数值优化（Numerical Optimization）学习系列-共轭梯度方法（Conjugate Gradient）

下一步 于 2015-12-27 18:46:50 发布

概述

> 共轭梯度算法在最优化问题中备受关注，有两层用途，一是可以求解线性方程$Ax=b$；二是可以求解最优化问题。相对于最速下降法，它没有额外的矩阵存储并且比更快，一般N步内收敛。实际收敛效率依赖于系数矩阵特征值的分布。

主要介绍一下内容：
1. 线性共轭梯度算法
2. 共轭方向算法
3. 共轭梯度算法
4. 收敛性
5. 非线性共轭梯度算法

线性共轭梯度算法

共轭梯度算法是一个求解线性方程的迭代方法

问题形式

CG算法求解问题的两种形式：
1. 线性方程$Ax=b$并且要求A是**对称正定矩阵。**
2. 最优化问题：

$$min\phi(x)=\frac{1}{2}x^TAx-b^Tx$$

，要求A对称正定，这样该问题是一个凸问题并且有最优解，根据最优解满足 $\nabla\phi(x)=Ax-b=0$，一般即为$r(x)$。在迭代过程中第K步的残差为$r_k=Ax_k-b$

共轭性

给定一个非零向量集合 $\{p_0, p_1, p_2 ... p_{n-1}\}$和一个对称正定矩阵A；如果**向量集合相对于A是共轭的**当且仅当 $p_i^T A p_j = 0,\ i \neq j$

> 如果向量集合是共轭的，则他们之间是相互线性独立。

证明：反证法。如果不是线性独立，则有 $p_i = \lambda p_j$ ，则$p_i^T A p_j = \lambda p_j^T A p_j \neq 0$，不满足共轭条件。

共轭方向算法

共轭方向算法（Conjugate Direction Method）不同于共轭梯度方法，共轭向量提前给出。

算法描述

1. 给定共轭方向集合$\{p_0, p_1, p_2 ... p_{n-1}\}$和任意初始点$x_0$
2. 计算$x_{k+1} = x_k + \alpha_k p_k$
3. 计算最优步长α，即优化$\phi(x_k + \alpha p_k)$，通过求解单变量最优化问题，可以容易得到最优步长

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$

关于α_k的计算

$$\begin{aligned}\alpha_k &= arg\ min(\phi(x_k + \alpha p_k)) \\ &= argmin \frac{1}{2}(x_k + \alpha p_k)^T A (x_k + \alpha p_k) - b^T(x_k + \alpha p_k) \\ \nabla \phi(\alpha) = 0 &\rightarrow \alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}\end{aligned}$$

> 对于任何初始点x_0共轭方向算法最多N步可以收敛到最优解x^*

证明：思路通过上述算法可以表示出最优解。

1. 由于p_i线性独立，则在N维空间中可以生成整个空间，因此

$$x^* - x_0 = \delta_0 p_0 + \delta_1 p_1 + ... + \delta_{n-1} p_{n-1}$$

左边同时乘上 $p_k^T A$得到

$$p_k^T A(x^* - x_0) = p_k^T A(\delta_0 p_0 + \delta_1 p_1 + \ldots + \delta_{n-1} p_{n-1}) = \delta_k p_k^T A p_k$$

，可以推出

$$\delta_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$$

2. 根据共轭方向算法可以得到

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{k-1} p_{k-1}$$

，同样两边同时乘上 $p_k^T A$可以得到

$$p_k^T A(x_k - x_0) = 0$$

即 $p_k^T A x_k = p_k^T A x_0$

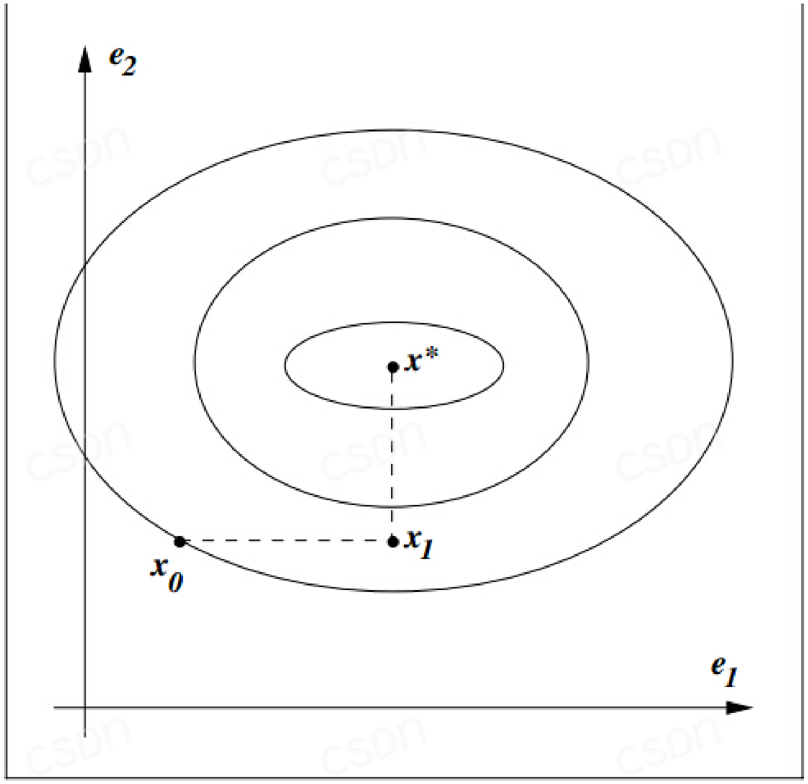
3. 因此有

$$\begin{aligned} p_k^T A(x^* - x_0) &= p_k^T A(x^* - x_k) \\ &= p_k^T (Ax^* - Ax_k) \\ &= p_k^T (b - Ax_k) \\ &= -p_k^T r_k \\ &= \delta_k p_k^T A p_k \end{aligned}$$

4. 可见 $\alpha_k = \delta_k$，从而任意起点都可以再N步内得到最优解。

算法理解

如果假设矩阵A是对角阵，则我们可以沿着坐标轴方向依次优化各个方向的值。坐标轴方向为$e_0, e_1 ... e_{n-1}$。二维情况如下：



但是如果A是非对角阵，如果还沿着坐标轴方向进行优化，有可能不会收敛。
但是可以通过转换的方式将A变成对角阵。
定义$\hat{x} = S^{-1}x$，其中S的定义为

$$S = [p_0, p_1 ... p_{n-1}]$$

则原问题可以表示为:

$$\hat{\phi}(\hat{x}) = \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)x - (S^Tb)^T\hat{x}$$

由于共轭性，则有 (S^TAS)是对角阵，可以按照坐标轴方向进行优化。其中的关键点是

> 在$\hat{x}$空间内的e_i方向等价于在x空间内的p_i方向，从而进一步验证该算法的正确性。

算法性质

从上述算法理解中可以看出，在计算到e_k方向时，此时的x_k是扩展子空间$e_1 ... e_k$中的最优解。类比于共轭方向算法有

> 对于任意初始值x_0，解序列$x_0, x_1 ... x_k$由共轭方向算法生成，则有以下性质
> 1. $r_k^T p_i = 0 \quad i = 0, 1, \ldots, k-1$
> 2. x_k在以下集合中是$\phi(x)$的最优解，$\{x | x = x_0 + span(p_0, p_1 ... p_{k-1})\}$

在证明之间简单理解一下，第K步的残差r_k和前面K-1个共轭方向是正交的。
证明：用归纳法可以得到。
1. 当k=1时，需要证明$r_1^T p_0 = 0$，由于$x_1 = x_0 + \alpha_0 p_0$，$\alpha_0 = -\frac{r_0^T p_0}{p_0^T A p_0}$，又

$$\begin{aligned} r_1^T p_0 &= (Ax_1 - b)^T p_0 \\ &= (Ax_0 + \alpha_0 A p_0 - b)^T p_0 \\ &= (r_0 + \alpha_0 A p_0)^T p_0 \\ &= r_0^T p_0 + \alpha_0 p_0^T A p_0 \\ &= 0 \end{aligned}$$

2. 假设k-1时也成立，即 $r_{k-1}^T p_i = 0;\ i = 0, 1, 2 ... k-2$
3. 证明k时也成立，由于

$$r_k = Ax_k - b = A(x_{k-1} + \alpha_{k-1} p_{k-1}) - b = r_{k-1} + \alpha_{k-1} A p_{k-1}$$

4. $p_{k-1}^T r_k = p_{k-1} r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0$，根据 α_{k-1}的定义可以得到
5. $p_i^T r_k = p_i r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0 \quad i = 0, 1 ... k-2$根据归纳假设可以得到。

共轭方向计算

1. 可以直接采用特征向量，但是特征向量计算复杂度比较高
2. 可以采用Gram_Schmidt求解正交分解的方式得到共轭方向，对于大规模数据计算复杂度也高。

共轭梯度算法

共轭梯度算法是共轭方向算法的一种，它提供了一种计算共轭方向集合的方法，并且当前方向的计算仅和上一步方向相关，从而减少矩阵存储。
当前共轭方向p_k选择为当前残差方向和上一步方向p_{k-1}的线性表示，即$p_k = -r_k + \beta_k p_{k-1}$。由于要满足$p_k A p_{k-1} = 0$，从而可以推出：

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}{}^T A p_{k-1}}$$

初始点 x_0可以随机选择，初始搜索方向 p_0选择为最速下降方向 $-r_0$

CG-Preliminary算法

1. 初始化x_0
2. $r_0 = Ax_0 - b, p_0 = r_0, k = 0$
3. while $r_k \neq 0$
    - $$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$
    - $$x_{k+1} = x_k + \alpha_k p_k$$
    - $$r_{k+1} = Ax_{k+1} - b$$
    - $$\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$
    - $$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$$
    - $$k = k + 1$$
4. end while

算法证明

现在需要解决的问题是上述算法的正确性？如何更好的优化？
证明正确性需要解决的问题是根据上述算法求解得到的p_k是否满足相对于A是共轭的。

> 定理：共轭算法产生的解序列$\{x_k\}$满足一下性质：
> 1. $r_k^T r_i = 0 \quad i = 0, 1, 2 ... k-1$
> 2. $p_k^T A p_i = 0 \quad i = 0, 1, 2 ... k-1$

即当前步骤产出的残差和之前所有残差正交；当前搜索方向和之前所有方向共轭。
证明：首先回顾一下在共轭方向算法中的一个性质

$$r_k^T p_i = 0 \quad i = 0, 1, 2 \ldots k-1$$

即当前步骤的残差方向和之前所有的搜索方向正交。
性质1证明：由于 $p_k = -r_k + \beta_k p_{k-1}$则有 $r_k = -p_k + \beta_k p_{k-1}$，两边同时乘上 r_i可以推出

$$r_i^T r_k = r_i^T(-p_k + \beta_k p_{k-1})$$

根据上述性质，显而易见结果为0.
性质2证明：可以采用归纳方法证明。
1. k=0时， $p_0 = -r_0$
2. k=1时， $p_1 = -r_1 + \beta_1 p_0$，根据定义肯定满足 $p_1^T A p_0 = 0$
3. k=2时， $p_2 = -r_2 + \beta_2 p_1$，根据定义 $p_2^T A p_1 = 0$，下面需要证明 $p_2^T A p_0 = 0$。

$$\begin{aligned}
p_2^T A p_0 &= (-r_2 + \beta_2 p_1)^T A p_0 \\
&= -r_2^T A p_0 + \beta_2 p_1 A p_0 \\
&= -r_2^T A p_0 + 0 \\
&= -(r_1 + \alpha_1 A p_1)^T A p_0 \\
&= -(r_0 + \alpha_0 A p_0 + \alpha_1 A p_1)^T A p_0 \\
&= \text{代入} \alpha_0 \ and \ \alpha_1 \\
&= 0
\end{aligned}$$

CG算法

下面正式介绍CG算法，根据上述性质可以对基础算法进行优化改进。
改进思路1：

$$\begin{aligned}
\alpha_k &= -\frac{r_k^T p_k}{p_k^T A p_k} \\
&= -\frac{r_k^T(-r_k + \beta_k p_{k-1})}{p_k^T A p_k} \\
&= \frac{r_k^T r_k}{p_k^T A p_k}
\end{aligned}$$

改进思路2：由于 $r_{k+1} = r_k + \alpha_k A p_k$可以得到 $A p_k = \frac{r_{k+1} - r_k}{\alpha_k}$

$$
\begin{aligned}
\beta_{k+1} &= \frac{r_{k+1}^T A p_k}{p_k^T A p_k} \\
&= \frac{r_{k+1}^T A p_k}{p_k^T A p_k} \\
&= \frac{r_{k+1}^T (r_{k+1} - r_k)}{\alpha_k p_k^T A p_k} \\
&= \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}
\end{aligned}
$$

最后一步变换根据 **改进思路1和上述性质1。**

因此最后的算法为：

1. 初始化 x_0
2. $r_0 = Ax_0 - b, p_0 = r_0, k = 0$
3. while $r_k \neq 0$

$$
\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}
$$

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

$$
r_{k+1} = Ax_{k+1} - b
$$

$$
\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}
$$

$$
p_{k+1} = -r_{k+1} + \beta_{k+1} p_k
$$

$$
k = k + 1
$$

4. end while

CG算法收敛性质

> 如果矩阵A有r个不同的特征值，则最多循环r步。
> 算法的收敛速度和A的特征值分布相关，如果A的条件数越大收敛越慢。

根据收敛性质，在实际应用中可以先对A进行预处理，从而使得A特征值分布更均匀一些。

非线性共轭梯度算法

将共轭的思想应用到一般化的最优化问题中，甚至非线性问题中去。

Fletcher-Reeves方法

Algorithm 5.4 (FR).

Given x_0;

Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;

Set $p_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$;

while $\nabla f_k \neq 0$

$\boxed{\text{Compute } \alpha_k \text{ and set } x_{k+1} = x_k + \alpha_k p_k;}$

Evaluate ∇f_{k+1};

$$\boxed{\beta_{k+1}^{\mathrm{FR}} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k};}$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{\mathrm{FR}} p_k;$$

$$k \leftarrow k+1;$$

end (while)

算法如下：

主要改变如上图红框所示

1. 计算步长α_k不一定能找到最优步长，可以采用wolf 条件进行计算。
2. 残差的计算可以直接用梯度代替。
3. 重点是**需要保证每一步的搜索方向都满足下降**，否则不保证收敛。选择强wolf条件可以保证上述约束

FR其他变种

主要思路优化β_k的计算。

总结

共轭梯度算法可以认为解决了一类特殊最优化问题:$min\phi(x)=\frac{1}{2}x^TAx-b^Tx$，并且矩阵A对称且正定。
解决思路和lineSearch比较相似，首先确定搜索方向然后计算步长。CG特别的地方在于搜索方向相对于A共轭，并且步长均选择为最优步长。其收敛速度线性。
通过本节内容需要了解

1. CG算法解决问题的形式
2. 共轭性
3. 共轭方向算法
4. 共轭梯度算法以及正确性证明
5. 收敛性