

数学优化

数学优化

- (Mathematical) Optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & g_i(x) = 0, \quad i = 1, \dots, p \end{array}$$

- ▶ $x \in \mathbf{R}^n$ is (vector) variable to be chosen
- ▶ f_0 is the *objective function*, to be minimized
- ▶ f_1, \dots, f_m are the *inequality constraint functions*
- ▶ g_1, \dots, g_p are the *equality constraint functions*

最大化问题可以转化为最小化问题

几乎任何问题都可以建模成数学优化问题来求解

数学优化

- 如何把问题建模成最优化问题?
 - 确定优化变量
 - 制定目标函数
 - 写出限制条件

Example

某军工厂生产甲、乙、丙三种产品，生产三种产品需要A、B两种资源，其单位需求量及利润由下表1给出，问每天生产甲、乙、丙三种产品各多少，可使总利润最大？

	甲	乙	丙	资源的最大量
A	2	3	1	100kg
B	3	3	2	120kg
利润	40元	45元	24元	

$$\max_{x_1, x_2, x_3} 40x_1 + 45x_2 + 24x_3$$

$$s.t. \begin{cases} 2x_1 + 3x_2 + x_3 \leq 100 \\ 3x_1 + 3x_2 + 2x_3 \leq 120 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases}$$

线性回归问题

- 线性模型:

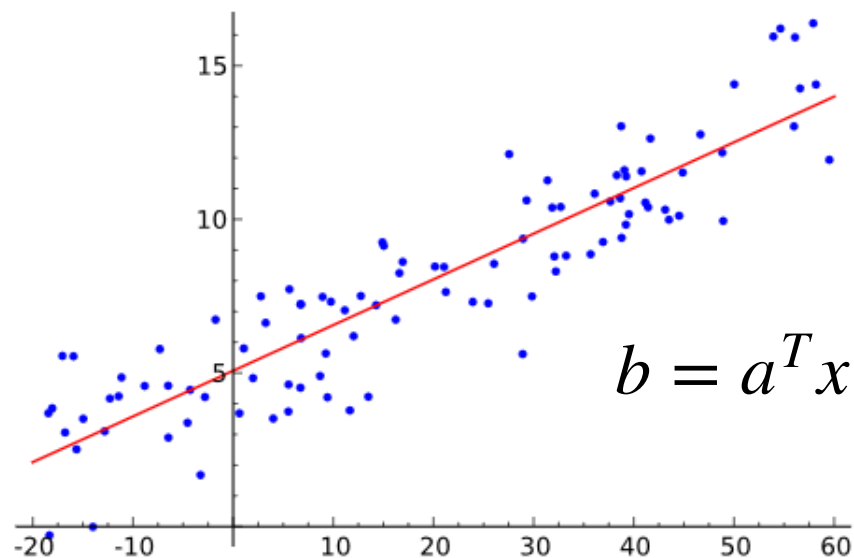
$$b = a^T x$$

输入 a , 输出 b , 系数是 x

- 例如:
 - 学习时间和考试成绩
 - 疫情死亡人数与确诊人数

线性回归问题

- 给定了一系列观测值 (a_i, b_i) , 如何确定 x ?



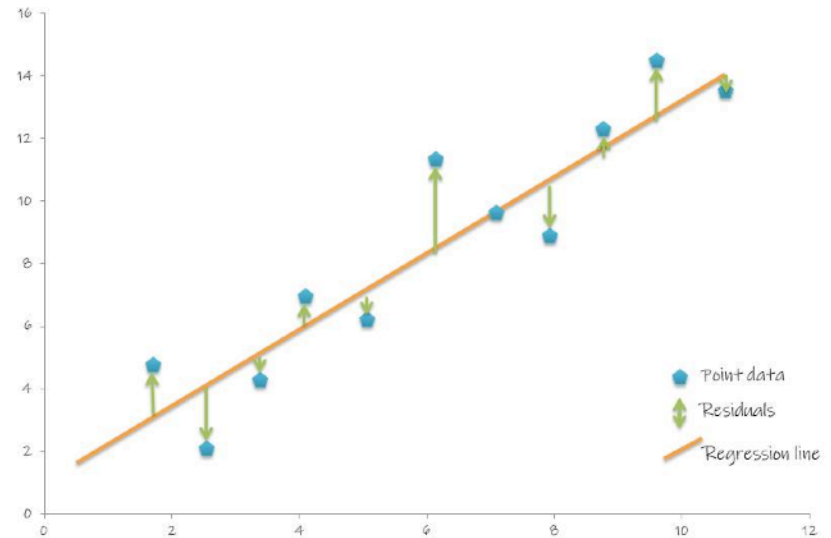
- 从数据中估计模型参数一般称为模型拟合 (model fitting) 或者回归 (regression)

最小二乘法

- 最小二乘法：最小化**均方误差**（Mean Square Error, 简称MSE）

$$\hat{x} = \arg \min_x \sum_i (b_i - a_i^T x)^2$$

- 残差： $r_i = b_i - a_i^T x$
- 残差向量： $R = [r_1, \dots, r_n]^T$



L2范数

$$\text{L2范数: } \|x\|_2 = \sqrt{\sum_i x_i^2}$$

- 向量x的欧式距离

MSE 等于残差向量 $R = [r_1, \dots, r_n]^T$ 的L2范数的平方

$$MSE = \|R\|_2^2$$

最小二乘的统计解释

- 假设数据带有高斯噪音

$$b_i = a_i^T x + n, n \sim G(0, \sigma)$$

- 给定 x 观察到 (a_i, b_i) 的可能性:

$$P[(a_i, b_i)|x] = P[b_i - a_i^T x] \propto \exp - \frac{(b_i - a_i^T x)^2}{2\sigma^2}$$

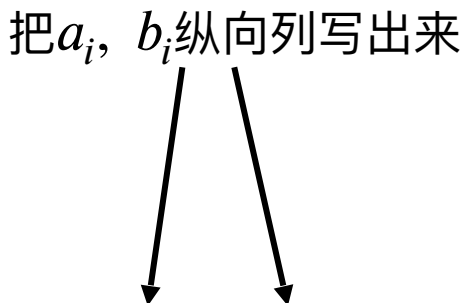
- P通常被称为: **似然函数 / 似然性 (likelihood)**

最小二乘的统计解释

- 假定观测数据点彼此独立，则联合似然函数如下

$$\begin{aligned} & P[(a_1, b_1)(a_2, b_2)\dots|x] \\ &= \prod_i P[(a_i, b_i)|x] \\ &= \prod_i P[b_i - a_i^T x] \\ &\propto \exp - \frac{\sum_i (b_i - a_i^T x)^2}{2\sigma^2} = \exp - \frac{\|Ax - b\|_2^2}{2\sigma^2} \end{aligned}$$

把 a_i, b_i 纵向列写出来



最大似然估计

- 最大似然估计寻找使似然函数最大化的 x

$$\begin{aligned}\hat{x} &= \arg \max_x P[(a_1, b_1)(a_2, b_2)\dots|x] \\ &= \arg \max_x \exp -\frac{\|Ax - b\|_2^2}{2\sigma^2} \\ &= \arg \min_x \|Ax - b\|_2^2\end{aligned}$$

- 最小二乘对应于高斯噪声假设下的最大似然估计

非线性最小二乘

- 一般的非线性模型: $b = f_x(a)$

- 定义残差
$$R(x) = \begin{pmatrix} b_1 - f_x(a_1) \\ \vdots \\ b_n - f_x(a_n) \end{pmatrix}$$

- 最大似然估计等效于最小化MSE

$$\hat{x} = \arg \min_x \|R(x)\|_2^2$$

病态问题

- 病态问题指的是问题的解不唯一

例如： $Ax = b$ 当方程数小于变量数

- 正则化：通过添加先验知识来约束解的性质

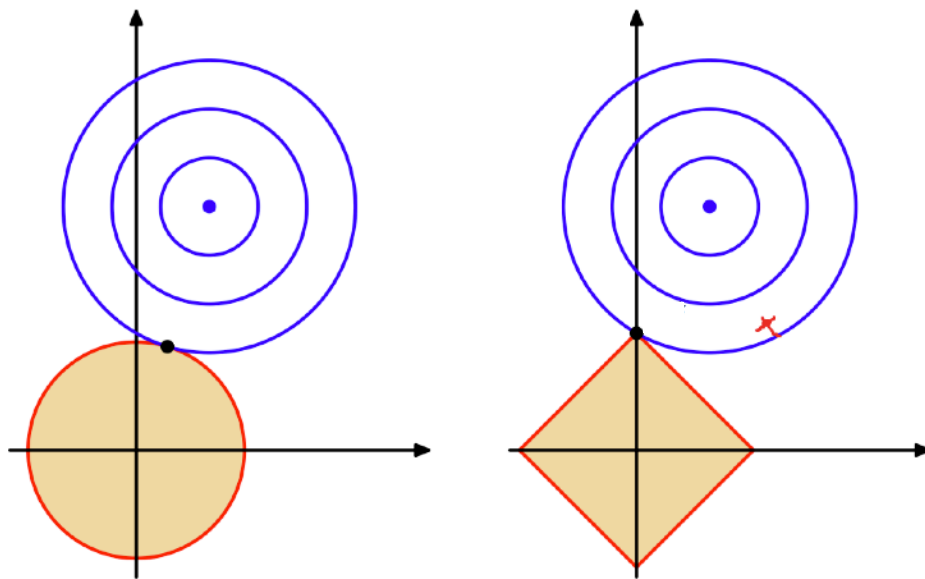
- L2正则： $\min_x \|Ax - b\|_2^2 + \lambda \|x\|_2^2$

- L2正则可以让x的值趋向于0

- 作用：抑制冗余的变量

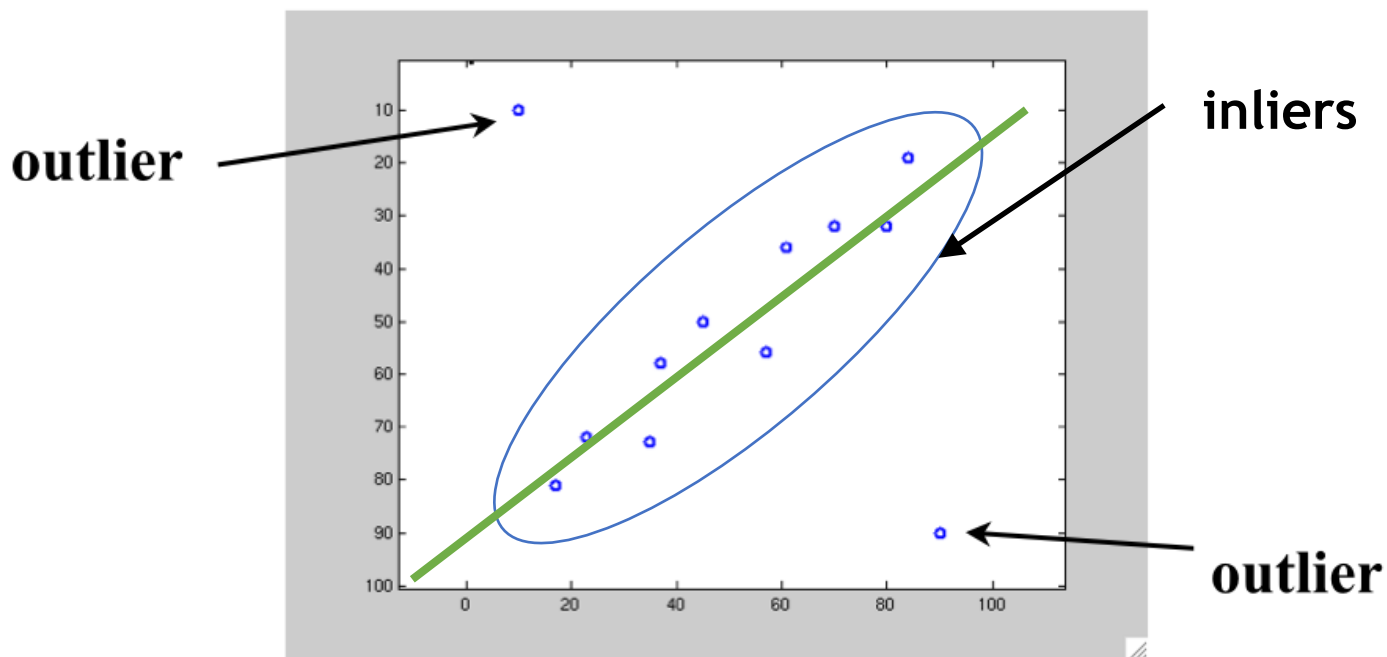
L1正则

- L1范数: $\|x\|_1 = \sum_i |x_i|$
- L1正则: $\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1$
- L1正则可以让x变得稀疏



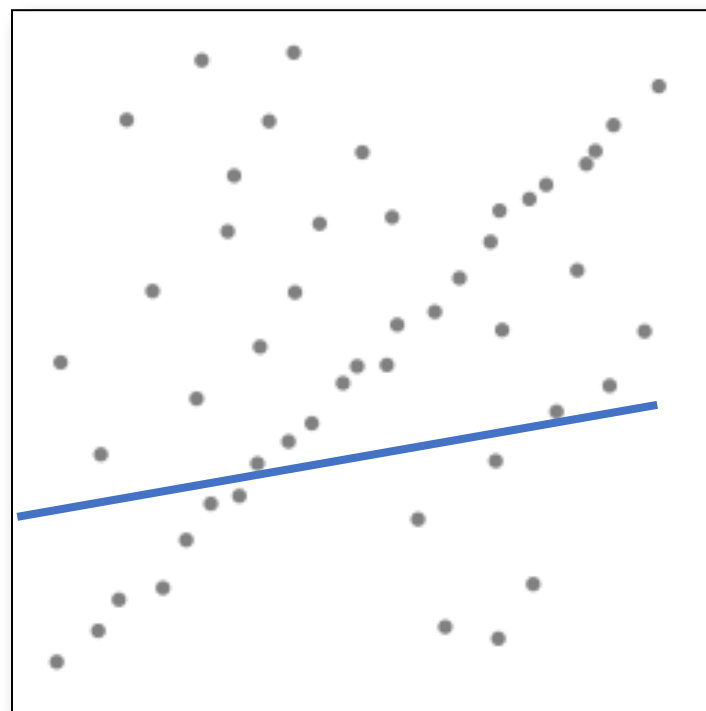
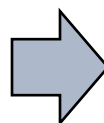
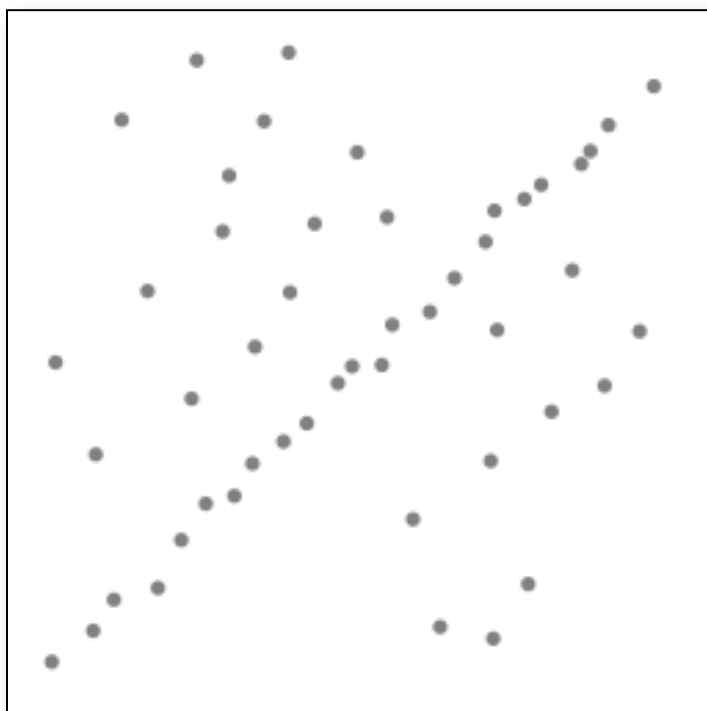
正常值与离群值

- 正常值 (Inlier) : 满足模型/噪音假设的数据点
- 离群值 (Outlier) : 不满足模型/噪音假设的错误点



离群值

- 离群值会让最小二乘回归失败

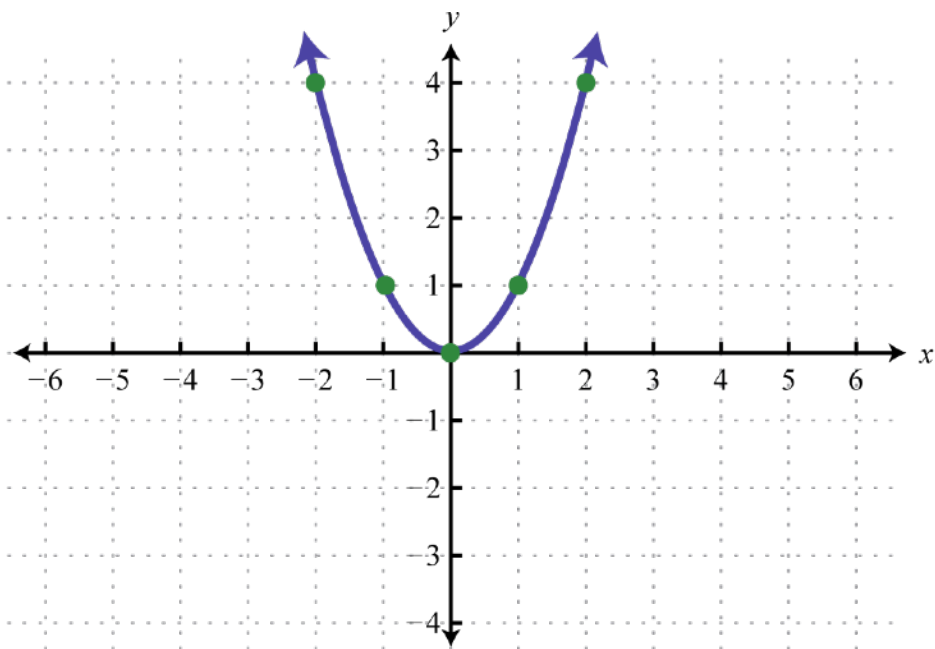


离群值

- 为什么离群值会让最小二乘回归失败
 - MSE（平方误差）与残差平方成正比
 - 受离群值影响过大

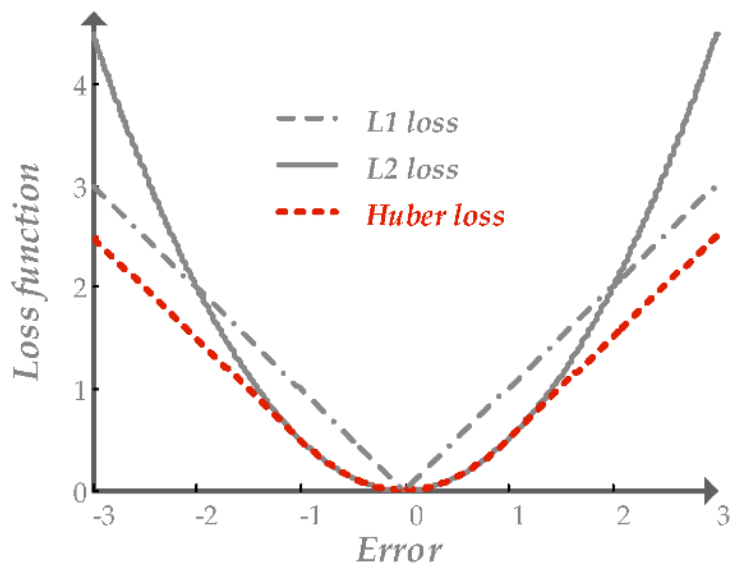
$$f(x) = x^2$$

x	$f(x)$
-2	4
-1	1
0	0
1	1
2	4



鲁棒估计

- 如何降低离群值的影响?
 - 用其他目标函数替换MSE，降低对离群值的惩罚
 - 例如L1范数、Huber函数

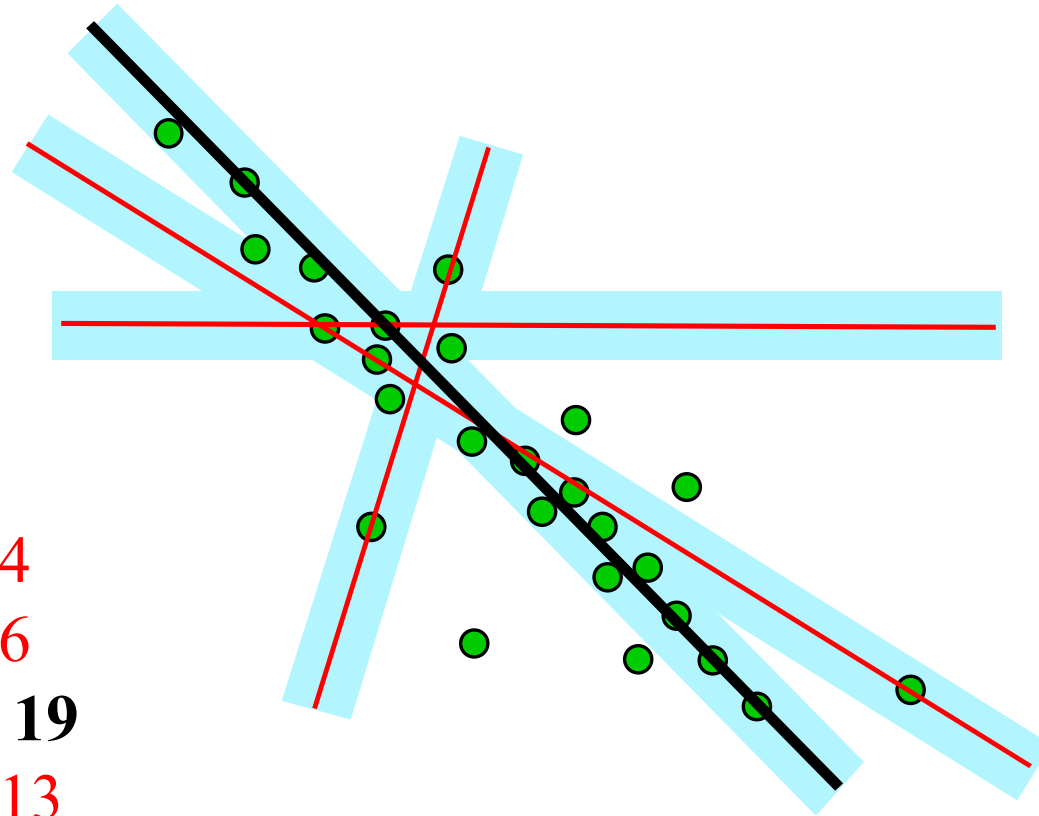


- 这种目标函数一般称为鲁棒函数

RANSAC

- 除了鲁棒函数，还有其他处理离群值的方法
- RANSAC: Random Sample Consensus
 - 目前最常用的处理离群值的方法
 - 核心思想：
 - Inlier的分布总是相似， outlier的分布各有各的不同
 - 用数据点对可能的模型参数进行投票

Ransac Procedure



Count = 4

Count = 6

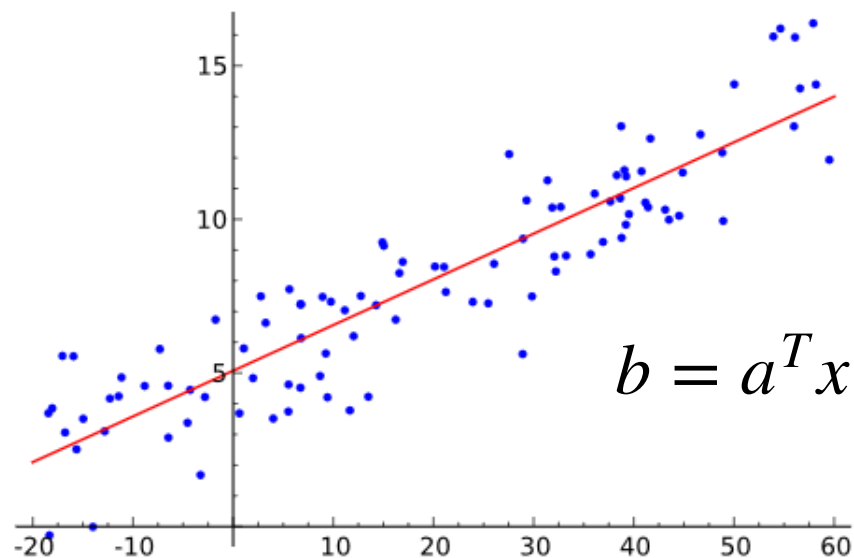
Count = 19

Count = 13

数值优化方法

Recap: 线性回归

- 给定了一系列观测值 (a_i, b_i) , 如何确定 x ?



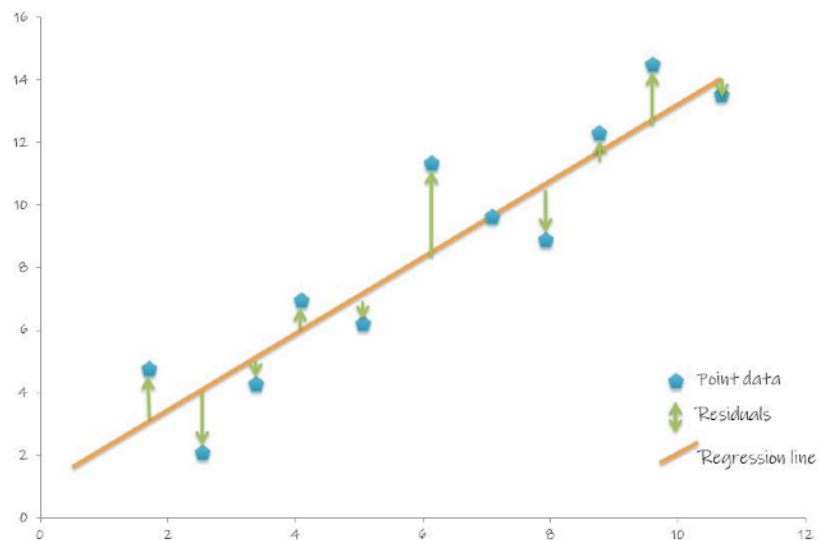
- 从数据中估计模型参数一般称为模型拟合 (model fitting) 或者回归 (regression)

Recap: 最小二乘法

- 最小二乘法: 最小化**均方误差** (Mean Square Error, 简称MSE)

$$\hat{x} = \arg \min_x \sum_i (b_i - a_i^T x)^2$$

- 残差: $r_i = b_i - a_i^T x$
- 残差向量: $R = [r_1, \dots, r_n]^T$



Recap: 非线性最小二乘

- 一般的非线性模型: $b = f_x(a)$

- 定义残差
$$R(x) = \begin{pmatrix} b_1 - f_x(a_1) \\ \vdots \\ b_n - f_x(a_n) \end{pmatrix}$$

- 最大似然估计等效于最小化MSE

$$\hat{x} = \arg \min_x \|R(x)\|_2^2$$

如何求解最优化问题？

- 有些问题有解析解
 - 线性最小二乘

$$\hat{x} = \arg \min_x \|Ax - b\|_2^2$$

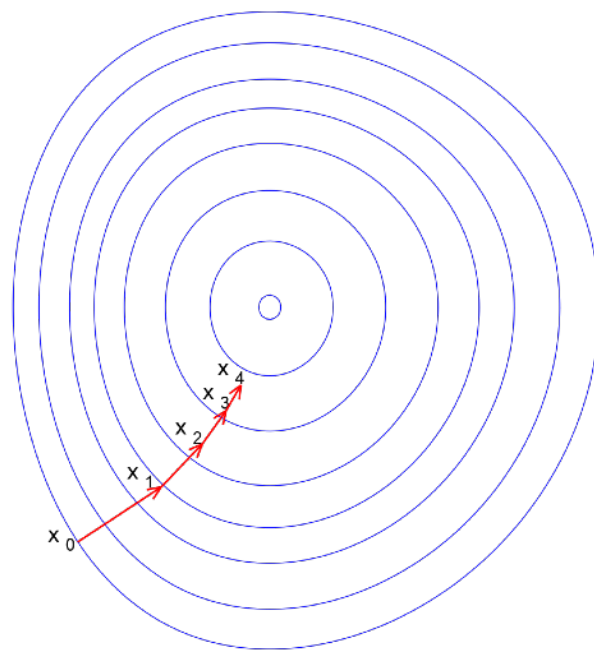
的解为以下线性方程组的解（目标函数关于x求导为0）

$$A^T Ax = A^T b$$

如何求解最优化问题？

- 大部分问题没有解析解的问题
- 通用策略：寻找一系列使目标函数不断下降的变量值完成优化

$$F(x_0) > F(x_1) > \dots > F(x_k) > \dots$$



梯度下降

- $x \leftarrow x_0$ %初始化
- while not converge
 - $p \leftarrow \text{descending_direction}(x)$ %确定下降方向
 - $\alpha \leftarrow \text{descending_step}(x, p)$ %确定步长
 - $x \leftarrow x + \alpha p$ %更新变量
- 如何确定下降方向与步长?

Preliminary

- 函数f的雅可比矩阵 (Jacobian)

- 一阶导数

- 单输出函数的梯度

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

- 函数f的海森矩阵 (Hessian)

- 二阶导数

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

下降方向的确定

- 对目标函数 $F(x)$ 在 x_0 上进行一阶Taylor展开

$$F(x_0 + \Delta x) \approx F(x_0) + J_F \Delta x$$

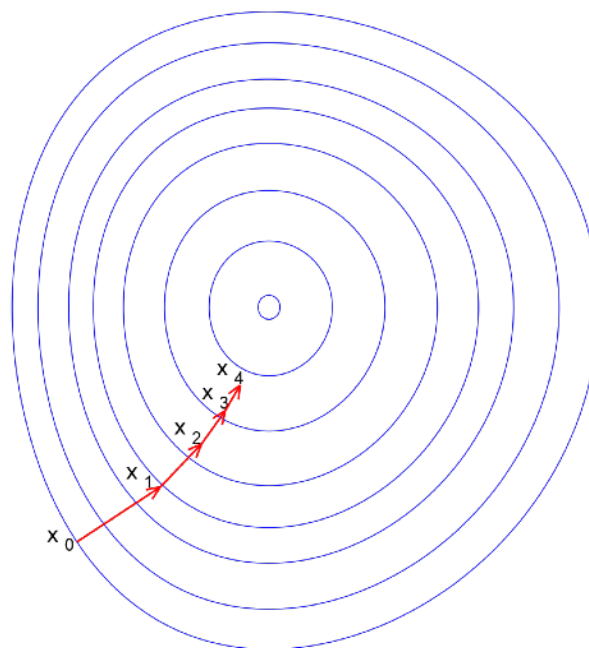
- 当 $J_F \Delta x < 0$ 时，函数的值会下降（ Δx 要足够小）

最速下降法

- $F(x_0 + \Delta x)$ 何时下降速度最快
 - 当 Δx 的方向与 $-\mathbf{J}_F^T$ （负梯度方向）相同时

- 算法:

- $x \leftarrow x_0$
- while not converge
 - $x \leftarrow x - \alpha \mathbf{J}_F^T$



梯度方向垂直于等高线

步长的确定

- 步长 α 如何确定?

确定 h 之后 F 是 α 的函数 $\phi(\alpha) = F(\mathbf{x} + \alpha\mathbf{h})$, \mathbf{x} and \mathbf{h} fixed, $\alpha \geq 0$.

1. α 太小, 函数值变化太小

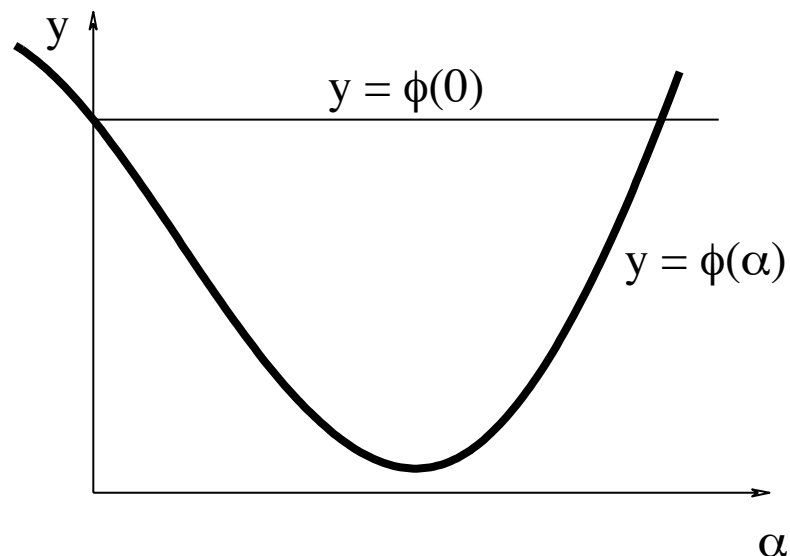
- 需要增大 α

2. α 太大, $\phi(\alpha) > \phi(0)$

- 需要减小 α

3. α 接近于 $\phi(\alpha)$ 的minimizer

- 可接受的 α 值



步长的确定

- 步长 α 如何确定?

1. Exact line search (太耗时)

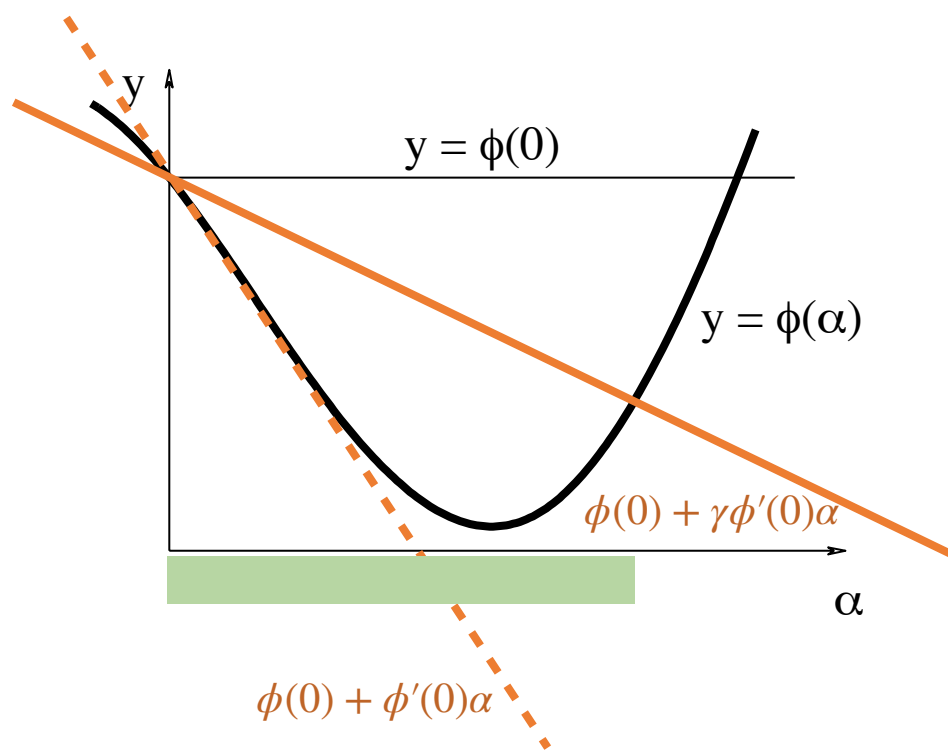
2. Backtracking algorithm

- Initialize α with a big value

- Decrease α until

$$\phi(\alpha) \leq \phi(0) + \gamma\phi'(0)\alpha$$

- 物理含义：满足上式时， α 处于右图中绿色区间



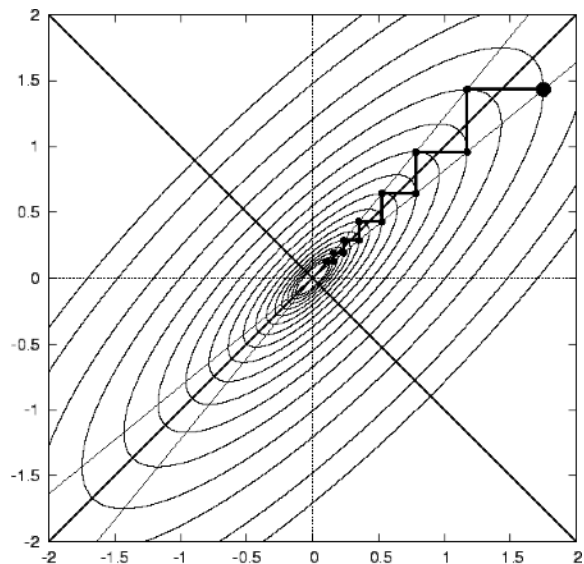
最速下降法

- 优点:

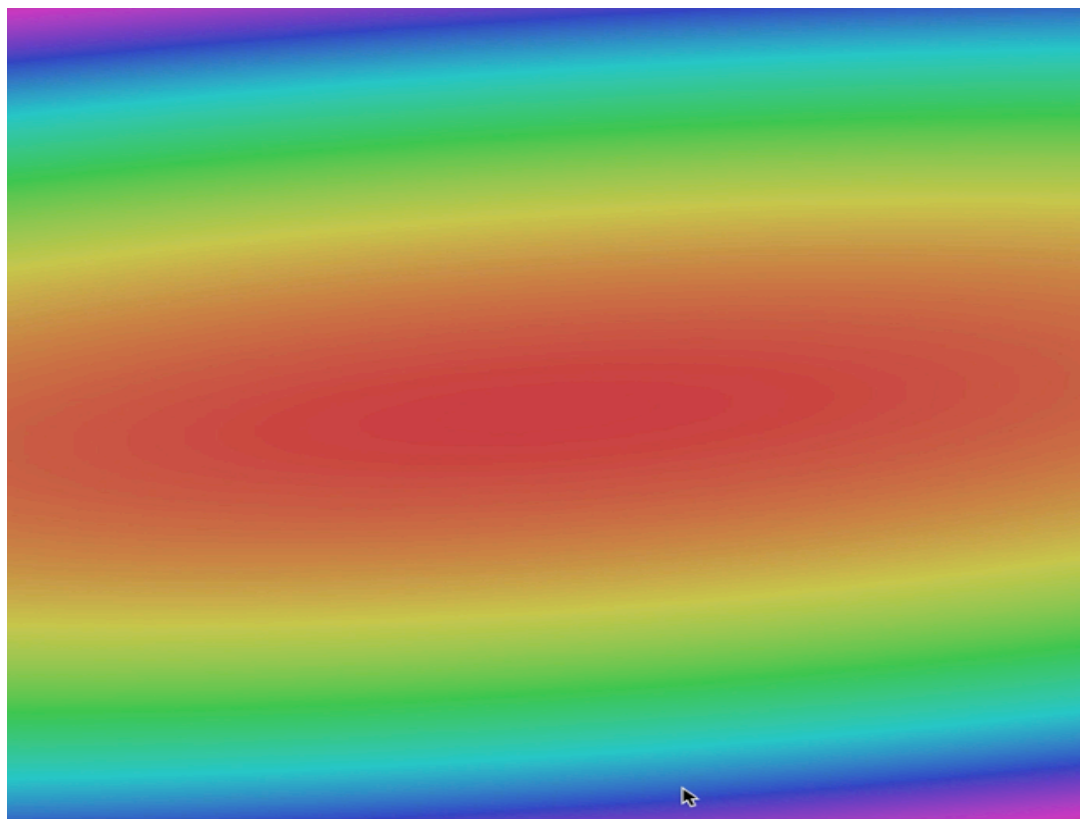
- 实现简单计算量少
- 在距离最优值较远时往往表现很好（启动快）

- 缺点:

- 在最优值附近收敛缓慢
- 当能量函数性质不好时会浪费很多迭代
- 为什么收敛慢?
 - 只利用了一阶梯度信息
 - 没有利用曲率信息



最速下降法



牛顿法

- 利用目标函数的二阶导数（曲率）信息
- 在 x_k 附近进行二阶展开

$$F(x_k + \Delta x) \approx F(x_k) + J_F \Delta x + \frac{1}{2} \Delta x^T H_F \Delta x$$

- 求解最小化 $F(x_k + \Delta x)$ 的 Δx

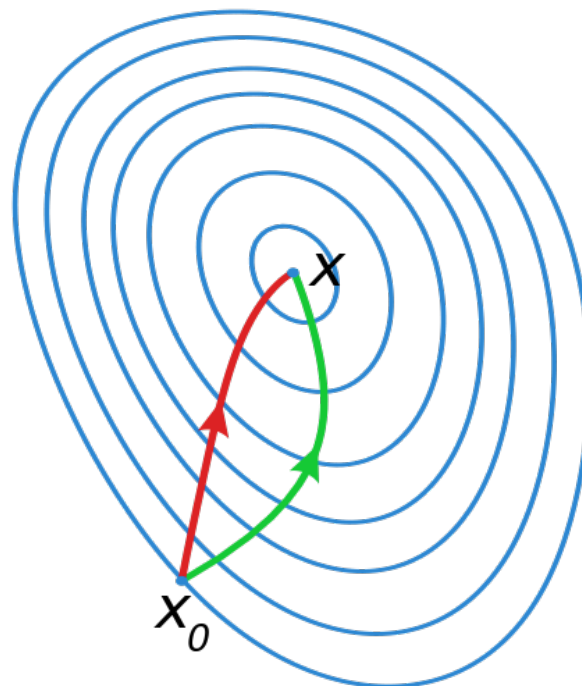
$$H_F \Delta x + J_F^T = 0$$

- 最佳下降方向（牛顿步）：

$$\Delta x = -H_F^{-1} J_F^T$$

牛顿法

- 优点：收敛快
 - 在极值的邻域内二次收敛
 - 适合接近最终结果时使用
- 缺点：Hessian 计算量很大
 - 有时无法计算
 - 能不能近似？



绿色：梯度下降
红色：牛顿法

高斯牛顿法

- 利用问题的性质来近似求解二阶导
 - 最小二乘问题

$$\hat{x} = \arg \min_x \|R(x)\|_2^2$$

- 对R进行一阶近似:

$$\begin{aligned} \|R(x_k + \Delta x)\|_2^2 &\approx \|R(x_k) + J_R \Delta x\|_2^2 \\ &= \underbrace{\|R(x_k)\|_2^2}_{F(x_k)} + \underbrace{2R(x_k)^T J_R \Delta x}_{J_F} + \Delta x^T J_R^T J_R \Delta x \end{aligned}$$

高斯牛顿法

- 此时最优的 Δx 满足

$$J_R^T J_R \Delta x + J_R^T R(x_k) = 0$$

- 最优下降方向:

$$\Delta x = - (J_R^T J_R)^{-1} J_R^T R(x_k)$$

- 对比牛顿法:

- 牛顿步: $\Delta x = - H_F^{-1} J_F^T$

- 高斯牛顿法用 $J_R^T J_R$ 近似Hessian矩阵 H_F

高斯牛顿法

- 优点
 - 不需要Hessian, 容易计算
 - 收敛快
- 缺点
 - 如果 $J_R^T J_R$ 不可逆, 算法会变得不稳定

Levenberg-Marquardt

- LM法通过“正则化”回避这个问题

$$\Delta x = - (J_R^T J_R + \lambda I)^{-1} J_R^T R(x_k)$$

- 对于全部 $\lambda > 0$, $J_R^T J_R + \lambda I$ 一定是正定的

Levenberg-Marquardt

$$\Delta x = - (J_R^T J_R + \lambda I)^{-1} J_R^T R(x_k)$$

- λ 的效果
 - $\lambda \rightarrow \infty$: 梯度下降步, 并且长度短
 - $\lambda \rightarrow 0$: Gauss-Newton 步
- λ 的选择
 - 每轮迭代更新
 - 当下降明显时, $\lambda \downarrow$
 - 当下降不明显时, $\lambda \uparrow$

Levenberg-Marquardt

- 优点：
 - 启动快 ($\lambda \uparrow$)
 - 收敛快 ($\lambda \downarrow$)
 - 不退化 ($J_R^T J_R + \lambda I$ 总是正定)
 - LM = 梯度下降+高斯牛顿

局部最优与全局最优

- 梯度下降只能找到局部最优解
- 对于特殊性质的函数，局部最优即全局最优（比如凸函数）

