优化方法读书笔记

1.1 问题定义

1.1.1 优化问题定义

最优化问题数学上定义，最优化问题的一般形式为

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in X \end{aligned} \tag{1}$$

其中的$x \in R^n$是自变量，f(x)是目标函数，$X \subset R^n$为约束集或者说可行域。

可行域这个东西，有等式约束，也有不等式约束，或者没有约束，这次的精力主要还是集中在无约束的优化问题上，如果有精力，再讨论有约束的。

1.1.2 最优化方法定义

优化问题的解法叫最优化方法。

最优化方法通常采用迭代的方法求它的最优解，其基本思想是：给定一个初始点$x_0 \in R^n$，按照某一迭代规则产生一个点列$\{x_k\}$，使得当$\{x_k\}$是有穷点列时，其最后一个点是最优化模型问题的最优解，当$\{x_k\}$是无穷点列时，它有极限点，且其极限点是最优化模型问题的最优解。一个好的算法应具备的典型特征为：迭代点x_k能稳定地接近局部极小值点x^*的邻域，然后迅速收敛于x^*。当给定的某个收敛准则满足时，迭代即终止。

好吧，上面是一堆描述，来点定义，设x_k为第 k 次迭代点，d_k第 k 次搜索方向，α_k为第 k 次步长因子，则第 k 次迭代为

$$x_{k+1} = x_k + \alpha_k d_k \tag{2}$$

然后后面的工作几乎都在调整$\alpha_k d_k$这个项了，不同的步长α_k和不同的搜索方向d_k分别构成了不同的方法。

当然步长α_k和搜索方向d_k是得满足一些条件，毕竟是求最小值，不能越迭代越大了，下面是一些条件

$$\nabla f(x_k)^T d_k < 0 \tag{3}$$

或者

$$f(x_k + \alpha_k d_k) < f(x_k) \tag{4}$$

式子（3）的意义是搜索方向必须跟梯度方向（梯度也就是$\nabla f(x_k)$）夹角大于 90 度，也就是基本保证是向梯度方向的另外一边搜索，至于为什么要向梯度方向的另外一边搜索，就是那样才能保证是向目标函数值更小的方向搜索的，因为梯度方向一般是使目标函数增大的（不增大的情况是目标函数已经到达极值）。

式子（4）的意义就很明显了，迭代的下一步的目标函数比上一步小。

最优化方法的基本结构为：

给定初始点x_0，

（a） 确定搜索方向d_k，即按照一定规则，构造目标函数 f 在x_k点处的下降方向作为搜索方向。

（b） 确定步长因子α_k，使目标函数值有某种意义的下降。

（c） 令

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

若x_{k+1}满足某种终止条件，则停止迭代，得到近似最优解x_{k+1}，否则，重复以上步骤。能不能收敛到最优解是衡量最优化算法的有效性的一个重要方面。

1.1.3 收敛速度简介

收敛速度也是衡量最优化方法有效性的一个重要方面。

设算法产生的迭代点列$\{x_k\}$在某种范数意义下收敛，即

$$\lim_{k\to\infty}\|x_k - x^*\| = 0 \qquad (5)$$

若存在实数$\alpha > 0$（这个α不是步长）以及一个与迭代次数 k 无关的常数$q > 0$，使得

$$\lim_{k\to\infty}\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^\alpha} = q \qquad (6)$$

则称算法产生的迭代点列$\{x_k\}$具有$Q-\alpha$阶收敛速度。特别地，常用的说法有以下几种。

（a） 当$\alpha = 0$，$q > 0$时，迭代点列$\{x_k\}$叫做具有 Q-线性收敛速度；

（b） 当$1 < \alpha < 2$，$q > 0$时或$\alpha = 1$，$q = 0$时，迭代点列$\{x_k\}$叫做具有 Q-超线性收敛速度；

（c） 当$\alpha = 2$时，迭代点列$\{x_k\}$叫做具有 Q-二阶收敛速度；

还有一种叫做 R-收敛速度，不讨论了，有兴趣可以查阅相关资料。

一般认为，具有超线性收敛速度和二阶收敛速度的方法是比较快速的。但是对于一个算法，收敛性和收敛速度的理论结果并不保证算法在实际执行时一定有好的实际计算特性。一方面是由于这些结果本身并不能保证方法一定有好的特性，另一方面是由于算法忽略了计算过程中十分重要的舍入误差的影响。

1.2 步长确定

1.2.1 一维搜索

如前面的讨论，优化方法的关键是构造搜索方向d_k和步长因子α_k，这一节就讨论如何确定步长。

设

$$\varphi(\alpha) = f(x_k + \alpha d_k)$$

这样，从x_k出发，沿搜索方向d_k，确定步长因子α_k，使得

$$\varphi(\alpha_k) < \varphi(0)$$

的问题就是关于α的一维搜索问题。注意这里是假设其他的x_k和d_k都确定了的情况下做的搜索，要搜索的变量只有α。

如果求得的α_k，使得目标函数沿搜索方向d_k达到最小，即达到下面的情况

$$f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k)$$

或者说

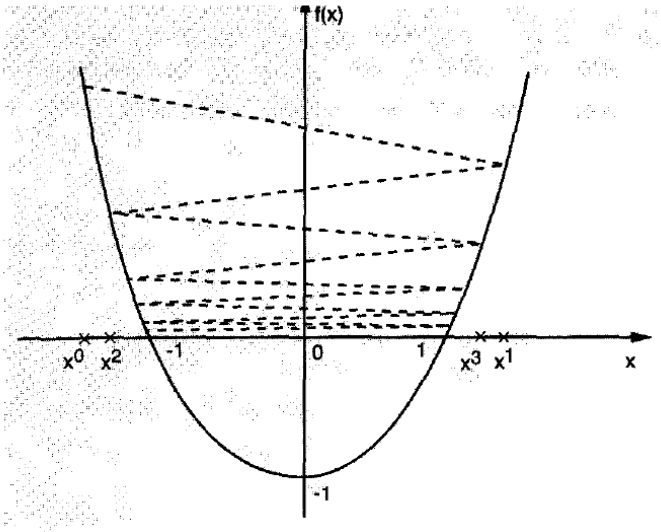
$$\varphi(\alpha_k) = \min_{\alpha>0} \varphi(\alpha)$$

如果能求导这个最优的α_k，那么这个α_k就称为最优步长因子，这样的搜索方法就称为最优一维搜索，或者精确一维搜索。

但是现实情况往往不是这样，实际计算中精确的最优步长因子一般比较难求，工作量也大，所以往往会折中用不精确的一维搜索。

不精确的一维搜索，也叫近似一维搜索。方法是选择α_k，使得目标函数 f 得到可接受的下降量，即使得下降量$\mathrm{f}(x_k) - \mathrm{f}(x_k + \alpha_k d_k) > 0$是用户可接受的。也就是，只要找到一个步长，使得目标函数下降了一个比较满意的量就可以了。

为啥要选步长？看下图，步长选不好，方向哪怕是对的，也是跑来跑去，不往下走，二维的情况简单点，高维的可能会弄出一直原地不动的情况来。



一维搜索的主要结构如下：1）首先确定包含问题最优解的搜索区间，再采用某种分割技术或插值方法缩小这个区间，进行搜索求解。

当然这个搜索方法主要是适应单峰区间的，就是类似上面的那种，只有一个谷底的。

1.2.1.1 确定搜索区间

确定搜索区间一般用进退法，思想是从某一点出发，按某步长，确定函数值呈现“高-低-高”的三个点，一个方向不成功，就退回来，沿相反方向寻找。

下面是步骤。

(1) 选取初始数据。$\alpha_0 \in [0,\infty), h_0 > 0$，加倍系数 t>1(t 一般取 2)，计算$\varphi(\alpha_0)$，k:=0.

(2) 比较目标函数值。令$\alpha_{k+1} = \alpha_k + h_k$，计算$\varphi_{k+1} = \varphi(\alpha_{k+1})$，若$\varphi_{k+1} < \varphi_k$，转步 3，否则转步 4.

(3) 加大探索步长，令$h_{k+1} := t h_k$，$\alpha := \alpha_k$，$\alpha_k := \alpha_{k+1}$，$\varphi_k := \varphi_{k+1}$，$k := k+1$，转步 2.

(4) 反向探索。若 k=0，转换探索方向，令$h_k := -h_k$，$\alpha_k := \alpha_{k+1}$，转步 2；否则，停止迭代，令$a = \min\{\alpha, \alpha_{k+1}\}$，$b = \max\{\alpha, \alpha_{k+1}\}$，输出[a,b]。

1.2.1.2 搜索求解

搜索求解的话，0.618 法简单实用。虽然 Fibonacci 法，插值法等等虽然好，复杂，就不多说了。下面是 0.618 法的步骤。

(1) 选取初始数据。确定初始搜索区间$[a_1, b_1]$和精度要求$\delta > 0$。计算最初两个试探点λ_1，μ_1，

$$\lambda_1 = \alpha_1 + 0.382(b_1 - a_1),$$
$$\mu_1 = \alpha_1 + 0.618(b_1 - a_1)$$

计算$\varphi(\lambda_1)$和$\varphi(\mu_1)$，令 k=1.

(2) 比较函数值。$\varphi(\lambda_k) > \varphi(\mu_k)$，则转步 3；若$\varphi(\lambda_k) \le \varphi(\mu_k)$，则转步 4。

(3) 若$b_k - \lambda_k \le \delta$，则停止计算，输出$\mu_k$。否则令

$a_{k+1} \coloneqq \lambda_k$，$b_{k+1} \coloneqq b_k$，$\lambda_{k+1} := \mu_k$，$\varphi(\lambda_{k+1}) \coloneqq \varphi(\mu_k)$，

$\mu_{k+1} \coloneqq a_{k+1} + 0.618(b_{k+1} - a_{k+1})$.

计算$\varphi(\mu_{k+1})$，转步 5。

(4) 若$\mu_k - a_k \le \delta$，则停止计算，输出λ_k，否则，令

$a_{k+1} \coloneqq a_k$，$b_{k+1} \coloneqq \mu_k$，$\mu_{k+1} := \lambda_k$，$\varphi(\mu_{k+1}) \coloneqq \varphi(\lambda_k)$，

$\mu_{k+1} \coloneqq a_{k+1} + 0.382(b_{k+1} - a_{k+1})$.

计算$\varphi(\lambda_{k+1})$，转步 5。

(5) $k \coloneqq k + 1$，转步 2。

普通的0.618法要求一维搜索的函数是单峰函数，实际上遇到的函数不一定是单峰函数，这时，可能产生搜索得到的函数值反而大于初始区间端点出函数值的情况。有人建议每次缩小区间是，不要只比较两个内点处的函数值，而是比较两内点和两端点处的函数值。当左边第一个或第二个点是这四个点中函数值最小的点是，丢弃右端点，构成新的搜索区间；否则，丢弃左端点，构成新的搜索区间，经过这样的修改，算法会变得可靠些。步骤就不列了。

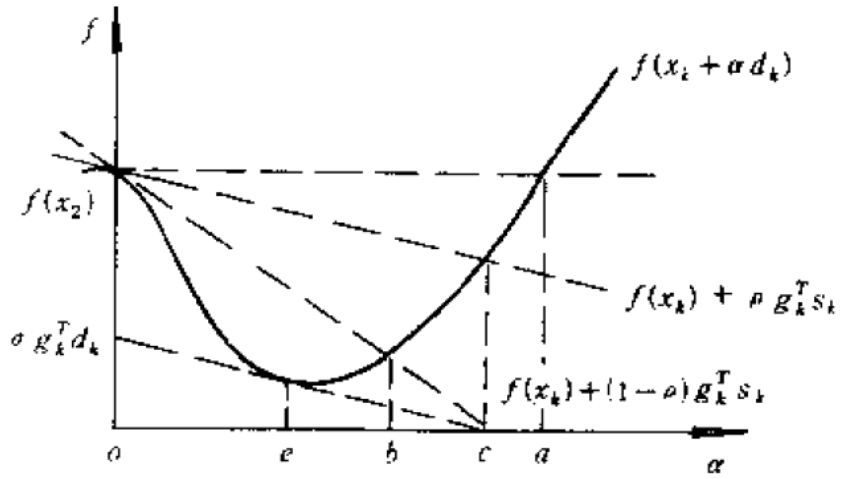
1.2.2 不精确一维搜索方法

一维搜索过程是最优化方法的基本组成部分，精确的一维搜索方法往往需要花费很大的工作量。特别是迭代点远离问题的解时，精确地求解一个一维子问题通常不是十分有效的。另外，在实际上，很多最优化方法，例如牛顿法和拟牛顿法，其收敛速度并不依赖于精确一维搜索过程。因此，只要保证目标函数 f(x)在每一步都有满意的下降，这样就可以大大节省工作量。

有几位科学家 Armijo(1966)和 Goldstein(1965)分别提出了不精确一维搜索过程。设

$$J = \{\alpha > 0 | f(x_k + \alpha d_k) < f(x_k)\} \qquad (2.5.1)$$

是一个区间。看下图



在图中，区间 J=(0,a)。为了保证目标函数单调下降，同时要求 f 的下降不是太小（如果 f 的下降太小，可能导致序列{f(x_k)}的极限值不是极小值），必须避免所选择的α太靠近区间 j 的短短。一个合理的要求是

$$\mathrm{f}(x_k+s_k)\le \mathrm{f}(x_k)+\rho g_k^T s_k \qquad (2.5.2)$$

$$\mathrm{f}(x_k+s_k)\ge \mathrm{f}(x_k)+(1-\rho) g_k^T s_k \qquad (2.5.3)$$

其中$0<\rho<\frac{1}{2}$，$s_k=\alpha_k d_k$。满足(2.5.2)要求的α_k构成区间$J_1=(0,c]$，这就扔掉了区间 J 右端附件的点。但是为了避免α太小的情况，又加上了另一个要求(2.5.3)，这个要求扔掉了区间 J 的左端点附件的点。看图中$\mathrm{f}(x_k)+\rho g_k^T s_k$和$\mathrm{f}(x_k)+(1-\rho)g_k^T s_k$两条虚线夹出来的区间$J_2=[b,c]$就是满足条件(2.5.2)和(2.5.3)的搜索区间，称为可接受区间。条件(2.5.2)和(2.5.3)称为 Armijo-Goldstein 准则。无论用什么办法得到步长因子α，只要满足条件(2.5.2)和(2.5.3)，就可以称它为可接受步长因子。

其中$\rho<\frac{1}{2}$这个要求是必须的，因为不用这个条件，可能会影响牛顿法和拟牛顿法的超线性收敛。

在图中可以看到一种情况，极小值 e 也被扔掉了，为了解决这种情况，Wolfe-Powell 准则给出了一个更简单的条件代替

$$g_{k+1}^T d_k\ge \sigma g_k^T d_k,\ \sigma\in(\rho,1) \qquad (2.5.4)$$

其几何解释是在可接受点处切线的斜率$\varphi'(\alpha_k)$大于或等于初始斜率的σ倍。准则(2.5.2)和(2.5.4)称为 Wolfe-Powell 准则，其可接受区间为$J_3=[e,c]$。

要求$\rho<\sigma<1$是必要的，它保证了满足不精确线性搜索准则的步长因子α_k的存在，不这么弄，可能$\sigma g_k^T d_k$这个虚线会往下压，没有交点，就搞不出一个区间来了。

一般地，σ值越小，线性搜索越精确。取$\sigma=0.1$，就得到一个相当精确的线性搜索，而取$\sigma=0.9$，则得到一个相当弱的线性搜索。不过σ值越小，工作量越大。所以不精确线性搜索不要求过小的σ，通常可取$\rho=0.1$，$\sigma=0.4$。

下面就给出 Armijo-Goldstein 准则和 Wolfe-Powell 准则的框图。

从算法框图中可以看出，两种方法是类似的，只是在准则不成立，需要计算新的σ时，一个利用了简单的求区间中点的方法，另一个采用了二次插值方法。

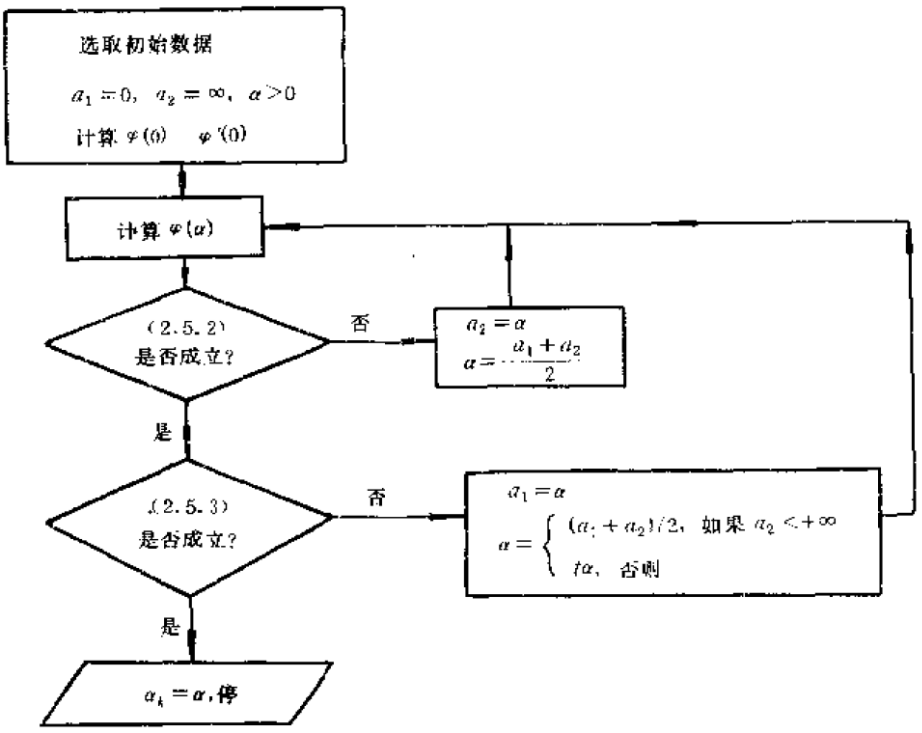


图 2.5.2　Armijo-Goldstein 不精确线性搜索法框图

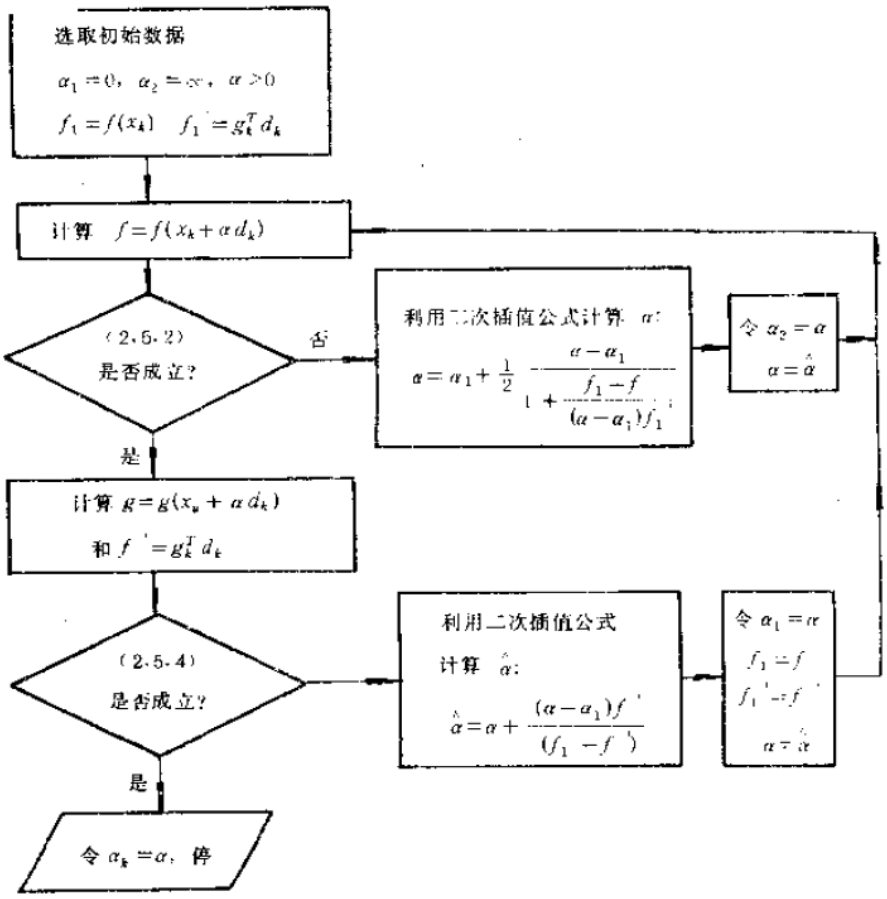


图 2.5.3　Wolfe-Powell 不精确一维搜索法框图

算法步骤只给出 Armijo-Goldstein 不精确一维搜索方法的，下面就是

(1) 选取初始数据，在搜索区间$[0,+\infty)$（或者$[0,\alpha_{max}]$）中取定初始点α_0，计算$\varphi(0)$，$\varphi'(0)$，给出$\rho \in (0,\frac{1}{2})$，t>1。令$a_0 \coloneqq 0$，$b_0 \coloneqq +\infty$（或α_{max}），k:=0。

(2) 检验准则(2.5.2)。计算$\varphi(\alpha_k)$，若

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

转步 3；否则，令$a_{k+1} \coloneqq a_k$，$b_{k+1} \coloneqq \alpha_k$，转步 4。

(3) 检验准则(2.5.3)。若

$$\varphi(\alpha_k) \geq \varphi(0) + (1-\rho)\alpha_k\varphi'(0)$$

停止迭代，输出α_k；否则，令$a_{k+1} \coloneqq \alpha_k$，$b_{k+1} \coloneqq b_k$。若$b_{k+1} < +\infty$，转步 4；否则$a_{k+1} \coloneqq t\alpha_k$，k:=k+1，转步 2。

(4) 选取新的探索点，取

$$\alpha_{k+1} \coloneqq \frac{a_{k+1} + b_{k+1}}{2}$$

令 k:=k+1，转步 2.

好了，说到这，确定步长的方法也说完了，其实方法不少，实际用到的肯定是最简单的几种，就把简单的几种提了一下，至于为什么这样，收敛如何，证明的东西大家可以去书中慢慢看。

1.3 方向确定

1.3.1 最速下降法

最速下降法以负梯度方向作为最优化方法的下降方向，又称为梯度下降法，是最简单实用的方法。

1.3.1.1 算法步骤

下面是步骤。

(1) 给出$x_0 \in R^n$，$0 \leq \epsilon \leq 1$，k:=0；

(2) 计算$d_k = -g_k$；如果$\|g_k\| \leq \epsilon$，则停止；

(3) 由一维搜索确定步长因子α_k，使得

$$f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k)$$

(4) 计算$x_{k+1} = x_k + \alpha_k d_k$；

(5) K:=k+1，转步 2

其中$g_k = \nabla f(x_k) \neq 0$，是梯度。

看个例子。

$$\min f(x) = 2x_1^2 + x_2^2.$$

$$\nabla f(x) = \begin{pmatrix} 4x_1 \\ 2x_2 \end{pmatrix}, \quad \nabla^2 f(x) = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix} = A$$

$$x^{(0)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \nabla f(x^{(0)}) = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad p^{(0)} = -\nabla f(x^{(0)}) = \begin{pmatrix} 0 \\ -2 \end{pmatrix}$$

$$\alpha_0 = \frac{\left\|\nabla f(x^{(0)})\right\|^2}{(p^{(0)})^T A p^{(0)}} = \frac{4}{8} = \frac{1}{2}$$

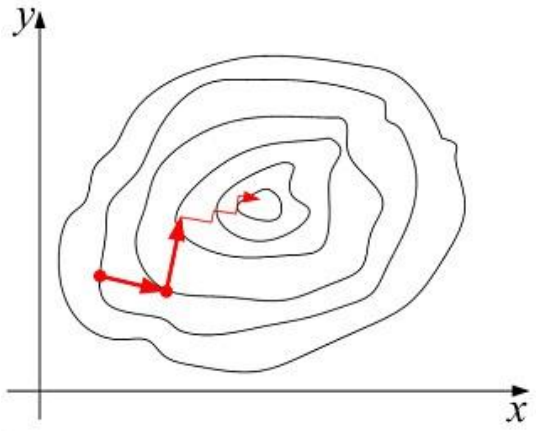
$$x^{(1)} = x^{(0)} + \alpha_0 p^{(0)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

这个选步长的方法是对二次函数下的特殊情况，是比较快而且好的显式形式，说明步长选得好，收敛很快。

1.3.1.2 缺点

数值实验表明，当目标函数的等值线接近一个圆（球）时，最速下降法下降较快，当目标函数的等值线是一个扁长的椭球是，最速下降法开始几步下降较快，后来就出现锯齿线性，下降就十分缓慢。原因是一维搜索满足$g_{k+1}^T d_k = 0$，即$g_{k+1}^T g_k = d_{k+1}^T d_k = 0$，这表明在相邻两个迭代点上函数 f(x)的两个梯度繁星是互相直交（正交）的。即，两个搜索方向互相直交，就产生了锯齿性质。当接近极小点时，步长越小，前进越慢。

下图是锯齿的一个图。



1.3.2 牛顿法

1.3.2.1 算法思想和步骤

牛顿法的基本思想是利用目标函数的二次 Taylor 展开，并将其极小化。

设 f(x)是二次可微实函数，$x_k \in R^n$，Hesse 矩阵$\nabla^2 f(x_k)$正定。在x_k附件用二次 Taylor

展开近似 f，

$$\mathrm{f}(x_k+s)\approx q^{(k)}(s)=\mathrm{f}(x_k)+\nabla\mathrm{f}(x_k)^Ts+\frac{1}{2}s^T\nabla^2 f(x_k)s$$

其中，$s=\mathrm{x}-x_k$，$q^{(k)}(s)$为 f(x)的二次近似。将上式右边极小化，便得

$$x_{k+1}=x_k-[\nabla^2 f(x_k)]^{-1}\nabla\mathrm{f}(x_k)$$

这就是牛顿迭代公式。在这个公式中，步长因子$\alpha_k=1$。令$G_k=\nabla^2 f(x_k)=H(f)$，$g_k=\nabla\mathrm{f}(x_k)$，则上面的迭代式也可以写成

$$x_{k+1}=x_k-\mathrm{G}_k^{-1}g_k$$

其中的 Hesse 矩阵的形式如下。

$$H(f)=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\partial x_n}\\ \frac{\partial^2 f}{\partial x_2\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_n\partial x_1} & \frac{\partial^2 f}{\partial x_n\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}.$$

一个例子如下。

$$\min f(x)=2x_1^{\,2}+x_2^{\,2},\quad x^{(0)}=(1,2)^T$$

$$\nabla f(x)=\begin{pmatrix}4x_1\\2x_2\end{pmatrix},\quad \nabla^2 f(x)=\begin{pmatrix}4&0\\0&2\end{pmatrix},\quad \nabla^2 f(x)^{-1}=\begin{pmatrix}\frac{1}{4}&0\\0&\frac{1}{2}\end{pmatrix}.$$

$$p^{(0)}=-\nabla^2 f(x^{(0)})^{-1}\nabla f(x^{(0)})=-\begin{pmatrix}\frac{1}{4}&0\\0&\frac{1}{2}\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}=\begin{pmatrix}-1\\-2\end{pmatrix}$$

$$x^{(1)}=x^{(0)}+p^{(0)}=\begin{pmatrix}1\\2\end{pmatrix}+\begin{pmatrix}-1\\-2\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}.$$

对于正定二次函数，牛顿法一步就可以得到最优解。

对于非二次函数，牛顿法并不能保证经过有限次迭代求得最优解，但由于目标函数在极小点附近近似于二次函数，所以当初始点靠近极小点时，牛顿法的收敛速度一般是快的。

当初始点远离最优解，G_k不一定正定，牛顿方向不一定是下降方向，其收敛性不能保证，这说明步长一直是 1 是不合适的，应该在牛顿法中采用某种一维搜索来确定步长因子。但是要强调一下，仅当步长因子$\{\alpha_k\}$收敛到 1 时，牛顿法才是二阶收敛的。

这时的牛顿法称为阻尼牛顿法，步骤如下。

(1) 选取初始数据，取初始点$x_0\in R^n$，终止误差$\epsilon>0$，令 k:=0。

(2) 计算g_k。若$\|g_k\|<\epsilon$，停止迭代，输出x_k。

(3) 解方程组构造牛顿方向，即解$\mathrm{G}_k d_k=-g_k$，求出d_k，或者用某种方法求得G_k^{-1}，直

接令$d_k = -\mathrm{G}_k^{-1} g_k$。

(4) 进行一维搜索，求α_k使得

$$\mathrm{f}(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k)$$

令$x_{k+1} = x_k + \alpha_k d_k$，k:=k+1 转步 2.

下面看个例子。

$$\min f(x) = (1-x_1)^2 + 2(x_2 - x_1^{\,2})^2$$

- iteration 1 (0,0)

$$\nabla f(x) = \begin{pmatrix} 8x_1^{\,3} + 2x_1 - 8x_1x_2 - 2 \\ 4x_2 - 4x_1^{\,2} \end{pmatrix}, \quad \nabla^2 f(x) = \begin{pmatrix} 24x_1^{\,2} + 2 - 8x_2 & -8x_1 \\ -8x_1 & 4 \end{pmatrix}$$

$$\nabla f(x^{(0)}) = \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \quad \nabla^2 f(x^{(0)}) = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}, \quad p^{(0)} = -\nabla^2 f(x^{(0)})^{-1} \nabla f(x^{(0)}) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\min_{\alpha\geq 0} f(x^{(0)} + \alpha p^{(0)}) = \min_{\alpha\geq 0} [(1-\alpha)^2 + 2\alpha^4]$$

$$x^{(1)} = x^{(0)} + \alpha_0 p^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix}$$

- iteration 2

$$\nabla f(x^{(1)}) = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad \nabla^2 f(x^{(1)}) = \begin{pmatrix} 8 & -4 \\ -4 & 4 \end{pmatrix}, \quad \nabla^2 f(x^{(1)})^{-1} = \frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

$$p^{(1)} = -\nabla^2 f(x^{(1)})^{-1} \nabla f(x^{(1)}) = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \end{pmatrix}$$

$$\min_{\alpha\geq 0} f(x^{(1)} + \alpha p^{(1)}) = \min_{\alpha\geq 0} \frac{1}{128}[8(2-\alpha)^2 + (2-\alpha)^4]$$

$$x^{(2)} = x^{(1)} + \alpha_1 p^{(1)} = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} + 2\begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\nabla f(x^{(2)}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \left\|\nabla f(x^{(2)})\right\| = 0 < \varepsilon, \quad \nabla^2 f(x^{(2)}) = \begin{pmatrix} 18 & -8 \\ -8 & 4 \end{pmatrix} > 0$$

这样的牛顿法是总体收敛的。

1.3.2.2 缺点

牛顿法面临的主要困难是 Hesse 矩阵G_k不正定。这时候二次模型不一定有极小点，甚至没有平稳点。当G_k不正定时，二次模型函数是无界的。

为了克服这种困难，有多种方法，常用的方法是使牛顿方向偏向最速下降方向$-g_k$。具体做法是将 Hesse 矩阵G_k改变成$\mathrm{G}_k + \nu_k I$，其中$\nu_k > 0$，使得$\mathrm{G}_k + \nu_k I$正定。ν_k一般希望是比较小，最好是刚刚好能使$\mathrm{G}_k + \nu_k I$正定。

1.3.3 拟牛顿法

牛顿法在实际应用中需要存储二阶导数信息和计算一个矩阵的逆，这对计算机的时间和空间要求都比较高，也容易遇到不正定的 Hesse 矩阵和病态的 Hesse 矩阵，导致求出来的逆很古怪，从而算法会沿一个不理想的方向去迭代。

有人提出了介于最速下降法与牛顿法之间的方法。一类是共轭方向法，典型的是共轭梯度法，还有拟牛顿法。

其中拟牛顿法简单实用，这里就特地介绍，其他方法感兴趣的读者可以去看相关资料。

1.3.3.1 算法思想和步骤

牛顿法的成功的关键是利用了 Hesse 矩阵提供的曲率信息。但是计算 Hesse 矩阵工作量大，并且有些目标函数的 Hesse 矩阵很难计算，甚至不好求出，这就使得仅利用目标函数的一阶导数的方法更受欢迎。拟牛顿法就是利用目标函数值 f 和一阶导数 g（梯度）的信息，构造出目标函数的曲率近似，而不需要明显形成 Hesse 矩阵，同时具有收敛速度快的特点。

设f: $R^n \to R$在开集$\mathcal{D} \subset \mathrm{R}^n$上二次可微，f 在$x_{k+1}$附近的二次近似为

$$\mathrm{f(x)} \approx \mathrm{f}(x_{k+1}) + g_{k+1}^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T G_{k+1}(x - x_{k+1})$$

两边求导，得

$$\mathrm{g(x)} \approx g_{k+1} + G_{k+1}(x - x_{k+1})$$

令$x = x_k$，$s_k = x_{k+1} - x_k$，$y_k = g_{k+1} - g_k$，得

$$G_{k+1}^{-1} y_k \approx s_k$$

其中$g_k = \nabla f(x_k) \neq 0$，是梯度。那么，只要构造出 Hesse 矩阵的逆近似H_{k+1}满足这种上式就可以，即满足关系

$$\mathrm{H}_{k+1} y_k = s_k$$

这个关系就是拟牛顿条件或拟牛顿方程。

拟牛顿法的思想就是——用一个矩阵H_k去近似 Hesse 矩阵的逆矩阵，这样就避免了计算矩阵的逆。

当然H_k需要满足一些条件

(a) H_k是一个正定的矩阵

(b) 如果$\nabla^2 f(x_k)^{-1}$存在，则$\mathrm{H}_k \approx \nabla^2 f(x_k)^{-1}$。

(c) 初始正定矩阵H_0取定后，H_{k+1}应该由H_k递推给出，即$\mathrm{H}_{k+1} = \mathrm{H}_k + \mathrm{E}_k$；其中$\mathrm{E}_k$是修正矩阵，$\mathrm{H}_{k+1} = \mathrm{H}_k + \mathrm{E}_k$是修正公式。

常用而且有效的修正公式是 BFGS 公式，如下

$$H_{k+1} = \mathrm{H}_k + \left(1 + \frac{y_k^T \mathrm{H}_k y_k}{s_k^T y_k}\right)\frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T \mathrm{H}_k + \mathrm{H}_k y_k s_k^T}{s_k^T y_k}$$

下面给出 BFGS 公式下的拟牛顿法

(1) 选取初始数据，取初始点$x_0 \in R^n$，初始正定矩阵H_0，终止误差$\epsilon > 0$，令 k:=0；

(2) 如果$\|g_k\| \le \epsilon$，则停止，输出x_k作为近似最优解；否则计算$d_k = -H_k g_k$；

(3) 进行一维搜索，求α_k使得

$$f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k)$$

令$x_{k+1} = x_k + \alpha_k d_k$

(4) 计算$s_k = x_{k+1} - x_k$，$y_k = g_{k+1} - g_k$，和

$$H_{k+1} = H_k + \left(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}\right)\frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k}$$

(5) 令 k:=k+1，转步 2

在上述步骤中，初始正定矩阵H_0通常取为单位矩阵，即$H_0 = I$。这样，拟牛顿法的第一次迭代相当于一个最速下降迭代。

1.3.3.2 优缺点

与牛顿法相比，有两个优点：

(a) 仅需一阶导数

(b) 校正保持正定性，因而下降性质成立

(c) 无需计算逆矩阵，但具有超线性收敛速度

(d) 每次迭代仅需要$3n^2 + O(n)$次乘法计算

缺点是初始点距离最优解远时速度还是慢。

解决方法是，迭代前期用最速下降法进行迭代，得到一定解以后用拟牛顿法迭代。