



组合优化与凸优化

第5章 无约束最优化方法 (Unconstrained Optimization Methods)

刘绍辉

计算机科学与技术学院 哈尔滨工业大学

shliu@hit.edu.cn

2022年春季





### ◆ 使用导数和不使用导数的线搜索方法

- Dichotomous, Golden Section method, Fibonacci method
- Bisection search, Newton's method

### ◆ 多维搜索下使用导数和不使用导数的搜索方法

- The Cyclic coordinate method, Hooke and Jeeves, Rosenbrock's method
- The Steepest descent and The method of Newton

### ◆ 牛顿方法的变种: LM和信任域方法

- Levenberg-Marquardt, Trust Region Methods

### ◆ 共轭方向法: 拟牛顿法和共轭梯度法

- 目标函数如果是二次的, 有限步内可以收敛

### ◆ 次梯度优化方法

- 不可微目标函数中的Steepest Descent algorithm: 投影思想!

### ◆ 总结





## ◆ 实际的优化问题一般都有很多的约束，那为什么还要研究无约束最优化方法呢？

- 许多算法可以通过Lagrangian乘子法将有约束优化问题转化为一系列的无约束优化问题来求解，例如后续要介绍的Lagrangian对偶和鞍点最优性条件，又如后续要提到的惩罚和障碍函数法等
- 大多数方法都是通过找到一个方向，然后沿方向最小化来推进求解，这种线搜索方法实际上是无约束或简单约束的最小化问题
- 最后，有几种无约束最优化方法可以自然推广到有约束问题的求解上去

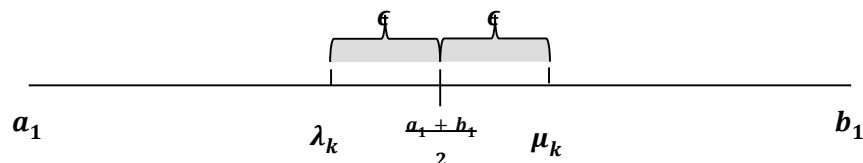


# 第5章 无约束最优化方法 (Unconstrained)



◆ 我们只介绍每种方法的基本思想，其收敛速度等的证明均略过

◆ Dichotomous方法



$$\lambda_k = \frac{a_k + b_k}{2} - \epsilon, \mu_k = \frac{a_k + b_k}{2} + \epsilon$$

◆ 若  $f(\lambda_k) < f(\mu_k)$ , 则  $a_{k+1} = a_k, b_{k+1} = \mu_k$ ;

◆ 否则,  $a_{k+1} = \lambda_k, b_{k+1} = b_k$

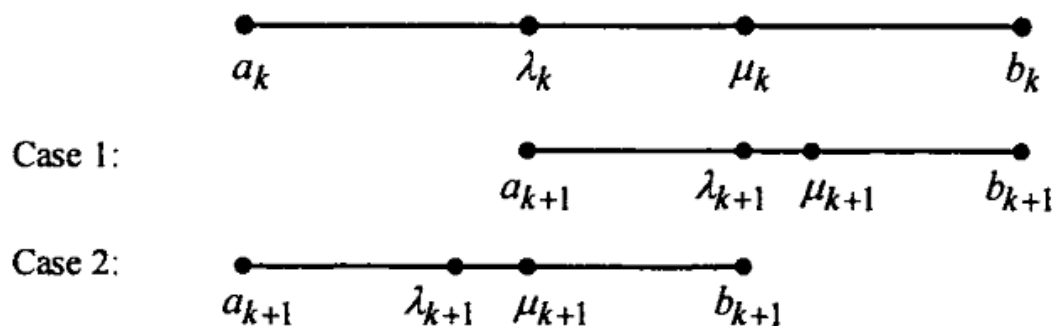
◆ 易见:  $b_{k+1} - a_{k+1} = \frac{1}{2^k} (b_1 - a_1) + 2\epsilon(1 - \frac{1}{2^k})$

◆ 从而可以根据精度要求来确定迭代次数，编程时可以预先确定需要申请的数组数目

# 第5章 无约束最优化方法-线搜索回顾



## ◆黄金分割(Golden Section)



◆若 $f(\lambda_k) > f(\mu_k)$ , 则 $a_{k+1} = \lambda_k$ ,  $b_{k+1} = b_k$ , 且 $\mu_k = \lambda_{k+1} = a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1}) = \lambda_k + (1 - \alpha)(b_k - \lambda_k)$

◆否则 $a_{k+1} = a_k$ ,  $b_{k+1} = \mu_k$ ,  $\lambda_k = \mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1}) = a_k + \alpha(\mu_k - a_k)$

◆易见:  $b_{k+1} - a_{k+1} = 0.618(b_k - a_k)$



# 第5章 无约束最优化方法-线搜索回顾

◆例:  $\text{Min } \lambda^2 + 2\lambda, s. t. -3 \leq \lambda \leq 5$

◆解: 区间长度为8, 因此采用黄金分割法求解时, 其前两个观察量

◆  $\lambda_1 = -3 + 0.382 \cdot 8 = 0.056, \mu_1 = -3 + 0.618 \cdot 8 = 1.944$ , 目标函数值  $f(\lambda_1) < f(\mu_1)$ , 因此新区间为  $[-3, 1.944]$

Iteration $k$	$a_k$	$b_k$	$\lambda_k$	$\mu_k$	$\theta(\lambda_k)$	$\theta(\mu_k)$
1	-3.000	5.000	0.056	1.944	0.115*	7.667*
2	-3.000	1.944	-1.112	0.056	-0.987*	0.115
3	-3.000	0.056	-1.832	-1.112	-0.308*	-0.987
4	-1.832	0.056	-1.112	-0.664	-0.987	-0.887*
5	-1.832	-0.664	-1.384	-1.112	-0.853*	-0.987
6	-1.384	-0.664	-1.112	-0.936	-0.987	-0.996*
7	-1.112	-0.664	-0.936	-0.840	-0.996	-0.974*
8	-1.112	-0.840	-1.016	-0.936	-1.000*	-0.996
9	-1.112	-0.936				



# 第5章 无约束最优化方法-线搜索回顾



## ◆ Fibonacci搜索(斐波那契搜索)

$$\text{◆ } \lambda_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}} (b_k - a_k), \mu_k = a_k + \frac{F_{n-k}}{F_{n-k+1}} (b_k - a_k)$$

◆ 若  $f(\lambda_k) > f(\mu_k)$ , 则  $a_{k+1} = \lambda_k, b_{k+1} = b_k$

◆ 否则:  $a_{k+1} = a_k, b_{k+1} = \mu_k$

◆ 计算后可验证

$$\text{➤ } b_{k+1} - a_{k+1} = b_k - \lambda_k = \frac{F_{n-k}}{F_{n-k+1}} (b_k - a_k)$$

◆ 注意: 除了第一次需要计算两次函数值外, 每次迭代只需要计算一次函数值





# 第5章 无约束最优化方法-线搜索回顾

◆ 例:  $\text{Min } \lambda^2 + 2\lambda, s. t. -3 \leq \lambda \leq 5$

◆ 解: 区间长度为8, 因此上述方法求解时,  $F_n > \frac{8}{0.2} = 40$ , 因此  $n = 9$ . 若采用停止准则常数为0.01, 其前两个观察量

◆  $\lambda_1 = -3 + \frac{F_7}{F_9} \cdot 8 = 0.054545, \mu_1 = -3 + \frac{F_8}{F_9} \cdot 8 = 1.945454$ , 目标函数值  $f(\lambda_1) < f(\mu_1)$ , 因此新区间为  $[-3, 1.945454]$

Iteration $k$	$a_k$	$b_k$	$\lambda_k$	$\mu_k$	$\theta(\lambda_k)$	$\theta(\mu_k)$
1	-3.000000	5.000000	0.054545	1.945454	0.112065*	7.675699*
2	-3.000000	1.945454	-1.109091	0.054545	-0.988099*	0.112065
3	-3.000000	0.054545	-1.836363	-1.109091	-0.300497*	-0.988099
4	-1.836363	0.054545	-1.109091	-0.672727	-0.988099	-0.892892*
5	-1.836363	-0.672727	-1.399999	-1.109091	-0.840001*	-0.988099
6	-1.399999	-0.672727	-1.109091	-0.963636	-0.988099	-0.998677*
7	-1.109091	-0.672727	-0.963636	-0.818182	-0.998677	-0.966942*
8	-1.109091	-0.818182	-0.963636	-0.963636	-0.998677	-0.998677
9	-1.109091	-0.963636	-0.963636	-0.953636	-0.998677	-0.997850*





# 第5章 无约束最优化方法-线搜索回顾

- ◆ 给定最后区间长度 $l$ , 这些方法所需要计算观察值的数目 $n$ 的情况如下
- ◆ 一致搜索(均分分割区间):  $n \geq \frac{b_1-a_1}{l/2}-1$
- ◆ Dichotomous搜索:  $\left(\frac{1}{2}\right)^{\frac{n}{2}} \leq \frac{l}{b_1-a_1}$
- ◆ 黄金分割:  $(0.618)^{n-1} \leq \frac{l}{b_1-a_1}$
- ◆ Fibonacci搜索:  $F_n \geq \frac{b_1-a_1}{l}$
- ◆ 固定比率 $\frac{b_1-a_1}{l}$ , 观察数目越少, 算法越有效, 从上面可以看出, 最有效的为Fibonacci, 其次为黄金分割, 再次为Dichotomous搜索, 最差的为一致搜索, 注意 $\frac{1}{F_n}$ 渐近于 $(0.618)^{n-1}$ , 所以后两个方法基本等同
- ◆ 在所有无导数最小化闭区间上的严格凸函数的方法中, Fibonacci搜索方法最有效, 要求最小数目的观察量
- ◆ 上述方法都是在凸函数的情况下获得, 对于一般的函数, 其不确定区间大, 此时可以分割成小区间, 找到小区间的局部极小点, 然后从所有局部极小点中找到全局极小点



# 第5章 无约束最优化方法-线搜索回顾

- ◆ 前述方法不要求导数信息，如果假设函数可导，例如假设  $f'(\lambda_k)$  已知，此时可以考虑如下三种情况：
- ◆ 如果  $f'(\lambda_k) = 0$ ，则  $\lambda_k$  即为最小值点
- ◆ 如果  $f'(\lambda_k) > 0$ ， $a_{k+1} = a_k$ ， $b_{k+1} = \lambda_k$
- ◆ 否则  $a_{k+1} = \lambda_k$ ， $b_{k+1} = b_k$
- ◆  $\lambda_k$  的位置必须使得新的不确定区间的长度的最大可能度最小，也就是说最小化  $\lambda_k - a_k$  和  $b_k - \lambda_k$ ，显然最优位置就是中点： $\lambda_k = \frac{a_k + b_k}{2}$
- ◆ 本方法与Dichotomous搜索非常类似，不过只需要计算一次中点的导数信息，**注意此时也可以看做是有限差分对导数的近似！**
- ◆ 固定最终区间长度为  $l$ ，计算的次数  $n$  必须满足： $\left(\frac{1}{2}\right)^n \leq \frac{l}{b_1 - a_1}$





## 第5章 无约束最优化方法-线搜索回顾

◆例如,  $\text{Min } \lambda^2 + 2\lambda, s. t. -3 \leq \lambda \leq 6$

◆设定  $l \leq 0.2$ , 此时:  $\left(\frac{1}{2}\right)^n \leq \frac{l}{b_1 - a_1} = 0.2/9 = 0.0222, n = 6$ ,  
最终区间为  $[-1.0313, -0.8907]$ , 因此极小点选为中点 -0.961

Iteration $k$	$a_k$	$b_k$	$\lambda_k$	$\theta'(\lambda_k)$
1	-3.0000	6.0000	1.5000	5.0000
2	-3.0000	1.5000	-0.7500	0.5000
3	-3.0000	-0.7500	-1.8750	-1.7500
4	-1.8750	-0.7500	-1.3125	-0.6250
5	-1.3125	-0.7500	-1.0313	-0.0625
6	-1.0313	-0.7500	-0.8907	0.2186
7	-1.0313	-0.8907		





# 第5章 无约束最优化方法-线搜索回顾

◆ Newton's方法：利用二次式来逼近原函数

◆  $q(\lambda) = f(\lambda_k) + f'(\lambda_k)(\lambda - \lambda_k) + \frac{1}{2}f''(\lambda_k)(\lambda - \lambda_k)^2$

◆ 令其导数为0，得  $\lambda_{k+1} = \lambda_k - \frac{f'(\lambda_k)}{f''(\lambda_k)}$

◆ 注意原函数二次可微，且  $f''(\lambda_k) \neq 0$

◆ 例：  $f(\lambda) = \begin{cases} 4\lambda^3 - 3\lambda^4 & \text{若 } \lambda \geq 0 \\ 4\lambda^3 + 3\lambda^4 & \text{若 } \lambda < 0 \end{cases}$ ，令  $\lambda_1 = 0.4$ ，结果如下，若令  $\lambda_1 = 0.6$ ，则发生震荡，但若初始点离极小点足够近，则保证收敛

Iteration $k$	$\lambda_k$	$\theta'(\lambda_k)$	$\theta''(\lambda_k)$	$\lambda_{k+1}$
1	0.400000	1.152000	3.840000	0.100000
2	0.100000	0.108000	2.040000	0.047059
3	0.047059	0.025324	1.049692	0.022934
4	0.022934	0.006167	0.531481	0.011331
5	0.011331	0.001523	0.267322	0.005634
6	0.005634	0.000379	0.134073	0.002807

Iteration $k$	$\lambda_k$	$\theta'(\lambda_k)$	$\theta''(\lambda_k)$	$\lambda_{k+1}$
1	0.600	1.728	1.440	-0.600
2	-0.600	1.728	-1.440	0.600
3	0.600	1.728	1.440	-0.600
4	-0.600	1.728	-1.440	0.600

$\lambda_1 = 0.6$ ，发生震荡



- ◆ 前述方法要求顺序求出一些函数值，并且利用函数信息也不会加速收敛过程，牛顿法不是全局收敛
- ◆ 但插值法在满足凸性和连续二次可微的前提下，可到达全局最优解
- ◆ 非精确一维搜索
- ◆ Goldstein

➤ 设  $f: R^n \rightarrow R$ 。在  $x$  取方向  $d$ ，有  $\nabla f^T(x)d < 0$ （即  $d$  为下降方向），令  $s^{(k)} = x^{(k+1)} - x^{(k)} = \lambda_k d^{(k)}$  求  $\lambda$  使

$$(1) f(x^{(k+1)}) - f(x^{(k)}) \leq \rho \nabla f^T(x^{(k)}) s^{(k)}$$

$$(2) f(x^{(k+1)}) - f(x^{(k)}) \geq (1 - \rho) \nabla f^T(x^{(k)}) s^{(k)}$$

# 第5章 无约束最优化方法-非精确线搜索回顾



◆ 设  $f: R^n \rightarrow R$ 。在  $x$  取方向  $d$ ，有  $\nabla f^T(x)d < 0$  (即  $d$  为下降方向), 令  $s^{(k)} = x^{(k+1)} - x^{(k)} = \lambda_k d^{(k)}$  求  $\lambda$  使

➤ Goldstein规则

$$(1) f(x^{(k+1)}) - f(x^{(k)}) \leq \rho \nabla f^T(x^{(k)})s^{(k)}$$

$$(2) f(x^{(k+1)}) - f(x^{(k)}) \geq (1 - \rho) \nabla f^T(x^{(k)})s^{(k)}$$

其中  $\rho \in (1, \frac{1}{2})$ , 实际中常取  $\rho = 0.1$  或更小

➤ Armijo规则

$$(1) f(x^{(k+1)}) - f(x^{(k)}) \leq \rho \nabla f^T(x^{(k)})s^{(k)}$$

$$(2) f(x^{(k+1)}) - f(x^{(k)}) \geq \mu \rho \nabla f^T(x^{(k)})s^{(k)}$$

$\mu$  取 5-10

➤ 1967年, Goldstein提出更一般的方法, 把(2)式改为:

$$(1) f(x^{(k+1)}) - f(x^{(k)}) \leq \rho \nabla f^T(x^{(k)})s^{(k)}$$

$$(2) f(x^{(k+1)}) \geq f(x^{(k)}) + \sigma \nabla f^T(x^{(k)})s^{(k)},$$

其中  $\sigma \in (\rho, 1)$

➤ Wolfe-Powell, 前面 Goldstein 方法中规则(2)改为对导数的要求

$$(\text{WP规则1}) f(x^{(k+1)}) \leq f(x^{(k)}) + \rho \nabla f^T(x^{(k)})s^{(k)}$$

$$(\text{WP规则2}) \nabla f^T(x^{(k+1)})s^{(k)} \geq \sigma \nabla f^T(x^{(k)})s^{(k)}$$

其中  $\rho \in (0, \frac{1}{2})$ ,  $\sigma \in (\rho, 1)$

➤ 如果需要较高的精度时, WP规则(2)可进一步改为:

(WP改进规则

$$2) |\nabla f^T(x^{(k+1)})d^{(k)}| \leq -\eta \nabla f^T(x^{(k)})d^{(k)}, \eta \in (0, 1)$$



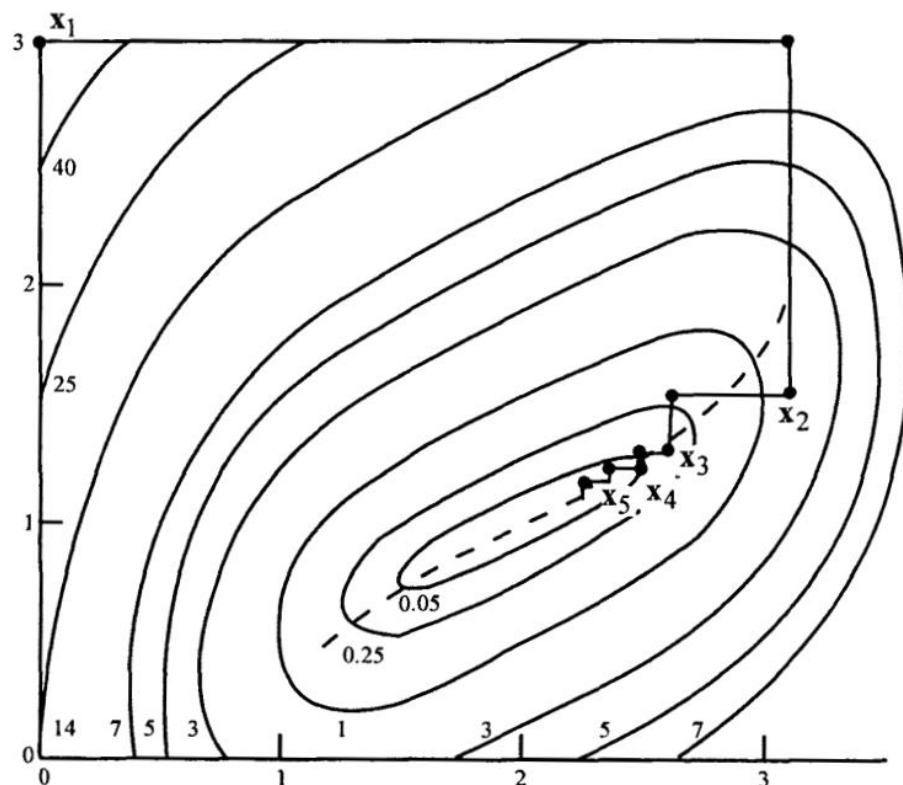
# 第5章无约束最优化方法(Unconstrained Optimization Methods)-坐标轮换法



## ◆坐标轮换法(Cyclic Coordinate Method)

- 坐标轴作为搜索方向,沿方向 $d_1, d_2, \dots, d_n$ 搜索, 其中 $d_j$ 是除第 $j$ 个位置为1别的位置为0的向量
- 此时只改变第 $j$ 个变量, 其它变量保持不动
- 例:  $\min (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点:  $(0,3)$ , 最优解 $(2,1)$
- $x^{(1)} = x^{(0)} + \lambda_0(1, 0) \Rightarrow \lambda_0 \in (3, 4)$
- $x^{(2)} = x^{(1)} + \lambda_1(0, 1) \Rightarrow \lambda_1 = -1.44$

Iteration $k$	$x_k$ $f(x_k)$	$j$	$d_j$	$y_j$	$\lambda_j$	$y_{j+1}$
1	(0.00, 3.00)	1	(1.0, 0.0)	(0.00, 3.00)	3.13	(3.13, 3.00)
	52.00	2	(0.0, 1.0)	(3.13, 3.00)	-1.44	(3.13, 1.56)
2	(3.13, 1.56)	1	(1.0, 0.0)	(3.13, 1.56)	-0.50	(2.63, 1.56)
	1.63	2	(0.0, 1.0)	(2.63, 1.56)	-0.25	(2.63, 1.31)
3	(2.63, 1.31)	1	(1.0, 0.0)	(2.63, 1.31)	-0.19	(2.44, 1.31)
	0.16	2	(0.0, 1.0)	(2.44, 1.31)	-0.09	(2.44, 1.22)
4	(2.44, 1.22)	1	(1.0, 0.0)	(2.44, 1.22)	-0.09	(2.35, 1.22)
	0.04	2	(0.0, 1.0)	(2.35, 1.22)	-0.05	(2.35, 1.17)
5	(2.35, 1.17)	1	(1.0, 0.0)	(2.35, 1.17)	-0.06	(2.29, 1.17)
	0.015	2	(0.0, 1.0)	(2.29, 1.17)	-0.03	(2.29, 1.14)
6	(2.29, 1.14)	1	(1.0, 0.0)	(2.29, 1.14)	-0.04	(2.25, 1.14)
	0.007	2	(0.0, 1.0)	(2.25, 1.14)	-0.02	(2.25, 1.12)
7	(2.25, 1.12)	1	(1.0, 0.0)	(2.25, 1.12)	-0.03	(2.22, 1.12)
	0.004	2	(0.0, 1.0)	(2.22, 1.12)	-0.01	(2.22, 1.11)





# 第5章无约束最优化方法 (Unconstrained Optimization Methods)-坐标轮换法



◆问题：每一次迭代有多少步骤？

◆假设初始点为  $x^{(1)} = (x_1^{(1)}, \dots, x_n^{(1)})^T$ , 则

◆  $x^{(2)} = \operatorname{argmin}_{x_1} f(x_1, x_2^{(1)}, x_3^{(1)}, \dots, x_n^{(1)})$

◆  $x^{(3)} = \operatorname{argmin}_{x_2} f(x_1^{(2)}, x_2, x_3^{(2)}, \dots, x_n^{(2)})$



# 第5章无约束最优化方法 (Unconstrained Optimization Methods)-坐标轮换法



## ◆注意

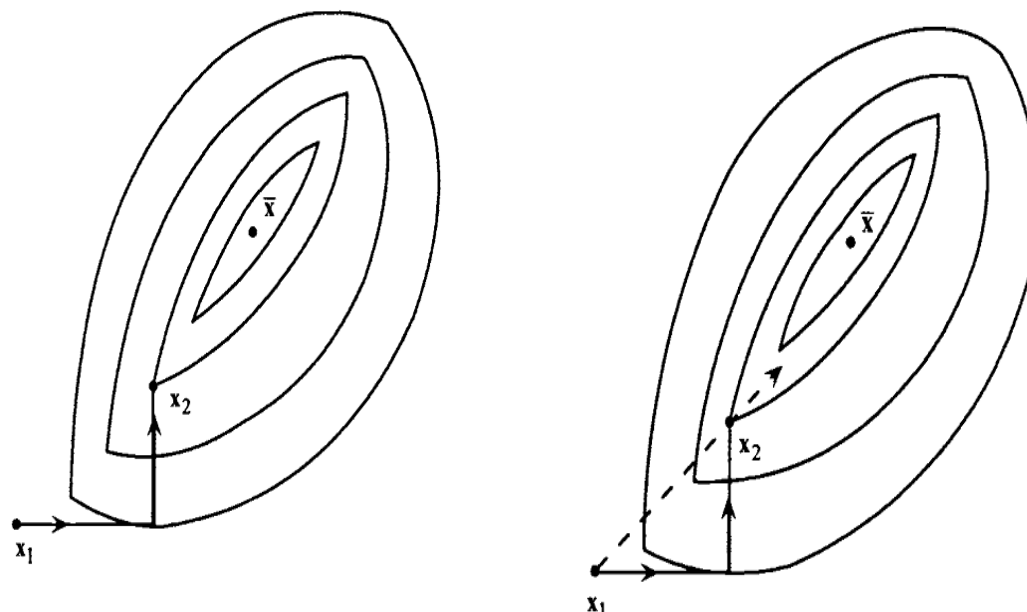
- 一次迭代在1,2,  $n$ 方向上做,下次重复此过程, 要求 $n$ 次迭代
- 也可在返回时做: **Aitken Double Sweep Method**,此时要求 $n - 1$ 次迭代
- 如果函数可微, 梯度存在, **Gauss-Southwell variant**推荐最小化坐标方向时选择偏导数成份幅度最大的方向进行最小化
- 这种顺序一维最小化有时称为**Gauss-Seidel**迭代, 可用于解线性方程组
- 该方法与最速下降法的收敛速度相当



# 第5章无约束最优化方法(Unconstrained Optimization Methods)-坐标轮换法



- ◆ 函数可微时，方法会收敛到梯度为0的点，但不可微时，则可能会在非最优点停止，如下左图，任何坐标方向都不会有函数值下降的点，此时可以通过搜索 $x_2 - x_1$ 方向来克服



- ◆ 注意，这种沿方向 $x_{k+1} - x_k$ 搜索的方式在坐标轮换方法中经常使用，有时函数可微时也这样，并且固定 $k$ 次迭代后进行一次这样的搜索，通常会加速收敛，经常称为加速步(acceleration step, pattern search step)

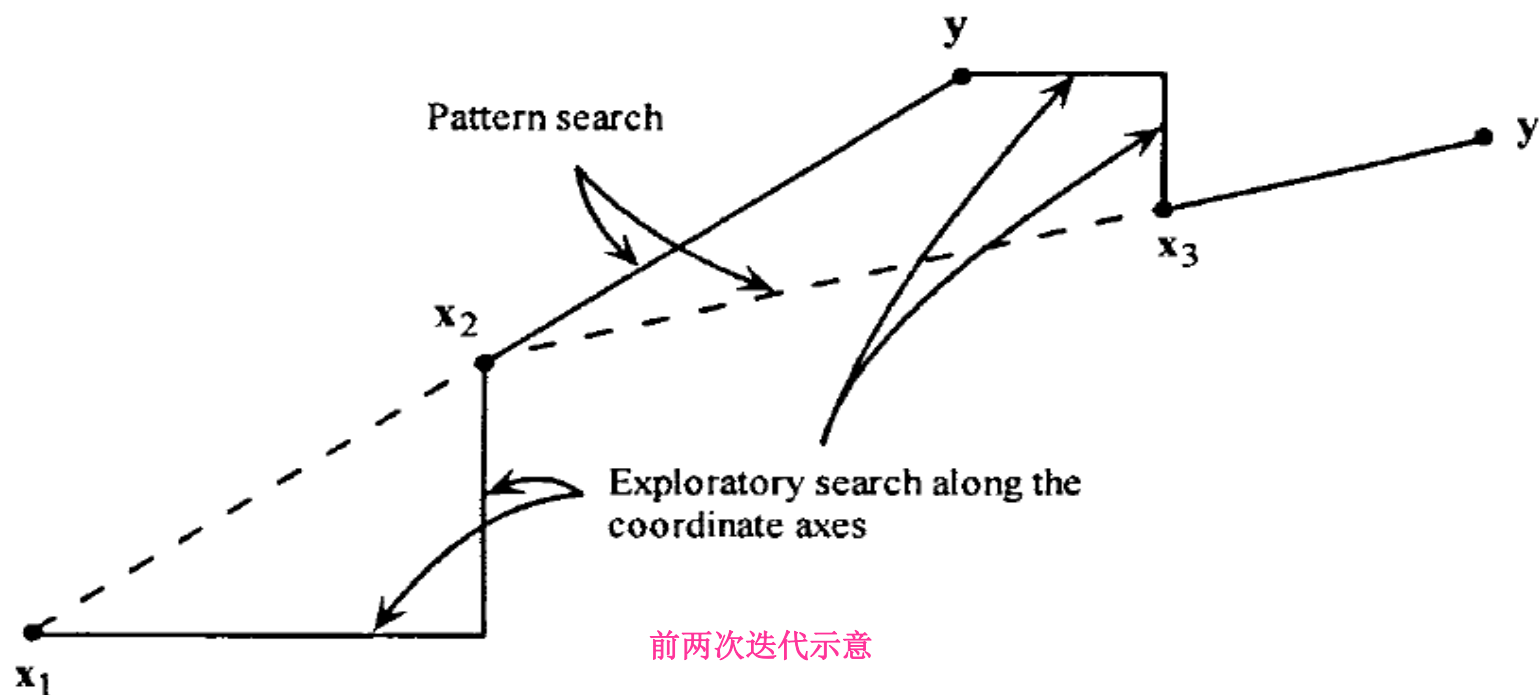
# 第5章无约束最优化方法(Unconstrained Optimization Methods)-Hooke and Jeeves方法



## ◆ Hooke and Jeeves方法，执行两种类型的搜索

- 试探搜索(Exploratory 搜索)
- 模式搜索(Pattern搜索)

◆ 步骤：给定点 $x^{(0)}$ ,沿坐标轴方向试探搜索到 $x^{(1)}$ ,然后沿方向 $x^{(1)} - x^{(0)}$ 进行模式搜索得到点 $y$ ,从该点采用试探搜索得到点 $x^{(2)}$ ,再沿方向 $x^{(2)} - x^{(1)}$ 执行模式搜索，产生点 $y'$ ,该过程再次重复



# 第5章无约束最优化方法(Unconstrained Optimization Methods)-Hooke and Jeeves方法



◆例:  $Min (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3)

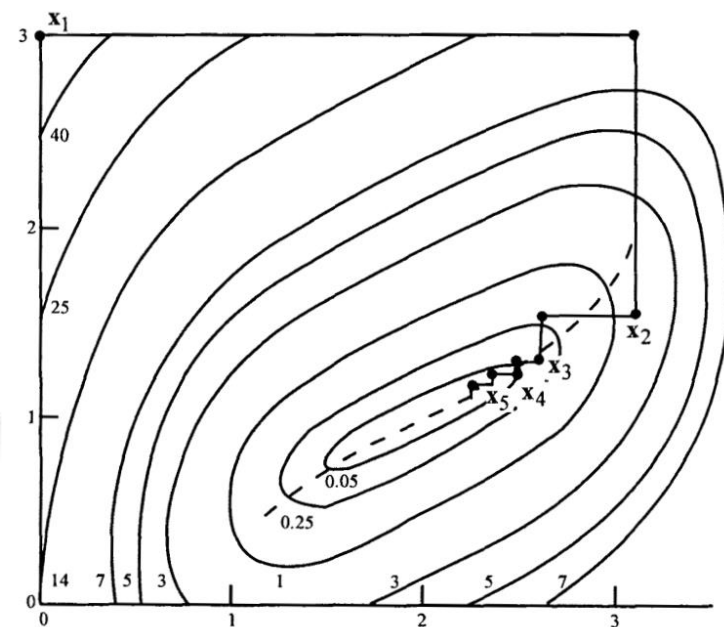
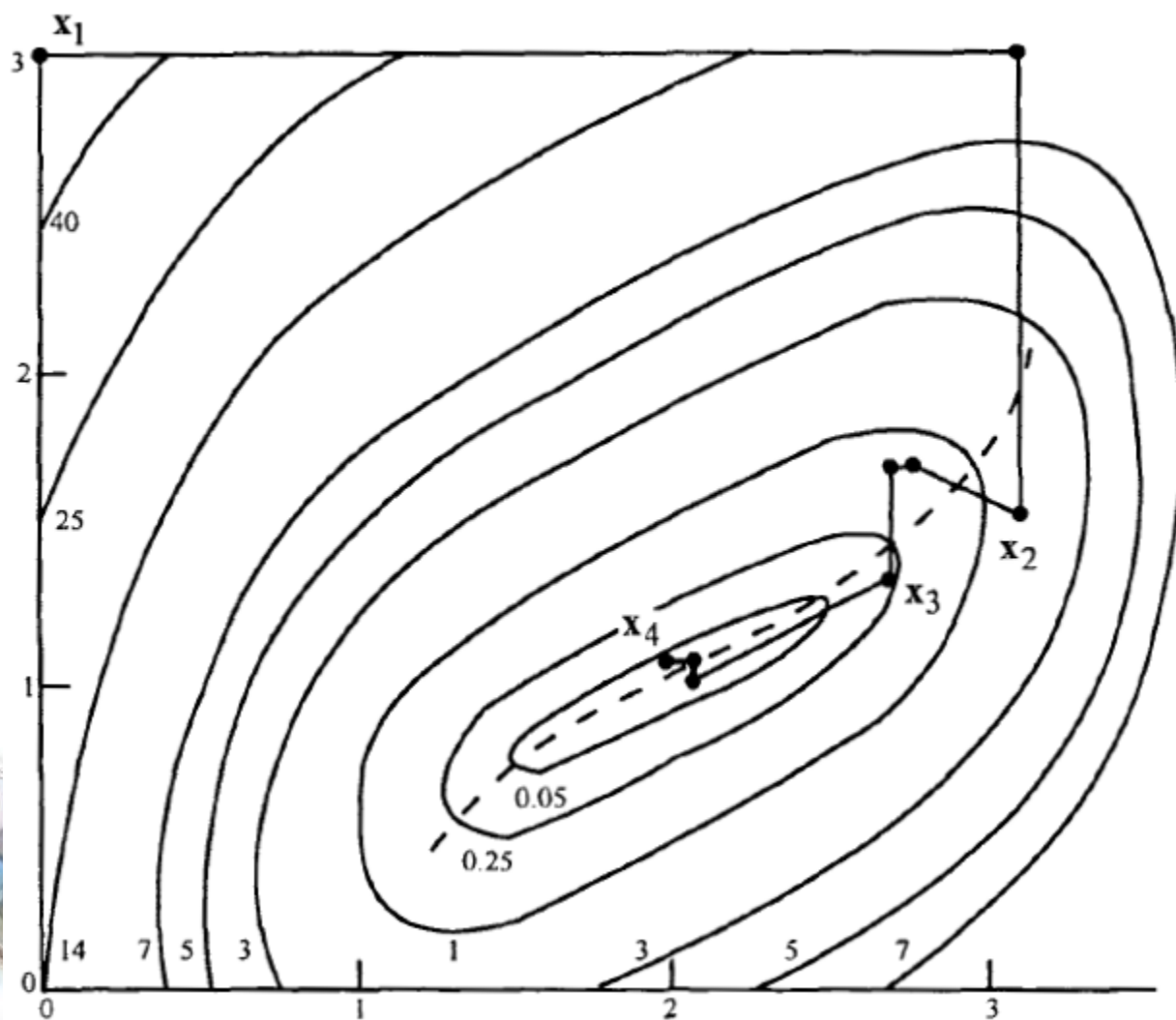
Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$	$\mathbf{d}_j$	$\hat{\lambda}_j$	$\mathbf{y}_{j+1}$	$\mathbf{d}$	$\hat{\lambda}$	$\mathbf{y}_3 + \hat{\lambda}\mathbf{d}$
1	(0.00, 3.00) 52.00	1	(0.00, 3.00)	(1.0, 0.0)	3.13	(3.13, 3.00)	—	—	—
		2	(3.13, 3.00)	(0.0, 1.0)	-1.44	(3.13, 1.56)	(3.13, 1.44)	-0.10	(2.82, 1.70)
2	(3.13, 1.56) 1.63	1	(2.82, 1.70)	(1.0, 0.0)	-0.12	(2.70, 1.70)	—	—	—
		2	(2.70, 1.70)	(0.0, 1.0)	-0.35	(2.70, 1.35)	(-0.43, -0.21)	1.50	(2.06, 1.04)
3	(2.70, 1.35) 0.24	1	(2.06, 1.04)	(1.0, 0.0)	-0.02	(2.04, 1.04)	—	—	—
		2	(2.04, 1.04)	(0.0, 1.0)	-0.02	(2.04, 1.02)	(-0.66, -0.33)	0.06	(2.00, 1.00)
4	(2.04, 1.02) 0.000003	1	(2.00, 1.00)	(1.0, 0.0)	0.00	(2.00, 1.00)	—	—	—
		2	(2.00, 1.00)	(0.0, 1.0)	0.00	(2.00, 1.00)			
5	(2.00, 1.00) 0.00								



# 第5章无约束最优化方法(Unconstrained Optimization Methods)-Hooke and Jeeves方法



## ◆ 图示：沿峡谷方向显著改进收敛速度



# 第5章无约束最优化方法(Unconstrained Optimization Methods)-Hooke and Jeeves方法

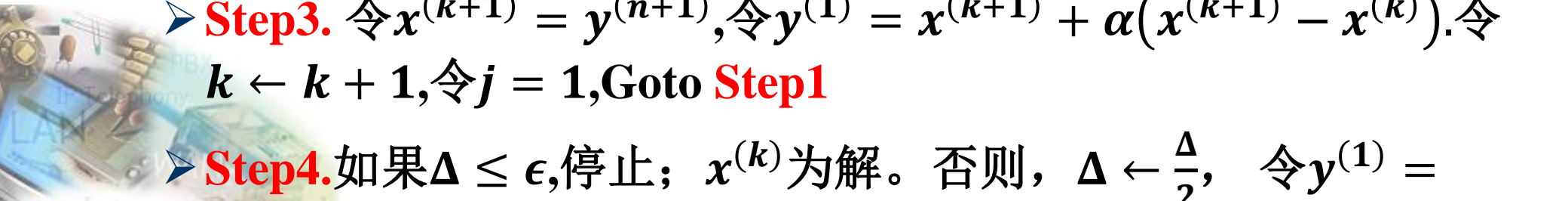


## ◆离散步长的Hooke和Jeeves方法

- $d_1, d_2, \dots, d_n$  为坐标轴方向, 标量  $\epsilon > 0$  作为终止算法的参数, 步长  $\Delta \geq \epsilon$ , 加速因子  $\alpha > 0$ , 初始点  $x^{(1)}$ , 令  $y^{(1)} = x^{(1)}$ ,  $k = j = 1$
- **Step1.** 在  $f(y^{(j)} + \Delta d_j) < f(y^{(j)})$ , 试验成功, 令  $y^{(j+1)} = y^{(j)} + \Delta d_j$ , Goto **Step2**, 否则如果  $f(y^{(j)} - \Delta d_j) \geq f(y^{(j)})$ , 试验失败。此时, 如果  $f(y^{(j)} - \Delta d_j) < f(y^{(j)})$ , 令  $y^{(j+1)} = y^{(j)} - \Delta d_j$ , Goto **Step2**; 否则  $y^{(j+1)} = y^{(j)}$ , Goto **Step2**;
- **Step2.** 若  $j < n$ ,  $j \leftarrow j + 1$  重复 **Step1**. 否则, 若  $f(y^{(n+1)}) < f(x^{(k)})$ , Goto **step 3**; 否则 Goto **Step4**
- **Step3.** 令  $x^{(k+1)} = y^{(n+1)}$ , 令  $y^{(1)} = x^{(k+1)} + \alpha(x^{(k+1)} - x^{(k)})$ . 令  $k \leftarrow k + 1$ , 令  $j = 1$ , Goto **Step1**
- **Step4.** 如果  $\Delta \leq \epsilon$ , 停止;  $x^{(k)}$  为解。否则,  $\Delta \leftarrow \frac{\Delta}{2}$ , 令  $y^{(1)} = x^{(k)}$ ,  $x^{(k+1)} = x^{(k)}$ ,  $k \leftarrow k + 1$ , 令  $j = 1$ , 重复 **Step1**.

Exploratory

Pattern  
搜索





# 第5章无约束最优化方法(Unconstrained Optimization Methods)-Hooke and Jeeves方法

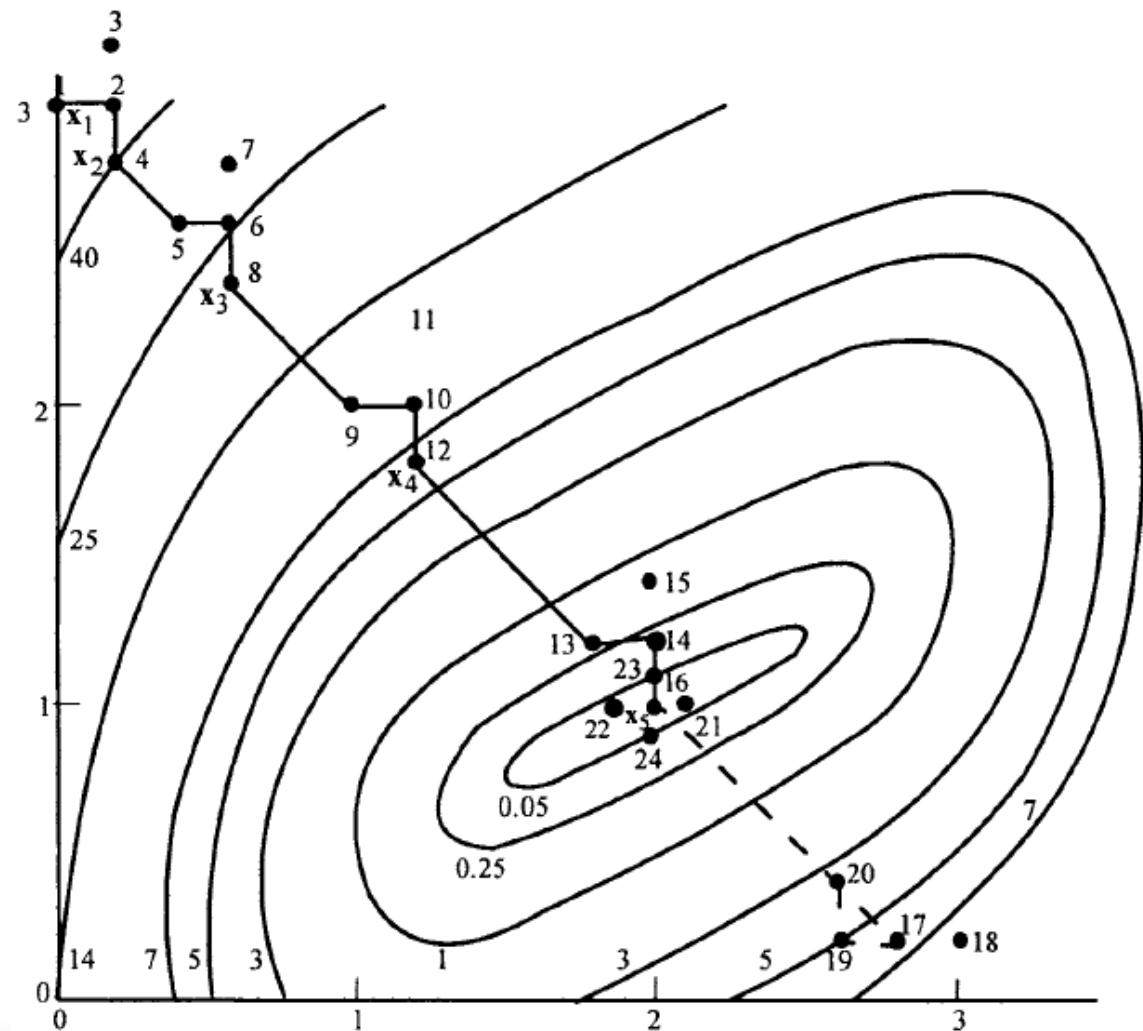


◆ 例:  $\text{Min } (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3)

◆ 参数  $\alpha = 1.0, \Delta = 0.2$

◆ 数字表示点的顺序

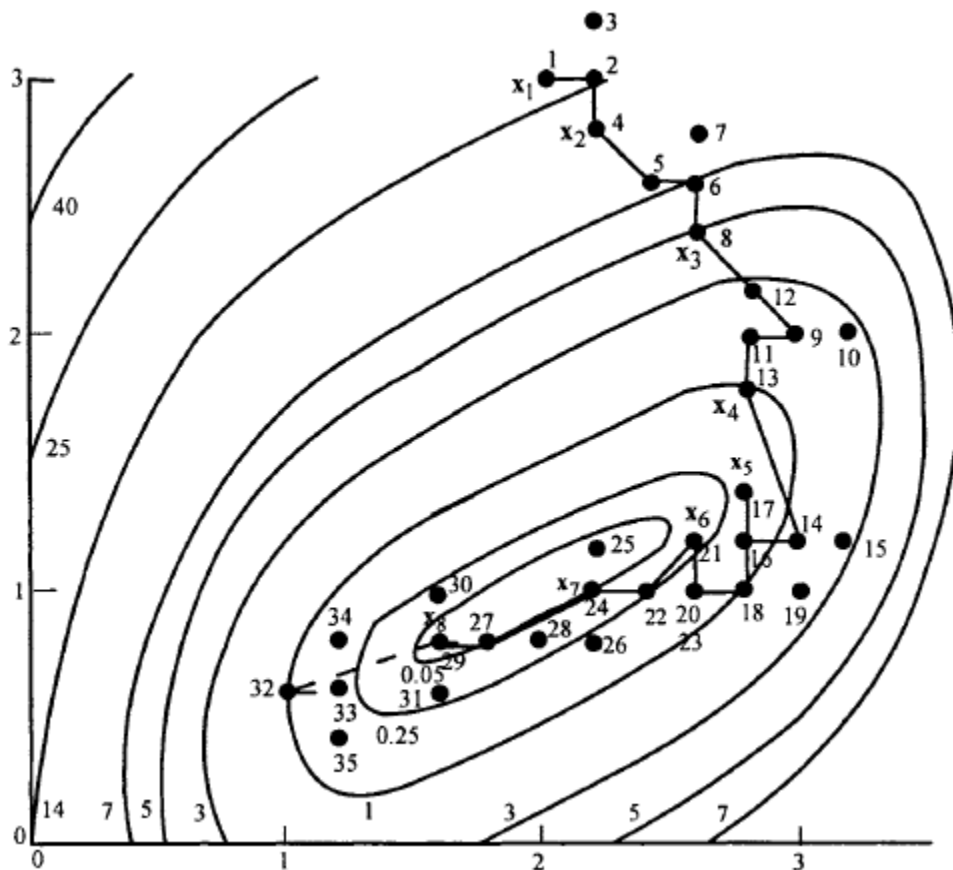
◆ 虚线表示拒绝的加速步



# 例:

◆ 例:  $\text{Min} (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(2, 3)

◆ 参数  $\alpha = 1.0, \Delta = 0.2$ , (S) 表示成功, (F) 表示失败



Iteration $k$	$\Delta$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$ $f(\mathbf{y}_j)$	$\mathbf{d}_j$	$\mathbf{y}_j + \Delta \mathbf{d}_j$ $f(\mathbf{y}_j + \Delta \mathbf{d}_j)$	$\mathbf{y}_j - \Delta \mathbf{d}_j$ $f(\mathbf{y}_j - \Delta \mathbf{d}_j)$
1	0.2	(2.00, 3.00) 16.00	1	(2.00, 3.00) 16.00	(1.0, 0.0)	(2.20, 3.00) 14.44(S)	—
			2	(2.20, 3.00) 14.44	(0.0, 1.0)	(2.20, 3.20) 17.64(F)	(2.20, 2.80) 11.56(S)
2	0.2	(2.20, 2.80) 11.56	1	(2.40, 2.60) 7.87	(1.0, 0.0)	(2.60, 2.60) 6.89(S)	—
			2	(2.60, 2.60) 6.89	(0.0, 1.0)	(2.60, 2.80) 9.13(F)	(2.60, 2.40) 4.97(S)
3	0.2	(2.60, 2.40) 4.97	1	(3.00, 2.00) 2.00	(1.0, 0.0)	(3.20, 2.00) 2.71(F)	(2.80, 2.00) 1.85(S)
			2	(2.80, 2.00) 1.85	(0.0, 1.0)	(2.80, 2.20) 2.97(F)	(2.80, 1.80) 1.05(S)
4	0.2	(2.80, 1.80) 1.05	1	(3.00, 1.20) 1.36	(1.0, 0.0)	(3.20, 1.20) 2.71(F)	(2.80, 1.20) 0.57(S)
			2	(2.80, 1.20) 0.57	(0.0, 1.0)	(2.80, 1.40) 0.41(S)	—
5	0.2	(2.80, 1.40) 0.41	1	(2.80, 1.00) 1.05	(1.0, 0.0)	(3.00, 1.00) 2.00(F)	(2.60, 1.00) 0.49(S)
			2	(2.60, 1.00) 0.49	(0.0, 1.0)	(2.60, 1.20) 0.17(S)	—
6	0.2	(2.60, 1.20) 0.17	1	(2.40, 1.00) 0.19	(1.0, 0.0)	(2.60, 1.00) 0.49(F)	(2.20, 1.00) 0.04(S)
			2	(2.20, 1.00) 0.04	(0.0, 1.0)	(2.20, 1.20) 0.04(F)	(2.20, 0.80) 0.36(F)
7	0.2	(2.20, 1.00) 0.04	1	(1.80, 0.80) 0.04	(1.0, 0.0)	(2.00, 0.80) 0.16(F)	(1.60, 0.80) 0.03(S)
			2	(1.60, 0.80) 0.03	(0.0, 1.0)	(1.60, 1.00) 0.19(F)	(1.60, 0.60) 0.19(F)
8	0.2	(1.60, 0.80) 0.03	1	(1.00, 0.60) 0.67	(1.0, 0.0)	(1.20, 0.60) 0.41(S)	—
			2	(1.20, 0.60) 0.41	(0.0, 1.0)	(1.20, 0.80) 0.57(F)	(1.20, 0.40) 0.57(F)
9	0.1	(1.60, 0.80) 0.03	1	(1.60, 0.80) 0.03	(1.0, 0.0)	(1.70, 0.80) 0.02(S)	—
			2	(1.70, 0.80) 0.02	(0.0, 1.0)	(1.70, 0.90) 0.02(F)	(1.70, 0.70) 0.10(F)
10	0.1	(1.70, 0.80) 0.02	1	(1.80, 0.80) 0.04	(1.0, 0.0)	(1.90, 0.80) 0.09(F)	(1.70, 0.80) 0.02(S)
			2	(1.70, 0.80) 0.02	(0.0, 1.0)	(1.70, 0.90) 0.02(F)	(1.70, 0.70) 0.10(F)

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Rosenbrock方法



- ◆ Rosenbrock方法:原始方法并没有线索搜而是直接采取沿搜索方向的离散步长搜索, 每次迭代沿 $n$ 个线性无关的正交方向进行, 当达到每次迭代终点时, 构造一组正交向量
- ◆ 假设 $d_1, d_2, \dots, d_n$ 是一组线性无关的范数为1的向量。并假设这组向量相互正交, 从当前向量 $x^{(k)}$ 开始, 目标函数 $f$ 沿每个方向进行最小化, 得到 $x^{(k+1)}$ 点, 此时:  $x^{(k+1)} - x^{(k)} = \sum_{j=1}^n \lambda_j d_j$ ,  $\lambda_j$ 为该方向上移动的距离, 可通过Gram-Schmidt过程来获得一组正交方向集合 $\bar{d}_j, j = 1, 2, \dots, n$ , (5-1)式如下:

$$a_j = \begin{cases} d_j, & \lambda_j = 0 \\ \sum_{i=1}^n \lambda_i d_i, & \lambda_j \neq 0 \end{cases}, b_j = \begin{cases} a_j, & j = 1 \\ a_j - \sum_{i=1}^{j-1} (a_j^t \bar{d}_i) \bar{d}_i, & j \geq 2 \end{cases}, \bar{d}_j = \frac{b_j}{\|b_j\|}$$

- ◆ 可以证明: 如果 $d_1, d_2, \dots, d_n$ 是一组线性无关相互正交的向量, 则上述构造的 $\bar{d}_j, j = 1, 2, \dots, n$ 对任意 $\lambda_j$ 也是线性无关相互正交的向量, 且如果 $\lambda_j = 0$ , 则 $d_j = \bar{d}_j$

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Rosenbrock方法



◆例:  $\text{Min } (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3)

◆下左图是实用离散步长的Rosenbrock方法示意图,  
下右图是采用线搜索的Rosenbrock方法

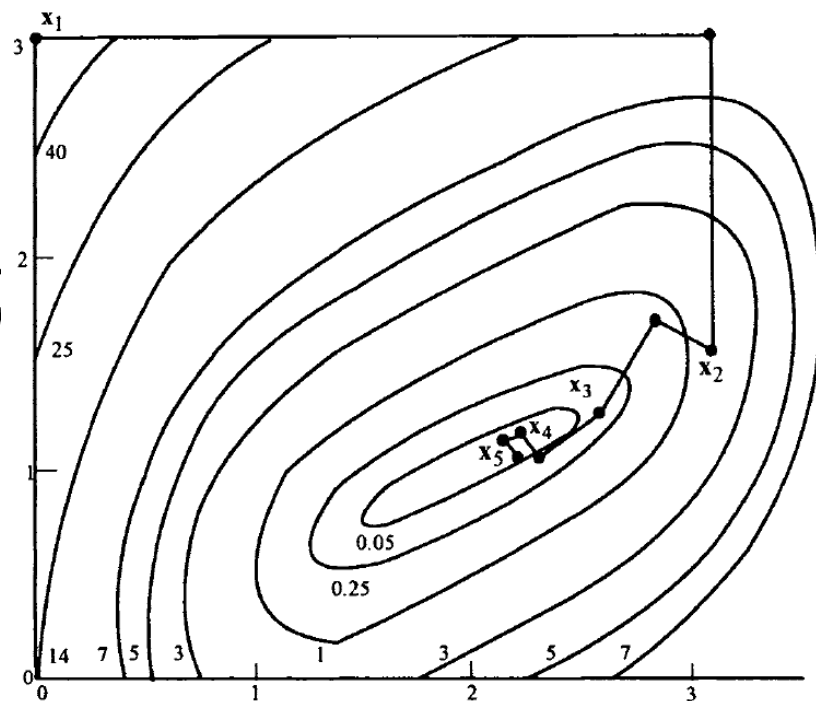
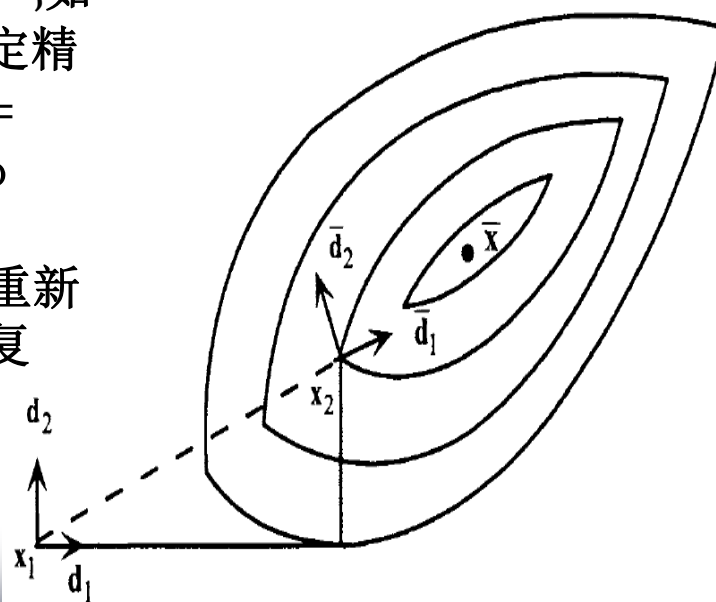
◆ Rosenbrock方法步骤:

Step1. 沿坐标方向搜索得到  
 $\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \lambda_j \mathbf{d}_j$ ;

Step2. 令  $\mathbf{x}^{(k+1)} = \mathbf{y}^{(k+1)}$ , 如果两次解的距离小于预定精度, 停止, 否则令  $\mathbf{y}^{(k)} = \mathbf{x}^{(k+1)}$ ,  $k \leftarrow k + 1$ , Goto

Step3.

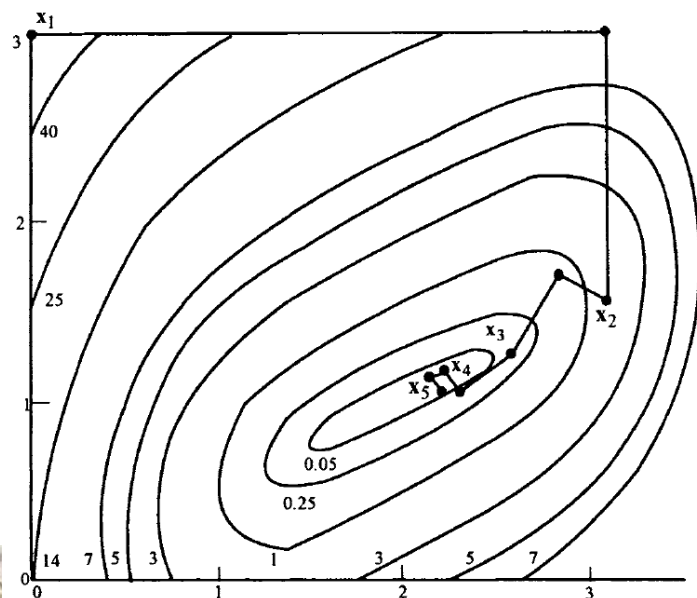
Step3. 根据(5-1)式, 重新得到新的正交方向, 重复Step1.



# 第5章无约束最优化方法(Unconstrained Optimization Methods) – Rosenbrock方法



## ◆采用线搜索的Rosenbrock方法计算实例



Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$ $f(\mathbf{y}_j)$	$\mathbf{d}_j$	$\lambda_j$	$\mathbf{y}_{j+1}$ $f(\mathbf{y}_{j+1})$
1	(0.00, 3.00) 52.00	1	(0.00, 3.00) 52.00	(1.00, 0.00)	3.13	(3.13, 3.00) 9.87
		2	(3.13, 3.00) 9.87	(0.00, 1.00)	-1.44	(3.13, 1.56) 1.63
2	(3.13, 1.56) 1.63	1	(3.13, 1.56) 1.63	(0.91, -0.42)	-0.34	(2.82, 1.70) 0.79
		2	(2.82, 1.70) 0.79	(-0.42, -0.91)	0.51	(2.16, 1.24) 0.16
3	(2.61, 1.24) 0.16	1	(2.61, 1.24) 0.16	(-0.85, -0.52)	0.38	(2.29, 1.04) 0.05
		2	(2.29, 1.04) 0.05	(0.52, -0.85)	-0.10	(2.24, 1.13) 0.004
4	(2.24, 1.13) 0.004	1	(2.24, 1.13) 0.004	(-0.96, -0.28)	0.04	(2.20, 1.12) 0.003
		2	(2.20, 1.12) 0.003	(0.28, -0.96)	0.02	(2.21, 1.10) 0.002



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Rosenbrock方法

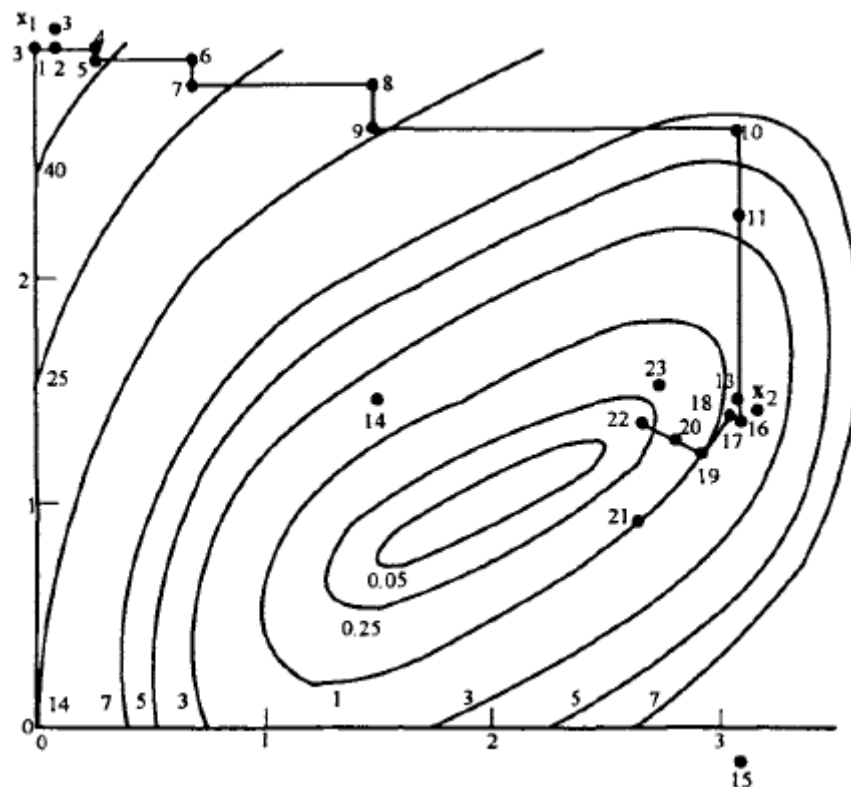


## ◆ 带离散步长的Rosenbrock方法

➤ 避免线搜索，函数值在具体的点上计算，可通过适当增加或减少步长来加速收敛，与以前的离散Jeeves方法类似，这里不再介绍

◆  $\text{Min } (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3), 方向步长=(0.1, 0.1), 增加步长因子 $\alpha = 2$ , 减少步长因子 $\beta = -0.5$

Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$ $f(\mathbf{y}_j)$	$\Delta_j$	$\mathbf{d}_j$	$\mathbf{y}_j + \Delta_j \mathbf{d}_j$ $f(\mathbf{y}_j + \Delta_j \mathbf{d}_j)$
1	(0.00, 3.00) 52.00	1	(0.00, 3.00) 52.00	0.10	(1.00, 0.00)	(0.10, 3.00) 47.84(S)
		2	(0.10, 3.00) 47.84	0.10	(0.00, 1.00)	(0.10, 3.10) 50.24(F)
		1	(0.10, 3.00) 47.84	0.20	(1.00, 0.00)	(0.30, 3.00) 40.84(S)
		2	(0.30, 3.00) 40.84	-0.05	(0.00, 1.00)	(0.30, 2.95) 39.71(S)
		1	(0.30, 2.95) 39.71	0.40	(1.00, 0.00)	(0.70, 2.95) 29.90(S)
		2	(0.70, 2.95) 29.90	-0.10	(0.00, 1.00)	(0.70, 2.85) 27.86(S)
		1	(0.70, 2.85) 27.86	0.80	(1.00, 0.00)	(1.50, 2.85) 17.70(S)
		2	(1.50, 2.85) 17.70	-0.20	(0.00, 1.00)	(1.50, 2.65) 14.50(S)
		1	(1.50, 2.65) 14.50	1.60	(1.00, 0.00)	(3.10, 2.65) 6.30(S)
		2	(3.10, 2.65) 6.30	-0.40	(0.00, 1.00)	(3.10, 2.25) 3.42(S)
		1	(3.10, 2.25) 3.42	3.20	(1.00, 0.00)	(6.30, 2.25) 345.12(F)
		2	(3.10, 2.25) 3.42	-0.80	(0.00, 1.00)	(3.10, 1.45) 1.50(S)



# 带离散步长的Rosenbrock方法数值例子



Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$ $f(\mathbf{y}_j)$	$\Delta_j$	$\mathbf{d}_j$	$\mathbf{y}_j + \Delta_j \mathbf{d}_j$ $f(\mathbf{y}_j + \Delta_j \mathbf{d}_j)$
1	(0.00, 3.00) 52.00	1	(0.00, 3.00) 52.00	0.10	(1.00, 0.00)	(0.10, 3.00) 47.84(S)
		2	(0.10, 3.00) 47.84	0.10	(0.00, 1.00)	(0.10, 3.10) 50.24(F)
		1	(0.10, 3.00) 47.84	0.20	(1.00, 0.00)	(0.30, 3.00) 40.84(S)
		2	(0.30, 3.00) 40.84	-0.05	(0.00, 1.00)	(0.30, 2.95) 39.71(S)
		1	(0.30, 2.95) 39.71	0.40	(1.00, 0.00)	(0.70, 2.95) 29.90(S)
		2	(0.70, 2.95) 29.90	-0.10	(0.00, 1.00)	(0.70, 2.85) 27.86(S)
		1	(0.70, 2.85) 27.86	0.80	(1.00, 0.00)	(1.50, 2.85) 17.70(S)
		2	(1.50, 2.85) 17.70	-0.20	(0.00, 1.00)	(1.50, 2.65) 14.50(S)
		1	(1.50, 2.65) 14.50	1.60	(1.00, 0.00)	(3.10, 2.65) 6.30(S)
		2	(3.10, 2.65) 6.30	-0.40	(0.00, 1.00)	(3.10, 2.25) 3.42(S)
		1	(3.10, 2.25) 3.42	3.20	(1.00, 0.00)	(6.30, 2.25) 345.12(F)
		2	(3.10, 2.25) 3.42	-0.80	(0.00, 1.00)	(3.10, 1.45) 1.50(S)

Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$j$	$\mathbf{y}_j$ $f(\mathbf{y}_j)$	$\Delta_j$	$\mathbf{d}_j$	$\mathbf{y}_j + \Delta_j \mathbf{d}_j$ $f(\mathbf{y}_j + \Delta_j \mathbf{d}_j)$
2	(3.10, 1.45) 1.50	1	(3.10, 1.45) 1.50	-1.60	(1.00, 0.00)	(1.50, 1.45) 2.02(F)
		2	(3.10, 1.45) 1.50	-1.60	(0.00, 1.00)	(3.10, -0.15) 13.02(F)
		1	(3.10, 1.45) 1.50	0.10	(0.89, -0.45)	(3.19, 1.41) 2.14(F)
		2	(3.10, 1.45) 1.50	0.10	(-0.45, -0.89)	(3.06, 1.36) 1.38(S)
		1	(3.06, 1.36) 1.38	-0.05	(0.89, -0.45)	(3.02, 1.38) 1.15(S)
		2	(3.02, 1.38) 1.15	0.20	(-0.45, -0.89)	(2.93, 1.20) 1.03(S)
		1	(2.93, 1.20) 1.03	-0.10	(0.89, -0.45)	(2.84, 1.25) 0.61(S)
		2	(2.84, 1.25) 0.61	0.40	(-0.45, -0.89)	(2.66, 0.89) 0.96(F)
		1	(2.84, 1.25) 0.61	-0.20	(0.89, -0.45)	(2.66, 1.34) 0.19(S)
		2	(2.66, 1.34) 0.19	-0.20	(-0.45, -0.89)	(2.75, 1.52) 0.40(F)



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Gradient Descent 方法

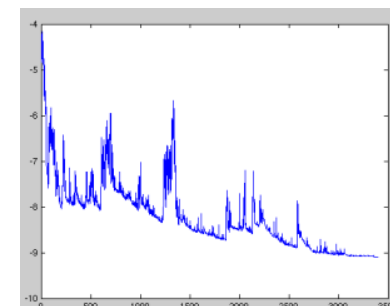


## ◆根据计算目标函数所使用数据的多少，有很多变种

- Batch GD: Vanilla GD:  $\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$ , 所有训练数据集来计算梯度
- SGD:  $\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)})$ ,
- Mini-Batch GD:  $\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$ ,

## ◆Mini-Batch GD

- 不保证收敛
- 学习率很难选
- 预定义规则，但阈值能无法动态适应不同数据集
- 同一学习率用于所有参数，无法自适应只更新很少发生的特征
- 鞍点给其带来很大困难



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



- ◆ 最速下降法 (Steepest Descent Method): 1847年Cauchy提出, 因此也称为Cauchy方法
- ◆ 方向 $d$ 为下降方向, 则在非零梯度 $x$ 处,  $d = -\nabla f(x) / \|\nabla f(x)\|$ , 为最速下降方向, 因此也称为梯度方法
- ◆ 例:  $\text{Min } (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点 $(0, 3)$
- ◆ 经过7次迭代后得 $x_8 = (2.28, 1.15)^T$
- ◆ 此时 $\|\nabla f(x_8)\| = 0.09$ , 终止

为什么?

引理:  $f: R^n \rightarrow R$  在 $x$ 处可微, 假设 $\nabla f(x) \neq 0$ .

则问题  $\min f'(x; d) = \lim_{\lambda \rightarrow 0^+} \frac{f(x + \lambda d) - f(x)}{\lambda} = \nabla f(x)^T d$ ,

s.t.  $\|d\| \leq 1$  的解为  $\bar{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$ , 即为 $f$ 在 $x$ 处的最速下降方向。

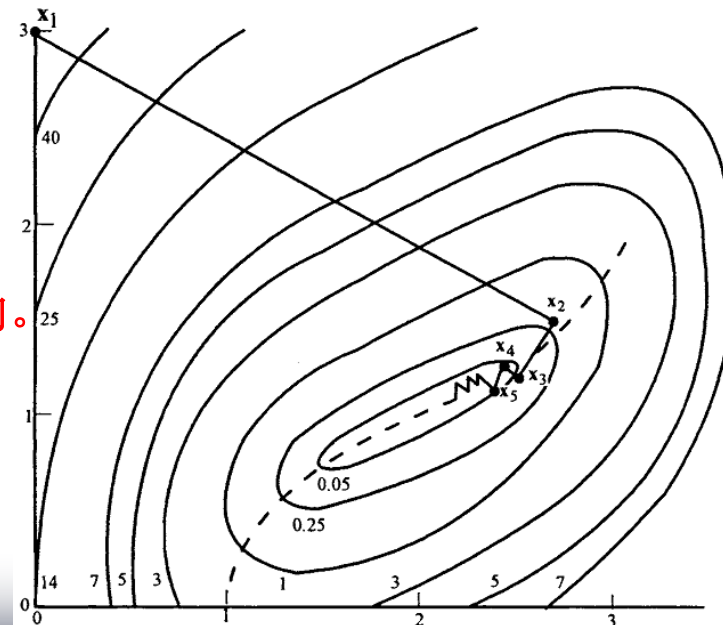
初始化, 令 $\epsilon > 0$ , 初始点 $x^{(1)}$ ,  $k = 1$

IF  $\|\nabla f(x^{(k)})\| < \epsilon$ , stop;

ELSE  $d_k = -\nabla f(x^{(k)})$ , 解得 $\lambda_k =$

$\text{argmin } f(x^{(k)} + \lambda d_k), \lambda \geq 0$ ,

令 $x^{(k+1)} = x^{(k)} + \lambda_k d_k, k = k + 1$ ;

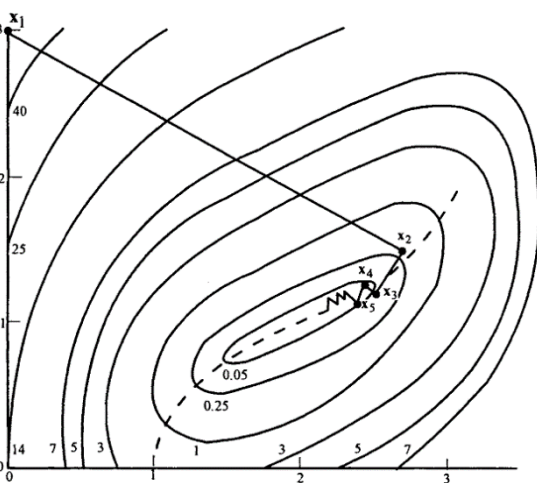


# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆最速下降法

Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)$	$\ \nabla f(\mathbf{x}_k)\ $	$\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$	$\lambda_k$	$\mathbf{x}_{k+1}$
1	(0.00, 3.00) 52.00	(-44.00, 24.00)	50.12	(44.00, -24.00)	0.062	(2.70, 1.51)
2	(2.70, 1.51) 0.34	(0.73, 1.28)	1.47	(-0.73, -1.28)	0.24	(2.52, 1.20)
3	(2.52, 1.20) 0.09	(0.80, -0.48)	0.93	(-0.80, 0.48)	0.11	(2.43, 1.25)
4	(2.43, 1.25) 0.04	(0.18, 0.28)	0.33	(-0.18, -0.28)	0.31	(2.37, 1.16)
5	(2.37, 1.16) 0.02	(0.30, -0.20)	0.36	(-0.30, 0.20)	0.12	(2.33, 1.18)
6	(2.33, 1.18) 0.01	(0.08, 0.12)	0.14	(-0.08, -0.12)	0.36	(2.30, 1.14)
7	(2.30, 1.14) 0.009	(0.15, -0.08)	0.17	(-0.15, 0.08)	0.13	(2.28, 1.15)
8	(2.28, 1.15) 0.007	(0.05, 0.08)	0.09			



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



◆ **练习**: 假设  $f(x) = x_1 x_2^2$ , 则其梯度为  $\nabla f = ?$  若  $x^{(k)} = [1, 2]^T$ , 则该点处的归一化最速下降方向  $d^{(k)}$  为多少?

◆ 若  $\alpha_k = \operatorname{argmin}_{\alpha} f(x^{(k)} + \alpha d^{(k)})$ , 则有

◆  $\nabla f(x^{(k)} + \alpha d^{(k)})^T d^{(k)} = 0$  (为什么)

$$\alpha = -\frac{d^T(Ax + b)}{d^T A d}$$

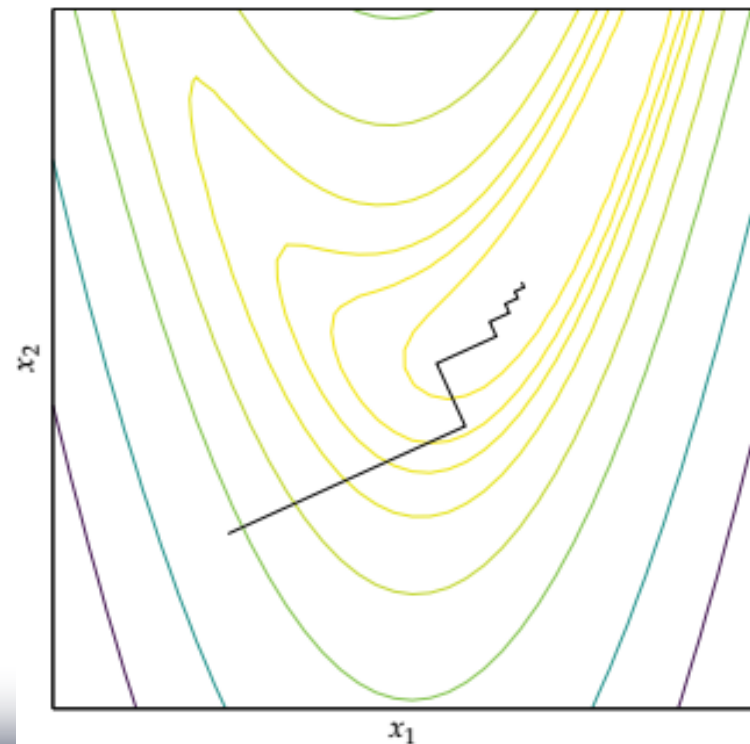
◆ 则必有:  $d^{(k+1)T} d^{(k)} = 0$

◆ 若  $f = \frac{1}{2} x^T A x + b x + c$ , 请计算

◆  $\min_{\alpha} f(x + \alpha d)$  中的  $\alpha = ?$

◆ 若  $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$

◆ 根据共轭  $\Rightarrow \beta_k = \frac{g^{(k+1)T} A d^{(k)}}{d^{(k)T} A d^{(k)}}$



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方

◆ 根据共轭  $\Rightarrow \beta_k = \frac{g^{(k+1)T} A d^{(k)}}{d^{(k)T} A d^{(k)}}$

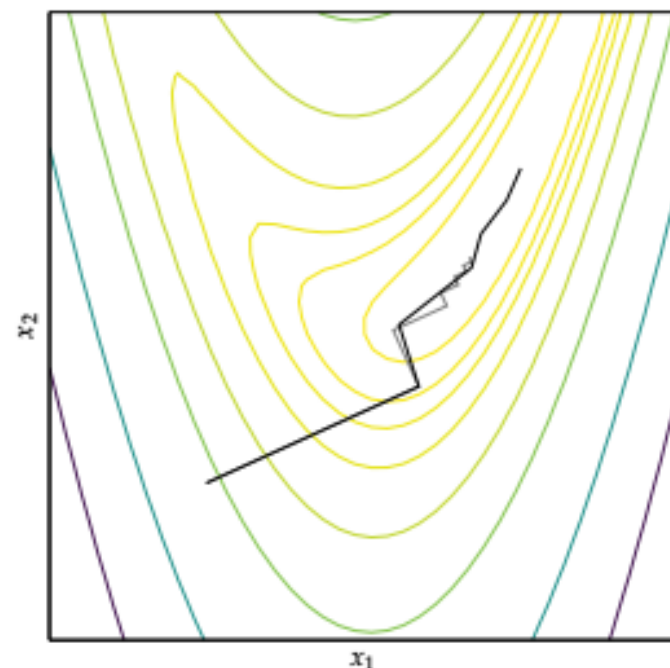
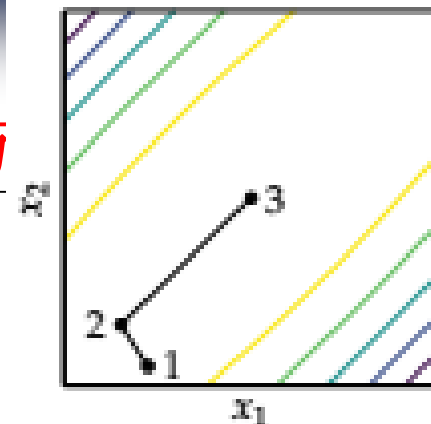
◆ 显然每次计算都需要知道共轭矩阵A，这比较困难，因此将 $\beta_k$ 放松

➤ Dai-Yuan:  $\beta_k = \frac{\|\nabla f(x_{k+1})\|^2}{\langle \nabla f(x_{k+1}) - \nabla f(x_k), d_k \rangle}$

➤ Fletcher-Reeves:  $\beta_k = \frac{g^{(k)T} g^{(k)}}{g^{(k-1)T} g^{(k-1)}}$

➤ Polak-Ribiere:  $\beta_k = \frac{g^{(k)T} (g^{(k)} - g^{(k-1)})}{g^{(k-1)T} g^{(k-1)}}$

➤ PR方法中，若令 $\beta$ 可自动重设 $\beta \leftarrow \max(\beta, 0)$ 则确保其收敛





# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆ $\min_{x \in R^n} f(x)$ , 满足如下假设

- $f \in C_M^{2,2}(R^n)$
- $f$  有一个局部极小值点  $x^* \in R^n$ , 该点处的 Hessian 矩阵正定
- 该点处的 Hessian 矩阵, 知道其上下界  $0 < \mu \leq L < \infty$ , 即
$$\mu I_n \leq \nabla^2 f(x^*) \leq L I_n$$
- 初始点  $x_0$  足够接近  $x^*$

## ◆ 定理: 令 $f \in C_M^{2,2}(R^n)$ , $\forall x, y \in R^n$ , 我们有

- $\|\nabla f(y) - \nabla f(x) - \nabla^2 f(x)(y - x)\| \leq \frac{M}{2} \|y - x\|^2$
- $|f(y) - f(x) - \langle \nabla f(x), y - x \rangle - \frac{1}{2} \langle \nabla^2 f(x)(y - x), y - x \rangle| \leq \frac{M}{6} \|y - x\|^3$

## ◆ 推论: 令 $f \in C_M^{2,2}(R^n)$ 和 $x, y \in R^n$ , 满足 $\|y - x\| = r$ , 则有

➤  $\nabla^2 f(x) - MrI_n \leq \nabla^2 f(y) \leq \nabla^2 f(x) + MrI_n$



## ◆ 梯度法及其收敛性分析

➤  $x_0 \in R^n$

➤  $x_{k+1} = x_k - h_k \nabla f(x_k), k = 0, 1, \dots$ , 步长  $h_k > 0$

## ◆ $h_k$ 的选取有很多变形

➤  $\{h_k\}_{k=0}^{\infty}$ : 如  $h_k = h > 0, h_k = \frac{h}{\sqrt{k+1}}$

➤ 全松弛 (精确步长):  $h_k = \operatorname{argmin}_{h \geq 0} f(x_k - h \nabla f(x_k))$

➤ Armijo规则: 对  $h > 0$ , 确定  $x_{k+1} = x_k - h \nabla f(x_k)$ , 满足

✓  $\alpha < \nabla f(x_k), x_k - x_{k+1} > \leq f(x_k) - f(x_{k+1})$

✓  $\beta < \nabla f(x_k), x_k - x_{k+1} > \geq f(x_k) - f(x_{k+1})$

✓ 其中,  $0 < \alpha < \beta < 1$  是一些固定参数.







## ◆ 梯度法及其收敛性分析

◆  $\min_{x \in R^n} f(x)$ , 满足如下假设

- $f \in C_M^{2,2}(R^n)$
- $f$  有一个局部极小值点  $x^* \in R^n$ , 该点处的 Hessian 矩阵正定
- 该点处的 Hessian 矩阵, 知道其上下界  $0 < \mu \leq L < \infty$ , 即
$$\mu I_n \leq \nabla^2 f(x^*) \leq L I_n$$
- 初始点  $x_0$  足够接近  $x^*$

◆ 定理: 令  $f \in C_M^{2,2}(R^n)$ ,  $\forall x, y \in R^n$ , 我们有

- $\|\nabla f(y) - \nabla f(x) - \nabla^2 f(x)(y - x)\| \leq \frac{M}{2} \|y - x\|^2$
- $|f(y) - f(x) - \langle \nabla f(x), y - x \rangle - \frac{1}{2} \langle \nabla^2 f(x)(y - x), y - x \rangle| \leq \frac{M}{6} \|y - x\|^3$

◆ 推论: 令  $f \in C_M^{2,2}(R^n)$  和  $x, y \in R^n$ , 满足  $\|y - x\| = r$ , 则有

- $\nabla^2 f(x) - Mr I_n \leq \nabla^2 f(y) \leq \nabla^2 f(x) + Mr I_n$



## ◆ 梯度法及其收敛性分析

◆ 定理：设函数  $f(\cdot)$  满足我们的假设，且初始点  $x_0$  足够接近一个严格局部极小点  $x^*$ ，即

$$r_0 = \|x_0 - x^*\| < \bar{r} = \frac{2\mu}{M}$$

则步长为  $h_k^* = \frac{2}{L+\mu}$  的梯度法收敛如下：

◆  $\|x_k - x^*\| \leq \frac{\bar{r}r_0}{\bar{r}-r_0} \left(1 - \frac{2\mu}{L+3\mu}\right)^k$ ，这种收敛速度称为线性收敛



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆ 靠近终点的之字形现象(Zigzagging): 收敛慢?

$$\begin{aligned} & (5-2) f(x_k + \lambda d) \\ &= f(x_k) + \lambda \nabla f(x_k)^T d + \lambda \|d\| \alpha(x_k; \lambda d) \end{aligned}$$

◆ 这里当  $\lambda d \rightarrow 0$  时,  $\alpha(x_k; \lambda d) \rightarrow 0$ , 且  $\|d\| = 1$ , 显然靠近最优点时  $\|\nabla f(x_k)\|$  很小, 使得(5-2)式右边第二项很小, 也即此时只用  $f$  的线性逼近来寻找移动方向, 可以预见, 后面的可忽略项对函数  $f$  的描述起到重要作用

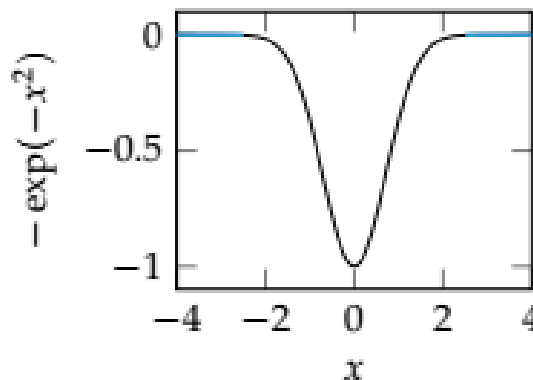
◆ 一般来说, 可通过偏转梯度方向来避免 Zigzagging.

◆ 例如  $d = -\nabla f(x) \rightarrow d = -D\nabla f(x)$  或者  $d = -\nabla f(x) + g$ , 这里  $D$  是一个合适的矩阵,  $g$  是一个合适的向量

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



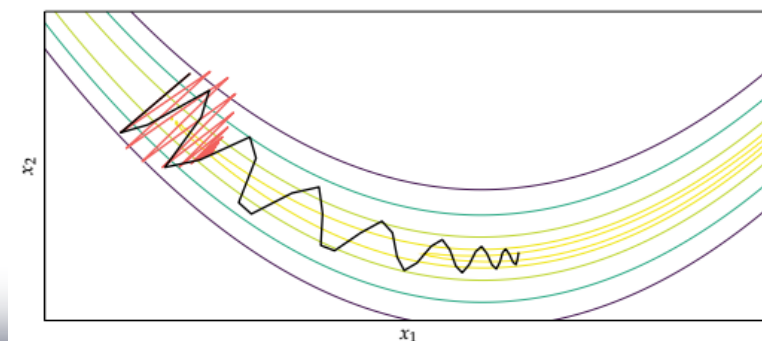
◆一般来说，一阶方法收敛慢，尤其是一些特殊的平坦曲面，下降慢



◆显然上述方法都是一阶方法，实际上对一阶方法还有很多变种

➤ Momentum更新:  $v^{(k+1)} = \beta v^{(k)} - \alpha g^{(k)}$ ;  $x^{(k+1)} = x^{(k)} + v^{(k+1)}$

➤ Rosenbrock函数  $b = 100$



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



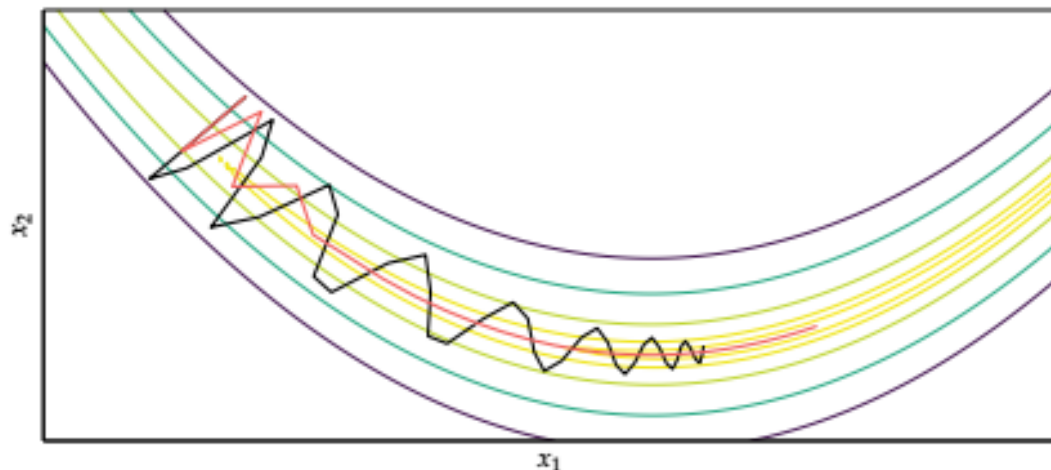
## ◆显然上述方法都是一阶方法，实际上对一阶方法还有很多变种

➤ **Momentum更新:**  $v^{(k+1)} = \beta v^{(k)} - \alpha g^{(k)}$ ;  $x^{(k+1)} = x^{(k)} + v^{(k+1)}$

✓ 动量法在底部不会减缓步长

➤ **Nesterov Momentum使用将来位置梯度**

✓  $v^{(k+1)} = \beta v^{(k)} - \alpha \nabla f(x^{(k)} + \beta v^{(k)})$ ;  $x^{(k+1)} = x^{(k)} + v^{(k+1)}$



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆显然上述方法都是一阶方法，实际上对一阶方法还有很多变种

➤ Momentum更新:  $v^{(k+1)} = \beta v^{(k)} - \alpha g^{(k)}; x^{(k+1)} = x^{(k)} + v^{(k+1)}$

动量法和N-动量法对所有分量统一参数

➤ Nesterov Momentum使用将来位置梯度

✓  $v^{(k+1)} = \beta v^{(k)} - \alpha \nabla f(x^{(k)} + \beta v^{(k)}); x^{(k+1)} = x^{(k)} + v^{(k+1)}$

➤ Adagrad方法(自适应子梯度方法): 为每个变量维度更新学习率

✓  $x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}} g_i^{(k)}$ , 其中  $s_i^{(k)} = \sum_{j=1}^k (g_i^{(j)})^2, \epsilon \sim 1 \times 10^{-8}$ ,

防止被0除

✓ Adagrad对学习率不敏感，一般设置为0.1

✓ 缺陷是s非下降，导致学习率会下降，影响收敛



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆ 显然上述方法都是一阶方法，实际上对一阶方法还有很多变种

- Momentum更新:  $v^{(k+1)} = \beta v^{(k)} - \alpha g^{(k)}; x^{(k+1)} = x^{(k)} + v^{(k+1)}$
- Nesterov Momentum使用将来位置梯度
  - ✓  $v^{(k+1)} = \beta v^{(k)} - \alpha \nabla f(x^{(k)} + \beta v^{(k)}); x^{(k+1)} = x^{(k)} + v^{(k+1)}$
- Adagrad方法(自适应子梯度方法):  $x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}} g_i^{(k)}$ , 其中  $s_i^{(k)} = \sum_{j=1}^k (g_i^{(j)})^2$ ,  $\epsilon \sim 1 \times 10^{-8}$ , 防止被0除
- RMSProp: 在adagrad上避免学习率下降:
- $\hat{s}^{(k+1)} = \gamma \hat{s}^{(k)} + (1 - \gamma)(g^{(k)} \odot g^{(k)})$ , 衰减率  $\gamma \in [0, 1]$  设定靠近0.9
- Update:  $x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{\hat{s}_i^{(k)}}} g_i^{(k)} = x_i^{(k)} - \frac{\alpha}{\epsilon + RMS(g_i)} g_i^{(k)}$
- Adadelta: 另一种避免Adagrad单调递减学习率的方法
- $x_i^{(k+1)} = x_i^{(k)} - \frac{RMS(\Delta x_i)}{\epsilon + RMS(g_i)} g_i^{(k)}$ , 完全消除学习率的影响

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



## ◆ 显然上述方法都是一阶方法，实际上对一阶方法还有很多变种

- Momentum更新; Nesterov Momentum; Adagrad方法(自适应子梯度方法)
- RMSProp; Adadelata
- Adam(Adaptive moment estimation)-2015
- 修改每一个参数的学习率，存储指数衰减平方梯度，如 RMSProp, Adadelata, 以及指数衰减梯度，如 Momentum
- 初始化梯度和平方梯度为0时会引入偏差，因此需要专门的偏差校正来减轻这个问题 ( $\alpha = 0.01, \gamma_v = 0.9, \gamma_s = 0.999, \epsilon = 1 \times 10^{-8}$ )
- 带偏差的衰减momentum:  $v^{(k+1)} = \gamma_v v^{(k)} + (1 - \gamma_v) g^{(k)}$
- 带偏差的衰减sq. gradient:  $s^{(k+1)} = \gamma_s s^{(k)} + (1 - \gamma_s) (g^{(k)} \odot g^{(k)})$

- 校正的衰减momentum:  $\hat{v}^{(k+1)} = \frac{v^{(k+1)}}{1 - \gamma_v^k}$

- 校正的衰减sq. gradient:  $\hat{s}^{(k+1)} = \frac{s^{(k+1)}}{1 - \gamma_s^k}$

- 迭代:  $x^{(k+1)} = x^{(k)} - \frac{\alpha \hat{v}^{(k+1)}}{\epsilon + \sqrt{\hat{s}^{(k+1)}}}$

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



- ◆ 梯度加速方法一般要么对学习率特别敏感，要么需要在执行过程中浪费很长时间来修改学习率
- ◆ 而学习率表明方法对梯度信号到底有多敏感！学习率太高或太低都会显著影响算法的性能
- ◆ 2018年ICML提出Hypergradient descent方法，认为学习率的的导数应该对改进优化器的性能有用
- ◆ 超梯度是对超参数的导数，超梯度算法降低了对超参数的敏感性，允许更快速的自适应
- ◆ 超梯度下降将梯度下降应用到隐含下降方法的学习率上，要求目标函数对学习率求偏导数。对于梯度下降，其偏导数为：
$$\frac{\partial f(x^{(k)})}{\partial \alpha} = (g^{(k)})^T \frac{\partial}{\partial \alpha} (x^{(k-1)} - \alpha g^{(k-1)}) = (g^{(k)})^T (-g^{(k-1)})$$
- ◆ 超梯度的计算要求上一次的梯度，最终的更新规则为：
$$\alpha^{(k+1)} = \alpha^{(k)} - \mu \frac{\partial f(x^{(k)})}{\partial \alpha} = \alpha^{(k)} + \mu (g^{(k)})^T g^{(k-1)}, \text{为超参数学习率}$$

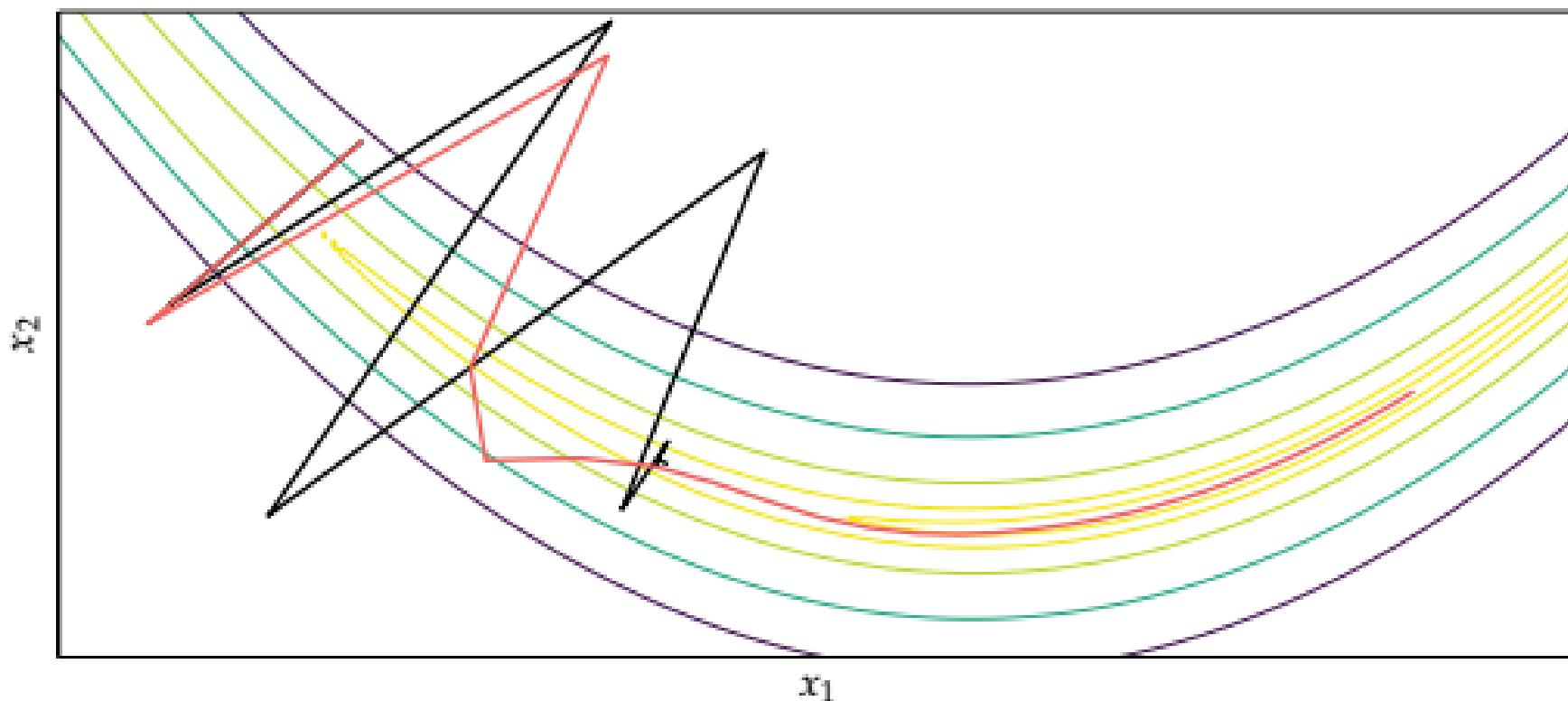
A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood, "Online Learning Rate Adaptation with Hypergradient Descent," in *International Conference on Learning Representations (ICLR)*, 2018.

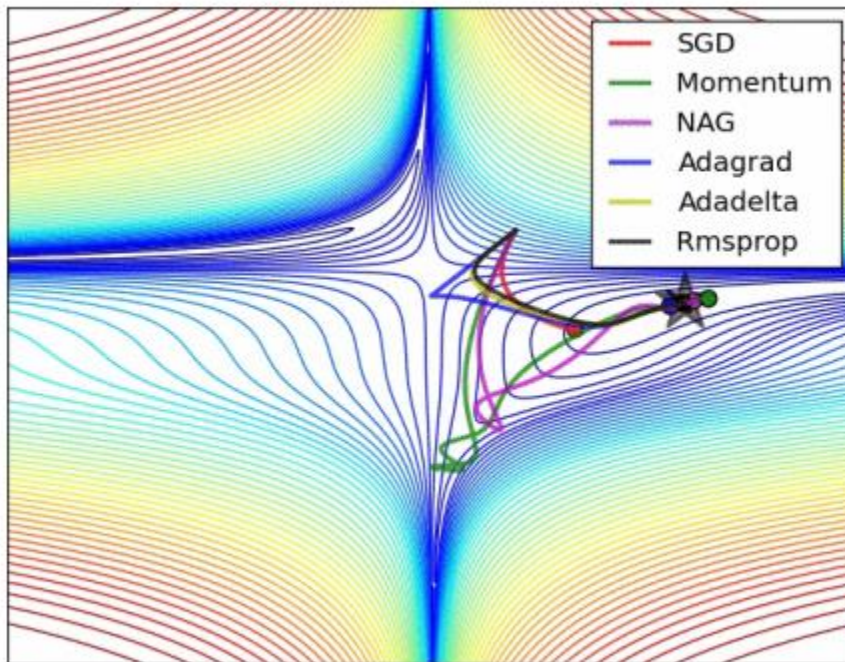
# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



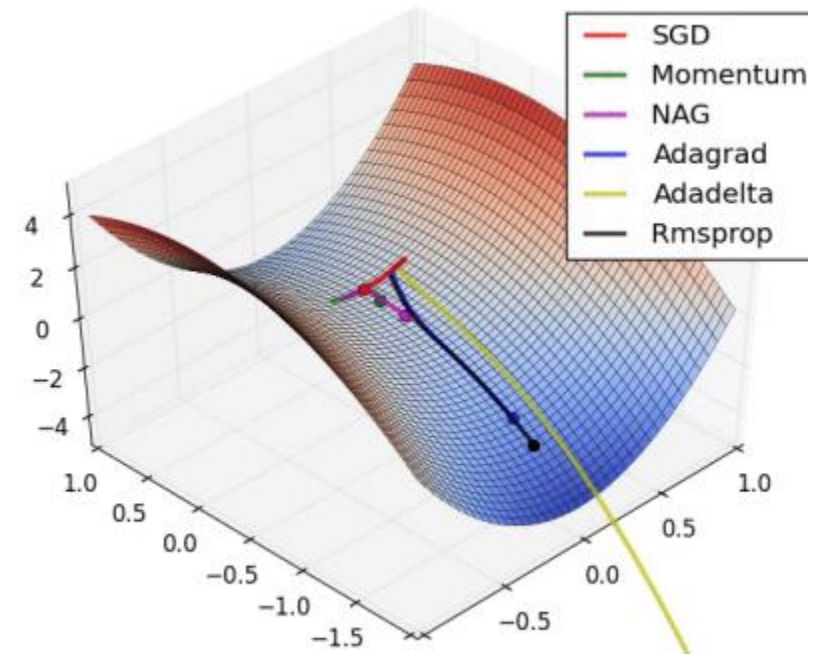
## ◆ 超梯度下降可以应用到任何类似梯度下降算法中

- 黑线表示Hypermomentum
- 红线表示Hyper-Nesterov





(a) SGD optimization on loss surface contours



(b) SGD optimization on saddle point

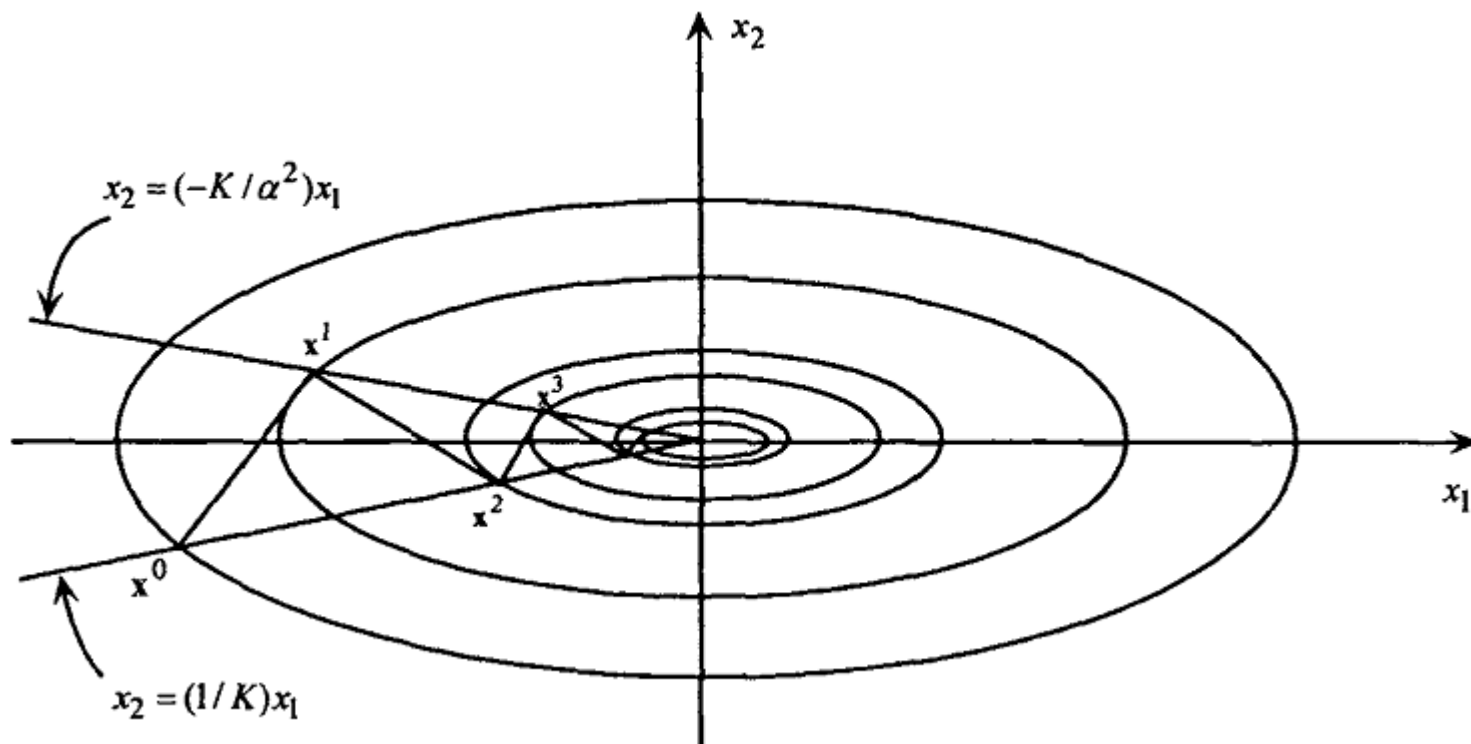


# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent 方法



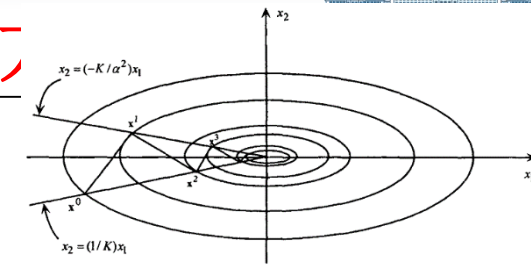
## ◆最速下降法的收敛速率分析示例

◆考虑双变量函数  $f(x_1, x_2) = \frac{1}{2}(x_1^2 + \alpha x_2^2)$ ,  $\alpha > 1$ , 其 Hessian 矩阵  $H = \text{diag}\{1, \alpha\}$ , 定义其条件数为其最大特征值与最小特征值之比





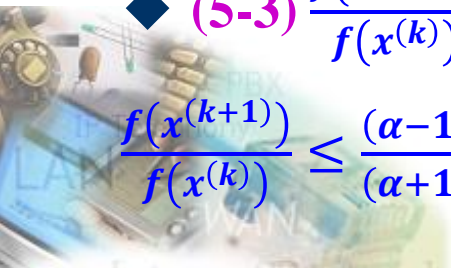
# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Steepest Descent



- ◆ 设初始点  $x = (x_1, x_2)^T$ , 最速下降法得  $x_{new} = x + \lambda d$ ,
- ◆  $d = -\nabla f(x) = -(x_1, \alpha x_2)^T$ , 注意  $x_1 \neq 0, x_2 \neq 0, \lambda > 0$
- ◆ 线搜索可得  $\min f(\lambda) \Rightarrow \lambda = \frac{x_1^2 + \alpha^2 x_2^2}{x_1^2 + \alpha^3 x_2^2}$ , 从而  $x_{new} = [\frac{\alpha^2 x_1 x_2 (\alpha - 1)}{x_1^2 + \alpha^3 x_2^2}, \frac{x_1^2 x_2 (1 - \alpha)}{x_1^2 + \alpha^3 x_2^2}]$ , 显然  

$$\frac{x_{1new}}{x_{2new}} = -\alpha^2 \frac{x_2}{x_1}$$
- ◆ 因此, 从初始解  $\frac{x_1}{x_2} = K \neq 0$ , SD方法产生迭代序列  $\{x^{(k)}\}, k = 1, 2, \dots$ , 且其两个元素之比  $\{\frac{x_1^{(k)}}{x_2^{(k)}}\}$  总是在  $K, -\frac{\alpha^2}{K}$  之间, 并且序列收敛到最优点  $x^* = (0, 0)^T$
- ◆ 注意:  $\alpha$  越大, 这种Zigzagging现象越明显, 为1时一次迭代收敛
- ◆ 此时考察  $\{f(x^{(k)})\}$  的收敛情况, 易得
- ◆ (5-3)  $\frac{f(x^{(k+1)})}{f(x^{(k)})} = \frac{K_k^2 \alpha (\alpha - 1)^2}{(K_k^2 + \alpha^3)(K_k^2 + \alpha)}, K_k \stackrel{\text{def}}{=} \frac{x_1^k}{x_2^k}$ , (5-3)式当  $K_k^2 = \alpha^2$  时最大化, 因此可得  

$$\frac{f(x^{(k+1)})}{f(x^{(k)})} \leq \frac{(\alpha - 1)^2}{(\alpha + 1)^2}, \text{因此逐渐} \rightarrow 1, \text{且速度越来越慢}$$



# 第5章无约束最优化方法(Unconstrained Optimization Methods) – Steepest Descent 方法

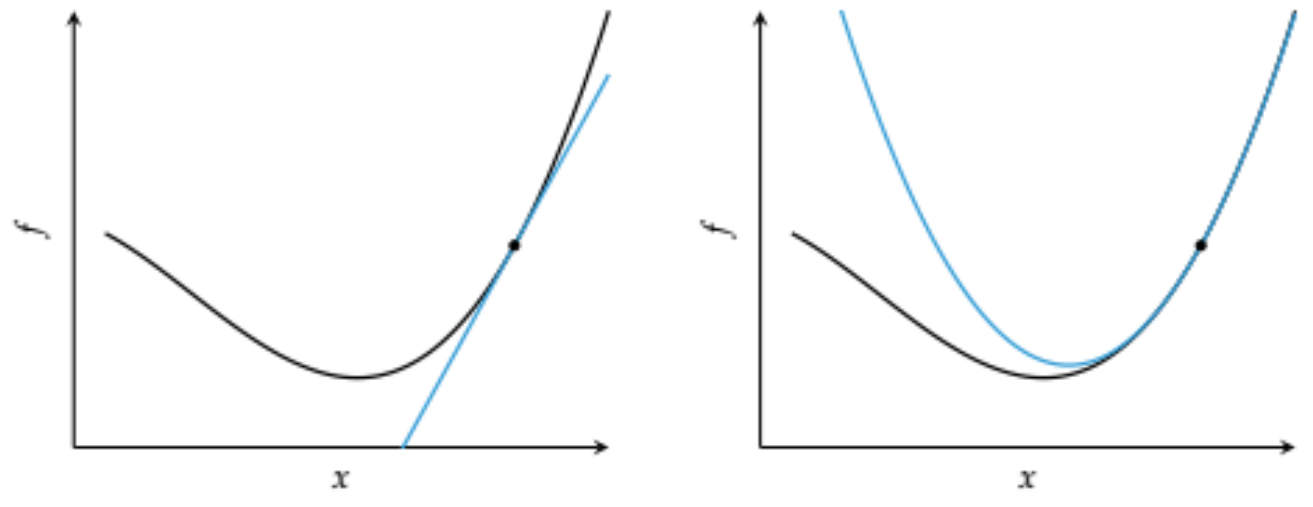


- ◆ 扩展到一般的二次函数  $f(x) = \frac{1}{2}x^T Hx + c^T x + b$ ,  $H$  为对称正定, 显然最优解  $x^*: Hx^* = -c$ , 令  $g_k = \nabla f(x_k) = c + Hx_k$ ,  $x_{k+1} = x_k + \lambda d_k \Rightarrow$
- ◆  $\lambda = \frac{g_k^T g_k}{g_k^T H g_k}$ ,  $x_{k+1} = x_k - \lambda g_k$
- ◆ 看看它的收敛速度如何?
- ◆ 设误差函数为  $e(x) = \frac{1}{2}(x - x^*)^T H(x - x^*) = f(x) + \frac{1}{2}x^{*T} Hx^*$ , 可以看出其与  $f(x)$  仅差一常数项
- ◆  $e(x_{k+1}) = \left[ 1 - \frac{(g_k^T g_k)^2}{(g_k^T H g_k)(g_k^T H^{-1} g_k)} \right] e(x_k) \leq \frac{(\alpha-1)^2}{(\alpha+1)^2} e(x_k)$ , 这里  $\alpha$  为  $H$  的条件数
- ◆ 因此,  $\{e(x_k)\} \rightarrow 0$  的速度以  $(\alpha-1)^2/(\alpha+1)^2$  为上界线性或几何收敛, 与前面的例子一样, 随  $\alpha$  增加, 收敛速率持续减慢, 并且与初始值相关
- ◆ 对于二次可微的非二次函数也有类似的结果, 这里不再赘述
- ◆ 如果使用不精确 Armijo 规则线搜索, 来进行最速下降法求解, 则只需梯度函数是  $G > 0$  Lipschitz 连续的,  $\|\nabla f(x) - \nabla f(y)\| \leq G\|x - y\|, \forall x, y \in S(x_0) \stackrel{\text{def}}{=}$
- ◆  $\{x: f(x) \leq f(x_0)\}$ , 则获得的序列  $\{x^{(k)}\}$  要么  $K$  次后达到梯度为 0 的点, 要么达到其极限为梯度为 0 的点

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) - Newton方法-二阶方法



- ◆ 利用目标函数的一阶导数信息来近似目标函数，求解优化问题的方法，统称为一阶方法
- ◆ 利用目标函数的二阶信息来近似目标函数，求解优化问题的方法，称为二阶方法
  - 最典型的的就是牛顿法
  - 一阶方法实际上并没有告诉应该沿该方向走多远



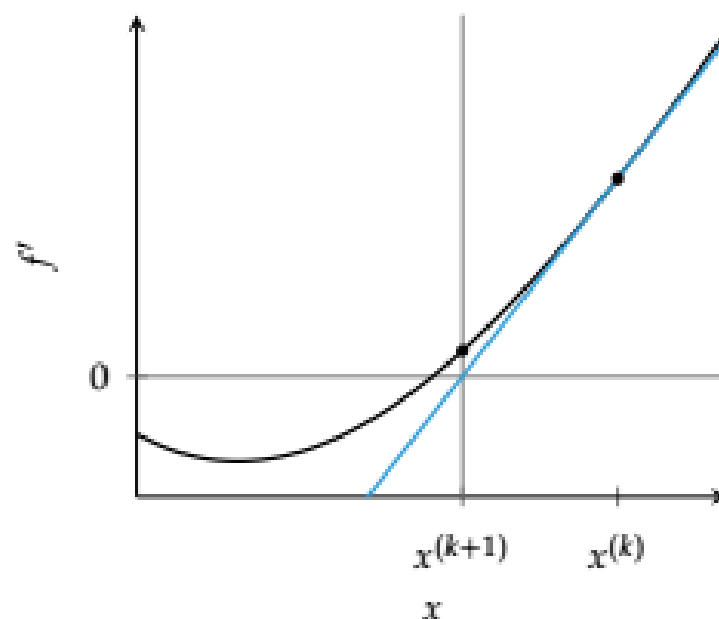
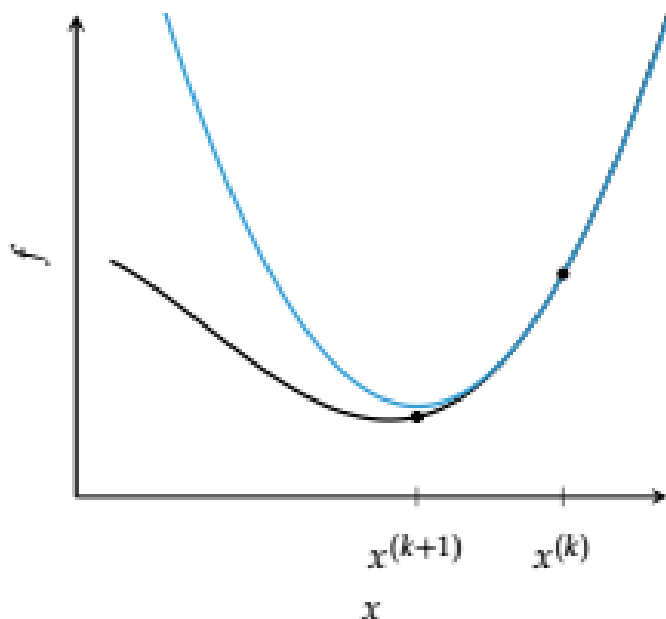
# 第5章无约束最优化方法 (Unconstrained Optimization Methods) - Newton方法-二阶方法



## ◆局部二阶近似

- $q(x) = f(x^{(k)}) + (x - x^{(k)})f'(x^{(k)}) + \frac{(x - x^{(k)})^2}{2}f''(x^{(k)})$
- 令其导数为0，得递推公式： $x^{(k+1)} = x^{(k)} - \frac{1}{f''(x^{(k)})}f'(x^{(k)})$

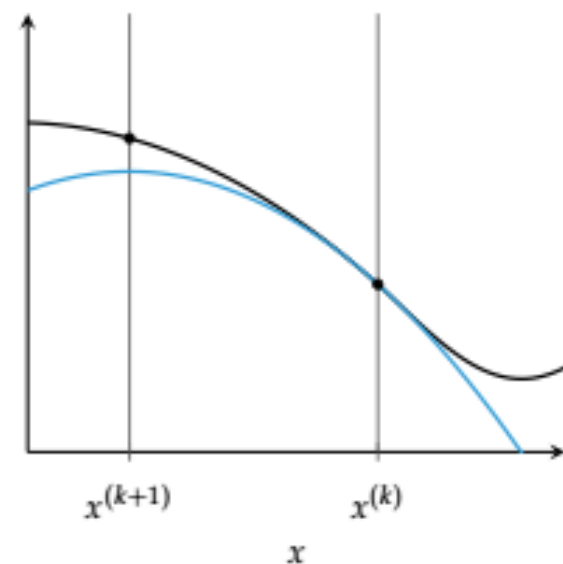
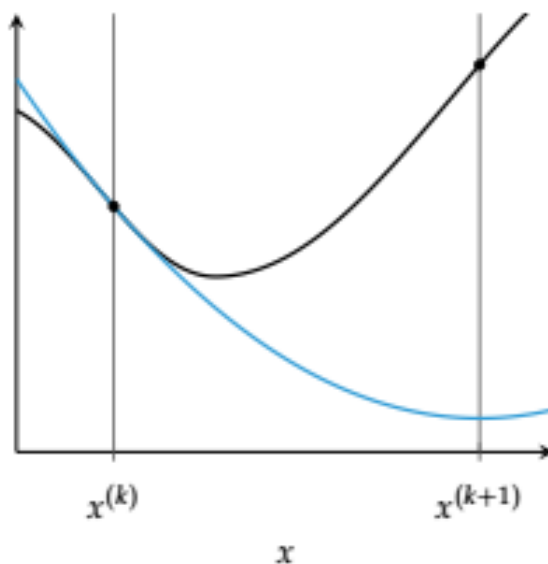
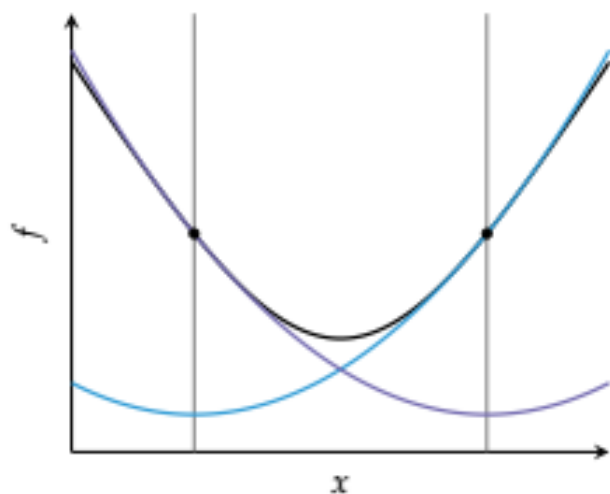
思考：  
可能会  
出现什  
么问题？



# 第5章无约束最优化方法 (Unconstrained Optimization Methods) - Newton方法-二阶方法



## ◆牛顿法的问题



## ◆收敛条件

- 初始点  $x^{(1)} \in I = \{x | x \in [x^* - \delta, x^* + \delta]\}$ ,  $f''(x) \neq 0$ ,
- $f'''(x)$  在  $I$  上连续

$$\frac{1}{2} \left| \frac{f'''(x^{(1)})}{f''(x^{(1)})} \right| < c \left| \frac{f'''(x^*)}{f''(x^*)} \right|, c < \infty$$

用到多维情况，二阶导数用Hessian矩阵替换

# 第5章无约束最优化方法(Unconstrained Optimization Methods) - Newton方法-二阶方法



## ◆实际使用中，可以使用一阶导数来近似二阶导数

$$\text{➤ } f''(x^{(k)}) \approx \frac{(f'(x^{(k)}) - f'(x^{(k-1)}))}{x^{(k)} - x^{(k-1)}}$$

$$\text{➤ } x^{(k+1)} \leftarrow x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)}) \text{---割法(secant)}$$

用到多维情况，二阶导数用Hessian矩阵替换





# 第5章无约束最优化方法(Unconstrained Optimization Methods) – Newton方法



- ◆ 牛顿法(Newton):通过左乘上Hessian矩阵的逆来偏离最速下降法方向来完成求解,本质上是找到函数的一个二次逼近的合适的方向而驱动的

- ◆  $q(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$

- ◆ 令 $q'(x) = 0 \Rightarrow$ 迭代解:  $x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k)$

- ◆ 注意: Newton法可解释为带仿射缩放的最速下降法

- 给定迭代点 $x_k$ ,假设 $H(x_k)$ 对称正定,因而可以进行Cholesky分解,其逆矩阵表示为 $H(x_k)^{-1} = LL^T$ , $L$ 为下三角矩阵,且对角元素为正,现考虑仿射缩放变换 $x = Ly$ ,变换后函数 $F(y) \stackrel{\text{def}}{=} f(Ly)$ ,当前点 $y_k = L^{-1}x_k$ ,因此有 $\nabla F(y_k) = L^T \nabla f(Ly_k) = L^T \nabla f(x_k)$ ,然后沿负梯度方向进行单位步长搜索可得新点

$$y_{k+1} = y_k - L^T \nabla f(x_k)$$

- 这与最速下降法比较,发现只是在方向上左乘了 $L$ .
  - 从这里看到使用合适的伸缩变换的好处。实际上,如果函数 $f$ 是二次的,则上述沿最速下降方向上的单位步长也是该方向上的最优步长,可直接从任意给定解一次迭代到最优解

- ◆ 需要注意的是,这些方法里面都有各自缺陷,把最速下降,牛顿,修正牛顿计

算公式统一:  $x_{k+1} = x_k - \lambda_k H_k \nabla f(x_k)$ ,  $H_k = I$ ;  $H_k = [\nabla^2 f(x_k)]^{-1}$ ;  $H_k =$

$[\nabla^2 f(x_k)]^{-1}$ ,  $\lambda_k$ 用一维搜索则为修正牛顿法;如果利用一阶导数信息来逼近二阶Hessian矩阵信息,则称为拟牛顿法,例如变尺度法中采用近似矩阵来逼近 $H_k = H_{k-1} + C_k$ ,则称为变尺度法,例如 $C_k = t_k \alpha \alpha^T$ ,  $\alpha = (a_1, a_2, \dots, a_n)^T$ ,此时 $C_k$ 秩为1,称为秩1校正,若 $C_k = t_k \alpha \alpha^T + s_k \beta \beta^T$ ,则称为秩2校正,例如后续的DFP方法

# 第5章无约束最优化方法(Unconstrained Optimization Methods)–Newton方法



◆ 牛顿法(Newton) :  $x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k)$

◆ 实际上根据一维情况的推导, 有如下近似关系:

$$\nabla^2 f(x_{k+1})(x_{k+1} - x_k) \approx \nabla f(x_{k+1}) - \nabla f(x_k)$$

◆ 令  $s_k = x_{k+1} - x_k$ ,  $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ , 新的Hessian近似矩阵  $B_{k+1}$  满足如下的割方程 (secant equation)

$$B_{k+1} s_k = y_k$$

◆ 一般要求矩阵  $B_{k+1}$  有一些比较好的性质, 例如

➤ 对称

➤  $B_{k+1} - B_k$  的秩很低, 一般为1 (DFP) 或2 (BFGS)

◆ DFP:  $B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$

◆ BFGS:  $B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$ , 只要初始  $B_0$  正定, 且  $s_k^T y_k > 0$ , BFGS产生正定近似

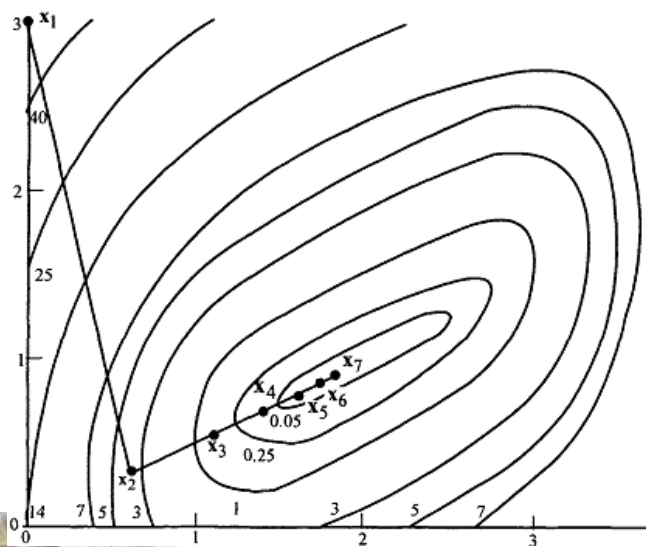
曲率条件

# 第5章无约束最优化方法(Unconstrained Optimization Methods)–Newton方法



◆例:  $Min (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3)

◆七次迭代后终止



Iteration $k$	$\mathbf{x}_k$ $f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)$	$\mathbf{H}(\mathbf{x}_k)$	$\mathbf{H}(\mathbf{x}_k)^{-1}$	$-\mathbf{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$	$\mathbf{x}_{k+1}$
1	(0.00, 3.00) 52.00	(-44.0, 24.0)	$\begin{bmatrix} 50.0 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{384} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 50.0 \end{bmatrix}$	(0.67, -2.67)	(0.67, 0.33)
2	(0.67, 0.33) 3.13	(-9.39, -0.04)	$\begin{bmatrix} 23.23 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{169.84} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 23.23 \end{bmatrix}$	(0.44, 0.23)	(1.11, 0.56)
3	(1.11, 0.56) 0.63	(-2.84, -0.04)	$\begin{bmatrix} 11.50 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{76} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 11.50 \end{bmatrix}$	(0.30, 0.14)	(1.41, 0.70)
4	(1.41, 0.70) 0.12	(-0.80, -0.04)	$\begin{bmatrix} 6.18 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{33.44} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 6.18 \end{bmatrix}$	(0.20, 0.10)	(1.61, 0.80)
5	(1.61, 0.80) 0.02	(-0.22, -0.04)	$\begin{bmatrix} 3.83 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{14.64} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 3.83 \end{bmatrix}$	(0.13, 0.07)	(1.74, 0.87)
6	(1.74, 0.87) 0.005	(-0.07, 0.00)	$\begin{bmatrix} 2.81 & -4.0 \\ -4.0 & 8.0 \end{bmatrix}$	$\frac{1}{6.48} \begin{bmatrix} 8.0 & 4.0 \\ 4.0 & 2.81 \end{bmatrix}$	(0.09, 0.04)	(1.83, 0.91)
7	(1.83, 0.91) 0.0009	(0.0003, -0.04)				

◆从中看到目标函数值每次都下降，但实际上并不一定，函数 $f$ 并不能用为下降函数，但如果初始点足够靠近最优优点，则Newton法可达二次收敛

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) – Newton方法



- ◆ 不收敛的原因是 $H(x_k)$ 可能是奇异的, 这时 $x_{k+1}$ 并未很好的定义。甚至即使 $H(x_k)^{-1}$ 存在,  $f(x_{k+1})$ 也不必定小于 $f(x_k)$
- ◆ 但如果初始点足够接近最优解, 则 $H(\bar{x})$ 满秩, 此时牛顿法符合定义并收敛到最优点
- ◆ **定理:** 令函数 $f: R^n \rightarrow R$ 连续二次可微. 牛顿法中令 $\bar{x}$ 是梯度为0的点, 且 $H(\bar{x})^{-1}$ 存在。令初始点 $x_1$ 足够接近 $\bar{x}$ : 存在 $k_1, k_2 > 0$ , 且 $k_1 k_2 \|x_1 - \bar{x}\| < 1$ , 使得
  - $\|H(\bar{x})^{-1}\| \leq k_1$
  - $\|\nabla f(\bar{x}) - \nabla f(x) - H(x)(\bar{x} - x)\| \leq k_2 \|\bar{x} - x\|^2$对任意满足 $\|x - \bar{x}\| \leq \|x_1 - \bar{x}\|$ 的 $x$ 都成立。则Newton算法至少以二次速率超线性收敛到 $\bar{x}$

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-LM和Trust Region方法



- ◆ Levenberg-Marquardt 法
- ◆ 前面提到Newton由于初始点造成Hessian矩阵奇异，或者方向不是下降方向，或者是下降方向但单位步长不保证下降，对于最后的情况，可以通过线搜索达到下降。但更关键的问题是是否有定义好的算法不考虑初始点的情况，可以收敛到0梯度点呢，也就是全局收敛性，看下面的变化！
- ◆ 考虑 $x, d = -B\nabla f(x)$ ,  $B$ 是后面要确定的对称正定矩阵， $y = x + \hat{\lambda}d$ 为线搜索最优解对应的迭代点
- ◆  $B = (\epsilon I + H)^{-1}, H = H(x), \epsilon > 0$ 要求满足：
- ◆ 给定 $\delta > 0$ ， $\epsilon$ 是满足 $(\epsilon I + H)$ 的特征值都大于等于 $\delta$ 的最小的标量

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-LM和Trust Region方法



- ◆ 首先验证是否是下降方向:  $\nabla f(x)^T d = -\nabla f(x)^T B \nabla f(x) < 0$ , 因此  $f(y) < f(x)$
- ◆ 因此  $f$  确实是下降函数
- ◆ 注意如果  $H(\bar{x})$  的最小特征值大于等于  $\delta$ , 则算法生成收敛到  $\bar{x}$  的数列, 且  $\epsilon_k = 0$ , 因此  $d_k = -H(x_k)^{-1} \nabla f(x_k)$ ,
- ◆ 算法化为Newton法, 因此方法具有二阶收敛速率
- ◆ 注意  $\delta$  性质的影响
  - 太小, 可确保算法渐进二次收敛速率, 但在接近Hessian奇异处的点可能发病态性
  - 太大,  $B$  对角占优, 算法行为与最速下降法类似, 仅具有线性收敛速率
- ◆ 用如下迭代:  $[\epsilon_k I + H(x_k)](x_{k+1} - x_k) = -\nabla f(x_k)$  替代Newton法, 称为LM方法, 参数0.25, 0.75, 2, 4运行较好





# 第5章无约束最优化方法(Unconstrained Optimization Methods)-LM和Trust Region方法



- ◆ 给定迭代和参数  $x_k, \epsilon_k > 0$ , 首先确认  $\epsilon_k I + H(x_k)$  的正定性, 构造其Cholesky分解  $LL^T$ . 如果不成功, 则用  $\epsilon_k \leftarrow 4\epsilon_k$  重复
- ◆ 然后解  $LL^T(x_{k+1} - x_k) = -\nabla f(x_k)$  得  $x_{k+1}$ , 计算  $R_k = \frac{f(x_k) - f(x_{k+1})}{q(x_k) - q(x_{k+1})}$  作为二次式  $q$  对  $f$  的近似程度的预测,  $R_k$  越接近1, 近似越好,  $\epsilon_k$  可以越小, 据此

$$\begin{cases} \epsilon_{k+1} = 4\epsilon_k, R_k < 0.25 \\ \epsilon_{k+1} = \frac{\epsilon_k}{2}, R_k > 0.75, \text{ 而且若 } R_k \leq 0, \text{ 说明没有改进,} \\ \epsilon_{k+1} = \epsilon_k, \text{ otherwise} \end{cases}$$

重设  $x_{k+1} = x_k$ , 否则保留计算的  $x_{k+1}$

➤  $k \leftarrow k + 1$  直至结束

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-LM和Trust Region方法



## ◆上面的方法与信任域方法非常类似，信任域方法又称为受限制步长方法

- 牛顿法的主要困难就是信任域内并没有包含解，虽然该区域内的二次逼近函数在点 $x_k$ 认为是足够可靠的
- 为了克服这问题，考虑如下信任域子问题
- $\text{Min} \{q(x): x \in \Omega_k\}$ ,  $q$ 为 $f$ 在点 $x_k$ 的二次逼近， $\Omega_k = \{x: \|x - x_k\| \leq \Delta_k\}$ 为带信任参数 $\Delta_k > 0$ 的信任区域，注意这里的范数为 $l_2$ 范数，若为 $l_\infty$ ，则称为Box-step, Hypercube方法)
- 跟前述一样，定义 $R_k$ 为实际与预期下降的比率。如果 $R_k$ 相对于1太小，则需要减小信任域，否则可以扩展信任域.下次迭代设定 $\Delta_{k+1}$

$$\Delta_{k+1} = \begin{cases} \frac{\|x_{k+1} - x_k\|}{4}, & R_k < 0.25 \\ 2\Delta_k, & R_k > 0.75 \\ \Delta_k, & \text{otherwise} \end{cases}$$

若 $R_k \leq 0$ ,  $f$ 在本次迭代中并没有改进，重设 $x_{k+1} = x_k$ , 然后 $k \leftarrow k + 1$ , 直至达到0梯度点。如果 $H(\bar{x})$ 正定，则对足够大的 $k$ , 信任域界无效，方法退化为Newton方法，具有二阶收敛速率



## ◆上述讨论提及两点

➤ 无论上述 $f$ 的二次表示中什么地方使用Hessian矩阵, 实际操作中都可以使用Hessian的近似矩阵来替换, 如后面的拟牛顿法

➤ 将 $\delta = x - x_k$ , 信任域子问题可写为

➤ (5-4)  $\text{Min} \{ \nabla f(x_k)^T \delta + \frac{1}{2} \delta^T H(x_k) \delta : \frac{1}{2} ||\delta||^2 \leq \frac{1}{2} \Delta_k^2 \}$

➤ 上述问题的KKT条件要求非负Lagrange乘子, 以及主可行解 $\delta$ 满足除互补松弛性条件外的下列条件

$$[H(x_k) + \lambda I] \delta = -\nabla f(x_k)$$

➤ 这与LM方法类似。尤其如果 $\Delta_k = -[H(x_k) + \epsilon_k I]^{-1} \nabla f(x_k)$ , 这里 $H(x_k) + \epsilon_k I$ 正定, 容易验证 $\delta = x_{k+1} - x_k$ ,  $\lambda = \epsilon_k$ 满足 (5-4) 的最优性条件

3/23 2022 ◆ 因此, LM方法可以看做是一种信任域方法

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-拟牛顿和共轭方向法



◆ 如果目标函数是二次的, 则从任何共轭方向搜索, 可在 $n$ 步之内达到极小点

◆  $f(x) = c^T x + \frac{1}{2} x^T H x$ ,  $H$ 对称正定, 假设 $d_1, d_2, \dots, d_n$ 关于 $H$ 共轭, 则给定初始点 $x_1$ , 任意点 $x$ 可唯一表示为 $x = x_1 + \sum_{j=1}^n \lambda_j d_j$ , 从而 $f(x)$ 表示为 $\lambda$ 的函数, 最后化简为 $F(\lambda) = \sum_{j=1}^n \left[ c^T (x_1 + \lambda_j d_j) + \frac{1}{2} (x_1 + \lambda_j d_j)^T H (x_1 + \lambda_j d_j) \right]$

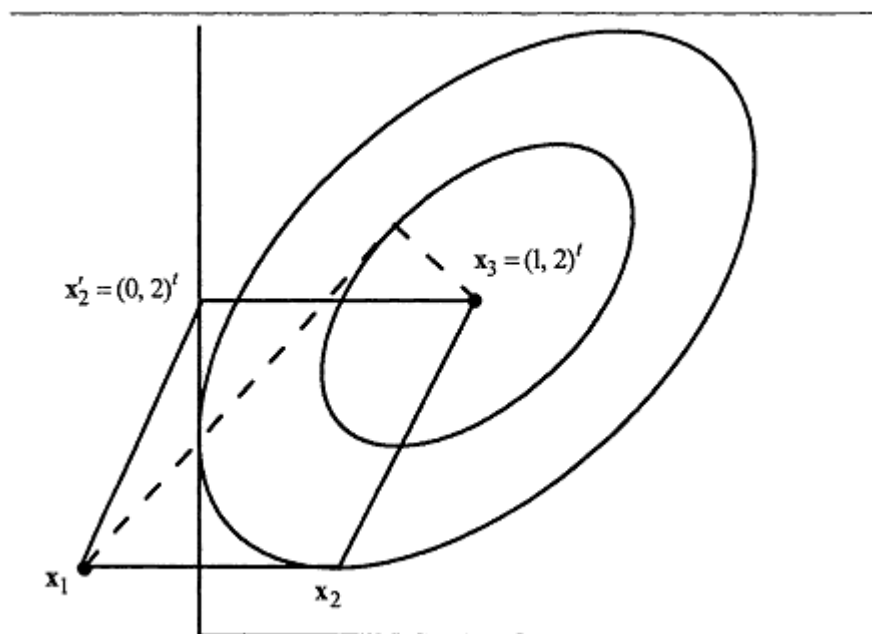
◆ 对每个求和项求导数得:  $\lambda_j^* = -\frac{[c^T d_j + x_1^T H d_j]}{d_j^T H d_j}$ ,  $j = 1, \dots, n$ , 因此由任何方向开始都可得最优解

◆ 例:  $\text{Min } -12x_2 + 4x_1^2 + 4x_2^2 + 4x_1x_2$ ,  $H = \begin{bmatrix} 8 & -4 \\ -4 & 8 \end{bmatrix}$ , 假设 $d_1^T = (1, 0)$ , 求得共轭方向 $d_2^T = (k, 2k)$ , 不妨令为 $(1, 2)$ , 此时从初始点 $x_1^T = (-\frac{1}{2}, 1)$ 沿方向 $d_1$ 最小化目标函数得 $x_2^T = (\frac{1}{2}, 1)$ , 然后从该点沿 $d_2$ 最小化目标函数得 $x_3^T = (1, 2)$ , 为最小点

## 第5章无约束最优化方法 (Unconstrained Optimization Methods) - 拟牛顿和共轭方向法



◆从图中可以看出沿任何方向都可以在2步之内达到最优

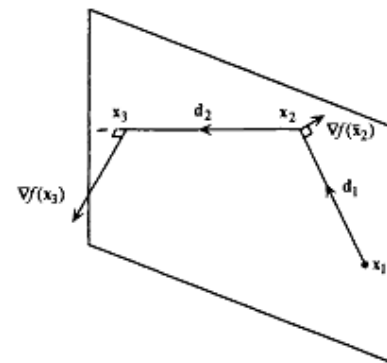


◆这个结果对二次函数一般都正确。事实就是一般函数在最优点附近可以由二次函数很好的逼近，从而使得共轭在二次和非二次函数中非常有用

# 第5章无约束最优化方法 (Unconstrained Optimization Methods) - 拟牛顿和共轭方向法



- ◆ 下面的定理表明, 如果我们从 $x_1$ 开始, 在每步中 $x_{k+1}$ 由在包含 $x_1$ , 由向量 $d_1, d_2, \dots, d_k$ 张成的线性子空间上最小化 $f$ 得到。而且梯度 $\nabla f(x_{k+1})$ 若不为0, 则与这个子空间正交. 这有时称为扩张子空间属性
- ◆ **定理:**  $f(x) = c^T x + \frac{1}{2} x^T H x$ ,  $H$ 对称正定,  $d_1, d_2, \dots, d_n$ 关于 $H$ 共轭,  $x_1$ 为任意开始点,  $\lambda_k$ 为 $\min f(x_k + \lambda d_k)$ 的最优解,  $\lambda \in R$ , 令 $x_{k+1} = x_k + \lambda_k d_k$ , 则必定有



- $\nabla f(x_{k+1})^T d_j = 0, j = 1, \dots, k$
- $\nabla f(x_1)^T d_k = \nabla f(x_k)^T d_k$
- $x_{k+1}$  是  $\min f(x), s.t. x - x_1 \in L(d_1, \dots, d_k)$ , 表示线性子空间, 即  $L(d_1, \dots, d_k) = \{\sum_{j=1}^k \mu_j d_j : \mu_j \in R\}$ ,  $x_{n+1}$  是  $f$  在  $R^n$  上的最小点



## 第5章无约束最优化方法(Unconstrained Optimization Methods)-拟牛顿和共轭方向法



- ◆ 怎么来生成二次型的共轭方向：这些方法很自然就导致了各种最小化二次函数和非二次函数的算法
- ◆ Quasi-Newton: 拟牛顿法, Davidon-Fletcher-Powell 1959年Davidon提出, 后来1963年由Fletcher和Powell开发, DFP方法搜索方向为:
- ◆  $d_j = -D_j \nabla f(y)$ , 来代替牛顿法中的  $-H^{-1}(y) \nabla f(y)$
- ◆ 即负梯度方向左乘  $D_j$ , 这里  $D_j$  为Hessian矩阵的逆矩阵的逼近矩阵, 对称正定。正定型确保  $d_j$  是下降方向, 只要  $\nabla f(y) \neq 0, d_j^T \nabla f(y) < 0$ , 下一步  $D_{j+1}$  通过给  $D_j$  增加两个秩为1的对称矩阵来形成, 这个过程有时称为秩2校正过程(Rank-two correction procedure)

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-拟牛顿和共轭方向法

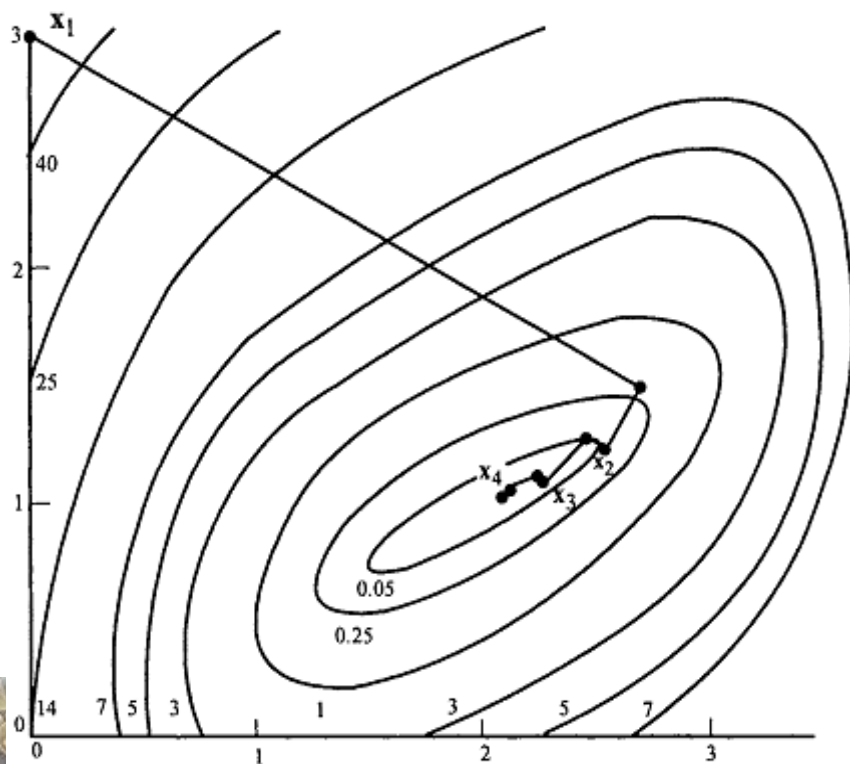


- ◆对于二次函数，这种更新过程在 $n$ 步之内产生实际Hessian逆矩阵的精确表示；如果拟牛顿方法中允许二次逼近为不定的，则称为割法(Secant Method)
- ◆DFP方法过程与以前类似，这里只写出更新 $D_{j+1}$ 的步骤
- ◆
$$D_{j+1} = D_j + \frac{p_j p_j^T}{p_j^T q_j} - \frac{D_j q_j q_j^T D_j}{q_j^T D_j q_j},$$
 这里 $p_j = \lambda_j d_j = y_{j+1} - y_j$ ,  $q_j = \nabla f(y_{j+1}) - \nabla f(y_j)$
- ◆如果上述过程中内部每 $n$ 步重置算法，如果重置间隔 $n'$ 小于 $n$ ，则称为部分拟牛顿算法，若 $n' \ll n$ ,可减少存储，因为这是只需存储 $p_j, q_j$

# 第5章无约束最优化方法(Unconstrained Optimization Methods)-拟牛顿和共轭方向法



◆例DFP求解:  $Min (x_1 - 2)^4 + (x_1 - 2x_2)^2$ , 初始点(0, 3)



$x_k$ $f(x_k)$	$j$	$y_j$ $f(y_j)$	$\nabla f(y_j)$	$\ \nabla f(y_j)\ $	$D_j$	$d_j$	$\lambda_j$	$y_{j+1}$
(0.00, 3.00) 52.00	1	(0.00, 3.00) 52.00	(-44.00, 24.00)	50.12	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	(44.00, -24.00)	0.062	(2.70, 1.51)
	2	(2.70, 1.51) 0.34	(0.73, 1.28)	1.47	$\begin{bmatrix} 0.25 & 0.38 \\ 0.38 & 0.81 \end{bmatrix}$	(-0.67, -1.31)	0.22	(2.55, 1.22)
(2.55, 1.22) 0.1036	1	(2.55, 1.22) 0.1036	(0.89, -0.44)	0.99	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	(-0.89, 0.44)	0.11	(2.45, 1.27)
	2	(2.45, 1.27) 0.0490	(0.18, 0.36)	0.40	$\begin{bmatrix} 0.65 & 0.45 \\ 0.45 & 0.46 \end{bmatrix}$	(-0.28, -0.25)	0.64	(2.27, 1.11)
(2.27, 1.11) 0.008	1	(2.27, 1.11) 0.008	(0.18, -0.20)	0.27	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	(-0.18, 0.20)	0.10	(2.25, 1.13)
	2	(2.25, 1.13) 0.004	(0.04, 0.04)	0.06	$\begin{bmatrix} 0.80 & 0.38 \\ 0.38 & 0.31 \end{bmatrix}$	(-0.05, -0.03)	2.64	(2.12, 1.05)
(2.12, 1.05) 0.0005	1	(2.12, 1.05) 0.0005	(0.05, -0.08)	0.09	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	(-0.05, 0.08)	0.10	(2.115, 1.058)
	2	(2.115, 1.058) 0.0002	(0.004, 0.004)	0.006				

## 第5章无约束最优化方法 (Unconstrained Optimization Methods) - 拟牛顿和共轭方向法



- ◆ 上述方法形成的方向都是下降方向。对于二次目标函数而言，DFP生成的方向都是共轭的，一次完整迭代即可达到最优点，并且其迭代最终矩阵  $D_{n+1}$  正好是Hessian矩阵  $H$  的逆矩阵
- ◆ 注意到DFP方法中只是采用  $D_j$  来近似Hessian矩阵的逆矩阵，并用其来左乘牛顿法中的梯度方向，然后执行线搜索
- ◆ 此外可以思考  $D_{n+1} = H^{-1}$ ，每一步中  $D_{j+1}H$  都有为1的特征值，因此方法的每一步修改的逼近累积附加的一个线性独立的特征向量，其特征值对应  $D_{j+1}H$  的单位特征值
- ◆ 实际上对于  $D_{j+1}$  的生成，有很多方法，只需要保证对称正定，不同的方式对应不同的方法，例如Broyden族，BFGS更新方法等，请参考相关文献



- ◆ 问题  $P: \min f(x), x \in X, f: R^n \rightarrow R$  为凸函数但不必是可微的函数,  $X$  为  $R^n$  中的非空闭凸集. 假设存在最优解.
- ◆ 次梯度优化算法看作是最速下降法的一个直接推广, 其负梯度方向用负的基于次梯度的方向进行替换!
- ◆ 但这方向不必是下降方向, 但如果步长足够小, 确实将使新的迭代点更靠近最优解, 因此也不执行线搜索! 而是在每次迭代时规定一个步长, 保证生成的点列最终收敛到最优解.
- ◆  $f: R^n \rightarrow (-\infty, \infty], g \in R^n$  为函数  $f$  在点  $x \in \text{dom}(f)$  的次梯度:  $f(z) \geq f(x) + g' \cdot (z - x), \forall z \in R^n$ , 次梯度集合称为次微分:  $\partial f(x)$



- ◆ 问题  $P: \min f(x), x \in X, f: R^n \rightarrow R$  为凸函数但不必是可微的函数,  $X$  为  $R^n$  中的非空闭凸集. 假设存在最优解.
- ◆ 给定迭代  $x^{(k)} \in X$ , 步长  $\lambda_k$ , 方向  $d_k = -\frac{g_k}{\|g_k\|}$ , 方向  $g_k \in \partial f(x^{(k)})$ , 新的点  $\overline{x^{(k+1)}} = x^{(k)} + \lambda_k \cdot d_k \notin X$ ! 不可行!
- ◆ 投影法(projecting): 投影  $\overline{x^{(k+1)}}$  到凸集  $X$  上, 也就是找到  $X$  中与  $\overline{x^{(k+1)}}$  最近的点  $x^{(k+1)} = P_X(\overline{x^{(k+1)}})$ , 其中  $P_X(\bar{x}) = \operatorname{argmin}\{\|x - \bar{x}\|: x \in X\}$
- ◆ 投影定理:  $C \in R^n$  为非空闭凸集,  $z \in R^n$ , 则存在唯一向量  $\min_{x \in C} \|z - x\|$ , 称该向量为  $z$  在集合  $C$  上的投影. 而且向量  $x^*$  为  $z$  在集合  $C$  上的投影  $\Leftrightarrow (z - x^*)'(x - x^*) \leq 0, \forall x \in C$





## ◆ 实际上这些投影当 $X$ 很简单时，投影的计算也非常简单

➤ 例. 若  $X \geq 0$ , 则  $x_i^{(k+1)} = \max \left\{ 0, \left( \overline{x^{(k+1)}} \right)_i \right\}$

➤ 若  $X = \{x: l_i \leq x_i \leq u_i, i = 1, 2, \dots, n\}$ , 则

$$\text{➤ } x_i^{(k+1)} = \begin{cases} \left( \overline{x^{(k+1)}} \right)_i & l_i \leq \left( \overline{x^{(k+1)}} \right)_i \leq u_i \\ l_i & \left( \overline{x^{(k+1)}} \right)_i < l_i \\ u_i & \left( \overline{x^{(k+1)}} \right)_i > u_i \end{cases}, i = 1, 2, \dots, n$$





## ◆ 次梯度算法

- 初始化：选择初始点  $x^{(1)} \in X, x^* = x^{(1)}, k = 1, UB^{(1)} = f(x^{(1)})$ ;
- 迭代求解：给定  $x^{(k)}$ , 计算  $g_k \in \partial f(x^{(k)})$ .
  - ✓ **IF**  $g_k = 0$ , Stop;  $x^{(k)}$  求解问题  $P: \min f(x), x \in X$ . 得  $x^*$ .
  - ✓ **ELSE** 令  $d_k = -\frac{g_k}{\|g_k\|}$ , 选择步长  $\lambda_k > 0$ , 计算  $x^{(k+1)} = P_X(\overline{x^{(k+1)}}), \overline{x^{(k+1)}} = x^{(k)} + \lambda_k \cdot d_k$ .
    - **IF**  $f(x^{(k+1)}) < UB^{(k)}$ , 令  $UB^{(k+1)} = f(x^{(k+1)}), x^* = x^{(k+1)}$ .
    - **ELSE** 令  $UB^{(k+1)} = UB^{(k)}$ .
- ✓  $k = k + 1$ , 迭代!

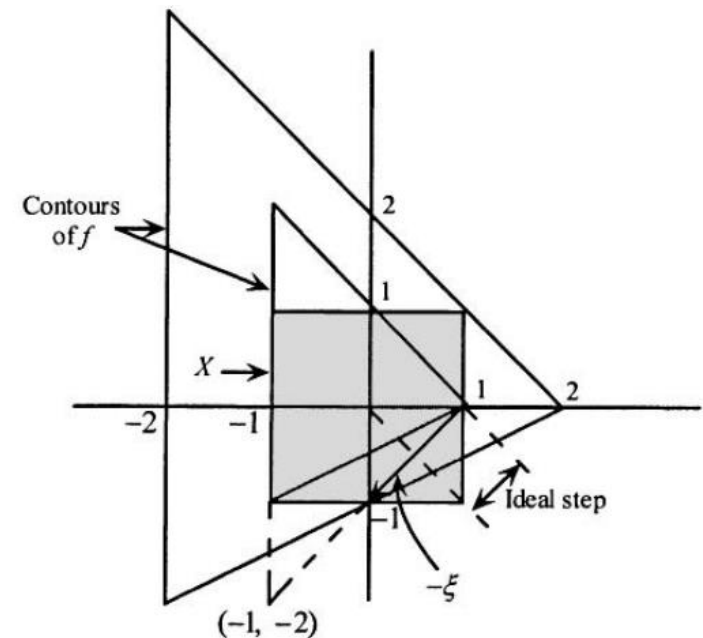
◆ 注意停止准则  $g_k = 0$  可能从来不会实现，即使存在一个内部最优点，并且我们对于  $0 \in \partial f(x^{(k)})$  也确实找到了一个解  $x^{(k)}$ ，因为算法任意选择次梯度  $g_k$ 。因此也需要最大迭代次数的限制!

# 第5章无约束最优化方法 (Unconstrained Optimization Methods)-次梯度优化方法 (Subgradient Optimization)



- ◆ 注意, 当 $x^{(k+1)} = x^{(k)}$ 时, 也能终止迭代. 如果最优目标函数值 $f^*$ 已知, 则 $UB^{(k)} \leq f^* + \epsilon$ 也可以作为停止准则.
- ◆ 例:  $\min \{f(x, y): -1 \leq x \leq 1, -1 \leq y \leq 1\}, f(x, y) = \max\{-x, x + y, x - 2y\}$
- ◆ 解: 考虑 $f(x, y) \leq c, c$ 为常数, 然后检查约束区域 $\{-x \leq c, x + y \leq c, x - 2y \leq c\}$ , 画出 $f$ 的轮廓如图所示. 注意不可微点都是 $(t, 0), (-t, 2t), (-t, -t), t \geq 0$ 这样的点. 最优解解为 $(x, y) = (0, 0)$ . 因此尽管 $(0, 0)^t \in \partial f(0)$ , 显然也有 $(-1, 0)^t, (1, 1)^t, (1, -2)^t \in \partial f(0)$ .

- 考虑点 $(x, y) = (1, 0)$ ,  $f(1, 0) = 1$ , 由 $x + y$ 与 $x - 2y$ 确定. 因此 $g = \xi = (1, 1)^t \in \partial f(1, 0)$ , 考虑 $-g = -\xi = (-1, -1)^t$ , 这不是下降方向
- 但沿这个方向移动, 确实接近最优解 $(0, 0)^t$ . 如图所示, 沿方向 $d = -g = -\xi$ , 达到离最优解最近的点
- 但若选步长为 $\lambda = 2$ , 导致 $(1, 0) - 2 \cdot (1, 1) = (-1, -2)$  投影 $P_X(-1, -2) = (-1, -1)$ . 这构成上述算法的一次迭代.

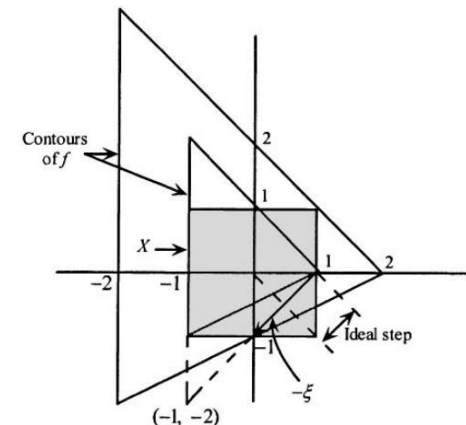


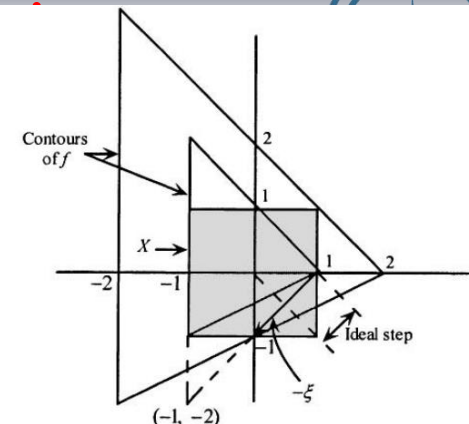


◆ **定理：** 问题  $P: \min f(x), x \in X$  如前所述，假设存在最优解。考虑前述的次梯度优化算法来求解问题  $P$ 。假设前述非负步长  $\{\lambda_k\}$  满足条件： $\{\lambda_k\} \rightarrow 0^+, \sum_{k=0}^{\infty} \lambda_k = \infty$ 。则要么**算法有限步终止于最优解**，或者**生成的无穷序列极限为最优解**，即满足： $\{UB^{(k)}\} \rightarrow f^* = \min\{f(x): x \in X\}$

◆ 定理理论上为真，但实际实行并没那么好。例如选择  $\lambda_k = \frac{1}{k}$ ，显然满足条件，但几千步以后仍然远离最优解。

◆ 如何选择步长为好？？？





◆ 观察例子中的图

◆ 理想步长满足新的点与方向  $d_k$  正交

◆ 即:  $d_k^t \cdot [x^{(k)} + \lambda_k \cdot d_k - x^*] = 0$ , 求出理想步长为

◆  $\lambda_k^* = (x^* - x^{(k)})^t d_k = \frac{(x^{(k)} - x^*)^t g_k}{\|g_k\|}$ , 但此时最优点  $x^*$  未知, 怎么办? 逼近思想!

◆  $f$  是凸的,  $f^* = f(x^*) \geq f(x^{(k)}) + (x^* - x^{(k)})^t g_k$ , 从而有  $\lambda_k^* \geq \frac{f(x^{(k)}) - f^*}{\|g_k\|}$ , 但这是  $f^*$  也未知, 因此推荐使用  $f^*$  的一个

低估值  $\bar{f}$  来代替  $f^*$ , 从而步长选择为:  $\lambda_k = \frac{\beta_k (f(x^{(k)}) - \bar{f})}{\|g_k\|}$ , 一

般选  $\beta_k \in (\epsilon_1, 2 - \epsilon_2]$ . 一般可达线性收敛!

# 第5章无约束最优化方法 (Unconstrained Optimization Methods)



## ◆ 使用导数和不使用导数的线搜索方法

- Dichotomous, Golden Section method, Fibonacci method
- Bisection search, Newton's method

## ◆ 多维搜索下使用导数和不使用导数的搜索方法

- The Cyclic coordinate method, Hooke and Jeeves, Rosenbrock's method
- The Steepest descent and The method of Newton

## ◆ 牛顿方法的变种：LM和信任域方法

- Levenberg-Marquardt, Trust Region Methods

## ◆ 共轭方向法：拟牛顿法和共轭梯度法

- 目标函数如果是二次的，有限步内可以收敛

## ◆ 次梯度优化方法

- 不可微目标函数中的Steepest Descent algorithm: 投影思想!

## ◆ 主要掌握其基本思想： $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$ ,

3/23/2022 方向： $p_k = -B_k^{-1} \nabla f_k$ ,  $x^{(k+1)} = x^{(k)} + \alpha_k p_k$