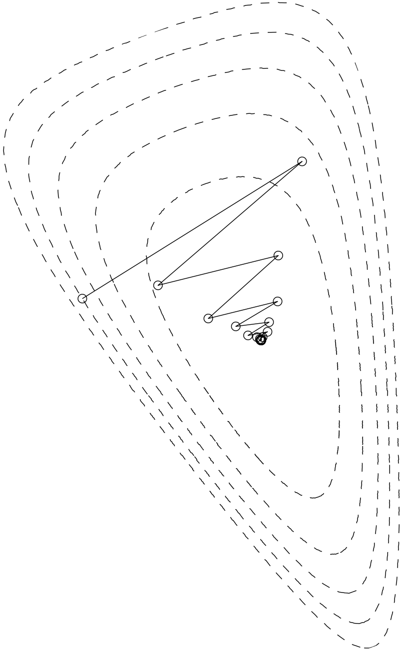


Optimization Algorithms

Newton Method & Steepest Descent

steepest descent, Newton, damping, trust region, non-convex fallback

Marc Toussaint
Technical University of Berlin
Winter 2024/25

Detour: Steepest Descent Direction

- The gradient $-\nabla f(x)$ is sometimes called *steepest descent direction*

Is it really?

Detour: Steepest Descent Direction

- The gradient $-\nabla f(x)$ is sometimes called *steepest descent direction*

Is it really?

- Here is a possible definition:

The steepest descent direction is the one where, when you make a step of length 1, you get the largest decrease of f in its linear approximation.

$$\underset{\delta}{\text{argmin}}\, \nabla f(x)^\top \delta \qquad \text{s.t. } \|\delta\| = 1$$

Detour: Steepest Descent Direction

- But the norm $\|\delta\|^2 = \delta^\top A \delta$ depends on the metric A!

  Let $A = B^\top B$ (Cholesky decomposition) and $z = B\delta$

$$
\begin{aligned}
\delta^* &= \operatorname*{argmin}_\delta \nabla f^\top \delta \qquad \text{s.t. } \delta^\top A \delta = 1 \\
&= B^{\text{-}1} \operatorname*{argmin}_z (B^{\text{-}1} z)^\top \nabla f \qquad \text{s.t. } z^\top z = 1 \\
&= B^{\text{-}1} \operatorname*{argmin}_z z^\top B^{\text{-}\top} \nabla f \qquad \text{s.t. } z^\top z = 1 \\
&\propto B^{\text{-}1} [-B^{\text{-}\top} \nabla f] = -A^{\text{-}1} \nabla f
\end{aligned}
$$

- The steepest descent direction is $\delta = -A^{\text{-}1} \nabla f$

Detour: Steepest Descent Direction

- **Behavior under linear coordinate transformations:**
  - Let B be a matrix that describes a linear transformation in coordinates
  - A coordinate vector x transforms as $z = Bx$
  - The plain gradient $\nabla_x f(x)$ transforms as $\nabla_z f(z) = B^{-\top} \nabla_x f(x)$
  - The metric A transforms as $A_z = B^{-\top} A_x B^{-1}$
  - The steepest descent transforms as $A_z^{-1} \nabla_z f(z) = B A_x^{-1} \nabla_x f(x)$

$\Rightarrow$ **The steepest descent vector is a covariant.** (I.e., it's coordinates transform like those of an ordinary vector.)

(more details in the *Maths script*)

Detour: Steepest Descent Direction

- **Behavior under linear coordinate transformations:**
  - Let B be a matrix that describes a linear transformation in coordinates
  - A coordinate vector x transforms as $z = Bx$
  - The plain gradient $\nabla_x f(x)$ transforms as $\nabla_z f(z) = B^{-\top} \nabla_x f(x)$
  - The metric A transforms as $A_z = B^{-\top} A_x B^{-1}$
  - The steepest descent transforms as $A_z^{-1} \nabla_z f(z) = B A_x^{-1} \nabla_x f(x)$

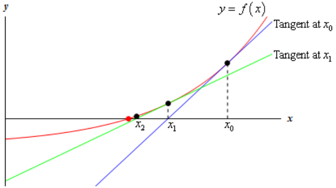
$\Rightarrow$ **The steepest descent vector is a covariant.** (I.e., it's coordinates transform like those of an ordinary vector.) (more details in the *Maths script*)

- Relevance in practise:
  - When the decision variable x lives in a non-Euclidean space
  - E.g. when x is a probability distribution $\rightarrow$ use the **Fisher metric** in probability space $\rightarrow$ leads to the **natural gradient**

Newton Method

Newton Step

- For finding roots (zero points) of $f(x)$



$$x \leftarrow x - \frac{f(x)}{f'(x)}$$

- For finding optima of $f(x)$ in 1D (which are roots of $f'(x)$):

$$x \leftarrow x - \frac{f'(x)}{f''(x)}$$

- For finding optima in higher dimensions $x \in \mathbb{R}^n$:

$$x \leftarrow x - \nabla^2 f(x)^{-1} \nabla f(x)$$

Hessian

- The **Hessian** of f is defined as

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2}{\partial_{x_1}\partial_{x_1}} f(x) & \frac{\partial^2}{\partial_{x_1}\partial_{x_2}} f(x) & \cdots & \frac{\partial^2}{\partial_{x_1}\partial_{x_n}} f(x) \\ \frac{\partial^2}{\partial_{x_1}\partial_{x_2}} f(x) & & & \vdots \\ \vdots & & & \vdots \\ \frac{\partial^2}{\partial_{x_n}\partial_{x_1}} f(x) & \cdots & \cdots & \frac{\partial^2}{\partial_{x_n}\partial_{x_n}} f(x) \end{pmatrix} \in \mathbb{R}^{n\times n}$$

- Provides the Taylor expansion:

$$f(x+\delta) \approx f(x) + \nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x)\ \delta$$

Note: $\nabla^2 f(x)$ acts like a **metric** for δ

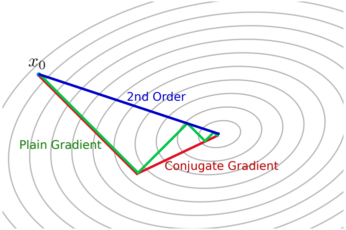
- $\nabla f(x)^\top \delta$ is the **directional derivative**, and $\delta^\top \nabla^2 f(x)\ \delta$ the **directional 2nd derivative**

Notes on the Newton Step

- If f is a 2nd-order polynomial, the Newton step jumps to the optimum in just one step.
- *Unlike the gradient magnitude* $|\nabla f(x)|$, the magnitude of the Newton step δ is meaningful and scale invariant!
  - If you'd rescale f or x, δ is unchanged
- *Unlike the gradient* $\nabla f(x)$, the Newton step δ is truely a vector!
  - The Newton step is invariant under coordinate transformations; the coordinates of δ transform contra-variant, as it is supposed to for vector coordinates
  - The proof is exactly the same as for the steepest descent with a non-Euclidean metric – the Hessian acts as a metric

Why 2nd order information is better

- Better direction:



- Better stepsize:
  - A full Newton step jumps directly to the minimum of the local squared approx.
  - Robust Newton methods combine this with line search and damping (Levenberg-Marquardt)

Basic Newton method

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, tolerance θ, parameters (defaults: $\varrho_\alpha^+ = 1.2, \varrho_\alpha^- = 0.5, \varrho_{\text{ls}} = 0.01, \lambda$)

1: initialize stepsize $\alpha = 1$, fixed damping λ
2: **repeat**
3: compute δ to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\, \delta = -\nabla f(x)$
4: **while** $f(x + \alpha\delta) > f(x) + \varrho_{\text{ls}} \nabla f(x)^\top (\alpha\delta)$ **do** *// line search*
5: $\alpha \leftarrow \varrho_\alpha^- \alpha$ *// decrease stepsize*
6: **end while**
7: $x \leftarrow x + \alpha\delta$ *// step is accepted*
8: $\alpha \leftarrow \min\{\varrho_\alpha^+ \alpha, 1\}$ *// increase stepsize*
9: **until** $\|\alpha\delta\|_\infty < \theta$

- Notes:
  - Line 3 computes the Newton step $\delta = -\nabla^2 f(x)^{-1} \nabla f(x)$,
    e.g. using a special Lapack routine `dposv` to solve $Ax = b$ (using Cholesky)

Basic Newton method

- What if the Hessian is *negative* definite? $\rightarrow$ The Newton step jumps to a *maximum*!

Basic Newton method

- What if the Hessian is *negative* definite? $\to$ The Newton step jumps to a *maximum*!

- What if some eigenvalues are positive, some negative? (This is called a *saddle point*?

Basic Newton method

- What if the Hessian is *negative* definite? $\to$ The Newton step jumps to a *maximum*!

- What if some eigenvalues are positive, some negative? (This is called a *saddle point*?

$\to$ For robust minimization, we need to have a fallback for non-positive definite Hessian

Newton method with non-pos-def fallback

initialize stepsize $\alpha = 1$
repeat
    try to compute δ to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\ \delta = -\nabla f(x)$
    if $\nabla f(x)^\top \delta > 0$ (non-descent) or fails (ill-def. linear system) **then**
        $\delta \leftarrow - \frac{\nabla f(x)}{|\nabla f(x)|}$ *// (gradient direction)*
        (Or: choose $\lambda > [-$minimal eigenvalue of $\nabla^2 f(x)]^+$ and repeat)
    end if
    while $f(x + \alpha\delta) > f(x) + \varrho_{\text{ls}} \nabla f(x)^\top (\alpha\delta)$ **do** *// line search*
        $\alpha \leftarrow \varrho_\alpha^- \alpha$ *// decrease stepsize*
        optionally: $\lambda \leftarrow \varrho_\lambda^+ \lambda$ and recompute δ *// increase damping*
    end while
    $x \leftarrow x + \alpha\delta$ *// step is accepted*
    $\alpha \leftarrow \min\{\varrho_\alpha^+ \alpha, 1\}$ *// increase stepsize*
    optionally: $\lambda \leftarrow \varrho_\lambda^- \lambda$ *// decrease damping*
until $\|\alpha\delta\|_\infty < \theta$ repeatedly

Newton method with non-pos-def fallback – Notes

- The λ shifts the eigenvalues: Adding to the diagonal of a matrix, all eigenvalues are shifted

Newton method with non-pos-def fallback – Notes

- The λ shifts the eigenvalues: Adding to the diagonal of a matrix, all eigenvalues are shifted
- This is also called *damping* or **Levenberg-Marquardt**, and related to trust regions

Newton method with non-pos-def fallback – Notes

- The λ shifts the eigenvalues: Adding to the diagonal of a matrix, all eigenvalues are shifted
- This is also called *damping* or **Levenberg-Marquardt**, and related to trust regions
- The specific algo on previous slide is subjective – literal from our research code. But other extensions might be better in other applications; and existing optimization libraries use other tricks to robustify their Newton method.
  - Line 3 of the method on slide 20 says "try to". This assumes that a solver might fail to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\ \delta = -\nabla f(x)$ for δ. This is in particular the case when the solver is based on a Cholesky decomposition, which is highly efficient but only defined for pos-def matrices. The Newton method would have to catch the error signal of this solver.
  - Other solvers can solve also non-pos-dev linear equation systems, but then the computed step δ might not point downhill (e.g., it might point to a sattle point or maximum of $f(x)$). To catch this case, line 4 additionally tests whether δ points downhill.
  - In these failure cases, the extended Newton method uses the plain gradient direction as the fallback (Line 5).
  - Note that the scaling and meaning of α when transitioning between Newton steps and gradient steps is an issue. Both δ's have very different scales and adapting α for one does not translate automatically to the other. A solution might be to maintain separate α's for Newton steps and gradient steps – I have not tested.
  - Lines 6, 10 and 14 mention possible heuristics to adapt the damping λ (which is related to adapting the implicit trust region). However, by default, I would not use these options.

Relation to Trust-Region

- The damped Newton step δ solves the problem

$$\min_{\delta} \left[\nabla f(x)^\top \delta + \frac{1}{2} \delta^\top \nabla^2 f(x) \delta + \frac{1}{2} \lambda \delta^2) \right] .$$

  – where λ introduces a squared penalty for large steps

Relation to Trust-Region

- The damped Newton step δ solves the problem

$$\min_\delta \left[\nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x)\delta + \frac{1}{2}\lambda\delta^2)\right] .$$

  – where λ introduces a squared penalty for large steps

- **Trust region method**:

$$\min_\delta \left[\nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x)\delta\right] \text{ s.t. } \delta^2 \le \beta$$

  – where β defines the *trust region*

Relation to Trust-Region

- The damped Newton step δ solves the problem

$$\min_\delta \left[\nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x) \delta + \frac{1}{2}\lambda \delta^2)\right] .$$

  – where λ introduces a squared penalty for large steps

- **Trust region method**:

$$\min_\delta \left[\nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x) \delta\right] \text{ s.t. } \delta^2 \le \beta$$

  – where β defines the *trust region*

- Solving this using Lagrange parameters (as we will learn it later):

$$L(\delta, \lambda) = \nabla f(x)^\top \delta + \frac{1}{2}\delta^\top \nabla^2 f(x) \delta + \lambda(\delta^2 - \beta) \;, \quad \nabla_\delta L(\delta, \lambda) = \nabla f(x)^\top + \delta^\top (\nabla^2 f(x) + 2\lambda \mathbf{I})$$

  gives the step $\delta = -(\nabla^2 f(x) + 2\lambda \mathbf{I})^{-1} \nabla f(x)$, with λ the **dual variable**

- For $\lambda \to \infty$, δ becomes aligned with $-\nabla f(x)$ (but $|\delta| \to 0$)