Centro de Investigación en Matemáticas, A.C.
Optimización

Tarea 8

José Miguel Saavedra Aguilar

Abstract

In this homework we present the classic Trust-Region method and the Retrospective Filter Trust-Region method.

1 Introduction

For the unconstrained optimization problem:

$$x^* = \underset{x \in \mathrm{dom}(f)}{\arg\min} \quad f(x) \tag{1}$$

a descent directions algorithm updates x_k as

$$x_{k+1} = x_k + d_k$$

On homework 4, 5 and 7, we have explored the line search methods where given a descent direction d_k one updates $d_k = \alpha_k d_k$ for some α_k. Suppose we haven't selected a descent direction d_k, however we want $\|d_k\| \leq \Delta : k$ for a trust-region $\Delta_k > 0$. Trust-region methods consist on selecting d_k as:

$$\begin{aligned} d_k = \ & \underset{d \in \mathbb{R}^n}{\arg\min} \quad m_k(d) := \frac{1}{2} d^\top B_k d + g_k^\top d + f(x_k) \\ & \text{s.t.} \qquad \|d\| \leq \Delta_k \end{aligned} \tag{2}$$

Where $g_k = \nabla f(x_k)$ and B_k is a SPD matrix.

1.1 Classic Trust-Region method

For the classic Trust-Region method, one should compute the *trust ratio* ρ as:

$$\rho_k = \frac{f(x_k + d_k) - f(x_k)}{m_k(d_k) - m_k(0)}$$

Based on the value of ρ_k and given $\eta_1 < \eta_2$ and $\lambda_1 < 1 < \lambda_2$ we shall update Δ_k as follows:

$$\Delta_{k+1} = \begin{cases} \lambda_1 \Delta_k & \rho_k < \eta_1 \\ \Delta_k & \rho_k \in [\eta_1, \eta_2] \\ \lambda_2 \Delta_k & \rho_k > \eta_2 \end{cases}$$

Finally, we update x_k as:

$$x_{k+1} = \begin{cases} x_k & \rho_k < \eta_1 \\ x_k + d_k & \rho_k \geq \eta_1 \end{cases}$$

1.2 Retrospective Filter Trust-Region method

A retrospective filter Trust-Region method was presented by Bastin et al. in 2008[1]. The main difference between this algorithm and classic Trust-Region methods is one shall update Δ_k retrospectively, this is, one calculates a retrospective trust ratio $\hat{\rho}_k$ as:

$$\hat{\rho}_k = \frac{f(x_{k-1}) - f(x_k)}{m_k(0) - m_k(-d_{k-1})}$$

then, we choose Δ_k based on values $\hat{\eta}_1, \hat{\eta}_2, \lambda_1, \lambda_2, \lambda_2$ as:

$$\Delta_k \in \begin{cases} [\lambda_1 \Delta_{k-1}, \lambda_2 \Delta_{k-1}) & \hat{\rho}_k < \hat{\eta}_1 \\ [\lambda_2 \Delta_{k-1}, \Delta_{k-1}) & \hat{\rho}_k \in [\hat{\eta}_1, \hat{\eta}_2] \\ [\Delta_{k-1}, \lambda_3 \Delta_{k-1}) & \hat{\rho}_k > \hat{\eta}_2 \end{cases}$$

Then, one computes the trust ratio:

$$\rho_k = \frac{f(x_k + d_k) - f(x_k)}{m_k(d_k) - m_k(0)}$$

and update x_k. The Retrospective Trust-Region method was presented by Lu and Chen [4] and uses filters to update x_k. We shall define filters:

Definition 1 (Domination). *We say a vector $v \in \mathbb{R}^n$ dominates another vector $u \in \mathbb{R}^n$ if*

$$|v_i| \leq |u_i|$$

for all $i = 1, \ldots, n$.

Definition 2 (Filter). *A filter F is a set of vectors $F = \{v_1, \ldots, v_k\}$ such that no vector $v_i \in F$ dominates any other $v_j \in F, i \neq j$.*

Definition 3 (Acceptability). *We say a vector w is acceptable for a filter $F = \{v_1, \ldots, v_k\}$ if and only if for all $v_i \in F$ there exists $j \in \{1, \ldots, n\}$ such that*

$$|w_j| \leq \left|v_{ij}\right| - \gamma_g \left\|v_i\right\|$$

for some $\gamma_g \in (0, 1/\sqrt{n})$.

Now, we shall update x_k as follows:

- If $\rho_k \geq \eta_1$, then $x_{k+1} = x_k + d_k$.
- If $\rho_k < \eta_1$ and $\nabla f(x_k + d_k)$ is acceptable for filter F, then $x_{k+1} = x_k + d_k$ and $\nabla f(x_k + d_k)$ is added to the filter F.
- If $\rho_k < \eta_1$ and $\nabla f(x_k + d_k)$ is not acceptable for filter F, then $x_{k+1} = x_k$.

For the implementation of this method, we will take the trust region update proposed by [3] just as [4] do:

$$\Delta_k = \begin{cases} \lambda_1 \|d_{k-1}\|, & \hat{\rho}_k < \hat{\eta}_1 \\ \Delta_{k-1} & \hat{\rho}_k \in [\hat{\eta}_1, \hat{\eta}_2] \\ \max\{\lambda_2 \|d_{k-1}\|, \Delta_{k-1}\}, & \hat{\rho}_k > \hat{\eta}_2 \end{cases}$$

1.3 Dogleg

Solving (2) is very complicated, even with constrained optimization tools it would be quite expensive to solve every iteration, so we approximate the solution d_k. Note if $\Delta << 1$, we have $d^\top B d \approx 0$ and the model can be reduced to

$$m_k(d) \approx g_k^\top d + f(x_k)$$

with minimum

$$d_k = -\frac{\Delta_k}{\|g_k\|} g_k \tag{3}$$

Based on this, we shall consider the solution can be approximated by a vector parallel to the gradient g_k, and we consider

$$m_k(\alpha g_k) = \alpha^2 \frac{1}{2} g_k^\top B_k g_k + \alpha g_k^\top g_k + f(x_k)$$

If we minimize $m_k(\alpha g_k)$ for α we get

$$\alpha = -\frac{g_k^\top g_k}{g_k^\top B g_k}$$
$$p^U = -\frac{g_k^\top g_k}{g_k^\top B g_k} g_k$$

p^U is the unconstrained minimizer on the gradient direction. Now, since $p^N = B_k^{-1} g_k$ is the unconstrained minimizer of $m_k(d)$, we have if $\|p^N\| \leq \Delta$, the exact solution of (2) is p^N. The dogleg approximation of d_k consists on taking a convex combination of p^U and p^N as:

$$d_k = (1-t)p^U + tp^N$$

for t such that $\|d_k\| = \Delta$ whenever $\|p^U\| < \Delta$ and $\|p^N\| > \Delta$, taking (3) when $\|p^U\| \geq \Delta$ and p^N when $\|p^N\| \leq \Delta$.

2 Algorithm

In order to approximate (2), we will compare the following two algorithms:

Algorithm 1: Dogleg approximation of (2).

Input: Δ_k, g_k, B_k
Output: d_k
Compute $p^U = -\frac{g_k^\top g_k}{g_k^\top B_k g_k}$;
if $\left\|p^U\right\| > \Delta_k$ **then**
  $d_k \leftarrow \frac{\Delta_k}{\|p^U\|} p^U$;
else
  Compute $p^N = -B_k^{-1} g_k$;
  if $\left\|p^N\right\| > \Delta_k$ **then**
    $d_k \leftarrow p^N$;
  else
    Solve for t $\left\|(1-t)p^U + tp^N\right\| = \Delta_k$;
    $d_k \leftarrow (1-t)p^U + tp^N$
  end
end

Algorithm 2: Not so naive approximation of (2).

Input: Δ_k, g_k, B_k
Output: d_k
Compute $p^U = -\frac{g_k^\top g_k}{g_k^\top B_k g_k}$;
if $\left\|p^U\right\| > \Delta_k$ **then**
  $d_k \leftarrow \frac{\Delta_k}{\|p^U\|} p^U$;
else
  Compute $p^N = -B_k^{-1} g_k$;
  if $\left\|p^N\right\| > \Delta_k$ **then**
    $d_k \leftarrow p^N$;
  else
    $d_k \leftarrow \frac{\Delta_k}{\|p^N\|} p^N$;
  end
end

Algorithm 2 is not precisely the requested *naive* algorithm, however it is similar to it except for the case $\left\|p^U\right\| < \Delta_k < \left\|p^N\right\|$ instead of taking $d_k = p^U$, we take the best approximation in the direction of p^N, that is $\frac{\Delta_k}{\|p^N\|} p^N$, thus the name *not so naive*.
The Trust-Region method is given by:

Algorithm 3: Trust-Region algorithm.

Input: f, x_0
Output: x^*

$k \leftarrow 0$;
while $\|\nabla f(x_k)\| > 0$ **do**
  Approximate d_k, the solution of (2);
  Compute the trust ratio $\rho_k = \frac{f(x_k+d_k)-f(x_k)}{m_k(d_k)-m_k(0)}$;
  if $\rho_k < \eta_1$ **then**
    $\Delta_{k+1} \leftarrow \lambda_1 \Delta_k$
  else
    $x_{k+1} \leftarrow x_k$;
    if $\rho_k > \eta_2$ **then** $\Delta_{k+1} \leftarrow \lambda_2 \Delta_k$;
  end
  $k \leftarrow k + 1$;
end

The Retrospective Filter Trust-Region method is given by:

Algorithm 4: Retrospective Filter Trust-Region algorithm.

Input: f, x_0
Output: x^*

$k \leftarrow 0$;
Initialize an empty filter F;
while $\|\nabla f(x_k)\| > 0$ **do**
  Approximate d_k, the solution of (2);
  Compute the trust ratio $\rho_k = \frac{f(x_k+d_k)-f(x_k)}{m_k(d_k)-m_k(0)}$;
  if $\rho_k \geq \eta_1$ **then**
    $x_{k+1} \leftarrow x_k + d_k$
  else if $\nabla f(x_k + d_k)$ *is acceptable for* F **then**
    $x_{k+1} \leftarrow x_k + d_k$;
    add g_{k+1} to filter F;
  end
  $k \leftarrow k + 1$;
  Compute the retrospective trust ratio $\hat{\rho}_k = \frac{f(x_k)-f(x_{k-1})}{m_k(0)-m_k(-d_{k-1})}$;
  if $\hat{\rho}_k < \hat{\eta}_1$ **then**
    $\Delta_k \leftarrow \lambda_1 \|d_{k-1}\|$;
  else if $\hat{\rho}_k < \hat{\eta}_2$ **then**
    $\max\{\lambda_2 \|d_{k-1}\|, \Delta_{k-1}\}$
  end
end

3 Results

Algorithms 3 and 4 were implemented in Julia[2] with the dogleg approximation 1, as well as 2 for algorithm 3. In order to test these algorithms, as well as the modified Newton's descent direction with backtracking. We will consider the Rosenbrock function with $n = 100$, the Wood function and the Branin function. We take 30 random starting points:

$$x_0 = x^* + \zeta$$

for $\zeta_i \sim \mathrm{U}(-2, 2)$ an n-dimensional random variable. We present the average number of iterations and the average time elapsed for each algorithm to solve for x^*.

	Trust-Region (dogleg)	Trust-Region (not so naive)	Newton's method	RFTR (dogleg)
Rosenbrock	0.15389	0.16197	0.07916	0.1753
Wood	0.00712	0.00888	0.00408	0.00789
Branin	0.02269	0.02066	0.0053	0.02831

Table 1: Average time elapsed to solve for x^* starting from x_0 with algorithms 3 and 4.

	Trust-Region (dogleg)	Trust-Region (not so naive)	Newton's method	RFTR (dogleg)
Rosenbrock	101.03333	102.23333	45.5	111.5
Wood	17.03333	19.1	15.4	19.86667
Branin	3742.33333	3422.76667	186	4199.5

Table 2: Average iterations taken to solve for x^* starting from x_0 with algorithms 3 and 4.

4 Results discussion and conclusions

We have the worst performing algorithm was 4, both in time elapsed and average iterations taken to solve (1) for all three functions. The best performing algorithm was Newton's method with backtracking, which should take the quadratic convergence of Newton's method and search for a step length so we stay in a region that satisfies the conditions that satisfy the convergence to a minimum. Performance of algorithm 3 is pretty similar with 1 and 2 approximations of (2). On average, dogleg is better for the Rosenbrock and Wood functions, but worse for the Branin function. Now, if doing line search was not viable (i.e. f takes very long to evaluate), trust region methods should be faster than Netwon's method.

References

[1] F. Bastin, V. Malmedy, M. Mouffe, P. L. Toint, and D. Tomanos, "A retrospective trust-region method for unconstrained optimization," *Mathematical Programming*, vol. 123, no. 2, pp. 395–418, Dec. 2008. [Online]. Available: https://doi.org/10.1007/s10107-008-0258-1

[2] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://epubs.siam.org/doi/10.1137/141000671

[3] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Society for Industrial and Applied Mathematics, Jan. 2000. [Online]. Available: https://doi.org/10.1137/1.9780898719857

[4] Y. Lu and Z. Chen, "A retrospective filter trust region algorithm for unconstrained optimization," *Applied Mathematics*, vol. 01, no. 03, pp. 179–188, 2010. [Online]. Available: https://doi.org/10.4236/am.2010.13022