数值分析 (/tags/#数值分析)

共轭梯度法的简单分析

数值分析初步

Posted by Alkane on May 18, 2019

共轭梯度法及其浅显分析

背景

无处不在的线性方程组,昭示着一个优秀的算法能带来的巨大效益。对于简单的低阶方程组,只需要用普通的高斯消元法就能得到相当不错的结果,但是对于大型的方程组,常常与之相伴的就是稀疏性,如果还坚持使用高斯消元法,那么将带来的是空间上的巨大浪费。为此,迭代法求解方程组便应运而生。

共轭梯度法就是其中的佼佼者。

> *预警:数学基础不够的读者可以略去公式推导,只需要关注思想即可。*

由来

二次型的最小值

我们熟知的线性方程组,常常可以写成$Ax = b$的形式,实际上当A是实对称矩阵时,就是二次型$f(x) = \frac{1}{2}x^TAx - b^Tx + c$对$x$的导数为0时的表达式。

$$f(x) = \frac{1}{2}x^TAx - b^Tx + c \tag{1}$$

$$f'(x) = \frac{1}{2}A^Tx + \frac{1}{2}Ax - b$$

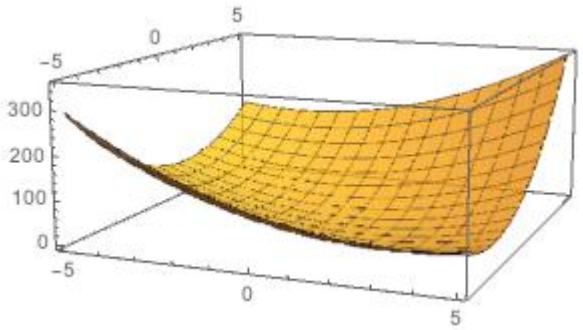
当A为实对称矩阵的时候,$f'(x)=0$就等价于$Ax=b$.

而我们求解线性方程组的问题,也可以转化成求解$minf(x)$

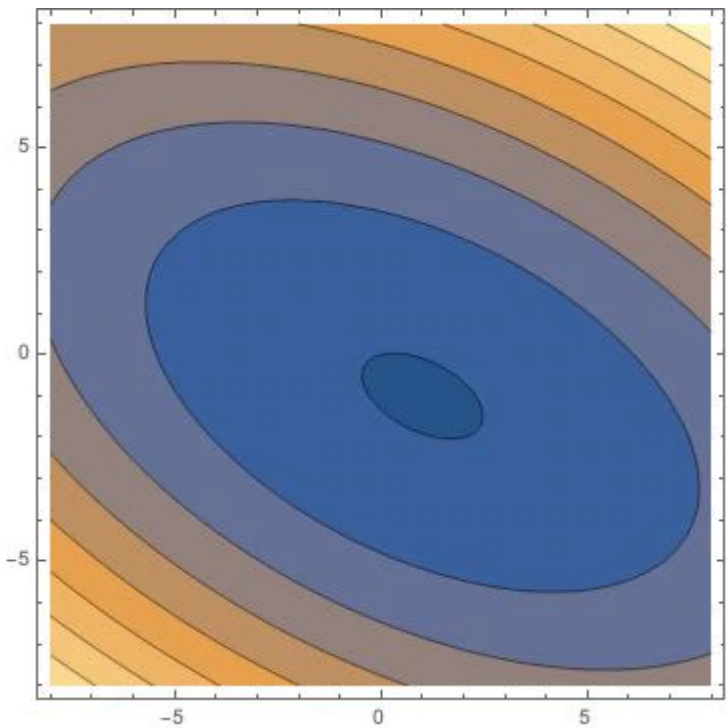
举个例子,当

$$A=\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix}, b=\begin{bmatrix}2\\ 8\end{bmatrix}, c=0 \tag{2}$$

\ 时,$f(x)$的图如下



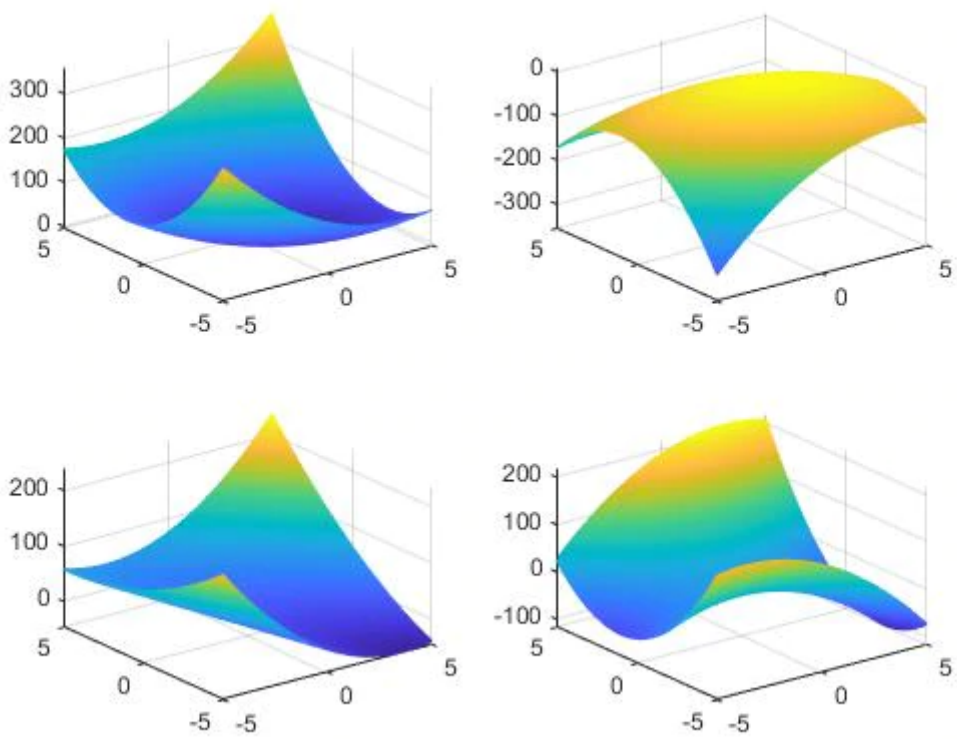
等高线图如下



看得出,此时的函数具有唯一的最小值。

由代数学的知识,我们知道,当矩阵A为正定、半正定、负定、以及不定的时候,方程组$Ax = b$具有不同的解的情况。对应$f`(x) = 0$则是不同的最小值情况。

下面这幅图说明了矩阵A的性质对于$f(x)$的影响，依次是正定、负定、半正定和不定的情况。



可以看到,当A不定的时候,$f(x)$具有鞍点,这个时候无法通过导数为0来求得最值。

在接下来的共轭梯度方法中,我们都首先假定,A具有良好的性质,也就是对称和正定。

最速下降法(梯度下降法)

如何得到$f(x)$的最小值?

有一个形象的方法,从任意一点$x_{(0)}$出发,每次沿着当前位置下滑最快的方向走,也就是该点处的梯度方向。

从而得到一系列的解序列:$x_{(1)}, x_{(2)}, \ldots$直到两次下降的差小于给定的误差限为止。

由上面的结论,$f'(x_{(i)}) = Ax_{(i)} - b$，下面记误差(error)为$e_{(i)} = x_{(i)} - x$，残差(residual)为$r_{(i)} = b - Ax_{(i)}$

于是有

$$\begin{aligned} r_{(i)} &= -f'(x_{(i)}) \\ x_{(i+1)} &= x_{(i)} + \alpha_{(i)} r_{(i)} \end{aligned} \tag{3}$$

其中$\alpha_{(i)}$为第i步沿着方向$r_{(i)}$的步长

如何选取步长$\alpha_{(i)}$呢？朴素的想法是选取的步长尽量要让$f(x_{i+1})$的值最小,这样才能更快的到达$f(x)$的全局最小值。

从简单的微积分知识中,我们把$f(x_{(i+1)})$看作是$\alpha_{(i)}$的函数,要使得步长最合适,也就相当于

$$\frac{d}{d\alpha_{(i)}} f(x_{(i+1)}) = 0 \tag{4}$$

根据链式法则

$$\begin{aligned} \frac{d}{d\alpha_{(i)}} f(x_{(i+1)}) &= f'(x_{(i+1)})^T \frac{d}{d\alpha_{(i)}} x_{(i+1)} \\ &= -r_{(i+1)}^T r_{(i)} \end{aligned}$$

也就是说,$r_{(i)}$与$r_{(i+1)}$正交

$$\begin{aligned} r_{(i+1)}^T r_{(i)} &= 0 \\ (b - Ax_{(i+1)})^T r_{(i)} &= 0 \\ (b - A(x_{(i)} + \alpha_{(i)} r_{(i)}))^T r_{(i)} &= 0 \\ (b - Ax_{(i)})^T r_{(i)} - \alpha_{(i)} (Ar_{(i)})^T r_{(i)} &= 0 \\ (b - Ax_{(i)})^T r_{(i)} &= \alpha_{(i)} (Ar_{(i)})^T r_{(i)} \\ \alpha_{(i)} &= \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}}. \end{aligned}$$

也就是说,每一步的步长都可以根据当前的残差$r_{(i)}$来确定

总结一下,最速下降法就是:

$$
\begin{aligned}
r_{(i)} &= b - Ax_{(i)} \\
\alpha_{(i)} &= \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}} \\
x_{(i+1)} &= x_{(i)} + \alpha_{(i)} r_{(i)}.
\end{aligned}
$$

这样迭代下去,直到达到事先给定的误差限ϵ为止

当然,有的时候,出于减小复杂度的考虑,我们会换用这种手段来求$r_{(i)}$

$$
r_{(i+1)} = r_{(i)} - \alpha_{(i)} A r_{(i)} \tag{5}
$$

这个公式是在$x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}$的两边左乘$-A$再加上$b$得到的。

这样每次迭代过程我们只需要算一次矩阵乘法,可以减少运算。

雅可比迭代法

在开始共轭梯度法前,我们还需要再来看看它的同行们都是怎么干活的。

对于方程$Ax = b$，我们把A分成两个矩阵D和E，其中D是对角阵，而E是剩下的部分，也就是说:

$$
\begin{aligned}
Ax &= b \\
(D + E)x &= b \\
Dx &= b - Ex \\
x &= D^{-1}(b - Ex).
\end{aligned}
$$

这里由于D是对角阵,所以能保证它的可逆性。

这里为了便于分析,把上面的式子写作

$$
x = Bx + z \tag{6}
$$

其中$B = -D^{-1}E, z = D^{-1}b$

下面考察该方法迭代过程的收敛性,依旧把误差(error)记作$e_{(i)}$

$$\begin{aligned} x_{(i+1)} &= Bx_{(i)} + z \\ &= B(x + e_{(i)}) + z \\ &= Bx + z + Be_{(i)} \\ &= x + Be_{(i)} \end{aligned}$$

故而

$$e_{(i+1)} = Be_{(i)} \tag{7}$$

这里判断迭代过程的敛散性主要看矩阵B的性质,如果B是实对称矩阵,那么$e_{(i)}$就可以用B的特征向量线性表示,迭代过程就可以写成:(下面以二阶矩阵为例)

$$\begin{aligned} e_{(i+1)} &= Be_{(i)} \\ &= B^i e_{(0)} \\ &= B^i(x_1 + x_2) \\ &= \lambda_1^i x_1 + \lambda_2^i x_2 \end{aligned}$$

其中λ_1 ，λ_2分别是矩阵B的特征向量x_1, x_2的特征值。

由此看得出,如果B的最大的特征值小于1,这个迭代过程就收敛,不然迭代过程会发散。

而一个矩阵的最大特征值也就是它的谱半径(spectral radius)$\rho(B)$

对于B不是实对称矩阵的情况,也有类似的结论,只是说明过于复杂,在此不提。

总之，雅可比迭代法收敛的充分条件是迭代矩阵B的谱半径$\rho(B) < 1$

下面还是以最初的方程举例子，来分析雅可比迭代的具体过程，这有助于改进迭代过程，得到更好的算法

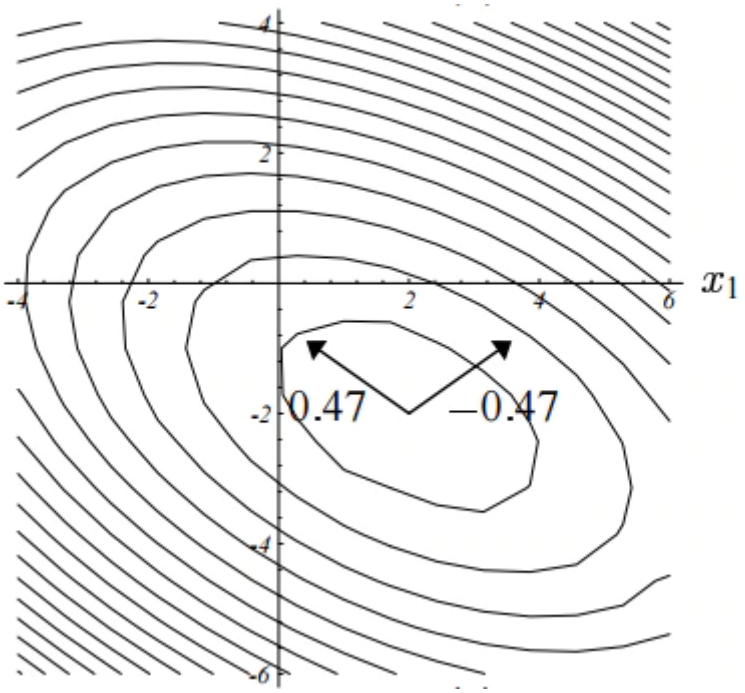
迭代过程现在是

$$x_{(i+1)} = \begin{bmatrix} 0 & -\frac{2}{3} \\ -\frac{1}{3} & 0 \end{bmatrix} x_{(i)} + \begin{bmatrix} \frac{2}{3} \\ -\frac{4}{3} \end{bmatrix} \tag{8}$$

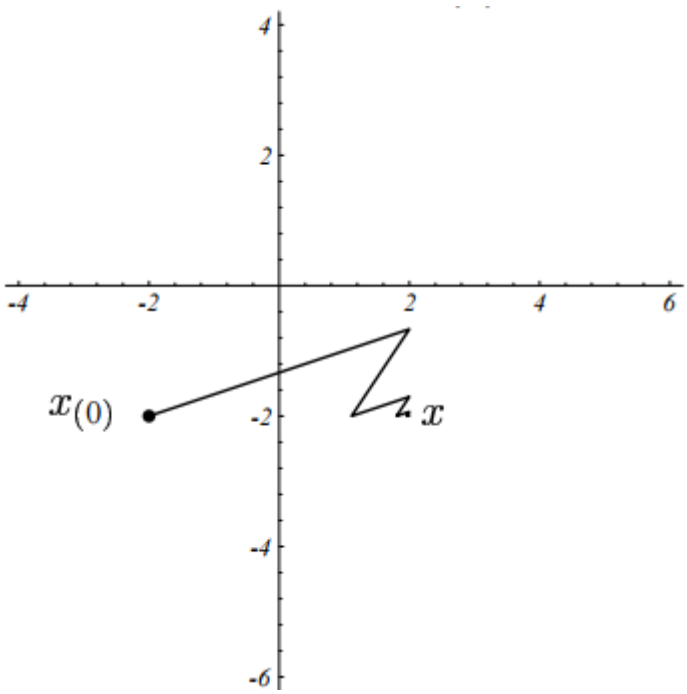
此时B的特征向量以及与其对应的特征值分别为

$$x_1 = \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix}, \lambda_1 = -\frac{\sqrt{2}}{3} \tag{9}$$
$$x_2 = \begin{bmatrix} -\sqrt{2} \\ 1 \end{bmatrix}, \lambda_1 = \frac{\sqrt{2}}{3}$$

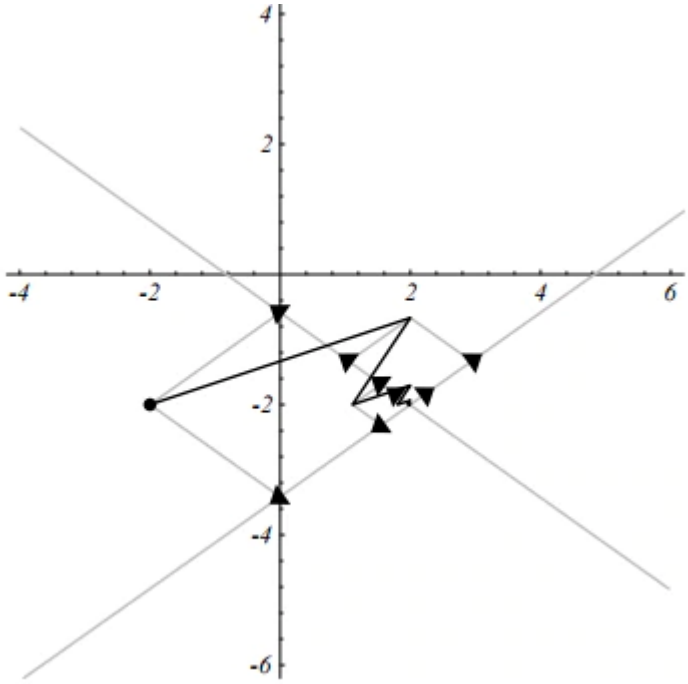
下图是迭代过程中,二次型函数$f(x)$的等高线图，注意两个箭头分别表示特征向量



而下图则是迭代过程



考虑每一次迭代与特征向量的关系



我们会发现,每一次的迭代都朝着两个特征向量的线性组合方向前进。

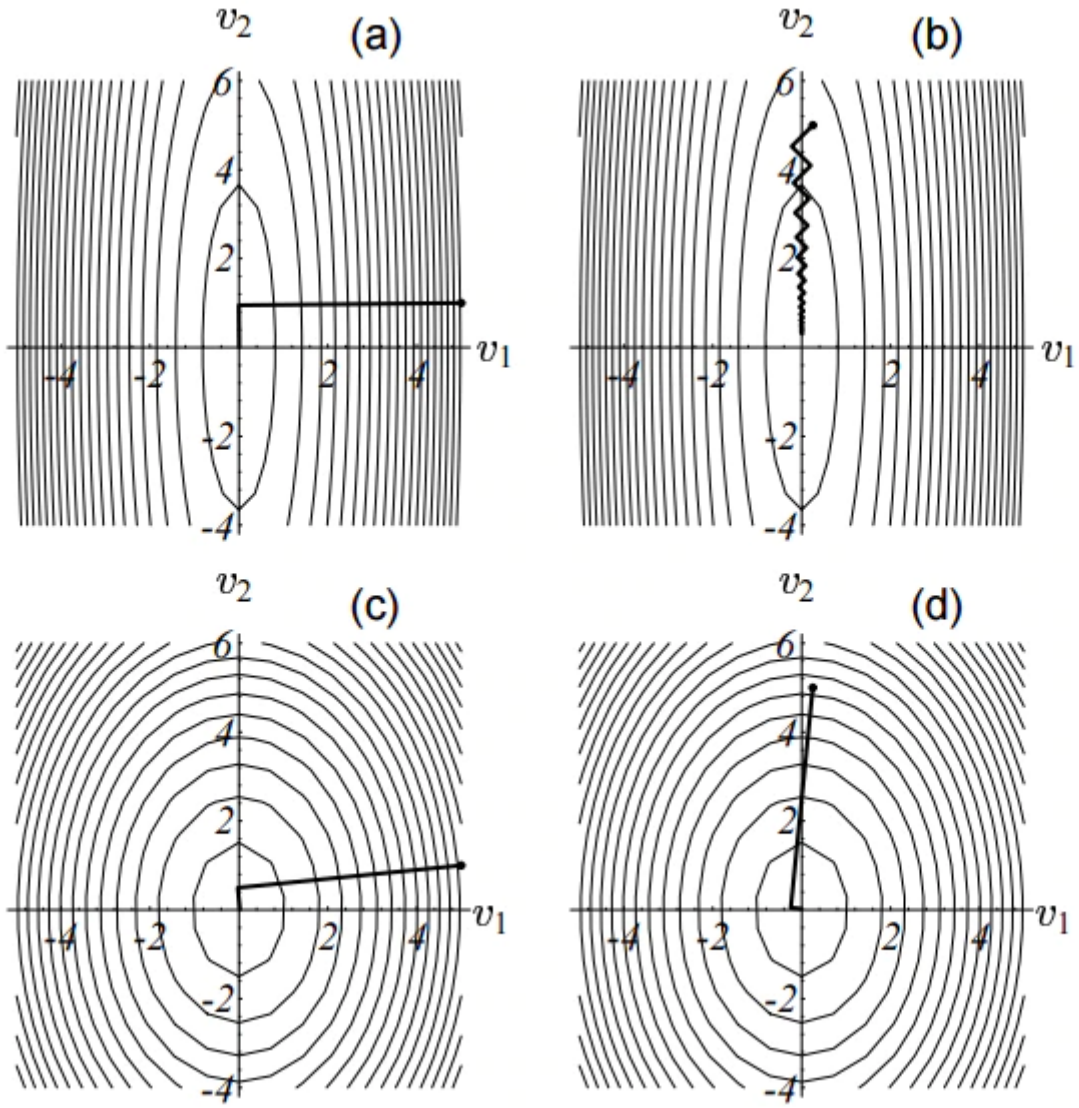
最速下降法的分析

前面我们得到了最速下降法的迭代过程，并且得到了一个十分有意思的结论

$$r_{(i+1)}^{T} r_{(i)} = 0 \tag{10}$$

这意味着,最速下降法每一次的迭代过程,下降的方向都与上一次的方向正交。

用具体例子来表述是这样的：



可以看到,(b)是最差的情况,需要蜿蜒前进很多次才接近解,而其他比较好的情况则两次迭代就收敛了.

但即便迭代的次数不同,每一次迭代的方向都与上一次正交，这就是最速下降法的特点。

在(b)中还有一个情况值得注意,由于这是一个二维的图形，故而如果每一次都和上一次正交,那么相邻一次迭代的方向是平行的!

换句话来说,在最速下降法中,有一个很糟糕的现象,为了收敛到解附近,同样的迭代方向可能走了不止一次。

这个现象不只是最速下降法中会出现，雅可比迭代法中也同样的出现了，表现就是每一次的方向都是特征向量的线性组合，而且大多数情况下,前一次迭代过的特征向量方向上的分量,在下一次的迭代中还继续存在。

共轭的想法

如果我能选取一系列线性无关的方向向量$d_{(0)}, d_{(1)}, d_{(2)}, \dots, d_{(n-1)}$,沿着每个方向只走一次，最后就能到达解$x$处,是不是能很好的解决前面雅可比迭代和最速下降法的问题?

最直接的想法来源于直角坐标系，如果每一个方向都是正交的，既自然，有具有很好的性质，是不是能得到很好的结果呢?

如果这样选取了方向,那么应该有误差$e_{(i+1)}$与方向$d_{(i)}$正交

$$\begin{aligned} d_{(i)}^T e_{(i+1)} &= 0 \\ d_{(i)}^T (e_{(i)} + \alpha_{(i)} d_{(i)}) &= 0 \end{aligned}$$

所以

$$\alpha_{(i)} = -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}} \tag{11}$$

但是这里步长大小依赖于当前的误差$e_{(i)}$，而不知道正确解x是无法求出误差来的,故而这种想法虽然很自然,但是无法得到有用的结果。

考虑到矩阵具有变换向量的作用,不妨记$x^T A y = 0$为向量x, y关于矩阵A共轭。

接下来选取的一系列方向向量都关于矩阵A两两共轭,$d_{(0)}, d_{(1)}, \dots, d_{(n-1)}$

我们从上一个想法得到灵感，要求这里的$e_{(i+1)}$也与$d_{(i)}$共轭，这是很自然的，只需要想象是对原本的空间做了一个变换,原本不正交的向量在这里正交,那么这里的要求就是合乎情理的了。

下面也一样，对步长大小进行简单的分析

$$\begin{aligned} \frac{d}{d\alpha_{(i)}} f(x_{(i+1)}) &= f'(x_{(i+1)})^T \frac{d}{d\alpha_{(i)}} x_{(i+1)} \\ f'(x_{(i+1)})^T d_{(i)} &= 0 \\ d_{(i)}^T r_{(i+1)} &= 0 \\ d_{(i+1)}^T A (e_{(i)} + \alpha_{(i)} d_{(i)}) &= 0 \end{aligned}$$

所以得

$$\alpha_{(i)} = \frac{d_i^T r_{(i)}}{d_{(i)}^T A d_{(i)}} \tag{12}$$

这里得到的步长不仅可以算,而且可以证明,这样迭代,对于n阶的方程组,至多n步就可以收敛到正确解。

简单的证明

首先，把误差$e_{(0)}$表示为方向向量的线性组合$e_{(0)} = \sum_{j=0}^{n-1} \delta_j d_{(j)}$

两侧左乘$d_{(k)}^T A$ 有:

$$\begin{aligned} d_{(k)}^T A e_0 &= \sum_{j=0}^{n-1} \delta_j d_{(k)}^T A d_{(j)} \\ &= \delta_k d_{(k)}^T A d_{(k)} \end{aligned}$$

上式利用了之前提到的共轭性质,由此得

$$\delta_k = \frac{d_{(k)}^T A e_{(k)}}{d_{(k)}^T A d_{(k)}} \tag{13}$$

与上面的步长比较，有

$$\alpha_{(k)} = -\delta_k \tag{14}$$

于是对于第i步迭代的误差$e_{(i)}$

$$\begin{aligned} e_{(i)} &= e_{(0)} + \sum_{j=0}^{i-1} \alpha_{(i)} d_{(j)} \\ &= \sum_{j=0}^{n-1} \delta_{(j)} d_{(j)} - \sum_{j=0}^{i-1} \delta_{(j)} d_{(j)} \\ &= \sum_{j=i}^{n-1} \delta_{(j)} d_{(j)} \end{aligned}$$

由此可知,当$i = n$的时候,$e_{(n)} = 0$

$Q.E.D$

并且我们能从上面的证明中得到这样一个事实,第i迭代时,有关前$i-1$次迭代的方向上都已经没有误差了,也就是这样的方法,每次都完全消除某个方向上的误差，这样至多n步得到精确解就是很自然的了。

共轭梯度法

上面的由来中，已经充分做好了准备来得到共轭梯度法，现在只剩一个问题，那一组两两共轭的方向向量该如何选取?

选取了一组基后，只需要进行Gram-Schmidt正交化就可以得到一组正交基，那么同样的也可以用这样的方法来得到一组共轭的方向向量。

问题是，如果随便选取一组线性无关的方向基，就进行Gram-Schmidt共轭化是否足够好呢?

我们总希望这组方向基不仅容易得到，而且具有很好的性质，减少计算量。

考虑到之前得到的结论，每一次迭代之间的残差都是相互正交的，我们不妨就选残差

$$\{r_{(0)}, r_{(1)}, r_{(2)}, \dots, r_{(n-1)}\} \tag{15}$$

作为共轭化之前的基。由于使用共轭化的方向向量来迭代至多只有n步，且每步都将该方向上的误差消灭掉，故而这一组基不仅线性无关，而且由于它们是残差，还具有正交的良好性质。

推导

首先，对于第i+1次迭代后的残差$r_{(i+1)}$有

$$\begin{aligned} r_{(i+1)} &= -Ae_{(i+1)} \\ &= -A(e_{(i)} + \alpha_{(i)}d_{(i)}) \\ &= r_{(i)} - \alpha_{(i)}Ad_{(i)}. \end{aligned}$$

接着我们来推导一下Gram-Schmidt共轭化的公式

第i次的方向向量为$d_{(i)}$，我们希望$d_{(i)}$从$r_{(i)} + d_{(0)}, d_{(1)}, d_{(2)}, \dots, d_{(i-1)}$中得到

,也就是

$$d_{(i)} = r_{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}, (i > k) \tag{16}$$

在上式的两侧同左乘上$d_{(i)}^T A$有

$$\begin{aligned} d_{(i)}^T A d_{(j)} &= r_{(i)}^T A d_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}^T A d_{(j)} \\ 0 &= r_{(i)}^T A d_{(j)} + \beta_{ij} d_{(j)}^T A d_{(j)}, \quad i > j \\ \beta_{ij} &= -\frac{r_{(i)}^T A d_{(j)}}{d_{(j)}^T A d_{(j)}} \end{aligned}$$

下面来结合残差$r_{(j+1)} = r_{(j)} - \alpha_{(j)} d_{(j)}$,继续做一些推演

在两侧同左乘$r_{(i)}^T$

$$\begin{aligned} r_{(i)}^T r_{(j+1)} &= r_{(i)}^T r_{(j)} - \alpha_{(j)} r_{(i)}^T A d_{(j)} \\ \alpha_{(j)} r_{(i)}^T A d_{(j)} &= r_{(i)}^T r_{(j)} - r_{(i)}^T r_{(i)} \end{aligned}$$

故而

$$r_{(i)}^T A d_{(j)} = \begin{cases} \dfrac{1}{\alpha_{(i)}} r_{(i)}^T r_{(i)}, & i = j \\ -\dfrac{1}{\alpha_{(i-1)}} r_{(i)}^T r_{(i)}, & i = j+1 \\ 0, & otherwise \end{cases}$$

结合β_{ij}的值，有

$$\beta_{ij} = \begin{cases} \dfrac{1}{\alpha_{i-1}} \dfrac{r_{(i)}^T r_{(i)}}{d_{(i-1)}^T A d_{(i-1)}}, & i = j+1 \\ 0, & i > j \end{cases}$$

考虑到完全可以把β_{ij}写成β_i，因为j只能取固定的值。

神奇的事情发生了,原本需要i个β才能确定的方向向量,在共轭化的条件下,只需要当前的数据和前一步的数据就可以得到,而不必存储之前所有走过的路径信息。

结合前面得到的$\alpha_(i) = \frac{d_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$

$$\beta_i = \frac{r_{(i)}^T r_{(i)}}{d_{(i-1)}^T r_{(i-1)}} \tag{17}$$

实际上，上面的式子还能够写成更加漂亮的样子

先由$d_{(i)} = r_{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}, (i > k)$

做其与$r_{(j)}$的内积有

$$\begin{aligned} d_{(i)}^T r_{(j)} &= r_{(i)}^T r_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}^T r_{(j)} \\ 0 &= r_{(i)}^T r_{(j)}, \quad i < j \end{aligned}$$

令$j = i$有

$$d_{(i)}^T r_{(i)} = r_{(i)}^T r_{(i)} \tag{18}$$

带入β_i中有

$$\beta_i = \frac{r_{(i)}^T r_{(i)}}{r_{(i-1)}^T r_{(i-1)}} \tag{19}$$

这样，Gram-Schmidt共轭化就完美的实现了，不仅实现了每一个方向只迭代一次，而且需要存储的数据只有上一步的残差。

下面是共轭梯度法涉及到的所有公式

$$d_{(0)} = r_{(0)} = b - Ax_{(0)}$$
$$\alpha_{(i)} = \frac{r_i^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$$
$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$$
$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)}$$
$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$$
$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}$$

最后给出共轭梯度法的Matlab实现

```
function x=Conjugate_gradient(A,b,x0)
x=x0;
d=b-A*x0;
r=d;
for k=0:size(A)-1
    if r==0
        break;
    end
    alpha=(r'*r)/(d'*A*d);
    x=x+alpha*d;
    temp=r'*r;
    r=r-alpha*A*d;
    belta=(r'*r)/(temp);
    d=r+belta*d;
end
end
```

分析

正交性是共轭梯度法成功的关键,注意到每一次迭代中，新的残差$r_{(i+1)}$与前面所有的$r_{(i)}$正交，如果一个$r_{(i)} = 0$那么$Ax_{(i)} = b$方程已经解完了。

否则在n步迭代后，$r_{(n)}$和n个两两正交的向量$r_{(0)}, r_{(1)}, \ldots, r_{(n-1)}$所张成的空间正交,而$r_{(0)}, r_{(1)}, \ldots, r_{(n-1)}$这n个向量是$R^n$空间中的所有正交向量,由此$r_{(n)}$必须是零向量,所以$Ax_{(n)} = b$

比较

共轭梯度法从某种程度上要简单于高斯消元法,不必考虑行和列的相消,而且代码实现也十分简洁。下面来比较一下共轭梯度法和高斯消元法在复杂度上的优势。

共轭梯度法的每一次迭代都要做一次矩阵向量乘法和一些向量内积的计算，复杂度为n^2，当做完n次迭代后，复杂度变成了n^3，而高斯消元法只是$\frac{1}{3}n^3$左右，从这一点上，当矩阵不是稀疏矩阵的时候，共轭梯度法没有优势，而当矩阵变得稀疏的时候，n大的惊人，高斯消元法如果要得到解，就必须要做完所有的运算才能得到解，这样需要消耗大量的资源。而共轭梯度法不一样，它每一次迭代都能在某个分量上得到解，并且可以通过残差来度量解的精确情况，我们并不需要将算法进行到底，只需要解达到精度要求就可以退出。

同样的，与其他迭代法相比，共轭梯度法又能在确定的步数内收敛，这就是共轭梯度法的优势。

但是，当矩阵变成病态矩阵的时候，由于每一步的误差的累计，很有可能导致方向向量出现偏差，从而出现很糟糕的结果，这也是共轭梯度法的一个缺陷，我们可以通过预条件处理来减小病态矩阵带来的误差。

预条件处理

预条件处理的思想是降低方程系数矩阵的条件数，方法是左乘一个矩阵。

即

$$M^{-1}Ax = M^{-1}b \tag{20}$$

其中M是可逆的n阶矩阵，称为**预条件子**

常用的预条件子有：

- 雅可比预条件子：$M = D$,D是A的对角矩阵
- 高斯-塞尔德预条件子：$M = (D + \omega L)D^{-1}(D + \omega U)$，其中$A = L + D + U$,分别是下三角、对角、上三角矩阵，$\omega$是介于0和2之间的常数

PREVIOUS
随笔
(/2019/05/16/%E6%83%B3%E6%B3%95/)

NEXT
深度学习笔记【0】
(/2019/07/13/%E6%B7%B1%E5%BA%A6%E5%0/)

Leave a comment

ⓘ Markdown is supported (https://guides.github.com/features/mastering-markdown/)

Login with GitHub Preview

HarRRyCHang (https://github.com/HarRRyCHang) commented 5 months ago

great!thx!

pickerxxr (https://github.com/pickerxxr) commented 4 months ago

写的很好

leesusu (https://github.com/leesusu) commented 11 days ago

公式(16)下面一句，应该是：
在上式的两侧同右乘上Ad_(j)

下面来结合残差r(j+1)=r(j)−α(j)d(j),继续做一些推演
应该是：
下面来结合残差r(j+1)=r(j)−α(j)Ad(j),继续做一些推演
少了个A

FEATURED TAGS (/tags/)

想法 (/tags/#想法) 数值分析 (/tags/#数值分析) 神经网络 (/tags/#神经网络) 微积分 (/tags/#微积分) 人工智能 (/tags/#人工智能)

FRIENDS

浩源 (https://www.jianshu.com/u/484503532487) Apple (https://apple.com)
Apple Developer (https://developer.apple.com/) ()

● (https://www.zhihu.com/people/alkane0050)

● (https://github.com/Alkane0050)