

# 1. 基于线搜索的无约束问题的最优化算法

朱天宇

最优化方法从分类级别从高到低的顺序来说，可以分为几层：

基于迭代下降的方法 - 基于线搜索的方法 - 基于梯度的方法 - 具体的各种方法

本节内容主要介绍基于线搜索的方法。

## 1 Prototype 算法原型

需要准备 初始点，搜索方向，步长

- (a) 给定初始点  $\mathbf{x}^{(0)}$
- (b) 计算搜索方向  $\mathbf{d}^{(k)}$ ，即构造某价值函数  $\psi$  在  $\mathbf{x}^{(k)}$  点处的下降方向作为搜索方向；
- (c) 确定步长因子  $\alpha_k$ ，使该价值函数值有某种程度的下降；
- (d) 迭代更新，令  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ .
- (e) 判断停机准则，若  $\mathbf{x}^{(k+1)}$  满足某种终止条件，则停止迭代，得到近似最优解  $\bar{\mathbf{x}} = \mathbf{x}^{(k+1)}$ 。否则，返回(b)重复以上步骤。

## 2 基于梯度的方法

基于梯度的方法的重点是如何构造搜索方向。其使用负梯度方向来构造搜索方向  $\mathbf{d}^k$ 。主要步骤有 3 步：

1. 初始化，选择一个合适的起点  $\mathbf{x}^0 \in \mathbb{R}^n$ ， $k \leftarrow 0$
2. 搜索方向  $\mathbf{d}^k \leftarrow -\mathbf{H}_k \nabla f(\mathbf{x}^k)$ ，其中  $\mathbf{H}_k$  是一个正定、对称矩阵
3. 确定步长因子，这通过解一个一维的最优化问题  $\min_{\alpha \geq 0} f(\mathbf{x}^k + \alpha \mathbf{d}^k)$  来获得。
4. 更新： $k \leftarrow k + 1, \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha_k \mathbf{d}^k$

### 2.1 $\mathbf{H}_k$ 是单位阵，最速下降法

此时  $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$ 。对于无约束的最优化问题  $\min_{\mathbf{x} \in \mathbb{R}^n}$ ，泰勒展开有：

$$f(\mathbf{x}) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\top (\mathbf{x} - \mathbf{x}^k) + O(\|\mathbf{x} - \mathbf{x}^k\|^2)$$

当  $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$  的时候，总是存在一个足够小的  $\alpha_k$ ，使得

$$f(\mathbf{x}^k + \alpha_k \mathbf{d}^k) < f(\mathbf{x}^k)$$

这通过将  $f(\mathbf{x}^k + \alpha_k \mathbf{d}^k)$  进行泰勒展开后就可以验证。

虽然  $\mathbf{H}_k$  为单位阵的时候，搜索方向是函数下降最快的方向，但这并不意味着这么做结果能达到最好。

## 2.2 $H_k = (\nabla^2 f(\mathbf{x}^k))^{-1}$ 是 Hessian 矩阵的逆矩阵

这里要求 **Hessian** 矩阵是正定的，也就是每个特征值都大于 0。同样，将  $f(\mathbf{x}^k + \alpha_k \mathbf{d}^k)$  在  $\mathbf{x}^k$  处进行二阶的泰勒展开：

$$\begin{aligned} f(\mathbf{x}^k + \alpha_k \mathbf{d}^k) &= f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\top (\mathbf{x}^k + \alpha_k \mathbf{d}^k - \mathbf{x}^k) \\ &\quad + \frac{1}{2} (\mathbf{x}^k + \alpha_k \mathbf{d}^k - \mathbf{x}^k)^\top \nabla^2 f(\mathbf{x}^k) (\mathbf{x}^k + \alpha_k \mathbf{d}^k - \mathbf{x}^k) \\ &\quad + O(\|(\mathbf{x}^k + \alpha_k \mathbf{d}^k - \mathbf{x}^k)\|^3) \\ &= f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\top (\alpha_k \mathbf{d}^k) \\ &\quad + \frac{1}{2} (\alpha_k \mathbf{d}^k)^\top \nabla^2 f(\mathbf{x}^k) (\alpha_k \mathbf{d}^k) \\ &\quad + O(\|(\alpha_k \mathbf{d}^k)\|^3) \\ &= f(\mathbf{x}^k) + \alpha_k \nabla f(\mathbf{x}^k)^\top \mathbf{d}^k \\ &\quad + \frac{1}{2} \alpha_k^2 \mathbf{d}^{(k)\top} \nabla^2 f(\mathbf{x}^k) \mathbf{d}^k \\ &\quad + O(\|(\alpha_k \mathbf{d}^k)\|^3) \end{aligned}$$

在这个式子中，一阶展开为

$$\begin{aligned} \alpha_k \nabla f(\mathbf{x}^k)^\top \mathbf{d}^k &= -\alpha_k \nabla f(\mathbf{x}^k)^\top (\nabla^2 f(\mathbf{x}^k))^{-1} \nabla f(\mathbf{x}^k) \\ &= -\alpha_k \nabla f(\mathbf{x}^k)^\top \mathbf{G}_k^{-1} \nabla f(\mathbf{x}^k) \end{aligned}$$

这是一个二次型，中间的 **Hessian** 矩阵的逆矩阵也是正定的。所以此项的值为负。

对于二阶展开

$$\begin{aligned} \frac{1}{2} \alpha_k^2 (\mathbf{d}^k)^\top \nabla^2 f(\mathbf{x}^k) \mathbf{d}^k &= \frac{1}{2} \alpha_k^2 \nabla f(\mathbf{x}^k)^\top (\mathbf{G}_k^{-1})^\top \mathbf{G}_k \mathbf{G}_k^{-1} \nabla f(\mathbf{x}^k) \\ &= \frac{1}{2} \alpha_k^2 \nabla f(\mathbf{x}^k)^\top (\mathbf{G}_k^{-1})^\top \nabla f(\mathbf{x}^k) \end{aligned}$$

这也是一个二次型，且由于 **Hessian** 矩阵是正定对称的，所以  $(\mathbf{G}_k^{-1})^\top = \mathbf{G}_k^{-1}$ ，所以第二项的中间的矩阵和第一项的中间的矩阵是一样的。而由于  $\alpha_k$  是一个很小的正数，所以  $\frac{1}{2} \alpha_k^2 - \alpha_k < 0$ ，所以一阶项与二阶项的和是小于 0 的。

因此同样有

$$f(\mathbf{x}^k + \alpha_k \mathbf{d}^k) < f(\mathbf{x}^k)$$

实际上， $\mathbf{d}^k$  是函数在  $\mathbf{x}^k$  点处二次展开的极小点。

## 2.3 确定步长因子 $\alpha$

一般来说， $\alpha$  肯定是要尽可能的大，这样下降就会比较快。但如果  $\alpha$  太大，可能根本无法保证下降。因此要通过优化以下这个一维问题来确定最大步长。

$$\min_{\alpha \geq 0} \varphi(\alpha) = f(\mathbf{x}^k + \alpha \mathbf{d}^k)$$

如果以这个方程的最优解作为步长，此时就称为“**精确的一维搜索**”。通常用黄金分割法或者插值迭代法（指先采样若干个点，插值出一个函数再求最优解）来处理。

与之相区别，“**非精确的一维搜索**”只希望找到某些条件的粗略近似解，可以提高整体的计算效率。

设  $\bar{\alpha}_k$  是使得  $f(\mathbf{x}^k + \alpha \mathbf{d}^k) = f(\mathbf{x}^k)$  的最小的正数。由于预设函数  $\varphi$  满足 **单峰性**，所以就需从区间  $[0, \bar{\alpha}_k]$  中寻找一个可以接受的步长  $\alpha$  作为步长。比如通过 **Goldstein** 条件就可以快速找到一个令人满意的值。

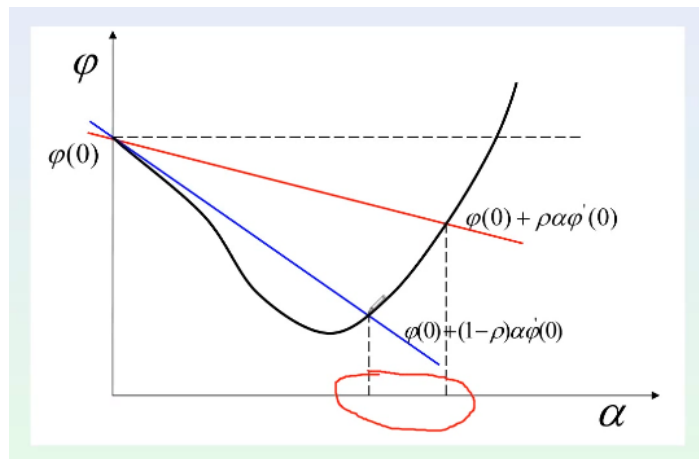
**Goldstein 条件**

$$\begin{aligned} \varphi(\alpha) &\leq \varphi(0) + \rho \alpha \varphi'(0) \\ \varphi(\alpha) &\geq \varphi(0) + (1 - \rho) \alpha \varphi'(0) \end{aligned}$$

其中  $\rho \in (0, \frac{1}{2})$ ，是一个固定的参数。

上式可以保证  $\alpha$  满足一定的下降率。因为  $\varphi(0)$  表示  $f$  在  $\mathbf{x}^k$  的取值， $\varphi(0) + \rho \alpha \varphi'(0)$  表示关于  $\alpha$  的一条直线，而  $\varphi(\alpha)$  要在这条直线的下方（ $\varphi'(0) = \nabla f^\top \mathbf{d}^k < 0$ ，斜率小于 0）。而  $\varphi(\alpha)$  要更小。

同理，下面的式子则表示图中蓝色的线。由于  $\rho \in (0, \frac{1}{2})$ ，所以  $1 - \rho > \rho$ ，所以蓝色的线更加陡一点。添加一条蓝色的线是为了防止  $\alpha$  无线接近于 0，防止下降太慢。



这样， $\alpha$  就落在图中的红圈内部。

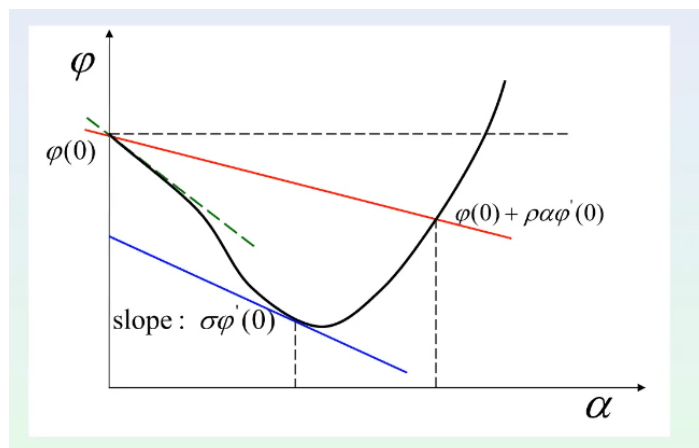
显然，从图中就可以看出 Goldstein 条件的问题。在图中，最好的  $\alpha$  的取值被忽略了。最好能找到一个不排除最优解的区域。

### Wolfe-Powell 条件

$$\varphi(\alpha) \leq \varphi(0) + \rho\alpha\varphi'(0)$$

$$\varphi'(\alpha) \geq \sigma\varphi'(0)$$

其中， $\sigma \in (\rho, 1)$ ，是另一个固定的参数。上面的约束和 Goldstein 条件相同。下面的约束对蓝色直线的斜率进行了约束。具体可以参见下图。因为在最优解的地方，切线的斜率为 0； $\sigma\varphi'(0)$  是一个小于 0 的值，因此约束  $\varphi'(\alpha) \geq \sigma\varphi'(0)$  是可以保证  $\alpha$  最小取值是在最优解左侧的。但保证一个负的最小值又可以防止  $\alpha$  太靠近 0。



有时候，也可以使用这个条件：

$$|\varphi'(\alpha)| \leq -\sigma\varphi'(0)$$

**基于 Wolfe-Powell 准则的一维搜索算法** 如果从区间内随便选点的话很容易代价过大，因此需要根据准则，定制一套明确的算法。

#### 1. 初始化

规定初始搜索区间为  $[0, \bar{\alpha}]$ ;

选择固定的  $\rho \in (0, 1/2)$ ;  $\sigma \in (\rho, 1)$ ,

令  $\varphi_0 = \varphi(0) = f(\mathbf{x}^k)$ ;  $\varphi'_0 = \varphi'(0) = \nabla f(\mathbf{x}^k)^\top \mathbf{d}^k$ ,

定义  $a_1 = 0, a_2 = \bar{\alpha}$  作为  $\alpha$  取值的上下界,  $\varphi_1 = \varphi_0, \varphi'_1 = \varphi'_0$ ,

选择适当的  $\alpha \in (a_1, a_2)$

#### 2. 调整下界

计算  $\varphi(\alpha) = f(\mathbf{x}^k + \alpha\mathbf{d}^k)$ ，如果满足  $\varphi(\alpha) \leq \varphi(0) + \rho\alpha\varphi'(0)$ ，则进入下一步；

否则更新  $\hat{\alpha}$

$$\hat{\alpha} = a_1 + \frac{1}{2} \frac{(a_1 - \alpha)^2 \varphi'_1}{(\varphi_1 - \varphi) - (a_1 - \alpha) \varphi'_1}$$

这实际上是对  $\varphi_1, \varphi'_1, \varphi$  三点构造的二次插值多项式的极小点。

令  $a_2 = \alpha, \alpha = \hat{\alpha}$ , 缩小取值范围, 并重复此步。

### 3. 调整上界

计算  $\varphi' = \varphi'(\alpha) = \nabla f(\mathbf{x}^k + \alpha \mathbf{d}^k)^\top \mathbf{d}^k$ , 如果满足  $\varphi'(\alpha) \geq \sigma \varphi'(0)$ , 则输出  $\alpha_k = \alpha$ , 算法结束。

否则更新  $\hat{\alpha}$

$$\hat{\alpha} = \alpha - \frac{(a_1 - \alpha)\varphi'}{\varphi'_1 - \varphi'}$$

这同样是二次插值的极小点。

令  $a_1 = \alpha, \alpha = \hat{\alpha}, \varphi_1 = \varphi, \varphi'_1 = \varphi'$ , 返回上一步重新调整下界。

## 2.4 全局收敛与局部收敛

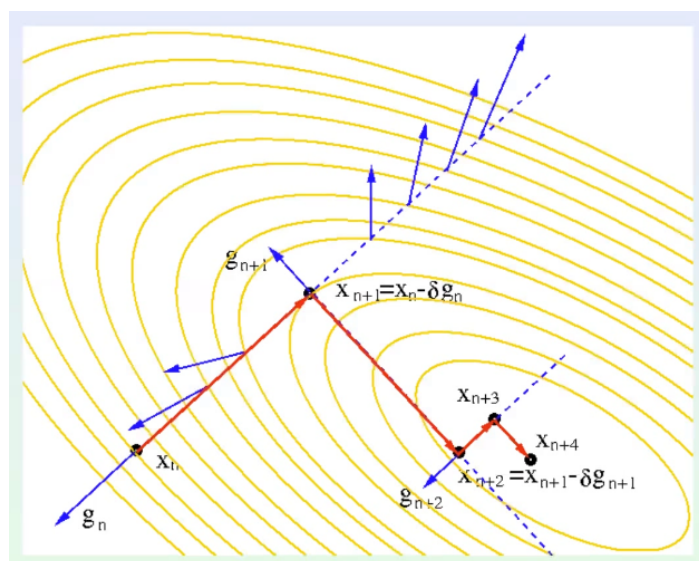
指初始点选择的范围大小。以上的方法都满足全局收敛的要求。

**全局收敛** 从任意初始点出发, 都可以收敛到最优解, 称为全局收敛。

**局部收敛** 初始点必须在最优解的附近。

## 3 最速下降法

最速下降法以负梯度方向作为下降方向。其满足全局收敛性, 也就是说任意一个点采用最速下降法, 其梯度方向都会趋向于 0 向量。具体迭代过程可以参考这张图:



图中, 每一次移动的方向都是当前点  $\mathbf{x}_n$  的负梯度方向。而每次移动的长度则由线搜索来决定。

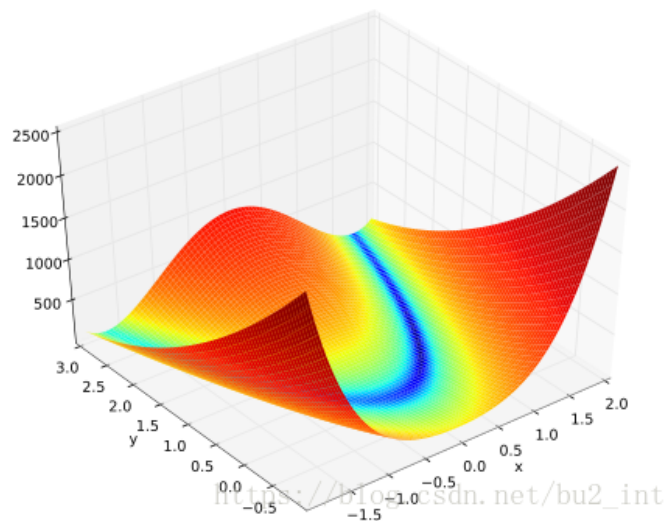
在图中,  $\mathbf{g}^{k+1}$  和  $\mathbf{g}^k$  是相互垂直的。这是因为图中的线搜索是“精确的一维搜索”, 那么这两个梯度就是互相垂直的。呈现如图的情况。如果不是精确的一维搜索, 则有可能更快也可能变慢。

这种方法, 点的移动轨迹是一个“Z”字型。如果采用一些策略, 比如改变方向或者改变步长, 则可能可以更快的收敛到结果。因此, 最速下降法的收敛取决于函数等值线的形状。

最速下降法通常只有线性的收敛速度, 对于无约束的问题来说, 这是远远不够的。比如对于著名的测试函数 Rosenbrock function, 这个函数的等值线是一个长短轴比例非常悬殊的椭圆形。

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

这个函数的图像是:



## 4 牛顿法

牛顿法的下降方向是 **Hessian** 矩阵的逆矩阵乘以梯度方向。设  $f(\mathbf{x})$  是二次可微的实函数，在  $\mathbf{x}^k$  处做二阶的泰勒展开，有

$$f(\mathbf{x}^k + \mathbf{s}) \approx q^k(\mathbf{s}) = f(\mathbf{x}^k) + \mathbf{g}^{(k)\top} \mathbf{s} + \frac{1}{2} \mathbf{s}^\top \mathbf{G}_k \mathbf{s}$$

其中  $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$ ，是梯度； $\mathbf{G}_k = \nabla^2 f(\mathbf{x}^k)$ ，是 **Hessian** 矩阵；将  $q^k(\mathbf{s})$  极小化，就可以得到

$$\mathbf{s} = -\mathbf{G}_k^{-1} \mathbf{g}^k$$

将  $\mathbf{s}$  作为搜索方向，就称为**牛顿方向**。

对于正定二次函数

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{G} \mathbf{x} - \mathbf{c}^\top \mathbf{x}$$

对于任意的  $\mathbf{x}$ ，都有  $\nabla^2 f(\mathbf{x}) = \mathbf{G}$ 。那么

$$\mathbf{d}^0 = -\mathbf{G}^{-1} \nabla f(\mathbf{x}^0) = -\mathbf{G}^{-1} (\mathbf{G} \mathbf{x}^0 - \mathbf{c}) = -(\mathbf{x}^0 - \mathbf{x}^*)$$

使用  $\mathbf{x}^* = \mathbf{G}^{-1} \mathbf{c}$  表示问题的最优解。如果  $\mathbf{x}^0 \neq \mathbf{x}^*$ ，那么取步长  $\alpha_0 = 1$ ，就有  $\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0 \mathbf{d}^0 = \mathbf{x}^*$ ，也就是说经过一次迭代之后就可以达到最优解，而最速下降法需要多次迭代。最速下降法下降的速度取决于 **Hessian** 矩阵的特征值的差异，也对应椭圆的长短轴。

基于以上事实，可以认为对于一般的非线性函数，在迭代中取这样的搜索方向也是比较合适的。

特别的，令步长  $\alpha \equiv 1$ ，迭代公式就变为

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{G}_k^{-1} \mathbf{g}^k$$

这就是经典的牛顿法更新公式。

### 性质

1. 对于正定二次函数，牛顿法可以一步达到最优解。对于非二次函数，牛顿法无法保证有限次迭代；但是在初始点靠近极小点的时候，收敛速度一般会很快，因为此时目标函数可以较好的用二次函数来模拟。
2. 牛顿法是 **局部收敛** 的，在满足一定条件的情况下，收敛速率是二阶的。

收敛性证明 (略，详见视频)

### 4.1 改进策略

**阻尼牛顿法** 阻尼牛顿法对步长的步骤进行改进，采用一维搜索来确定步长，从而避免局部收敛。

1. **初始化** 初始点  $\mathbf{x}^0$ ，设置终止误差  $\epsilon > 0$
2. 计算  $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$ ，如果  $\|\mathbf{g}^k\| < \epsilon$ ，则算法结束
3. **解线性方程组**  $\mathbf{G}_k \mathbf{d} = -\mathbf{g}^k$ ，获得牛顿方向  $\mathbf{d}^k$
4. 采用一维搜索确定  $\alpha_k$ ，更新  $\mathbf{x}$ ，并且回到第 1 步

这种方法可以缓解经典牛顿法局部收敛的缺点，但是需要解线性方程组，且无法确保  $\mathbf{d}^k$  满足全局收敛的条件。

一般来说早期会使用一维搜索确定步长，后期则固定  $\alpha_k = 1$

$$\mathbf{d}^k = \begin{cases} -\mathbf{G}_k^{-1} \mathbf{g}^k, & \text{if } \cos \theta_k > \eta \\ -\mathbf{g}^k, & \text{otherwise} \end{cases}$$

也就是根据  $\theta$  的取值来决定使用最速下降还是牛顿法。可以保证全局收敛性。缺点是不太好分析收敛效率。

Levenberg, Marquardt, Goldfeld, et. al

$$(\mathbf{G}_k + \mu_k \mathbf{I}) \mathbf{d}^k = -\mathbf{g}^k$$

L-M 方法的优势是，在阻尼牛顿法中，需要  $\mathbf{G}_k \mathbf{d} = -\mathbf{g}^k$ ，这么做可以便于分析。是一种牛顿方向和负梯度方向的综合，可以通过  $\mu_k$  调节倾向于牛顿方向还是负梯度方向。 $\mu_k$  取值可以大于  $\mathbf{G}_k$  的最小特征值，使其变成正定矩阵。

负曲率方向法 如果方向  $\mathbf{d}$ ，满足

$$\mathbf{d}^\top \nabla^2 f(\mathbf{x}) \mathbf{d} < 0$$

则称其为负曲率方向。

当 Hessian 矩阵不正定的时候，可以用负曲率方向来修正牛顿法。

## 5 拟牛顿法

运用牛顿法需要计算二阶导数，代价高，且 Hessian 矩阵不一定正定，甚至奇异。如果使用不含二阶导的矩阵  $H_k$  来近似牛顿法中的 Hessian 矩阵的逆，这种方法被称为 **拟牛顿法**。

拟牛顿法使用 Hessian 矩阵或者其逆矩阵来近似，来替代求解 Hessian 矩阵的过程。其近似的矩阵记为  $\mathbf{B}^k$ ，保留了 Hessian 矩阵的部分性质。 $\mathbf{B}^k$  的逆矩阵记为  $\mathbf{H}^k$ 。

构造近似矩阵有不同的方法，因此拟牛顿法也有多种类型。

### 5.1 割线方程

牛顿法的推导基于函数  $f(\mathbf{x})$  的泰勒二次展开：

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

这里，令  $\mathbf{x} = \mathbf{x}^k$ ,  $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ ,  $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$ ，上式可以写成

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

忽略高阶项  $\mathcal{O}$ ，那么我们希望 Hessian 的近似矩阵  $\mathbf{B}^{k+1}$  满足

$$\mathbf{y}^k = \mathbf{B}^{k+1} \mathbf{s}^k$$

使用  $\mathbf{H}^{k+1}$  表示  $\mathbf{B}^k$  的逆矩阵，就有

$$\mathbf{s}^k = \mathbf{H}^{k+1} \mathbf{y}^k$$

这两个式子被称为**割线方程**。

割线方程的本质是希望目标函数  $f(\mathbf{x})$  和其二次近似  $m_{k+1}(\mathbf{d}) = f(\mathbf{x}^{k+1}) + \nabla f(\mathbf{x}^{k+1})^\top \mathbf{d} + \frac{1}{2} \mathbf{d}^\top \mathbf{B}^{k+1} \mathbf{d}$ ，在  $\mathbf{x} = \mathbf{x}^k$ ,  $\mathbf{x} = \mathbf{x}^k + 1$  处有相同的梯度。

**曲率条件** 近似矩阵  $\mathbf{B}^k$  需要正定，对式子  $\mathbf{y}^k = \mathbf{B}^{k+1} \mathbf{s}^k$  两边同时左乘  $(\mathbf{s}^k)^\top$ ，就可以得到**必要条件**

$$(\mathbf{s}^k)^\top \mathbf{B}^{k+1} \mathbf{s}^k = (\mathbf{s}^k)^\top \mathbf{y}^k > 0$$

这被称为曲率条件。

在线搜索的时候，需要使用 Wolfe 准则来使得曲率条件成立。根据 Wolfe 准则的第二个条件  $\varphi'(\alpha) \geq \sigma \varphi'(0)$ ，写成梯度形式，两侧同时乘以  $\mathbf{s}^k$ ，就有

$$\nabla f(\mathbf{x}^{k+1})^\top \mathbf{s}^k \geq \sigma \nabla f(\mathbf{x}^k)^\top \mathbf{s}^k$$

两边同时减去  $\nabla f(\mathbf{x}^k)^\top \mathbf{s}^k$ ，就有

$$(\mathbf{y}^k)^\top \mathbf{s}^k \geq (\sigma - 1) \nabla f(\mathbf{x}^k)^\top \mathbf{s}^k > 0$$

因为  $\sigma < 1$ ，且  $\mathbf{s}^k = \alpha_k \mathbf{d}^k$ ，是下降方向。

通常，近似矩阵  $\mathbf{B}^k, \mathbf{H}^k$  都迭代更新的。有的方法近似  $\mathbf{B}^k$ ，有的方法近似  $\mathbf{H}^k$ 。



## 5.2 秩一更新 SR1

使用待定系数法，已知  $\mathbf{B}^k$ ，满足割线方程，求  $\mathbf{B}^{k+1}$ 。

设  $\mathbf{B}^{k+1} = \mathbf{B}^k + a\mathbf{u}\mathbf{u}^\top$ ，带入割线方程

$$\mathbf{B}^{k+1}\mathbf{s}^k = (\mathbf{B}^k + a\mathbf{u}\mathbf{u}^\top)\mathbf{s}^k = \mathbf{y}^k$$

从而有

$$(a \cdot \mathbf{u}^\top \mathbf{s}^k)\mathbf{u} = \mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k$$

由于  $(a \cdot \mathbf{u}^\top \mathbf{s}^k)$  是标量，因此  $\mathbf{u}$  与  $\mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k$  方向是相同的。不妨令  $\mathbf{u} = \mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k$ ，带入上式，有

$$a((\mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k)^\top \mathbf{s}^k)(\mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k) = \mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k$$

可得  $a = \frac{1}{(\mathbf{y}^k - \mathbf{B}^k \mathbf{s}^k)^\top \mathbf{s}^k}$ ，带入最初的式子，就有

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

相同方法可得

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

一个有趣的观察是，将公式 1 做如下替换则可获得公式 2:

$$\mathbf{B}^k \rightarrow \mathbf{H}^k, \quad \mathbf{s}^k \rightarrow \mathbf{y}^k$$

SR1 公式结构简单，但无法保证矩阵在迭代中保持正定。之前的条件只是充分条件。

## 5.3 秩二更新 BFGS

同样使用待定系数法，设

$$\mathbf{B}^{k+1} = \mathbf{B}^k + a\mathbf{u}\mathbf{u}^\top + b\mathbf{v}\mathbf{v}^\top$$

可推出基于  $\mathbf{B}^k$  的更新公式:

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

带入 SMW 公式可以获得  $\mathbf{H}^k$  的更新公式:

$$\mathbf{H}^{k+1} = (\mathbf{I} - \rho_k \mathbf{s}^k (\mathbf{y}^k)^\top)^\top \mathbf{H}^k (\mathbf{I} - \rho_k \mathbf{s}^k (\mathbf{y}^k)^\top) + \rho_k \mathbf{s}^k (\mathbf{s}^k)^\top$$

其中， $\rho_k = \frac{1}{(\mathbf{s}^k)^\top \mathbf{y}^k}$

BFGS 公式产生的矩阵  $\mathbf{H}^{k+1}$  正定的条件是满足不等式  $(\mathbf{s}^k)^\top \mathbf{y}^k > 0$ ，这可以通过线搜索保证。

## 5.4 有限内存的 BFGS

大规模问题需要存储拟牛顿矩阵  $\mathbf{B}^k$  或者  $\mathbf{H}^k$ ，需要消耗  $\mathcal{O}(n^2)$  的内存，不现实。LBFGS 可以解决这个问题。

首先，基于  $\mathbf{H}^k$  的 BFGS，引入新符号后的表示为

$$\mathbf{H}^{k+1} = (\mathbf{V}^k)^\top \mathbf{H}^k \mathbf{V}^k + \rho_k \mathbf{s}^k (\mathbf{s}^k)^\top$$

其中， $\rho_k = \frac{1}{(\mathbf{s}^k)^\top \mathbf{y}^k}$ ,  $\mathbf{V}^k = \mathbf{I} - \rho_k \mathbf{y}^k (\mathbf{s}^k)^\top$

这个公式是一个 **类似递推的形式**，因此尝试展开  $m$  次:

$$\begin{aligned} \mathbf{H}^k &= (\mathbf{V}^{k-m} \dots \mathbf{V}^{k-1})^\top \mathbf{H}^{k-m} (\mathbf{V}^{k-m} \dots \mathbf{V}^{k-1}) + \\ &\quad \rho_{k-m} (\mathbf{V}^{k-m+1} \dots \mathbf{V}^{k-1})^\top \mathbf{s}^{k-m} (\mathbf{s}^{k-m})^\top (\mathbf{V}^{k-m+1} \dots \mathbf{V}^{k-1}) + \\ &\quad \rho_{k-m+1} (\mathbf{V}^{k-m+2} \dots \mathbf{V}^{k-1})^\top \mathbf{s}^{k-m+1} (\mathbf{s}^{k-m+1})^\top (\mathbf{V}^{k-m+2} \dots \mathbf{V}^{k-1}) + \\ &\quad \dots + \\ &\quad \rho_{k-1} \mathbf{s}^{k-1} (\mathbf{s}^{k-1})^\top \end{aligned}$$

这样，通过  $\mathbf{H}^{k-m}$  就可以获得  $\mathbf{H}^k$ 。但  $\mathbf{H}^{k-m}$  无法显式求出，所以要找近似矩阵  $\hat{\mathbf{H}}^{k-m}$ ，一般来说这个矩阵的结构要比较简单。

实际上， $\mathbf{H}^k$  的显式形式根本无需计算。只要计算  $\mathbf{H}^k \nabla f(\mathbf{x}^k)$  即可。可以通过这个算法巧妙地直接计算  $\mathbf{H}^k \nabla f(\mathbf{x}^k)$ 。

---

**Algorithm 1** L-BFGS 双循环递归算法

---

```

1: 初始化  $\mathbf{q} \leftarrow \nabla f(\mathbf{x}^k)$ 
2: for  $i=k-1, k-2, \dots, k-m$  do
3:   计算并且保存  $\alpha_i \leftarrow \rho_i(\mathbf{s}^i)^\top \mathbf{q}$ 
4:   更新  $\mathbf{q} \leftarrow \mathbf{q} - \alpha_i \mathbf{y}^i$ 
5: end for
6: 初始化  $\mathbf{r} \leftarrow \hat{\mathbf{H}}^{k-m} \mathbf{q}$ ，其中  $\hat{\mathbf{H}}$  是近似矩阵
7: for  $i=k-m, k-m+1, \dots, k-1$  do
8:    $\beta \leftarrow \rho_i(\mathbf{y}^i)^\top \mathbf{r}$ 
9:    $\mathbf{r} \leftarrow \mathbf{r} + (\alpha_i - \beta) \mathbf{s}^i$ 
10: end for
11: 输出  $\mathbf{r} = \mathbf{H}^k \nabla f(\mathbf{x}^k)$ 

```

---

这个算法的时间复杂度是  $\mathcal{O}(mn)$ ，空间复杂度是  $\mathcal{O}(m)$ ，是一个非常高效的算法。

最后，近似矩阵  $\hat{\mathbf{H}}^{k-m}$  的取值可以是

$$\hat{\mathbf{H}}^{k-m} = \frac{(\mathbf{s}^{k-1})^\top \mathbf{y}^{k-1}}{(\mathbf{y}^{k-1})^\top \mathbf{y}^{k-1}}$$

## 5.5 DFP 公式

在 BFGS 方法中，使用割线方程  $\mathbf{s}^k = \mathbf{H}^{k+1} \mathbf{y}^k$  来推导基于  $\mathbf{H}^k$  的秩二修正的拟牛顿修正，可以得到基于  $\mathbf{H}^k$  的拟牛顿矩阵更新

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

并且同样可以求出其对称的  $\mathbf{B}^k$  格式。

## 5.6 各种拟牛顿方法的评论

1. 拟牛顿法是 **全局收敛** 的，在一定条件下，收敛速度是 **Q-超线性收敛** 的。
2. LBFGS 是应用最广泛的拟牛顿类方法。

## 6 共轭方向法

### 6.1 定义

设  $\mathbf{G}^{n \times n}$  是一个 **正定阵**， $\{\mathbf{d}^0, \mathbf{d}^1, \mathbf{d}^2, \dots, \mathbf{d}^k\}$  如果任选其中两个，都满足  $(\mathbf{d}^i)^\top \mathbf{G} \mathbf{d}^j = 0, (i \neq j)$ ，则称呼  $\mathbf{d}^0, \mathbf{d}^1, \mathbf{d}^2, \dots, \mathbf{d}^k$  是  $\mathbf{G}$ -共轭的。

共轭是正交概念的推广。当  $\mathbf{G} = \mathbf{I}$  的时候，共轭即为正交。

### 6.2 方法流程

---

**Algorithm 2** 共轭方向法 (类)

---

```

1:  $\mathbf{G}$  是正定的，初始点  $\mathbf{x}^0$ ， $\mathbf{g}^0 \leftarrow \nabla f(\mathbf{x}^0)$ ，寻找  $\mathbf{d}^0$  使得  $(\mathbf{g}^0)^\top \mathbf{d}^0 < 0$ 。  $k \leftarrow 0$ 
2:  $\alpha_k$  是精确一维搜索的步长，即  $\alpha_k \leftarrow \arg \min_{\alpha > 0} f(\mathbf{x}^k + \alpha \mathbf{d}^k)$ 
3: 迭代点  $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha_k \mathbf{d}^k$ 
4: 寻找和之前迭代方向都共轭的方向  $\mathbf{d}^{k+1}$ ，也就是满足  $(\mathbf{d}^{k+1})^\top \mathbf{G} \mathbf{d}^j = 0, j = 0, 1, \dots, k$ 
5:  $k \leftarrow k + 1$ ，寻找新的步长。

```

---



在之前的方法中， $\mathbf{d}^{k+1} = -\nabla f(\mathbf{x}^{k+1})$  或者是类似的形式。共轭方向法则根据历史取过的方向来寻找新的方向。

### 6.3 共轭方法的评价

1. 如果执行**精确的一维搜索**，那么该算法具有二次终止性，也就是说用这种方法处理二次正定函数的规划问题的时候，算法通过有限轮迭代可以找到最优解。