南昌大学
NANCHANG UNIVERSITY

# 大数据优化：理论、算法及其应用

——来源于《最优化计算方法》（高教社）

★★★ 主讲人：彭振华 ★★★

联系方式：zhenhuapeng@ncu.edu.cn
zhenhuapeng@whu.edu.cn
15870605317（微信同号）

研究兴趣：1. 非凸非光滑优化算法与理论
2. 智能决策
3. 智能计算与机器学习

数学与计算机学院

2025年

课件资源：https://zhenhuapeng.github.io//coursematerials/

# 目录

1. 大数据优化简介（3课时）
   - I. 模型与基本概念
   - II. 优化建模技术
   - III. 应用实例
   - IV. 求解器与大模型
2. 基础知识（7课时）
   - I. 范数与导数
   - II. 凸集与凸函数
   - III. 共轭函数与次梯度
3. 无约束优化理论（2课时）
   - I. 最优性问题解的存在性
   - II. 无约束可微问题的最优性理论
   - III. 无约束不可微问题的最优性理论
4. 无约束优化算法（15课时）
   - I. 线搜索方法
   - II. (次)梯度类算法
   - III. 共轭梯度算法
   - IV. (拟)牛顿类算法
   - V. 信赖域算法
   - VI.非线性最小二乘算法
5. 约束优化理论（6课时）
   - I. 对偶理论
   - II. 一般约束优化问题的最优性理论
   - III. 凸优化问题的最优性理论
6. 约束优化算法（3课时）
   - I. 罚函数法
   - II. 增广拉格朗日函数法
7. 复合优化算法（9课时）
   - I. PPA
   - II. BCD
   - III. PGD
   - IV. SGD
   - V. SDP

南昌大学
NANCHANG UNIVERSITY

# 无约束优化算法

线搜索方法

梯度类算法

共轭梯度算法

(拟)牛顿算法

信赖域算法与最小二乘

第 四 部分

# 拟牛顿算法

$$\min_x \ f(x)$$

**海瑟矩阵计算存储困难，矩阵求逆、修正或求解牛顿方程代价高**

- 设 $f(x)$ 是二阶连续可微函数. 对 $\nabla f(x)$ 在点 $x^{k+1}$ 处一阶泰勒近似, 得

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

- 令 $x = x^k$, 且 $s^k = x^{k+1} - x^k$ 为点差, $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ 为梯度差, 得

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

- 现忽略高阶项, 只希望近似海瑟矩阵的矩阵 $B^{k+1}$ 或其逆矩阵 $H^{k+1}$ 满足方程

$$B^{k+1} s^k = y^k \quad \text{或} \quad H^{k+1} y^k = s^k$$

割线方程

$$\min_x \ f(x)$$

由于近似矩阵必须保证迭代收敛，$B^k$ 正定也是必须的，即

$$\left(s^k\right)^T B^{k+1} s^k > 0$$

$$\left(s^k\right)^T y^k > 0$$

曲率条件

Wolfe第二个条件

$$\nabla f\left(x^k + \alpha_k d^k\right)^T d^k \geq c_2 \nabla f\left(x^k\right)^T d^k \ \left(c_2 \in (0,1)\right)$$

$$\nabla f\left(x^{k+1}\right)^T s^k \geq c_2 \nabla f\left(x^k\right)^T s^k$$

$$\left(\nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right)\right)^T s^k \geq (c_2 - 1) \nabla f\left(x^k\right)^T s^k$$

$$\left(y^k\right)^T s^k \geq (c_2 - 1) \nabla f\left(x^k\right)^T s^k > 0$$

下降方向$s^k$

## 拟牛顿算法

**Algorithm 6** 拟牛顿算法框架

**Input:** 初始坐标 $x^0 \in \mathbb{R}^n$, 初始矩阵 $B^0 \in \mathbb{R}^{n\times n}$(或 $H^0$), $k = 0$.

**Output:** $x^K, B^K$(或 $H^K$).

1: 检查初始元素.
2: **while** 未达到停机准则 **do**
3: 计算方向 $d^k = -(B^k)^{-1}\nabla f(x^k)$ 或 $d^k = -H^k\nabla f(x^k)$.
4: 通过线搜索 (Wolfe) 产生步长 $\alpha_k > 0$, 令 $x^{k+1} = x^k + \alpha_k d^k$.
5: 更新海瑟矩阵的近似矩阵 $B^{k+1}$ 或其逆矩阵 $H^{k+1}$.
6: $k \leftarrow k + 1$.
7: **end while**

## 秩一更新(SR1)

$$B^{k+1} = B^k + auu^T$$

割线方程

$$B^{k+1}s^k = y^k \qquad (B^k + auu^T)s^k = y^k$$

$$auu^Ts^k := a(u^Ts^k)u = y^k - B^ks^k$$

取$u = y^k - B^ks^k$，得

$$a(u^Ts^k) = a(y^k - B^ks^k)^Ts^k = 1$$

$$a = 1/((y^k - B^ks^k)^Ts^k)$$

$$B^{k+1} = B^k + 1/((y^k - B^ks^k)^Ts^k) \cdot (y^k - B^ks^k)(y^k - B^ks^k)^T$$

$$H^{k+1} = H^k + 1/((s^k - H^ky^k)^Ty^k) \cdot (s^k - H^ky^k)(s^k - H^ky^k)^T$$

$H^k = (B^k)^{-1}$

# 拟牛顿算法

- 即使 $B^k$ 正定，由秩一公式更新的 $B^{k+1}$ 无法保证正定.
- **秩一更新公式使 $B^{k+1}$ 正定的充分条件：** 使用秩一更新公式从 $B^k$ 更新 $B^{k+1}$, $B^{k+1}$ 正定的充分条件可以是：（1）$B^k$ 正定；（2）$u^T s^k > 0$.

- Proof. 设 $0 \neq w \in R^n$, 则

$$w^T B^{k+1} w = w^T B^k w + (w^T u u^T w)/(u^T s^k) = w^T B^k w + (u^T w)^2/(u^T s^k) > 0$$

由于无法保证 $u^T s^k$ 或 $v^T y^k$ 恒大于0,
上述的秩一更新公式一般不用.

## 秩二更新(BFGS)

$$B^{k+1} = B^k + auu^T + bvv^T$$

割线方程

$$B^{k+1}s^k = y^k \qquad (B^k + auu^T + bvv^T)s^k = y^k$$

$$auu^Ts^k + bvv^Ts^k := a(u^Ts^k)u + b(v^Ts^k)v = y^k - B^ks^k$$

取$u = y^k$，$v = B^ks^k$，得

$$a(u^Ts^k) = a(y^k)^Ts^k = 1 \qquad b(v^Ts^k) = b(s^k)^TB^ks^k = -1$$

$$a = 1/((y^k)^Ts^k) \qquad b = -1/((s^k)^TB^ks^k)$$

$$B^{k+1} = B^k + 1/((y^k)^Ts^k) \cdot y^k(y^k)^T - 1/((s^k)^TB^ks^k) \cdot (B^ks^k)(B^ks^k)^T$$

$$H^{k+1} = (I - (y^k(s^k)^T)/(y^k)^Ts^k))^TH^k(I - (y^k(s^k)^T)/(y^k)^Ts^k)) + (s^k(s^k)^T)/((y^k)^Ts^k)$$

$H^k = (B^k)^{-1}$

BFGS 公式产生的 $B^{k+1}$ 或 $H^{k+1}$ 是否正定 ？

- BFGS 公式使拟牛顿矩阵正定的充分条件:

1. $B^k$ 或 $H^k$ 正定;
2. 满足曲率条件$(s^k)^T y^k > 0, \forall k \in N+$.

确定步长时使用Wolfe 准则线搜索即可满足曲率条件

- DFP:

$$H^{k+1} = H^k + 1/((y^k)^T s^k) \cdot s^k (s^k)^T - 1/((y^k)^T B^k y^k) \cdot (H^k y^k)(H^k y^k)^T$$

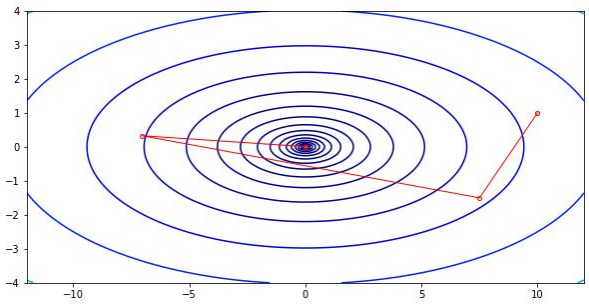
$$B^{k+1} = (I - s^k (y^k)^T/(s^k)^T y^k))^T B^k (I - s^k (y^k)^T/(s^k)^T y^k)) + (y^k (y^k)^T)/((s^k)^T y^k)$$

# 拟牛顿算法

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x[0]**2 + 10*x[1]**2
def grad_f(x):
    return np.array([2*x[0], 20*x[1]])
def wolfe_line_search(f, grad_f, x, direction, alpha=1.0, beta=0.5, c1=1e-3, c2 = 1e-2, max_iter=100):
    fx = f(x)
    grad = grad_f(x)
    slope = np.dot(grad, direction)

    for _ in range(max_iter):
        candidate = x + alpha * direction
        if np.logical_and(f(candidate) <= fx + c1 * alpha * slope, np.dot(grad_f(candidate),direction) >= c2 * slope):
            return alpha
        alpha *= beta
    return alpha
def bfgs_optimizer(f, grad_f, x0, max_iter=100, tol=1e-6):
    n = len(x0)
    H = np.eye(n)  # 初始Hessian逆近似矩阵
    x = x0.copy()
    history = [x.copy()]
    for k in range(max_iter):
        grad = grad_f(x)
        if np.linalg.norm(grad) < tol:
            break
        d = -H @ grad
        alpha = wolfe_line_search(f, grad_f, x, d)
        s = alpha * d
        x_new = x + s
        y = grad_f(x_new) - grad
        if np.dot(y, s) > 1e-10:
            rho = 1.0 / np.dot(y, s)
            I = np.eye(n)
            H = (I - rho * np.outer(s, y)) @ H @ (I - rho * np.outer(y, s)) + rho * np.outer(s, s)
        x = x_new
        history.append(x.copy())
    return history, k
x0 = np.array([10, 1])
gd_history, k = bfgs_optimizer(f, grad_f, x0)
print("最优解为", gd_history[-1])
print("梯度范数值为", np.linalg.norm(grad_f(gd_history[-1])))
x = np.linspace(-12, 12, 100)
y = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + 10*Y**2  # 定义目标函数
# 绘制等高线图
plt.figure(figsize=(10, 5))
plt.contour(X, Y, Z, levels=np.logspace(-2, 3, 20), cmap='jet')

plt.plot([x[0] for x in gd_history], [x[1] for x in gd_history], marker='o',markersize=4, linewidth=1,
color = 'red', markerfacecolor='none', label='Gradient Descent')
plt.show()
```



```
最优解为 [-4.21148797e-08  5.25312383e-09]
梯度范数值为 1.3465799773659666e-07
```

# 拟牛顿算法

- 尽管DFP 格式与BFGS 对偶, 但从实际效果而言, DFP 格式的求解效率整体上不如BFGS 格式. M.J.D. Powell 曾求解问题

$$\min_x \ 0.5\ ||x||^2$$

设置初始值$B^0$和$x_0$分别为

$$\begin{bmatrix} 1 & 0 \\ 0 & \tan^2\psi \end{bmatrix} \qquad \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}$$

- 例题：$\min_{x,y}\ 0.5x^2+y^2$ 初始点 $(2,1)^T$
- 练习：$\min_{x,y}\ x^2+2y^2$ 初始点 $(1,1)^T$

当误差阈$\epsilon = 10^{-4}$时, 分别取$\lambda$为不同的值, 使用BFGS 算法与DFP 算法所产生的迭代步数分别如下表所示

表: BFGS 方法的迭代次数

| $\lambda \backslash \epsilon$ | 0.1 | 0.01 | $10^{-4}$ | $10^{-8}$ |
|---|---|---|---|---|
| 10 | 5 | 6 | 8 | 10 |
| 100 | 7 | 8 | 10 | 12 |
| $10^4$ | 12 | 13 | 15 | 17 |
| $10^6$ | 17 | 18 | 20 | 22 |
| $10^9$ | 24 | 25 | 27 | 29 |

表: DFP 方法的迭代次数

| $\lambda \backslash \epsilon$ | 0.1 | 0.01 | $10^{-4}$ | $10^{-8}$ |
|---|---|---|---|---|
| 10 | 10 | 13 | 16 | 19 |
| 30 | 25 | 32 | 37 | 40 |
| 100 | 80 | 99 | 107 | 111 |
| 300 | 237 | 290 | 307 | 313 |
| $10^3$ | 787 | 958 | 1006 | 1014 |

# 拟牛顿算法

**BFGS全局收敛性: 设初始矩阵 $B^0$ 是对称正定矩阵, 目标函数 $f(x)$ 是二阶连续可微函数, 下水平集 $L = \{x : f(x) \leqslant f(x_0)\}$凸。且存在$m,M \in R_+$ 使得对$\forall z \in R^n, x \in L$ 满足**

$$m \|z\|^2 \leqslant z^T \nabla^2 f(x) z \leqslant M \|z\|^2$$

**那么BFGS 格式结合Wolfe 线搜索的拟牛顿算法全局收敛到 $f(x)$ 的极小值点$x^*$ .**

# 拟牛顿算法

**BFGS局部收敛性**: 设 $f(x)$ 二阶连续可微, 点列$\{xk\}$ 是由BFGS 格式产生的,并收敛于$x^*$ , $\nabla^2 f(x^*)$ 是对称正定矩阵. 设初始矩阵 $B^0$ 为任意的对称正定矩阵, 那么存在 $0 \leqslant c < 1$, $K \in N^+$, 使得对 $\forall k > K$, 成立

$$f\left(x^{k+1}\right)-f\left(x^*\right) \leq c^{k-K+1}\left(f\left(x^K\right)-f\left(x^*\right)\right), \quad \sum_{k=0}^{\infty}\left\|x^k-x^*\right\|<\infty.$$

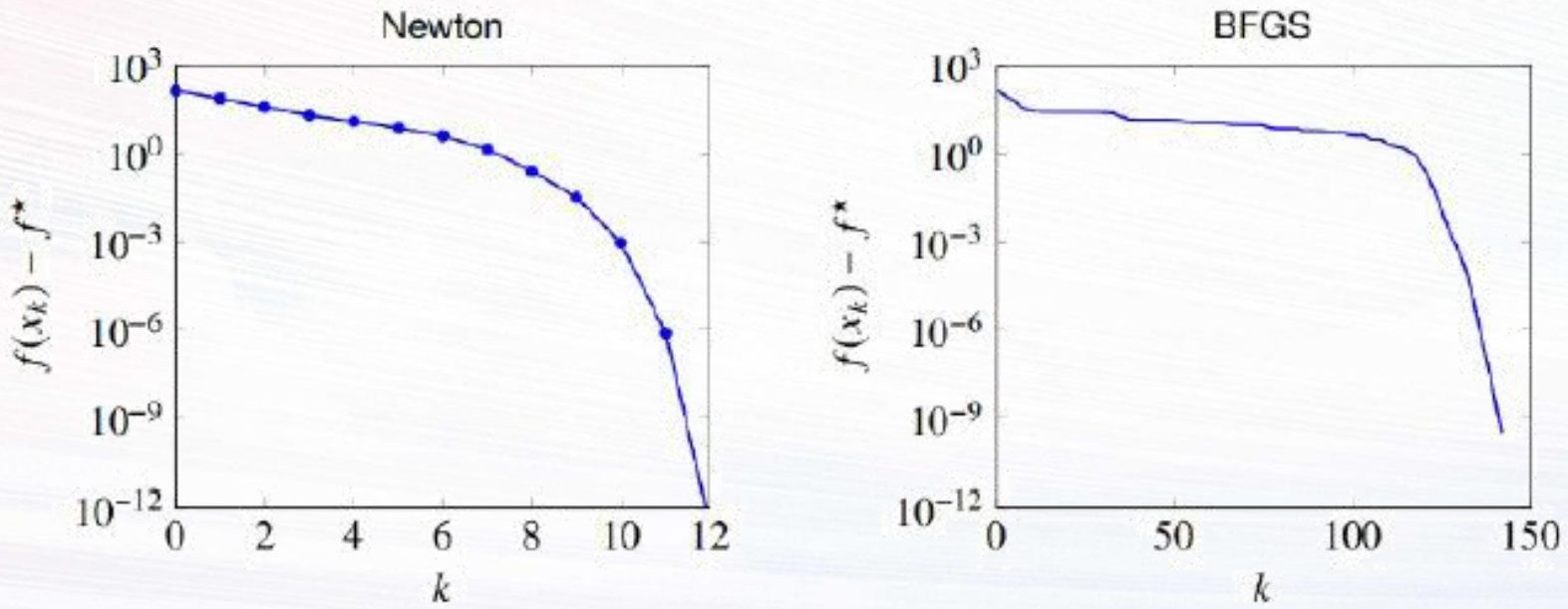
R-超线性收敛

**BFGS 的Q-超线性收敛速度**: 除要求BFGS 局部收敛性的假设外, 再要求 $f$ 的海瑟矩阵在 $x^*$ 处Lip- 连续, 则 $\{x^k\}$ 为Q- 超线性收敛到 $x^*$

以BFGS 格式为代表的拟牛顿类算法由于仅仅使用了海瑟矩阵的近似, 因此很难达到二阶收敛速度, 最多只能达到Q-超线性收敛速度. 但是, 由于拟牛顿方法对近似矩阵的更新代价可能远小于牛顿方法计算海瑟矩阵的代价, 因此它在大规模问题中的开销可能远小于牛顿算法, 较为实用.

# 拟牛顿算法

$$\min_{x\in R^{100}} c^T x - \sum_{i=1}^{500} \ln\left(b_i - a_i^T x\right)$$




虽然 BFGS 方法的迭代次数显著得多, 但由于牛顿法每次迭代的计算代价为$O(n^3)$加上计算海瑟矩阵的代价, 而 BFGS 方法的每步计算代价仅为$O(n^2)$, 因此 BFGS 算法可能更快取得优势解.

# 信赖域算法

$$f\left(x^k+d\right)=f\left(x^k\right)+\nabla f\left(x^k\right)^T d+\frac{1}{2}d^T\nabla^2 f\left(x^k+td\right)d,\ t\in(0,1)$$

近似

$$m_k(d)=f\left(x^k\right)+\nabla f\left(x^k\right)^T d+\frac{1}{2}d^T B^k d$$

$$\min_d m_k(d)\quad s.t.\ \|d\|\le\Delta_k.$$

$$\rho_k=\frac{f\left(x^k\right)-f\left(x^k+d^k\right)}{m_k(0)-m_k\left(d^k\right)}$$

- $\rho_k$接近于1，$m_k(d)$近似$f(x)$比较成功，应扩大$\Delta_k$;
- $\rho_k$非常小甚至为负，过分相信$m_k(d)$，近似效果差,应缩小$\Delta_k$

# 信赖域算法

**Algorithm 1** 信赖域算法

1: 给定最大半径$\Delta_{\max}$，初始半径$\Delta_0$，初始点$x^0$，$k \leftarrow 0$.
2: 给定参数$0 \leq \eta < \bar{\rho}_1 < \bar{\rho}_2 < 1$，$\gamma_1 < 1 < \gamma_2$.
3: **while** 未达到收敛准则 **do**
4: 计算子问题(2)得到迭代方向$d^k$.
5: 根据(3)式计算下降率$\rho_k$.
6: 更新信赖域半径：

$$\Delta_{k+1} = \begin{cases} \gamma_1 \Delta_k, & \rho_k < \bar{\rho}_1, \\ \min\{\gamma_2 \Delta_k, \Delta_{\max}\}, & \rho_k > \bar{\rho}_2 \text{ 以及 } \|d^k\| = \Delta_k, \\ \Delta_k, & \text{其他}. \end{cases}$$

7: 更新自变量：

$$x^{k+1} = \begin{cases} x^k + d^k, & \rho_k > \eta, \\ x^k, & \text{其他}. \end{cases} \quad /* \text{ 只有下降比例足够大才更新} */$$

8: $k \leftarrow k + 1$.
9: **end while**

- $\rho_1 = 0.25$
- $\rho_2 = 0.75$
- $\gamma_1 = 0.25$
- $\gamma_2 = 2$

# 信赖域算法

$$\min_s q(s) := g^T s + \frac{1}{2} s^T B s$$

**Algorithm 7** 截断共轭梯度法 1 (Steihaug-CG)

1: 给定精度 $\varepsilon > 0$，初始化 $s^0 = 0, r^0 = g, p^0 = -g$，$k \leftarrow 0$.
2: **if** $\|p^0\| \leq \varepsilon$ **then**
3: 算法停止，输出 $s = 0$.
4: **end if**
5: **LOOP**
6: **if** $(p^k)^{\mathrm{T}} B p^k \leq 0$ **then**
7: 计算 $\tau > 0$ 使得 $\|s^k + \tau p^k\| = \Delta$.
8: 算法停止，输出 $s = s^k + \tau p^k$.
9: **end if**
1: 计算 $\alpha_k = \frac{\|r^k\|^2}{(p^k)^{\mathrm{T}} B p^k}$ 更新 $s^{k+1} = s^k + \alpha_k p^k$.
2: **if** $\|s^{k+1}\| \geq \Delta$ **then**
3: 计算 $\tau > 0$ 使得 $\|s^k + \tau p^k\| = \Delta$.
4: 算法停止，输出 $s = s^k + \tau p^k$.
5: **end if**
6: 计算 $r^{k+1} = r^k + \alpha_k B p^k$
7: **if** $\|r^{k+1}\| < \varepsilon \|r^0\|$ **then**
8: 算法停止，输出 $s = s^{k+1}$.
9: **end if**
10: 计算 $\beta_k = \frac{\|r^{k+1}\|^2}{\|r^k\|^2}$ 更新 $p^{k+1} = -r^{k+1} + \beta_k p^k$
11: $k \leftarrow k + 1$.
12: **ENDLOOP**

精确线搜索

$r = g + Bs$

共轭梯度算法

# 信赖域算法

```
import numpy as np
def steihaug_cg(g, B, delta, eps=1e-6):
    n = len(g)
    s = np.zeros(n)
    r = g.copy()
    p = -g.copy()
    k = 0
    if np.linalg.norm(p) <= eps:
        return s
    while True:
        pTBp = p @ B @ p
        if pTBp <= 1e-12:
            a = np.linalg.norm(p)**2
            b = 2 * (s @ p)
            c = np.linalg.norm(s)**2 - delta**2
            discriminant = b**2 - 4*a*c
            tau = (-b + np.sqrt(discriminant)) / (2*a)
            return s + tau * p
        r_norm_sq = np.linalg.norm(r)**2
        alpha = r_norm_sq / pTBp
        s_new = s + alpha * p
        if np.linalg.norm(s_new) >= delta:
            a = np.linalg.norm(p)**2
            b = 2 * (s @ p)
            c = np.linalg.norm(s)**2 - delta**2
            discriminant = b**2 - 4*a*c
            tau = (-b + np.sqrt(discriminant)) / (2*a)
            return s + tau * p
        r_new = r + alpha * B @ p
        if np.linalg.norm(r_new) < eps * np.linalg.norm(g):
            return s_new
        beta = np.linalg.norm(r_new)**2 / r_norm_sq
        p = -r_new + beta * p
        s, r = s_new, r_new
        k += 1
if __name__ == "__main__":
    g = np.array([-4, -3])
    B = np.array([[2, 0], [0, 1]])
    delta = 2.0
    s = steihaug_cg(g, B, delta)
    print(f"最优搜索方向 s: {s}")
    print(f"||s||: {np.linalg.norm(s)}")
```

```
最优搜索方向 s: [1.6 1.2]
||s||: 2.0
```

# 信赖域算法

1. 在算法终止前 $q(s^j)$ 是严格单调递减的，且 $\|s^j\|$ 是严格单调递增的；

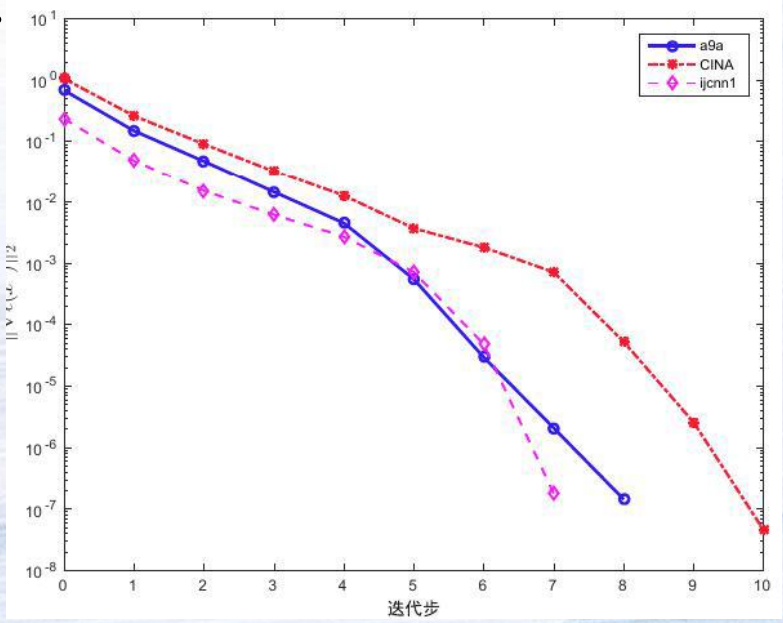
2. 若无条件接受信赖域子问题的更新，则信赖域算法仅有子序列的收敛性，迭代点序列本身不一定收敛．根据下面的定理则说明选取η > 0 可以改善收敛性结果；

3. 信赖域算法产生的迭代序列$\{x^k\}$ 具有Q- 超线性收敛速度．

$$\min_x \ell(x) = \frac{1}{m}\sum_{i=1}^{m} \ln\left(1+\exp\left(-b_i a_i^T x\right)\right) + \lambda \|x\|_2^2$$

$\lambda = 1/(100m)$

*LIBSVM 上的数据集*



# 最小二乘算法

- 定义：一种数学优化技术，它通过最小化误差的平方和寻找数据的最佳函数匹配。在误差估计、不确定度、系统辨识及预测、预报等数据处理诸多学科领域得到了广泛应用。
- 提出与发扬：由法国科学家勒让德在19世纪发现的[1]。1829年，高斯提供了最小二乘法的优化效果强于其他方法的证明[2]。
- 数学原理：设$\{(x^i, y^i)\}_{i=1}^N$是观测数据，其中$x^i = (x_1^i, x_2^i, \ldots, x_n^i)^T$是特征向量，$y^i$是对应的观测值。假设它们之间存在如下函数关系：

$$y^i = f(x^i) + \epsilon^i,$$

其中$\epsilon^i$是对应的数据误差。


1. Von Lindenau, B. A. (1806). OJber den Gebrauch der Gradmessungen zur Bestimmung der Gestalt der Erde. Monatliche Correspondenz zur Befdrderung der Erd- und Himmels-Kunde, 14, 113-158.
2. Gauss C. (1829). Uber ein neues allgemeines Grundgesetz der Mechanik. Journal fur Reine und Angewandte Mathematik, 232-235.

# 最小二乘算法

## 模型推导

$$y^i = f(x^i) + \epsilon^i$$

- 假设$\epsilon^i$是高斯白噪声，即服从均值为0的正态分布。应用极大似然估计可得：

$$Argmax_f \ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left(f(x^i)-y^i\right)^2}{2\sigma^2}\right)\right)$$

$$= Argmax_f -\ln\sqrt{2\pi}\sigma - \sum_{i=1}^{N} \frac{\left(f(x^i)-y^i\right)^2}{2\sigma^2}$$

$$= Argmin_f \sum_{i=1}^{N} \left(f(x^i)-y^i\right)^2$$

- 最小二乘算法的目标函数可理解为

$$\sum (\text{理论值} - \text{观测值})^2$$

# 最小二乘算法

## 一元线性回归

- 设$n=1$，且$f(x^i)=\theta_1 x^i+\theta_0$，则对应的最小二乘算法模型为

$$\min_{\theta_0,\theta_1} L(\theta_0,\theta_1):=\frac{1}{2N}\sum_{i=1}^{N}\left(\theta_1 x^i+\theta_0-y^i\right)^2$$

- 令

$$X=\begin{bmatrix}1 & x^1\\ \vdots & \vdots\\ 1 & x^N\end{bmatrix},y=\begin{bmatrix}y^1\\ \vdots\\ y^N\end{bmatrix},\theta=\begin{bmatrix}\theta_0\\ \theta_1\end{bmatrix}$$

- 最小二乘算法模型的矩阵形式为

$$\min_{\theta}\frac{1}{2}||X\theta-y||^2=\frac{1}{2}\theta^T X^T X\theta-X^T y\theta+\frac{1}{2}||y||^2$$

# 最小二乘算法

## 一元线性回归

$$\min_{\theta_0,\theta_1} L(\theta_0,\theta_1) := \frac{1}{2N}\sum_{i=1}^{N}\left(\theta_1 x^i + \theta_0 - y^i\right)^2$$

对变量$\theta_0,\theta_1$求偏导：

$$\begin{cases} \frac{1}{N}\sum_{i=1}^{N}\left(\theta_1 x^i + \theta_0 - y^i\right) = 0, \\ \frac{1}{N}\sum_{i=1}^{N}\left(\theta_1 x^i + \theta_0 - y^i\right)x^i = 0. \end{cases}$$

$$\begin{cases} \theta_0 = \bar{y} - \theta_1 \bar{x} \\ \theta_1 = \dfrac{\sum_{i=1}^{N}\left(x^i - \bar{x}\right)\left(y^i - \bar{y}\right)}{\sum_{i=1}^{N}\left(x^i - \bar{x}\right)^2} \end{cases}$$

$$\min_{\theta} \frac{1}{2}||X\theta - y||^2 = \frac{1}{2}\theta^T X^T X\theta - X^T y\theta + \frac{1}{2}||y||^2$$

- 对$\theta$进行求导

$$X^T(X\theta - y) = 0.$$

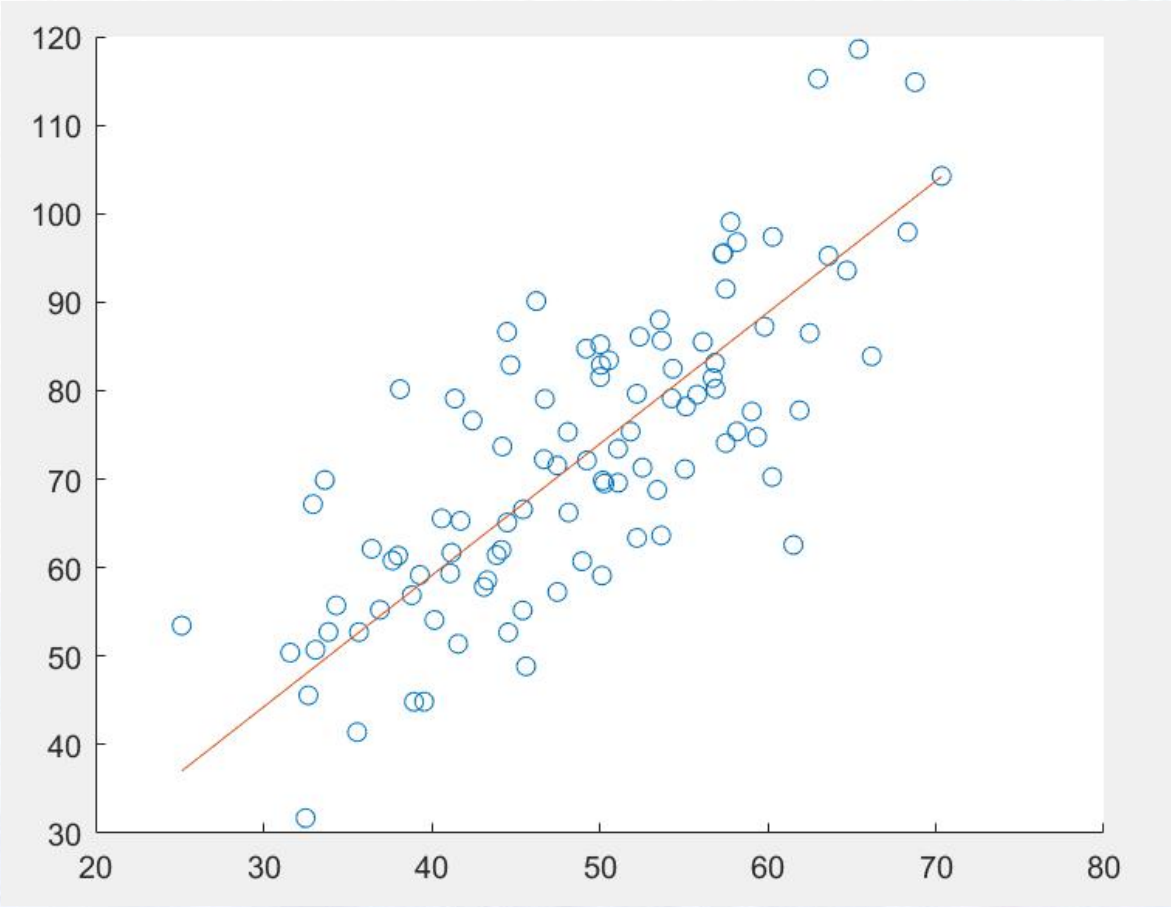
- 故

$$\theta = (X^T X)^{-1} X^T y.$$

# 最小二乘算法

## 一元线性回归案例

- 导入数据集： Lregression.txt
- data = load('Lregression.txt');
- x = data(:,1);
- y = data(:,2);
- scatter(x,y)
- hold on
- H=[1,sum(x);sum(x),sum(x.^2)];
- f=-[sum(y);sum(x.*y)];
- b = quadprog(H,f);
- z = b(2)*x + b(1);
- plot(x,z)



## 模型检验

- 假设$\hat{y}^i = \theta_1 x^i + \theta_0$，定义总离差平方和：

$$SST = \sum_{i=1}^{N} \left(y^i - \bar{y}\right)^2$$

- 回归平方和：

$$SSR = \sum_{i=1}^{N} \left(\hat{y}^i - \bar{y}\right)^2$$

- 残差平方和：

$$SSE = \sum_{i=1}^{N} \left(\hat{y}^i - y^i\right)^2$$

## 模型检验

- 总离差平方和分解式：$SST = SSR + SSE$，推导如下：

$$SST = \sum_{i=1}^{N} \left(y^i - \hat{y}^i + \hat{y}^i - \bar{y}\right)^2$$

$$= SSR + SSE + 2\sum_{i=1}^{N} (y^i - \hat{y}^i)(\hat{y}^i - \bar{y})$$

$$= SSR + SSE + 2\sum_{i=1}^{N} (y^i - \theta_1 x^i - \theta_0)\theta_1 (x^i - \bar{x})$$

- 根据最小二乘模型求导可知最后一项为0.

| 方差来源 | 偏差平方和 | 自由度 | 均方 | $F$ | $p$值 |
|---|---|---|---|---|---|
| 回归 | $SSR$ | $p$ | $MSR = \frac{SSR}{p}$ | $\frac{SSR/p}{SSE/(n-p-1)}$ | $P(F_{\alpha,(p,n-p-1)}) > F$ |
| 残差 | $SSE$ | $n-p-1$ | $MSE = \frac{SSE}{n-p-1}$ | – | – |
| 总变异 | $SST$ | $n-1$ | – | – | – |

## 模型检验

- 要判断一个模型好不好，仅仅看显著性是不够的。还需要看看总的回归变异（回归平方和）是否比总的误差变异（残差平方和）大很多。通常用拟合优度（Goodness of Fit），指回归直线对观测值的拟合程度

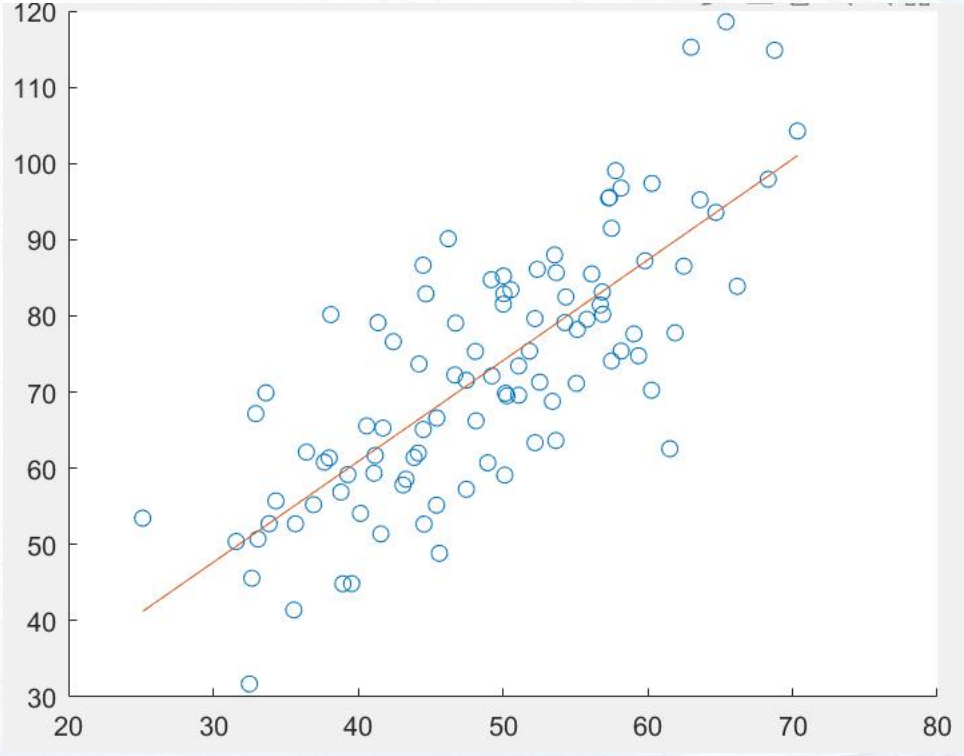
$$R^2 = \frac{SSR}{SST} = \frac{SSR}{SSR + SSE} = 1 - \frac{SSE}{SSR + SSE}$$
$$= \frac{\sum_{i=1}^{N}(\hat{y}^i - \bar{y})^2}{\sum_{i=1}^{N}(y^i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{N}(y^i - \hat{y}^i)^2}{\sum_{i=1}^{N}(y^i - \bar{y})^2}$$

- R的平方越接近1，说明回归方程对于样本数据点的拟合优度越高；反之，R的平方越接近0，说明回归方程对于样本数据点的拟合优度越低。

# 最小二乘算法

## 一元线性回归案例

- 导入数据集：Lregression.txt
- data = load('Lregression.txt');
- x = data(:,1);
- y = data(:,2);
- scatter(x,y)
- hold on
- 进行一元线性回归
- X=[ones(size(x)) x];
- [b,bint,r,rint,stats]=regress(y,X)或b = lsqlin(X,y)%b是系数，r是残差，stats是拟合优度、F统计量、出错概率、剩余标准差
- rcoplot(r,rint)%残差图，残差置信区间
- z = b(2)*x + b(1);
- plot(x,z)



# 最小二乘算法

## 多元线性回归

- 设$f(x^i)=\theta_1 x_1^i+\theta_2 x_2^i+\cdots+\theta_n x_n^i+\theta_0$，则对应最小二乘算法模型为

$$\min_{\theta} L(\theta):=\frac{1}{2N}\sum_{i=1}^{N}\left(\theta_1 x_1^i+\theta_2 x_2^i+\cdots+\theta_n x_n^i+\theta_0-y^i\right)^2$$

- 令

$$X=\begin{bmatrix}1 & x_1^1\cdots & x_n^1\\ \vdots & \vdots\ \ddots & \vdots\\ 1 & x_1^N\cdots & x_n^N\end{bmatrix},y=\begin{bmatrix}y^1\\ \vdots\\ y^N\end{bmatrix},\theta=\begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix}$$

- 最小二乘算法模型的矩阵形式为

$$\min_{\theta}\frac{1}{2}||X\theta-y||^2$$

## 多元线性回归

- 对$\theta$进行求导

$$X^T(X\theta - y) = 0.$$

- 故

$$\theta = (X^T X)^{-1} X^T y.$$

- 模型检验：拟合优度

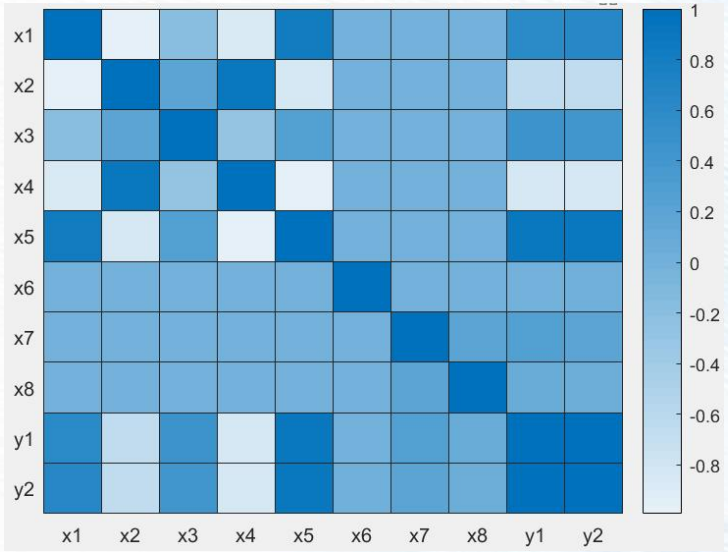
$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2}{\sum_{i=1}^{N}\left(y^i - \bar{y}\right)^2}$$

- F检验

$$F = \frac{SSR/p}{SSE/(n-p-1)} = \frac{R^2/p}{(1-R^2)/(n-p-1)}$$

# 最小二乘算法

## 多元线性回归案例




- 导入数据： MLregression.xlsx
- data = xlsread('MLregression.xlsx');
- 相关性分析+热力图
- rho = corr(data, 'type','pearson');
- string_name={'x1','x2','x3','x4','x5','x6','x7','x8','y1','y2'};
- xvalues = string_name;
- yvalues = string_name;
- h = heatmap(xvalues,yvalues, rho);
- 取绝对值后，0-0.09为没有相关性，0.3-弱，0.1-0.3为弱相关，0.3-0.5为中等相关，0.5-1.0为强相关。选取x1,x2,x3,x4,x5,x7

## 多元线性回归案例

- 多元线性回归
- x=[ones(size(data(:,1))),data(:,1),data(:,2),data(:,3),data(:,4),data(:,5),data(:,7)];
- y=data(:,9);
- [b,bint,r,rint,stats]=regress(y,x)或b = lsqlin(x,y)

- b =[84.3865, -64.7734, -0.0873, 0.0608, 0, 4.1700, 20.4380]

$$y1 = 84.3865 + -64.7734 * x1 + -0.0873 * x2 + 0.0608 * x3 + 4.1700 * x5 + 20.4380 * x7$$

作业：试利用多元线性回归进行分析该案例中的$y2$。

## 非线性最小二乘

- 引入残差函数

$$r_i(\theta) = f(x^i) - y^i$$

- 非线性最小二乘算法模型可写为

$$\min_{\theta} \frac{1}{2} \sum_{i=1}^{N} (r_i(\theta))^2$$

- 令$r(\theta) = (r_i(\theta))$，则可写出向量形式

$$\min_{\theta} f(x) := \frac{1}{2} ||r(\theta)||^2 \qquad (1)$$

- 根据残差分为小残差非线性最小二乘和大残差非线性最小二乘。

## 非线性最小二乘

- 问题(1)是一无约束优化问题，但一般非凸。可以直接使用线搜索或信赖域法求解。残差函数$r(x)$在点$x$处的雅可比矩阵为

$$J(x) = \begin{bmatrix} \nabla r_1(x)^T \\ \vdots \\ \nabla r_N(x)^T \end{bmatrix}$$

- 目标函数$f(x)$的梯度和海瑟矩阵为

$$\nabla f(x) = J(x)^T r(x)$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{N} r_i(x) \nabla^2 r_i(x)$$

- $\nabla^2 f(x)$中，第一项容易得到，第二项需要额外计算。残量值较小或残量函数接近线性函数，第二项可以忽略。如果第二项不可忽略，则引入带结构的拟牛顿方法。

## 非线性最小二乘

- 利用$\nabla^2 f(x) \approx J(x)^T J(x)$，**高斯-牛顿法**的迭代方向满足

$$J(x)^T J(x) d^{GN} = -J(x)^T r(x)$$

- 即$d^{GN}$是如下优化问题的最优解。

$$\min_d \frac{1}{2} ||J(x)d + r(x)||^2$$

**Algorithm 1** 高斯– 牛顿法

1: 给定初始值$x_0$，$k \leftarrow 0$.
2: **while** 未达到收敛准则 **do**
3: 计算残差向量$r_k$，雅可比矩阵$J_k$.
4: 求解线性最小二乘问题 $\min_d \frac{1}{2}\|J_k d + r_k\|^2$ 确定下降方向$d_k$.
5: 使用线搜索准则计算步长$\alpha_k$.
6: 更新：$x_{k+1} = x_k + \alpha_k d_k$.
7: $k \leftarrow k + 1$.
8: **end while**

## 非线性最小二乘

- **Levenberg-Marquardt (LM) 方法**
- 考虑如下信赖域子问题

$$\min_d \frac{1}{2}||J(x)d + r(x)||^2 \ \ s.t. \ ||d|| \leq \Delta(x). \tag{2}$$

- 标记

$$m_k(d) = \frac{1}{2}||r^k||^2 + d^T(J^k)^T r^k + \frac{1}{2}d^T(J^k)^T J^k d$$

- 函数值实际下降量与预估下降量的比值衡量$m_k(d)$近似程度的好坏

$$\rho_k = \frac{f(x^k) - f(x^k + d^k)}{m_k(0) - m_k(d^k)}$$

# 最小二乘算法

## 非线性最小二乘

- 由于问题(2)是凸规划，它等价于

- $(J(x)^T J(x) + \lambda I)d = -J(x)^T r(x), \quad (3)$
- $0 \le \lambda \perp \Delta(x) - ||d|| \ge 0.$

- 系统(3)等价于如下线性最小二乘问题

- $\min_d \frac{1}{2} \left\| \begin{bmatrix} J(x) \\ \sqrt{\lambda} I \end{bmatrix} d + \begin{bmatrix} r(x) \\ 0 \end{bmatrix} \right\|^2$

**Algorithm 3** LMF 方法

1: 给定初始点$x_0$，初始乘子$\lambda_0$，$k \leftarrow 0$.
2: 给定参数$0 \le \eta < \bar{\rho}_1 < \bar{\rho}_2 < 1$，$\gamma_1 < 1 < \gamma_2$.
3: **while** 未达到收敛准则 **do**
4: 求解LM 方程$((J_k)^{\mathrm{T}} J_k + \lambda I)d = -(J_k)^{\mathrm{T}} r_k$得到迭代方向$d_k$.
5: 根据(14)式计算下降率$\rho_k$.
6: 更新乘子：

$$\lambda_{k+1} = \begin{cases} \gamma_2 \lambda_k, & \rho_k < \bar{\rho}_1, \quad /\star \text{ 扩大乘子（缩小信赖域半径）} \star/ \\ \gamma_1 \lambda_k, & \rho_k > \bar{\rho}_2, \quad /\star \text{ 缩小乘子（扩大信赖域半径）} \star/ \\ \lambda_k, & \text{其他}. \quad /\star \text{ 乘子不变} \star/ \end{cases}$$

7: 更新自变量：

$$x_{k+1} = \begin{cases} x_k + d_k, & \rho_k > \eta, \quad /\star \text{ 只有下降比例足够大才更新} \star/ \\ x_k, & \text{其他}. \end{cases}$$

8: $k \leftarrow k + 1$.
9: **end while**

## 非线性最小二乘

- 大残量问题的拟牛顿算法

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{N} r_i(x) \nabla^2 r_i(x)$$

- 近似T

$$B_k = J_k^T J_k + T_k$$

- 令$s_k = x_{k+1} - x_k$，则

$$T_{k+1} s_k \approx \sum_{i=1}^{N} r_i(x_{k+1}) \nabla^2 r_i(x_{k+1}) s_k \approx \sum_{i=1}^{N} r_i(x_{k+1}) (\nabla r_i(x_{k+1}) - \nabla r_i(x_k))$$

$$= J_{k+1}^T r_{k+1} - J_k^T r_{k+1}$$

## 非线性最小二乘

- 大残量问题的拟牛顿算法

$$T_{k+1} = T_k + \frac{(y^{\#} - T_k s_k)y^T + y(y^{\#} - T_k s_k)^T}{y^T s_k} - \frac{(y^{\#} - T_k s_k)^T s_k}{(y^T s_k)^2} yy^T$$

其中

$$s_k = x_{k+1} - x_k$$
$$y = J_{k+1}^T r_{k+1} - J_k^T r_k$$
$$y^{\#} = J_{k+1}^T r_{k+1} - J_k^T r_{k+1}$$

## 非线性最小二乘

### ➢ 大残量问题的拟牛顿算法

步1. 取初始点$x_0$，初始矩阵$T_0$及精确参数$\varepsilon \geq 0$，令$k=0$.

步2. 若终止准则满足,算法终止，否则进入下一步.

步3. $x_{k+1}=x_k-\left(J_k^TJ_k+T_k\right)^{-1}J_k{}^Tr_k$.

步4. $$T_{k+1}=T_k+\frac{(y^{\#}-T_ks_k)y^T+y(y^{\#}-T_ks_k)^T}{y^Ts_k}-\frac{(y^{\#}-T_ks_k)^Ts_k}{(y^Ts_k)^2}yy^T$$

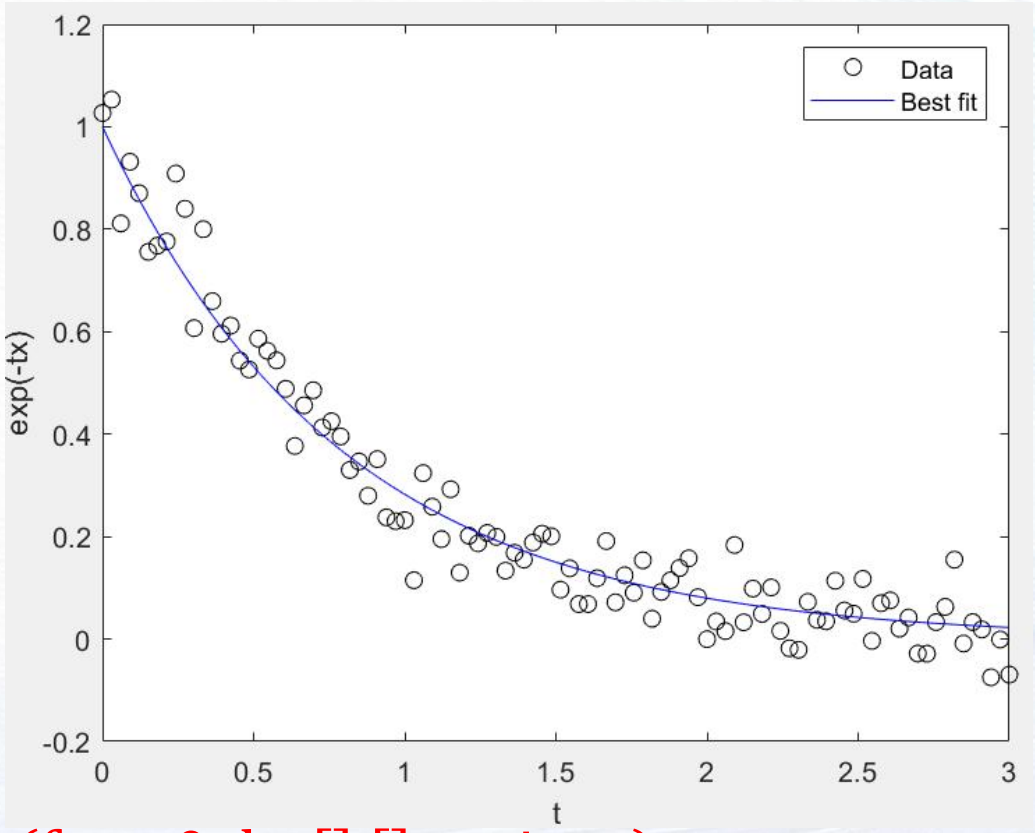
其中$s_k=x_{k+1}-x_k$，$y=J_{k+1}^Tr_{k+1}-J_k^Tr_k$，$y^{\#}=J_{k+1}^Tr_{k+1}-J_k^Tr_{k+1}$.

步5. 令$k:=k+1$，转步2.

# 最小二乘算法

## 非线性最小二乘

- 小残量问题的LM算法代码实现



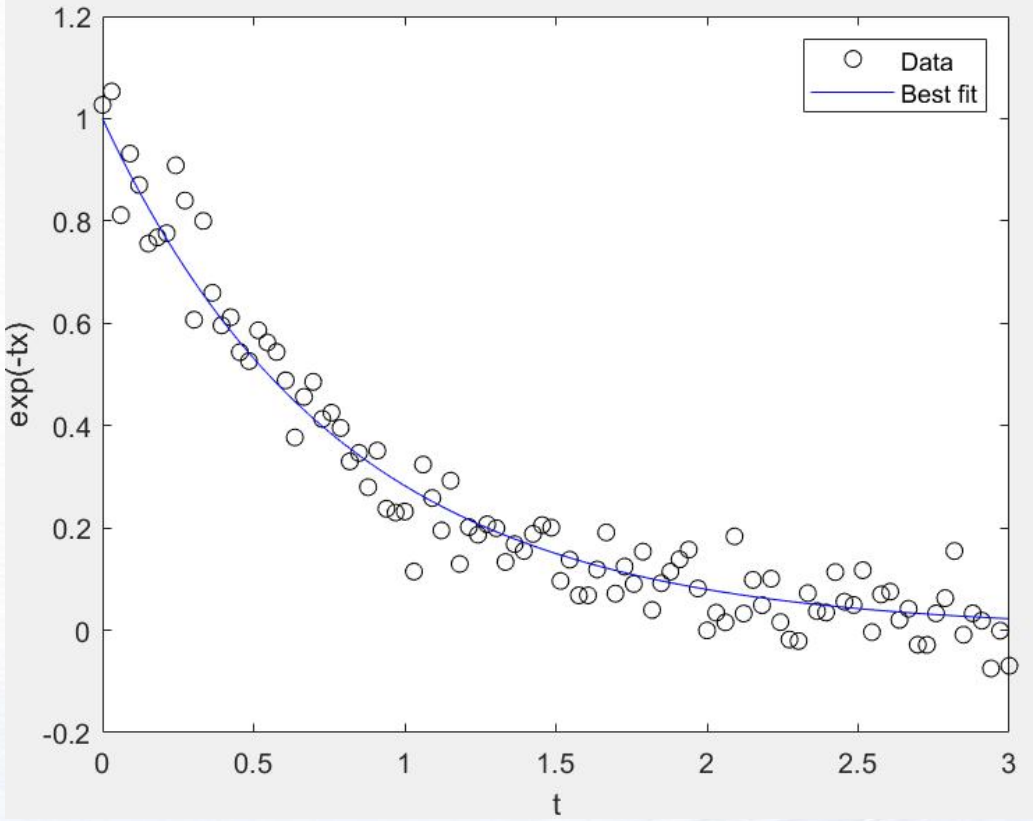

```
rng default
d = linspace(0,3);
y = exp(-1.3*d) + 0.05*randn(size(d));
fun = @(r)exp(-d*r)-y;或fun = @(x,d)exp(-d*x);
x0 = 4;
options.Algorithm = 'levenberg-marquardt';
x = lsqnonlin(fun,x0,[],[],options);或x = lsqcurvefit(fun,x0,d,y,[],[],options)
plot(d,y,'ko',d,exp(-x*d),'b-')
legend('Data','Best fit')
xlabel('t')
ylabel('exp(-tx)')
```

# 最小二乘算法

## 非线性最小二乘

### 小残量问题的高斯牛顿算法代码实现

```
a=1;b=2;c=1;        x=(0:0.01:1)';
w=rand(length(x),1)*2-1;  %生成噪声
y=exp(a*x.^2+b*x+c)+w;   %带噪声的模型
plot(x,y,'.')
pre=rand(3,1);
for i=1:1000
   f = exp(pre(1)*x.^2+pre(2)*x+pre(3));
   g = y-f;                %残差
   p1 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x.^2;  %对a求偏导
   p2 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x;     %对b求偏导
   p3 = exp(pre(1)*x.^2+pre(2)*x+pre(3));        %对c求偏导
   J = [p1 p2 p3];
   delta = inv(J'*J)*J'* g;
   pcur = pre+delta;
   if norm(delta) <1e-16
      break;
   end
   pre = pcur;
end
hold on;
plot(x,exp(a*x.^2+b*x+c),'r');      plot(x,exp(pre(1)*x.^2+pre(2)*x+pre(3)),'g');
```

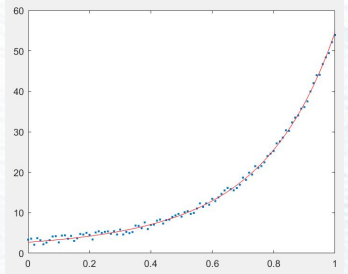


# 最小二乘算法

## 非线性最小二乘

### 大残量问题的拟牛顿算法代码实现

```
a=1;b=2;c=1;        x=(0:0.01:1)';
w=rand(length(x),1)*2-1;  %生成噪声
y=exp(a*x.^2+b*x+c)+w;    %带噪声的模型
plot(x,y,'.')
pre=rand(3,1);
Tk=eye(3);
k=0;
maxk=500;
while(k<maxk)
    f = exp(pre(1)*x.^2+pre(2)*x+pre(3));
    g = y-f;                %残差
    p1 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x.^2;   %对a求偏导
    p2 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x;      %对b求偏导
    p3 = exp(pre(1)*x.^2+pre(2)*x+pre(3));         %对c求偏导
    J = [p1 p2 p3];
    delta = inv(J'*J+Tk)*J'* g;
    pcur = pre+delta;
    if norm(delta) <1e-16
       break;
    end
    pre = pcur;
    f1 = exp(pre(1)*x.^2+pre(2)*x+pre(3));
    g1 = y-f1;                %残差
    p11 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x.^2;   %对a求偏导
    p21 = exp(pre(1)*x.^2+pre(2)*x+pre(3)).*x;      %对b求偏导
    p31 = exp(pre(1)*x.^2+pre(2)*x+pre(3));         %对c求偏导
    J1 = [p11 p21 p31];
    sk=pcur-pre;
    yk=J1'* g1-J'* g;
    ykk=J1'* g1-J'* g1;
    Tk=Tk+((ykk-Tk*sk)*yk'+yk*(ykk-Tk*sk)')/(yk'*sk)-((ykk-Tk*sk)'*sk)/(yk'*sk)*(yk*yk');
    k=k+1;
end
hold on;
plot(x,exp(a*x.^2+b*x+c),'r');
plot(x,exp(pre(1)*x.^2+pre(2)*x+pre(3)),'g');
```



# 最小二乘算法

## 求解线性系统

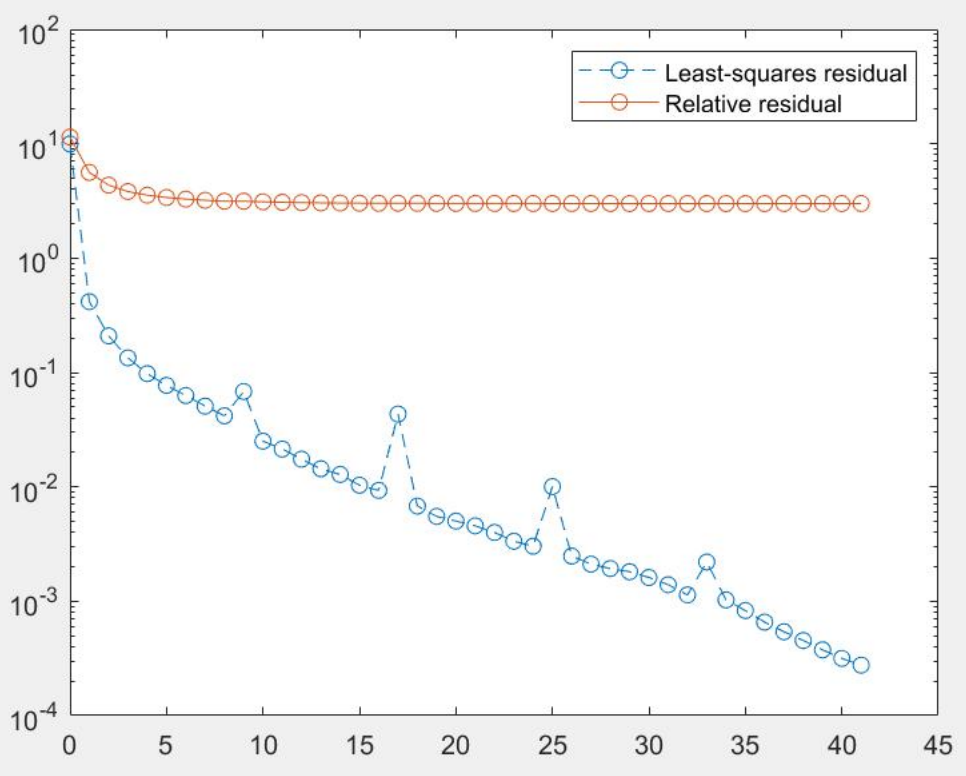
- 线性系统

$$Ax = b$$

- 等价于

$$\min_x \frac{1}{2} \| Ax - b \|^2$$

```
rng default
A = sprand(400,300,.5);
b = rand(400,1);
[x,flag,relres,iter,resvec,lsvec] = lsqr(A,b,1e-4,70);
N = length(resvec);
semilogy(0:N-1,lsvec,'--o',0:N-1,resvec,'-o')
legend("Least-squares residual","Relative residual")
```




**作业：**

1. 随机生成一400*300的矩阵A；
2. 生成300*1的稀疏向量x；
3. b=Ax+高斯白噪声
4. 求解Ax=b

# 最小二乘算法

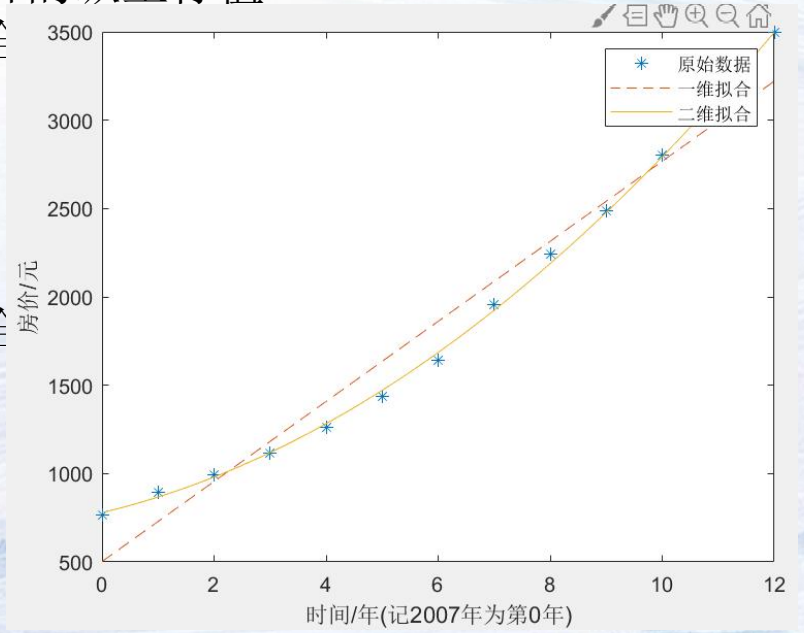
## 数据拟合

| 时间(年) | 平均房价(元/平米) |
|---|---|
| 1997 | 767 |
| 1998 | 895 |
| 1999 | 995 |
| 2000 | 1117 |
| 2001 | 1261 |
| 2002 | 1437 |
| 2003 | 1640 |
| 2004 | 1957 |
| 2005 | 2244 |
| 2006 | 2489 |
| 2007 | 2801 |
| 2008 | 3096 |
| 2009 | 3500 |

```
x = 0:2009 - 1997;
y = [767 895 995 1117 1261 1437 1640 1957 2244 2489 2801 3096
3500];
fy1 = polyfit(x,y,1);    % 线性拟合
fy2 = polyfit(x,y,2);    % 二次拟合

t = 0:0.1:2009 - 1997
y1 = zeros( size(t) );   % 初始化
y2 = zeros( size(t) );   % 初始化
y1 = polyval(fy1,t);     % 得到线性拟合的纵坐标值
y2 = polyval(fy2,t);     % 得到二次拟合

plot(x,y,'*',t,y1,'--',t,y2,'-');
xlabel('时间/年(记2007年为第0年)');
ylabel('房价/元');
legend('原始数据','一维拟合','二维拟合
```

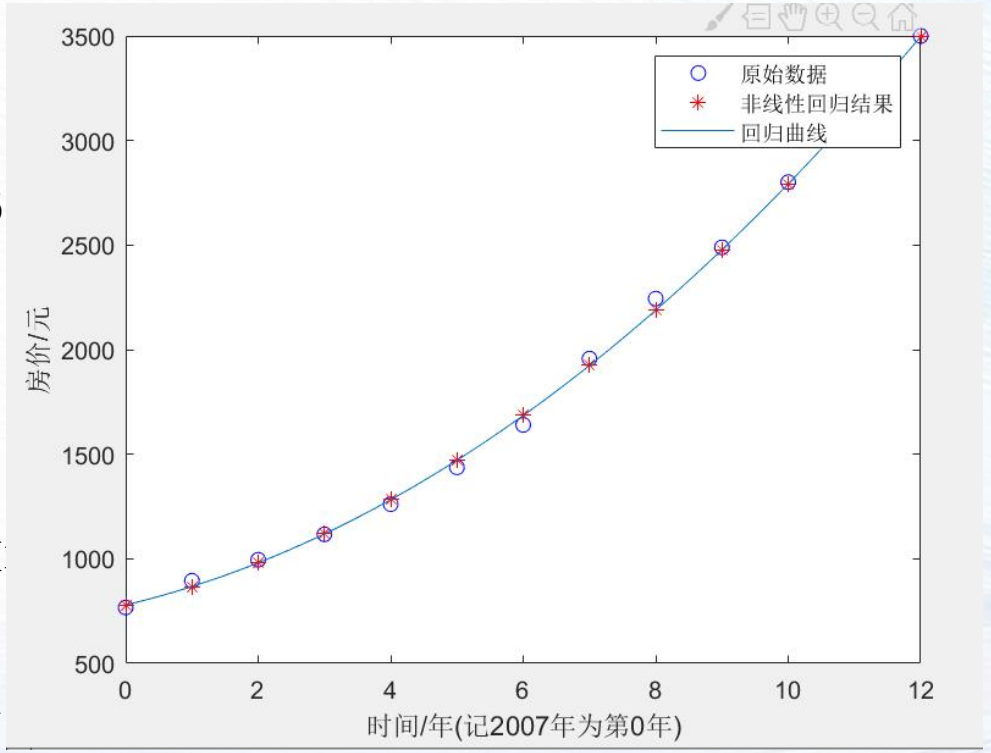


# 最小二乘算法

## 数据拟合

```
x = 0:2009 - 1997;
y = [767 895 995 1117 1261 1437 1640 1957 2244 2489 2801 3096
x_new = zeros(length(x),3);

x_new(:,1) = 1;
x_new(:,2) = x';
x_new(:,3) = (x.^2)';
[b,bint,r,rint,stats] = regress(y',x_new)        % 调用回归函数
plot(x,y,'bo');
hold on;
y2 = b(3,1) * x.^2 + b(2,1) * x + b(1,1);        % 非线性回归结
plot(x,y2,'r*');
t = 0:0.01:12;
xian_y = b(3,1) * t.^2 + b(2,1) * t + b(1,1);    % 回归曲线
plot(t,xian_y,'-');
legend('原始数据','非线性回归结果','回归曲线');
xlabel('时间/年(记2007年为第0年)');
ylabel('房价/元');
```




**作业：**

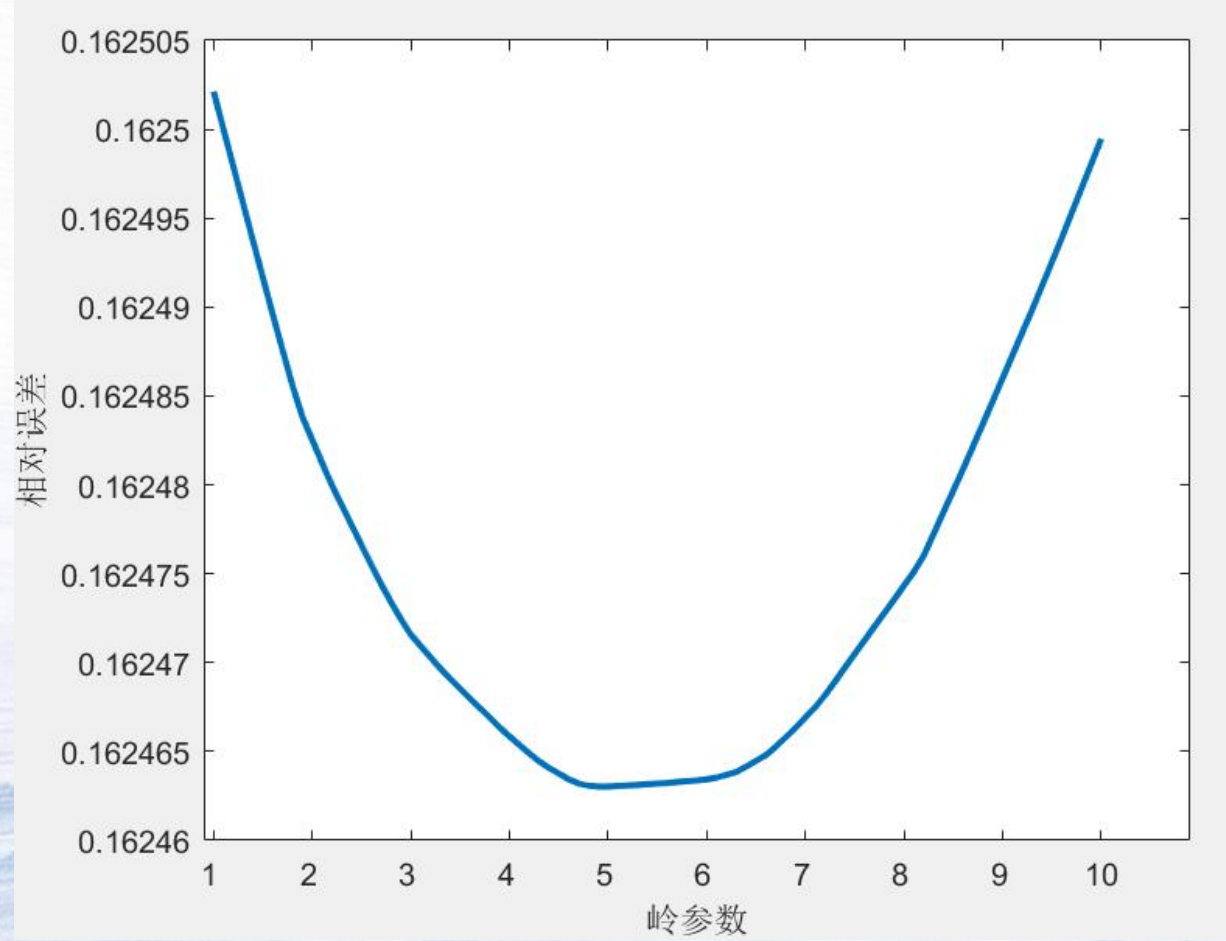
1. 利用3次多项式拟合

# 最小二乘算法

## 岭回归 - Ridge Regression

$$\min_{\theta} \frac{1}{2}||X\theta - y||^2 + \lambda||\theta||^2$$

```
load('data.txt')
n=size(data,2);
x=data(:,1:n-1); %自变量
y=data(:,n);          %因变量
xm=mean(x);     %x的平均值
xs=std(x);            %x的方差
ym=mean(y); %y的平均值
 x_o=zscore(x);
k1=[1:0.1:10];
for k=1:length(k1)
lamda=k1(k);
xishu=(x_o'*x_o+lamda*eye(size(x_o,2)))^(-1)*x_o'*(y-ym);
xishu1=[ym-sum(xishu.*xm'./xs');xishu./xs'];
y_n1=xishu1(1)+x*xishu1(2:end);
wucha(k)=sum(abs(y_n1-y)./y)/length(y);
end
plot(wucha,'LineWidth',2)
ylabel('相对误差')
xlabel('岭参数')
set(gca,'XTick',1:10:100)
set(gca,'XtickLabel',1:1:10)
```

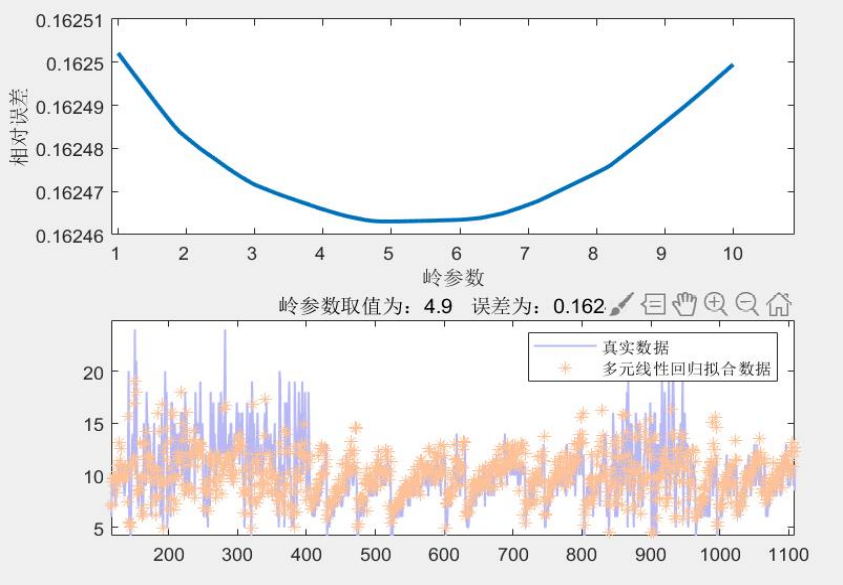


# 最小二乘算法

## 岭回归 - Ridge Regression

```
load('data.txt')
n=size(data,2);
x=data(:,1:n-1);y=data(:,n);
k = 1:0.1:10;
B = ridge(y,x,k,0);
for  k1 = 1:length(k)
    A=B(:,k1);
    yn= A(1)+x*A(2:end);
    wucha(k1)=sum(abs(y-
yn)./y)/length(y);
end
figure(1)
subplot(2,1,1)
plot(1:length(k),wucha,'LineWidth',2)
ylabel('相对误差')
xlabel('岭参数')
set(gca,'XTick',1:10:100)
set(gca,'XtickLabel',1:1:10)
```

```
subplot(2,1,2)
index=find(wucha==min(wucha));
xishu = ridge(y,x,k(index),0);
y_p= A(1)+x*A(2:end);
color1=[184  184  246]/255;
color2=[255  193  151]/255;
titlestr=['岭参数取值为：',num2str(k(index)),'
误差为：',num2str(min(wucha))];
plot(y(3000:end),'Color',color1,'LineWidth',1)
hold on
plot(y_p(3000:end),'*','Color',color2)
hold on
legend('真实数据','多元线性回归拟合数据')
title(titlestr)
```

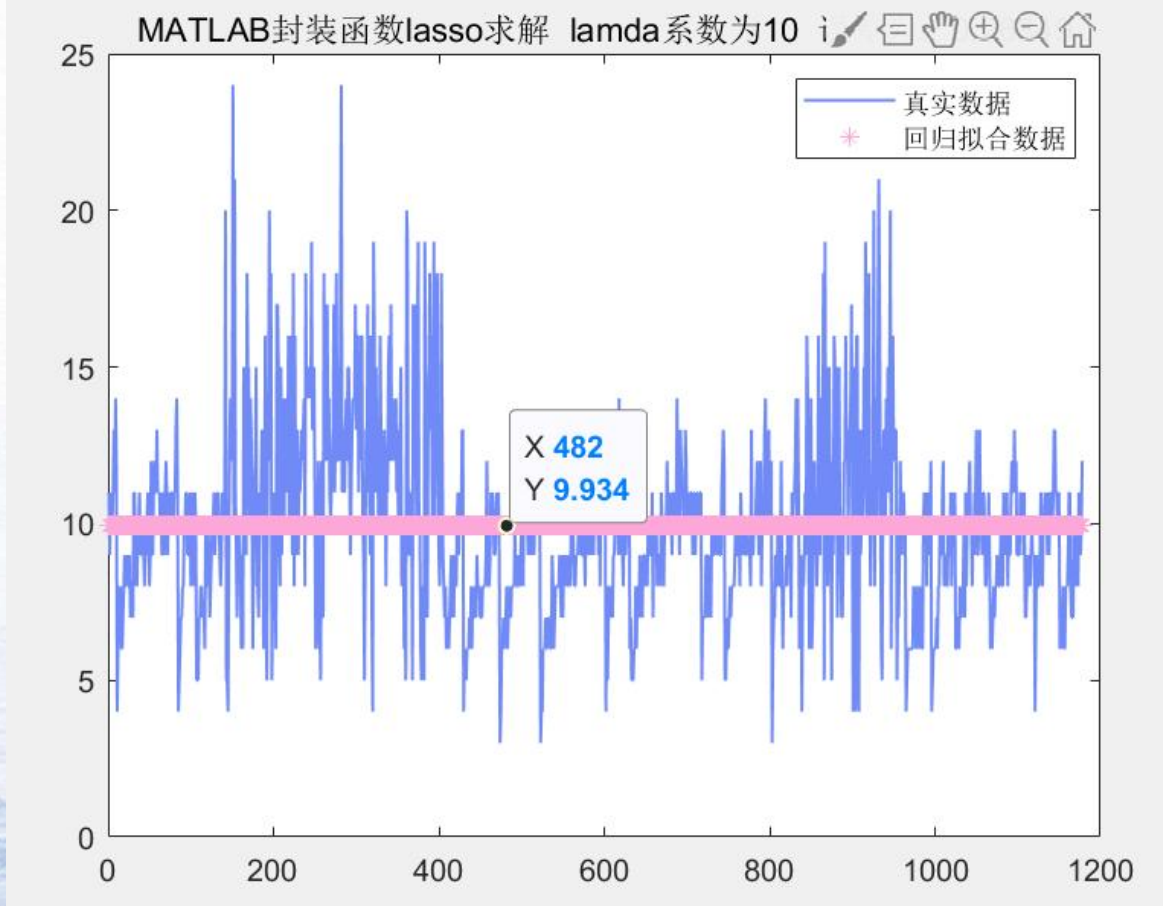


# 最小二乘算法

## Lasso回归

$$\min_{\theta} \frac{1}{2}||X\theta - y||^2 \qquad \min_{\theta} \frac{1}{2}||X\theta - y||^2 + \lambda||\theta||_1$$

```
load('data.txt')
n=size(data,2);
x=data(:,1:n-1);y=data(:,n);
x1=[ones(size(x,1),1),x];
[B,FitInfo] = lasso(x,y, 'Lambda',lamda);
y_p1=x*B+FitInfo.Intercept;
wucha1=sum(abs(y_p1-y)./y)/length(y);
figure(3)
color1=[112  138  248]/255;
color2=[254  168  217]/255;
titlestr=['MATLAB封装函数lasso求解  lamda系数为
',num2str(min(lamda)),'  误差为：',num2str(wucha1)];
plot(y(3000:length(y)),'Color',color1,'LineWidth',1)
hold on
plot(y_p1(3000:length(y)),'*','Color',color2)
hold on
legend('真实数据','回归拟合数据')
title(titlestr)
```

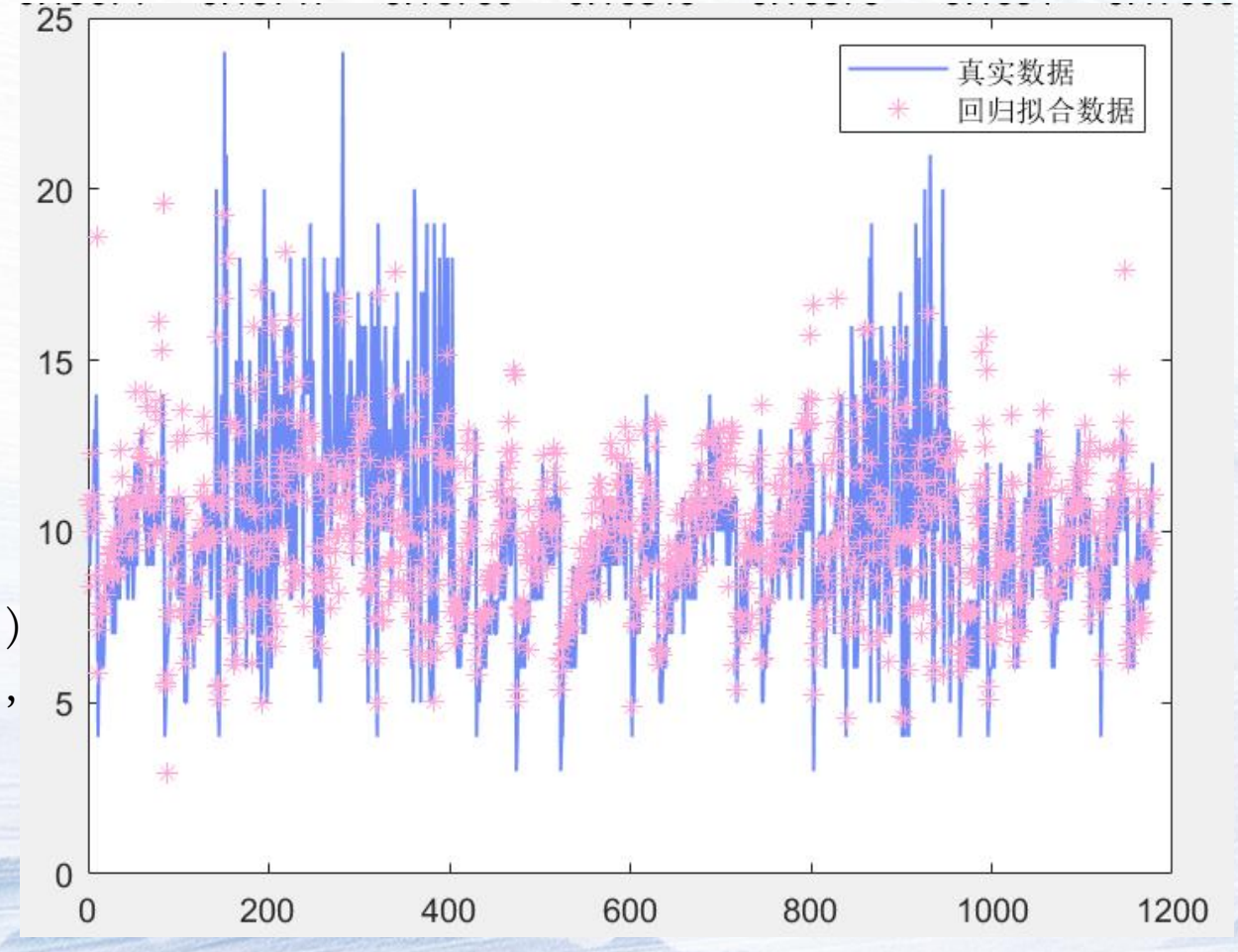


# 最小二乘算法

## 弹性网络回归:Elastic Net

$$\min_{\theta} \frac{1}{2}||X\theta - y||^2 + \lambda||\theta||_1 + \frac{1-\lambda}{2}||\theta||^2$$

```
load('data.txt')
n=size(data,2);
x=data(:,1:n-1);y=data(:,n);
x1=[ones(size(x,1),1),x];
[B,FitInfo] = lasso(x,y,'CV',10,'Alpha',0.2);
y_p1=x*B+FitInfo.Intercept;
wucha1=sum(abs(y_p1-y)./y)/length(y);
figure(3)
color1=[112  138  248]/255;
color2=[254  168  217]/255;
titlestr=['MATLAB封装函数lasso求解  lamda系数为
',num2str(min(lamda)),'  误差为：',num2str(wucha1)
plot(y(3000:length(y)),'Color',color1,'LineWidth',
hold on
plot(y_p1(3000:length(y)),'*','Color',color2)
hold on
legend('真实数据','回归拟合数据')
title(titlestr)
```



# 最小二乘算法

## 图像去噪

```
I=imread('lena.tif');
PSF=fspecial('gaussian',10,4);
Blurred=imfilter(I,PSF,'conv');
BN=imnoise(Blurred,'gaussian',0,0.01);
NP=0.01*prod(size(I));
[reg LAGRA]=deconvreg(BN,PSF,NP);
edged=edgetaper(BN,PSF);
reg2=deconvreg(edged,PSF,[],LAGRA);
subplot(2,2,1)
imshow(I);
title('原始图像');
subplot(2,2,2)
imshow(BN);
title('加入高斯噪声的图像');
subplot(2,2,3)
imshow(reg);
title('恢复后图像');
subplot(2,2,4)
imshow(reg2);
title('拉格朗日恢复图像');
```

南昌大学
NANCHANG UNIVERSITY

课件资源：https://zhenhuapeng.github.io//coursematerials/

# 感　谢　观　看

## 大数据优化：理论、算法及其应用

——来源于《最优化计算方法》（高教社）


★★★ 主讲人：彭振华 ★★★

联系方式：zhenhuapeng@ncu.edu.cn
zhenhuapeng@whu.edu.cn
15870605317（微信同号）

研究兴趣：1. 非凸非光滑优化算法与理论
2. 智能决策
3. 智能计算与机器学习

数学与计算机学院