基于线搜索的最优化算法

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

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ψ在$\mathbf{x}^{(k)}$点处的下降方向作为搜索方向;

(c) 确定步长因子α_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 $\min_{\mathbf{x} \in \mathbf{R}^n}$，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 α_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：

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

在这个式子中，一阶展开为

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

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

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

这也是一个二次型，且由于 Hessian 矩阵是正定对称的，所以 $(G_k^{-1})^\top = G_k^{-1}$，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 α_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，d^k 是函数在 x^k 点处二次展开的极小点。

2.3 确定步长因子 α

一般来说，α 肯定是要尽可能的大，这样下降就会比较快。但如果 α 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 φ 满足 单峰性 ，所以就需要从区间 $[0, \bar{\alpha}_k]$ 中寻找一个可以接受的步长 α 作为步长。比如通过 Glodstein 条件就可以快速找到一个令人满意的值。

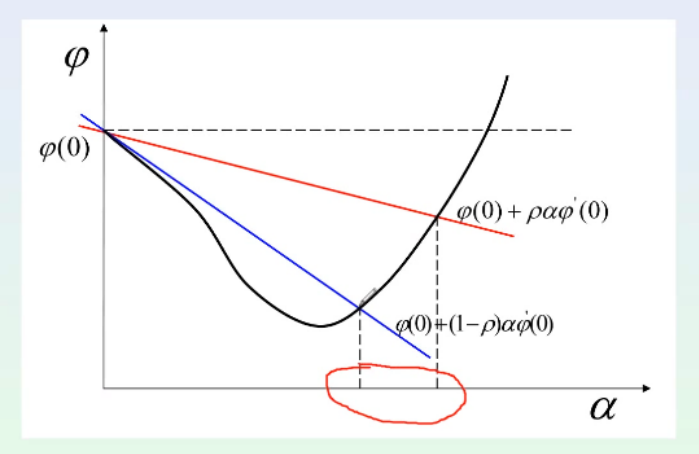
Goldstein 条件

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

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

上式可以保证 α 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 α 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 α 无线接近于 0，防止下降太慢。



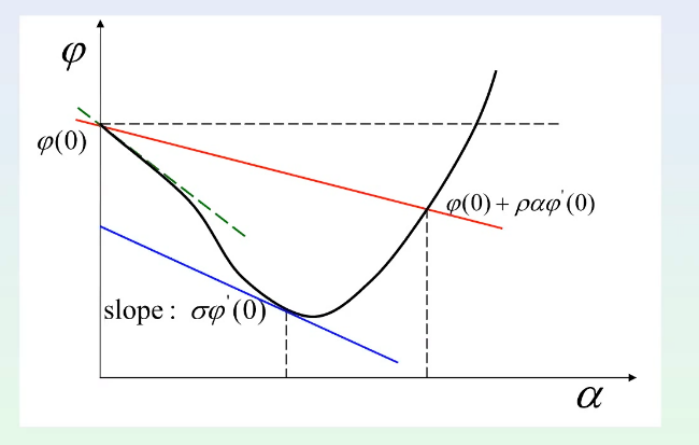
这样，α 就落在图中的红圈内部。

显然，从图中就可以看出 Glodstein 条件的问题。在图中，最好的 α 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 α 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 α 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 α 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

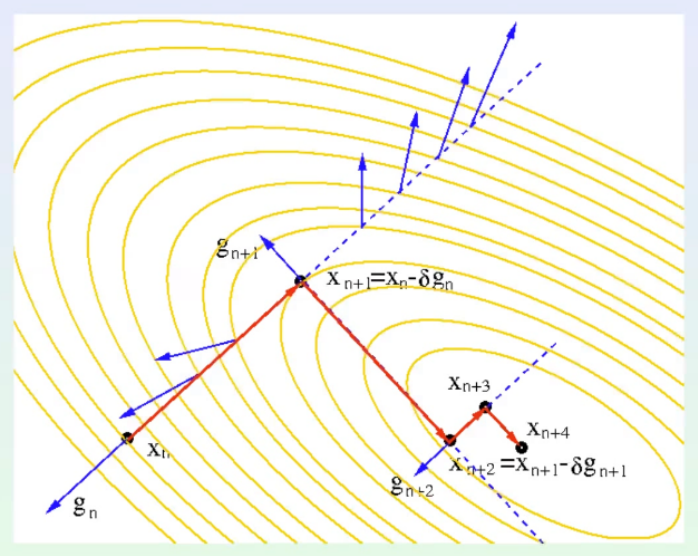
指初始点选择的范围大小。以上的方法都满足全局收敛的要求。

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。

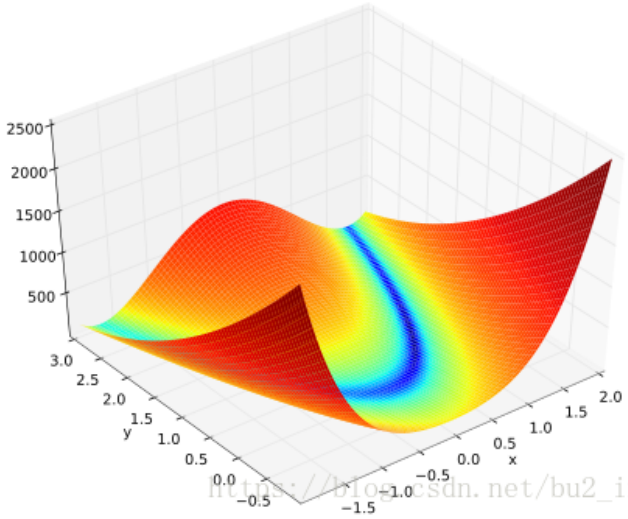
在图中，$\mathbf{g}^{k+1}$ 和 $\mathbf{g}^k$ 是相互垂直的。这是因为图中的线搜索是 “精确的一维搜索”，那么这两个梯度就是互相垂直的。呈现如图的情况。如果不是精确的一维搜索，则有可能更快也可能变慢。

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

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

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

$$\mathbf{s} = -G_k^{-1}\mathbf{g}^{(k)}$$

将 $\mathbf{s}$ 作为搜索方向，就称为**牛顿方向**。

对于正定二次函数

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

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

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

使用 $\mathbf{x}^\star = G^{-1}\mathbf{c}$ 表示问题的最优解。如果 $\mathbf{x}^{(0)} \neq \mathbf{x}^\star$，那么取步长 $\alpha_0 = 1$，就有 $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = \mathbf{x}^\star$，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

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - G_k^{-1}\mathbf{g}^{(k)}$$

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 $G_k\mathbf{d} = -\mathbf{g}^{(k)}$，获得牛顿方向 $\mathbf{d}^{(k)}$

4. 采用一维搜索确定 α_k，更新 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 $\mathsf{d}^{(k)}$ 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 $\alpha_k = 1$

Goldstein & Price

$$\mathsf{d}^{(k)} = \begin{cases} -G_k^{-1}\mathsf{g}^{(k)}, & if\ \cos\theta_k > \eta \\ -\mathsf{g}^{(k)}, & otherwise \end{cases}$$

也就是根据 θ 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

$$(G_k + \mu_k I)\mathsf{d}^{(k)} = -\mathsf{g}^{(k)}$$

L-M 方法的优势是，在阻尼牛顿法中，需要 $G_k\mathsf{d} = -\mathsf{g}^{(k)}$，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 μ_k 调节倾向于牛顿方向还是负梯度方向。μ_k 取值可以大于 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 d，满足

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

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。

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 。

构造近似矩阵有不同的方法，因此拟牛顿法也有多种类型。