优化实用算法: 第三次作业

2022 年 5 月 17 日

指导老师：徐翔

徐圣泽 3190102721

Problem 1

1. Try to prove that when $\phi = \phi_k^c = \frac{1}{1-\mu_k}$ where $\mu_k = \frac{(s_k^T B_k s_k)(y_k^T H_k y_k)}{(s_k^T y_k)^2}$, the Broyden class

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k (s_k^T B_k s_k) v_k v_k^T$$

where

$$v_k = (\frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k})$$

becomes sigular.

解：要证 B_{k+1} 为奇异矩阵，只需找到 $B_{k+1}x = 0$ 的非零解 x。
取 $x = s_k - \rho_k H_k y_k$，其中 $\rho_k = \frac{y_k^T s_k}{y_k^T H_k y_k}$ 且有关系 $B_{k+1} s_k = y_k$ 和 $B_k H_k = I$ 成立.
下面代入 x，化简证明 $B_{k+1}x = 0$。

$$\begin{aligned}
B_{k+1}x &= B_{k+1}(s_k - \rho_k H_k y_k) \\
&= y_k - \rho_k (B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k (s_k^T B_k s_k) v_k v_k^T) H_k y_k \\
&= (1-\rho_k) y_k + \rho_k \frac{B_k s_k s_k^T y_k}{s_k^T B_k s_k} - \rho_k \frac{y_k y_k^T H_k y_k}{y_k^T s_k} - \rho_k \phi_k (s_k^T B_k s_k) v_k v_k^T H_k y_k \\
&= [(1-\rho_k) y_k^T s_k - \rho_k y_k^T H_k y_k] \frac{y_k}{y_k^T s_k} + \rho_k s_k^T y_k \frac{B_k s_k}{s_k^T B_k s_k} - \rho_k \phi_k (s_k^T B_k s_k)(v_k^T H_k y_k) v_k \\
&= -\rho_k y_k^T s_k \frac{y_k}{y_k^T s_k} + \rho_k s_k^T y_k \frac{B_k s_k}{s_k^T B_k s_k} - \rho_k \phi_k (s_k^T B_k s_k)(v_k^T H_k y_k) v_k \\
&= -\rho_k y_k^T s_k (\frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}) - \rho_k \phi_k (s_k^T B_k s_k)(v_k^T H_k y_k) v_k \\
&= -\rho_k [y_k^T s_k + \phi_k (s_k^T B_k s_k)(v_k^T H_k y_k)] v_k \\
&= -\rho_k [y_k^T s_k + \phi_k (s_k^T B_k s_k)(\frac{y_k^T H_k y_k}{y_k^T s_k} - \frac{s_k^T y_k}{s_k^T B_k s_k})] v_k \\
&= -\rho_k [(1-\phi_k) s_k^T y_k + \phi_k (s_k^T B_k s_k) \frac{y_k^T H_k y_k}{y_k^T s_k}] v_k
\end{aligned}$$

代入 $\phi = \phi_k^c = \frac{1}{1-\mu_k}$ 和 $\mu_k = \frac{(s_k^T B_k s_k)(y_k^T H_k y_k)}{(s_k^T y_k)^2}$，可以得到：

$$\begin{aligned}
B_{k+1}x &= -\rho_k \frac{-\mu_k s_k^T y_k + (s_k^T B_k s_k) \frac{y_k^T H_k y_k}{y_k^T s_k}}{1-\mu_k} v_k \\
&= 0
\end{aligned}$$

因此，当 $\phi = \phi_k^c = \frac{1}{1-\mu_k}$ 时，B_{k+1} 为奇异矩阵。

Problem 2

2. Using BFGS method to minimize the extended Rosenbrock function

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2],$$

with $x_0 = [-1.2, 1, \cdots, -1.2, 1]^T$**,** $x^\star = [1, 1, \cdots, 1, 1]^T$ **and** $f(x^\star) = 0$**. Try different** $n = 6, 8, 10$ **and** $\epsilon = 10^{-5}$**. Moreover, using BFGS method to minimize the Powellsingular function**

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$$

with $\epsilon = 10^{-5}$**,** $x_0 = [3, -1, 0, 1]^T$**,** $x^\star = [0, 0, 0, 0]$ **and** $f(x^\star) = 0$**.**

解：首先我们写出题目中两函数及其梯度：

```
function y=f(x)
n=length(x);
y=0;
for i=1:n-1
        y=y+100*(x(i+1)-x(i)^2)^2+(1-x(i))^2;
end
end
```

```
function y=g(x)
y=(x(1)+10*x(2))^2+5*(x(3)-x(4))^2+(x(2)-2*x(3))^4+10*(x(1)-x(4))^4;
end
```

```
function y=Gradf(x)
n=length(x);
y=zeros(n,1);
y(1)=-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1));
for i=2:n-1
        y(i)=-400*x(i)*(x(i+1)-x(i)^2)-2*(1-x(i))+200*(x(i)-x(i-1)^2);
end
y(n)=200*(x(n)-x(n-1)^2);
end
```

```
function y=Gradg(x)
y=zeros(4,1);
y(1)=2*(x(1)+10*x(2))+40*(x(1)-x(4))^3;
y(2)=20*(x(1)+10*x(2))+4*(x(2)-2*x(3))^3;
y(3)=10*(x(3)-x(4))-8*(x(2)-2*x(3))^3;
y(4)=-10*(x(3)-x(4))-40*(x(1)-x(4))^3;
end
```

下面我们根据 BFGS 拟牛顿法的原理编写代码。

Algorithm 1 BFGS Method

Require: x_0, $\epsilon > 0$, H_0;

$k \leftarrow 0$;

while $\|\nabla f_k\| > \epsilon$ **do**

  $p_k = -H_k \nabla f_k$;

  Compute α_k from a line search procedure tosatisfy the Wolfe conditions;

  $x_{k+1} = x_k + \alpha_k p_k$;

  $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;

  Compute H_{k+1} by means of BFGS;

  $k \leftarrow k + 1$;

end while

```
function [x,k] = BFGS_Method(f,Gradf,x0,epsilon)
k=0; x=x0;
n=length(x0); H=eye(n)/sqrt(Gradf(x)'*Gradf(x));
while sqrt(Gradf(x)'*Gradf(x))>epsilon
        p=-H*Gradf(x);
        alpha=Wolfe(f,Gradf,x,p);
        y=Gradf(x+alpha*p)-Gradf(x);
        x=x+alpha*p;
        s=alpha*p;
        H=(eye(n)-s*y'/(s'*y))*H*(eye(n)-y*s'/(s'*y))+s*s'/(s'*y);
        k=k+1;
end
end
```

其中我们用到了 Wolfe 条件下的不精确一位线搜索，调用了 $Wolfe(f, gradf, x, p, max)$ 函数。$Wolfe$ 函数的编写过程中也调用了 $zoom$ 函数，这两个函数的算法原理和代码呈现如下。

Algorithm 2 Line Search Algorithm for Wolfe Conditions

Require: α_{low}, α_{high};

Set $\alpha_0 \leftarrow 0$, choose $\alpha_{max} > 0$ and $\alpha_1 \in (0, \alpha_{max})$, $i \leftarrow 1$;
repeat
    Evaluate $\phi(\alpha_i)$;
    if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1]$ **then**
        Set $\alpha_\star \leftarrow zoom(\alpha_{i-1}, \alpha_i)$ and stop;
    end if
    Evaluate $\phi'(\alpha_i)$;
    if $|\phi'(\alpha_i)| \leq -c_2\phi'(0)$ **then**
        Set $\alpha_\star \leftarrow \alpha_i$ and stop;
    end if
    if $\phi'(\alpha_i) \geq 0$ or $\phi'(\alpha_i) < c_2\phi'(0)$ **then**
        Set $\alpha_\star \leftarrow zoom(\alpha_{i-1}, \alpha_i)$ and stop;
    end if
    Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{max})$;
    $i \leftarrow i + 1$;
until find out α

```
function alpha=Wolfe(f,gradf,x,p,max)
a0=0;a1=1;amax=10;c1=0.01;c2=0.4;i=1;
while i<=max
        if f(x+a1*p)>f(x)+c1*a1*gradf(x)'*p |(f(x+a1*p)>f(x+a0*p) & i>1)
                alpha=zoom(a0,a1,f,gradf,x,p);
                break;
        end
        if abs(gradf(x+a1*p)'*p)<=-c2*gradf(x)'*p
                alpha=a1;
                break;
        end
        if gradf(x+a1*p)'*p>=0 | gradf(x+a1*p)'*p<c2*gradf(x)'*p
                alpha=zoom(a0,a1,f,gradf,x,p);
                break;
        end
        a0=a1;
        if a0==amax
                alpha=a0; break;
        end
        a1=2*a1;
        if(a1>amax)
                a1=amax;
        end
        i=i+1;
end
end
```

下面我们编写 zoom 函数的代码。

Algorithm 3 Zoom

Require: α_{low}, α_{high};

repeat

  Interpolate (using quadratic, cubic or bisection) to find a trial step length α_j between α_{low}, α_{high};

  Evaluate $\phi(\alpha_j)$;

  if $\phi(\alpha_j) > \phi(0) + c_1\alpha_i\phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{low})]$ **then**

    Set $\alpha_{high} \leftarrow \alpha_j$;

  else

    Evaluate $\phi'(\alpha_i)$;

    if $|\phi'(\alpha_i)| \leq -c_2\phi'(0)$ **then**

      Set $\alpha_\star \leftarrow \alpha_j$ and stop;

    end if

    if $\phi'(\alpha_j)(\alpha_{high} - \alpha_{low}) \geq 0$ **then**

      Set $\alpha_{high} \leftarrow \alpha_{low}$;

    end if

    $\alpha_{low} \leftarrow \alpha_j$;

  end if

until find out α

```
function alpha=zoom(a_low,a_high,f,gradf,x,p)
alow=a_low;ahigh=a_high;c1=0.01;c2=0.4;
while alow>=0
        alpha=alow+1/2*(ahigh-alow)^2*gradf(x+alow*p)'*p/(f(x+alow*p)-
        f(x+ahigh*p)+(ahigh-alow)*gradf(x+alow*p)'*p);
        if f(x+alpha*p)>f(x)+c1*alpha*gradf(x)'*p | f(x+alpha*p)>=f(x+alow*p)
                ahigh=alpha;
        else
                if abs(gradf(x+alpha*p)'*p)<=-c2*gradf(x)'*p
                        break;
                end
                if (gradf(x+alpha*p)'*p)*(ahigh-alow)>=0
                        ahigh=alow;
                end
                alow=alpha;
        end
end
end
```

(1) extended Rosenbrock function $f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(1-x_i)^2]$
根据 $x_0=[-1.2,1,\cdots,-1.2,1]^T$ 编写程序:

```
x0=zeros(n,1);
for k=1:n/2
        x0(2*k-1)=-1.2;
        x0(2*k)=1;
end
```

下面我们利用 BFGS 拟牛顿法 $BFGS\_Method(f,Gradf,x0,epsilon)$ 函数得到如下结果:

```
n=6时迭代次数为: 52

极小值点为:
9.999999982394704e-01
9.999999990417563e-01
9.999999977048970e-01
9.999999955780986e-01
9.999999929513159e-01
9.999999871974238e-01

极小值为:
1.242386047587804e-15
```

```
n=8时迭代次数为: 66

极小值点为:
9.999999992972118e-01
1.000000000075751e+00
9.999999996216207e-01
9.999999993944044e-01
9.999999990185398e-01
9.999999985364060e-01
9.999999965513700e-01
9.999999934240992e-01

极小值为:
3.335310431010405e-16
```

```
n=10时迭代次数为: 72

极小值点为:
9.999999984675131e-01
9.999999964773540e-01
9.999999955780727e-01
9.999999920372136e-01
```

```
9.999999993778991e-01
9.999999984507908e-01
9.999999974200803e-01
9.999999991240293e-01
1.000000001821922e+00
9.999999952410858e-01

极小值为：
3.452687660445402e-14
```

(2) Powell singular function $f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$
下面我们利用 BFGS 拟牛顿法 $BFGS\_Method(f, Gradf, x0, epsilon)$ 函数得到如下结果：

```
迭代次数为：22

极小值点为：
9.040472224875362e-03
-9.040413100860243e-04
4.289113263481258e-03
4.289408586012169e-03

极小值为：
1.318008495276774e-08
```

从运行结果可以看出，BFGS 方法得到的结果较为准确，且迭代次数也较少。