2 - x_2\bigr)^2 + \bigl(x_1 - b\bigr)^2,$$

where $a, b > 0$, usually $b = 1$ and $a \gg b$, here: $a = 2 \cdot 10^5$.

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

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

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

[2] available online in ManoptExamples.jl

A “Rosenbrock-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 $(X, Y)_p = X^\mathrm{T} G_p Y$

The exponential and logarithmic map are given as

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

`Manifolds.jl`:
Implement these functions on `MetricManifold(ℝ^2, RosenbrockMetric())`.

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

Let $f\colon \mathcal{M} \to \mathbb{R}$. Given the Euclidean gradient $\nabla f(p)$, its Riemannian gradient $\operatorname{grad} f\colon \mathcal{M} \to T\mathcal{M}$ is given by

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

While we could implement this denoting $\nabla f(p) = \begin{pmatrix} f_1'(p) & f_2'(p) \end{pmatrix}^{\mathrm{T}}$ using

$$\Bigl\langle \operatorname{grad} f(q), \log_q p \Bigr\rangle_q = (p_1 - q_1) f_1'(q) + \bigl(p_2 - q_2 - (p_1 - q_1)^2\bigr) f_2'(q),$$

but it 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 $\mathbb{R}^2$,
4. The Difference of Convex Algorithm on $\mathcal{M}$.

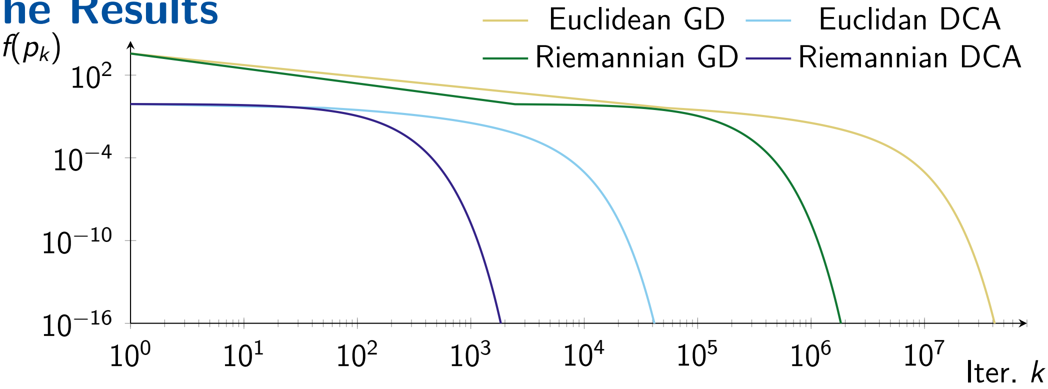
For DCA third we split f into $f(x) = g(x) - h(x)$ with

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

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

Stopping Criterion. $d_{\mathcal{M}}(p_k, p_{k-1}) < 10^{-16}$ or $\lVert \operatorname{grad} f(p_k)\rVert_p < 10^{-16}$.

The Results



Algorithm	**Runtime**	**# Iterations**
Euclidean GD	305.567 sec.	53 073 227
Euclidean DCA	58.268 sec.	50 588
Riemannian GD	18.894 sec.	2 454 017
Riemannian DCA	7.704 sec.	2 459

Selected 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.

Axen, S. D., M. Baran, RB, and K. Rzecki (2023). "Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds". In: *ACM Transactions on Mathematical Software*. Accepted for pulication. 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.

RB, O. P. Ferreira, E. M. Santos, and J. C. O. Souza (2023). *The difference of convex algorithm on Hadamard manifolds*. arXiv: 2112.05250.

Souza, J. C. d. O. and P. R. Oliveira (Feb. 2015). "A proximal point algorithm for DC fuctions on Hadamard manifolds". In: *Journal of Global Optimization* 63.4, pp. 797–810. DOI: 10.1007/s10898-015-0282-7.