

拟牛顿类算法

宋晓良

大连理工大学数学科学学院

教材《最优化：建模、算法与理论》配套电子教案

<http://bicmr.pku.edu.cn/~wenzw/optbook.html>

- 1 介绍
- 2 割线方程
- 3 拟牛顿矩阵更新方式
- 4 拟牛顿类算法的收敛性和收敛速度
- 5 有限内存BFGS方法
- 6 应用举例

提出拟牛顿类(近似)算法的必要性

牛顿类算法是二阶算法,其理论完备,实践中经常被用于解决规模不大的优化问题,优势是其求解效率高.

但对于大规模问题,求出牛顿方向并非易事,原因如下:

首先,对于 $x \in \mathbb{R}^n$ 规模的问题,单值函数 $f(x)$ 的Hesse矩阵 $\nabla^2 f(x)$ 的规模有 $n \times n$ 之多,需要做 n^2 次偏导计算;

其次,即使获得了Hesse矩阵:

- (1)经典牛顿方法由 $d^k = -\nabla^2 f(x^k)^{-1} \nabla f(x^k)$ 得到牛顿方向;
- (2)阻尼方法需确定修正矩阵 E^k 以及求解方程组 $B^k d^k = -\nabla f(x^k)$ 才能得到修正牛顿方向;
- (3)非精确牛顿方法需要近似求解方程组 $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$ 才能得到非精确牛顿方向.

以上计算步骤都将在数据规模较大时产生困难.

对近似方法的基本要求

我们希望提出一类新的牛顿方法,它也是通过求解牛顿方向进行寻优的,但具有以下的特点:

- 不直接计算Hesse矩阵,而是通过合理近似的方式提出,目的在于避免或减少计算偏导数;
- 不直接求上述近似矩阵的逆,而是通过巧妙的代数结构计算逆,目的在于避免求逆以及方程组带来的数值困难;
- 上述近似过程的计算复杂度相比牛顿方法有较大的降低;
- 近似矩阵及其逆还保有Hesse矩阵的性质,即使得牛顿方向 d^k 仍为下降方向;
- 使用近似Hesse矩阵以及逆时要保证算法是收敛的,而且最好只比二阶收敛慢一点,即超线性收敛.

我们接下来对其基本的构造过程和思想进行阐述.

1 介绍

2 割线方程

3 拟牛顿矩阵更新方式

4 拟牛顿类算法的收敛性和收敛速度

5 有限内存BFGS方法

6 应用举例

割线方程的推导

回顾牛顿法的推导过程. 设 $f(x) \in \mathbb{R}, x \in \mathbb{R}^n$ 是二阶连续可微函数. 对 $\nabla f(x)$ 在点 x^{k+1} 处一阶Taylor近似, 得

$$\nabla f(x) = \nabla f(x^{k+1}) + \nabla^2 f(x^{k+1})(x - x^{k+1}) + \mathcal{O}(\|x - x^{k+1}\|^2),$$

令 $x = x^k$, 且 $s^k = x^{k+1} - x^k$ 为点差, $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ 为梯度差, 得

$$\nabla^2 f(x^{k+1})s^k + \mathcal{O}(\|s^k\|^2) = y^k.$$

这是Hesse矩阵满足的方程.

现忽略高阶项 $\|s^k\|^2$, 只希望近似Hesse矩阵的矩阵 B^{k+1} 满足方程

$$B^{k+1}s^k = y^k,$$

或其逆矩阵 H^{k+1} 满足

$$H^{k+1}y^k = s^k.$$

上述2个方程即称为割线方程.

割线方程的推导

由于上述割线方程是由Hesse矩阵满足的方程近似得来的, 因此我们可以期待 B^{k+1} 以及 H^{k+1} 具有一些Hesse矩阵的性质.

我们还可以从另一角度理解割线方程. 既然牛顿法是对目标函数 $f(x)$ 在迭代点 x^k 处做二阶近似再求解, 考虑函数 $f(x)$ 在点 x^{k+1} 处的二阶Taylor近似

$$m_{k+1}(d) = f(x^{k+1}) + \nabla f(x^{k+1})^T d + \frac{d^T B^{k+1} d}{2},$$

从迭代过程以及梯度的意义而言, 我们要求 $m_{k+1}(d)$ 在点 $d = -s^k, 0$ 处的梯度与 $f(x)$ 在点 $x = x^k, x^{k+1}$ 处的梯度分别相等.

针对第一个要求,

$$\nabla m_{k+1}(-s^k) = \nabla f(x^{k+1}) - B^{k+1} s^k = \nabla f(x^k),$$

整理即得割线方程.

至于第二个要求, 将 $d = 0$ 代入即得 $\nabla m_{k+1}(0) = \nabla f(x^{k+1})$, 是自然满足的.

曲率条件

由于近似矩阵必须保证迭代收敛, 正如牛顿法要求Hesse矩阵正定, B^k 正定也是必须的, 即有必要条件

$$(s^k)^T B^k s^k > 0 \implies (s^k)^T y^k > 0,$$

这一条件针对的是一切的 B^k , 因此要在迭代过程中始终满足.

定义

曲率条件 在迭代过程中满足 $(s^k)^T y^k > 0, \forall k \in \mathbb{N}^+$.

Wolfe准则实现曲率条件

为实现曲率条件, 在牛顿方向上执行进行线搜索(此时不要求步长恒为1)时, 我们可以使用Wolfe准则, 因为由Wolfe条件

$$\nabla f(x^k + \alpha d^k)^T d^k \geq c_2 \nabla f(x^k)^T d^k,$$

其中 $c_2 \in (0, 1)$. 在本情况下, 上式即

$$\nabla f(x^{k+1})^T s^k \geq c_2 \nabla f(x^k)^T s^k,$$

为了凑出 y^k , 在不等式两边同时减去 $\nabla f(x^k)^T s^k$, 得

$$\nabla(y^k)^T s^k \geq (c_2 - 1) \nabla f(x^k)^T s^k,$$

注意到 $c_2 - 1 < 0$ 且 s^k 是下降方向, 因此最终有

$$\nabla(y^k)^T s^k \geq (c_2 - 1) \nabla f(x^k)^T s^k > 0,$$

满足了曲率条件.

注意: 仅仅使用Armijo准则并不能保证曲率条件成立.

提纲

- 1 介绍
- 2 割线方程
- 3 拟牛顿矩阵更新方式
- 4 拟牛顿类算法的收敛性和收敛速度
- 5 有限内存BFGS方法
- 6 应用举例

修正矩阵的基本思路

类似于牛顿方法, 我们在 x^k 处根据 $f(x)$ 给出一个合理的下降方向, 然后执行Wolfe准则确定步长, 即可完成一步迭代.

根据割线方程, 要确定一个类似牛顿方向的下降方向, 就要确定每步迭代时的 B^k 或者 H^k . 我们通常将拟牛顿法产生的方向称为拟牛顿方向, 而将 B^k, H^k 称为拟牛顿矩阵.

设计拟牛顿矩阵是有要求的. 为尽可能减少算法在迭代时的计算量, 我们希望拟牛顿矩阵的更新简单, 即是通过修正迭代更新每步矩阵的.

基于上述的思路, 我们可以写出拟牛顿算法的基本框架.

拟牛顿算法的基本框架

拟牛顿算法的基本框架为：

算法 1 拟牛顿算法框架

Require: 初始坐标 $x^0 \in \mathbb{R}^n$, 初始矩阵 $B^0 \in \mathbb{R}^{n \times n}$ (或 H^0), $k = 0$.

Ensure: x^K, B^K (或 H^K).

- 1: 检查初始元素.
 - 2: **while** 未达到停机准则 **do**
 - 3: 计算方向 $d^k = -(B^k)^{-1} \nabla f(x^k)$ 或 $d^k = -H^k \nabla f(x^k)$.
 - 4: 通过线搜索(Wolfe)产生步长 $\alpha_k > 0$, 令 $x^{k+1} = x^k + \alpha_k d^k$.
 - 5: 更新Hesse矩阵的近似矩阵 B^{k+1} 或其逆矩阵 H^{k+1} .
 - 6: $k \leftarrow k + 1$.
 - 7: **end while**
-

秩一更新(SR1)

秩一更新是一种迭代式更新矩阵的手段, 它的结构非常简单. 我们可以用秩一更新的方式更新拟牛顿矩阵.

定义

秩一更新 对于拟牛顿矩阵 $B^k \in \mathbb{R}^{n \times n}$, 设 $0 \neq u \in \mathbb{R}^n$ 且 $a \in \mathbb{R}$ 待定, 则 uu^T 是秩一矩阵, 且有秩一更新

$$B^{k+1} = B^k + a uu^T.$$

进一步我们确定参量 u 和 a . 根据割线方程 $B^{k+1}s^k = y^k$, 代入秩一更新的结果, 得到

$$(B^k + a uu^T)s^k = y^k,$$

整理得

$$a uu^T s^k = (a \cdot u^T s^k)u = y^k - B^k s^k.$$

由于 $a \cdot u^T s^k$ 是标量, 因此上式表明 u 和 $y^k - B^k s^k$ 同向.

B^k 的SR1更新

根据

$$auu^T s^k = (a \cdot u^T s^k)u = y^k - B^k s^k,$$

我们推出 u 和 $y^k - B^k s^k$ 同向. 此时将会有诸多选择以确定具体的参量. 处于简单考虑, 不妨就令 u 和 $y^k - B^k s^k$ 相等, 即 $u = y^k - B^k s^k$, 代入上式得

$$(a \cdot (y^k - B^k s^k)^T s^k)(y^k - B^k s^k) = y^k - B^k s^k,$$

再令 $(a \cdot (y^k - B^k s^k)^T s^k) \neq 0$, 则可以确定 a 为

$$a = \frac{1}{(y^k - B^k s^k)^T s^k}.$$

至此, u 和 a 均被确定.

根据以上算法, 一种 B^k 的秩一更新(SR1)公式为

$$B^{k+1} = B^k + \frac{uu^T}{u^T s^k}, \quad u = y^k - B^k s^k.$$

秩一更新公式

我们推出了基于 B^k 的秩一更新公式. 由同样的过程(请推导), 可以推出基于 H^k 的秩一更新公式.

定理

拟牛顿算法的秩一更新公式 拟牛顿矩阵 B^k 的秩一更新公式为

$$B^{k+1} = B^k + \frac{uu^T}{u^T s^k}, \quad u = y^k - B^k s^k,$$

拟牛顿矩阵 H^k 的秩一更新公式为

$$H^{k+1} = H^k + \frac{vv^T}{v^T y^k}, \quad v = s^k - H^k y^k.$$

观察秩一公式, 发现基于 B^k 和 H^k 的公式在形式上互为对偶. 这并不是巧合, 实际上, 在基于 B^k 的公式中, 注意 $H^k = (B^k)^{-1}$, 利用秩一更新的SMW公式即可推出基于 H^k 的公式, 反之亦然.

秩一更新公式的缺陷

秩一公式的推导很简单, 形式也很简洁, 但是由秩一公式更新的 B^{k+1} 无法保证正定, 即使 B^k 正定.

定理

秩一更新公式使 B^{k+1} 正定的充分条件 使用秩一更新公式从 B^k 更新 B^{k+1} , B^{k+1} 正定的充分条件可以是:

- (1) B^k 正定;
- (2) $u^T s^k > 0$.

证明是简单的. 设 $0 \neq w \in \mathbb{R}^n$, 则

$$\begin{aligned} w^T B^{k+1} w &= w^T B^k w + \frac{w^T u u^T w}{u^T s^k} = w^T B^k w + \frac{(u^T w)^2}{u^T s^k} \\ &> 0. \end{aligned}$$

同样地, 将上述定理中 B 换成 H , $u^T s^k$ 换成 $v^T y^k$, 仍然成立. 因此, 由于无法保证 $u^T s^k$ 或 $v^T y^k$ 恒大于0, 上述的秩一更新公式一般不用.

BFGS公式

由于SR1公式无法在迭代过程中始终保证 B^k 的正定性, 因此SR1公式的迭代过程可能不收敛. 为了克服这种缺陷, 我们介绍BFGS公式. BFGS公式的核心思想是对 B^k 进行秩二更新, 而不是秩一更新.

定义

秩二更新 对于拟牛顿矩阵 $B^k \in \mathbb{R}^{n \times n}$, 设 $0 \neq u, v \in \mathbb{R}^n$ 且 $a, b \in \mathbb{R}$ 待定, 则有秩二更新形式

$$B^{k+1} = B^k + auu^T + bvv^T.$$

根据割线方程, 将秩二更新的待定参量式代入, 得

$$B^{k+1}s^k = (B^k + auu^T + bvv^T)s^k = y^k,$$

整理可得

$$(a \cdot u^T s^k)u + (b \cdot v^T s^k)v = y^k - B^k s^k.$$

上式的形式和秩一更新的情形类似, 不过是多加了一项.

BFGS公式

类似处理, 我们通过选取合适的 u, v, a, b 使得上式成立, 一个简单的取法是令 $(a \cdot u^T s^k)u$ 对应 y^k 相等, $(b \cdot v^T s^k)v$ 对应 $-B^k s^k$ 相等, 即有

$$a \cdot u^T s^k = 1, \quad u = y^k,$$

$$b \cdot v^T s^k = -1, \quad v = B^k s^k.$$

将上述参量代入割线方程, 即得BFGS更新公式

$$B^{k+1} = B^k + \frac{uu^T}{(s^k)^T u} - \frac{vv^T}{(s^k)^T v},$$

其中 u, v 如上定义.

利用SMW公式以及 $H^k = (B^k)^{-1}$, 可以推出关于 H^k 的BFGS公式. 请读者尝试.

推导 H^k 的BFGS公式之提示

提示: SMW公式在 $k=2$ 时的形式为

$$\left(A + \sum_{i=1}^2 u_i v_i^T \right)^{-1} = A^{-1} - A^{-1} (u_1, u_2) C^{-1} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix} A^{-1},$$

其中 $C = \begin{pmatrix} 1 + v_1^T A^{-1} u_1 & v_1^T A^{-1} u_2 \\ v_2^T A^{-1} u_1 & 1 + v_2^T A^{-1} u_2 \end{pmatrix}$. 在上式中, 我们可以令

$$u_1 = v_1 = \frac{y_k}{\sqrt{s_k^T y_k}}, \quad v_1 = -v_2 = \frac{B_k s_k}{\sqrt{s_k^T B_k s_k}},$$

则SMW公式中二阶矩阵的逆可以化为

$$C^{-1} = \begin{pmatrix} 1 + \frac{y_k^T H_k y_k}{s_k^T y_k} & \frac{\sqrt{s_k^T B_k s_k}}{\sqrt{s_k^T y_k}} \\ -\frac{\sqrt{s_k^T B_k s_k}}{\sqrt{s_k^T y_k}} & 0 \end{pmatrix}^{-1} = \frac{s_k^T y_k}{s_k^T B_k s_k} \begin{pmatrix} 0 & -\frac{\sqrt{s_k^T B_k s_k}}{\sqrt{s_k^T y_k}} \\ \frac{\sqrt{s_k^T B_k s_k}}{\sqrt{s_k^T y_k}} & 1 + \frac{y_k^T H_k y_k}{s_k^T y_k} \end{pmatrix},$$

将 C^{-1} 代入SMW公式即可得到基于 H^k 的BFGS公式.

BFGS公式

综上所述, BFGS公式的定义如下.

定义

BFGS公式 在拟牛顿类算法中, 基于 B^k 的BFGS公式为

$$B^{k+1} = B^k + \frac{y^k (y^k)^T}{(s^k)^T y^k} - \frac{B^k s^k (B^k s^k)^T}{(s^k)^T B^k s^k},$$

基于 H^k 的BFGS公式为

$$H^{k+1} = \left(I - \frac{s^k (y^k)^T}{(s^k)^T y^k} \right)^T H^k \left(I - \frac{s^k (y^k)^T}{(s^k)^T y^k} \right) + \frac{s^k (s^k)^T}{(s^k)^T y^k}.$$

BFGS公式的有效性

BFGS公式产生的 B^{k+1} 或 H^{k+1} 是否正定呢？我们通过一个充分性定理说明.

定理

BFGS公式使拟牛顿矩阵正定的充分条件 使用秩一更新公式从 B^k 或 H^k 更新 B^{k+1} 或 H^{k+1} , 拟牛顿矩阵正定的充分条件可以是:

- (1) B^k 或 H^k 正定;
- (2) 满足**曲率条件** $(s^k)^T y^k > 0, \forall k \in \mathbb{N}^+$.

证明上述定理, 只需要从基于 H^k 的BFGS公式分析即可, 从而得到 H^{k+1} 和其逆 B^{k+1} 均正定.

因为在确定步长时使用某一Wolfe准则线搜索即可满足曲率条件, 因此BFGS公式产生的拟牛顿矩阵有望保持正定, 是有效算法.

从优化意义理解BFGS格式

基于 H^k 的BFGS格式恰好是优化问题

$$\begin{aligned} \min_H \quad & \mathbf{OPT} = \|H - H^k\|_W, \\ \text{s.t.} \quad & H = H^T, \\ & Hy^k = s^k. \end{aligned}$$

的解. 上式中 $\|\cdot\|_W$ 是加权范数, 定义为

$$\|H\|_W = \left\| W^{1/2} H W^{1/2} \right\|_F,$$

且 W 满足割线方程, 即 $Ws^k = y^k$.

注意 $Hy^k = s^k$ 是割线方程, 因此优化问题的意义是在满足割线方程的**对称矩阵**中找到距离 H^k 最近的矩阵 H 作为 H^{k+1} . 因此我们可以进一步认知, BFGS格式更新的拟牛顿矩阵是正定对称的, 且在满足割线方程的条件下采取的是最佳逼近策略.

DFP公式

DFP公式利用与BFGS公式类似的推导方法,不同的是其以割线方程 $H^{k+1}y^k = s^k$ 为基础进行对 H^k 的秩二更新.读者可以再练习推导.基于 H^k 满足的DFP公式,利用SMW公式以及 $B^k = (H^k)^{-1}$,可以推出关于 B^k 的DFP公式. (关键的推导步骤仍然可以参考推导BFGS公式时给出的提示)

定义

DFP公式 基于 H^k 的DFP更新公式为

$$H^{k+1} = H^k - \frac{H^k y^k (H^k y^k)^T}{(y^k)^T H^k y^k} + \frac{s^k (s^k)^T}{(y^k)^T s^k},$$

基于 B^k 的DFP更新公式为

$$B^{k+1} = (I - \frac{y^k (s^k)^T}{(s^k)^T y^k})^T B^k (I - \frac{y^k (s^k)^T}{(s^k)^T y^k}) + \frac{y^k (y^k)^T}{(s^k)^T y^k}.$$

DFP公式的有效性

类似BFGS公式, DFP公式产生的 B^{k+1} 或 H^{k+1} 也正定, 且其充分条件可以与BFGS公式的相同.

定理

DFP公式使拟牛顿矩阵正定的充分条件 使用秩一更新公式从 B^k 或 H^k 更新 B^{k+1} 或 H^{k+1} , 拟牛顿矩阵正定的充分条件可以是:

- (1) B^k 或 H^k 正定;
- (2) 满足**曲率条件** $(s^k)^T y^k > 0, \forall k \in \mathbb{N}^+$.

证明上述定理, 只需要从基于 H^k 的BFGS公式分析即可, 从而得到 H^{k+1} 和其逆 B^{k+1} 均正定.

可以看出, DFP公式与BFGS公式存在对偶的关系. 将BFGS公式中的 H^k 换成 B^k , s^k 换成 y^k , 即可得到DFP公式. 这一点也不奇怪, 因为它们所基于的割线方程恰好也满足这种对偶关系.

从优化意义上理解DFP公式

有了BFGS公式的优化意义做铺垫, 讨论DFP公式的优化意义显得十分简单. 利用对偶性质, 基于 B^k 的DFP格式将是优化问题

$$\begin{aligned} \min_B \quad & \mathbf{OPT} = \|B - B^k\|_W, \\ \text{s.t.} \quad & B = B^T, \\ & Bs^k = y^k. \end{aligned}$$

的解. 上式中 $\|\cdot\|_W$ 是加权范数, 定义为

$$\|B\|_W = \left\| W^{1/2} B W^{1/2} \right\|_F,$$

且 W 满足另一割线方程, 即 $Wy^k = s^k$.

注意 $Bs^k = y^k$ 是另一割线方程, 因此优化问题的意义是在满足割线方程的**对称矩阵**中找到距离 B^k 最近的矩阵 B 作为 B^{k+1} .

DFP公式的缺陷

遗憾的是, 尽管DFP格式与BFGS对偶, 但从实际效果而言, DFP格式的求解效率整体上不如BFGS格式.

M.J.D. Powell曾以问题

$$\min_{x \in \mathbb{R}^2} f(x) = \frac{1}{2} \|x\|_2^2$$

的迭代求解体现过这一结论. 他设置初始值

$$B^0 = \begin{pmatrix} 1 & 0 \\ 0 & \lambda \end{pmatrix}, \quad x_1 = \begin{pmatrix} \cos \psi \\ \sin \psi \end{pmatrix},$$

其中 $\tan^2 \psi = \lambda$. 当误差阈 $\epsilon = 10^{-4}$ 时, 分别取 λ 为不同的值, 使用BFGS算法与DFP算法所产生的迭代步数分别如下表(见下页)所示. 由此看出, 在本问题中, BFGS算法的求解效率要远高于DFP算法.

(参考文献: Powell M J D. How bad are the BFGS and DFP methods when the objective function is quadratic?[J]. Mathematical Programming, 1986, 34(1): 34-47.)

DFP公式的缺陷

Table: BFGS方法的迭代次数

$\lambda \backslash \epsilon$	0.1	0.01	10^{-4}	10^{-8}
10	5	6	8	10
100	7	8	10	12
10^4	12	13	15	17
10^6	17	18	20	22
10^9	24	25	27	29

Table: DFP方法的迭代次数

$\lambda \backslash \epsilon$	0.1	0.01	10^{-4}	10^{-8}
10	10	13	16	19
30	25	32	37	40
100	80	99	107	111
300	237	290	307	313
10^3	787	958	1006	1014

提纲

- 1 介绍
- 2 割线方程
- 3 拟牛顿矩阵更新方式
- 4 拟牛顿类算法的收敛性和收敛速度
- 5 有限内存BFGS方法
- 6 应用举例

BFGS全局收敛性

我们利用Zoutendijk条件得到基本收敛性. 需要复习的读者可参看纸质本Page 213的定理6.1.

根据对BFGS格式有效性的分析, 我们先确保初始矩阵 B^0 是对称正定的.

定理

BFGS全局收敛性 设初始矩阵 B^0 是对称正定矩阵, 目标函数 $f(x)$ 是二阶连续可微函数, 下水平集

$$\mathcal{L} = \{x \in \mathbb{R}^n | f(x) \leq f(x^0)\}$$

凸, 且存在 $m, M \in \mathbb{R}^+$ 使得对 $\forall z \in \mathbb{R}^n, x \in \mathcal{L}$ 满足

$$m \|z\|^2 \leq z^T \nabla^2 f(x) z \leq M \|z\|^2$$

(即 $z^T \nabla^2 f(x) z$ 被 $\|z\|$ 控制), 那么成立:

BFGS格式结合**Wolfe**线搜索的拟牛顿算法全局收敛到 $f(x)$ 的极小值点 x^* .

BFGS全局收敛性的证明

既然涉及到用Wolfe准则进行线搜索,自然会希望通过Zoutendijk条件来说明收敛性.同时,我们每步所取的拟牛顿方向也一定是下降方向,因为BFGS格式更新的拟牛顿矩阵可以在定理的条件下满足对称正定.

因此,证明收敛性,只需要证明搜索方向与负梯度的夹角不太差(Zoutendijk条件的核心思想)。这也是我们下面证明的一个基本思路.基于 B^k 的BFGS格式为

$$B^{k+1} = B^k + \frac{y^k (y^k)^T}{(s^k)^T y^k} - \frac{B^k s^k (B^k s^k)^T}{(s^k)^T B^k s^k},$$

通过这一公式,我们可以说明:

- (a) $\text{Tr}(B^{k+1}) = \text{Tr}(B^k) - \frac{\|B^k s^k\|^2}{(s^k)^T B^k s^k} + \frac{\|y^k\|^2}{(s^k)^T y^k},$
- (b) $\det(B^{k+1}) = \det(B^k) \frac{(s^k)^T y^k}{(s^k)^T B^k s^k}.$

关于迹的公式是显然的,因为迹的运算保和.我们主要说明为何关于行列式的结论成立(习题6.11).

BFGS全局收敛性的证明

为说明(b)式成立, 先证明一个结论.

定理

设 $x, y, u, v \in \mathbb{R}^n$, 则

$$\det(I_{n \times n} + xy^T + uv^T) = (1 + y^T x)(1 + v^T u) - (x^T v)(y^T u).$$

Proof: 只需注意到利用降阶公式成立

$$\begin{aligned}\det(I_{n \times n} + xy^T + uv^T) &= \det\left(I_{n \times n} + \begin{pmatrix} x & u \end{pmatrix} \begin{pmatrix} y^T \\ v^T \end{pmatrix}\right) \\ &= \det\left(I_{2 \times 2} + \begin{pmatrix} x^T \\ u^T \end{pmatrix} \begin{pmatrix} y & v \end{pmatrix}\right).\end{aligned}$$

利用上述定理, 将BFGS公式的左右两边乘以 $(B^k)^{-1}$ (B^k 正定), 并

设 $x = \frac{-s^k}{\sqrt{(s^k)^T B^k s^k}}$, $y = \frac{B^k s^k}{\sqrt{(s^k)^T B^k s^k}}$, $u = \frac{(B^k)^{-1} y^k}{\sqrt{(y^k)^T s^k}}$, $v = \frac{y^k}{\sqrt{(y^k)^T s^k}}$, 代入即可证明结论成立.

BFGS全局收敛性的证明

定义 $\cos \theta_k = \frac{(s^k)^T B^k s^k}{\|s^k\| \|B^k s^k\|}$ 是欲求夹角的余弦. 这是因为

$$\begin{aligned} s^k &= x^{k+1} - x^k = -\alpha_k (B^k)^{-1} \nabla f(x^k), \\ B^k s^k &= -\alpha_k \nabla f(x^k). \end{aligned}$$

再设

$$q_k = \frac{(s^k)^T B^k s^k}{(s^k)^T s^k}, \quad m_k = \frac{(y^k)^T s^k}{(s^k)^T s^k}, \quad M_k = \frac{(y^k)^T y^k}{(y^k)^T s^k},$$

将上述定义代入(b)式以及余弦式, 得到

(c) $\det(B^{k+1}) = \det(B^k) \frac{m_k}{q_k}.$

(d) $\frac{\|B^k s^k\|^2}{(s^k)^T B^k s^k} = \frac{q_k}{\cos^2 \theta_k}.$

上述(a),(b),(c),(d)均是准备公式.

BFGS全局收敛性的证明

我们的目标是通过构造一个不等式, 使得我们证明 $\cos \theta_k > 0, k \rightarrow \infty$.
设

$$\Psi(B) = \text{Tr}(B) - \ln(\det(B)),$$

注意上式成立 $\Psi(B) > 0$.

在上式中代入 $B = B^{k+1}$ 以及上述准备公式, 成立

$$\begin{aligned}\Psi(B^{k+1}) &= \text{Tr}(B^{k+1}) - \ln(\det(B^{k+1})) \\ &\leq \Psi(B^k) + (M_k - \ln(m_k) - 1) + 2 \ln(\cos \theta_k).\end{aligned}$$

(请补完上述跳步)

上述不等式的意义在于, 我们找到了一个关于 B^k 的函数不等式的递推.
同时注意到 $m_k \geq m, M_k \leq M$ (想一想为何可如此假设?), 所以又有

$$\begin{aligned}\Psi(B^{k+1}) &\leq \Psi(B^k) + (M_k - \ln(m_k) - 1) + 2 \ln(\cos \theta_k) \\ &\leq \Psi(B^0) + (k+1)(M - \ln(m) - 1) + 2 \sum_{j=0}^k \ln(\cos \theta_j),\end{aligned}$$

BFGS全局收敛性的证明

从上述不等式(控制式)可以证明, 如果我们假设迭代将不会停止, 则右式若无界, 将导出矛盾.

不妨设 $\cos\theta_k \rightarrow 0$, 因此会有 $\ln(\cos^2\theta_k) \rightarrow -\infty$. 这意味着, 存在 $K \in \mathbb{N}^+$, 使得对 $\forall j \geq K$, 均成立

$$\ln(\cos^2\theta_j) < -2(M - \ln(m) - 1) = -2C < 0,$$

初次证明, 上述不等式右侧的取法可能会引起困惑. 事实上, 右式可以随便取常数(因为只要是常数均成立), 这样取的原因是后续的证明会比较方便.

联立上述2个最近的控制式, 注意 $\Psi(B^{k+1}) > 0$, 则有 $k \rightarrow \infty$ 时

$$0 < \Psi(B^0) + (k+1)C + 2 \sum_{j=0}^K \ln(\cos\theta_j) + \sum_{j=K+1}^k (-2C) \rightarrow -\infty,$$

这就导出了矛盾.

BFGS全局收敛性的证明

因此 $\cos \theta_k \rightarrow 0$ 是不成立的. 换句话说, 存在子列 $\{j_k\}_{k=1,2,\dots}$, 使得 $\cos \theta_{j_k} \geq \delta > 0$. 根据Zoutendijk条件, 又可以得到

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| \rightarrow 0.$$

结合上述问题对 $x \in \mathcal{L}$ 是强凸的, 所以导出 $x^k \rightarrow x^*$.

(若函数的性质稍差, 即函数并非强凸的, 则全局收敛性可能将退化为局部的收敛性.)

BFGS局部收敛性与R-收敛速度

上述全局定理说明在一定假设下, 使用BFGS格式确定下降方向, 搭配Wolfe线搜索确定步长后是全局收敛的. 下面这个定理从局部收敛性给出了其收敛速度.

定理

BFGS局部收敛性 设 $f(x)$ 二阶连续可微, 点列 $\{x_k\}$ 是由BFGS格式产生的, 并收敛于 x^* , $\nabla^2 f(x^*)$ 是对称正定矩阵. 设初始矩阵 B^0 为任意的对称正定矩阵, 那么存在 $0 \leq c < 1$, $K \in \mathbb{N}^+$, 使得对 $\forall k > N$, 成立

$$f(x^{k+1}) - f(x^*) \leq c^{k-K+1} (f(x^K) - f(x^*)),$$

$$\sum_{k=0}^{\infty} \|x^k - x^*\| < \infty.$$

上述定理表明, 序列 $\{f(x^k) - f(x^*)\}_k$ 是压缩的, 收敛将具有R-超线性收敛速度.

BFGS格式的Q-收敛速度

由局部收敛性定理, 可以进一步推出BFGS的Q-收敛速度.

定理

BFGS的Q-超线性收敛速度 除要求BFGS局部收敛性的假设外, 再要求 f 的Hesse矩阵在 x^* 处Lip-连续, 则有

$$\lim_{k \rightarrow \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = 0.$$

即 $\{x^k\}$ 为Q-超线性收敛到 x^* .

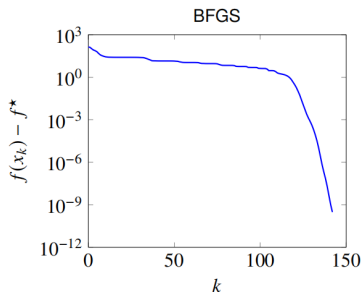
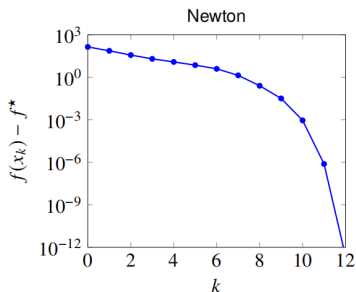
以BFGS格式为代表的拟牛顿类算法由于仅仅使用了Hesse矩阵的近似, 因此很难达到二阶收敛速度, 最多只能达到Q-超线性收敛速度. 但是, 由于拟牛顿方法对近似矩阵的更新代价可能远小于牛顿方法计算Hesse矩阵的代价, 因此它在大规模问题中的开销可能远小于牛顿算法, 较为实用.

BFGS方法的收敛速度之例

例 考虑极小化问题

$$\min_{x \in \mathbb{R}^{100}} c^T x - \sum_{i=1}^{500} \ln(b_i - a_i^T x),$$

下图展示了误差 $f(x_k) - f^*$ 与迭代次数 k 之间的关系(k 是迭代次数). 虽然BFGS方法的迭代次数显著得多, 但由于牛顿法每次迭代的计算代价为 $\mathcal{O}(n^3)$ 加上**计算Hesse矩阵的代价**, 而BFGS方法的每步计算代价仅为 $\mathcal{O}(n^2)$, 因此BFGS算法可能更快取得优势解.



提纲

- 1 介绍
- 2 割线方程
- 3 拟牛顿矩阵更新方式
- 4 拟牛顿类算法的收敛性和收敛速度
- 5 有限内存BFGS方法
- 6 应用举例

有限内存方法的基本思路

牛顿算法在大规模问题中的困难主要体现在计算Hesse矩阵的不易, 然而就算是得到了Hesse矩阵或近似矩阵(一般是稠密矩阵), 要存储它就需要给出 $O(n^2)$ 规模的存储空间, 这或许是极为困难的. 因此, 我们需要研究一类不必在每次迭代中存储近似矩阵的策略.

基本思路 我们先回顾拟牛顿法. 对于标准的拟牛顿法, 近似矩阵的更新公式可以记为

$$B^{k+1} = g(B^k, s^k, y^k),$$

其中 s^k, y^k 如本节定义. 拟牛顿法的基本思路是, 通过给定以上的参量, 我们希望迭代产生 B^{k+1} .

类似地, 既然算法要求限制空间迭代, 那么可被利用的信息一定会受限. 也就是说, 如果我们以迭代的方式将上述更新公式重写, 只保存**最近**的**m组数据**, 那么迭代公式可以写成

$$B^{k+1} = g(g(\cdots g(B^{k-m+1}, s^{k-m+1}, y^{k-m+1}))).$$

有限内存方法的基本思路

为什么只保存最近的 m 组数据是可行的呢？其实，我们在分析算法的收敛性与收敛速度时，对绝大多数问题，都是在最优点 x^* 的局部讨论的。而在算法的迭代过程中，起始点一般都会偏离 x^* 较远，所以丢弃一些历史较远的数据是可以的。

丢弃数据之后，必然会产生一个问题： B^{k-m+1} 的形式几何？

一个简单的做法是我们自己找一个已知的矩阵代替它。讨论如何给出 B^{k-m+1} 的过程，其实就是有限内存拟牛顿类算法的一般思路。

有限内存BFGS方法

先回顾BFGS方法, 它使用BFGS更新公式对拟牛顿矩阵进行迭代更新, 同时利用Wolfe准则执行线搜索确定步长. 我们依然考虑基于这个框架进行改进, 所以我们要选取的搜索方向其实还是

$$d^k = -(B^k)^{-1} \nabla f(x^k) = -H^k \nabla f(x^k).$$

下面我们以基于 H^k 的BFGS更新公式为基础推导算法. 重写BFGS更新公式为

$$H^{k+1} = (V^k)^T H^k V^k + \rho_k s^k (s^k)^T,$$

其中

$$\rho_k = \frac{1}{(y^k)^T s^k}, \quad V^k = I_{n \times n} - \rho_k y^k (s^k)^T.$$

有限内存BFGS方法

有了上述写迭代公式的思路,我们将上式递归地展开 m 次,即

$$\begin{aligned} H^k = & \left(\prod_{j=k-m}^{k-1} V^j \right)^T H^{k-m} \left(\prod_{j=k-m}^{k-1} V^j \right) + \\ & \rho_{k-m} \left(\prod_{j=k-m+1}^{k-1} V^j \right)^T s^{k-m} (s^{k-m})^T \left(\prod_{j=k-m}^{k-1} V^j \right) + \cdots + \\ & \rho_{k-1} s^{k-1} (s^{k-1})^T. \end{aligned}$$

为了节省内存,我们只展开 m 次,利用 H^{k-m} 进行计算,即可求出 H^{k+1} . 虽然 H^{k-m} 还是无法直接显式求出,但我们可以类似拟牛顿方法,用一个结构简单且性质与 H^{k-m} 基本一致的近似矩阵代替计算.

下面我们介绍一种不计算 H^k ,只利用展开式计算 $d^k = -H^k \nabla f(x^k)$ 的巧妙算法: **双循环递归算法**. 它利用迭代式的结构尽量节省计算 d^k 的开销.

有限内存BFGS方法

我们现在给出双循环递归算法的执行过程. 在上述递归式中, 将等式两边同时右乘 $\nabla f(x^k)$, 则等式左侧为 $-d^k$. 若观察等式右侧, 需要计算

$$V^{k-1}\nabla f(x^k), \dots, V^{k-m} \dots V^{k-1}\nabla f(x^k).$$

这些计算可以递归地进行, 而无需重复计算.

同时, 在计算 $V^{k-l} \dots V^{k-1}\nabla f(x^k)$ 的过程中, 恰好可以计算上一步的 $\rho_{k-l}(s^{k-l})^T[V^{k-l+1} \dots V^{k-1}\nabla f(x^k)]$, 这是一个标量. 我们可以记

$$q = V^{k-m} \dots V^{k-1}\nabla f(x^k),$$

$$\alpha_i = \rho_{k-l}(s^{k-l})^T[V^{k-l+1} \dots V^{k-1}\nabla f(x^k)],$$

因此递归公式可化为如下的形式:

$$H^k = \left(\prod_{j=k-m}^{k-1} V^j \right)^T H^{k-m} q + \left(\prod_{j=k-m+1}^{k-1} V^j \right)^T s^{k-m} \alpha_{k-m} + \dots + s^{k-1} \alpha_{k-1}.$$

有限内存BFGS方法

上述递归公式相比于原来的形式已经简化了不少, 因为 $\{\alpha_j\}_{j=k-m}^{k-1}$ 和 q 都是递归计算的结果.

在双循环递归算法中, 除了上述第一个循环递归过程(自下而上)外, 还有以下第二个循环递归过程. 我们需要在公式中自上而下合并每一项. 以前2项为例, 它们有公共的因子 $(V^{k-m+1} \dots V^{k-1})^T$, 提取后可以将前2项写为(注意将 V^{k-m} 的定义回代)

$$\begin{aligned} & (V^{k-m+1} \dots V^{k-1})^T \left[(V^{k-m})^T r + \alpha_{k-m} s^{k-m} \right] \\ &= (V^{k-m+1} \dots V^{k-1})^T \left(r + (\alpha_{k-m} - \beta) s^{k-m} \right), \end{aligned}$$

这正是第二个循环的迭代格式. 注意合并后原递归式的结构仍不变, 因此可以递归地计算下去. 最后, 变量 r 就是我们期望的结果 $H^k \nabla f(x^k)$.

L-BFGS双循环递归算法

拟牛顿算法的基本框架为:

算法 2 L-BFGS双循环递归

Require: 初始化 $q \leftarrow \nabla f(x^k)$.

Ensure: r , 即 $H^k \nabla f(x^k)$.

- 1: 检查初始元素.
 - 2: **for** $i = k - 1, \dots, k - m$ **do**
 - 3: 计算并保存 $\alpha_i \leftarrow \rho_i(s^i)^T q$.
 - 4: 更新 $q \leftarrow q - \alpha_i y^i$.
 - 5: **end for**
 - 6: 初始化 $r \leftarrow \hat{H}^{k-m} q$, 其中 \hat{H}^{k-m} 是 H^{k-m} 的近似矩阵.
 - 7: **for** $i = k - m, \dots, k - 1$ **do**
 - 8: 计算 $\beta \leftarrow \rho_i(y^i)^T r$.
 - 9: 更新 $r \leftarrow r + (\alpha_i - \beta)s^i$.
 - 10: **end for**
-

算法分析

请读者自己手动进行一次双循环递归算法的执行过程. 例如, 可以从 $m = 4$ 开始执行, \hat{H}^{k-m} 假设已知.

L-BFGS双循环递归算法约需要 $4mn$ 次乘法运算, $2mn$ 次加法运算; 若近似矩阵 \hat{H}^{k-m} 是对角矩阵, 则额外需要 n 次乘法运算. 由于 m 不会很大, 因此算法的复杂度是 $\mathcal{O}(mn)$. 算法需要的额外存储为临时变量 α_i , 其大小是 $\mathcal{O}(m)$.

综上所述, L-BFGS双循环算法是非常高效的.

现在的问题就是, \hat{H}^{k-m} 如何设置. 一种取法可以是取对角矩阵

$$\hat{H}^{k-m} = \gamma_k I_{n \times n} \triangleq \frac{(s^{k-1})^T y^{k-1}}{(y^{k-1})^T y^{k-1}} I_{n \times n}.$$

这恰好是BB方法的第一个步长.

综上所述, L-BFGS双循环递归算法的完整过程如下.

L-BFGS方法

Algorithm 3 L-BFGS方法

Input: 选择初始点 x^0 , 参数 $m > 0, k \leftarrow 0$.

Output: 达到收敛准则后的 x^{k+1} .

- 1 检查初始元素 **while** 未达到收敛准则 **do**
 - 2 选取近似矩阵 \hat{H}^{k-m} 使用算法2(L-BFGS双循环递归算法)计算 $d^k = -H^k \nabla f(x^k)$ 使用满足Wolfe准则的线搜索算法确定步长 α_k 更新 $x^{k+1} = x^k + \alpha_k d^k$.
 - if** $k > m$ **then**
 - 3 从内存空间中删除 s^{k-m}, y^{k-m} .
 - 4 计算并保存 $s^k = x^{k+1} - x^k, y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ $k \leftarrow k + 1$.
-

BFGS块迭代格式

实际上, L-BFGS格式下的拟牛顿矩阵的形式可以直接给出, 这为我们分析L-BFGS方法提供了很大的便利. 我们称这类方法为**块迭代**.

引入记号

$$S^k = [s^0, \dots, s^{k-1}], \quad Y^k = [y^0, \dots, y^{k-1}],$$

则成立如下定理.

定理

块迭代格式 设 H^0 为BFGS格式的初始矩阵, 它对称正定; 向量对 $\{s^i, y^i\}_{i=0}^{k-1}$ 满足 $(s^i)^T y^i > 0$, H^k 是由BFGS格式产生的拟牛顿矩阵, 则

$$H^k = H^0 + [S^k \quad H^0 Y^k] \begin{bmatrix} W^k & -((R^k)^{-1})^T \\ -(R^k)^{-1} & 0 \end{bmatrix} \begin{bmatrix} (S^k)^T \\ (Y^k)^T H^0 \end{bmatrix},$$

BFGS块迭代格式

在上述定理中,

$$\begin{aligned}W^k &= \left((R^k)^{-1} \right)^T \left(D^k + (Y^k)^T H^0 Y^k \right) (R^k)^{-1}, \\(R^k)_{ij} &= \begin{cases} (s^{i-1})^T y^{i-1}, & i \leq j, \\ 0, & i > j, \end{cases} \\D^k &= \text{Diag} \left((s^0)^T y^0, (s^1)^T y^1, \dots, (s^{k-1})^T y^{k-1} \right).\end{aligned}$$

上述公式的重要意义在于, 如果给定了 H^0, S^k, Y^k , 可以直接计算出BFGS迭代矩阵 H^k . 如果 H^0 是一个近似矩阵, 那么以上定理将给出L-BFGS迭代矩阵的显式格式, 进而求出 d^k .

BFGS块迭代格式

既然上述格式和L-BFGS双循环递归算法都可以计算 d^k , 使用谁更好呢? 我们有如下的分析.

- 双循环递归算法的复杂度较上述块迭代格式低一个常数量级;
- 块迭代格式更紧凑, 且只涉及矩阵运算, 因此实现更直观;
- 由于进行矩阵操作时可以使用BLAS2/3高效执行, 因此使用块迭代格式的速度可能反而更快.

BFGS块迭代格式

使用块迭代格式还有一个优势,那就是可以利用SMW公式(几乎是直接代入即可得),由 H^k 所满足的迭代格式直接推出 B^k 所满足的迭代格式,依据的还是割线方程以及 $B^k = (H^k)^{-1}$.

定理

块迭代格式 设 B^0 为BFGS格式的初始矩阵,它对称正定;向量对 $\{s^i, y^i\}_{i=0}^{k-1}$ 满足 $(s^i)^T y^i > 0$, B^k 是由BFGS格式产生的拟牛顿矩阵,则

$$B^k = B^0 + \begin{bmatrix} B^0 S^k & Y^k \end{bmatrix} \begin{bmatrix} (S^k)^T B^0 S^k & L^k \\ (L^k)^T & -D^k \end{bmatrix} \begin{bmatrix} (S^k)^T B^0 \\ (Y^k)^T \end{bmatrix},$$
$$(L^k)_{ij} = \begin{cases} (s^{i-1})^T y^{i-1}, & i > j, \\ 0, & i \leq j, \end{cases}$$

其他符号的含义同基于 H^k 的块迭代格式.

L-BFGS方法的优势

正如我们一开始所希望的, L-BFGS方法确实为大规模问题节省了内存开销, 同时它是具有超线性局部收敛性的算法, 因此可以在大规模优化问题中将效率极大提升, 以至于迅速成为了应用最为广泛的拟牛顿类算法之一.

有趣的是, 尽管DFP方法与BFGS方法是互为对偶的, 但很少有人专门研究L-DFP方法. 这可能是因为DFP方法本身的数值表现不如BFGS方法所致.

如有兴趣, 读者可将其作为拓展研究, 尝试写出L-DFP的算法流程, 并对比在某些实际问题中它与L-BFGS的求解结果和效率.

提纲

- 1 介绍
- 2 割线方程
- 3 拟牛顿矩阵更新方式
- 4 拟牛顿类算法的收敛性和收敛速度
- 5 有限内存BFGS方法
- 6 应用举例

基追踪问题

一般的基追踪问题可以描述为:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x\|_1 \\ \text{s.t.} \quad & Ax = b. \end{aligned}$$

其中 $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ 是给定的矩阵和向量.

这是一个约束优化问题, 我们可以尝试将其转为无约束优化问题. 我们现在转换这类问题的得力工具只有考虑对偶问题. 由于上述问题的目标函数不是强凸的, 因此其对应的对偶问题的无约束优化形式不可微, 进而无法计算梯度. 因此, 我们考虑正则化问题

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x\|_1 + \frac{1}{2\alpha} \|x\|_2^2 \\ \text{s.t.} \quad & Ax = b. \end{aligned}$$

其中 $\alpha > 0$ 是正则化参数, 当 $\alpha \rightarrow \infty$ 时, 问题的解将会逼近原基追踪问题的解.

基追踪问题

显然, 正则化问题的目标函数是强凸的, 因此其对应的对偶问题的无约束优化形式可微, 方便设计梯度类算法求解.

具体而言, 正则化问题的对偶问题为

$$\min_{y \in \mathbb{R}^n} f(y) = -b^T y + \frac{\alpha}{2} \|A^T y - \mathcal{P}_{[-1,1]^n}(A^T y)\|_2^2,$$

其中 $\mathcal{P}_{[0,1]^n}(z)$ 是 z 到集合 $[-1, 1]^n$ 的投影. 且

$$\nabla f(y) = -b + \alpha A(A^T y - \mathcal{P}_{[-1,1]^n}(A^T y)),$$

那么, 我们可以利用 **L-BFGS** 方法求解正则化问题的对偶问题. 在得到数值解 y^* 后, 原基追踪问题可以由近似式

$$x^* \approx \alpha(A^T y^* - \mathcal{P}_{[-1,1]^n}(A^T y^*))$$

得到. 利用 $\alpha \rightarrow \infty$ 时的逼近性, 选取足够大的 α , 求解正则化问题即可获得原问题的解.

基追踪问题

例 简单基追踪问题

考虑标准基追踪问题, 其中 A 是 512×1024 规模的随机矩阵(每个元素从标准正态分布中抽样), b 定义为

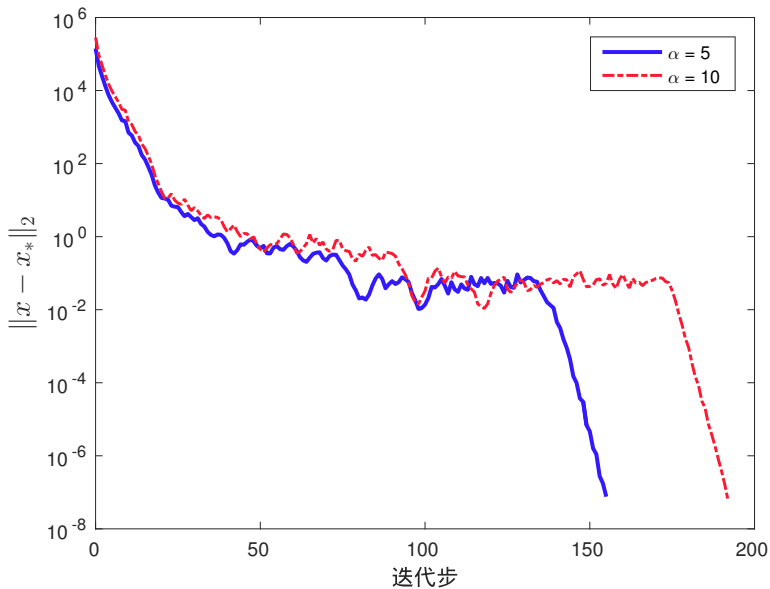
$$b = Au,$$

其中 $u \in \mathbb{R}^{1024}$ 是服从正态分布随机稀疏向量, 我们不妨设其稀疏度为0.1, 即约具有102个服从正态分布的非零分量. 上述 A, b 的设置和6.2节中的完全一致.

分别选取 $\alpha = 5, 10$. 调用L-BFGS算法求解以上述标准问题为原本的正则化问题, 内存长度不妨为 $m = 5$.

下图(见下页)是上述正则化问题的迭代收敛过程. 我们可以看到, 当靠近最优解 x^* 时, L-BFGS方法的迭代点列呈Q-线性收敛.

基追踪问题



逻辑回归问题

在解决基追踪问题时, 我们已经提到了正则化的手段. 下面要考虑的逻辑回归问题也要用到正则化技术.

考虑逻辑回归问题

$$\min_{x \in \mathbb{R}^n} \ell(x) = \frac{1}{m} \sum_{i=1}^m \ln(1 + \exp(-b_i a_i^T x)) + \lambda \|x\|_2^2,$$

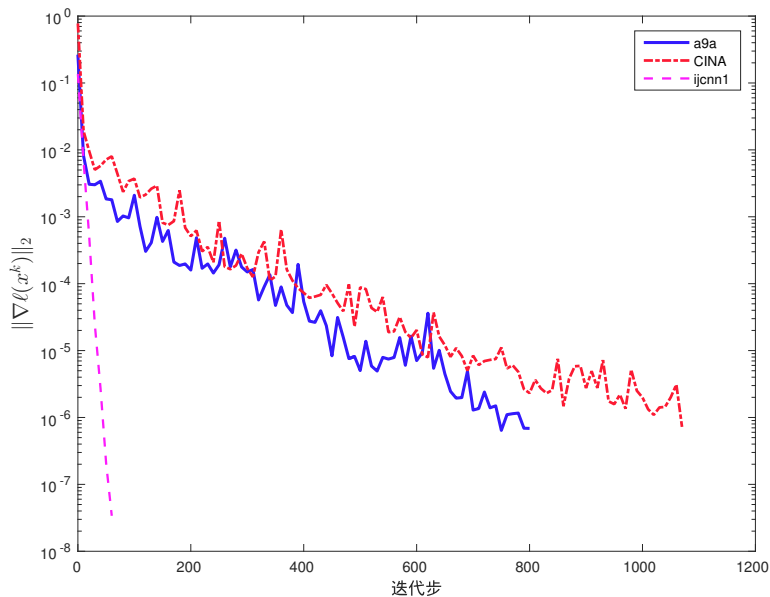
并取 $\lambda = \frac{1}{100m}$. 由于

$$\nabla \ell(x) = \frac{1}{m} \sum_{i=1}^m \frac{-b_i a_i}{1 + \exp(b_i a_i^T x)} + 2\lambda x,$$

对合适的数据集调用L-BFGS算法即可求解上述逻辑回归问题, 内存长度不妨为 $m = 5$.

下图(见下页)展示了上述问题及L-BFGS算法在LIBSVM上的三个数据集的表现结果. 请读者分析不同数据集中L-BFGS算法表现的异同, 并说明这些异同产生的原因.

逻辑回归问题



参考文献

- [1] 刘浩洋, 户将, 李勇峰, 文再文. 最优化: 建模、算法与理论[M]. 高等教育出版社, 2020, 12: 240-253.
- [2] Nocedal J, Wright S. Numerical optimization[M]. Springer Science and Business Media, 2006(135-161).
- [3] Li D H, Fukushima M. A modified BFGS method and its global convergence in nonconvex minimization[J]. Journal of Computational and Applied Mathematics, 2001, 129(1-2): 15-35.
- [4] Li D H, Fukushima M. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems[J]. SIAM Journal on Optimization, 2001, 11(4): 1054-1064.
- [5] Powell M J D. How bad are the BFGS and DFP methods when the objective function is quadratic?[J]. Mathematical Programming, 1986, 34(1): 34-47.
- [6] Dennis, Jr J E, Moré J J. Quasi-Newton methods, motivation and theory[J]. SIAM review, 1977, 19(1): 46-89.