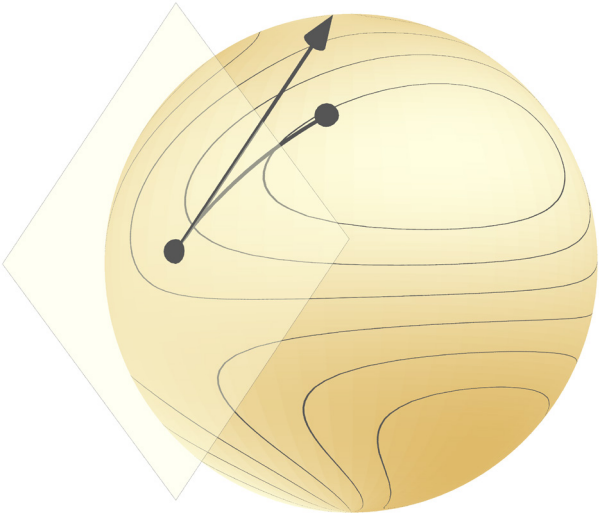
208

Computing the Newton step



Spring 2023

Optimization on manifolds, MATH 512 @ EPFL

Instructor: Nicolas Boumal

The linear equation of Newton steps

Let $f: \mathcal{M} \to \mathbf{R}$ be smooth on a Riemannian manifold $\mathcal{M}$.

The Newton step at x is the tangent vector $s \in \mathrm{T}_x\mathcal{M}$ which solves:

$$\mathrm{Hess} f(x)[s] = -\mathrm{grad} f(x)$$

This is a linear equation in the unknown s. How do we solve it?

(Once solved, move from x to $R_x(s)$ with the chosen retraction R.)

Matrix-based approaches

$H : T_x\mathcal{M} \to T_x\mathcal{M}$: $H = \mathrm{Hess} f(x)$

$b \in T_x\mathcal{M}$: $b = -\mathrm{grad} f(x)$

Pick a basis $u_1, \ldots, u_d \in T_x\mathcal{M}$ for $T_x\mathcal{M}$.

We have a linear system to solve: $Hs = b$, for $s \in T_x\mathcal{M}$.

$$s = \sum_{i=1}^{d} \alpha_i u_i \; ; \qquad \sum_i \alpha_i H u_i = b$$

$H_{ij} = \langle H u_i, u_j \rangle_x$

$b_j = \langle b, u_j \rangle_x$

$H\alpha = b$

$$\left\langle \sum_i \alpha_i H u_i, u_j \right\rangle_x = \langle b, u_j \rangle_x \quad \forall j$$

$$\sum_i \alpha_i \langle H u_i, u_j \rangle_x = \langle b, u_j \rangle_x \quad \forall j$$

Paradigm: We have access to $\mathrm{Hess} f(x)$ as an operator.

Calling that operator dominates computational costs.

Matrix-free approaches

Assuming $\mathrm{Hess} f(x) \succ 0$, recall the Newton step s minimizes:

$$m_x(v) = \frac{1}{2}\langle v, \mathrm{Hess} f(x)[v]\rangle_x + \langle \mathrm{grad} f(x), v\rangle_x + f(x)$$

Maybe we could run an optimization algorithm on $m_x \colon \mathrm{T}_x\mathcal{M} \to \mathbf{R}$?

$g : T_x M \to \mathbb{R}$, $g(v) = \frac{1}{2}\langle v, Hv\rangle_x - \langle b, v\rangle_x$, $\begin{cases} b = -\mathrm{grad} f(x) \\ H : \mathrm{Hess} f(x) \end{cases}$

Gradient descent to g?

$v_0 = 0$; $\quad v_n = v_{n-1} - d_n \, \mathrm{grad} g(v_{n-1})$

$$\text{grad}\, g(v) = Hv - b$$

$$g(v+\alpha r) = \frac{1}{2}\langle v+\alpha r,\ H(v+\alpha r)\rangle_x - \langle b,\ v+\alpha r\rangle_x$$

$$= \frac{1}{2}\langle v, Hv\rangle_x + \frac{\alpha}{2}\langle v, Hr\rangle_x + \frac{\alpha}{2}\langle r, Hv\rangle_x + \frac{\alpha^2}{2}\langle r, Hr\rangle_x - \langle b, v\rangle_x - \alpha\langle b, r\rangle_x$$

$$= g(v) + \alpha\langle r, Hv - b\rangle_x + \frac{\alpha^2}{2}\langle r, Hr\rangle_x$$

$\Rightarrow$ optimal α is: $\alpha = -\dfrac{\langle r, Hv-b\rangle_x}{\langle r, Hr\rangle_x}$

For GD: $\alpha = \dfrac{\|r\|_x^2}{\langle r, Hr\rangle_x}$

Gradient descent to minimize $g(v) = \frac{1}{2}\langle v, Hv\rangle_x - \langle b, v\rangle_x$ on $\mathrm{T}_x\mathcal{M}$:

Initialize $v_0 = 0$ and $r_0 = -\mathrm{grad}\, g(v_0) = b - Hv_0 = b$

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

- Compute Hr_{n-1}. (only call to H per iteration)
- $\alpha_n = \dfrac{\|r_{n-1}\|_x^2}{\langle r_{n-1}, Hr_{n-1}\rangle_x}$
- $v_n = v_{n-1} + \alpha_n r_{n-1}$
- $r_n = -\mathrm{grad}\, g(v_n) = b - Hv_n = b - H(v_{n-1} + \alpha_n r_{n-1}) = b - Hv_{n-1} - \alpha_n Hr_{n-1}$
  $= r_{n-1} - \alpha_n Hr_{n-1}$.
- **If** $\|r_n\|_x \leq \mathrm{tol} \cdot \|b\|_x$, **output** v_n

GD is ok, but we can surely do better.

Observe that GD iterates are linear combinations of the residues:

$$v_n = v_{n-1} + \alpha_n r_{n-1} = v_{n-2} + \alpha_{n-1} r_{n-2} + \alpha_n r_{n-1}$$
$$\vdots$$
$$= \alpha_1 r_0 + \alpha_2 r_1 + \dots + \alpha_n r_{n-1} .$$

Idea: Choose $\alpha_1, \dots, \alpha_n$ s.t. v_n minimizes $g(v) = \frac{1}{2}\langle v, Hv\rangle_x - \langle b, v\rangle_x$ over the whole subspace spanned by $r_0, \dots, r_{n-1}$.

$$g\Big(\sum_{i=1}^{n} \alpha_i r_{i-1}\Big) = \frac{1}{2}\Big\langle \sum_i \alpha_i r_{i-1}, H\Big(\sum_j \alpha_j r_{j-1}\Big)\Big\rangle_x - \Big\langle b, \sum_i \alpha_i r_{i-1}\Big\rangle_x$$

$$= \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \langle r_{i-1}, Hr_{j-1}\rangle_x - \sum_i \alpha_i \langle b, r_{i-1}\rangle_x$$

M_{ij} $\quad$ c_i

$$= \frac{1}{2} \alpha^T M \alpha - c^T \alpha .$$

$M > 0$ as long as the vectors r are lin. indep.

$\Rightarrow$ optimal α solves $M\alpha = c$.

optimal: $v_n = \sum_{i=1}^{n} \alpha_i p_{i-1}$ ← Gram-Schmidt the vectors $r_0, r_1, \ldots$

with: $M\alpha = c$,

$$M_{ij} = \langle p_{i-1}, H p_{j-1} \rangle_x$$

└ diagonal!

$$\alpha_i = \frac{\langle b, p_{i-1} \rangle_x}{\langle p_{i-1}, H p_{i-1} \rangle_x}$$

Gradient descent (GD)

Initialize $v_0 = 0, r_0 = b$

$r = -\text{grad}\, g(v_0)$

For n in 1, 2, 3, …

- Compute Hr_{n-1}
- $\alpha_n = \frac{\|r_{n-1}\|_x^2}{\langle r_{n-1}, Hr_{n-1}\rangle_x}$
- $v_n = v_{n-1} + \alpha_n r_{n-1}$
- $r_n = r_{n-1} - \alpha_n Hr_{n-1}$
- **If** $\|r_n\|_x \le \text{tol} \cdot \|b\|_x$, **output** v_n

$r_n = -\text{grad}\, g(v_n) = b - Hv_n$

Conjugate gradients (CG)

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

For n in 1, 2, 3, …

- Compute Hp_{n-1}
- $\alpha_n = \frac{\|r_{n-1}\|_x^2}{\langle p_{n-1}, Hp_{n-1}\rangle_x}$
- $v_n = v_{n-1} + \alpha_n p_{n-1}$
- $r_n = r_{n-1} - \alpha_n Hp_{n-1}$
- **If** $\|r_n\|_x \le \text{tol} \cdot \|b\|_x$, **output** v_n
- $\beta_n = \frac{\|r_n\|_x^2}{\|r_{n-1}\|_x^2}$
- $p_n = r_n + \beta_n p_{n-1}$

Cheap Gram–Schmidt

$$p_n = r_n - \sum_{i=0}^{n-1} \frac{\langle r_n, H p_i \rangle_x}{\langle p_i, H p_i \rangle_x} p_i$$

$$g(v) = \frac{1}{2}\langle v, Hv\rangle_x - \langle b, v\rangle_x$$

Finite termination of CG…

CG iterates are $v_n = \operatorname{argmin}_{v \in \mathcal{K}_n} g(v)$ with $\mathcal{K}_n = \operatorname{span}(r_0, \ldots, r_{n-1})$.

Assume $r_0, \ldots, r_{n-1}$ are linearly independent (for some n).

If $r_n = 0$, then v_n is the global optimum: stop.

If $r_n \neq 0$, then $r_0, \ldots, r_n$ are linearly independent.

Since $\mathcal{K}_n \subseteq \mathrm{T}_x\mathcal{M}$, unless we stop early, $\mathcal{K}_n = \mathrm{T}_x\mathcal{M}$ for $n = \dim \mathcal{M}$.

Thus, v_n is the exact solution for some $n \leq \dim \mathcal{M}$.

… is a mathematical fairy tale.

Numerically, $p_0, p_1, p_2, \ldots$ progressively lose H-orthogonality.

As a result, implementations of CG do *not* find the exact solution.

But CG is still much better than GD!

In theory *and* in practice, both GD and CG enjoy linear convergence.
Only, CG's rate is typically much better than GD's.
Enough to get machine precision (or stop early).