BFGS算法

山东理工大学 数学学院

Contents

拟牛顿法BFGS

https://blog.csdn.net/u012507032/article/details/77923960

BGFS是一种准牛顿算法，所谓的"准"是指牛顿算法会使用Hessian矩阵来进行优化，但是直接计算Hessian矩阵比较麻烦，所以很多算法会使用近似的Hessian，这些算法就称作准牛顿算法(Quasi Newton

牛顿算法(Newton Algorithm)
牛顿算法考虑了函数的二阶单数，是一种二阶优化方法，并且是所有其他二阶优化方法的鼻祖．作为对比，梯度下降(Gradient Descent)只考虑了函数的一阶导数，是一阶优化方法．BFGS算法被认为是超线性收敛速度。

牛顿法的特点是：收敛速度快，迭代次数少，但是当Hessian 矩阵很稠密时，每次迭代的计算量很大。随着数据规模的增大，那么Hessian矩阵会越大， 需要的存储空间会增多，计算量也会增大，有时候大到不可计算，所以针对海量数据的计算，牛顿法不再适用。 拟牛顿法是在牛顿法的基础上引入了Hessian矩阵的近似矩阵，避免每次迭代都计算Hessian矩阵的逆，它 的收敛速度介于梯度下降法和牛顿法之间。拟牛顿法跟牛顿法一样，也是不能处理太大规模的数据，因为计算 量和存储空间会开销很多。拟牛顿法虽然每次迭代不像牛顿法那样保证是最优化的方向，但是近似矩阵始终是 正定的，因此算法始终是朝着最优化的方向在搜索。

拟牛顿法的基本思想是在牛顿法中用Hessian矩阵的某个近似矩阵来代替它。在高数中，学过泰勒公式，如下

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+R_n(x)$$

关于泰勒展开的更多内容，可以参考这里：泰勒公式介绍，泰勒公式应用。泰勒公式是用一个近似的多项式
来代替原来复杂的函数表达式。对于拟牛顿法来说，构造二次模型。如下

$$f(X)=f(X_{i+1})+(X-X_{i+1})^T\bigtriangledown f(X_{i+1})+\frac{1}{2}(X-X_{i+1})^TH_{i+1}(X-X_{i+1})+o(X-X_{i+1})$$

忽略高阶无穷小部分，进行求导得到$\bigtriangledown f(X)\doteq\bigtriangledown f(X_{i+1})+H_{i+1}(X-X_{i+1})$,令$X=X_i$,得到$H_{i+1}^{-1}(\bigtriangledown f(X_{i+1}-\bigtriangledown f(X_i))\doteq(X_{i+1}-X_i)$,由于H矩阵维度超大，求逆矩阵非常困难，计算复杂度非常高。BFGS算法 算法本质：一种通过迭代逼近的拟牛顿算法。

BFGS算法。它是Broyden，Fletcher, Goldfarb，Shanno四位牛人发明出来到现在的40多年时间里，它仍然被认为是最好的拟牛顿算法

Logistic预测模型

某农产品亩产量(kg/亩)的历史数据已知，试预测1985年，1990年，1995年，2000年的亩产量， Logistic曲线模型$y=\frac{k}{1+me^{-at}}$,其中，a,m是待定系数，k是饱和值，可事先给定。 画出1985-1982年依次为1,2,...,25，作出历史数据的散点图。发现呈S型。

即求参数a,m满足：

$$\min_{a,m}\sum_{i=1}^{25}(\hat{y}-\frac{k}{1+me^{-at}})^2$$

非线性最小二乘问题。

先建立目标函数文件examLogistic.m

```
% function y=examLogistic(x)
% yhat=1e-3*[185 350 263 335 560 570 263 214 276 435 476 452 442 404 458 427 409 430 421 444 539 632 749 690 679]';
% for i=1:25
%     z(i)=(yhat(i)-1/(1+x(2)*exp(-i*x(1))))^2;
% end
% y=sum(z);
```

```
% function [ xstar,fxstar,iter ] = BFGS( f_name,x0,flag,varepsilon )
% %UNTITLED3 此处显示有关此函数的摘要
% %   此处显示详细说明
% tic
% k=0;
% n=length(x0);B=eye(n,n);xk=x0;a0=1;h0=1;t=2;e=1e-3;
% if flag==0
%     g=MyGradient(f_name,xk); %计算梯度
% elseif flag==1
%     [f,g]=feval(f_name,xk);%根据梯度公式直接计算目标函数的梯度值
% end
% while norm(g)>varepsilon
%     dk=Gauss2(B,-g,1);
%     lambda=p618(f_name,xk,dk,a0,h0,t,e);
%     xkplus1=xk+lambda*dk;
%     if flag==0
%         gkplus1=MyGradient(f_name,xkplus1);
%     elseif flag==1
%         [f,gkplus1]=feval(f_name,xkplus1);
%     end
%     yk=gkplus1-g;sk=xkplus1-xk;
%     B=B+(yk*yk')/(yk'*sk)-(B*sk*sk'*B)/(sk'*B*sk);
%     xk=xkplus1;g=gkplus1;
%     k=k+1;
```

```
% end
% xstar =xk;fxstar=feval(f_name,xk);iter=k;
% toc;
% end
```

```
% unction x = Gauss2(A,b,flag  )
% %UNTITLED4 此处显示有关此函数的摘要
% %   此处显示详细说明
% if nargin<3
%     flag=0;
% end
% n=length(b);
% A=[A,b];
% for k=1:n-1 %选主元
%     [Ap,p]=max(abs(A(k:n,k)));
%     p=p+k-1;
%     if p>k
%         t=A(k,:);A(k,:)=A(p,:);A(p,:)=t;
%     end
%     A((k+1):n,(k+1):(n+1))=A((k+1):n,(k+1):(n+1))-A((k+1):n,k)/A(k,k)*A(k,(k+1):(n+1));
%     A((k+1):n,k)=zeros(n-k,1);
%     if flag==0
%         A
%     end
% end
%
% x=zeros(n,1) %回代
% x(n)=A(n,n+1)/A(n,n);
% for k=n-1:-1:1
%     x(k,:)=(A(k,n+1)-A(k,(k+1):n)*x((k+1):n))/A(k,k);
% end
%
%
% end
```

```
% function [xstar,fxstar,iter] = p618( f_name,xk,dk,a0,h0,t,e )
% % 求一元函数极值
% %   此处显示详细说明
% tic;
% [a,b]=Trial(f_name,xk,dk,a0,h0,t,e);
% if a<0
%     a=0;
% else
%  a=a;
% end
% k=1;
% a(k)=a;b(k)=b;
% lambda(k)=a(k)+0.382*(b(k)-a(k));mu(k)=a(k)+0.618*(b(k)-a(k));
% m(k)=feval(f_name,xk+lambda(k)*dk);n(k)=feval(f_name,xk+mu(k)*dk);
% flag=0;
% while flag==0
%     if m(k)>n(k)
%         if (b(k)-lambda(k))<e
%             xstar=mu(k);fxstar=feval(f_name,xk+xstar*dk);
%             iter=k-1;flag=1;
%         else
%             a(k+1)=lambda(k);b(k+1)=b(k);
%             lambda(k+1)=mu(k);
%             m(k+1)=n(k);
%             mu(k+1)=a(k+1)+0.618*(b(k+1)-a(k+1));
%             n(k+1)=feval(f_name,xk+mu(k+1)*dk);
%             k=k+1;flag=0;
%         end
%     else
%         if (mu(k)-a(k))<e
%             xstar=lambda(k);fxstar=feval(f_name,xk+xstar*dk);iter=k-1;
%             flag=1;
%         else
%             a(k+1)=a(k);b(k+1)=mu(k);mu(k+1)=lambda(k);n(k+1)=m(k);
%             lambda(k+1)=a(k+1)+0.382*(b(k+1)-a(k+1));
%             m(k+1)=feval(f_name,xk+lambda(k+1)*dk);
%             k=k+1;flag=0;
%         end
%     end
% end
% iter=k-1;
% toc;
% end
```

```
% function g= MyGradient( f_name,x )
% % 常用的有插复值型求导公式用于求某点上导数，
% % 样条求导公式用于求利用插值的结果拟合出的结果。一般有3点公式或者制5点公式。
% %
% n=length(x);h0=1e-3;E=eye(n);g=zeros(n,1);y0=feval(f_name,x);
% for j=1:n
%     h=h0.*E(:,j);
%     for i=1:4
%         y(i)=feval(f_name,x+i*h); z(i)=feval(f_name,x-i*h);
%     end
%    d(1)=(-25*y0+48*y(1)-36*y(2)+16*y(3)-3*y(4))/(12*h0);
%    d(2)=(-3*z(1)-10*y0+18*y(1)-6*y(2)+y(3))/(12*h0);
%    d(3)=(z(2)-8*z(1)+8*y(1)-y(2))/(12*h0);
%    d(4)=(3*y(1)+10*y0-18*z(1)+6*z(2)-z(3))/(12*h0);
```

```
%     d(5)=(25*y0-48*z(1)+36*z(2)-16*z(3)+3*z(4))/(12*h0);
%     g(j)=mean(d);
% end
% end
```

```
% function [a,b,c ] = Trial( f_name,xk,dk,a0,h0,t,e )
% %UNTITLED2 此处显示有关此函数的摘要
% %    此处显示详细说明
% MaxIterations=500;
% Iterations=0;
% f0=feval(f_name,xk+a0*dk);%初始点函数值
% a1=a0+h0;f1=feval(f_name,xk+a1*dk);
% while abs(f1-f0)<e
%     a1=a1+t*h0;f1=feval(f_name,xk+a1*dk);
% end
% if f1>f0
%     a2=a0-h0;
%     f2=feval(f_name,xk+a2*dk);
%     while f2<=f0
%         a2=a2-t*h0;
%         f2=feval(f_name,xk+a2*dk);
%         Iterations=Iterations+1;
%     if Iterations>MaxIterations
%         break
%     end
%     end
%
%
% a=a2;b=a1;c=a0;
% elseif f1<f0
%     a2=a1+t*h0;
%     f2=feval(f_name,xk+a2*dk);
%     while f2<=f1
%         a2=a2+t*h0;
%         f2=feval(f_name,xk+a2*dk);
%     Iterations=Iterations+1;
%     if Iterations>MaxIterations
%         break
%     end
% end
% a=a0;b=a2;c=a1;
% end
% fa=feval(f_name,xk+a*dk);fb=feval(f_name,xk+b*dk);fc=feval(f_name,xk+c*dk);
% if (fa<fc)&(fc<fb)
%     fprintf('到达最大迭代次数，这是单调增函数，没有包含极小点的的区间')
% elseif(fa>fc) &(fc>fb)
%      fprintf('到达最大迭代次数，这是单调减函数，没有包含极小点的的区间')
% end
```

选取初值(1,1)^T,并调用BFGS.m计算：

```
[xstar,fxstar,iter]=BFGS(@examLogistic,[1,1]',0,1e-3)

% xstar =
%
%     0.0596
%     2.7571
% fxstar =
%
%     0.2548
```

```
x =

     0
     0

时间已过 0.001072 秒。

x =

     0
     0

时间已过 0.000558 秒。

x =

     0
     0

时间已过 0.000499 秒。

x =

     0
     0

时间已过 0.000600 秒。

x =

     0
```

```
     0

时间已过 0.000616 秒。

x =

     0
     0

时间已过 0.000586 秒。

x =

     0
     0

时间已过 0.000782 秒。
时间已过 0.001109 秒。

xstar =

    0.0596
    2.7571


fxstar =

    0.2548


iter =

     7
```

练习

在某个工业产品的制造中，其单位产品必须含有0.5%的有效氯气，已知产品中的氯气 随时间增加而减少，在产品到达用户之前的最初8周内，氯气含量衰减到0.49%。表中 是产品生产一段时间后观察到的若干产品中有效氯气的含量。 假定8周后非线性模型$y = a + (0.49 - a)e^{-b(x-8)}$可以解释产品生产8周后x>8与有效氯气含量y的关系，试用最小二乘法求 参数a,b. 初始迭代点选[1,1]

产品生产后的时间/周	8	8	10	10	10	10	12	12	12	12	14
有效卡其的百分比/%	0.49	0.49	0.48	0.47	0.48	0.47	0.46	0.46	0.45	0.45	0.43
产品生产后的时间/周	14	14	16	16	16	18	18	20	20	20	22
有效卡其的百分比/%	0.43	0.43	0.44	0.43	0.43	0.46	0.45	0.42	0.42	0.43	0.41
产品生产后的时间/周	22	22	24	24	24	26	26	26	28	28	30
有效卡其的百分比/%	0.41	0.40	0.42	0.40	0.40	0.41	0.41	0.41	0.41	0.40	0.40
产品生产后的时间/周	30	30	32	32	34	36	36	38	38	40	42
有效卡其的百分比/%	0.40	0.38	0.41	0.40	0.40	0.38	0.38	0.40	0.40	0.39	0.39

参考解：[0.3868,0.0953]

最速下降法

```
% function   [ xstar,fxstar,iter ] =SteepDescent( f_name,x0,flag,varepsilon )
%   最速下降法，求解无约束优化问题，
%     优点是计算量小，存储量小，对初始点无要求，缺点是容易限入局部最优，收敛较慢
% tic;
% k=0;xk=x0;e=1e-3;a0=1;h0=1;t=2;
% if flag==0
%     g=MyGradient(f_name,xk);
% elseif flag==1
%     [f,g]=feval(f_name,xk);
% end
% while norm(g)>varepsilon
%     dk=-g;
%     lambda=p618(f_name,xk,dk,a0,h0,t,e);
%     xk=xk+lambda*dk;
%     if flag==0
%         g= MyGradient(f_name,xk);
%    elseif flag==1
%     [f,g]=feval(f_name,xk);
%     end
%     k=k+1;
% end
% xstar =xk;fxstar=feval(f_name,xk);iter=k;
% toc;
% end

% MATLAB自带的函数fminunc.m可以实现最速下降法求解无约束极小化问题。最速下降法
% 只是该函数使用的算法之一。
```

案例

(Sylverter问题) 在平面上有6个点，试找出覆盖这六个点的最小圆盘。坐标为 (-1,5),(4,1),(7,-4),(10,9),(3,4),(8,2).

Sylvester问题： n个点，找出覆盖这 n个点的最小圆盘。设坐标为 $p_i=(a_i,b_i),i=1,2,\cdots,n$ 平面上任意点 $p(x_1,x_2)$ 到 p_i距离为：

$$d_i=\sqrt{(x_1-a_i)^2+(x_2-b_i)^2},i=1,2,\cdots,n$$

于是，让 p点到各点距离中最大者最小即为所求最小圆盘，即p点坐标满足：

$$\min_{x_1,x_2}\max_{1\le i\le n}d_i=\min_{x_1,x_2}\max_{1\le i\le n}\sqrt{(x_1-a_i)^2+(x_2-b_i)^2}$$

化为无约束的极小化问题求解，实际上，任意两个距离 d_i,d_j中的最大值可以表示为：

$$\max\{d_i,d_j\}=\frac{(d_i+d_j)+|d_i-d_j|}{2}$$

编写目标函数：

```
%function y = Sylvester(x)
%UNTITLED4 此处显示有关此函数的摘要
%   此处显示详细说明
% A=[-1 -5;4 1;7 -4;10 9;3 4;8 2];[m,n]=size(A);
% for i=1:m
%     d(i)=sqrt((x(1)-A(i,1))^2+(x(2)-A(i,2))^2);
% end
% y=d(1);
% for i=2:m
%     y=((y+d(i))+abs(y-d(i)))/2;
% end
```

```
   [ xstar,fxstar,iter ] =SteepDescent(@Sylvester,[1,1]',0,1e-3 )

% 参考解答： xstar = 4.4974    2.0020
```

```
时间已过 0.000435 秒。
时间已过 0.000419 秒。
时间已过 0.000330 秒。
时间已过 0.000382 秒。
时间已过 0.000337 秒。
时间已过 0.000305 秒。
时间已过 0.000354 秒。
时间已过 0.000369 秒。
时间已过 0.000336 秒。
时间已过 0.000324 秒。
时间已过 0.000306 秒。
时间已过 0.000356 秒。
时间已过 0.000373 秒。
时间已过 0.000308 秒。
时间已过 0.000304 秒。
时间已过 0.000308 秒。
时间已过 0.000337 秒。
时间已过 0.000309 秒。
时间已过 0.000426 秒。
时间已过 0.000308 秒。
时间已过 0.000308 秒。
时间已过 0.000304 秒。
时间已过 0.000310 秒。
时间已过 0.000349 秒。
时间已过 0.000405 秒。
时间已过 0.000403 秒。
时间已过 0.000371 秒。
时间已过 0.000665 秒。
时间已过 0.000313 秒。
时间已过 0.000311 秒。
时间已过 0.000308 秒。
时间已过 0.000328 秒。
时间已过 0.000534 秒。
时间已过 0.000564 秒。
时间已过 0.001082 秒。
时间已过 0.000431 秒。
时间已过 0.000355 秒。
时间已过 0.000616 秒。
时间已过 0.000561 秒。
时间已过 0.000885 秒。
时间已过 0.001396 秒。
时间已过 0.000386 秒。
时间已过 0.000399 秒。
时间已过 0.000696 秒。

xstar =

    4.4974
    2.0020


fxstar =

    8.9022
```

```
iter =

    43
```

参考文献

```
% 李工农，《运筹学基础及其MATLAB应用》，清华大学出版社，2016年10月。
```

Published with MATLAB® R2014a