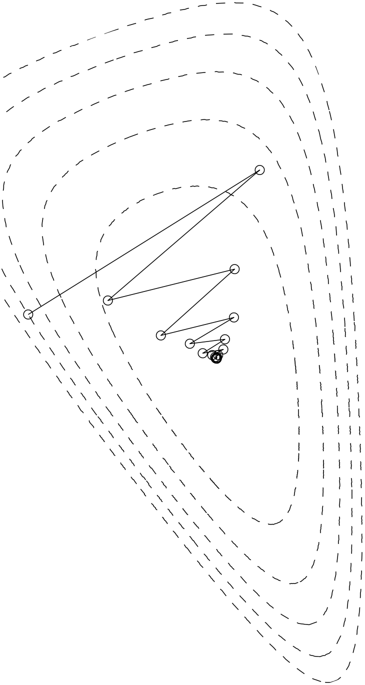


Introduction to Optimization

Second Order Optimization Methods

Marc Toussaint
U Stuttgart

Planned Outline

- Gradient-based optimization (1st order methods)
  - plain grad., steepest descent, conjugate grad., Rprop, stochastic grad.
  - adaptive stepsize heuristics
- Constrained Optimization
  - squared penalties, augmented Lagrangian, log barrier
  - Lagrangian, KKT conditions, Lagrange dual, log barrier $\leftrightarrow$ approx. KKT
- **2nd order methods**
  - Newton, Gauss-Newton, Quasi-Newton, (L)BFGS
  - constrained case, primal-dual Newton
- Special convex cases
  - Linear Programming, (sequential) Quadratic Programming
  - Simplex algorithm
  - relation to relaxed discrete optimization
- Black box optimization ("0th order methods")
  - blackbox stochastic search
  - Markov Chain Monte Carlo methods
  - evolutionary algorithms

- So far we relied on gradient-based methods only, in the unconstrained and constrained case

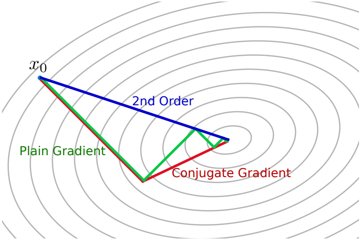
- Today: 2nd order methods, which approximate $f(x)$ locally
  - using 2nd order Taylor expansion (Hessian $\nabla^2 f(x)$ given)
  - estimating the Hessian from data

- **2nd order methods only work if the Hessian is everywhere positive definite $\leftrightarrow$ $f(x)$ is convex**

- Note: Approximating $f(x)$ locally or globally is a core concept also in black box optimization

Why can 2nd order optimization be better than gradient?

- Better direction:



- Better stepsize:
  - a *full step* jumps directly to the minimum of the local squared approx.
  - often this is already a good heuristic
  - additional stepsize reduction and dampening are straight-forward

Outline: 2nd order method

- Newton
- Gauss-Newton
- Quasi-Newton
- BFGS, (L)BFGS

- Their application on constrained problems

2nd order optimization

- *Notation:*

  objective function: $f: \mathbb{R}^n \to \mathbb{R}$

  gradient vector: $\nabla f(x) = \left[\frac{\partial}{\partial_x} f(x)\right]^\top \in \mathbb{R}^n$

  Hessian (symmetric matrix):

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

  Taylor expansion:

$$f(x') = f(x) + (x'-x)^\top \nabla f(x) + \frac{1}{2}(x'-x)^\top \nabla^2 f(x)\,(x'-x)$$

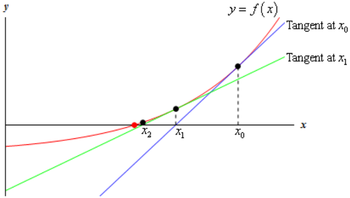
- *Problem:*

$$\min_x f(x)$$

  where we can evaluate $f(x)$, $\nabla f(x)$ and $\nabla^2 f(x)$ for any $x \in \mathbb{R}^n$

Newton method

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



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

- For finding optima of $f(x)$ in 1D:

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

For $x \in \mathbb{R}^n$:

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

Newton method with adaptive stepsize α

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, tolerance θ
Output: x

```
1:  initialize stepsize α = 1 and damping λ = 10^-10
2:  repeat
3:      compute Δ to solve (∇²f(x) + λI) Δ = −∇f(x)
4:      repeat                                              // "line search"
5:          y ← x + αΔ
6:          if f(y) ≤ f(x) then                             // step is accepted
7:              x ← y
8:              α ← α^0.5                // increase stepsize towards α = 1
9:          else                                            // step is rejected
10:             α ← 0.1α                                    // decrease stepsize
11:         end if
12:     until step accepted or (in bad case) α‖Δ‖∞ < θ/1000
13: until ‖Δ‖∞ < θ
```

- Notes:
  - Line 3 computes the Newton step $\Delta = \nabla^2 f(x)^{-1} \nabla f(x)$,
    use special Lapack routine `dposv` to solve $Ax = b$ (using Cholesky decomposition)
  - λ is called **damping**, makes the parabola more "steep" around current x
    for $\lambda \to \infty$: Δ becomes colinear with $-\nabla f(x)$ but $|\Delta| = 0$

Newton method with adaptive damping λ (Levenberg-Marquardt)

- I usually use stepsize adaptation instead of Levenberg-Marquardt

---

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, tolerance θ
Output: x

initialize damping $\lambda = 10^{-10}$
repeat
    compute Δ to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\ \Delta = -\nabla f(x)$
    if $f(x+\Delta) \le f(x)$ **then** *// step is accepted*
        $x \leftarrow x + \Delta$
        $\lambda \leftarrow 0.2\lambda$ *// decrease damping*
    else *// step is rejected*
        $\lambda \leftarrow 10\lambda$ *// increase damping*
    end if
until $\lambda < 1$ and $\|\Delta\|_\infty < \delta$

Computational issues

- Let
  C_f be computational cost of evaluating $f(x)$ only
  C_{eval} be computational cost of evaluating $f(x), \nabla f(x), \nabla^2 f(x)$
  C_Δ be computational cost of solving $(\nabla^2 f(x) + \lambda \mathbf{I})\ \Delta = -\nabla f(x)$

- If $C_{\text{eval}} \gg C_f\ \ \to$ proper line search instead of stepsize adaptation
  If $C_\Delta \gg C_f\ \ \to$ proper line search instead of stepsize adaptation
- However, in many applications (in robotics at least) $C_{\text{eval}} \approx C_f \gg C_\Delta$

- Often, $\nabla^2 f(x)$ is banded (non-zero around diagonal only)
  $\to Ax = b$ becomes super fast using `dpbsv` (Dynamic Programming)

  (If $\nabla^2 f(x)$ is a "tree": Dynamic Programming on the "Junction Tree")

Demo

Gauss-Newton method

- *Problem:*

$$\min_x f(x) \qquad \text{where } f(x) = \phi(x)^\top \phi(x)$$

  and we can evaluate $\phi(x)$, $\nabla\phi(x)$ for any $x \in \mathbb{R}^n$

- $\phi(x) \in \mathbb{R}^d$ is a vector; **each entry contributes a squared cost term** to $f(x)$
- $\nabla\phi(x)$ is the **Jacobian** $(d \times n$-matrix)

$$\nabla\phi(x) = \begin{pmatrix} \frac{\partial}{\partial_{x_1}}\phi_1(x) & \frac{\partial}{\partial_{x_2}}\phi_1(x) & \cdots & \frac{\partial}{\partial_{x_n}}\phi_1(x) \\ \frac{\partial}{\partial_{x_1}}\phi_2(x) & & & \vdots \\ \vdots & & & \vdots \\ \frac{\partial}{\partial_{x_1}}\phi_d(x) & \cdots & \cdots & \frac{\partial}{\partial_{x_n}}\phi_d(x) \end{pmatrix} \in \mathbb{R}^{d\times n}$$

  with 1st-order Taylor expansion $\phi(x') = \phi(x) + \nabla\phi(x)(x' - x)$

Gauss-Newton method

- The gradient and Hessian of $f(x)$ become

$$\begin{aligned} f(x) &= \phi(x)^\top \phi(x) \\ \nabla f(x) &= 2\nabla\phi(x)^\top \phi(x) \\ \nabla^2 f(x) &= 2\nabla\phi(x)^\top \nabla\phi(x) + 2\phi(x)^\top \nabla^2\phi(x) \end{aligned}$$

The Gauss-Newton method is the Newton method for $f(x) = \phi(x)^\top \phi(x)$ with approximating $\nabla^2\phi(x) \approx 0$

The approximate Hessian $2\nabla\phi(x)^\top \nabla\phi(x)$ is always semi-pos-def!

- In the Newton algorithm, replace line 3 by

3: compute Δ to solve $(\nabla\phi(x)^\top \nabla\phi(x) + \lambda\mathbf{I})\ \Delta = -\nabla\phi(x)^\top \phi(x)$

Quasi-Newton methods

Quasi-Newton methods

- Let's take a step back: Assume we *cannot* evaluate $\nabla^2 f(x)$.
  Can we still use 2nd order methods?

- Yes: We can approximate $\nabla^2 f(x)$ from the data $\{(x_i, \nabla f(x_i))\}_{i=1}^k$ of previous iterations

Basic example

- We've seen already two data points $(x_1, \nabla f(x_1))$ and $(x_2, \nabla f(x_2))$
  How can we estimate $\nabla^2 f(x)$?

- In 1D:

$$\nabla^2 f(x) \approx \frac{\nabla f(x_2) - \nabla f(x_1)}{x_2 - x_1}$$

- In $\mathbb{R}^n$: let $y = \nabla f(x_2) - \nabla f(x_1),\ \Delta x = x_2 - x_1$

$$\nabla^2 f(x)\ \Delta x \stackrel{!}{=} y \qquad\qquad \Delta x \stackrel{!}{=} \nabla^2 f(x)^{-1} y$$

$$\nabla^2 f(x) = \frac{y\ y^\top}{y^\top \Delta x} \qquad\qquad \nabla^2 f(x)^{-1} = \frac{\Delta x \Delta x^\top}{\Delta x^\top y}$$

Convince yourself that the last line solves the desired relations
[Left: how to update $\nabla^2 f$(x). Right: how to update directly $\nabla^2 f(x)$-1.]

BFGS

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) method:

Input: initial $x \in \mathbb{R}^n$, functions $f(x)$, $\nabla f(x)$, tolerance θ
Output: x

```
initialize H^{-1} = I_n
repeat
    compute Δ = -H^{-1} ∇f(x)
    perform a line search min_α f(x + αΔ)
    Δ ← αΔ
    y ← ∇f(x + Δ) - ∇f(x)
    x ← x + Δ
    update H^{-1} ← (I - yΔ^T/(Δ^T y))^T H^{-1} (I - yΔ^T/(Δ^T y)) + ΔΔ^T/(Δ^T y)
until ||Δ||_∞ < θ
```

$$H^{-1} \leftarrow \left(\mathbf{I} - \frac{y\Delta^\top}{\Delta^\top y}\right)^\top H^{-1} \left(\mathbf{I} - \frac{y\Delta^\top}{\Delta^\top y}\right) + \frac{\Delta\Delta^\top}{\Delta^\top y}$$

- Notes:
  - The blue term is the H^{-1}-update as on the previous slide
  - The red term "deletes" previous H^{-1}-components

Quasi-Newton methods

- BFGS is the most popular of all Quasi-Newton methods
  Others exist, which differ in the exact H^{-1}-update

- **L-BFGS** (limited memory BFGS) is a version which does not require to explicitly store H^{-1} but instead stores the previous data $\{(x_i, \nabla f(x_i))\}_{i=1}^k$ and manages to compute $\Delta = -H^{-1}\nabla f(x)$ directly from this data

- Some thought:
  In principle, there are alternative ways to estimate H^{-1} from the data $\{(x_i, f(x_i), \nabla f(x_i))\}_{i=1}^k$, e.g. using Gaussian Process regression with derivative observations
  - Not only the derivatives but also the value $f(x_i)$ should give information on $H(x)$ for non-quadratic functions
  - Should one weight 'local' data stronger than 'far away'?
    (GP covariance function)

2nd Order Methods for Constrained Optimization

2nd Order Methods for Constrained Optimization

- *No changes at all for*
  - log barrier
  - augmented Lagrangian
  - squared penalties

  Directly use (Gauss-)Newton/BFGS $\rightarrow$ will boost performance of these constrained optimization methods!

Primal-Dual interior-point Newton Method

- Reconsider slide 03-33 *(Algorithmic implications of the Lagrangian view)*

- A core outcome of the Lagrangian theory was the shift in problem formulation:

$$\text{find } x \text{ to } \min_x f(x) \quad \text{s.t.} \quad g(x) \le 0$$
$$\to \quad \text{find } x \text{ to solve the KKT conditions}$$

Primal-Dual interior-point Newton Method

- The first and last modified (=approximate) KKT conditions

$$\nabla f(x) + \textstyle\sum_{i=1}^m \lambda_i \nabla g_i(x) = 0 \qquad \text{("force balance")}$$

$$\forall_i : \; g_i(x) \le 0 \qquad \text{(primal feasibility)}$$

$$\forall_i : \; \lambda_i \ge 0 \qquad \text{(dual feasibility)}$$

$$\forall_i : \; \lambda_i g_i(x) = -\mu \qquad \text{(complementary)}$$

can be written as the $n+m$-dimensional equation system

$$r(x,\lambda) = 0 \; , \quad r(x,\lambda) := \begin{pmatrix} \nabla f(x) + \lambda^\top \nabla g(x) \\ -\mathrm{diag}(\lambda) g(x) - \mu \mathbf{1}_n \end{pmatrix}$$

- Newton method to find the root $r(x,\lambda) = 0$

$$\begin{pmatrix} x \\ \lambda \end{pmatrix} \leftarrow \begin{pmatrix} x \\ \lambda \end{pmatrix} - \nabla r(x,\lambda)^{-1} r(x,\lambda)$$

$$\nabla r(x,\lambda) = \begin{pmatrix} \nabla^2 f(x) + \sum_i \lambda_i \nabla^2 g_i(x) & \nabla g(x)^\top \\ -\mathrm{diag}(\lambda) \nabla g(x) & -\mathrm{diag}(g(x)) \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$$

Primal-Dual interior-point Newton Method

- The method requires the Hessians $\nabla^2 f(x)$ and $\nabla^2 g_i(x)$
  - One can approximate the constraint Hessians $\nabla^2 g_i(x) \approx 0$
  - Gauss-Newton case: $f(x) = \phi(x)^\top \phi(x)$ only requires $\nabla\phi(x)$

- This primal-dual method does a joint update of both
  – the solution x
  – the lagrange multipliers (constraint forces) λ
  No need for nested iterations, as with penalty/barrier methods!

- The above formulation allows for a duality gap μ; choose $\mu = 0$ or consult Boyd how to update on the fly (sec 11.7.3)

- The **feasibility constraints** $g_i(x) \le 0$ and $\lambda_i \ge 0$ need to be handled explicitly by the root finder (the line search needs to ensure these constraints)

Planned Outline

- Gradient-based optimization (1st order methods)
  - plain grad., steepest descent, conjugate grad., Rprop, stochastic grad.
  - adaptive stepsize heuristics
- Constrained Optimization
  - squared penalties, augmented Lagrangian, log barrier
  - Lagrangian, KKT conditions, Lagrange dual, log barrier $\leftrightarrow$ approx. KKT
- **2nd order methods**
  - Newton, Gauss-Newton, Quasi-Newton, (L)BFGS
  - constrained case, primal-dual Newton
- Special convex cases
  - Linear Programming, (sequential) Quadratic Programming
  - Simplex algorithm
  - relation to relaxed discrete optimization
- Black box optimization ("0th order methods")
  - blackbox stochastic search
  - Markov Chain Monte Carlo methods
  - evolutionary algorithms