优化方法

2016-2017学年工科硕士课程

第3章　无约束优化方法

3.1 下降递推算法

第一章讲过,无约束非线性规划问题的提法是:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathbf{R}^n \end{aligned}$$

第一章我们还指出:若$f(x) \in C^1$,则x^*为f(x) 的一个极小点的必要条件是$\nabla f(x^*) = 0$或

$$\frac{\partial f(x^*)}{\partial x_i} = 0, \ i = 1, \cdots, n. \tag{1-1}$$

这是n个未知量,n个方程的方程组,并且一般是非线性的. 在一般情况下,都不能用解析方法求得准确解,只能用数值方法逐步求其近似解,这就是本章所要介绍的求解无约束最优化问题的各种迭代方法,简称为无约束优化方法.

迭代法的基本思想

首先给出$f(x)$的极小点x^*的一个初始估计x^0(称为初始点),然后计算一系列的点$x^1, x^2, \cdots, x^k, \cdots$,希望点列$\{x^k\}$的极限就是$f(x)$的一个极小点$x^*$.
怎样产生这个点列呢?也就是说在有了点x^k之后,如何求得点x^{k+1}呢?

迭代法的基本思想

首先给出$f(x)$的极小点x^*的一个初始估计x^0(称为初始点),然后计算一系列的点$x^1, x^2, \cdots, x^k, \cdots$,希望点列$\{x^k\}$的极限就是$f(x)$的一个极小点$x^*$.
怎样产生这个点列呢?也就是说在有了点x^k之后,如何求得点x^{k+1}呢?
我们这样来考虑,因为$x^{k+1} - x^k$是一个向量,而一个向量是由它的方向和长度确定的,即$x^{k+1} - x^k = \lambda_k s^k$ 或

$$x^{k+1} = x^k + \lambda_k s^k \tag{1-2}$$

其中s^k是一个向量,λ_k是一个正实数(称为步长).当s^k与λ_k确定以后,由x^k就可以唯一的确定x^{k+1}.

迭代法的基本思想

这样就可以确定逼近极小点的一个序列$\{x^k\}$,通常称它为一个极小化序列.各种不同算法的差别就在于选取方向s^k和步长λ_k的方法不同,特别是选取s^k的方法不同.那么, 应该按照什么原则来选取方向和步长呢?

迭代法的基本思想

这样就可以确定逼近极小点的一个序列$\{x^k\}$,通常称它为一个极小化序列.各种不同算法的差别就在于选取方向s^k和步长λ_k的方法不同,特别是选取s^k的方法不同.那么, 应该按照什么原则来选取方向和步长呢?

(1) 要求极小化序列对应的函数值是逐次减少的,至少是不增的,即

$$f(x^0) \geq f(x^1) \geq \cdots \geq f(x^k) \geq \cdots$$

具有这种性质的算法,称为下降递推算法或下降算法.

迭代法的基本思想

这样就可以确定逼近极小点的一个序列$\{x^k\}$,通常称它为一个极小化序列.各种不同算法的差别就在于选取方向s^k和步长λ_k的方法不同,特别是选取s^k的方法不同.那么, 应该按照什么原则来选取方向和步长呢?

(1) 要求极小化序列对应的函数值是逐次减少的,至少是不增的,即

$$f(x^0) \geq f(x^1) \geq \cdots \geq f(x^k) \geq \cdots$$

具有这种性质的算法,称为下降递推算法或下降算法.

(2) 要求极小化序列具有这样的性质,或者序列$\{x^k\}$中的某一点本身是极小点,或者$\{x^k\}$有一个极限x^*, 它是$f(x)$的一个极小点.具有上述性质的算法称为收敛的.

迭代法的基本步骤

(1) 选择初始点x^0;

迭代法的基本步骤

(1) 选择初始点x^0;

(2) 如果x^k已求得,但不是极小点,设法选取一个方向s^k,使目标函数$f(x)$沿s^k是下降的,至少是不增的,s^k称为搜索方向;

迭代法的基本步骤

(1) 选择初始点x^0;

(2) 如果x^k已求得,但不是极小点,设法选取一个方向s^k,使目标函数$f(x)$沿s^k是下降的,至少是不增的,s^k称为搜索方向;

(3) 在方向s^k确定以后,在射线$x^k+\lambda s^k(\lambda>0)$上选取适当的步长$\lambda_k$,使$f(x^k+\lambda_k s^k)\le f(x^k)$, 如此确定出下一个点$x^{k+1}=x^k+\lambda_k s^k$(在多数算法中,$\lambda_k$的选取是使$f(x)$的值下降最多,即沿$x^k+\lambda s^k$求$f(x)$的极小值,这是关于单变量$\lambda$的函数求极小的问题).这种确定步长$\lambda_k$的方法称为一维搜索或线搜索.

迭代法的基本步骤

(1) 选择初始点x^0;

(2) 如果x^k已求得,但不是极小点,设法选取一个方向s^k,使目标函数$f(x)$沿s^k是下降的,至少是不增的,s^k称为搜索方向;

(3) 在方向s^k确定以后,在射线$x^k+\lambda s^k(\lambda>0)$上选取适当的步长$\lambda_k$,使$f(x^k+\lambda_k s^k)\leq f(x^k)$, 如此确定出下一个点$x^{k+1}=x^k+\lambda_k s^k$(在多数算法中,$\lambda_k$的选取是使$f(x)$的值下降最多,即沿$x^k+\lambda s^k$求$f(x)$的极小值,这是关于单变量$\lambda$的函数求极小的问题).这种确定步长$\lambda_k$的方法称为一维搜索或线搜索.

(4) 检验所得的新点x^{k+1}是否为极小点,或满足精度要求的近似极小点,检验的方法因算法而不同,例如,有些算法看梯度的模是否小于某个预先指定的小正数ε,即若$\|\nabla f(x^{k+1})\|\leq\varepsilon$,就认为$x^{k+1}$是$f(x)$的近似极小点,迭代结束,否则继续进行迭代.

算法的收敛速度

给定一个迭代算法,我们不仅要求它是收敛的,而且希望由这个算法产生的点列$\{x^k\}$, 以较快的速度收敛于最优解.收敛的快慢通常用收敛的阶来度量.

定义1.1

设x^*为问题$f(x), x \in \mathbf{R}^n$的最优解,由算法A产生的序列$\{x^k\}$收敛于x^*,即

$$\lim_{k \to +\infty} x^k = x^* \tag{1-3}$$

若存在一个与k无关的常数$\beta \in (0,1)$,某个正整数k_0，使当$k > k_0$时

$$\|x^{k+1} - x^*\| \leq \beta \|x^k - x^*\| \tag{1-4}$$

成立,则称序列$\{x^k\}$为线性收敛,有时也称算法A为线性收敛.

不失一般性,可以假定(1-4)式对于$k \geq 0$都成立,于是有

$$\|x^k - x^*\| \leq \beta\|x^{k-1} - x^*\| \leq \beta^2\|x^{k-2} - x^*\| \leq \cdots \leq \beta^k\|x^0 - x^*\|$$

上式说明:点x^k到点x^*的距离,当$k \to +\infty$时,大致以公比为β的等比序列(又称几何序列)减小,因此线性收敛速度相当于相应的等比序列的收敛速度.

定义1.2

设由算法A产生的序列$\{x^k\}$收敛于最优解x^*,如果存在一个与k无关的常数$\beta > 0$和$\alpha > 1$, 某个正整数k_0,使当$k > k_0$时,

$$\|x^{k+1} - x^*\| \leq \beta \|x^k - x^*\|^\alpha \tag{1-5}$$

成立,则称序列$\{x^k\}$的收敛阶为α,或称序列$\{x^k\}$是α阶收敛的,有时也说算法A是α阶收敛的.

当$\alpha = 2$时,称序列$\{x^k\}$是二阶收敛的; 当$1 < \alpha < 2$时,称序列$\{x^k\}$是超线性收敛的.

停止准则

显然,当$\|x^k - x^*\| < \varepsilon$时(其中$\varepsilon$是预先给定的小正数),$x^k$就是所求的近似最优解.但$x^*$是未知的,因此$\|x^k - x^*\|$无法计算.当$\|x^k - x^*\|$很小时,$\|x^{k+1} - x^k\|$也必然很小. 所以,可以用

$$\|x^{k+1} - x^k\| < \varepsilon \tag{1-6}$$

作为计算结束的检验条件,ε是根据计算要求给定的小正数,以后我们称为计算精度或终止限.

3.2 一维搜索

在不少算法中,步长λ_k是由求$\varphi(\lambda) = f(x^k + \lambda s^k)$的极小值确定的,即$f(x^k + \lambda_k s^k) = \min_{\lambda \geq 0} f(x^k + \lambda s^k)$, 这种确定步长$\lambda_k$的方法称为精确一维搜索或简称一维搜索.

3.2 一维搜索

在不少算法中,步长λ_k是由求$\varphi(\lambda) = f(x^k + \lambda s^k)$的极小值确定的,即$f(x^k + \lambda_k s^k) = \min_{\lambda \geq 0} f(x^k + \lambda s^k)$, 这种确定步长$\lambda_k$的方法称为精确一维搜索或简称一维搜索. 设要求一元函数$\varphi(x)$的极小点问题为

$$\min \quad \varphi(x) \tag{2-1}$$

若$\varphi(x) \in C^1$,则在极小点x^*处,应有

$$\varphi'(x^*) = 0 \tag{2-2}$$

精确一维搜索方法,可以分为两类:

- 不用导数的方法,如0.618法,抛物线法,“成功—失败”法等.
- 使用导数的方法,如Newton法,平分法,三次插值法等.

3.2.1 牛顿(Newton)法

牛顿法的基本思想是:用$\varphi(x)$在已知点x_0处的二阶Taylor展开式来近似代替$\varphi(x)$, 即取$\varphi(x) \approx g(x)$,其中

$$g(x) = \varphi(x_0) + \varphi'(x_0)(x - x_0) + (1/2)\varphi''(x_0)(x - x_0)^2,$$

用g(x)的极小点$\hat{x}$作为$\varphi(x)$的近似极小点.

3.2.1 牛顿(Newton)法

牛顿法的基本思想是:用$\varphi(x)$在已知点x_0处的二阶Taylor展开式来近似代替$\varphi(x)$, 即取$\varphi(x) \approx g(x)$,其中

$$g(x) = \varphi(x_0) + \varphi'(x_0)(x - x_0) + (1/2)\varphi''(x_0)(x - x_0)^2,$$

用g(x)的极小点$\hat{x}$作为$\varphi(x)$的近似极小点.
由于$g'(x) = \varphi'(x_0) + \varphi''(x_0)(x - x_0)$, 令$g'(x) = 0$, 得

$$x_1 = x_0 - \frac{\varphi'(x_0)}{\varphi''(x_0)} \tag{2-3}$$

类似的,若已知x_k点,则有

$$x_{k+1} = x_k - \frac{\varphi'(x_k)}{\varphi''(x_k)} \quad k = 0, 1, 2, \cdots \tag{2-3'}$$

按公式(2-3')进行迭代计算,可求得一个序列$\{x_k\}$,这种求一元函数极小值的一维搜索方法称为牛顿法. 当$|\varphi'(x_k)| < \varepsilon$时(其中$\varepsilon$为计算精度),则迭代结束,$x_k$为$\varphi(x)$的近似极小点, 即$x^* \approx x_k$.

牛顿法的优点是收敛速度快,可以证明,是二阶收敛的.缺点是:

- 需计算二阶导数;
- 要求初始点x_0选得好,即要求$x_0 \in N(x^*, \delta), \delta > 0$,否则可能不收敛.

例2.1

求$f(x) = x^4 - 4x^3 - 6x^2 - 16x + 4$的极小值.

例2.1

求$f(x)=x^4-4x^3-6x^2-16x+4$的极小值.

- 直接法.令$f'(x)=0$,求得$x_1=4, x_2=\frac{-1\pm\sqrt{3}i}{2}$.因此,在实数域,只有$x=4$使$f'(x)=0$.因为

$$f''(x)=12x^2-24x-12,\ f''(4)=84>0$$

所以$f(x)$在$x=4$处取得极小值.

例2.1

求$f(x)=x^4-4x^3-6x^2-16x+4$的极小值.

- 直接法.令$f'(x)=0$,求得$x_1=4, x_2=\frac{-1\pm\sqrt{3}i}{2}$.因此,在实数域,只有$x=4$使$f'(x)=0$.因为

$$f''(x)=12x^2-24x-12,\ f''(4)=84>0$$

所以$f(x)$在$x=4$处取得极小值.

- 牛顿法.取$x_0=6, \varepsilon=10^{-3}$,则

$$\begin{aligned}
x_1 &= 6-\frac{f'(6)}{f''(6)}=4.75\\
x_2 &= 4.75-\frac{f'(4.75)}{f''(4.75)}=4.163\\
x_3 &= 4.163-\frac{f'(4.163)}{f''(4.163)}=4.010\\
x_4 &= 4.010-\frac{f'(4.010)}{f''(4.010)}=4.00004
\end{aligned}$$

应当指出牛顿法产生的序列即使收敛,其极限也不一定是$f(x)$的极小点,而只能保证它是$f(x)$的驻点,驻点可能是极小点,也可能是极大点,也可能既不是极小点,又不是极大点.

应当指出牛顿法产生的序列即使收敛,其极限也不一定是$f(x)$的极小点,而只能保证它是$f(x)$的驻点,驻点可能是极小点,也可能是极大点,也可能既不是极小点,又不是极大点.

例如,对函数$f(x) = 3x^4 - 16x^3 + 30x^2 - 24x + 8$,它有两个驻点$x = 1$, $x = 2$,而$x = 2$是$f(x)$的极小点,$x = 1$则不是$f(x)$的极值点.若取$x_0 = 1.5$则不难验证,当$n \to \infty$时,牛顿法的点列$x_n \to 1$. 因此,为了保证牛顿法收敛到极小点,应要求$f''(x_n) > 0$至少对足够大的n成立.

3.2.1 抛物线法

- 基本思想 也是用二次函数$g(x)$来近似地代替原来的函数$f(x)$,并以它的极小点作为函数$f(x)$的近似极小点.

3.2.1 抛物线法

- 基本思想 也是用二次函数$g(x)$来近似地代替原来的函数$f(x)$,并以它的极小点作为函数$f(x)$的近似极小点.
- 牛顿法用$f(x)$在x_0点的二阶Taylor展开式作为$g(x)$来逼近$f(x)$,即利用x_0点的函数值$f(x_0)$及其一,二阶导数值$f'(x_0)$, $f''(x_0)$来构造二次函数$g(x)$.

3.2.1 抛物线法

- 基本思想 也是用二次函数$g(x)$来近似地代替原来的函数$f(x)$,并以它的极小点作为函数$f(x)$的近似极小点.
- 牛顿法用$f(x)$在x_0点的二阶Taylor展开式作为$g(x)$来逼近$f(x)$,即利用x_0点的函数值$f(x_0)$及其一,二阶导数值$f'(x_0), f''(x_0)$来构造二次函数$g(x)$.
- 不同之处抛物线法则是利用$f(x)$在三个点x_0, x_1, x_2处的函数值来构造一个二次函数$y = g(x) = a_0 + a_1 x + a_2 x^2$使它满足

$$\begin{cases} y_1 = g(x_0) = f(x_0) \\ y_2 = g(x_1) = f(x_1) \\ y_3 = g(x_2) = f(x_2) \end{cases} \tag{2-4}$$

则$g'(x) = a_1 + 2a_2 x$.令$g'(x) = 0$,得$\bar{x} = -\frac{a_1}{2a_2}$就是$f(x)$的近似极小点.

令$f(x_0)=f_0, f(x_1)=f_1, f(x_2)=x_2$,由(2-4)得

$$\begin{cases} a_0+a_1x_0+a_2x_0^2=f_0 \\ a_0+a_1x_1+a_2x_1^2=f_1 \\ a_0+a_1x_2+a_2x_2^2=f_2 \end{cases}$$

令$f(x_0)=f_0, f(x_1)=f_1, f(x_2)=x_2$,由(2-4)得

$$\begin{cases} a_0+a_1x_0+a_2x_0^2=f_0 \\ a_0+a_1x_1+a_2x_1^2=f_1 \\ a_0+a_1x_2+a_2x_2^2=f_2 \end{cases}$$

求解上述线性方程组可得

$$\begin{aligned} a_1 &= b_1-a_2(x_1+x_0) \\ a_2 &= (b_2-b_1)/(x_2-x_0) \end{aligned}$$

其中

$$\begin{aligned} b_1 &= (f_1-f_0)/(x_1-x_0) \\ b_2 &= (f_2-f_1)/(x_2-x_1), \end{aligned}$$

即可求得$f(x)$的近似极小点$\bar{x}$.

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种：

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种:
  (I) 若$|f(\bar{x}) - f(x_0)| < \varepsilon$(其中$\varepsilon$为已给的计算精度), 则结束,取$x^* \approx \bar{x}$,否则,在点$\bar{x}, x_0, x_1, x_2$中,选取使$f$值最小的点为新的$x_0$,并使新的$x_1, x_2$各是新的$x_0$近旁的左右两点,继续进行迭代,直到获得近似最优解为止.

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种:
  - (I) 若$|f(\bar{x}) - f(x_0)| < \varepsilon$(其中$\varepsilon$为已给的计算精度), 则结束,取$x^* \approx \bar{x}$,否则,在点$\bar{x}, x_0, x_1, x_2$中,选取使$f$值最小的点为新的$x_0$,并使新的$x_1, x_2$各是新的$x_0$近旁的左右两点,继续进行迭代,直到获得近似最优解为止.
  - (II) 若$|x_0 - \bar{x}| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$;否则继续进行迭代.

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种:
  - (I) 若$|f(\bar{x}) - f(x_0)| < \varepsilon$(其中$\varepsilon$为已给的计算精度), 则结束,取$x^* \approx \bar{x}$,否则,在点$\bar{x}, x_0, x_1, x_2$中,选取使$f$值最小的点为新的$x_0$,并使新的$x_1, x_2$各是新的$x_0$近旁的左右两点,继续进行迭代,直到获得近似最优解为止.
  - (II) 若$|x_0 - \bar{x}| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$;否则继续进行迭代.
  - (III) 若$|f(\bar{x}) - g(\bar{x})| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$,否则继续进行迭代.

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是g(x)的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种:
  - (I) 若$|f(\bar{x}) - f(x_0)| < \varepsilon$(其中$\varepsilon$为已给的计算精度), 则结束,取$x^* \approx \bar{x}$,否则,在点$\bar{x}, x_0, x_1, x_2$中,选取使$f$值最小的点为新的$x_0$,并使新的$x_1, x_2$各是新的$x_0$近旁的左右两点,继续进行迭代,直到获得近似最优解为止.
  - (II) 若$|x_0 - \bar{x}| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$;否则继续进行迭代.
  - (III) 若$|f(\bar{x}) - g(\bar{x})| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$,否则继续进行迭代.
  - (IV) 还可以采用其他的结束准则.

- 通常假设,$x_1 < x_0 < x_2, f(x_0) < f(x_1), f(x_0) < f(x_2)$,这时可以证明$a_2 > 0$,因而求得的$\bar{x}$确是$g(x)$的极小点,$\bar{x} \in [x_1, x_2]$
- 迭代的结束准则可以采用以下几种:
  - (I) 若$|f(\bar{x}) - f(x_0)| < \varepsilon$(其中$\varepsilon$为已给的计算精度), 则结束,取$x^* \approx \bar{x}$,否则,在点$\bar{x}, x_0, x_1, x_2$中,选取使$f$值最小的点为新的$x_0$,并使新的$x_1, x_2$各是新的$x_0$近旁的左右两点,继续进行迭代,直到获得近似最优解为止.
  - (II) 若$|x_0 - \bar{x}| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$;否则继续进行迭代.
  - (III) 若$|f(\bar{x}) - g(\bar{x})| < \varepsilon$,则迭代结束,取$x^* \approx \bar{x}$,否则继续进行迭代.
  - (IV) 还可以采用其他的结束准则.
- 抛物线法并不能保证算法一定收敛.但是若已知由抛物线法产生的点列$\{x_k\}$收敛于$f(x)$的极小点x^*,则可以证明:在一定的条件下,抛物线法是超线性收敛的,其收敛阶约为1.3.

3.2.3 三次插值法

基本思想

三次插值法是用a, b两点处的函数值$f(a), f(b)$和导数值$f'(a), f'(b)$来构造三次插值多项式$g(x)$,并以$g(x)$的极小点作为f(x)的近似极小点.一般来说,三次插值法比抛物线法的收敛速度要快些.

为了保证$f(x)$的极小点$x^* \in [a, b]$,假定$f(x) \in C^1$.

$$a < b,\ f'(a) < 0,\ f'(b) > 0 \tag{2-11}$$

设插值多项式为

$$g(x) = \alpha(x-a)^3 + \beta(x-a)^2 + \gamma(x-a) + \delta \tag{2-12}$$

由插值条件知$\alpha, \beta, \gamma, \delta$应满足

$$\begin{cases} \delta = f(a) \\ \gamma = f'(a) \\ \alpha(b-a)^3 + \beta(b-a)^2 + \gamma(b-a) + \delta = f(b) \\ 3\alpha(b-a)^2 + 2\beta(b-a) + \gamma = f'(b) \end{cases} \tag{2-13}$$

下面来求$f(x)$的近似极小点.

$$g'(x) = 3\alpha(x-a)^2 + 2\beta(x-a) + \gamma$$

令$g'(x) = 0$,分两种情况进行讨论.

- 当$\alpha = 0$时,插值多项式$g(x)$为二次多项式,

$$\hat{x} = a - \frac{\gamma}{2\beta} \tag{2-14}$$

- 当$\alpha \neq 0$时,

$$\hat{x} = a + \frac{-\beta \pm \sqrt{\beta^2 - 3\alpha\gamma}}{3\alpha} \tag{2-15}$$

为了保证所求的根为$g(x)$的极小点,在该点还需要满足

$$g''(x) = 6\alpha(x - a) + 2\beta > 0 \tag{2-16}$$

故(2-15)式中的根号前应取正号,并且(2-14)与(2-15)可合写为

$$\hat{x} = a - \frac{\gamma}{\beta + \sqrt{\beta^2 - 3\alpha\gamma}} \tag{2-17}$$

这就是计算$f(x)$的近似极小点的公式.

求解线性方程组(2-13)可得

$$\begin{cases} \delta = f(a) \\ \gamma = f'(a) \\ \beta = \frac{3u-v}{b-a} \\ \alpha = \frac{v-2u}{(b-a)^2} \end{cases}$$

其中

$$u = \frac{f(b)-\delta}{b-a} - \gamma, \quad v = f'(b) - \gamma.$$

将α, β, γ代入(2-17)式,即可求得$f(x)$的近似极小点$\hat{x}$. 若$|f'(\hat{x})| < \varepsilon$(其中$\varepsilon$为要求的精度),则取$x^* \approx \hat{x}$;否则继续进行迭代,当$f'(\hat{x}) > 0$时,用$\hat{x}$代替$b$,否则用$\hat{x}$代替$a$,直到求得所要的近似极小点为止.

3.2.4 平分法(对分法)

设$f(x) \in C^1$,x^*为$f(x)$的极小点.如果我们找到了一个区间$[a, b]$,且有$f'(a) < 0$和$f'(b) > 0$.则在a, b之间,必有$f(x)$的一个极小点x^*,使$f'(x^*) = 0$. 我们可以这样来求x^*:

- 取$x_0 = (a + b)/2$,
- 若$f'(x_0) > 0$,则$f(x)$在$[a, x_0]$中有极小点, 用$[a, x_0]$代替$[a, b]$,再取$x_1 = (a + x_0)/2$,
- 若$f'(x_0) < 0$,则$f(x)$在$[x_0, b]$中有极小点, 用$[x_0, b]$代替$[a, b]$,再取$x_1 = (x_0 + b)/2$;
- 经过k次迭代,设所得的小区间为$[a_k, b_k]$.
  若$|b_k - a_k| < \delta$,或$|f'(x_k)| < \varepsilon$时,取$x^* \approx x_k = (a_k + b_k)/2$,迭代结束;否则继续进行迭代.

平分法的优缺点

平分法的优点:

- 每一步的计算量小，程序简单;
- 总能收敛于一个局部极小点x^*.

缺点:收敛速度很慢.

3.2.5 “成功-失败”法

设要求解的问题为

$$\min_{a\le x\le b} F(x),$$

令

$$f(x)=\begin{cases} F(x), & x\in[a,b] \\ +\infty, & \text{否则} \end{cases}$$

则上述问题等价于无条件极值问题,

$$\min_{x\in\mathbf{R}} f(x).$$

以下方法叫做“成功-失败”法:

(1) 取定初始点$x_0 \in \mathbf{R}$,搜索步长$h > 0$及精度$\varepsilon > 0$;

(2) 计算$x_1 = x_0 + h, f(x_1)$;

(3) 若$f(x_1) < f(x_0)$,称为搜索成功,下一次搜索就大步前进,用x_1代替x_0,$2h$代替h,继续进行搜索.
若$f(x_1) \geq f(x_0)$,称为搜索失败,下一次搜索就小步后退.首先看是否有$|h| < \varepsilon$ 若$|h| < \varepsilon$,则取$x^* \approx x_0$,计算结束;否则,用$-h/4$代替h,返回第2步,继续进行搜索.

注意:初始搜索步长$h > 0$不能选得太小.

3.2.6 0.618法

考察一维搜索问题:

$$\begin{aligned} \min \quad & \varphi(\lambda) \\ \text{s.t.} \quad & a_1 \leq \lambda \leq b_1, \end{aligned} \tag{2-24}$$

其中$\varphi(\lambda)$为凸函数.

设$\varepsilon > 0$为允许的最后的搜索区间长度,令$\alpha = 0.618$,则0.618法的计算步骤如下:

1. 令$k = 1$. 计算

$$\lambda_1 = a_1 + (1-\alpha)(b_1 - a_1), \varphi(\lambda_1), \mu_1 = a_1 + \alpha(b_1 - a_1), \varphi(\mu_1).$$

2. 若$b_k - a_k < \varepsilon$,则计算结束,最优解$\lambda^* \in [a_k, b_k]$,可取$\lambda^* \approx (1/2)(b_k + a_k)$; 否则,若$\varphi(\lambda_k) > \varphi(\mu_k)$,则转3;
若$\varphi(\lambda_k) \leq \varphi(\mu_k)$, 则转4.
3. 令$a_{k+1} = \lambda_k, b_{k+1} = b_k$,再令

$$\lambda_{k+1} = \mu_k, \mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1}),$$

计算$\varphi(\mu_{k+1})$,转5.
4. 令$a_{k+1} = a_k, b_{k+1} = \mu_k$,再令

$$\mu_{k+1} = \lambda_k, \lambda_{k+1} = a_{k+1} + (1-\alpha)(b_{k+1} - a_{k+1}),$$

计算$\varphi(\lambda_{k+1})$,转5.
5. 令$k = k + 1$,返回2.

例2.3 求解$\min \varphi(\lambda) = \lambda^2 + 2\lambda, \text{s.t.} -3 \le \lambda \le 5$,取$\varepsilon = 0.2$.

1.
$$\lambda_1 = -3 + 0.382 * 8 = 0.056$$
$$\mu_1 = -3 + 0.618 * 8 = 1.944$$
$$\varphi(\lambda_1) = \varphi(0.056) = 0.115$$
$$\varphi(\mu_1) = \varphi(1.944) = 7.667$$

2. 由于$\varphi(\lambda_1) < \varphi(\mu_1)$,所以
$$a_2 = a_1 = -3,\ b_2 = \mu_1 = 1.944$$
新的搜索区间为$[-3, 1.944]$.

3.
$$\lambda_2 = -3 + 0.382 * (1.944 + 3) = -1.112$$
$$\mu_2 = \lambda_1 = 0.056$$
$$\varphi(\lambda_2) = \varphi(-1.112) = -0.987$$
$$\varphi(\mu_2) = 0.115$$
所以$a_3 = a_2 = -3, b_3 = \mu_2 = 0.056$.

4. 重复计算到$k = 9$,可得$a_9 = -1.112, b_9 = -0.936$.因为$|b_9 - a_9| = 0.176 < \varepsilon$,所以计算结束,$x^* \approx (a_9 + b_9)/2 = -1.024$.

3.2.7 确定初始搜索区间和初始点的方法

前面讲的一维搜索方法,有的需要一个初始的搜索区间,有些需要一个初始点.下面给出一种进退算法,它可以同时确定初始的搜索区间和初始点.设要求解的问题为$\min \varphi(x)$.

(1) 选定初始点的一个估计值t_0,初始步长$h > 0$,计算$\varphi(t_0)$.

(2) 令$t_2 = t_0 + h$,计算$\varphi(t_2)$.

(3) 若$\varphi(t_2) \leq \varphi(t_0)$,转第4步;否则,若$\varphi(t_2) > \varphi(t_0)$, 令$h = -h$,转第4步.

(4) 令$t_1 = t_0 + h$,计算$\varphi(t_1)$.

(5) 若$\varphi(t_1) \leq \varphi(t_0)$,则令$h = 2h, t_2 = t_0, t_0 = t_1$,转第4步; 否则,若$\varphi(t_1) > \varphi(t_0)$,转第6步.

(6) 令$a = \min(t_1, t_2), b = \max(t_1, t_2)$,则$[a, b]$即为所求的初始搜索区间,而$t_0 = (a + b)/2$可作为所求的初始点,计算结束.

3.2.8 不精确一维搜索

在实际计算中,一般做不到精确的一维搜索,实际上,也没有必要做到这一点. 因为精确的一维搜索需要付出较高的代价,而对加速收敛作用不大. 所以不精确一维搜索方法受到了广泛的重视和欢迎.
设要求$\min \varphi(\lambda) = f(x^k + \lambda s^k)$,在不精确一维搜索中,通常要求$f(x^{k+1})$比$f(x^k)$下降一定的数量,而且在新点$x^{k+1} = x^k + \lambda_k s^k$处沿方向$s^k$的方向导数值比在$x^k$点沿$s^k$的方向导数值大一定的数量,这里"一定的数量"是用$-\nabla f(x^k)^T s^k > 0$来度量的.

通常采用的是Wolfe-Powell不精确一维搜索准则,简称Wolfe-Powell准则.
即对给定的常数$c_1, c_2, 0 < c_1 < c_2 < 1$, 要求$\lambda_k$满足如下条件:

(i)

$$f(x^k) - f(x^{k+1}) \geq -c_1 \lambda_k \nabla f(x^k)^T s^k \tag{2-25}$$

(ii)

$$\nabla f(x^{k+1})^T s^k \geq c_2 \nabla f(x^k)^T s^k \tag{2-26}$$

根据计算经验,常取$c_1 = 0.1, c_2 = 0.5$.

不精确一维搜索算法一(直接法)

计算步骤

设点x^k,搜索方向s^k已求得,为书写简便起见,令

$$f_k \triangleq f(x^k), g_k \triangleq \nabla f(x^k).$$

(1) 给定$c_1 \in (0,1), c_2 \in (c_1,1)$,令$a=0, b=+\infty, \lambda=1, j=0$.

(2) 令$x^{k+1}=x^k+\lambda s^k$,计算f_{k+1}, g_{k+1},若λ满足条件(i),(ii), 则令$\lambda_k=\lambda$,计算结束;否则令$j=j+1$,若λ不满足条件(i),则转第3步;若λ 满足条件(i)不满足条件(ii),则转第4步.

(3) 令$b=\lambda, \lambda=(\lambda+a)/2$,返回第2步.

(4) 令$a=\lambda, \lambda=\min\{2\lambda,(\lambda+b)/2\}$,返回第2步.

例2.4

设

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

已求得

$$x^k = (0,0)^T, s^k = (1,0)^T,$$

试确定在x^k点,沿方向s^k的步长λ_k,使条件(i),(ii)成立.

解

$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

$f_k = f(0,0) = 1, g_k = (-2,0)^T, g_k^T s_k = -2$.给定$c_1 = 0.1, c_2 = 0.5$.

j	x^k	f_k	λ	x^{k+1}	f_{k+1}	条件(i)	条件(ii)
0	$(0,0)^T$	1	1	$(1,0)^T$	100	-	
1	$(0,0)^T$	1	0.5	$(0.5,0)^T$	6.25	-	
2	$(0,0)^T$	1	0.25	$(0.25,0)^T$	0.953	-	
3	$(0,0)^T$	1	0.125	$(0.125,0)^T$	0.790	+	+

不精确一维搜索算法二(二次插值法)

设$\varphi(\lambda)=f(x^k+\lambda s^k)$,若已知$(\lambda_1,\varphi_1),(\lambda_2,\varphi_2)$, 和$(\lambda_1,\varphi_1')$,其中$\varphi_1'=\varphi'(\lambda_1)$, $\varphi_i=\varphi(\lambda_i)(i=1,2)$. 求通过$(\lambda_1,\varphi_1),(\lambda_2,\varphi_2)$两点和在点$(\lambda_1,\varphi_1)$处具有相同切线的二次曲线方程$g(\lambda)=a\lambda^2+b\lambda+c$,则

$$g(\lambda_1)=\varphi_1, g(\lambda_2)=\varphi_2, g'(\lambda_1)=\varphi_1'$$

因为$g'(\lambda)=2\lambda a+b$,所以

$$\hat{\lambda}=-b/2a \tag{2-27}$$

只要算出a和b,即可求得$\hat{\lambda}$,用$\hat{\lambda}$作为$\varphi(\lambda)$ 的近似极小点,可以证

$$\hat{\lambda}=\lambda_1+\frac{1}{2}(\lambda_2-\lambda_1)/[1+\frac{\varphi_1-\varphi_2}{(\lambda_1-\lambda_2)\varphi_1'}] \tag{2-28}$$

算法步骤

(1) 给定$c_1 \in (0,1), c_2 \in (c_1,1), T > 0$.令$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = +\infty, j = 0$

(2) 若λ_2满足条件(i),(ii),则令$\lambda_k = \lambda_2$,计算结束;否则转第3步.

(3) 计算$\varphi_1, \varphi_2, \varphi_1' = \nabla f(x^k + \lambda_1 s^k)^T s^k$和$\hat{\lambda}$, 令j=j+1.

(4) 若$\lambda_1 < \hat{\lambda} < \lambda_2$,则令$\lambda_3 = \lambda_2, \lambda_2 = \hat{\lambda}$, 返回第2步;若$\lambda_2 \leq \hat{\lambda} < \lambda_3$则令$\lambda_1 = \lambda_2, \lambda_2 = \hat{\lambda}$,返回第2步;否则转第5步.

(5) 若$\hat{\lambda} \leq \lambda_1$,则令$\hat{\lambda} = \lambda_1 + T\delta\lambda$; 若$\hat{\lambda} \geq \lambda_3$,则令$\hat{\lambda} = \lambda_3 - T\delta\lambda$, 其中$\delta\lambda = \lambda_3 - \lambda_1$,令$\lambda_2 = \hat{\lambda}$,返回第2步.

3.3 求多变量函数极值的基本下降法

本节介绍求多个变量函数的无约束极小值的一些基本的下降算法:

- 最速下降法
- Newton法
- 阻尼Newton法

3.3.1 最速下降法

基本思想

设$f(x) \in C^1$.由于函数$f(x)$沿负梯度方向$-\nabla f(x)$下降最快,所以选取它作为每次迭代的搜索方向,即$s^k = -\nabla f(x^k)$.最速下降法也称为梯度法.

计算步骤

任给一个初始点x^0,计算$s^0 = -\nabla f(x^0)$,从x^0出发沿s^0方向在半直线$x^0 + \lambda s^0 (\lambda \geq 0)$求得$f(x)$的第一个近似极小点$x^1 = x^0 + \lambda_0 s^0$,再以$x^1$为起点,重复上述过程求得$f(x)$的第二个近似极小点$x^2 = x^1 + \lambda_1 s^1$, 直到得到满意的结果为止.

最速下降方向

为什么在x^0的附近,$f(x)$沿$-\nabla f(x^0)$的方向可使函数值下降最快? 设p为任给的一个单位向量,ε为正实数,则$f(x)$在x^0处的Taylor 展开式为

$$f(x^0+\varepsilon p)=f(x^0)+\varepsilon(\nabla f(x^0))^T p+o(\varepsilon),$$

其中$o(\varepsilon)$是ε的高阶无穷小,而

$$\begin{aligned}(\nabla f(x^0))^T p &= \|\nabla f(x^0)\|\cdot\|p\|\cdot\cos(\nabla f(x^0),p)\\ &= \|\nabla f(x^0)\|\cos(\nabla f(x^0),p)\end{aligned}$$

所以当$(\nabla f(x^0),p)=\pi$时,$f(x)$在x^0点附近下降最快.因此, 称$-\nabla f(x^0)$为$f(x)$在x^0点的最速下降方向.最速下降法的名称就由此而来.

梯度法的迭代步骤

(1) 给定初始点x^0,精度$\varepsilon > 0$,令$k = 0$;

(2) 计算$\nabla f(x^k)$;

(3) 若$\|\nabla f(x^k)\| < \varepsilon$,则迭代结束,取$x^* = x^k$,否则转到(4);

(4) 这时$\|\nabla f(x^k)\| \geq \varepsilon$,用精确一维搜索求

$$\varphi(\lambda) = f(x^k - \lambda \nabla f(x^k))$$

的一个极小点λ_k,使$f(x^k - \lambda \nabla f(x^k)) < f(x^k)$;

(5) 令$x^{k+1} = x^k - \lambda_k \nabla f(x^k), k = k + 1$, 返回(2).

收敛性定理

设$f:\mathbf{R}^n\to\mathbf{R}^1$连续可微,$x^0\in\mathbf{R}^n$,如果水平集

$$L=\{x|f(x)\le f(x^0)\}$$

有界,则梯度法或在有限步迭代后停止,或者得到点列$\{x^k\}$,它的任何极限点都是$f(x)$的稳定点.

若进一步假设$f(x)$为凸函数,则应用梯度法,或在有限步迭代后达到$f(x)$的最小点;或者点列$\{x^k\}$ 的任何极限点都是$f(x)$的最小点.

优缺点评述 I

优点:

1. 方法简单,每迭代一次的工作量较小,所需要的存贮量也少;
2. 从一个不好的初始点x^0出发,也能保证算法的收敛性.

缺点:

1. 在极小点附近收敛得很慢.从表面上看,这似乎与"最速下降"矛盾;其实不然,因为梯度是函数的局部性质,从局部看在一点附近下降最快,但从总体上来看可能走许多弯路. 事实上,若x^{k+1}是沿方向$s^k = -\nabla f(x^k)$用一维搜索求得的,即λ_k为$\varphi(\lambda) = f(x^k + \lambda s^k)$的极小点,那么

$$\varphi'(\lambda_k) = (\nabla f(x^k + \lambda_k s^k))^T s^k = 0$$

即$(\nabla f(x^{k+1}))^T \nabla f(x^k) = 0$.这说明梯度法逐次下降的方向是彼此正交的, 因此在极小点附近,梯度法产生的点列逼近函数的极小点越来越慢.

优缺点评述 II

2 梯度法的收敛速度与变量的尺度关系很大.例如,
设$f(x_1, x_2) = x_1^2 + x_2^2$, 取$x^0 = (2, 2)^T$,进行一维搜索, 则迭代一次就达到极小点.但是,对于$f(x_1, x_2) = x_1^2 + 25x_2^2$,取$x^0 = (2, 2)^T$,则需进行10次迭代, 才能达到极小点.

3 梯度法关于小的扰动是不稳定的, 而小的扰动在计算过程中是容易产生的,例如舍入误差的存在,或一维搜索时步长确定得不准确, 这样就可能破坏方法的收敛性.

3.3.2 Newton法

与一维问题类似,在局部用一个二次函数$\varphi(x)$近似地代替目标函数$f(x)$,然后用$\varphi(x)$ 的极小点作为$f(x)$的近似极小点.

设x^k为$f(x)$的一个近似极小点,将$f(x)$在x^k点进行Taylor展开, 并略去高于二次的项,则得

$$f(x) \approx \varphi(x) = f(x^k) + (\nabla f(x^k))^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k) \tag{3-1}$$

令$\nabla\varphi(x) = 0$,得$\varphi(x)$的极小点为

$$\hat{x} = x^k - (\nabla^2 f(x^k))^{-1} \nabla f(x^k) \tag{3-2}$$

取$\hat{x}$作为$f(x)$的近似极小点,这样就得到Newton法的迭代公式

$$x^{k+1} = x^k - (\nabla^2 f(x^k))^{-1} \nabla f(x^k). \tag{3-2'}$$

收效速度及优缺点评述

设x^*为$f(x)$的极小点,可以证明,在适当的条件下,Newton法的收敛速度估计式为

$$\|x^{k+1} - x^*\| < M\|x^k - x^*\|^2 \tag{3-3}$$

即Newton法至少是二阶收敛的.所以,Newton法的突出优点是收敛速度快, 而缺点是:

(1) 要求$f(x)$二阶连续可微;

(2) 在迭代中,要计算$[\nabla^2 f(x)]^{-1}$是困难的;

(3) 从(3-3)式可看出,初始点x^0不能离极小点x^*太远, 否则迭代可能不收敛.为了克服这个缺点,人们对算法作了修正,提出了阻尼Newton法,又称为修正Newton法.

3.3.3 阻尼Newton法

在Newton法中,步长λ_k总是取为1.在阻尼Newton法中,每步迭代沿方向$s^k = -(\nabla^2 f(x^k))^{-1}\nabla f(x^k)$进行一维搜索来决定$\lambda_k$, 即取$\lambda_k$使

$$f(x^k + \lambda_k s^k) = \min_{\lambda \geq 0} f(x^k + \lambda s^k)$$

而用迭代公式

$$x^{k+1} = x^k - \lambda_k(\nabla^2 f(x^k))^{-1}\nabla f(x^k) \tag{3-4}$$

来代替(3-2').阻尼牛顿法保持了牛顿法快速收敛的优点, 又不要求初始点选得很好,因而在实际应用中取得了较好的效果. 当然(3-4)式也要求海赛矩阵$\nabla^2 f(x^k)$的逆矩阵.

迭代程序

(1) 给定初始点x^0,精度$\varepsilon > 0$,令$k = 0$.

(2) 计算$\nabla f(x^k)$,若$\|\nabla f(x^k)\| < \varepsilon$成立,则迭代停止,$x^k$即为所求; 否则转到(3).

(3) 计算$[\nabla^2 f(x^k)]^{-1}$及$s^k = -[\nabla^2 f(x^k)]^{-1}\nabla f(x^k)$.

(4) 沿s^k进行一维搜索,决定步长λ^k.

(5) 令$x^{k+1} = x^k + \lambda_k s^k$, $k = k + 1$,返回(2).

迭代程序

(1) 给定初始点x^0,精度$\varepsilon > 0$,令$k = 0$.

(2) 计算$\nabla f(x^k)$,若$\|\nabla f(x^k)\| < \varepsilon$成立,则迭代停止,$x^k$即为所求; 否则转到(3).

(3) 计算$[\nabla^2 f(x^k)]^{-1}$及$s^k = -[\nabla^2 f(x^k)]^{-1}\nabla f(x^k)$.

(4) 沿s^k进行一维搜索,决定步长λ^k.

(5) 令$x^{k+1} = x^k + \lambda_k s^k$, $k = k + 1$,返回(2).

收敛性质

设$f(x) \in C^2$,$\nabla^2 f(x)$为正定矩阵,水平集$\{x|f(x) \leq f(x^0)\}$有界, 则由阻尼牛顿法得到的点列$\{x^k\}$有如下的性质:

- $\{f(x^k)\}$为严格单调下降数列;
- $\{x^k\}$的任一极限点$\hat{x}$必为$f(x)$的极小点.

3.4 共轭方向法和共轭梯度法

- 最速下降法,计算步骤简单,但收敛速度太慢.
- 牛顿法和阻尼牛顿法收敛速度快,但要计算二阶偏导数矩阵及其逆矩阵,计算量太大.
- 共轭方向法比最速下降法的收敛速度要快得多,同时又避免了像牛顿法所要求的海赛矩阵的计算,存贮和求逆.
- 共轭方向法,主要是其中的共轭梯度法,它对一般目标函数的无约束优化问题的求解具有较高的效率.它的计算公式简单,存贮量少,可以用来求解比较大的问题.
- 由于二次函数是一种简单的非线性函数,而一般的目标函数$f(x)$在极小点附近的性态(当$f(x) \in C^2$时)又近似于二次函数, 所以人们常常把二次函数作为衡量算法优劣的尺子.

3.4.1 共轭方向和它的一些性质

定义4.1

设A为n阶对称矩阵,p, q为n维列向量, 若

$$p^T Aq = 0 \tag{4-1}$$

则称向量p和q为$A-$正交,或关于$A-$共轭.

3.4.1 共轭方向和它的一些性质

定义4.1

设A为n阶对称矩阵,p, q为n维列向量, 若

$$p^T Aq = 0 \tag{4-1}$$

则称向量p和q为$A-$正交,或关于$A-$共轭.

- 如果$A = I_n$,则(4-1)成为$p^T q = 0$,这就是通常意义下的正交,故$A-$共轭或$A-$正交是正交概念的推广.
- 如果对于有限个向量$p_1, p_2, \cdots, p_m$,有

$$p_i^T Ap_j = 0 (i \neq j, i, j = 1, 2, \cdots, m)$$

成立, 则称这个向量组为$A-$正交(或共轭)向量组,也称它们是一组$A-$共轭方向.

定理4.1

向量

$$p_i \in \mathbf{R}^n,\ i = 1, 2, \cdots, n$$

是线性无关的,$q \in \mathbf{R}^n$,若q与$p_1, p_2 \cdots, p_n$ 都正交,则$q = 0$.

定理4.1

向量

$$p_i \in \mathbf{R}^n,\ i = 1, 2, \cdots, n$$

是线性无关的,$q \in \mathbf{R}^n$,若q与$p_1, p_2 \cdots, p_n$ 都正交,则$q = 0$.

定理4.2

设A为n阶正定矩阵,$p_1, p_2, \cdots, p_n$为$A-$共轭的n维非零向量组,则此向量组必线性无关.

证明.

设$k_1 p_1 + k_2 p_2 + \cdots + k_n p_n = 0$,只需证明$k_1, k_2, \cdots, k_n$全部为零. □

下面来研究n元二次函数的无约束优化问题:

$$\min \quad f(x) = \frac{1}{2}x^T Hx + b^T x + c \tag{4-2}$$

其中c为常数,x, b为n维列向量, H为n阶对称正定矩阵.这时$f(x)$有唯一的极小点$x^* = -H^{-1}b$. 对n元二次函数$f(x)$,我们有下述结果.

定理4.3

设H为n阶对称正定矩阵,$s^0, s^1, s^2, \cdots, s^{n-1} \in \mathbf{R}^n$是一组$H-$共轭方向,对问题(4-2), 若从任一初始点$x^0$出发,依次沿方向$s^0, s^1, s^2, \cdots, s^{n-1}$进行精确一维搜索.则至多经过$n$次迭代, 即可求得$f(x)$的最小点.

3.4.2 共轭方向的生成和共轭梯度法

- 由上述可见,只要能选取一组$H-$共轭的方向$s^0, s^1, \cdots, s^{n-1}$,就可以用上述方法在$n$步之内求得$n$元二次函数$f(x)$ 的极小点.我们称上述算法为共轭方向法.这种算法对于形如(4-2)的二次函数,具有有限步收敛的性质.
- 共轭方向的选取有很大的任意性，而对应于不同的一组共轭方向就有不同的共轭方向法. 作为一种迭代算法,我们自然希望共轭方向能在迭代过程中逐次生成.
- 下面介绍一种生成共轭方向的方法,它是利用每次一维最优化所得到的点处的梯度来生成共轭方向,这种方法称为共轭梯度法.

共轭梯度法

任给初始点x^0,令$s^0 = -\nabla f(x^0)$,由精确一维搜索得到λ_0,令

$$x^1 = x^0 + \lambda_0 s^0, \quad s^1 = -\nabla f(x^1) + \mu_{10} s^0$$

其中μ_{10}为待定系数,要使s^0, s^1是共轭的,即要使$(s^1)^T H s^0 = 0$ 成立,由此可求得

$$\mu_{10} = \frac{\nabla f(x^1)^T H s^0}{(s^0)^T H s^0} \tag{4-8}$$

将μ_{10}代入s^1的表达式中,就可以求出搜索方向s^1,令

$$x^2 = x^1 + \lambda_1 s^1, \quad s^2 = -\nabla f(x^2) + \mu_{20} s^0 + \mu_{21} s^1$$

其中λ_1由一维精确搜索确定.而μ_{20}和μ_{21}为待定系数,要使s^2与s^0, s^1分别是$H-$共轭的,即要$(s^2)^T H s^1 = 0, \quad (s^2)^T H s^0 = 0$ 由此可求得

$$\mu_{20} = \frac{\nabla f(x^2)^T H s^0}{(s^0)^T H s^0}, \ \mu_{21} = \frac{\nabla f(x^2)^T H s^1}{(s^1)^T H s^1}$$

类似地可得

$$\begin{cases} s^i = -\nabla f(x^i) + \sum_{j=0}^{i-1} \mu_{ij} s^j, \ 0 \le i \le n-1 \\ \mu_{ij} = \frac{\nabla f(x^i)^T H s^j}{(s^j)^T H s^j}, \ j = 0, 1, \cdots, i-1 \end{cases} \tag{4-9}$$

应用上述方法就可生成n个关于$H-$共轭的方向$s^0, s^1, \cdots, s^{n-1}$.但是仔细分析一下上述公式就会发现,在求$\mu_{ij}$时,需要用到$f(x)$ 的海赛矩阵H,这是我们所不希望的.因此,需要对上述公式作进一步的化简,使在生成共轭方向$s^0, s^1, \cdots, s^{n-1}$时,不需要直接用到海赛矩阵$H$.因为

$$\nabla f(x) = b + Hx,$$

为以后书写简便,令

$$\nabla f(x) = g(x),\ \nabla f(x^i) = g(x^i) = g_i,$$

故$g_i = b + Hx^i, g_{i+1} = b + Hx^{i+1}$

$$g_{i+1} - g_i = H(x^{i+1} - x^i). \tag{4-10}$$

但是$x^{i+1} = x^i + \lambda s^i$,所以

$$g_{i+1} - g_i = \lambda_i H s^i. \tag{4-11}$$

因此(4-9)式中μ_{ij}可改写为

$$\mu_{ij} = \frac{g_i^T(g_{j+1} - g_j)}{(s^j)^T(g_{j+1} - g_j)},\ j = 0, 1, \cdots, i-1. \tag{4-12}$$

注意到:$s^0 = -g_0$及$g_1^T g_0 = 0$,由(4-12)式易得

$$\mu_{10} = \|g_1\|^2 / \|g_0\|^2 \tag{4-13}$$

由于$g_2 \perp g_0, g_2 \perp g_1$,易得$\mu_{20} = 0$.

由(4-9)式的第一个式子得$s^1 = -g_1 + \mu_{10}s^0 = -g_1 - \mu_{10}g_0$, 在(4-12)式中,当$i = 2, j = 1$时,并将上式代入得

$$\mu_{21} = \frac{g_2^T(g_2 - g_1)}{(s^1)^T(g_2 - g_1)} = \|g_2\|^2/\|g_1\|^2. \tag{4-14}$$

类似地可得

$$\mu_{ij} = 0,\ 0 \le j \le i - 2 \tag{4-15}$$

$$\mu_{i,i-1} = \|g_i\|^2/\|g_{i-1}\|^2, \tag{4-16}$$

其中$\|g_i\| \neq 0, i = 0, 1, \cdots, n - 1$.因此

$$s^i = -g_i + \mu_{i,i-1}s^{i-1}. \tag{4-17}$$

由(4-16),(4-17)式求得的方向$s^0, s^1, \cdots, s^{n-1}$是关于$H-$共轭的. 按照这些公式确定共轭方向的算法就是通常所说的F-R共轭梯度法,它是由Fletcher和Reeves [a]于1964年提出的.因为公式中不显含海赛矩阵,所以可毫无困难地将此方法应用于一般的目标函数.

---

[a] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, The Computer Journal(1964), 7(2):149-154

3.4.3 共轭梯度法的计算步骤

1. 给定初始点x^0及精度$\varepsilon > 0$.
2. 计算$g_0 = \nabla f(x^0)$,令$s^0 = -g_0, k = 0$;
3. 求

$$\min_{\lambda \geq 0} \ f(x^k + \lambda s^k)$$

   决定λ_k,计算$x^{k+1} = x^k + \lambda_k s^k, g_{k+1} = \nabla f(x^{k+1})$.
4. 若$\|g_{k+1}\| < \varepsilon$,迭代结束,否则转5.
5. 若$k < n-1$,计算

$$\mu_{k+1} \triangleq \mu_{k+1,k} = \|g_{k+1}\|^2 / \|g_k\|^2, \ s^{k+1} = -g_{k+1} + \mu_{k+1} s^k$$

   令$k = k+1$,转回3;若$k = n-1$,令$x^0 = x^n$,转回2.

3.4.4 应用举例 I

例4.1

求二次函数$f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 2x_1x_2$的极小点.

解

1. 取初始点$x^0 = (1, 1)^T$,则

$$f(x^0) = -3.\ \frac{\partial f}{\partial x_1} = 2x_1 - 4 - 2x_2,\ \frac{\partial f}{\partial x_2} = 4x_2 - 2x_1.$$

2. $s^0 = -g_0 = \begin{pmatrix} 4 \\ -2 \end{pmatrix}$.因$x^0 + \lambda s^0 = \begin{pmatrix} 1 + 4\lambda \\ 1 - 2\lambda \end{pmatrix}$,故$f(x^0 + \lambda s^0) = 40\lambda^2 - 20\lambda - 3$.由$\frac{df}{d\lambda} = 0$,得$\lambda_0 = 0.25$.

$$x^1 = x^0 + \lambda_0 s^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 0.25 \begin{pmatrix} 4 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0.5 \end{pmatrix}$$

3.4.4 应用举例 II

3.

$$f(x^1) = -5.5,\ g_1 = \nabla f(x^1) = \begin{pmatrix} -1 \\ -2 \end{pmatrix},$$

$$\mu_1 = \|g_1\|^2 / \|g_0\|^2 = 0.25$$

$$s^1 = -g_1 + \mu_1 s^0 = \begin{pmatrix} 2 \\ 1.5 \end{pmatrix}$$

$$x^1 + \lambda s^1 = \begin{pmatrix} 2 + 2\lambda \\ 0.5 + 1.5\lambda \end{pmatrix}$$

$$f(x^1 + \lambda s^1) = 2.5\lambda^2 - 5\lambda - 5.5,\ \frac{df}{d\lambda} = 5\lambda - 5, \lambda_1 = 1$$

$$x^2 = x^1 + \lambda_1 s^1 = \begin{pmatrix} 2 \\ 0.5 \end{pmatrix} + 1 \begin{pmatrix} 2 \\ 1.5 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

4. $\nabla f(x^2) = (0,0)^T$,即$\|g_2\| = 0$.因此x^2为极小点.

例4.2

求$f(x_1,x_2)=x_1^3+x_2^3-3x_1x_2$的极小点.

解

1. $\frac{\partial f}{\partial x_1}=3x_1^2-3x_2$, $\frac{\partial f}{\partial x_2}=3x_2^2-3x_1$.取初始点$x^0=(2,2)^T$,
   则$\nabla f(x^0)=g_0=(6,6)^T$.
2. $s^0=-g_0=\begin{pmatrix}-6\\-6\end{pmatrix}$, $x^0+\lambda s^0=\begin{pmatrix}2-6\lambda\\2-6\lambda\end{pmatrix}$,所
   以$f(x^0+\lambda s^0)=4(1-3\lambda)^2(1-12\lambda)$, $\frac{df}{d\lambda}=-72(1-3\lambda)(1-6\lambda)$.
   令$\frac{df}{d\lambda}=0$得$\lambda=1/3$或$1/6$.
3. 当$\lambda_0=1/6$时,$x^1=x^0+\lambda_0 s^0=(1,1)^T$, $\nabla f(x^1)=(0,0)^T$.又因

$$\nabla^2 f(x)=\begin{pmatrix}6x_1 & -3\\ -3 & 6x_2\end{pmatrix},\ \nabla^2 f(1,1)=\begin{pmatrix}6 & -3\\ -3 & 6\end{pmatrix}$$

   是正定的,因此$x^1=(1,1)^T$为严格极小点.
4. 当$\lambda_0=1/3$时,易验证$x^1=(0,0)^T$不是极小点.

3.5 变尺度法(Variable Metric Method)

变尺度法是由W.C.Davidon于1959年首先提出,1963年为R.Flecher和M.J.D.Powell 所发展,形成为著名的DFP变尺度算法,目前变尺度算法已发展成为一大类算法, 被认为是求解无约束优化问题最有效的算法之一,得到了广泛的研究和应用,也称为拟牛顿法(Quasi-Newton Method).

3.5.1 基本思想

最速下降法计算方法简便,但收敛速度太慢; 牛顿法收敛速度快,但计算量太大,实现困难. 能否设计一种算法,它既具有快速收敛的优点,又能不计算二阶偏导数矩阵及其逆阵, 就可以构造出每次迭代的搜索方向呢?

分析最速下降法和阻尼牛顿法的计算公式,发现它们可以用

$$x^{k+1} = x^k - \lambda_k H_k \nabla f(x^k) \tag{5-1}$$

来统一描述.

3.5.1 基本思想

最速下降法计算方法简便,但收敛速度太慢; 牛顿法收敛速度快,但计算量太大,实现困难. 能否设计一种算法,它既具有快速收敛的优点,又能不计算二阶偏导数矩阵及其逆阵, 就可以构造出每次迭代的搜索方向呢?

分析最速下降法和阻尼牛顿法的计算公式,发现它们可以用

$$x^{k+1} = x^k - \lambda_k H_k \nabla f(x^k) \tag{5-1}$$

来统一描述.

- 若$H_k = I$(I为单位矩阵),则得最速下降法的计算公式.
- 若$H_k = [\nabla^2 f(x^k)]^{-1}$,则得阻尼牛顿法的计算公式.特别地,若步长$\lambda = 1$, 则得到牛顿法的迭代公式.

为了保持牛顿法的优点, 我们希望H_k能近似地等于$[\nabla^2 f(x^k)]^{-1}$,并且希望H_k能在迭代计算中逐次生成,即

$$H_{k+1} = H_k + C_k \tag{5-2}$$

其中C_k称为修正矩阵,C_k不同,就可以得到不同的算法.

为了保持牛顿法的优点, 我们希望H_k能近似地等于$[\nabla^2 f(x^k)]^{-1}$,并且希望H_k能在迭代计算中逐次生成,即

$$H_{k+1} = H_k + C_k \tag{5-2}$$

其中C_k称为修正矩阵,C_k不同,就可以得到不同的算法.
为了研究使$H_k \approx [\nabla^2 f(x^k)]^{-1}$的条件,我们来考察$n$元二次函数

$$f(x) = a + b^T x + \frac{1}{2} x^T G x$$

其中$b, x \in \mathbf{R}^n$,G为n阶对称正定矩阵,a为常数.
令$g(x) = \nabla f(x), g_k = g(x^k)$,则

$$g_{k+1} - g_k = G(x^{k+1} - x^k).$$

令

$$\begin{cases} x^{k+1} - x^k = \Delta x_k \\ g_{k+1} - g_k = \Delta g_k, \end{cases} \tag{5-3}$$

则上式可写成

$$G\Delta x_k = \Delta g_k \tag{5-4}$$

或

$$\Delta x_k = G^{-1}\Delta g_k \tag{5-5}$$

令

$$\begin{cases} x^{k+1} - x^k = \Delta x_k \\ g_{k+1} - g_k = \Delta g_k, \end{cases} \tag{5-3}$$

则上式可写成

$$G\Delta x_k = \Delta g_k \tag{5-4}$$

或

$$\Delta x_k = G^{-1}\Delta g_k \tag{5-5}$$

对于一般的n元非二次函数$f(x) \in C^2$,可以将$f(x)$在点x^{k+1}进行Taylor展开,取其前三项,则可得

$$f(x) \approx f(x^{k+1}) + g_{k+1}^T(x - x^{k+1}) + (x - x^{k+1})^T G_{k+1}(x - x^{k+1})/2$$

其中$G_{k+1} = \nabla^2 f(x^{k+1})$.

因此$g(x) \approx g_{k+1} + G_{k+1}(x - x^{k+1})$, 所以$g_{k+1} - g_k \approx G_{k+1}\Delta x_k$ 即

$$G_{k+1}\Delta x_k \approx \Delta g_k \tag{5-4'}$$

或

$$\Delta x_k \approx G_{k+1}^{-1}\Delta g_k \tag{5-5'}$$

称(5-4)和(5-5)为拟牛顿方程.为了使$H_k \approx G_k^{-1}$,应要求H_{k+1}满足拟牛顿方程(5-4)或(5-5),即

$$H_{k+1}^{-1}\Delta x_k = \Delta g_k \tag{5-6}$$

或

$$\Delta x_k = H_{k+1}\Delta g_k. \tag{5-7}$$

3.5.2 对称秩1算法

为了使迭代计算简单易行,修正矩阵C_k应选取尽可能简单的形式, 通常是要求C_k的秩越小越好. 若要求C_k是秩为1的对称矩阵,则可设

$$C_k = \alpha_k u u^T \tag{5-8}$$

其中$\alpha_k \neq 0$为待定常数,$u = (u_1, u_2, \cdots, u_n)^T \neq 0$. 将(5-8)式代入(5-2)式得$H_{k+1} = H_k + \alpha_k u u^T$ 将上式代人(5-7)得

$$H_k \Delta g_k + \alpha_k u (u^T \Delta g_k) = \Delta x_k.$$

由于$\alpha_k, u^T \Delta g_k$为数量,所以u与$\Delta x_k - H_k \Delta g_k$成比例,不妨取

$$u = \Delta x_k - H_k \Delta g_k,$$

故

$$\alpha_k = \frac{1}{u^T \Delta g_k} = \frac{1}{\Delta g_k^T (\Delta x_k - H_k \Delta g_k)}.$$

其中$\Delta g_k^T (\Delta x_k - H_k \Delta g_k) \neq 0$.因此

$$H_{k+1} = H_k + \frac{(\Delta x_k - H_k \Delta g_k)(\Delta x_k - H_k \Delta g_k)^T}{\Delta g_k^T (\Delta x_k - H_k \Delta g_k)} \tag{5-9}$$

(5-9)式称为对称秩1公式.由对称秩1公式确定的变尺度算法称为对称秩1变尺度算法.

故

$$\alpha_k = \frac{1}{u^T \Delta g_k} = \frac{1}{\Delta g_k^T (\Delta x_k - H_k \Delta g_k)}.$$

其中$\Delta g_k^T (\Delta x_k - H_k \Delta g_k) \neq 0$.因此

$$H_{k+1} = H_k + \frac{(\Delta x_k - H_k \Delta g_k)(\Delta x_k - H_k \Delta g_k)^T}{\Delta g_k^T (\Delta x_k - H_k \Delta g_k)} \tag{5-9}$$

(5-9)式称为对称秩1公式.由对称秩1公式确定的变尺度算法称为对称秩1变尺度算法.此算法是简单的,但也存在明显的缺点:

(1) 当H_k正定时,由(5-9)式确定的H_{k+1}不一定是正定的,因此不能保证$s^k = -H_k g_k$是下降方向.

(2) (5-9)式的分母可能为零或近似为零,前者将使算法失效,后者将引起计算不稳定.

3.5.3 DFP算法

DFP算法[1] [2]是最先被研究的一种变尺度算法,也是目前仍在广泛使用的一种算法,它是一种秩2对称算法, 一般的秩2对称算法的修正矩阵公式可以写成

$$C_k = \alpha_k u u^T + \beta_k v v^T \tag{5-10}$$

其中u, v为待定的n维向量,α_k, β_k为待定常数,将上式代人(5-2)式, 再代入(5-7)式可得

$$\alpha_k u(u^T \Delta g_k) + \beta_k v(v^T \Delta g_k) = \Delta x_k - H_k \Delta g_k$$

[1] W.C. Davidon, *Variable metric method for minimization*, A.E.C. Research and Development Rept. ANL-5990(1959)

[2] R. Fletcher and M.J.D. Powell, *A Rapidly Convergent Descent Method for Minimization*, The Computer Journal(1963), 6(2):163-168

满足上式的α_k, β_k, u, v有无数多种取法,比较简单的一种取法

$$\begin{cases} u = H_k \Delta g_k \\ v = \Delta x_k \\ \alpha_k = -\dfrac{1}{u^T \Delta g_k} = -\dfrac{1}{\Delta g_k^T H_k \Delta g_k} \\ \beta_k = \dfrac{1}{v^T \Delta g_k} = \dfrac{1}{\Delta x_k^T \Delta g_k} \end{cases}$$

因此,

$$H_{k+1} = H_k + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta g_k (H_k \Delta g_k)^T}{\Delta g_k^T H_k \Delta g_k} \tag{5-11}$$

这就是DFP变尺度算法的计算公式.

变尺度算法的计算步骤

(1) 给定初始点x^0,计算精度$\varepsilon > 0$和初始矩阵$H_0 = I$(单位阵),令$k = 0$;

(2) 计算$s^k = -H_k g_k$,沿s^k进行精确一维搜索,求出步长λ_k,使

$$f(x^k + \lambda_k s^k) = \min_{\lambda \geq 0} f(x^k + \lambda s^k).$$

令$x^{k+1} = x^k + \lambda_k s^k$.

(3) 若$\|g_{k+1}\| < \varepsilon$,则取$x^* = x^{k+1}$,计算结束;否则由公式计算$H_{k+1}$,令$k = k + 1$,返回(2).

算法应用举例 I

例5.1

求函数$f(x) = x_1^2 + 2x_2^2 - 4x_1 - 2x_1x_2$的极小点.

解

1. $\frac{\partial f}{\partial x_1} = 2x_1 - 4 - 2x_2, \frac{\partial f}{\partial x_2} = 4x_2 - 2x_1$.取初始点$x^0 = (1,1)^T$.令$k = 0$.
2. $$s^0 = -H_0\nabla f(x^0) = -g_0 = (4,-2)^T$$
$$x^0 + \lambda s^0 = (1+4\lambda, 1-2\lambda)^T$$
$$f(x^0 + \lambda s^0) = 40\lambda^2 - 20\lambda - 3$$

因为$\frac{df}{d\lambda} = 80\lambda - 20$,得$\lambda_0 = 0.25$,所以$x^1 = x^0 + \lambda_0 s^0 = (2, 0.5)^T, g_1 = (-1,-2)^T$.

算法应用举例 II

3. 因为$\Delta x_0 = x^1 - x^0 = (1, -0.5)^T, \Delta g_0 = (3, -4)^T$, 所以

$$H_1 = H_0 + \frac{\Delta x_0 (\Delta x_0)^T}{(\Delta x_0)^T \Delta g_0} - \frac{(H_0 \Delta g_0)(H_0 \Delta g_0)^T}{\Delta g_0^T H_0 \Delta g_0} = \begin{pmatrix} 21/25 & 19/50 \\ 19/50 & 41/100 \end{pmatrix}$$

因为

$$\begin{aligned} s^1 &= -H_1 g_1 = (8/5, 6/5)^T \\ x^1 + \lambda s^1 &= (2 + 8\lambda/5, 0.5 + 6\lambda/5)^T \\ f(x^1 + \lambda s^1) &= 8\lambda^2/5 - 4\lambda - 5.5 \\ \frac{df}{d\lambda} &= 16\lambda/5 - 4 \end{aligned}$$

所以$\lambda_1 = 5/4$,则$x^2 = x^1 + \lambda_1 s^1 = (4, 2)^T$,这时$\nabla f(x^2) = (0, 0)^T$,因此$x^2$即为极小点.

DFP算法的性质和优缺点

首先我们指出:若H_k为对称正定矩阵,且$g_k \neq 0$,则DFP公式(5-11)后两项的分母均不为零,因此DFP公式(5-11)总是有意义的.
DFP算法的优点是:

(1) 若目标函数$f(x)$为n元二次严格凸函数,当初始矩阵取$H_0 = I$(单位阵)时, 由DFP法产生的搜索方向$s^0, s^1, \cdots, s^k$是关于$G = \nabla^2 f(x)$的共轭方向.因此, 最多经过n步迭代,就能达到最小点.即算法具有二次收敛性.

(2) 如果$f(x) \in C^1$为严格凸函数,则DFP算法是全局收敛的.

(3) 若H_k为对称正定矩阵,且$g_k \neq 0$,则由公式(5-11)确定的H_{k+1}也是对称正定的.

DFP算法的缺点是:

(1) 需要的存贮量较大,大约需要$O(n^2)$个存贮单元,因此对于大型问题,不如使用共轭梯度法方便.

(2) 数值计算的稳定性不如后面将要介绍的BFGS算法.

(3) 在使用不精确一维搜索时,它的计算效果不如BFGS算法的效果好.

3.5.4 吴桂变尺度算法类

在吴桂算法中,不要求H_k是对称的,并引入一个数量参数ρ_k, 将拟牛顿方程推广为更一般的形式

$$H_{k+1}\Delta g_k = \rho_k \Delta x_k. \tag{5-12}$$

吴桂算法的迭代公式为

$$H_{k+1} = H_k + a_k \Delta g_k^T H_k + b_k \Delta x_k^T, \tag{5-15}$$

将(5-15)式代入(5-12)式得

$$a_k(\Delta g_k^T H_k \Delta g_k) + b_k(\Delta x_k^T \Delta g_k) = \rho_k \Delta x_k - H_k \Delta g_k \tag{5-16}$$

所以在H_{k+1}的计算公式(5-15)中,实质上仅含有$n+1$个独立参数(n维列向量a_k,数量ρ_k).

3.5.5 Huang类算法

Huang类算法的迭代公式:有纯量$\alpha_k, \beta_k, \gamma_k, \delta_k$使

$$H_{k+1} = H_k - \frac{H_k \Delta g_k (\alpha_k H_k^T \Delta g_k + \beta_k \Delta x_k)^T}{(\alpha_k H_k^T \Delta g_k + \beta_k \Delta x_k)^T \Delta g_k} + \rho_k \frac{\Delta x_k (\gamma_k H_k^T \Delta g_k + \delta_k \Delta x_k)^T}{(\gamma_k H_k^T \Delta g_k + \delta_k \Delta x_k)^T \Delta g_k} \tag{5-18}$$

这里H_{k+1}表面上含有五个参数$\alpha_k, \beta_k, \gamma_k, \delta_k$和$\rho_k$, 但实质上只含有三个独立的参数$\alpha_k/\beta_k, \gamma_k/\delta_k$和$\rho_k$.

3.5.6 自调节变尺度算法类

在假设$\Delta g_k^T H_k \Delta g_k \neq 0$之下,有纯量$\alpha_k$,使

$$H_{k+1} = H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k}{\Delta g_k^T H_k \Delta g_k} + \rho_k \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} + \alpha_k (\Delta g_k^T H_k \Delta g_k) v_k v_k^T, \tag{5-20}$$

其中

$$v_k = \frac{\Delta x_k}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta g_k}{\Delta g_k^T H_k \Delta g_k}. \tag{5-21}$$

这就是Oren-Luenberger于1974年提出的自调节变尺度算法类,它含有两个参数α_k和ρ_k.

若$\rho_k = 1$,就得到Fletcher-Broyden算法类

$$H_{k+1} = H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k}{\Delta g_k^T H_k \Delta g_k} + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} + \alpha_k (\Delta g_k^T H_k \Delta g_k) v_k v_k^T, \quad (5\text{-}22)$$

它包含一个参数α_k.

- 当$\alpha_k = 0$时,就得到前面讲过的DFP变尺度算法的计算公式.
- 当$\alpha_k = 1$时,就得到熟知的BFGS变尺度算法的计算公式.

3.5.6 BFGS变尺度算法

BFGS变尺度算法的计算公式

$$H_{k+1} = H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k}{\Delta g_k^T H_k \Delta g_k} + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} + (\Delta g_k^T H_k \Delta g_k) v_k v_k^T, \quad (5\text{-}23)$$

或写成

$$H_{k+1} = H_k + \frac{\mu_k \Delta x_k \Delta x_k^T - H_k \Delta g_k \Delta x_k^T - \Delta x_k \Delta g_k^T H_k}{\Delta x_k^T \Delta g_k}, \quad (5\text{-}23')$$

其中

$$\mu_k = 1 + \frac{\Delta g_k^T H_k \Delta g_k}{\Delta x_k^T \Delta g_k}. \quad (5\text{-}24)$$

- BFGS算法是由Broyden,Fletcher,Goldfarb和Shanno等人在1970年给出的.
- 它与DFP算法都属于Broyden算法类. 因此,它与DFP算法具有类似的性质,也具有前面讲过的DFP算法的那些优点,但它的数值稳定性要比DFP算法好,而且在使用不精确一维搜索时,也能证明它是超线性收敛的. 因此BFGS算法被认为是目前最好的一种算法,得到了极为广泛的应用.
- 拟牛顿法的缺点之一:需要存贮很多的矩阵H_k.有限存贮的BFGS方法(L-BFGS)克服了这一缺点.

3.6 直接搜索法

上面所讲的几种方法,都要利用目标函数的一阶或二阶偏导数, 但在实际问题中,所遇到的目标函数往往比较复杂, 有的甚至难于写出其明显的解析表达式,因此,它们的导数很难求得,甚至根本无法求得. 这时,就必须采用不用导数的方法,即求多变量函数极值的直接搜索法,这类方法的特点是方法简单, 适用范围较广,但由于没有利用函数的分析性质,其收敛速度一般是比较慢的.

3.6.1 Hooke-Jeeves方法

这是一种简单而且容易实现的实用算法，由Hooke和Jeeves在1961年提出[a].这种方法由两类“移动”构成:一类称为探测搜索,其目的是探求下降的有利方向;另一类称为模式搜索,其目的是沿着有利方向进行加速.所以,也称为步长加速法或模式搜索法. 设要求解的问题为

$$\min f(x), \quad x \in \mathbf{R}^n.$$

设初始点和初始步长分别为x^1和d,$e_1, e_2, \cdots, e_n$为坐标向量,加速因子和计算精度分别为$\alpha > 0$和$\varepsilon > 0$.

---

[a] R. Hooke and T.A. Jeeves, *Direct Search Solution of Numerical and Statistical Problems*, Journal of the ACM, 8(2):212-229, 1961

Hooke-Jeeves方法的计算步骤

1. 令$y^1 = x^1, k = j = 1$;
2. 若$f(y^j + de_j) < f(y^j)$,则称为试验成功,令$y^{j+1} = y^j + de_j$,转3.;否则,若$f(y^j + de_j) \geq f(y^j)$,称为试验失败,此时,若$f(y^j - de_j) < f(y^j)$,令$y^{j+1} = y^j - de_j$,转3.;
   若$f(y^j - de_j) \geq f(y^j)$,令$y^{j+1} = y^j$,转3.;
3. 若$j < n$,令$j = j + 1$,返回2.;否则$j = n$,若$f(y^{n+1}) \geq f(x^k)$,转5.;若$f(y^{n+1}) < f(x^k)$, 则转4.;
4. 令$x^{k+1} = y^{n+1}, y^1 = x^{k+1} + \alpha(x^{k+1} - x^k), k = k + 1$,再令$j = 1$,返回2.;
5. 若$d \leq \varepsilon$,则计算结束,取$x^* \approx x^k$;否则,令$d = d/2, x^{k+1} = x^k$, 再令$y^1 = x^k, k = k + 1, j = 1$,返回2.

3.6.3 Powell方法

Powell方法是由M.J.D. Powell于1964年[a]首先提出的一种直接搜索方法. 可以证明:在一定条件下,它是一种共轭方向法,因此,对n元正定二次函数,至多迭代n次, 即可求得其极小点.Powell方法被认为是直接搜索法中比较有效的一种方法.

---

[a]M.J.D. Powell,*An efficient method for finding the minimum of a function of several variables without calculating derivatives*,The Computer Journal(1964), 7(2):155-162

基本思想

从选定的初始点x^0出发,先依次沿每个坐标方向求函数$f(x)$的极小点x^n,然后沿方向$s = x^n - x^0$ 再求一次极小,得一个新点,仍记为x^0,考虑到方向s可能比坐标方向更好,为组成下一次迭代要用的n 个方向,我们丢掉一个坐标方向s_1,加进方向s,然后重复上述过程得点x^n, 又得到一个新方向$s = x^n - x^0$,再用它来代替一个坐标方向s_2,如此迭代n次,即可得到一组彼此H共轭的方向(H 为二次目标函数的海赛阵).

Powell方法的计算步骤 I

1. 给定初始点x^0,计算精度$\varepsilon > 0$,n个初始的线性无关的搜索方向(一般取为n个坐标轴方向)为$e_1, e_2, \cdots, e_n$.令$k = 0$,

$$s_j = e_{j+1},\ j = 0, 1, \cdots, n-1.$$

2. 进行一维搜索,决定λ_k,使得

$$f(x^k + \lambda_k s_k) = \min\ f(x^k + \lambda s_k),$$

令$x^{k+1} = x^k + \lambda_k s_k$,若$k < n$,令$k = k + 1$,转向2.;否则转向3.;

3. 若$\|x^n - x^0\| < \varepsilon$,计算结束,取$x^* \approx x^n$;否则求整数$j(0 \le j \le n-1)$,使

$$\Delta = f(x^j) - f(x^{j+1}) = \max_{1 \le i \le n-1} [f(x^i) - f(x^{i+1})].$$

Powell方法的计算步骤 II

4. 令$f_1 = f(x^0), f_2 = f(x^n), f_3 = f(2x^n - x^0)$,若$2\Delta < f_1 - 2f_2 + f_3$,则方向$s_0, s_1, \cdots, s_{n-1}$不变,令$x^0 = x^n, k = 0$,返回2.;否则令

$$s_n = \frac{x^n - x^0}{\|x^n - x^0\|},$$

$s_i = s_{i+1}, i = j, j+1, \cdots, n-1$,转向5.;

5. 求λ_n,使得

$$f(x^n + \lambda_n s_n) = \min f(x^n + \lambda s_n),$$

令$x^0 = x^n + \lambda_n s_n, k = 0$,返回2.

3.6.2 单纯形法(Nelder-Mead Method)

无约束极小化的单纯形法与线性规划的单纯形法不同, 它是在1962年由Spendley,Hext和Himsworth首先提出的,1965年被Nelder和Mead [3]所改进.

基本思想

单纯形法与前面讲述的直接法不同,它不是沿某一个方向进行搜索,而是对n维空间的$n+1$ 个点(它们构成一个单纯形的顶点或极点)上的函数值进行比较,丢掉其中最"坏"的点,代之以新点, 构成一个新的单纯形,这样来逐步逼近函数的极小点.

- 反射(Reflection)
- 压缩(Contraction)
- 扩张(Expansion)

[3] J.A. Nelder and R. Mead,*A Simplex Method for Function Minimization*,The Computer Journal(1965), 7(4):308-313