最优化方法上机报告
无约束优化算法

米科润　19 信计二班

201905755824

June 10, 2021

目录

1 共轭梯度法

1.1 线性共轭梯度法

输入: A 是 n 阶对称正定矩阵,b 是 n 维列向量,x_0 是初始点,epsilon 是容许误差,N 是最大迭代次数

输出: k 是迭代次数,x,val 分别是近似最优点和最优值

```
function [k,x,val]=linecg(A,b,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
if nargin<3, x0=zeros(length(b),1); end
k=0;
gk=A*x0-b;
dk=-gk;
while(k<N)
    temp=A*dk;
    alpha=-gk'*dk/(dk'*temp);
    x=x0+alpha*dk;
    gk=A*x-b;
    betak=gk'*temp/(dk'*temp);
    dk=-gk+betak*dk;
    if(norm(gk)<epsilon), break; end
    x0=x;
    k=k+1;
end
val=0.5*x'*A*x-b'*x;
end
```

结果展示:

求解无约束优化问题 $\min\limits_{x\in R^n} f(x)=\frac{1}{2}x^{\mathrm{T}}Ax-b^{\mathrm{T}}x$

其中 $A=\begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}$, $B=\begin{pmatrix} 3 \\ 2 \\ \vdots \\ 2 \\ 3 \end{pmatrix}$

创建运行文件 test1.m

```matlab
A=zeros(100);
A(1,1:2)=[4,-1];
A(100,99:100)=[-1,4];
xx=[-1,4,-1];
for i = 2:99
    A(i,i-1:i+1)=xx;
end
B=2.*ones(100,1);
B(1)=3;B(100)=3;
x0=zeros(100,1);
epsilon=1e-5;
N=20;
[k,x,val]=linecg(A,B,x0,epsilon,N);
disp('k=');disp(k);disp('val='),disp(val);
```

结果如下

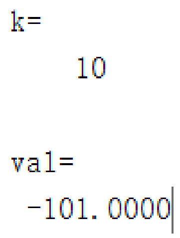


图 1: 线性共轭梯度法结果

1.2 非线性共轭梯度法

基于 Armijo 非精确线搜索的重开始 FR 非线性共轭梯度法

输入: fun,gfun 分别是目标函数及其梯度,x_0 是初始点, epsilon 是容许误差,N 是最大迭代次数

```matlab
function [k,x,val] = frcg(fun,gfun,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
beta=0.6; sigma=0.4;
n=length(x0); k=0;
```

```
while(k<N)
    gk=feval(gfun,x0);
    itern=k-(n+1)*floor(k/(n+1));
    itern=itern+1;
    if(itern==1)
        dk=-gk;
    else
        betak=(gk'*gk)/(g0'*g0);
        dk=-gk+betak*d0;  gd=gk'*dk;
        if(gd≥0.0), dk=-gk; end
    end
    if(norm(gk)<epsilon), break; end
    m=0; mk=0;
    while(m<20)
        if(feval(fun,x0+beta^m*dk)...
                ≤feval(fun,x0)+sigma*beta^m*gk'*dk)
            mk=m; break;
        end
        m=m+1;
    end
    x=x0+beta^mk*dk;
    g0=gk; d0=dk;
    x0=x;   k=k+1;
end
val=feval(fun,x);
end
```

结果展示:

求解无约束优化问题 $\min_{x\in R^2} f(x) = 2x_1^2 + x_2^2 - 2x_1x_2 - 2x_2$

编写目标函数 fun1.m 和梯度 gfun1.m

```
function  f=fun1(x)
f=2*x(1)^2+x(2)^2-2*x(1)*x(2)-2*x(2);
end
```

```
function  gf=gfun1(x)
gf=[4*x(1)-2*x(2);2*x(2)-2*x(1)-2];
end
```

命令窗口输入:

```
>> x0=[0.0;0.0];
>> [k,x,val]=frcg('fun1','gfun1',x0)
```

结果:

```
k =

    12


x =

    1.0000
    2.0000


val =

   -2.0000
```

图 2: 非线性共轭梯度法结果

2 拟牛顿法

2.1 对称秩 1 算法

输入: fun,gfun 分别是目标函数及其梯度,x_0 是初始点, epsilon 是容许误差,N 是最大迭代次数

输出: k 是迭代次数, x,val 分别是近似最优点和最优值

```
function [k,x,val] = sr1(fun,gfun,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
beta=0.55; sigma=0.4;
n=length(x0); Hk=eye(n); k=0;
while(k<N)
```

```
        gk=feval(gfun,x0);
        dk=-Hk*gk;
        if(norm(gk)<epsilon), break; end
        m=0; mk=0;
        while(m<20)
            if(feval(fun,x0+beta^m*dk)<=feval(fun,x0)...
            +sigma*beta^m*gk'*dk)
                mk=m; break;
            end
            m=m+1;
        end
        x=x0+beta^mk*dk;
        sk=x-x0; yk=feval(gfun,x)-gk;
        Hk=Hk+(sk-Hk*yk)*(sk-Hk*yk)'/((sk-Hk*yk)'*yk);
        k=k+1; x0=x;
    end
    val=feval(fun,x0);
    end
```

结果展示:

求解无约束优化问题 $\min\limits_{x \in R^2} f(x) = 2(x_1 - x_2^2)^2 + (x_2 - 2)^2$

编写 fun1.m 和 gfun1.m

```
function f=fun1(x)
f=2*(x(1)-x(2)^2)^2+(x(2)-2)^2;
end
```

```
function gf=gfun1(x)
gf=[4*(x(1)-x(2)^2);-8*x(2)*(x(1)-x(2)^2)+2*(x(2)-2)];
end
```

命令窗口输入:

```
>> x0=[1.0;1.0];
>> [k,x,val]=sr1('fun1','gfun1',x0)
```

```
k =

    12


x =

    4.0000
    2.0000


val =

   3.2595e-16
```

图 3: 对称秩 1 法结果

2.2 BFGS 算法

输入: fun,gfun 分别是目标函数及其梯度,x_0 是初始点,varargin 是输入的可变参数变量，简单调用 BFGS 是可以忽略

输出：k 是迭代次数,x,val 分别是近似最优点和最优值

```
function [k,x,val] = bfgs(fun,gfun,x0,varargin)
N=1000;
epsilon=1.e-5;
beta=0.55; sigma=0.4;
n=length(x0); Bk=eye(n);
k=0;
while(k<N)
    gk=feval(gfun,x0,varargin{:});
    if(norm(gk)<epsilon), break; end
    dk=-Bk\gk;
    m=0; mk=0;
    while(m<20)
        newf=feval(fun,x0+beta^m*dk,varargin{:});
        oldf=feval(fun,x0,varargin{:});
        if(newf<=oldf+sigma*beta^m*gk'*dk)
            mk=m; break;
        end
        m=m+1;
    end
    x=x0+beta^mk*dk;
    sk=x-x0;
```

```
        yk=feval(gfun,x,varargin{:})-gk;
        if(yk'*sk>0)
        Bk=Bk-(Bk*sk*sk'*Bk)/(sk'*Bk*sk)+(yk*yk')/(yk'*sk);
        end
        k=k+1;
        x0=x;
    end
    val=feval(fun,x0,varargin{:});
    end
```

在对称秩 1 算法问题的基础上，命令窗口输入

```
>> x0=[1.0;1.0];
>> [k,x,val]=bfgs('fun1','gfun1',x0)
```

```
k =

    10


x =

    4.0000
    2.0000


val =

   6.0956e-14
```

图 4: BFGS 算法结果

2.3 DFP 算法

输入: fun,gfun 分别是目标函数及其梯度,x_0 是初始点, epsilon 是容许误差,N 是最大迭代次数

输出：k 是迭代次数，x,val 分别是近似最优点和最优值

```
function [k,x,val] = dfp(fun,gfun,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
beta=0.55; sigma=0.4;
n=length(x0); Hk=eye(n); k=0;
```

```
    while(k<N)
        gk=feval(gfun,x0);
        if(norm(gk)<epsilon), break; end
        dk=-Hk*gk;
        m=0; mk=0;
        while(m<20)
            if(feval(fun,x0+beta^m*dk)<=feval(fun,x0)...
            +sigma*beta^m*gk'*dk)
            mk=m; break;
        end
        m=m+1;
        end
        x=x0+beta^mk*dk;
        sk=x-x0; yk=feval(gfun,x)-gk;
        if(sk'*yk>0)
        Hk=Hk-(Hk*yk*yk'*Hk)/(yk'*Hk*yk)+(sk*sk')/(sk'*yk);
        end
        k=k+1; x0=x;
    end
    val=feval(fun,x0);
    end
```

在对称秩 1 算法问题的基础上，命令窗口输入

```
>> x0=[1.0;1.0];epsilon=1e-5;N=20;
>> [k,x,val] = dfp('fun1','gfun1',x0,epsilon,N)
```

```
k =

     9


x =

    4.0000
    2.0000


val =

   2.0346e-11
```

图 5: DFP 算法结果

2.4 Broyden 族算法

输入: fun,gfun 分别是目标函数及其梯度,x_0 是初始点, epsilon 是容许误差,N 是最大迭代次数

输出：k 是迭代次数，x,val 分别是近似最优点和最优值

```matlab
function [k,x,val] = broyden(fun,gfun,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
beta=0.55; sigma=0.4; phi=0.5;
n=length(x0); Hk=eye(n); k=0;
while(k<N)
    gk=feval(gfun,x0);
    if(norm(gk)<epsilon), break; end
    dk=-Hk*gk;
    m=0; mk=0;
    while(m<20)
        if(feval(fun,x0+beta^m*dk)<=feval(fun,x0)...
        +sigma*beta^m*gk'*dk)
            mk=m; break;
        end
        m=m+1;
    end
    x=x0+beta^mk*dk; sk=x-x0;
    yk=feval(gfun,x)-gk; Hy=Hk*yk;
    sy=sk'*yk; yHy=yk'*Hk*yk;
    if(sy<0.2*yHy)
        theta=0.8*yHy/(yHy-sy);
        sk=theta*sk+(1-theta)*Hy;
        sy=0.2*yHy;
    end
    vk=sqrt(yHy)*(sk/sy - Hy/yHy);
    Hk=Hk-(Hy*Hy')/yHy+(sk*sk')/sy+phi*vk*vk';
    x0=x; k=k+1;
end
val=feval(fun,x0);
end
```

在对称秩 1 算法问题的基础上，命令窗口输入

```
>> x0=[1.0;1.0];epsilon=1e-5;N=20;
>> [k,x,val]=broyden('fun1','gfun1',x0,epsilon,N)
```

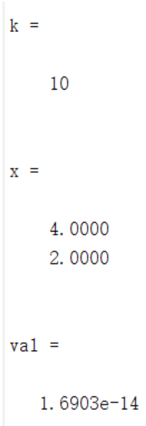


图 6: Broyden 算法结果

3 信赖域算法

3.1 光滑牛顿法求解信赖域子问题

输入：f_k 是 x_k 处的目标函数值,g_k 是 x_k 处的梯度,B_k 是第 k 次近似 Hesse 阵,Δ_k 是当前信赖域半径

输出：d,val 分别是子问题的最优点和最优值,lam 是乘子值,i 是迭代次数

```
function [d,val,lam,i] = trustq(fk,gk,Bk,Δk)
n=length(gk); beta=0.6; sigma=0.2;
mu0=0.05; lam0=0.05; gamma=0.05;
d0=ones(n,1); z0=[mu0,lam0,d0']';
zbar=[mu0,zeros(1,n+1)]';
i=0;
z=z0; mu=mu0; lam=lam0; d=d0;
while (i≤150)
    H=dah(mu,lam,d,gk,Bk,Δk);
    if(norm(H)≤1.e-8)
        break;
    end
    J=JacobiH(mu,lam,d,Bk,Δk);
```

```
        b=psi(mu,lam,d,gk,Bk,Δk,gamma)*zbar-H;
        dz=J\b;
        dmu=dz(1);  dlam=dz(2);  dd=dz(3:n+2);
        m=0;  mi=0;
        while  (m<20)
            t1=beta^m;
            Hnew=dah(mu+t1*dmu,lam+t1*dlam,d+t1*dd,...
                   gk,Bk,Δk);
            if(norm(Hnew)≤(1-sigma*(1-gamma*mu0)*beta^m)...
                *norm(H))
                mi=m;  break;
            end
            m=m+1;
        end
        alpha=beta^mi;
        mu=mu+alpha*dmu;
        lam=lam+alpha*dlam;
        d=d+alpha*dd;
        i=i+1;
    end
    val=fk+gk'*d+0.5*d'*Bk*d;
    end
    %%
    function  p=phi(mu,a,b)
    p=a+b-sqrt((a-b)^2+4*mu^2);
    end
    %%
    function  H=dah(mu,lam,d,gk,Bk,Δk)
    n=length(d);
    H=zeros(n+2,1);
    H(1)=mu;
    H(2)=phi(mu,lam,Δk^2-norm(d)^2);
    H(3:n+2)=(Bk+lam*eye(n))*d+gk;
    end
    %%
    function  J=JacobiH(mu,lam,d,Bk,Δk)
    n=length(d);
    J=zeros(n+2,n+2);
    t2=sqrt((lam+norm(d)^2-Δk^2)^2+4*mu^2);
    pmu=-4*mu/t2;
```

```
thetak=(lam+norm(d)^2-Δk^2)/t2;
J=[1, 0, zeros(1,n);
pmu, 1-thetak, -2*(1+thetak)*d';
zeros(n,1), d, Bk+lam*eye(n)];
end
%%
function si=psi(mu,lam,d,gk,Bk,Δk,gamma)
H=dah(mu,lam,d,gk,Bk,Δk);
si=gamma*norm(H)*min(1,norm(H));
end
```

结果可视化:

求解信赖域子问题最优解 $\underset{||d||_2\leq\Delta_k}{min}\ f(x)=-5+g_k^{\mathrm{T}}d+\frac{1}{2}d^{\mathrm{T}}B_kd$

式中: $g_k=\begin{pmatrix}400\\-200\end{pmatrix}$, $B_k=\begin{pmatrix}1202 & -400\\-400 & 200\end{pmatrix}$,$\Delta_k=5$

命令窗口输入:

```
>> fk=-5;gk=[400 -200]';
>> Bk=[1202 -400; -400 200];
>> Δk=5;
>> [d,val,lam,i] = trustq(fk,gk,Bk,Δk)
```

```
d =

   -0.0000
    1.0000


val =

  -105


lam =

   5.4329e-16


i =

     5
```

图 7: 光滑牛顿算法求解信赖域子问题结果

3.2 牛顿型信赖域方法

x_0 是初始点,epsilon 是容许误差

输出：k 是迭代次数，x,val 分别是近似极小点和近似极小值

```
function [k,x,val] = trustm(x0,epsilon)
n=length(x0); eta1=0.1; eta2=0.75;
tau1=0.5; tau2=2.0;
Δ=1; dtabar=2.0;
x=x0; Bk=Hess(x);k=0;
while(k<50)
    fk=fun(x);
    gk=gfun(x);
    if(norm(gk)<epsilon)
        break;
    end
    [d,val,lam,i]=trustq(fk,gk,Bk,Δ);
    Δq=fk-val;
    Δf=fun(x)-fun(x+d);
    rk=Δf/Δq;
    if(rk≤eta1)
        Δ=tau1*Δ;
    else if (rk≥eta2 & norm(d)==Δ)
            Δ=min(tau2*Δ,dtabar);
        else
            Δ=Δ;
        end
    end
    if(rk>eta1)
        x=x+d;
        Bk=Hess(x);
    end
    k=k+1;
end
val=fun(x);
end
```

求解无约束优化问题 $\underset{x\in R^2}{min} f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$

编写 fun.m,gfun.m,Hess.m

```
%%
function f=fun1(x)
f=100*(x(1)^2-x(2))^2+(x(1)-1)^2;
end
%%
function gf=gfun1(x)
gf=[400*x(1)*(x(1)^2-x(2))+2*(x(1)-1); -200*(x(1)^2-x(2))];
end
%%
function He=Hess1(x)
He=[1200*x(1)^2-400*x(2)+2, -400*x(1);
-400*x(1), 200 ];
end
```

结果可视化：命令窗口输入

```
>> x0=[0.0;0.0];epsilon=1e-6;
>> [k,x,val] = trustm(x0,epsilon)
```

```
k =

    14


x =

    1.0000
    1.0000


val =

   3.4655e-17
```

图 8: 牛顿型信赖域算法结果

4 最小二乘法

4.1 线性最小二乘问题

4.1.1 法方程 Cholesky 分解法

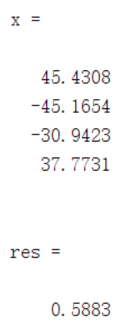
```
function [x,res]=nels(A,b)
B=A'*A; f=A'*b;
L=chol(B,'lower');
y=L\f; x=L'\y;
res=norm(b-A*x);
end
```

求解超定方程组 $Ax=b$, 其中

$$A=\begin{pmatrix}2&3&4&5\\4&3&2&1\\4&5&6&7\\9&5&7&2\\4&2&5&3\end{pmatrix},x=\begin{pmatrix}x1\\x2\\x3\\x4\end{pmatrix},b=\begin{pmatrix}20\\22\\35\\42\\50\end{pmatrix}$$

命令窗口输入:

```
>> A=[2 3 4 5; 4 3 2 1; 4 5 6 7; 9 5 7 2; 4 2 5 3];
>> b=[20 22 35 42 50]';
>> [x,res]=nels(A,b)
```



```
x =

   45.4308
  -45.1654
  -30.9423
   37.7731


res =

    0.5883
```

图 9: Cholesky 分解结果

4.1.2 QR 分解法

```
function [x,res]=qrls(A,b)
[Q,R]=qr(A); f=Q'*b;
x=R\f; res=norm(b-A*x);
end
```

求解 Cholesky 中的超定方程组

命令窗口输入：

```
>> A=[2 3 4 5; 4 3 2 1; 4 5 6 7; 9 5 7 2; 4 2 5 3];
>> b=[20 22 35 42 50]';
>> [x,res]=qrls(A,b)
```

```
x =

   45.4308
  -45.1654
  -30.9423
   37.7731


res =

    0.5883
```

图 10: QR 分解结果

4.1.3 SVD 求解亏秩最小二乘

```
A=[1 2 3 4; 1 4 5 6; 1 5 6 7; 1 8 9 10; 1 11 12 13];
b=[11 13 15 18 20]';
[m,n]=size(A); x=zeros(n,1);
[U,S,V]=svd(A);
r=rank(S);
for i=1:r
    x=x+(U(:,i)'*b/S(i,i))*V(:,i);
end
x
res=norm(b-A*x)
```

求解超定方程组 $Ax=b$, 其中

$$A=\begin{pmatrix}1&2&3&4\\1&4&5&6\\1&5&6&7\\1&8&9&0\\1&11&12&13\end{pmatrix},x=\begin{pmatrix}x1\\x2\\x3\\x4\end{pmatrix},b=\begin{pmatrix}11\\13\\15\\18\\20\end{pmatrix}$$

运行可得结果

```
x =

    2.7533
   -2.4133
    0.3400
    3.0933


res =

    1.0863
```

图 11: SVD 分解结果

4.2 非线性最小二乘问题

4.2.1 L-M 方法

求解非线性方程组 $F(x)=0$，可适用于未知数个数与方程的个数不相等情形

输入：F_k, JF_k 分别是求 F(xk) 及 F'(xk) 的函数,x_0 是初始点,epsilon 是容许误差，N 是最大迭代次数

输出：k 是迭代次数，x，val 分别是近似解和 $0.5*||F(xk)||^2$ 的值

```
function [k,x,val] = lmm(Fk,JFk,x0,epsilon,N)
if nargin<5, N=1000; end
if nargin<4, epsilon=1.e-5; end
beta=0.55; sigma=0.4;
n=length(x0);
muk=norm(feval(Fk,x0));
k=0;
while(k<N)
    fk=feval(Fk,x0);
```

```
        jfk=feval(JFk,x0);
        gk=jfk'*fk;  dk=-(jfk'*jfk+muk*eye(n))\gk;
        if(norm(gk)<epsilon), break; end
        m=0; mk=0;
        while(m<20)
            fnew=0.5*norm(feval(Fk,x0+beta^m*dk))^2;
            fold=0.5*norm(feval(Fk,x0))^2;
            if(fnew<fold+sigma*beta^m*gk'*dk)
                mk=m; break;
            end
            m=m+1;
        end
        x0=x0+beta^mk*dk;
        muk=norm(feval(Fk,x0));
        k=k+1;
    end
    x=x0;
    val=0.5*muk^2;
    end
```

结果可视化：编写 Fk.m 和 JFK.m

```
%%
function F=Fk(x)
n=length(x); F=zeros(n,1);
for i=1:n-1
    F(i)=x(i)*x(i+1)-1;
end
F(n)=x(1)*x(n)-1;
%%
function JF=JFk(x)
n=length(x); JF=zeros(n,n);
for i=1:n-1
    for j=1:n
        if j==i
            JF(i,j)=x(j+1);
        else if j==i+1
            JF(i,j)=x(j-1);
            else
```

```
                    JF(i,j)=0;
                end
            end
        end
    end
    JF(n,1)=x(n); JF(n,n)=x(1);
```

命令窗口输入：

```
>> x0=3*ones(100,1);epsilon=1e-6; N=20;
>> [k,x,val] = lmm('Fk','JFk',x0,epsilon,N)
```

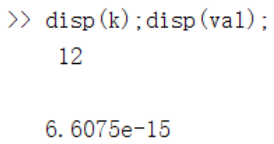


图 12: LM 方法结果