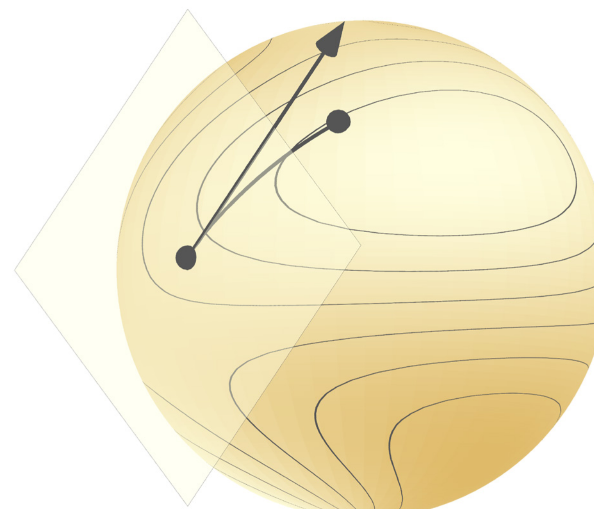
212

# Momentum methods and nonlinear conjugate gradients

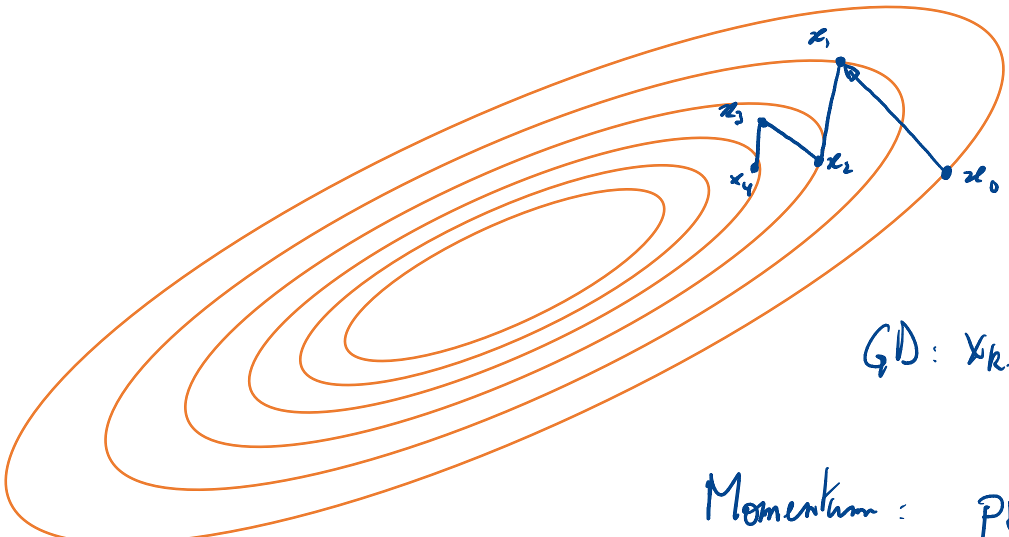


Spring 2023

Optimization on manifolds, MATH 512 @ EPFL

Instructor: Nicolas Boumal

# Gradient descent in $\mathbf{R}^2$, now with memory




GD: $x_{k+1} = x_k - \alpha_k \, \mathrm{grad} f(x_k)$

Momentum: $p_k = -\mathrm{grad} f(x_k) + \beta_k \overbrace{(x_k - x_{k-1})}^{p_{k-1}}$

$x_{k+1} = x_k + \alpha_k p_k$

$\hookrightarrow x_{k+1} - x_k \propto p_k$

$x_k - x_{k-1} \propto p_{k-1}$

# Conjugate gradients uses momentum

CG for $g(v) = \frac{1}{2}\langle v, Hv\rangle - \langle b, v\rangle$:

**Initialize** $v_0 = 0, r_0 = b, p_0 = r_0$

**For** $n$ in 1, 2, 3, ...

- $\alpha_n = \frac{\|r_{n-1}\|^2}{\langle p_{n-1}, Hp_{n-1}\rangle}$
- $v_n = v_{n-1} + \alpha_n p_{n-1}$
- $r_n = r_{n-1} - \alpha_n H p_{n-1}$ $r_n = -\mathrm{grad}\, g(v_n)$
- $\beta_n = \frac{\|r_n\|^2}{\|r_{n-1}\|^2}$
- $p_n = \underbrace{r_n}_{-\mathrm{grad}\, g(v_n)} + \beta_n p_{n-1}$

① Quid for more general nonlinear cost functions?

② On manifolds?

# Nonlinear CG on manifolds

Riemannian CG (RCG) for $f : \mathcal{M} \to \mathbf{R}$:

**Initialize** $x_0 \in \mathcal{M}$, $p_0 = -\mathrm{grad} f(x_0)$

**For** $k$ in 0, 1, 2, …

- **If** $\langle \mathrm{grad} f(x_k), p_k \rangle_{x_k} \geq 0$, set $p_k = -\mathrm{grad} f(x_k)$.
  "restart"
- $x_{k+1} = R_{x_k}(\alpha_k p_k)$ with line-search for $\alpha_k$
- $\beta_k = \cdots$ (many heuristics)
- $p_{k+1} \overset{!}{=} -\mathrm{grad} f(x_{k+1}) + \beta_k p_k$

  ???

$T_{x_{k+1}}\mathcal{M}$ $T_{x_{k+1}}\mathcal{M}$ $T_{x_k}\mathcal{M}$

# The take-aways

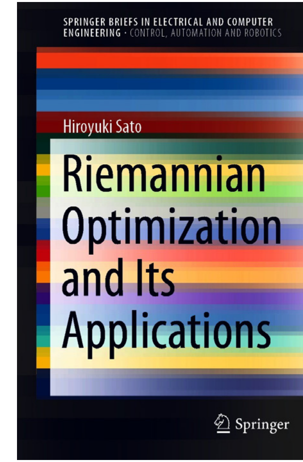
With transporters, RCG yields a rich family of algorithms.

Various rules for $\beta_k$ (and line-search) affect performance.

Manopt: `help conjugategradient` and check `options.beta_type`.

No Hessians needed.

Theory is delicate: see Sato 2021.




```
beta_type ('H-S')
       Conjugate gradient beta rule used to construct the new search
       direction, based on a linear combination of the previous search
       direction and the new (preconditioned) gradient. Possible values
       for this parameter are:
           'S-D', 'steep' for beta = 0 (preconditioned steepest descent)
           'F-R' for Fletcher-Reeves's rule
           'P-R' for Polak-Ribiere's modified rule
           'H-S' for Hestenes-Stiefel's modified rule
           'H-Z' for Hager-Zhang's modified rule
           'L-S' for Sato's Liu-Storey rule
       See Hager and Zhang 2006, "A survey of nonlinear conjugate gradient
       methods" for a description of these rules in the Euclidean case and
       for an explanation of how to adapt them to the preconditioned case.
       The adaption to the Riemannian case is straightforward: see in code
       for details. Modified rules take the max between 0 and the computed
       beta value, which provides automatic restart, except for H-Z and L-S
       which use a different modification. Sato's Liu-Storey rule is
       described in Sato 2021, "Riemannian conjugate gradient methods:
       General framework and specific algorithms with convergence analyses"
```