Writing an Optimizer

Joe Volzer

October 3, 2020

Chapter 1

The Beginning

1.1 Why write this document?

I originally wrote this optimizer in MATLAB as a graduate student. The code works but I would like to make it better- and port it to Python. This document will be used as a point of reference.

Chapter 2

Psuedocode

2.1 Main method

input : A function $f : \mathbb{R}^n \to \mathbb{R}$, steptol, gradtol, maxiter, x_0
output: A global extremum x_*.
initialization
τ = steptol
$\alpha = 10^{-4}$
while $n <$ *maxiter* **or not** *StoppingCriteria*() **do**
  $g_n = \nabla f(x_n)$
  Solve $-H_n s_n = g_n$ for s_n
  $x_{n+1} = x_n + s_n$
  $f_{n+1} = f(x_{n+1})$
  if $f_{n+1} > f_n + \alpha g_n^T s_n$ **then**
    x_{n+1}, f_{n+1}, flag = BacktrackingLineSearch(x_n, s_n, g_n, f_{n+1}, f_n)
    if *flag is TRUE* **then**
      x_{n+1}, f_{n+1} = TrustRegionSubproblem(x_n,g_c, H_c, s_n,δ,τ)
    end
  end
  Apply BFGS Update to H_n
end

Algorithm 1: Main loop

2.2 Back tracking line search

input : x_n, s_n, f_+
output: Some stuff.
$\alpha = 10^{-4}$
$d = g_c^T s_n$
$\lambda = 1$
$iter = 0$
while $iter < maxiter$ **and** $f_+ > f_c + \alpha\lambda d$ **do**
    $x_+ = x_c + \lambda s_n$
    $f_+ = f(x_+)$
    if $\lambda = 1$ **then** `/* Quadratic interpolation */`
        $\hat{\lambda} = \frac{-d}{2(f_+ - f_c - d)}$
    else `/* cubic interpolation */`
        Solve $\begin{bmatrix} \lambda^3 & \lambda^2 \\ \lambda_-^3 & \lambda_-^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f_+ - f_c - d\lambda \\ f_- - f_c - d\lambda_- \end{bmatrix}$
        $\hat{\lambda} = \frac{-a_2 + \sqrt{a_2^2 - 3a_1 d}}{3a_1}$
    end
    $\hat{\lambda} = \min(\max(\hat{\lambda}, 0.1\lambda), 0.5\lambda)$
    $\lambda_- = \lambda$
    $\lambda = \hat{\lambda}$
    $f_- = f_+$
    $iter = iter + 1$
end

Algorithm 2: Line Search

2.3 Trust region

input : Things
output: A usefule thing
$\alpha = g_c^T g_c$
$\beta = g_c^T H_c g_c$
$\gamma = \frac{\alpha^2}{\beta |g_c^T s_n|}$
$\eta = .2 + .8\gamma$
cauchystep $= -\frac{\alpha}{\beta} g_c$
cauchylen = norm(cauchystep)
$\hat{n} = \eta s_n$-cauchystep
while *flags* **and** $i < 50$ **do**
    do the things
end

Algorithm 3: Trust region

2.3.1 Trust Region Update

trupdate

2.3.2 Double Dog Leg

double dog leg

2.4 Helper Methods

2.4.1 BFGS Update

input : H,xc,xplus,gc,gplus,eta
output: H, skipupdate
s = $x_+ - x_c$
y = $g_+ - g_c$
skipupdate = true
if $y^T s \geq \sqrt{\epsilon}||s|| \times ||y||$ **then**
    t = Hs
    if $|y - t| >= \sqrt{\eta}\max(||g_c||, ||g_+||)$ **then**
        skipupdate = false
        $H = H + \frac{yy^T}{y^T s} - \frac{t*(s^T H)}{s^T t}$
        $H = \frac{H+H'}{2}$
    end
end

2.5 Stopping

input : xc,xplus,fplus,Gc,itrlimit,steptol,gradtol,itrcount
output: minimized flag.
minimized = false
scaledgrad = (abs(Gc).* abs(xplus))./abs(fplus)
stepdist = abs(xplus-xc)./abs(xplus)
if *max(scaledgrad) <= gradtol* **then**
    minimized = true
    msgbox('The norm of the scaled gradient is within the desired tolerance.',msgtitle)
else if *max(stepdist) <= steptol* **then**
    minimized = true
    msgbox('The distance between the last two steps was with the step tolerance. This is either a minimizer or we are stuck.'... ,msgtitle)
else if *itrcount >= itrlimit* **then**
    minimized = true
    msgbox('The maximum iteration limit has been attained. This may not be a good candidate for a minimizer.',... msgtitle,'error')