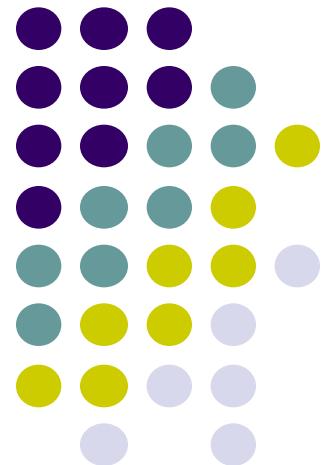


最优化方法 (III)

张宏鑫

2019-04-09

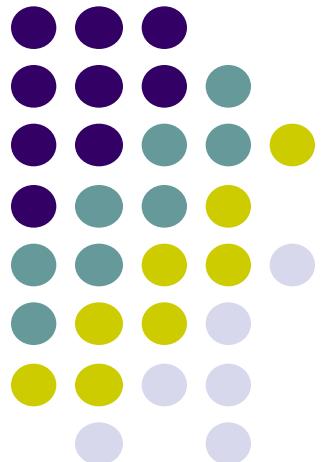
浙江大学计算机学院



签到

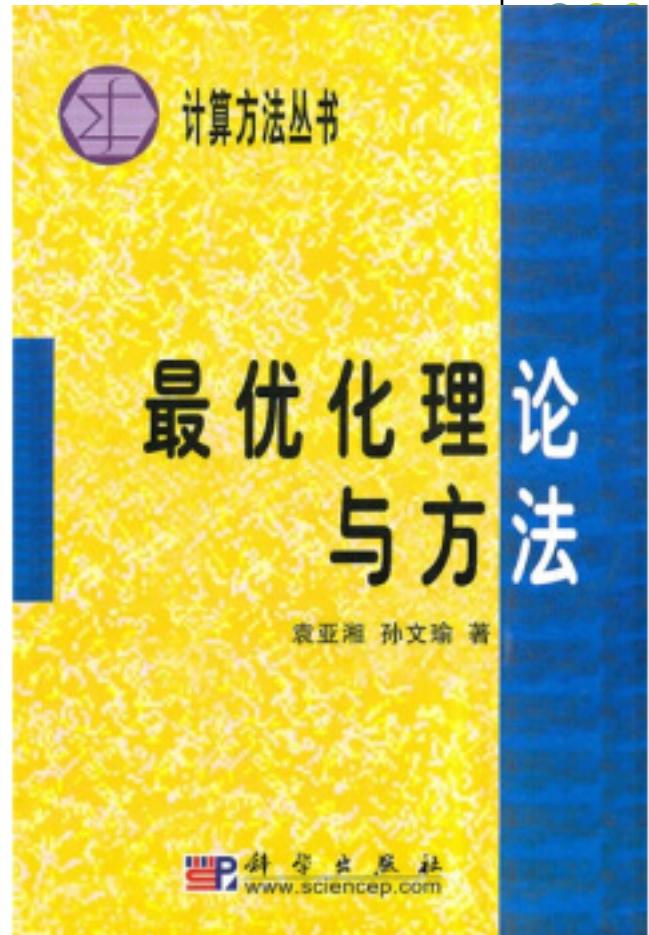


1. 请关注微信公众号: **RGB-Group**
2. 在该公众号中回复: **csmath**
3. 在该公众号中回复: 姓名_学号
4. 三次不同时间输入: 本教室位置

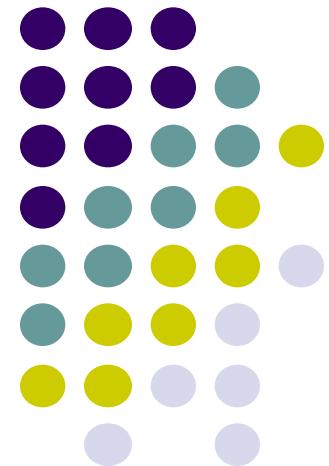


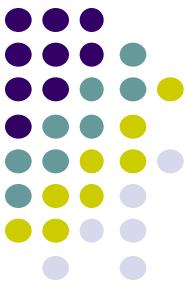
内容

- 线性规划
- 非线性优化
- 主要参考书：
 - 《线性规划》
 - 张建中，许绍吉，科学出版社
 - 《最优化理论与方法》
 - 袁亚湘，孙文瑜，科学出版社
 - 《数学规划》
 - 黄红选，韩继业，清华大学出版社



二、非线性最优化





引言

最优化的问题的一般形式为

$$\begin{aligned} & \text{Min } f(\mathbf{x}) \\ & \text{s.t. } \mathbf{x} \in X \end{aligned}$$

$f(\mathbf{x})$ 为目标函数， $X \subseteq E^n$ 为可行域。

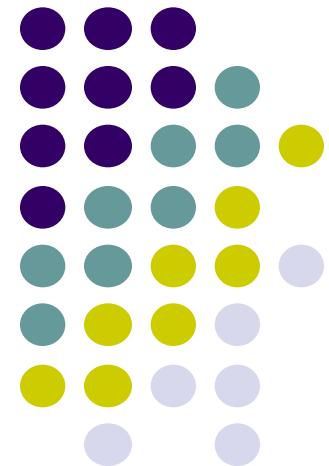
如 $X = E^n$ ，则以上最优化问题为**无约束最优化问题**

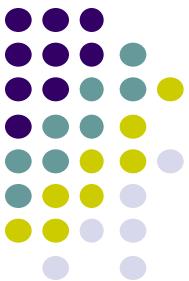
约束最优化问题通常写为

$$\begin{aligned} & \text{Min } f(\mathbf{x}) \\ & \text{s.t. } c_i(\mathbf{x}) = 0, i \in E, \\ & \quad c_i(\mathbf{x}) \geq 0, i \in I, \end{aligned}$$

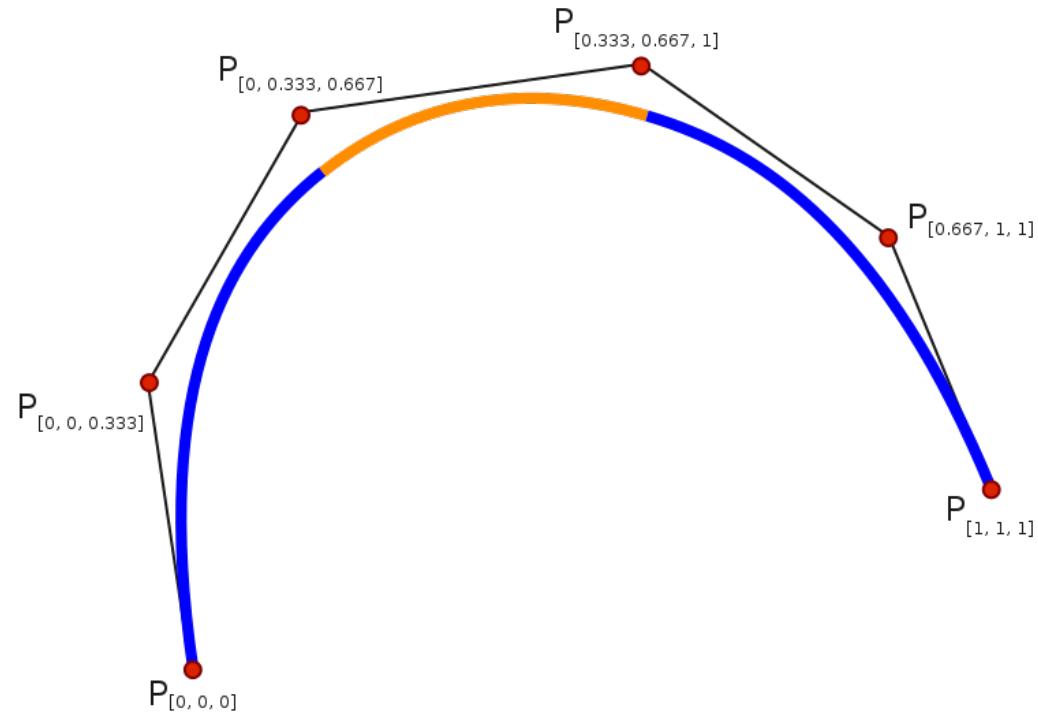
其中 E, I 分别为等式约束的指标集和不等式约束的指标集， $c_i(\mathbf{x})$ 是约束函数。

1. 无约束非线性最优化

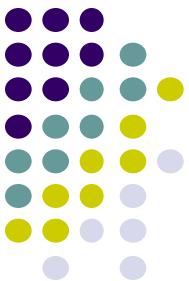




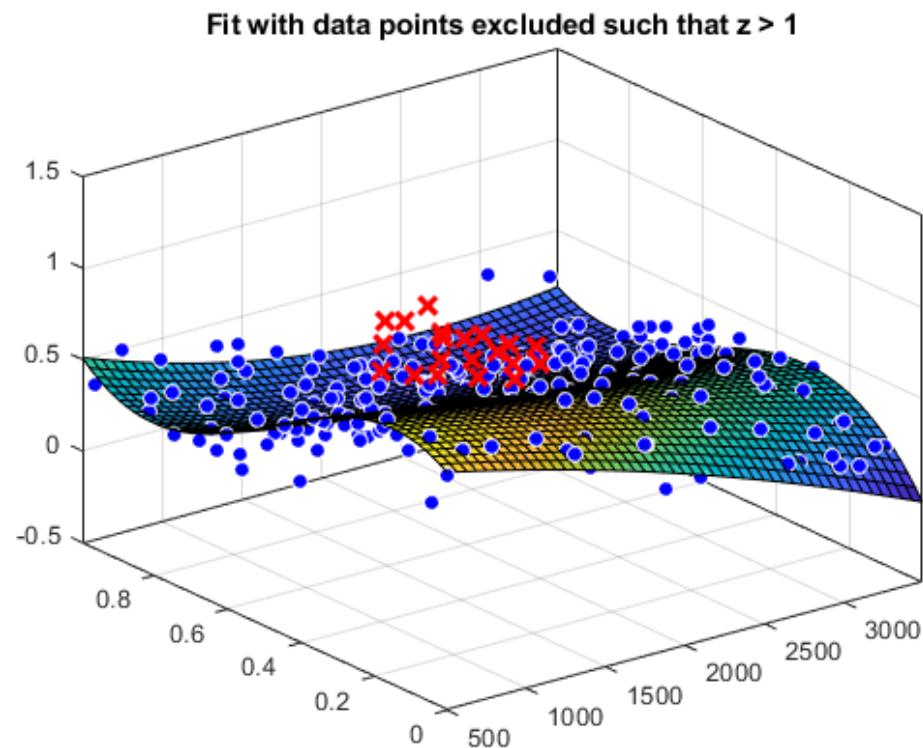
Nonlinear data and shapes



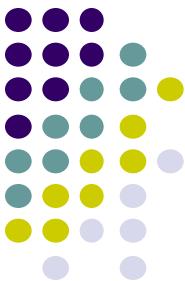
Spline Curve



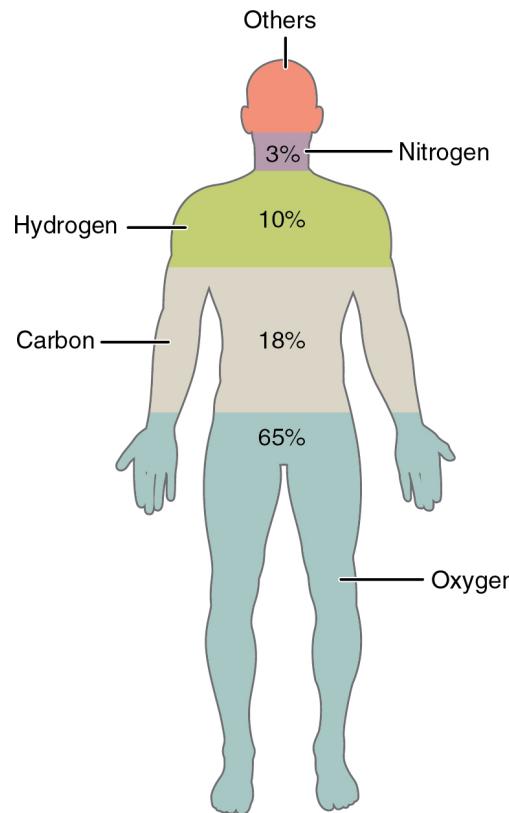
非线性的数据无处不在



Nonlinear Surface

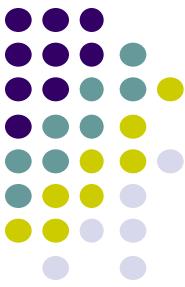


非线性的数据无处不在



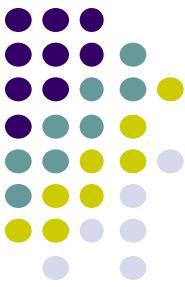
| Element | Symbol | Percentage in Body |
|--|--------|--------------------|
| Oxygen | O | 65.0 |
| Carbon | C | 18.5 |
| Hydrogen | H | 9.5 |
| Nitrogen | N | 3.2 |
| Calcium | Ca | 1.5 |
| Phosphorus | P | 1.0 |
| Potassium | K | 0.4 |
| Sulfur | S | 0.3 |
| Sodium | Na | 0.2 |
| Chlorine | Cl | 0.2 |
| Magnesium | Mg | 0.1 |
| Trace elements include boron (B), chromium (Cr), cobalt (Co), copper (Cu), fluorine (F), iodine (I), iron (Fe), manganese (Mn), molybdenum (Mo), selenium (Se), silicon (Si), tin (Sn), vanadium (V), and zinc (Zn). | | less than 1.0 |

Human body

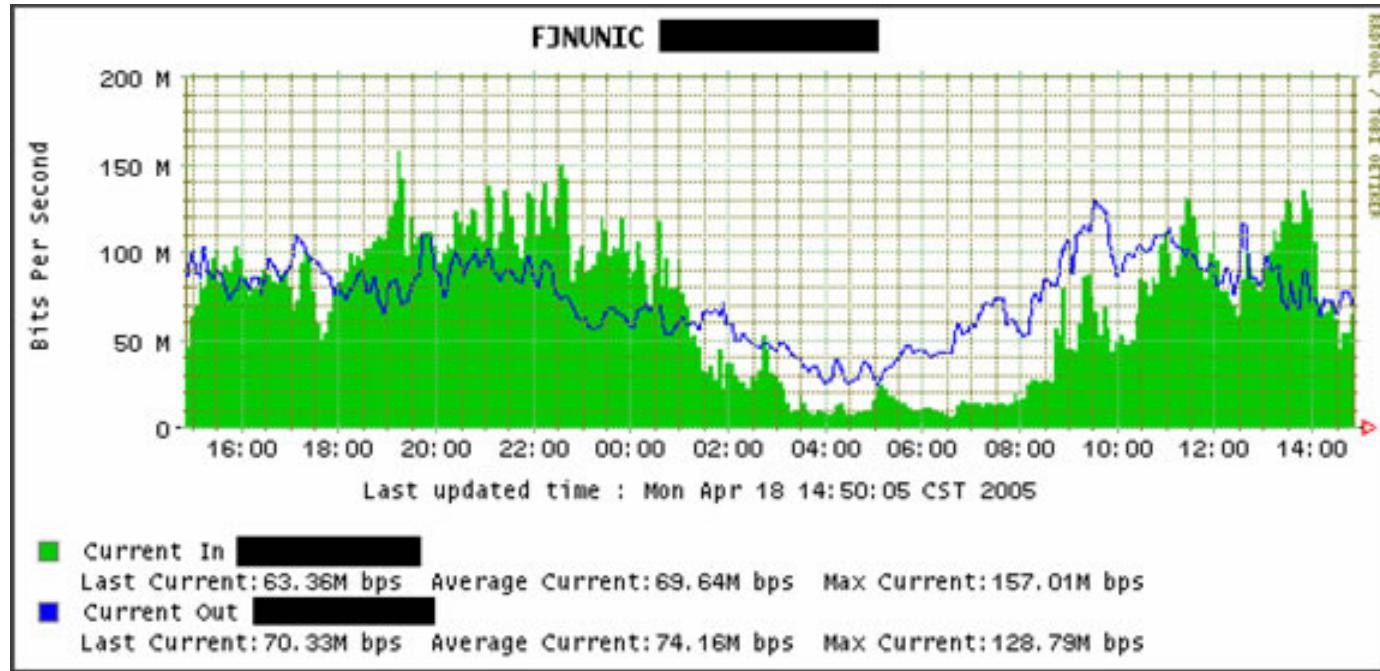


非线性的数据无处不在

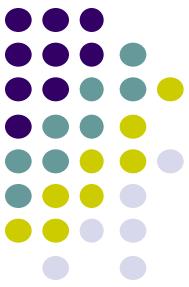




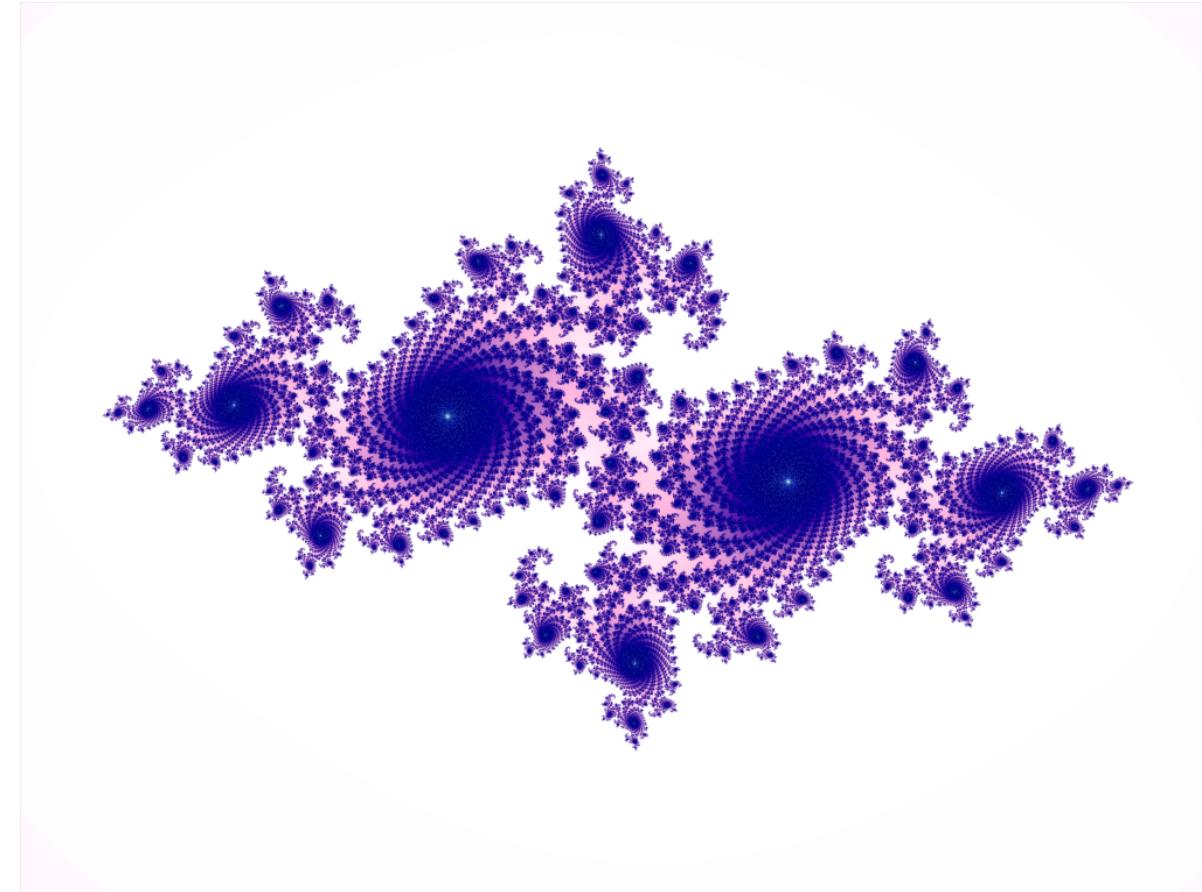
非线性的数据无处不在



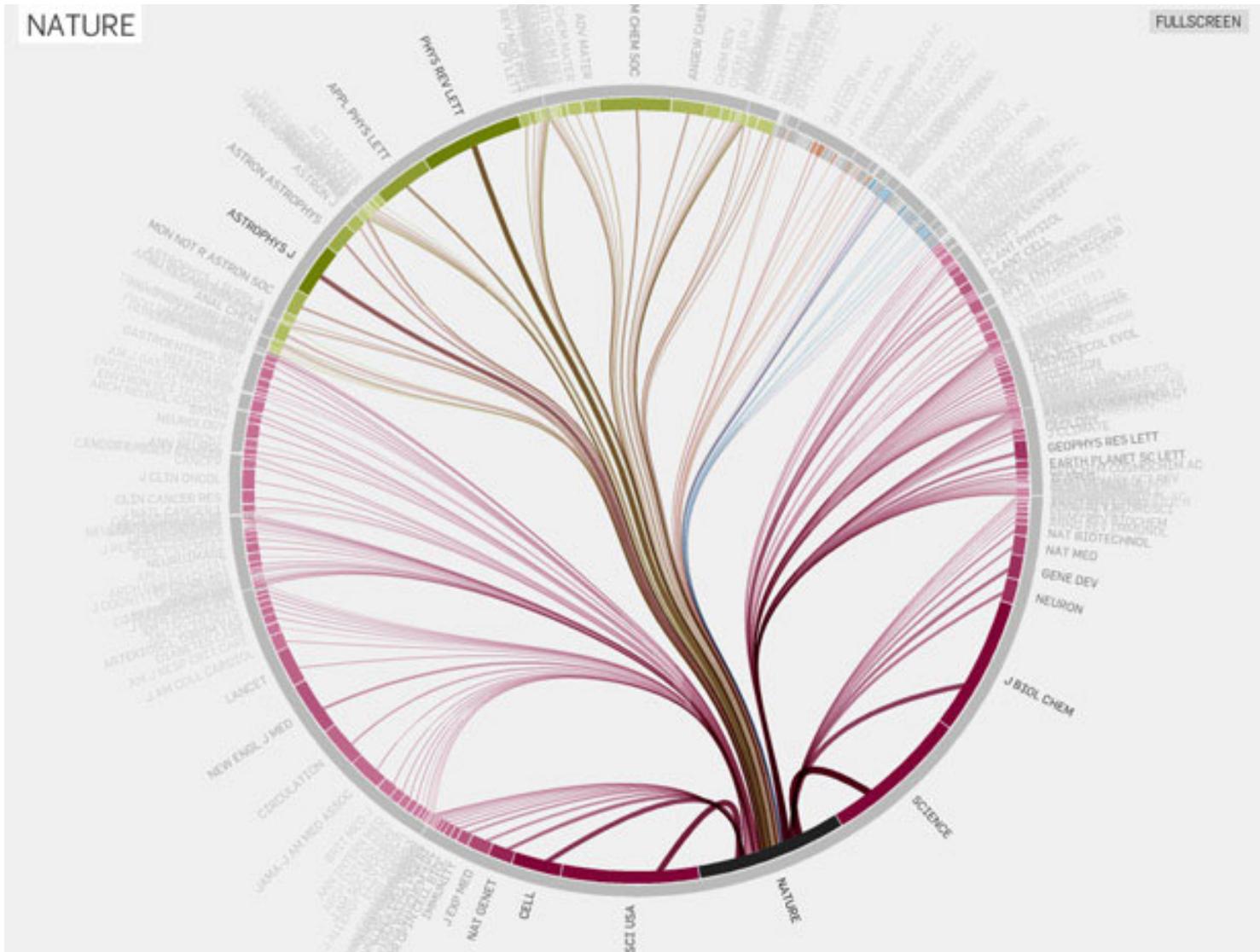
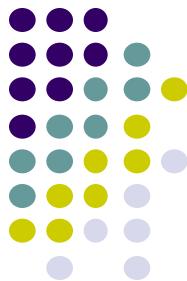
网络流量分析



非线性的数据无处不在

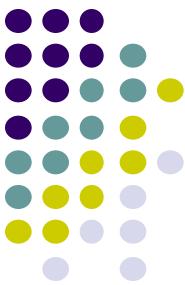


Julia Set



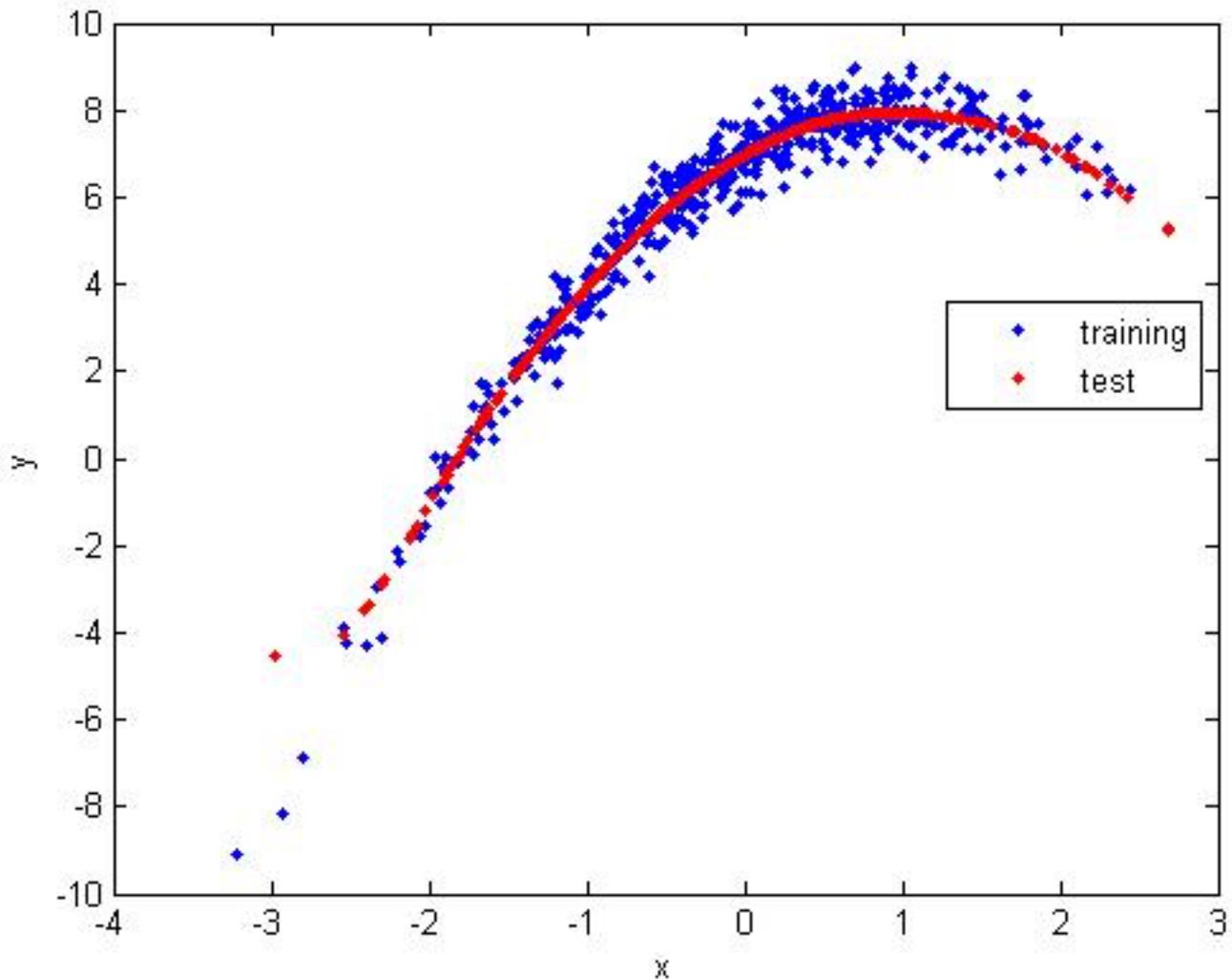
显示学术刊物被引用的关系，借此评价一份学术刊物的份量

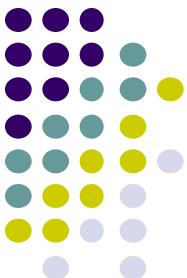
Visualizing Information Flow in Science



非线性的特点

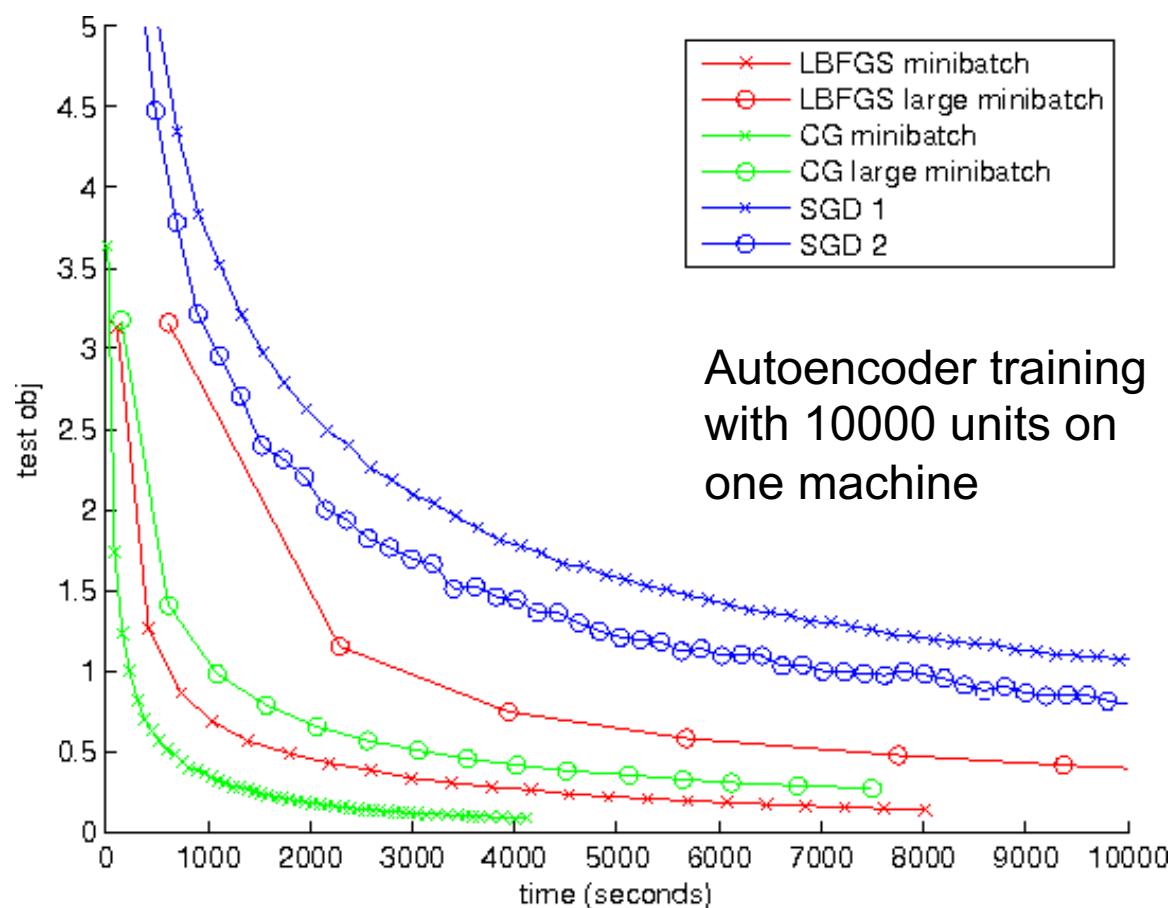
- 不是线性
- 复杂弯曲
- 没有通解

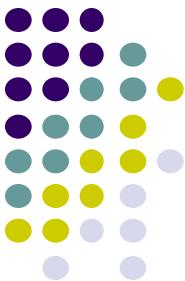




Deep Learning方法的兴起

On Optimization Methods for Deep Learning(ICML2011)





1.1 无约束问题的最优条件

- $\min f(\mathbf{x}), \mathbf{x} \in R^n$ 的最优性条件

- **局部极小** 若存在 $\delta > 0$, 使得对所有满足 $\|x - x^*\| < \delta$ 的 x , 都有

$$f(x) \geq f(x^*),$$

则称 x^* 为 f 的局部极小点。

如所有满足 $\|x - x^*\| < \delta$ 的 x , 都有 $f(x) > f(x^*)$,

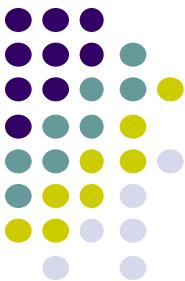
则称 x^* 为 f 的严格局部极小点。

- **全局极小** 若存在 $\delta > 0$, 使得对所有 x , 都有 $f(x) \geq f(x^*)$,

则称 x^* 为 f 的总体极小点。

如所有 x , 都有 $f(x) > f(x^*)$,

则称 x^* 为 f 的严格总体极小点。



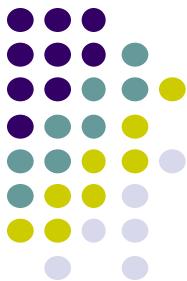
1.1 无约束问题的最优条件

- $\min f(\mathbf{x}), \mathbf{x} \in R^n$ 的最优性条件。

设 $g(x) = \nabla f(x), G(x) = \Delta f(x)$ 分别为 f 的一阶和二阶导数。

定理 (一阶必要条件) : 设 $f: D \subset R^n \rightarrow R^1$ 在开集 D 上连续可微, 若 $x^* \in D$ 是局部极小点, 则 $g(x^*) = 0$.

定理 (二阶必要条件) : 设 $f: D \subset R^n \rightarrow R^1$ 在开集 D 上二阶连续可微, 若 $x^* \in D$ 是局部极小点, 则 $g(x^*) = 0, G(x^*) \geq 0$



1.1 无约束问题的最优条件

$g(x^*) = 0$, 则 x 称为函数 f 的平稳点。平稳点有可能是极小点，也可能为极大点，也可能不是极值点（鞍点）。

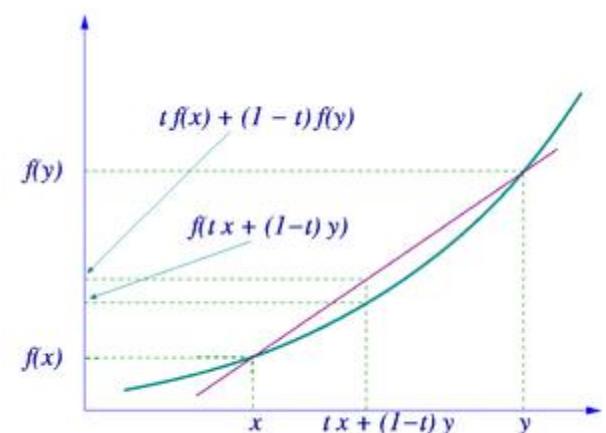
定理（二阶充分条件）：设 $f : D \subset R^n \rightarrow R^1$ 在开集 D 上二阶连续可微，若 $x^* \in D$ 是严格局部极小点的充分条件是，则 $g(x^*) = 0$, 且 $G(x^*)$ 为正定矩阵。

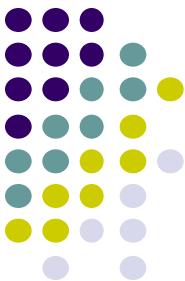
定理（凸充分性定理）：设 $f : D \subset R^n \rightarrow R^1$ 是凸函数且一阶连续可微，若 x^* 是总体极小点的充要条件是 $g(x^*) = 0$ 。

问题：什么是凸函数？

定义1. $f((1 - \alpha)x + \alpha y) < (1 - \alpha)f(x) + \alpha f(y)$

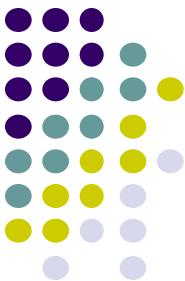
定义2. 函数的二阶导数 $G(x) \geq 0$





1.2 最优化方法的结构

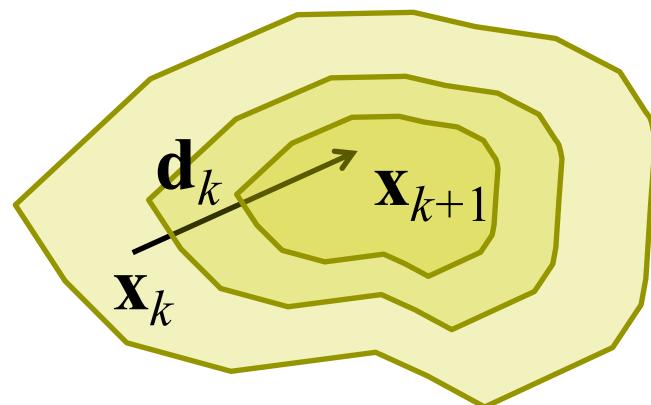
- 迭代优化方法的基本思想：
 - 给定一个初始点 \mathbf{x}_0 ,
 - 按照某一迭代规则产生一个点列 $\{\mathbf{x}_k\}$, 使得
 - 当 $\{\mathbf{x}_k\}$ 是有穷点列时, 其最后一个点是最优化模型问题的最优解。
 - 当 $\{\mathbf{x}_k\}$ 是无穷点列时, 其极限点为最优解。
- 一个**好的**算法应具备的典型特征为：
 - 迭代点 \mathbf{x}_k 能**稳定**地接近局部极小点 \mathbf{x}^* 的邻域, 然后迅速收敛于 \mathbf{x}^*
 - 当给定的某种**收敛**准则满足时, 迭代即终止。

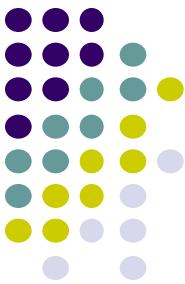


最优化（迭代）方法的结构

给定初始点 \mathbf{x}_0 ,

1. 确定搜索方向 \mathbf{d}_k , 即依照一定规则构造 f 在 \mathbf{x}_k 点处的下降方向为搜索方向
2. 确定步长因子 α_k , 使目标函数值有某种意义下降
3. 令 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
 - a) 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,
 - b) 否则, 重复以上步骤。





收敛速度

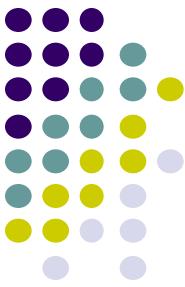
- 收敛速度是衡量最优化方法有效性的重要方面

若存在实数 $\alpha > 0$ 及一个与迭代次数 k 无关的常数 $q > 0$, 使得

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^\alpha} = q,$$

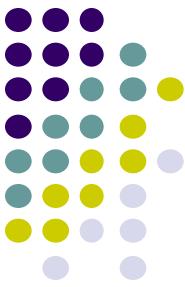
则称算法产生的迭代点列 $\{\mathbf{x}_k\}$ 具有 $Q-\alpha$ 阶收敛速度。特别地

- (1) 当 $\alpha=1, q > 0$ 时, $\{\mathbf{x}_k\}$ 具有 Q -线性收敛速度。
- (2) 当 $1 < \alpha < 2, q > 0$ 时(或者 $\alpha=1, q = 0$), $\{\mathbf{x}_k\}$ 具有 Q -超线性收敛速度。
- (3) 当 $\alpha=2, q > 0$ 时, $\{\mathbf{x}_k\}$ 具有 Q -二阶收敛速度。



收敛速度

- 一般认为，具有超线性和二阶收敛速度的方法是比较快速的
- 但对于任何一个算法，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结果
 - 忽略了误差；函数计算不满足限制条件
- 需要选择有代表性的检验函数进行数值计算



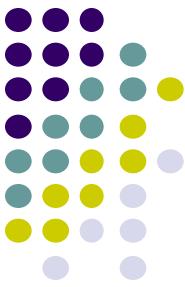
收敛速度

定理：如果序列 $\{x_k\}$ —超线性收敛到 x^* ，那么

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x_k\|}{\|x_k - x^*\|} = 1,$$

但反之一般不成立。

该定理表明，可以用 $\|x_{k+1} - x_k\|$ 来代替 $\|x_k - x^*\|$ 给出终止条件，并且该估计随着 k 的增加而改善。



终止条件

$$\|x_{k+1} - x_k\| \leq \varepsilon, \text{ 或 } |f(x_k) - f(x_{k+1})| \leq \varepsilon.$$

有时 $\|x_{k+1} - x_k\|$ 是小的，但 $|f(x_k) - f(x_{k+1})|$ 仍然很大（或者相反）。

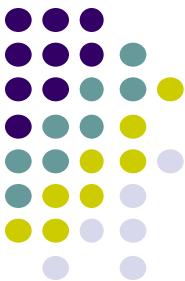
应同时使用两式，*Himmelblau* 提出：

当 $\|x_k\| > \varepsilon_2, |f(x_k)| > \varepsilon_2$ 时，采用 $\frac{\|x_{k+1} - x_k\|}{\|x_k\|} \leq \varepsilon_1, \frac{|f(x_k) - f(x_{k+1})|}{|f(x_k)|} \leq \varepsilon_1$,

否则采用 $\|x_{k+1} - x_k\| \leq \varepsilon_1, \text{ 或 } |f(x_k) - f(x_{k+1})| \leq \varepsilon_1$

对于有一阶导数信息，且收敛不太快的算法，可采用 $\|g_k\| \leq \varepsilon_3$, 其中 $g_k = \nabla f(x_k)$ 。
但由于平稳点也可能是鞍点，因此可与上式结合使用。

一般地，可取 $\varepsilon_1 = \varepsilon_2 = 10^{-5}, \varepsilon_3 = 10^{-4}$.



1.3 一维搜索

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \rightarrow \varphi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

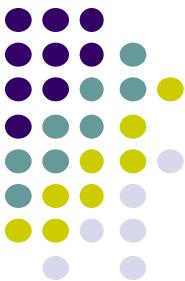
- 单变量函数的最优化。

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k,$$

其关键就是构造搜索方向 \mathbf{d}_k 和步长因子 α_k 。设 $\varphi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$,
这样, 确定 α_k , 使得 $\varphi(\alpha_k) < \varphi(0)$ 。这就是关于 α 的一维搜索问题。

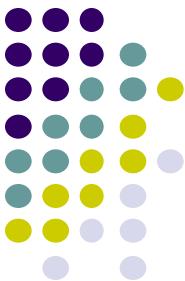
若 α_k 使得目标函数沿方向达到极小, 即 $\varphi(\alpha_k) = \min_{\alpha>0} \varphi(\alpha)$, 则称这样
的一维搜索为最优一维搜索 (或精确一维搜索), α_k 为最优步长因子。
若取 α_k 使得目标函数得到可以接受的下降量, 则称为近似一维搜索,
或不精确一维搜索。

实际中, 精确的最优步长因子一般不能求到, 求几乎精确的最优步长
因子需花费大量的工作量, 因而花费计算量较少的不精确一维搜索
受到重视。



一维搜索的主要结构

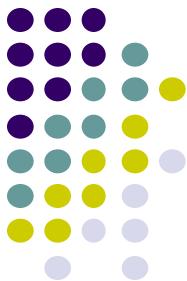
1. 确定包含问题最优解的搜索区间
 2. 再用某种分割技术或插值方法缩小这个区间，进行搜索求解
-
- 搜索区间：包含最优值的闭区间。
 - 确定搜索区间的简单方法——进退法。
 - 从一点出发，试图确定出函数值呈现“高—低—高”的三点。一个方向不成功，就退回来，再沿相反方向寻找。



一维区间搜索的进退法

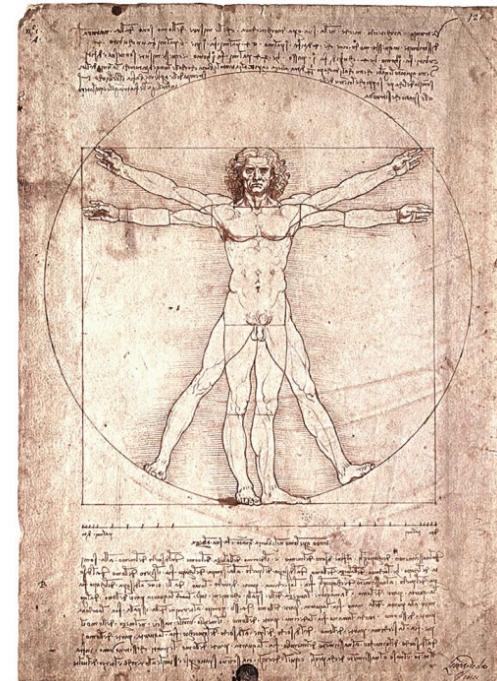
- 确定搜索区间的简单方法——进退法。
 - 从一点出发，试图确定出函数值呈现“高—低—高”的三点。一个方向不成功，就退回来，再沿相反方向寻找。

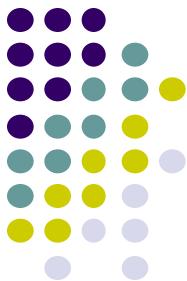
- 1.选取初始值 α_0 , h_0 ,加倍系数 $t > 1$ (一般 $t = 2$), $k = 0$
- 2.如 $\varphi(\alpha_k + h_k) < \varphi(\alpha_k)$,则 $h_{k+1} = th_k$, $\alpha_{k+1} = \alpha_k + h_{k+1}$, $k++$,返回2
- 3.若 $k == 0$, 转换搜索方向 (即 $h_k = -h_k$,转2) ;
否则, 停止迭代, 输出 $a = \min\{\alpha_0, \alpha_{k+1}\}$, $b = \max\{\alpha_0, \alpha_{k+1}\}$



一维搜索：黄金分割法

- 基本思想：通过取试探点和进行函数值比较，使包含极小点的搜索区间不断缩小，当缩短到一定程度后，该区间上的任意一点均可看作极小值的近似
- 该方法用途广泛
 - 尤其适合导数表达式复杂或写不出的情况
- 华罗庚优选法





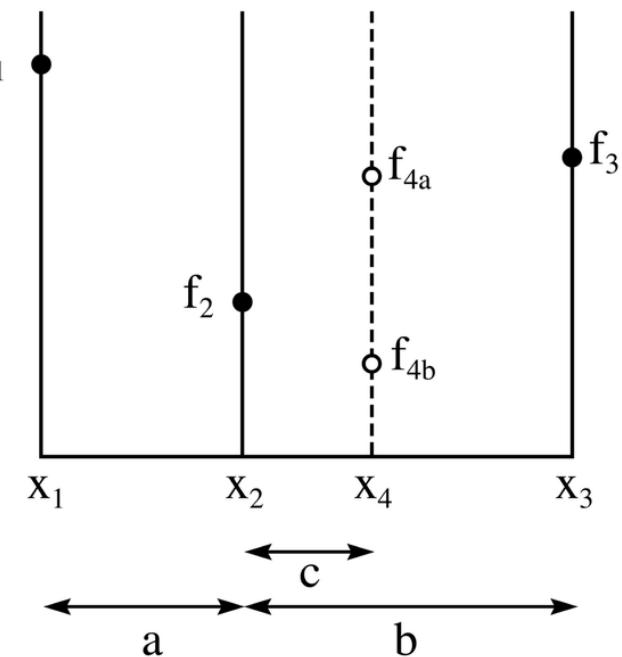
一维搜索：黄金分割法（0.618法）

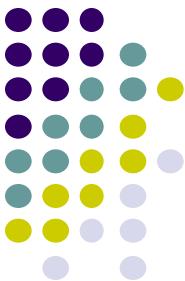
设 $\varphi(\alpha) = f(x_k + \alpha d_k)$ 是搜索区间上的单峰函数。

设第 k 次迭代时搜索区间为 $[a_k, b_k]$,

取两个试探点 λ_k, μ_k ($\lambda_k < \mu_k$), 计算 $\varphi(\lambda_k)$ 和 $\varphi(\mu_k)$ 。

- (1) 若 $\varphi(\lambda_k) \leq \varphi(\mu_k)$, 则令 $a_{k+1} = a_k, b_{k+1} = \mu_k$ (易证此时仍为单峰函数)
- (2) 若 $\varphi(\lambda_k) > \varphi(\mu_k)$, 则令 $a_{k+1} = \lambda_k, b_{k+1} = b_k$





一维搜索：黄金分割法（0.618法）

设 $\varphi(\alpha) = f(x_k + \alpha d_k)$ 是 搜索区间上的单峰函数。

设第 k 次迭代时搜索区间为 $[a_k, b_k]$,

取两个试探点 λ_k, μ_k ($\lambda_k < \mu_k$), 计算 $\varphi(\lambda_k)$ 和 $\varphi(\mu_k)$ 。

- (1) 若 $\varphi(\lambda_k) \leq \varphi(\mu_k)$, 则令 $a_{k+1} = a_k, b_{k+1} = \mu_k$ (易证此时仍为单峰函数)
- (2) 若 $\varphi(\lambda_k) > \varphi(\mu_k)$, 则令 $a_{k+1} = \lambda_k, b_{k+1} = b_k$

要求试探点 λ_k, μ_k 满足下列条件：

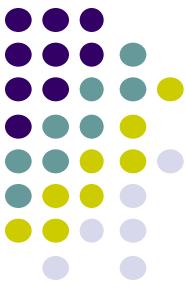
1. λ_k, μ_k 到搜索区间 $[a_k, b_k]$ 的端点等距, 即 $\lambda_k - a_k = b_k - \mu_k$ (或 $b_k - \lambda_k = \mu_k - a_k$)
2. 每次迭代, 搜索区间长度缩短率相同, 即 $b_{k+1} - a_{k+1} = \tau(b_k - a_k)$

则可推出 $\lambda_k = a_k + (1 - \tau)(b_k - a_k)$, $\mu_k = a_k + \tau(b_k - a_k)$ 。

3. $\lambda_k = \mu_{k+1}$

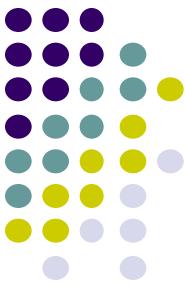
$$\text{可推出 } \tau = \frac{\sqrt{5}-1}{2} \approx 0.618$$

不难知道第 n 次迭代后的区间长度为 $\tau^{n-1}(b_1 - a_1)$, 收敛速度为线性。



改进

- 实际上所遇到的函数**不一定是单峰函数**，这时搜索出的值有可能大于初始区间的端点值。
- 改进：每次缩小区间时，不只比较两个内点处的函数值，而是比较两个内点和两个端点处的函数值：
 - 当左边第一个或第二个点是这四个点中函数值最小的点时，丢弃右端点；
 - 否则，丢弃左端点。



Fibonacci法

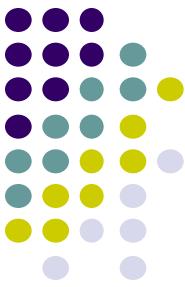
- 与0.618法的主要区别之一：搜索区间缩短率不是采用黄金分割数，而是采用Fibonacci数。
 - Fibonacci数满足：

$$F_0 = F_1 = 1, F_{k+1} = F_k + F_{k-1}, k=1, 2, \dots$$

1, 1, 2, 3, 5, 8, 13, ...

Fibonacci分割方法：

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



Fibonacci法

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

要求经过 n 次计算后，最后的区间长度不超过 δ ，即

$$b_n - a_n \leq \delta.$$

另一方面， $b_n - a_n = \frac{F_1}{F_2}(b_{n-1} - a_{n-1}) = \frac{F_1}{F_2} \frac{F_2}{F_3} \dots \frac{F_{n-1}}{F_n}(b_1 - a_1) = \frac{1}{F_n}(b_1 - a_1)$

故可得 $F_n \geq \frac{(b_1 - a_1)}{\delta}$

可以证明 $n \rightarrow \infty$ ，Fibonacci法与0.618法的区间缩短率相同。

因而Fibonacci法线性收敛。

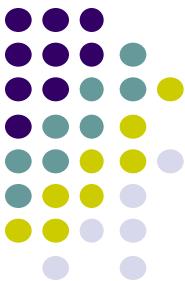
同时可以证明：

Fibonacci法是分割方法求一维极小化问题的最优策略，而0.618法是分割方法求一维极小化问题的近似最优解。



1.3.2 插值法

- 插值法是一类重要的搜索方法，其基本思想是：
 - 在搜索区间中不断用低次(通常不超过三次)多项式来近似目标函数，并逐步用插值多项式的极小点来逼近一维搜索问题的极小点。
- 当函数具有比较好的解析性质时，插值方法比直接方法(0.618法或Fibonacci法)效果更好。



二次插值法： 一点二次插值（牛顿法）

- 利用一点处的函数值、一阶和二阶导数值构造二次插值函数：

$q(\alpha) = a\alpha^2 + b\alpha + c$, 则 $\alpha = -\frac{b}{2a}$ 为近似极小点的计算公式。

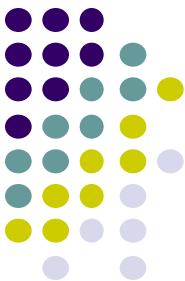
设端点处值为 $\varphi(\alpha_1)$, $\varphi'(\alpha_1)$, $\varphi''(\alpha_1)$ 。则可推出一维搜索的近似极小点为

$$\bar{\alpha} = -\frac{b}{2a} = \alpha_1 - \frac{\varphi'(\alpha_1)}{\varphi''(\alpha_1)}$$

写成迭代形式为

$$\alpha_{k+1} = \alpha_k - \frac{\varphi'(\alpha_k)}{\varphi''(\alpha_k)}$$

- 牛顿法的优点是收敛速度快，具有局部二阶收敛速度。



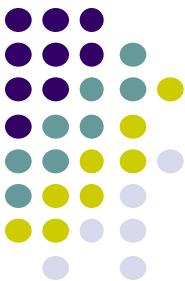
二点二次插值法

- 给出两点的函数值和其中一点的导数值，构造二次插值函数。类似的可得

$$\bar{\alpha} = -\frac{b}{2a} = \alpha_1 - \frac{(\alpha_1 - \alpha_2)\varphi'(\alpha_1)}{2\left[\varphi'(\alpha_1) - \frac{\varphi(\alpha_1) - \varphi(\alpha_2)}{\alpha_1 - \alpha_2}\right]}.$$

写成迭代形式为 $\alpha_{k+1} = \alpha_k - \frac{(\alpha_k - \alpha_{k-1})\varphi'(\alpha_k)}{2\left[\varphi'(\alpha_k) - \frac{\varphi(\alpha_k) - \varphi(\alpha_{k-1})}{\alpha_k - \alpha_{k-1}}\right]}$

- 可证明二点二次插值法的收敛阶为1.618，超线性收敛。



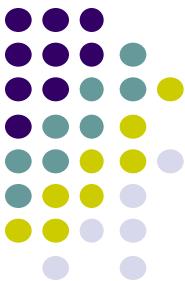
三点二次法（抛物线法）

$$\bar{\alpha} = -\frac{b}{2a} = \frac{1}{2}(\alpha_1 + \alpha_2) + \frac{1}{2} \frac{(\phi_1 - \phi_2)(\phi_2 - \phi_3)(\phi_3 - \phi_1)}{(\alpha_2 - \alpha_3)\phi_1 + (\alpha_3 - \alpha_1)\phi_2 + (\alpha_1 - \alpha_2)\phi_3}.$$

$$\phi_1 > \phi_2, \phi_3 > \phi_2$$

迭代时，从 $\alpha_1, \alpha_2, \alpha_3, \bar{\alpha}$ 中选择目标函数最小的点及其左右两点，进行下一步的迭代。

- 超线性收敛，收敛阶近似为1.32



二点三次插值法

- 用三次多项式来逼近。比二次插值法有较好的收敛效果，但通常要求计算导数值，且计算量较大。一般当导数易求时，用三次插值法较好。

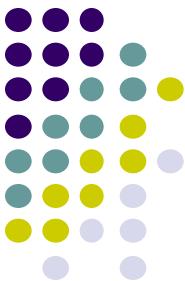
用函数 $\varphi(a)$ 在 a, b 两点的函数值 $\varphi(a), \varphi(b)$ 和导数值 $\varphi'(a), \varphi'(b)$ 来构造三次函数，初值条件 $a < b, \varphi'(a) < 0, \varphi'(b) > 0$ 。经过推导

$$\bar{\alpha} = a + (b - a) \frac{(w - \varphi'(a) - z)}{\varphi'(b) - \varphi'(a) + 2w},$$

$$\text{其中 } z = 3 \frac{\varphi(b) - \varphi(a)}{b - a} - \varphi'(a) - \varphi'(b), \quad w^2 = z^2 - \varphi'(a)\varphi'(b)$$

在迭代时，若 $f(\bar{\alpha}) < 0$ ，则令 $a = \bar{\alpha}$; 若 $f(\bar{\alpha}) > 0$ ，令 $b = \bar{\alpha}$.

- 收敛速度为2阶，一般优于抛物线法。



1.3.3 不精确的一维搜索方法

- 精确一维搜索往往需要花费很大的时间。
 - 当迭代点远离问题的解时，精确求解通常不十分有效。
 - 很多最优化方法，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。
- 只要保证目标函数有满意的下降，可大大节省计算量。



Armijo-Goldstein准则

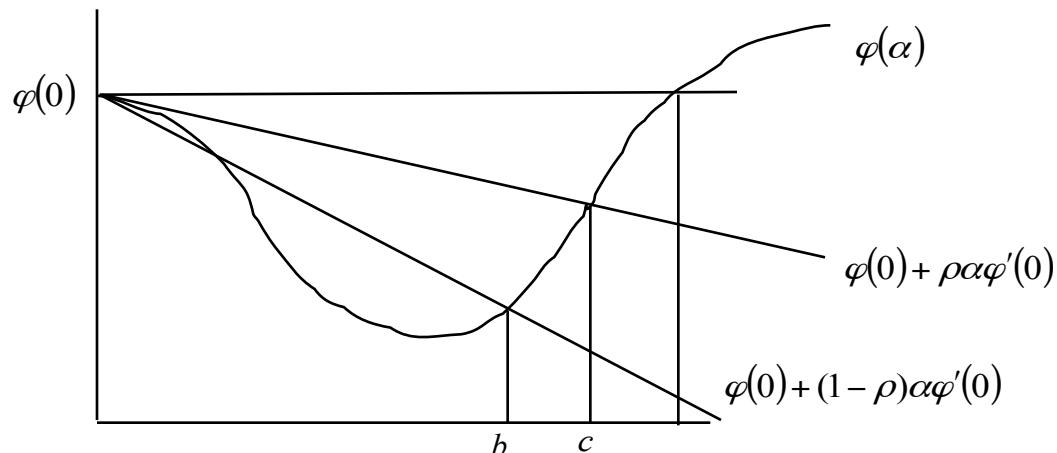
设区间 $J = \{\alpha > 0 \mid \varphi(\alpha) < \varphi(0)\}$ 。要保证目标函数下降，同时 f 的下降不是太小。一个合理的要求：

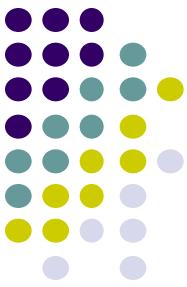
$$\varphi(\alpha) \leq \varphi(0) + \rho\alpha\varphi'(0), \text{ 其中 } 0 < \rho < 0.5.$$

为了避免 α 取得过小，加上另一个要求

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

上述两式称为 *Armijo - Goldstein* 不精确线性搜索准则。满足上述要求的 α 构成区间 $J_2 = [b, c]$ ，称为可接受区间， α 称为可接受步长因子。

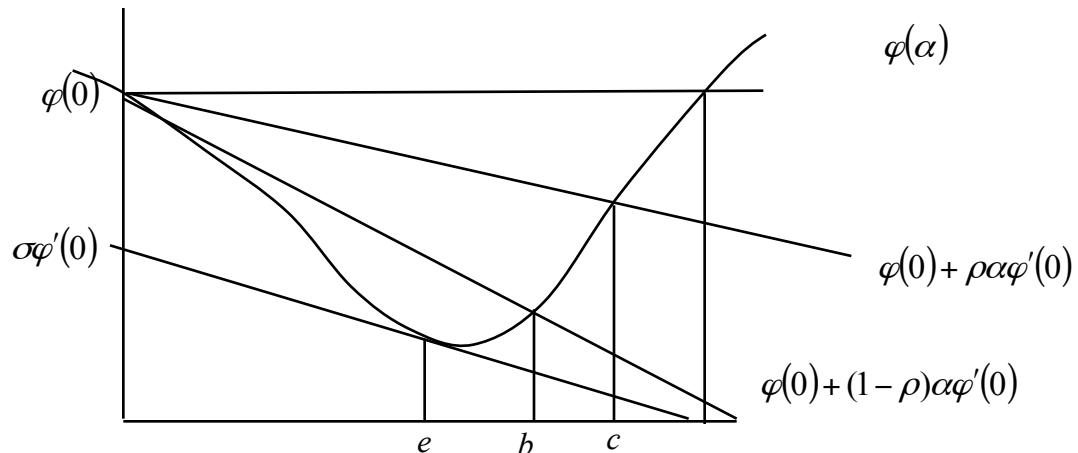




Wolfe-Powell准则

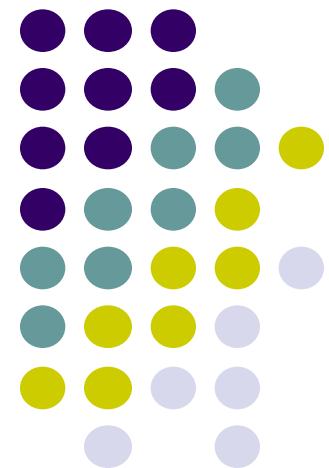
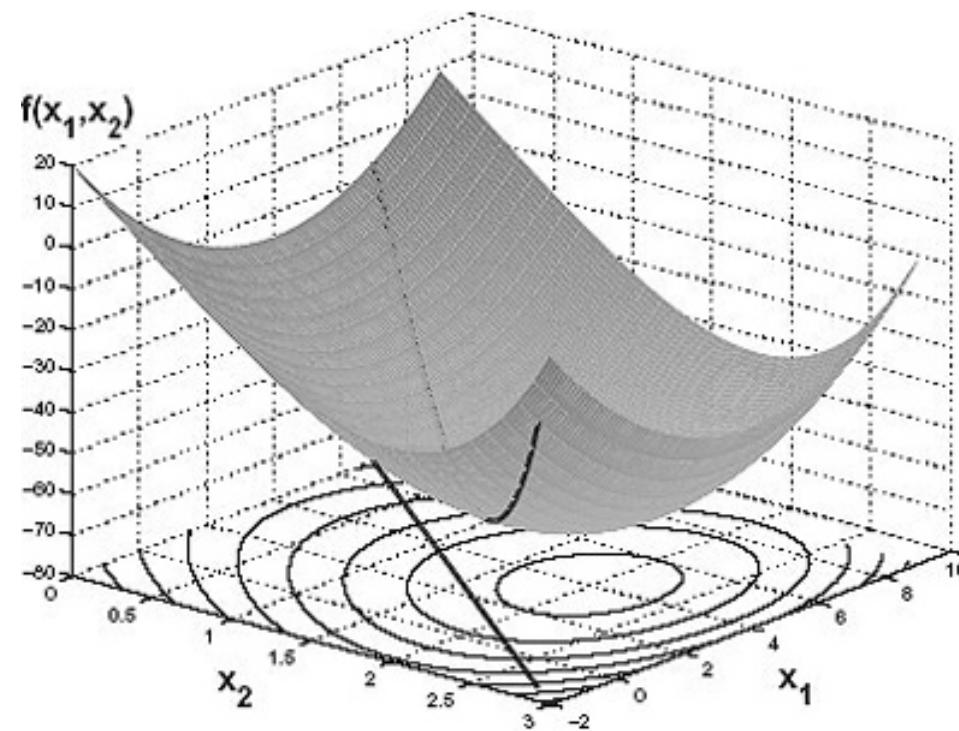
- Armijo-Goldstein准则有可能把最优步长因子排除在可接受区间外，因此更改第二个条件为

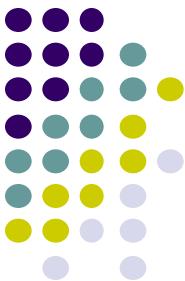
$$\varphi(\alpha) \geq \varphi(0) + (1-\rho)\alpha\varphi'(0) \Rightarrow |\varphi'(\alpha)| \leq \sigma |\varphi'(0)|, \text{ 其中 } \rho < \sigma < 1.$$



- 一般地， σ 值越小（0.1），线性搜索越精确。不过，计算量也越大。通常取 $\rho = 0.1, \sigma = 0.4$.

1.4 牛顿型方法





1.4.1 最速下降法

- 以负梯度方向作为极小化算法的搜索方向，即

$$\mathbf{d}_k = -\mathbf{g}_k$$

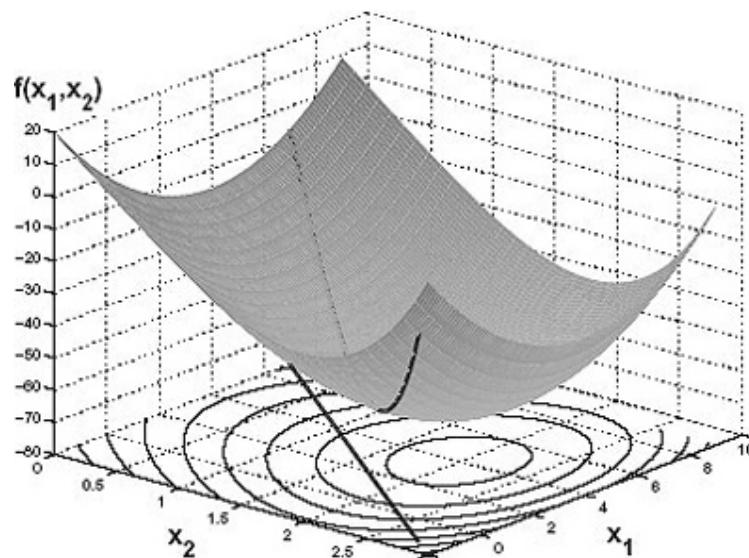
- 具有总体收敛性：

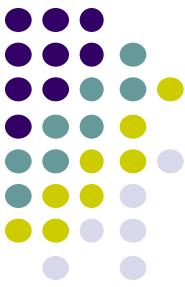
- 产生的迭代点列 $\{\mathbf{x}_k\}$ 的每一个聚点都是平稳点。

- 最速下降方向仅是局部性质

- 对于许多问题并非下降方向，而且下降非常缓慢
- 接近极小点时，步长越小前进越慢

- 线性收敛



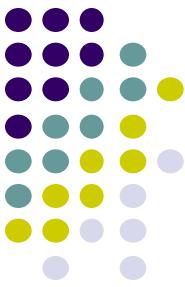


梯度法（最速下降法）：

1. 搜索方向: $d^k = -\nabla f(x^k)$, 也称为最速下降方向;
2. 搜索步长: λ_k 取最优步长, 即满足 $f(x^k + \lambda_k d^k) = \min_{\lambda} f(x^k + \lambda d^k)$ 。

梯度法算法步骤:

1. 给定初始点 $x^1 \in R^n$, 允许误差 $\varepsilon > 0$, 令 $k = 1$ 。
2. 计算搜索方向 $d^k = -\nabla f(x^k)$;
3. 若 $\|d^k\| \leq \varepsilon$, 则停止计算, x^k 为所求极值点; 否则, 求最优步长 λ_k 使得 $f(x^k + \lambda_k d^k) = \min_{\lambda} f(x^k + \lambda d^k)$ 。
4. 令 $x^{k+1} = x^k + \lambda_k d^k$, 令 $k := k + 1$, 转2。



例. 用最速下降法求解: $\min f(x) = x_1^2 + 3x_2^2$, 设初始点为 $x^1 = (2,1)^T$,
求迭代一次后的迭代点 x^2 .

解: $\because \nabla f(x) = (2x_1, 6x_2)^T$,

$$\therefore d^1 = -\nabla f(x^1) = (-4, -6)^T.$$

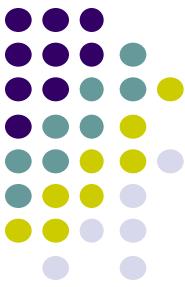
$$\therefore x^1 + \lambda d^1 = (2 - 4\lambda, 1 - 6\lambda)^T.$$

令 $\varphi(\lambda) = f(x^1 + \lambda d^1) = (2 - 4\lambda)^2 + 3(1 - 6\lambda)^2$,

求解 $\min_{\lambda} \varphi(\lambda)$

令 $\varphi'(\lambda) = -8(2 - 4\lambda) - 36(1 - 6\lambda) = 0 \Rightarrow \lambda_1 = \frac{13}{62}$

$$\therefore x^2 = x^1 + \lambda_1 d^1 = \left(\frac{36}{31}, \frac{-8}{31}\right)^T$$



收敛性

性质. 设 $f(x)$ 有一阶连续偏导数, 若 步长 λ_k 满足

$$f(x^k + \lambda_k d^k) = \min_{\lambda} f(x^k + \lambda d^k)$$

则有 $\nabla f(x^k + \lambda_k d^k)^T d^k = 0$ 。

证明: 令 $\varphi(\lambda) = f(x^k + \lambda d^k)$, 所以

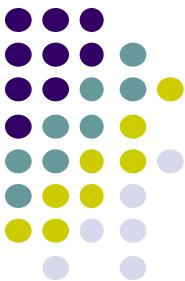
$$\varphi'(\lambda) = \nabla f(x^k + \lambda d^k)^T d^k.$$

$$\therefore f(x^k + \lambda_k d^k) = \min_{\lambda} f(x^k + \lambda d^k)$$

$$\therefore \varphi'(\lambda_k) = \nabla f(x^k + \lambda_k d^k)^T d^k = 0.$$

注: 因为梯度法的搜索方向 $d^{k+1} = -\nabla f(x^k + \lambda_k d^k)$, 所以

$$(d^{k+1})^T d^k = 0 \Rightarrow d^{k+1} \perp d^k.$$



在深度学习中的例子

在应用机器学习算法时，我们通常采用梯度下降法来对采用的算法进行训练。其实，常用的梯度下降法还具体包含有三种不同的形式，它们也各自有着不同的优缺点。

下面我们以线性回归算法来对三种梯度下降法进行比较。

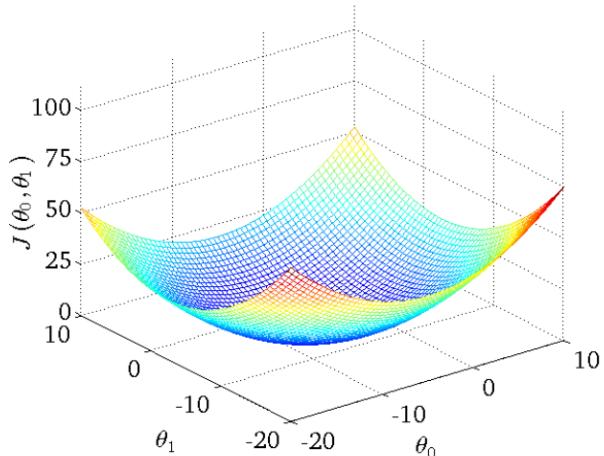
一般线性回归函数的假设函数为：

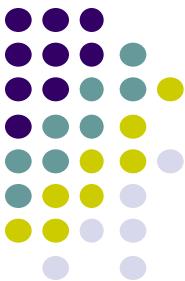
$$h_{\theta} = \sum_{j=0}^n \theta_j x_j$$

对应的能量函数（损失函数）形式为：

$$J_{train}(\theta) = 1/(2m) \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

下图为一个二维参数 (θ_0 和 θ_1) 组对应能量函数的可视化图：





在深度学习中的例子

1. 批量梯度下降法BGD

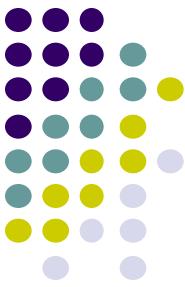
批量梯度下降法 (Batch Gradient Descent, 简称BGD) 是梯度下降法最原始的形式，它的具体思路是在更新每一参数时都使用所有的样本来进行更新，其数学形式如下：

(1) 对上述的能量函数求偏导：

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m} \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$

(2) 由于是最小化风险函数，所以按照每个参数 θ 的梯度负方向来更新每个 θ ：

$$\theta_j' = \theta_j + \frac{1}{m} \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$



在深度学习中的例子

1. 批量梯度下降法BGD

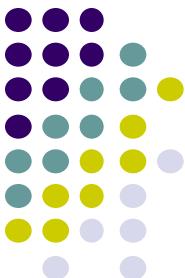
具体的伪代码形式为：

repeat{

$$\theta_j' = \theta_j + \frac{1}{m} \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$

(**for every** j=0, ... , n)

}



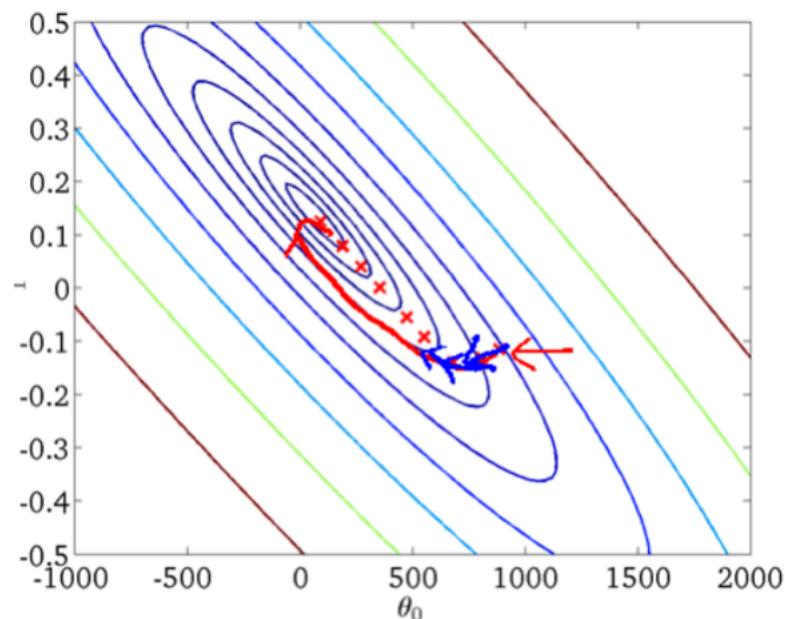
在深度学习中的例子

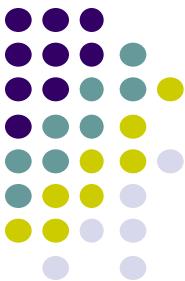
从上面公式可以注意到，它得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果样本数目 m 很大，那么可想而知这种方法的迭代速度！所以，这就引入了另外一种方法，随机梯度下降。

优点：全局最优解；易于并行实现；

缺点：当样本数目很多时，训练过程会很慢。

从迭代的次数上来看，BGD迭代的次数相对较少。其迭代的收敛曲线示意图可以表示如下：





在深度学习中的例子

2. 随机梯度下降法SGD

由于批量梯度下降法在更新每一个参数时，都需要所有的训练样本，所以训练过程会随着样本数量的加大而变得异常的缓慢。随机梯度下降法（*Stochastic Gradient Descent*，简称SGD）正是为了解决批量梯度下降法这一弊端而提出的。

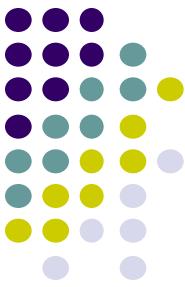
将上面的能量函数写为如下形式：

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^i - h_\theta(x^i))^2 = \frac{1}{m} \sum_{i=1}^m \text{cost}(\theta, (x^i, y^i))$$

$$\text{cost}(\theta, (x^i, y^i)) = \frac{1}{2} (y^i - h_\theta(x^i))^2$$

利用每个样本的损失函数对 θ 求偏导得到对应的梯度，来更新 θ ：

$$\theta_j' = \theta_j + (y^i - h_\theta(x^i))x_j^i$$



在深度学习中的例子

2. 随机梯度下降法SGD

具体的伪代码形式为：

1. Randomly shuffle dataset;

2. repeat{

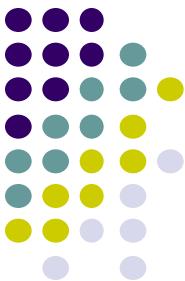
 for i=1, ..., m{

$$\theta_j^{(i)} = \theta_j^{(i)} + (y^{(i)} - h_{\theta}(x^{(i)}))x_j^{(i)}$$

 (for j=0, ..., n)

 }

}



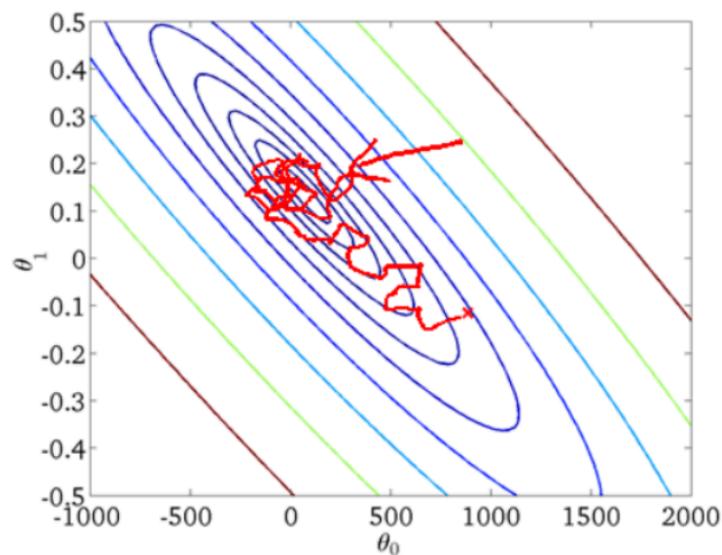
在深度学习中的例子

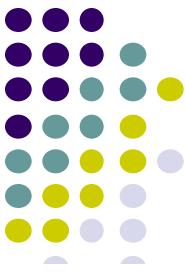
随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将theta迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代10次的话就需要遍历训练样本10次。但是，SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。

优点：训练速度快；

缺点：准确度下降，并不是全局最优；不易于并行实现。

从迭代的次数上来看，SGD迭代的次数较多，在解空间的搜索过程看起来很盲目。其迭代的收敛曲线示意图可以表示如下：





在深度学习中的例子

3. 小批量梯度下降法MBGD

有上述的两种梯度下降法可以看出，其各自均有优缺点，那么能不能在两种方法的性能之间取得一个折衷呢？即，算法的训练过程比较快，而且也要保证最终参数训练的准确率，而这正是小批量梯度下降法（*Mini-batch Gradient Descent*，简称MBGD）的初衷。

MBGD在每次更新参数时使用b个样本（b一般为10），其具体的伪代码形式为：

Say $b=10, m=1000$.

Repeat{

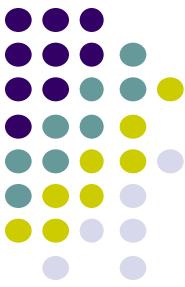
for $i=1, 11, 21, 31, \dots, 991$ {

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

 (**for every** $j=0, \dots, n$)

}

}



在深度学习中的例子

4. 总结

Batch gradient descent: Use all examples in each iteration;

Stochastic gradient descent: Use 1 example in each iteration;

Mini-batch gradient descent: Use b examples in each iteration.

以上资料来源: <http://www.cnblogs.com/maybe2030/>

And more: <http://adventuresinmachinelearning.com/stochastic-gradient-descent/>

Yet Another Accelerated SGD: ResNet-50 Training on ImageNet in 74.7 seconds

Masafumi Yamazaki, Akihiko Kasagi, Akihiro Tabuchi, Takumi Honda, Masahiro Miwa,
Naoto Fukumoto, Tsuguchika Tabaru, Atsushi Ike, Kohta Nakashima

Fujitsu Laboratories Ltd.

{m.yamazaki, kasagi.akihiko, tabuchi.akihiro, honda.takumi, masahiro.miwa,
fukumoto.naoto, tabaru, ike, nakashima.kouta}@fujitsu.com

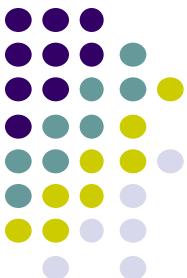
Abstract—There has been a strong demand for algorithms that can execute machine learning as faster as possible and the speed of deep learning has accelerated by 30 times only in the past two years. Distributed deep learning using the large mini-batch is a key technology to address the demand and is a great challenge as it is difficult to achieve high scalability on large clusters without compromising accuracy. In this paper, we introduce optimization methods which we applied to this challenge. We achieved the training time of 74.7 seconds using 2,048 GPUs on ABCI cluster applying these methods. The training throughput is over 1.73 million images/sec and the top-1 validation accuracy is 75.08%.

I. INTRODUCTION

use convolutional layers for 2D and 3D image data. Ioffe et al. [9] introduced the batch normalization technique, in which the feature values in hidden layers are normalized to avoid vanishing gradients. In addition, this technique enables training of models with a large number of layers, such as ResNet.

Generally, the mini-batch size should be large for distributed deep learning on large clusters. Goyal et al. [2] proposed the warm-up technique to keep the validation accuracy with 8,192 mini-batch size. Google [3] and Sony [7] used the variable mini-batch size which becomes larger and achieved highly parallel processing.

参考来源: <https://mp.weixin.qq.com/s/D68YzjYvpc2epGWFBP6rlQ>



Reducing BERT Pre-Training Time from 3 Days to 76 Minutes

Yang You², Jing Li¹, Jonathan Hseu¹, Xiaodan Song¹, James Demmel², Cho-Jui Hsieh^{1,3}

Yang You is a student researcher at Google Brain. This project is done when he is at Google Brain.

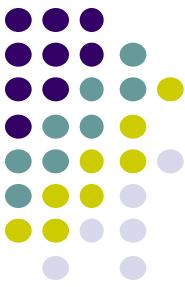
Google¹, UC Berkeley², UCLA³

{youyang, demmel}@cs.berkeley.edu, {jingli, jhseu, xiaodansong, chojui}@google.com

| Solver | Batch Size | Iterations | F1 score on dev set | Hardware | Time |
|------------|------------|------------|---------------------|-----------|--------|
| Baseline | 512 | 1000k | 90.395 | 16 TPUs | 81.4h |
| Our Method | 512 | 1000k | 90.720 | 16 TPUs | 82.8h |
| Our Method | 1k | 500k | 90.377 | 32 TPUs | 43.2h |
| Our Method | 2k | 250k | 90.751 | 64 TPUs | 21.4h |
| Our Method | 4k | 125k | 90.548 | 128 TPUs | 693.6m |
| Our Method | 8k | 62k | 90.556 | 256 TPUs | 390.5m |
| Our Method | 16k | 31k | 90.679 | 512 TPUs | 200.0m |
| Our Method | 32k | 15.6k | 91.460 | 1024 TPUs | 101.2m |
| Our Method | 64k/32k | 8599 | 90.584 | 1024 TPUs | 10.19m |

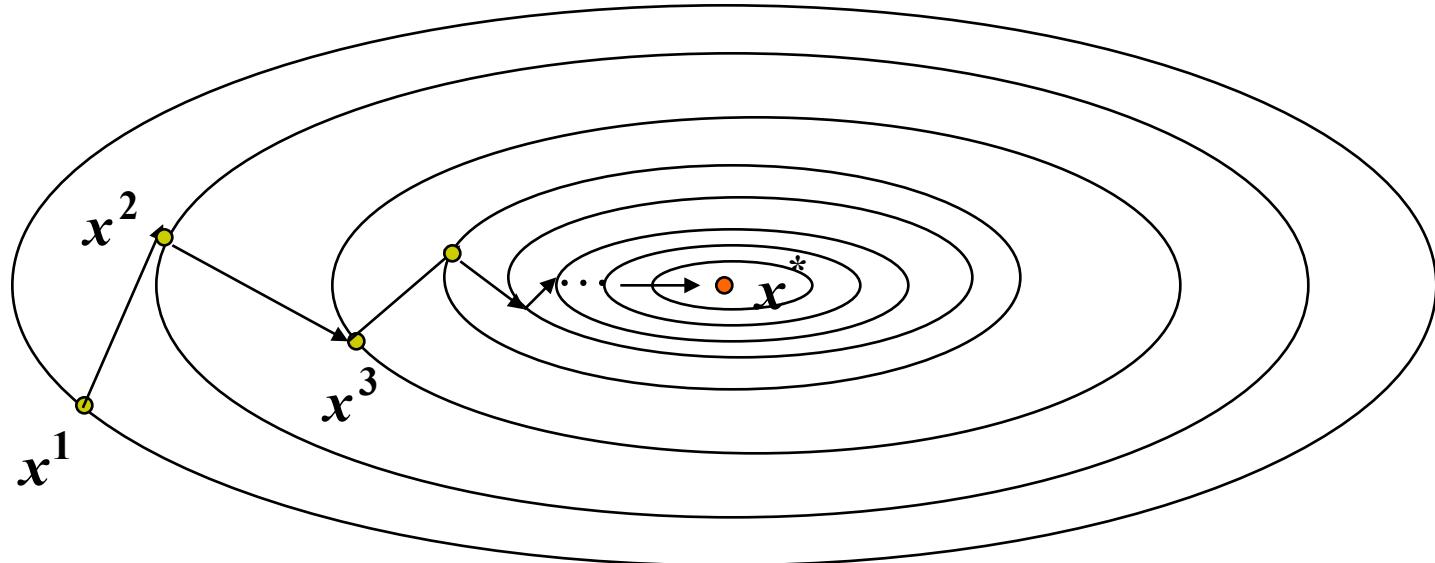
Reducing BERT Pre-Training Time from 3 Days to 76 Minutes

Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, Cho-Jui Hsieh
<https://arxiv.org/abs/1904.00962>

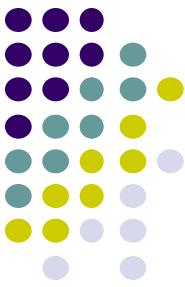


锯齿现象

在极小点附近，目标函数可以用二次函数近似，其等值面近似椭球面。



注 最速下降方向反映了目标函数的一种局部性质。它只是局部目标函数值下降最快的方向。
最速下降法是线性收敛的算法。



1.4.2 牛顿法

- 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。

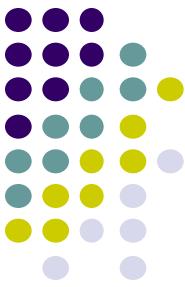
函数 $f(x)$ 在 x_k 处的二次*Taylor*展开为

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

其中 $s = x - x_k$. 将 $q^{(k)}(s)$ 极小化，得到

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

- 对于正定二次函数，一步即可得最优解。
- 由于目标函数在极点附件近似于二次函数，故在初始点接近极小点时，牛顿法收敛速度较快。
- 牛顿法具有局部收敛性，为二阶收敛。



牛顿法

- 当初始点远离最优解时， G_k 不一定是正定的，则牛顿方向不一定为下降方向，其收敛性不能保证。
- 这说明恒取步长因子为1是不合适的，应该采用一维搜索（仅当步长因子 $\{\alpha_k\}$ 收敛1时，牛顿法才是二阶收敛的）。

$$\text{迭代公式为: } \mathbf{d}_k = -G_k^{-1}\mathbf{g}_k, \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

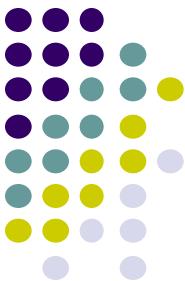
- 带步长因子的牛顿法是总体收敛的。



1.4.3 修正牛顿法

- 牛顿法的主要困难在于Hesse阵 G_k 不正定。这时二次型模型不一定有极小点，甚至没有平稳点。
- Goldstein and Price (1967)当 G_k 非正定时，采用最速下降方向结合方向，给出

$$\mathbf{d}_k = \begin{cases} \mathbf{d}_k^N = -G_k^{-1}\mathbf{g}_k, & \text{angle}\langle \mathbf{d}_k^N, -\mathbf{g}_k \rangle < \frac{\pi}{2} \\ -\mathbf{g}_k, & \text{otherwise} \end{cases}$$

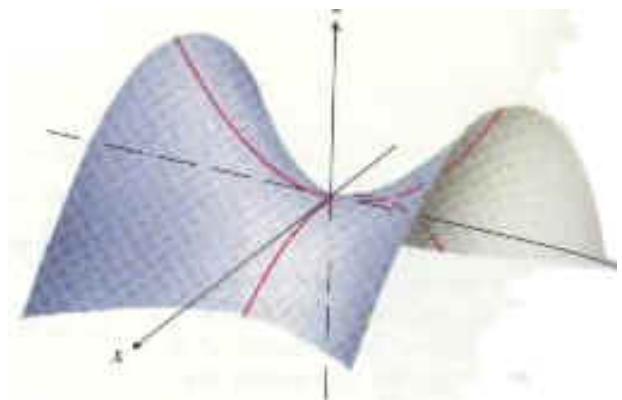


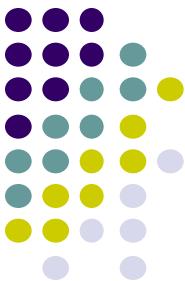
负曲率方向法

- 在鞍点处 $g_k=0$ 而 G_k 不是正半定时，修正牛顿法失效
 - 采用负曲率方向作为搜索方向，可使目标函数值下降
- 若 $G(\mathbf{x})$ 至少有一个负特征值，则 \mathbf{x} 叫做**不定点**
- 若 \mathbf{x} 是一个不定点，方向 \mathbf{d} 满足

$$\mathbf{d}^T G(\mathbf{x}) \mathbf{d} < 0,$$

则 \mathbf{d} 为 $f(\mathbf{x})$ 在 \mathbf{x} 处的负曲率方向





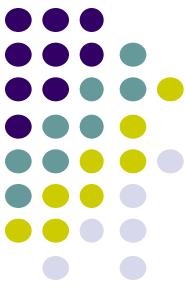
Gill-Murray稳定牛顿法

- 基本思想是：
 - 当 G_k 不定矩阵时，采用修改Cholesky分解强迫矩阵正定；
 - 当 \mathbf{g}_k 趋于0时，采用负曲率方向使函数值下降

设 $\bar{G}_k = G_k + E_k = L_k D_k L_k^T$

构造负曲率方向：

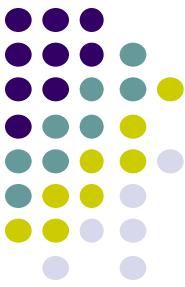
- 1。令 $w_j = d_{jj} - e_{jj}, j = 1, \dots, n$
- 2。求下标 t ，使得 $w_t = \min\{w_j \mid j = 1, \dots, n\}$
- 3。若 $w_t \geq 0$ ，则停止；否则求解 $L_k^T d = e_t$ ，求得方向 d 为负曲率方向。



Gill-Murray稳定牛顿法

1. 给定初始点 $x_0, \varepsilon > 0, k = 1$
2. 计算 g_k 和 G_k
3. 对 G_k 进行修改Cholesky分解, $L_k D_k L_k^T = G_k + E_k$
4. 若 $\|g_k\| > \varepsilon$, 则解方程 $L_k D_k L_k^T d = -g_k$, 求出搜索方向 d_k , 转6。
5. 构造负曲率方向, 若求不出方向 (即 $w_t \geq 0$) , 则停止;
否则, 求出方向 d_k , 再令 $d_k = \begin{cases} -d_k, & \text{若 } g_k^T d_k > 0 \\ d_k, & \text{其他} \end{cases}$
6. 一维搜索, 使得 $f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k)$, 并令 $x_{k+1} = x_k + \alpha_k d_k$
7. 若满足终止条件, 则停止; 否则, $k = k + 1$, 转2。

可证明该方法总体收敛!



Goldfeld修正牛顿法

- 使牛顿方向偏向最速下降方向。更明确地说，将Hesse阵 G_k 改变成 $G_k + \nu_k I$ ，使得 $G_k + \nu_k I$ 正定。比较理想的是， ν_k 为使 $G_k + \nu_k I$ 正定的最小 ν 。

利用Gill和Murray的修改Cholesky分解算法确定 ν_k .即

$$G_k + E = LDL^T \quad (L\text{为下三角阵, } D\text{为对角阵})$$

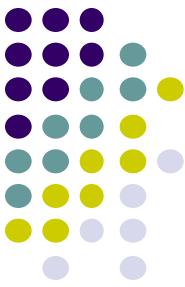
若 $E = 0$, 则 $\nu_k = 0$;

若 $E \neq 0$, 则利用Gershgorin圆盘定理计算 ν_k 的一个上界 b_1 ,

$$b_1 = \left| \min_{1 \leq i \leq n} \left\{ (G_k)_{ii} - \sum_{j \neq i} |(G_k)_{ij}| \right\} \right| \geq \left| \min_i \lambda_i \right|$$

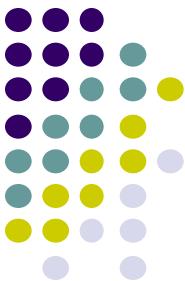
另外, 令 $b_2 = \max_i \{(E)_{ii}\}$ 也是 ν 的一个上界。

最后, 取 $\nu_k = \min \{b_1, b_2\}$



信赖域方法

- 不仅可以用来代替一维搜索，而且也可以解决Hessen矩阵不正定等困难
- 主要思想：
 - 首先选择一个步长 r ，使得在 $\|\mathbf{x}-\mathbf{x}_k\| < r$ 范围内(信赖域)
 - 目标函数用 n 维二次模型来逼近，并以此选择一个搜索方向 \mathbf{s}_k ，取 $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
- 具有牛顿法的快速局部收敛性，又具有理想的总体收敛性。



Levenberg-Marquardt方法

最重要的一类的信赖域是取 l_2 范数，此时原模型等效于

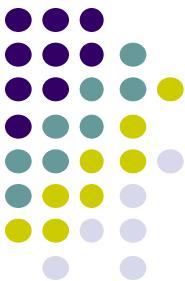
$$\min q^{(k)}(\mathbf{s}) = f_k + \mathbf{g}_k^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T G_k \mathbf{s}, \quad s.t. \|\mathbf{s}\|_2 \leq h_k$$

引入Lagrange函数 $L(\mathbf{s}, \mu) = q^{(k)}(\mathbf{s}) + \frac{1}{2} \mu (\mathbf{s}^T \mathbf{s} - h_k^2)$ 。

根据约束最优化的最优性条件知： $\nabla_s L = 0, \mu \geq 0$

从而可推出

$$\begin{aligned} L(s, \mu_k) &= q^{(k)}(s) + \frac{1}{2} \mu_k (s^T s - h_k^2) = q^{(k)}(s) + \frac{1}{2} \mu_k (s^T s - h_k^2) + s_k^T \nabla_s L(s_k, \mu_k) \\ &= f_k + g_k^T s + \frac{1}{2} s^T G_k s + \frac{1}{2} \mu_k (s^T s - h_k^2) + s_k^T \nabla_s L(s_k, \mu_k) + s_k^T [g_k + G_k s_k + \mu_k s_k] \\ &= q^{(k)}(s_k) + \frac{1}{2} (s - s_k)^T (G_k + \mu_k I)(s - s_k) \end{aligned}$$



Levenberg-Marquardt方法

信赖域采用 l_2 范数定义，原模型等效于

$$\min q^{(k)}(\mathbf{s}) = f_k + \mathbf{g}_k^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T G_k \mathbf{s}, \quad s.t. \|\mathbf{s}\|_2 \leq h_k$$

引入Lagrange函数 $L(\mathbf{s}, \mu) = q^{(k)}(\mathbf{s}) + \frac{1}{2} \mu (\mathbf{s}^T \mathbf{s} - h_k^2)$

根据约束最优化的最优性条件知： $\nabla_s L = 0, \mu \geq 0$

$$L(\mathbf{s}, \mu_k) = q^{(k)}(\mathbf{s}_k) + \frac{1}{2} (\mathbf{s} - \mathbf{s}_k)^T (G_k + \mu_k I) (\mathbf{s} - \mathbf{s}_k)$$

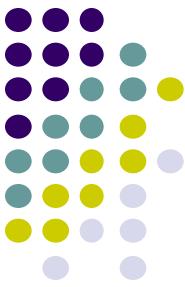
可证明：

总体解的二阶必要条件为 $(G_k + \mu_k I)$ 半正定。

总体解严格最小的充分条件为 $(G_k + \mu_k I)$ 正定。

因此，LM方法都是要确定一个 $\mu_k \geq 0$ ，使得 $(G_k + \mu_k I)$ 正定，

并用 $\mathbf{g}_k + (G_k + \mu_k I)\mathbf{s} = 0$ (即 $\nabla_s L = 0$) 求解 \mathbf{s}_k 。同时可以证明 $\|\mathbf{s}\|_2$ 随 μ 单调减小。

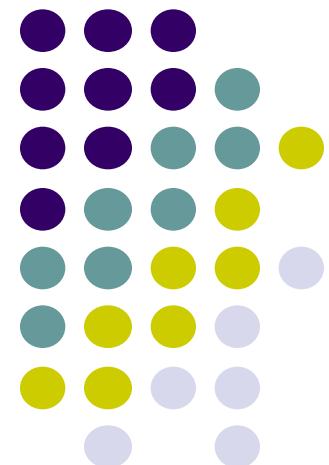


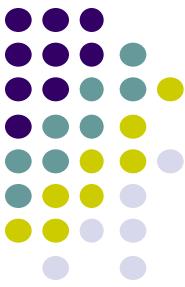
Levenberg-Marquardt方法

- 1。 给定初始点 $x_0, \mu_0 > 0, k = 1$
- 2。 计算 g_k 和 G_k
- 3。 若 $\|g_k\| < \varepsilon$, 停止。
- 4。 分解 $G_k + \mu_k I$, 若不正定, 置 $\mu_k = 4\mu_k$, 重复4直到正定。
- 5。 解 $(G_k + \mu_k I)s = -g_k$, 求出 s_k
- 6。 求 $f(x_k + s_k), q^{(k)}(s_k)$ 和 $r_k = \frac{\Delta f_k}{\Delta q^{(k)}}$
- 7。 若 $r_k < 0.25$, 置 $\mu_{k+1} = 4\mu_k$; 若 $r_k > 0.75$, 置 $\mu_{k+1} = \mu_k / 2$; 否则, $\mu_{k+1} = \mu_k$
- 8。 若 $r_k \leq 0$, 置 $x_{k+1} = x_k$; 否则, 置 $x_{k+1} = x_k + s_k$
- 9。 令 $k = k + 1$, 转2。

1.5 共轭方向法

介于最速下降法与牛顿法之间的一个方法，仅需利用一阶导数，但克服了最速下降法收敛慢的缺点，又避免存储和计算牛顿法所需要的二阶导数信息。





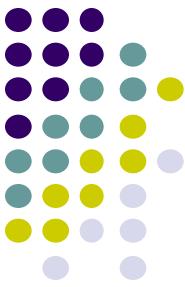
共轭

- 设 G 是 $n \times n$ 对称正定矩阵，若给定 n 维非零向量 \mathbf{d}_1 和 \mathbf{d}_2 ，满足

$$\mathbf{d}_1^T G \mathbf{d}_2 = 0,$$

则称向量 $\mathbf{d}_1, \mathbf{d}_2$ 是 G -共轭的

- 类似，设 n 维非零向量 $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m$ ，如果 $\mathbf{d}_i^T G \mathbf{d}_j = 0$ ($i \neq j$)，则称 $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m$ 是 G -共轭的



共轭

定义 设 A 是 $n \times n$ 的对称正定矩阵，对于 R^n 中的两个非零向量 d^1 和 d^2 ，

若有 $d^{1T} A d^2 = 0$ ，则称 d^1 和 d^2 关于 A 共轭。

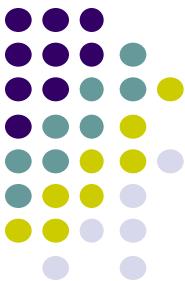
设 d^1, d^2, \dots, d^k 是 R^n 中一组非零向量，如果 它们两两关于 A 共轭，即 $d^{iT} A d^j = 0, i \neq j, i, j = 1, 2, \dots, k$ 。

则称这组方向是关于 A 共轭的，也称它们是一组 A 共轭方向。

注：如果 A 是单位矩阵，则

$$\begin{aligned} d^{1T} \cdot I \cdot d^2 = 0 &\Rightarrow d^{1T} \cdot d^2 = 0 \\ &\Rightarrow d^1 \perp d^2 \end{aligned}$$

共轭是正交的推广。



定理 1. 设 A 是 n 阶对称正定矩阵, d^1, d^2, \dots, d^k 是 k 个 A 共轭的非零向量, 则这个向量组线性无关。

证明 设存在实数 $\alpha_1, \alpha_2, \dots, \alpha_k$, 使得

$$\sum_{i=1}^k \alpha_i d^i = 0,$$

上式两边同时左乘 $d^{j^T} A$, 则有

$$\sum_{i=1}^k \alpha_i d^{j^T} A d^i = 0,$$

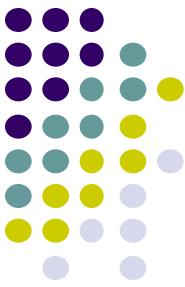
因为 d^1, d^2, \dots, d^k 是 k 个 A 共轭的向量, 所以上式可化简为

$$\alpha_j d^{j^T} A d^j = 0.$$

因为 $d^j \neq 0$, 而 A 是正定矩阵, 所以 $d^{j^T} A d^j > 0$,

所以 $\alpha_j = 0, j = 1, 2, \dots, k$.

因此 d^1, d^2, \dots, d^k 线性无关。



几何意义

设有二次函数

$$f(x) = \frac{1}{2}(x - \bar{x})^T A(x - \bar{x})$$

其中 A 是 $n \times n$ 对称正定矩阵， \bar{x} 是一个定点。

则函数 $f(x)$ 的等值面 $\frac{1}{2}(x - \bar{x})^T A(x - \bar{x}) = c$

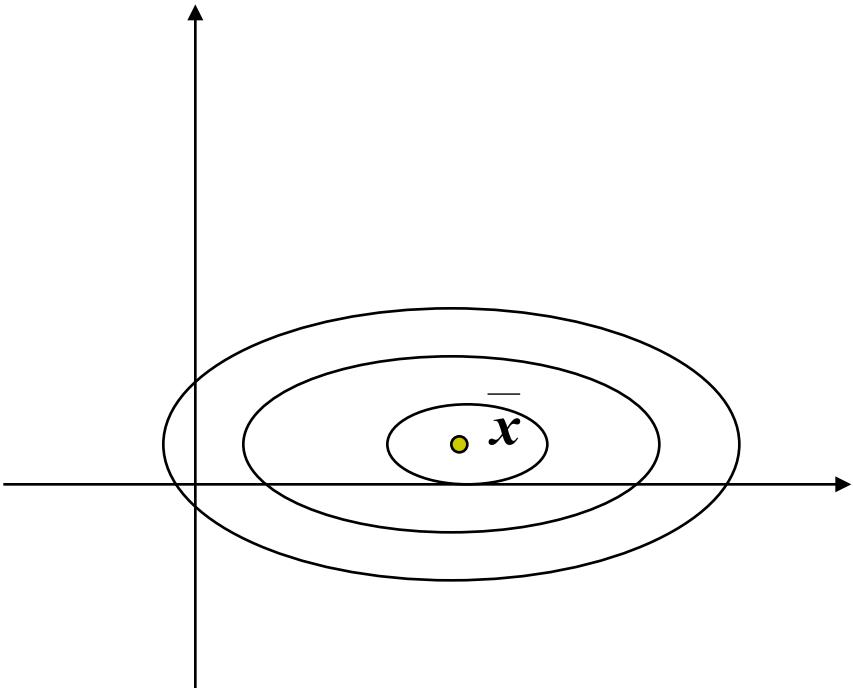
是以 \bar{x} 为中心的椭球面。

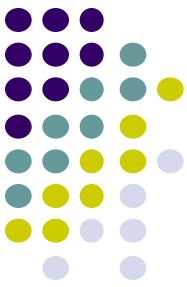
由于 $\nabla f(\bar{x}) = A(\bar{x} - \bar{x}) = 0$,

而 $\nabla^2 f(\bar{x}) = A$,

因为 A 正定，所以 $\nabla^2 f(\bar{x}) = A > 0$,

因此 \bar{x} 是 $f(x)$ 的极小点。





设 $x^{(0)}$ 是在某个等值面上的一点, $d^{(1)}$ 是 R^n 中的一个方向,

$x^{(0)}$ 沿着 $d^{(1)}$ 以最优步长搜索得到点 $x^{(1)}$ 。

则 $d^{(1)}$ 是点 $x^{(1)}$ 所在等值面的切向量。

该等值面在点 $x^{(1)}$ 处的法向量为

$$\nabla f(x^{(1)}) = A(x^{(1)} - \bar{x}).$$

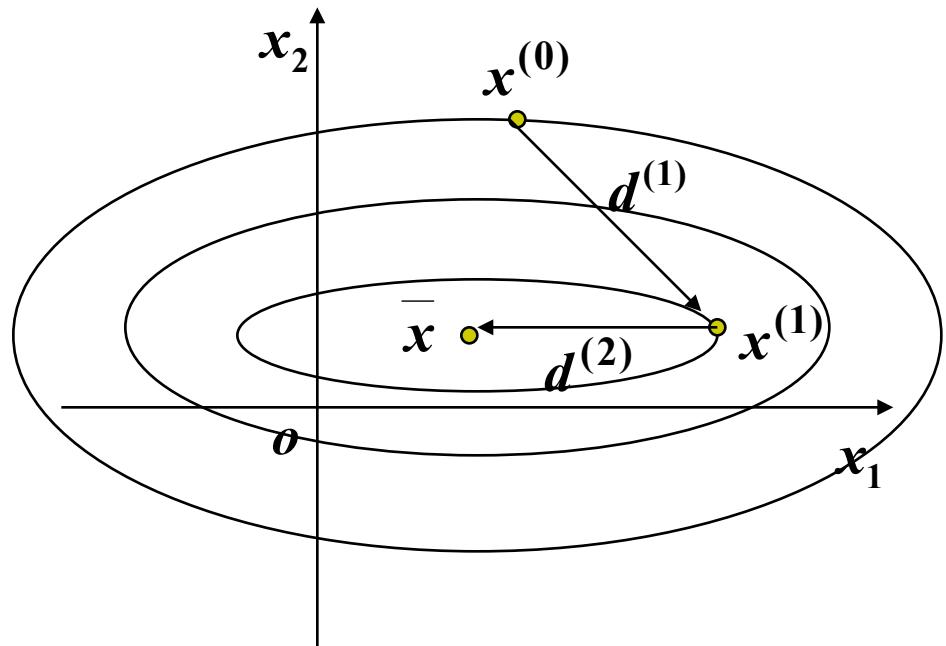
则 $d^{(1)}$ 与 $\nabla f(x^{(1)})$ 正交,

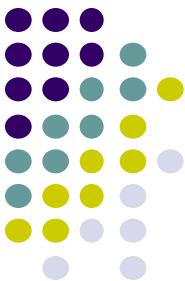
$$\text{即 } d^{(1)T} \nabla f(x^{(1)}) = 0,$$

$$\text{令 } d^{(2)} = \bar{x} - x^{(1)},$$

$$\text{所以 } d^{(1)T} A d^{(2)} = 0,$$

即等值面上一点处的切向量与由这一点指向极小点的向量关于 A 共轭。





定理 2. 设有函数 $f(x) = \frac{1}{2}x^T Ax + b^T x + c$,

其中 A 是 n 阶对称正定矩阵。 $d^{(1)}, d^{(2)}, \dots, d^{(k)}$ 是一组 A 共轭向量。

以任意的 $x^{(1)} \in R^n$ 为初始点, 依次沿 $d^{(1)}, d^{(2)}, \dots, d^{(k)}$ 进行搜索,

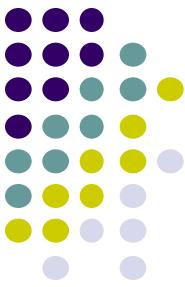
得到点 $x^{(2)}, x^{(3)}, \dots, x^{(k+1)}$, 则 $x^{(k+1)}$ 是函数 $f(x)$ 在 $x^{(1)} + B_k$ 上的极小点, 其中

$$B_k = \{x \mid x = \sum_{i=1}^k \lambda_i d^{(i)}, \lambda_i \in R\}$$

是由 $d^{(1)}, d^{(2)}, \dots, d^{(k)}$ 生成的子空间。特别地, 当 $k = n$ 时, $x^{(n+1)}$ 是 $f(x)$ 在 R^n 上的唯一极小点。

推论 在上述定理条件下, 必有

$$\nabla f(x^{(k+1)})^T d^{(i)} = 0, \quad i = 1, 2, \dots, k.$$



共轭方向法

对于极小化问题

$$\min f(x) = \frac{1}{2} x^T A x + b^T x + c,$$

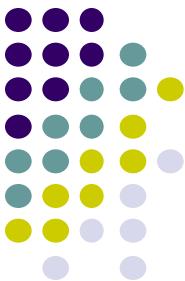
其中 A 是正定矩阵，称下述算法为共轭方向法：

- (1) 取定一组 A 共轭方向 $d^{(1)}, d^{(2)}, \dots, d^{(n)}$ ；
- (2) 任取初始点 $x^{(1)}$ ，依次按照下式由 $x^{(k)}$ 确定点 $x^{(k+1)}$ ，

$$\begin{cases} x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)} \\ f(x^{(k)} + \lambda_k d^{(k)}) = \min_{\lambda} f(x^{(k)} + \lambda d^{(k)}) \end{cases}$$

直到某个 $x^{(k)}$ 满足 $\nabla f(x^{(k)}) = 0$ 。

注 由定理2可知，利用共轭方向法求解上述极小化问题，至多经过 n 次迭代必可得到最优解。



如何选取一组共轭方向？

共轭梯度法

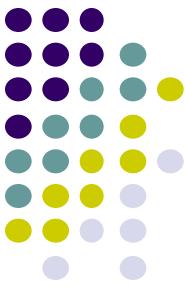
Fletcher – Reeves 共轭梯度法：

$$\min f(x) = \frac{1}{2} x^T A x + b^T x + c$$

其中 $x \in R^n$, A 是对称正定矩阵, $b \in R^n$, c 是常数。

基本思想：将共轭性和最速下降方向相结合，利用已知迭代点处的梯度方向构造一组共轭方向，并沿此方向进行搜索，求出函数的极小点。

以下分析算法的具体步骤。



- (1) 任取初始点 $x^{(1)}$, 第一个搜索方向取为 $d^{(1)} = -\nabla f(x^{(1)})$;
- (2) 设已求得点 $x^{(k+1)}$, 若 $\nabla f(x^{(k+1)}) \neq 0$, 令 $g_{k+1} = \nabla f(x^{(k+1)})$,

则下一个搜索方向 $d^{(k+1)}$ 按如下方式确定:

$$\text{令 } d^{(k+1)} = -g_{k+1} + \beta_k d^{(k)} \quad (1)$$

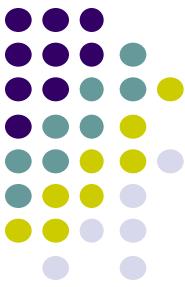
如何确定 β_k ?

要求 $d^{(k+1)}$ 和 $d^{(k)}$ 关于 A 共轭。

则在 (1) 式两边同时左乘 $d^{(k)T} A$, 得

$$0 = d^{(k)T} A d^{(k+1)} = -d^{(k)T} A g_{k+1} + \beta_k d^{(k)T} A d^{(k)}$$

解得 $\beta_k = \frac{d^{(k)T} A g_{k+1}}{d^{(k)T} A d^{(k)}} \quad (2)$



(3) 搜索步长的确定：

已知迭代点 $x^{(k)}$ 和搜索方向 $d^{(k)}$, 利用一维搜索确定最优步长 λ_k ,

即求解 $\min_{\lambda} f(x^{(k)} + \lambda d^{(k)})$ 。

记 $\varphi(\lambda) = f(x^{(k)} + \lambda d^{(k)})$,

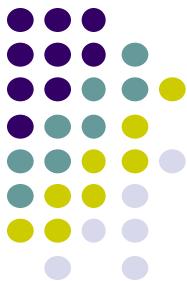
令 $\varphi'(\lambda) = \nabla f(x^{(k)} + \lambda d^{(k)})^T d^{(k)} = 0$,

即有 $[A(x^{(k)} + \lambda d^{(k)}) + b]^T d^{(k)} = 0$,

令 $g_k = \nabla f(x^{(k)}) = Ax^{(k)} + b$, 则有

$$[g_k + \lambda A d^{(k)}]^T d^{(k)} = 0,$$

解得 $\lambda_k = -\frac{g_k^T d^{(k)}}{d^{(k)T} A d^{(k)}}$ (3)



定理3 对于正定二次函数 $f(x) = \frac{1}{2}x^T Ax + b^T x + c$, FR算法在 $m \leq n$ 次一维搜索后即终止，并且对所有的 $i (1 \leq i \leq m)$ ，下列关系成立

$$(1) d^{(i)^T} A d^{(j)} = 0, j = 1, 2, \dots, i - 1;$$

$$(2) g_i^T g_j = 0, j = 1, 2, \dots, i - 1;$$

$$(3) g_i^T d^{(i)} = -g_i^T g_i.$$

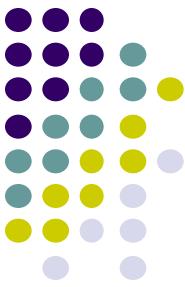
注

(1) 由定理3可知搜索方向 $d^{(1)}, d^{(2)}, \dots, d^{(m)}$ 是 A 共轭的。

(2) 算法中第一个搜索方向必须取负梯度方向，否则构造的搜索方向不能保证共轭性。

(3) 由定理3的 (3) 可知， $g_i^T d^{(i)} = -g_i^T g_i = -\|g_i\|^2 < 0$,

所以 $d^{(i)}$ 是迭代点 $x^{(i)}$ 处的下降方向。

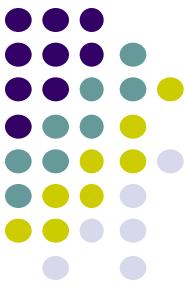


(4) 由定理3, FR算法中 β_i 的计算公式可以简化。

$$\begin{aligned}\beta_i &= \frac{d^{(i)T} A g_{i+1}}{d^{(i)T} A d^{(i)}} = \frac{{g_{i+1}}^T A d^{(i)}}{d^{(i)T} A d^{(i)}} \\ &= \frac{{g_{i+1}}^T A [(x^{(i+1)} - x^{(i)}) / \lambda_i]}{d^{(i)T} A [(x^{(i+1)} - x^{(i)}) / \lambda_i]}\end{aligned}$$

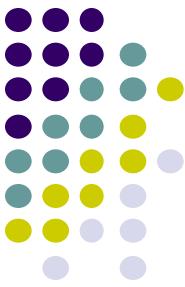
$$\because g_i = \nabla f(x^{(i)}) = A x^{(i)} + b.$$

$$\begin{aligned}\therefore \beta_i &= \frac{{g_{i+1}}^T (g_{i+1} - g_i)}{d^{(i)T} (g_{i+1} - g_i)} = \frac{\|g_{i+1}\|^2}{-d^{(i)T} g_i} \\ &= \frac{\|g_{i+1}\|^2}{\|g_i\|^2} \quad (4)\end{aligned}$$



FR算法步骤：

1. 任取初始点 $x^{(1)}$, 精度要求 ε , 令 $k = 1$ 。
2. 令 $g_1 = \nabla f(x^{(1)})$, 若 $\|g_1\| < \varepsilon$, 停止, $x^{(1)}$ 为所求极小点;
否则, 令 $d^{(1)} = -g_1$, 利用公式 (3) 计算 λ_1 , 令 $x^{(2)} = x^{(1)} + \lambda_1 d^{(1)}$ 。
3. 令 $g_{k+1} = \nabla f(x^{(k+1)})$, 若 $\|g_{k+1}\| < \varepsilon$, 停止, $x^{(k+1)}$ 为所求极小点;
否则, 令 $d^{(k+1)} = -g_{k+1} + \beta_k d^{(k)}$, 其中 β_k 用公式 (4) 计算。
令 $k := k + 1$ 。
4. 利用公式 (3) 计算 λ_k , 令 $x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$, 转3。



例 用FR算法求解下述问题:

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

初始点取为 $x^{(1)} = (2, 2)^T$ 。

解: $\because f(x) = \frac{1}{2}(x_1, x_2) \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \therefore A = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}.$

$$\nabla f(x) = (4x_1, 2x_2)^T.$$

第1次迭代:

令 $d^{(1)} = -g_1 = (-8, -4)^T,$

而 $\lambda_1 = -\frac{g_1^T d^{(1)}}{d^{(1)T} A d^{(1)}}$

$$= -\frac{(8, 4) \begin{bmatrix} -8 \\ -4 \end{bmatrix}}{(-8, -4) \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -8 \\ -4 \end{bmatrix}} = \frac{5}{18}$$



$$\text{所以 } x^{(2)} = x^{(1)} + \lambda_1 d^{(1)}$$

$$= (2, 2)^T + \frac{5}{18}(-8, -4)^T = \left(\frac{-2}{9}, \frac{8}{9}\right)^T$$

第2次迭代：

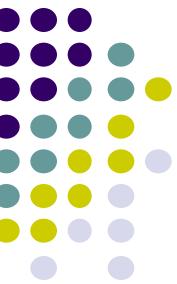
$$\therefore g_2 = \left(\frac{-8}{9}, \frac{16}{9}\right)^T.$$

$$\therefore \beta_1 = \frac{\|g_2\|^2}{\|g_1\|^2} = \frac{\left(\frac{-8}{9}\right)^2 + \left(\frac{16}{9}\right)^2}{8^2 + 4^2} = \frac{4}{81}.$$

$$\therefore d^{(2)} = -g_2 + \beta_1 d^{(1)}$$

$$= \left(\frac{8}{9}, \frac{-16}{9}\right)^T + \frac{4}{81}(-8, -4)^T$$

$$= \frac{40}{81}(1, -4)^T$$



$$\therefore \lambda_2 = -\frac{\mathbf{g}_2^T \mathbf{d}^{(2)}}{\mathbf{d}^{(2)T} A \mathbf{d}^{(2)}}$$

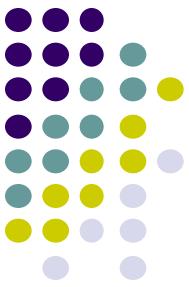
$$= -\frac{\frac{40}{81} \left(\frac{-8}{9}, \frac{16}{9}\right) \begin{bmatrix} 1 \\ -4 \end{bmatrix}}{\left(\frac{40}{81}\right)^2 (1, -4) \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -4 \end{bmatrix}} = \frac{9}{20}$$

$$\therefore \mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \lambda_2 \mathbf{d}^{(2)}$$

$$= \left(\frac{-2}{9}, \frac{8}{9}\right)^T + \frac{9}{20} \times \frac{40}{81} (1, -4)^T$$
$$= (0, 0)^T$$

$$\therefore \mathbf{g}_3 = (0, 0)^T$$

$\therefore \mathbf{x}^{(3)}$ 即为所求极小点。



用于一般非线性函数的共轭梯度法

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in R^n \end{aligned}$$

对用于正定二次函数的共轭梯度法进行修改：

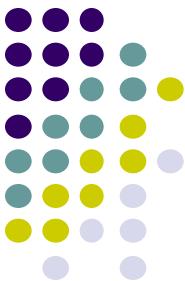
(1) 第一个搜索方向仍取最速下降方向，即 $d^{(1)} = -\nabla f(x^{(1)})$ 。

其它搜索方向按下式计算：

$$d^{(i+1)} = -\nabla f(x^{(i+1)}) + \beta_i d^{(i)},$$

$$\text{其中 } \beta_i = \frac{\|\nabla f(x^{(i+1)})\|^2}{\|\nabla f(x^{(i)})\|^2}.$$

(2) 搜索步长 λ_i 不能利用公式 (3) 计算，需由一维搜索确定。

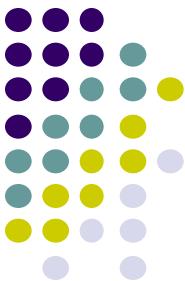


(3) 算法在有限步迭代后不一定能满足停止条件，此时可采取如下措施：

以 n 次迭代为一轮，每次完成一轮搜索后，如果还没有求得极小点，则以上一轮的最后一个迭代点作为新的初始点，取最速下降方向作为第一个搜索方向，开始下一轮搜索。

注 在共轭梯度法中，也可采用其它形式的公式计算 β_i ，如

$$\beta_i = \frac{\mathbf{g}_{i+1}^T (\mathbf{g}_{i+1} - \mathbf{g}_i)}{\mathbf{g}_i^T \mathbf{g}_i} \quad (PRP\text{共轭梯度法})。$$



一般共轭方向法

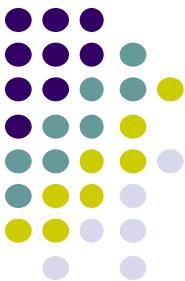
- 1。给定初始点 $\mathbf{x}_0, k = 0$
- 2。计算 $\mathbf{g}_0 = \mathbf{g}(\mathbf{x}_0)$
- 3。计算 \mathbf{d}_0 , 使得 $\mathbf{d}_0^T \mathbf{g}_0 < 0$
- 4。一维搜索得到 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
- 5。计算 \mathbf{d}_{k+1} , 使得 $\mathbf{d}_{k+1}^T G \mathbf{d}_j = 0, j = 0, 1, \dots, k$
- 6。 $k = k + 1$, 转4。

共轭方向法的基本性质：

- 共轭方向 $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_n$, 对于所有 $i \leq n-1$, 有

$$\mathbf{g}^T_{i+1} \mathbf{d}_j = 0, \quad (j=0, \dots, i)$$

- 对于正定二次函数, 共轭方向法至多经过 n 步精确线性搜索终止。



共轭梯度法

考慮目標函數: $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T G \mathbf{x} + \mathbf{b}^T \mathbf{x} + \mathbf{c}$

令 $\mathbf{d}_0 = -\mathbf{g}_0$, 根據一維精確搜尋的性質, $\mathbf{g}_1^T \mathbf{d}_0 = 0$

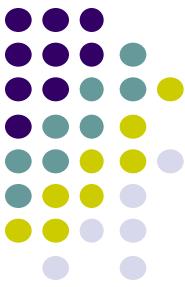
令

$$\mathbf{d}_1 = -\mathbf{g}_1 + \beta_0 \mathbf{d}_0 \quad (*)$$

選擇 β_0 使得 $\mathbf{d}_1^T G \mathbf{d}_0 = 0$. 則 (*) 式兩邊乘 $\mathbf{d}_0^T G$, 可得

$$\beta_0 = \frac{\mathbf{g}_1^T G \mathbf{d}_0}{\mathbf{d}_1^T G \mathbf{d}_0} \xrightarrow{\text{二次函數}} = \frac{\mathbf{g}_1^T (\mathbf{g}_1 - \mathbf{g}_0)}{\mathbf{d}_1^T (\mathbf{g}_1 - \mathbf{g}_0)} = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0}$$

- 共轭梯度法仅比最速下降法稍微复杂一点，但具有二次终止性（对于二次函数，算法在有限步终止）。



共轭梯度法

$$\beta_0 = \frac{\mathbf{g}_1^T G \mathbf{d}_0}{\mathbf{d}_1^T G \mathbf{d}_0} \xrightarrow{\text{二次函数}} = \frac{\mathbf{g}_1^T (\mathbf{g}_1 - \mathbf{g}_0)}{\mathbf{d}_1^T (\mathbf{g}_1 - \mathbf{g}_0)} = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0}$$

类似地利用共轭法的基本性质进行推导，可得一般在第 $k+1$ 次迭代时，

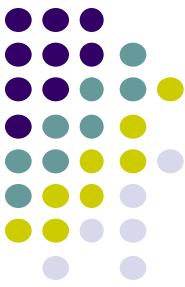
$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k, \text{其中}$$

$$\beta_k = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{d}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k)} \text{ (Crowder - Wolfe公式)}$$

$$= \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} \text{ (Fletcher - Reeves公式)}$$

$$= - \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{d}_k^T \mathbf{g}_k} \text{ (Dixon公式)}$$

- 共轭梯度法仅比最速下降法稍微复杂一点，但具有二次终止性（对于二次函数，算法在有限步终止）。

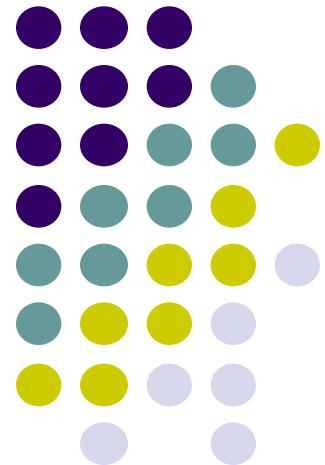


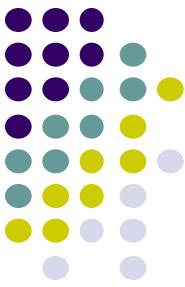
再开始FR共轭梯度法

- 对于一般非二次函数， n 步后共轭梯度法产生的搜索方向不再共轭。
- 再开始共轭梯度法
 - n 步后周期性采用最速下降方向作为搜索方向
 - 在精确线性搜索条件下，总体收敛。
 - 满足Lipschitz条件，线性搜索由Wolfe-Powell准则确定，且下降方向与负梯度方向小于90度，总体收敛。
 - 具有 n 步二阶收敛性。（ n 步可求得二次凸函数的极小点，相当牛顿法的执行一步）

1.6 拟牛顿法

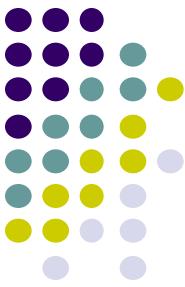
Hesse矩阵的计算工作量大，能否不算或者少算？





拟牛顿法

- Hesse矩阵的计算工作量大，有时目标函数的Hesse阵很难计算。
- 拟牛顿法利用目标函数和一阶导数，来构造目标函数的曲率近似，而不需要明显形成Hesse阵，同时具有收敛速度快的优点。



拟牛顿条件

f 在 \mathbf{x}_{k+1} 附近的二次近似为

$$f(\mathbf{x}) \approx f(\mathbf{x}_{k+1}) + \mathbf{g}_{k+1}^T (\mathbf{x} - \mathbf{x}_{k+1}) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_{k+1})^T G_{k+1} (\mathbf{x} - \mathbf{x}_{k+1}),$$

对上式求导，有

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}_{k+1} + G_{k+1} (\mathbf{x} - \mathbf{x}_{k+1}),$$

令 $\mathbf{x} = \mathbf{x}_k$, $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, 得

$$G_{k+1}^{-1} \mathbf{y}_k \approx \mathbf{s}_k$$

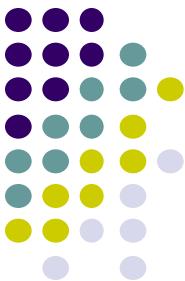
对于二次函数 f , 上述关系式精确成立。

要求在拟牛顿法中构造出 Hesse 逆近似 H_{k+1} , 满足

$$H_{k+1} \mathbf{y}_k = \mathbf{s}_k,$$

称为拟牛顿条件。或构造 Hesse 近似

$$B_{k+1} \mathbf{s}_k = \mathbf{y}_k$$

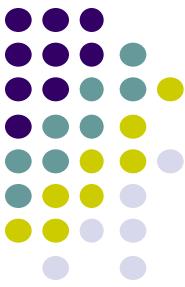


一般的拟牛顿算法

一般拟牛顿算法：

- 1。 给初值 $\mathbf{x}_0 \in R^n, H_0 \in R^{n \times n}, 0 \leq \varepsilon \leq 1, k = 0.$
- 2。 若 $\|\mathbf{g}_k\| \leq \varepsilon$, 则停止; 否则, 计算 $\mathbf{d}_k = -H_k \mathbf{g}_k$
- 3。 沿方向 \mathbf{d}_k 线性搜索求 α_k , 令 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
- 4。 校正 H_k 产生 H_{k+1} , 使得拟牛顿条件满足。
- 5。 $k = k + 1$, 转步2。

- 优点：
 - 仅需一阶导数
 - H_k 保持正定, 具有下降性。
 - 迭代每次需要 $O(n^2)$ 次乘法。 (牛顿法 $O(n^3)$ 次乘法)



对称秩一校正(SR1校正)

希望有 $H_{k+1} = H_k + E_k$, 其中 E_k 为一个低阶矩阵。在秩一校正情形下,
 $E_k = uv^T$.

根据拟牛顿条件有, $(H_k + uv^T)y_k = s_k \Rightarrow (v^T y_k)u = s_k - H_k y_k$, 故
 u 必在方向 $s_k - H_k y_k$ 上。

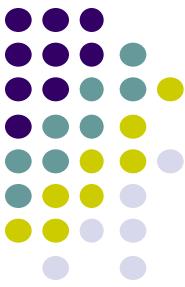
取 $u = \frac{1}{v^T y_k}(s_k - H_k y_k)$ 因为 Hesse 为对称阵, 故取 $v = (s_k - H_k y_k)$

那么 $H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}$, 称为对称秩一校正(SR1校正).

性质

1. 对于二次函数, 不需要精确一维搜索, 具有 n 步终止性质, $H_n = G^{-1}$

2. 不能保持正定性, 仅当 $(s_k - H_k y_k)^T y_k > 0$ 时。但这个条件往往很难满足。
这使得 SR1 校正在应用中受到限制。



DFP校正

设秩二校正为 $H_{k+1} = H_k + auu^T + bvv^T$.

若要拟牛顿条件 $(H_k + auu^T + bvv^T)y_k = s_k$ 成立,

对于 u, v 一个明显的取法为 $u = s_k, v = H_k y_k$ 。

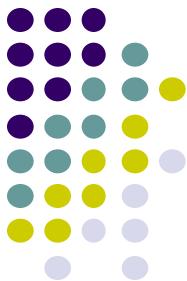
同时令 : $au^T y_k = 1, bv^T y_k = -1$, 可知(*)恒成立。

$$\text{那么 } a = \frac{1}{s_k^T y_k}, b = -\frac{1}{y_k^T H_k y_k}$$

那么 $H_{k+1} = H_k + \frac{s_k^T s_k}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k^T}{y_k^T H_k y_k}$, 称为DFP校正.

性质

- 1。 对于二次函数, 精确一维搜索, 具有 n 步终止性质, $H_n = G^{-1}$
- 2。 对于二次函数, $H_0 = I$ 时, 产生共轭方向与共轭梯度。
- 3。 校正保持正定性, 下降性质成立。
- 4。 具有超线性收敛。
- 5。 采用精确线性搜索时, 对于凸函数, 总体收敛。



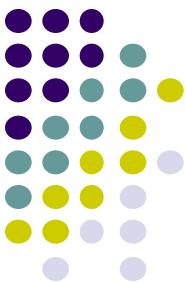
L-BFGS算法

- 一种非常流行和成功的拟牛顿法
- Limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm
- 有大量开源的实现

$$H_{i+1} = H_i + \frac{s_i s_i^T}{s_i^T q_i} - \frac{H_i q_i q_i^T H_i}{q_i^T H_i q_i} + [\nabla f(X_{i+1}) - \nabla f(X_i)]^T H_i [\nabla f(X_{i+1}) - \nabla f(X_i)] w w^T \quad \dots [14]$$

where

$$\begin{aligned} w &= \frac{s_i}{s_i^T q_i} - \frac{H_i q_i}{q_i^T H_i q_i} \\ &\text{codelast.com} \\ &= H_i + \frac{X_{i+1} - X_i}{(X_{i+1} - X_i)^T [\nabla f(X_{i+1}) - \nabla f(X_i)]} \\ &\quad - \frac{H_i [\nabla f(X_{i+1}) - \nabla f(X_i)]}{[\nabla f(X_{i+1}) - \nabla f(X_i)]^T H_i [\nabla f(X_{i+1}) - \nabla f(X_i)]} \quad \text{codelast.com} \quad \dots [15] \end{aligned}$$



L-BFGS算法

Two loops approach

$$y_k = g_{k+1} - g_k \quad \rho_k = \frac{1}{y_k^T s_k}$$

$$q = g_k$$

For $i = k-1, k-2, \dots, k-m$

$$\alpha_i = \rho_i s_i^T q$$

$$q = q - \alpha_i y_i$$

$$H_k = y_{k-1}^T s_{k-1} / y_{k-1}^T y_{k-1}$$

$$z = H_k q$$

For $i = k-m, k-m+1, \dots, k-1$

$$\beta_i = \rho_i y_i^T z$$

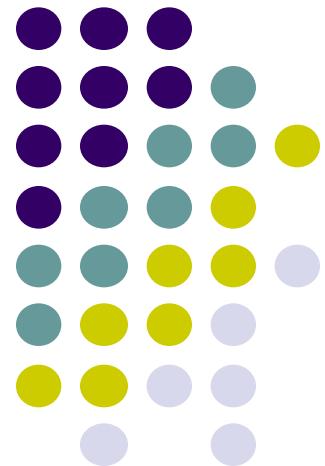
$$z = z + s_i (\alpha_i - \beta_i)$$

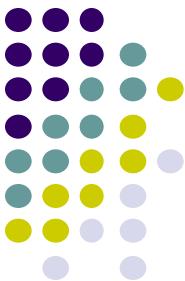
Stop with $H_k g_k = z$

The End

新浪微博: @浙大张宏鑫

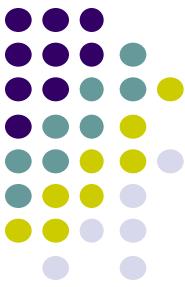
微信公众号:





Homework 05

- 实现SVM:
 - Kernel: 线性核, 指数核
 - 使用python
- 二次规划方法:
 - 有效集方法
 - 奖励 => 其他更高效的优化方法
- 5次编程作业, 请统一提供一个实验报告
 - 格式与课程论文的要求相似



The End

- 数学学习是一种修行
- 大音希声，大象无形
 - 节选自《道德经》



无所得，即是得
以是得，无所得
--《金刚经》

新浪微博：@浙大张宏鑫

微信公众号：

