Unconstrained optimization

Ludovic Stourm

June 23, 2025

1 Introduction

This document describes various methods to numerically minimize a function $f(x)$:

$$\min_x \ f(x) \tag{1}$$

We use the notations:

- f: objective function
- x: argument of f (vector of dimension $[J \times 1]$)
- ∇f: gradient of f ($[J \times 1]$ vector of 1st-degree derivatives)
- H_f: Hessian of f ($[J \times J]$ matrix of 2nd-degree derivatives)
- d: Direction of an update ($[J \times 1]$ vector)
- α: Step size of an update ($[J \times 1]$ vector)
- $s = \alpha d$: Update ($[J \times 1]$ vector)

We consider different methods. Some of them requires one to evaluate not only f but also its gradient ∇f, and some even the Hessian H_f. This is summarized in the table below:

Method	Requires gradient	Requires Hessian
Newton-Raphson	✓	✓
Gradient ascent	✓	
BFGS	✓	
Nelder-Mead		

2 Newton-Raphson

1. Set $k \leftarrow 0$
2. Initialize starting point x_0
3. Evaluate $f(x_k)$, $g_k \leftarrow \nabla f(x_k)$, and $H_k \leftarrow H_f(x_k)$
4. Set direction of update: $d_k \leftarrow -H_k^{-1} g_k$.
5. Set step size $\alpha_k \leftarrow 1$ or find a better value through a line search procedure.
6. Update point: $x_{k+1} \leftarrow x_k + \alpha_k d_k$
7. Set $k \leftarrow k + 1$ and go to step 3.

3 Gradient descent

1. Set $k \leftarrow 0$
2. Initialize starting point x_0
3. Evaluate $f(x_k)$ and $g_k \leftarrow \nabla f(x_k)$
4. Set direction of update: $d_k \leftarrow -g_k$.
5. Set step size α_k a priori (constant step size α), or find a better value through a line search procedure.
6. Update point: $x_{k+1} \leftarrow x_k + \alpha_k d_k$
7. Set $k \leftarrow k + 1$ and go to step 3.

4 BFGS (quasi-Newton method)

This method uses evaluations of f and its gradient ∇f. The Hessian H (or rather, its inverse H^{-1}) is "approximated" at each iteration (although there is no guarantee that it converges to the true Hessian!).

1. Set $k \leftarrow 0$
2. Initialize starting point x_0, inverse Hessian H_0^{-1}
3. Evaluate $f(x_k)$ and $g_k \leftarrow \nabla f(x_k)$
4. Set the direction of update $d_k = -H_k^{-1} g_k$.
5. Find a "good" step size α_k through a line search procedure (more on this later).
6. Set update $s_k \leftarrow \alpha_k d_k$

7. Update point $x_{k+1} = x_k + s_k$
8. Update Hessian $H_{k+1}^{-1} = H_k^{-1} + \frac{(s_k' y_k + y_k' H_k^{-1} y_k)(s_k s_k')}{(s_k' y_k)^2} - \frac{H_k^{-1} y_k s_k' + s_k y_k' H_k^{-1}}{s_k' y_k}$.
   where $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
9. Set $k \leftarrow k+1$ and go to step 3.

5 Nelder-Mead

This method only uses evaluations of f and does not require an evaluation of the gradient or the Hessian (which may not even exist). This algorithm **minimizes** function f.

1. Initialize $\alpha \leftarrow 1$, $\gamma \leftarrow 2$, $\rho \leftarrow 0.5$, $\sigma \leftarrow 0.5$.
2. Evaluate $f(x_0)$
3. Initial simplex: define $x_j = x_0 + \epsilon \times u_j$ and evaluate $f(x_j)$ for each dimension j (where u_j is the corresponding unit vector in that direction and ϵ is some small value)
4. Define the $(J+1)$ initial vertices as $[x_0, x_1, ..., x_J]$
5. Sort the $(J+1)$ vertices x_j by increasing values of $f(x_j)$
6. Compute centroid based on the J first vertices $xo \leftarrow 1/J \times \sum_{j=0}^{J-1} x_j$
7. Compute $xr \leftarrow xo + \alpha(xo - x_J)$
8. If $f(xr) \geq f(x_0)$ and $f(xr) < f(x_{J-1})$:
   (a) $x_J \leftarrow xr$ (reflect)
   (b) Go to 5.
9. If $f(xr) < f(x_0)$:
   (a) Compute $xe \leftarrow xo + \gamma(xr - xo)$
   (b) If $f(xe) < f(xr)$:
      i. $x_J \leftarrow xe$ (expand)
      ii. Go to 5.

      Else:
      i. $x_J \leftarrow xr$ (reflect)
      ii. Go to 5.
10. If $f(xr) < f(x_J)$:
   (a) $xc \leftarrow xo + \rho(xr - xo)$
   (b) If $f(xc) < f(xr)$

i. $x_J \leftarrow xc$ (contract outside)

ii. Go to 5.

Else: $xc \leftarrow xo + \rho(x_J - xo)$

(a) $xc \leftarrow xo + \rho(xr - xo)$

(b) If $f(xc) < f(x_J)$

i. $x_J \leftarrow xc$ (contract outside)

ii. Go to 5.

11. For all j in $1, ..., J$: do $x_j \leftarrow x_0 + \sigma(x_j - x_0)$ (shrink)

12. Go to 5.