Optimization HW2

Gan Haochen 1600017789

April 2020

1 Gradient-type Optimization

Gradient-type Optimization is defined as the iteration procedure:

$$d_k = -g_k + \beta d_{k-1} \tag{1}$$

where $d_k := x_{k+1} - x_k$, $g_k := \nabla f(x_k)$. And using line-search to find an α such that

$$x_{k+1} = x_k + \alpha d_k \tag{2}$$

1.1 Global BB

For Global Barzilai-Borwein, there is always $\beta = 0$, but for each iteration the initial value of α_k is given by the last time step:

$$\alpha_{k+1} = -\frac{(\lambda_k g_k^t g_k)}{(g_k^t y_k)} \tag{3}$$

where $\lambda_k = \frac{1}{\alpha_k}$

Note that for general non-linear optimization there is no guarantee for global BB method, so non-monotone line search is introduced as:
1. initialize $0 < c_1 < c_2 < 1, C_0 \leftarrow f\left(x^0\right), Q_0 \leftarrow 1, \eta < 1, k \leftarrow 0$
2. while not converged do
3. find α_k by line-search, to satisfy the modified Armijo condition:
sufficient decrease: $f\left(x^{(k)} + \alpha_k d^{(k)}\right) \leq C_k + c_1 \alpha_k \nabla f_k^T d^{(k)}$
4. $x^{k+1} \leftarrow x^{(k)} + \alpha_k d^{(k)}$
5. $Q_{k+1} \leftarrow \eta Q_k + 1, C_{k+1} \leftarrow \left(\eta Q_k C_k + f\left(\boldsymbol{x}^{k+1}\right)\right) / Q_{k+1}$

We should note that if $\eta = 1$, then $C_k = \frac{1}{k+1} \sum_{j=0}^{k} f_j$,since $\eta < 1, C_k$ is a weighted sum of all past f_j, more weights on recent f_j.

1.2 Conjugate Gradient Methods

Conjugate Gradient Methods (CG) generally is

$$d_k = -g_k + \beta d_{k-1} \tag{4}$$

Typical CG methods include FR method and PRP method. where FR:

$$\beta_k^{FR} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \tag{5}$$

And for PRP:

$$\beta_k^{PRP} = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}} \tag{6}$$

FR-PRP mixture method:

$$\beta_k = \begin{cases} -\beta_k^{\mathrm{FR}} & \text{if } \beta_k^{\mathrm{PRP}} < -\beta_k^{\mathrm{FR}} \\ \beta_k^{\mathrm{PRP}} & \text{if } \left|\beta_k^{\mathrm{PRP}}\right| \leq \beta_k^{\mathrm{FR}} \\ \beta_k^{\mathrm{FR}} & \text{if } \beta_k^{\mathrm{PRP}} > \beta_k^{\mathrm{FR}} \end{cases} \tag{7}$$

A special method is also implemented as follow:
Step 1. Set $k = 1, s_1 = -g_1$
Step 2. Line Search. Compute $x_{k+1} = x_k + a_k s_k$; set $k = k + 1$
Step 3. If $g_k^T g_k < \epsilon$, then stop; otherwise, go to Step 4
Step 4. If $k > n > 2$, go to Step 8; otherwise, go to Step 5
Step 5. Let $t_k = s_{k-1}^T H_k s_{k-1}, v_k = g_k^T H_k g_k$, and $u_k = g_k^T H_k s_{k-1}$

Step 6. If

$$t_k > 0, \quad v_k > 0, \quad 1 - u_k^2 / (t_k v_k) \geq 1/(4r)$$

and

$$\left(v_k / g_k^T g_k\right) / \left(t_k / s_{k-1}^T s_{k-1}\right) \leq r, \quad r > 0$$

then go to Step 7; otherwise, go to Step 8 Step 7. Let

$$s_k = \left[\left(u_k g_k^T s_{k-1} - t_k g_k^T g_k\right) g_k + \left(u_k g_k^T g_k - v_k g_k^T s_{k-1}\right) s_{k-1}\right] / w_k$$

where $w_k = t_k v_k - u_k^2$; go to Step 2 Step 8. Set x_k to x_1; go to Step 1
where Hessian H_k should not be computed thus an alternative is as follow:
Step 5. Let

$$\begin{aligned} t_k &= d_{k-1}^T \left(\nabla f\prime(x_k + \delta d_{k-1}) - g_k\right) / \delta \\ u_k &= g_k^T \left(\nabla f\prime(x_k + \delta d_{k-1}) - g_k\right) / \delta \\ v_k &= g_k^T \left[\nabla f\prime(x_k + \gamma g_k) - g_k\right] / \gamma \end{aligned}$$

where δ and γ are suitable, small, positive numbers.

2 Implementation

2.1 Stopping Criterion

In our implementation the stopping criterion is setted as:

$$||g_k||_2 < \epsilon(1 + |f_k|), \quad \epsilon = 1\mathrm{e} - 6 \tag{8}$$

Notice due to the magnitude of the function f, which could be very large or very small in some cases, an absolute criterion is not proper because of the machine accuracy.

2.2 Numerical Adjustment

Note that in mixture methods we have approximate the Hessian by difference of gradient. we have adopted the evaluation in the original paper:

$$\epsilon = \sqrt{\eta}, \quad r = 1/\sqrt{\eta}, \quad \delta = \sqrt{\eta}/\sqrt{s_{k-1}^T s_{k-1}}, \quad \gamma = \sqrt{\eta}/\sqrt{g_k^T g_k} \tag{9}$$

2.3 C++ Eigen Library Used

A library for C++, Eigen is used for the linear system calculation. And VectorXd class is also very useful in the program.

3 Results

Trigo(n) means Trigonometric Function with n dimension. Where the initial value is $x_0 = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})^T$
ExtendedPowell(n) means Extended Powell Function with n dimension. Where the initial value is $x_0 = (3, -1, 0, 3, 3, -1, 0, 3, ...)^T$
Tridig(n) means Tridiganol Function with n dimension. Where the initial value is $x_0 = (1, 1, 1, .., 1)$.
MS(n) means Matrix Square Root problem with $\sqrt{n}$ dimension. Note that $n = 1000$ does not make sense for this problem.

Timing of certain jobs

Problem	GBB	FR	PRP	FR-PRP	HU1991
$Trigo(100)$	15.651	54.103	68.234	67.963	539.975
$Trigo(1000)$	101.232	291.394	351.2	342.3	/
$Trigo(10000)$	/	/	/	/	/
$EP(100)$	137.656	75.216	488.517	517.407	431.16
$EP(1000)$	416.5	244.21	1018.73	1262.5	978.25
$EP(10000)$	812.35	645.23	1752.2	1642.2	1532.6
$Tridig(100)$	0.765	5.529	11.311	11.345	3.47
$Tridig(1000)$	156.056	75.107	417.041	429.283	?
$Tridig(10000)$	0.27	0.503	0.502	0.401	0.098
$MS(100)$	452.2	354.2	425.2	445.6	237.6
$MS(10000)$	/	/	/	/	/

3.1 Analysis

Trigo(10000) and MS(10000) could not be optimized on my PC (Core i7) in a acceptable time interval because every evaluation of the function needs $O(n^2) = 1e+8$ times computation, which is unacceptable for the optimization.

GBB method converges very quickly at the beginning, but it is hard for it to reach the optimal when the function is close to end. On the other hand, CG-type methods have a very satisfying convergence rate near the end.

iteration number

Problem	GBB	FR	PRP	FR-PRP	HU1991
$Trigo(100)$	79	112	152	152	369
$Trigo(1000)$	185	554	564	587	/
$Trigo(10000)$	/	/	/	/	/
$EP(100)$	24564	3400	18665	19543	248224
$EP(1000)$	23345	5620	15168	23546	12352
$EP(10000)$	59166	7542	17422	28545	15985
$Tridig(100)$	147	269	495	495	215
$Tridig(1000)$	26379	22158	103422	187323	195280
$Tridig(10000)$	21350	4120	25854	26563	3540
$MS(100)$	2654	452	545	456	256
$MS(10000)$	/	/	/	/	/

function evoking number

Problem	GBB	FR	PRP	FR-PRP	HU1991
$Trigo(100)$	262	1000	2157	3314	9510
$Trigo(1000)$	425	1452	3454	6542	/
$Trigo(10000)$	/	/	/	/	/
$EP(100)$	75190	48034	334305	640694	15682
$EP(1000)$	74938	40015	151938	200193	22563
$EP(10000)$	25546	7542	17422	28545	15985
$Tridig(100)$	644618	652586	660554	495	215
$Tridig(1000)$	26379	212295	81264	165165	185625
$Tridig(10000)$	58453	1586201	1465875	1897523	100595
$MS(100)$	2654	452	545	456	256
$MS(10000)$	/	/	/	/	/

Non-monotone line-search is critical for the convergence of GBB type methods, and the method we use has better attribution than the demanded one.

Restart strategy is critical for the convergence of CG-type methods. When every N steps or line-search fails, the decent direction should be reset as $d_k = -g_k$, and a new line-research should be conducted.

And we did not list the optimal values because all functions here have a theoretical optimal value as zero.