# 机器学习

## 第 3 章监督学习-回归分析

欧阳毅

浙江工商大学
管理工程与电子商务学院

2023 年 3 月 5 日

# 目录

# 单变量回归

**例**

若你是一个房屋租赁中介，手里已经有一些房屋的面积和租赁价格数据，对于新进的房屋已知面积如何确定它的租赁价格?

```
房屋面积X，租赁价格Y
[ 325,       3185],
[ 306,       2500],
[ 278,       1750],
[ 208,       1500],
[ 262,       1923]
```

# 单变量回归

**例**

若你是一个房屋租赁中介，手里已经有一些房屋的面积和租赁价格数据，对于新进的房屋已知面积如何确定它的租赁价格?

- 构建预测模型 (定义假设函数 h):

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- 定义预测损失函数,(最小二乘法)

$$L(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

# 单变量回归

$$h_\theta(x) \qquad X \qquad \theta$$

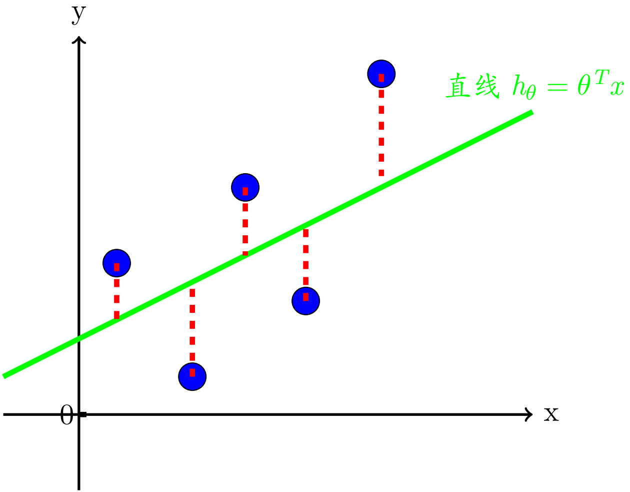
$$\begin{pmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ h_\theta(x^{(3)}) \\ h_\theta(x^{(4)}) \\ h_\theta(x^{(5)}) \end{pmatrix} = \begin{pmatrix} 1 & 325 \\ 1 & 306 \\ 1 & 278 \\ 1 & 208 \\ 1 & 262 \end{pmatrix} \times \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}$$

写成矩阵形式:

$$L(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y)$$

- 损失函数 L 只取决于 $\theta_0, \theta_1$
- 求 L 函数的极值点，可采用梯度下降法求解

# 单变量回归




线性回归线就是蓝色的点到回归线的垂直距离和最小的直线。
上图中红色的线，即真实数据到回归线的垂直距离，就是真实数据与回归线（预测数据）的误差

# 单变量回归

- 求 L 函数的极值点，可采用梯度下降法求解
- $\frac{\partial L(\theta_0,\theta_1)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}[\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$
  $= \frac{\partial}{\partial \theta_j}[\frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2]$

$$\theta_0 : \frac{\partial L(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} + y^{(i)})$$

$$\theta_1 : \frac{\partial L(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} + y^{(i)})x^{(i)}$$

```
def h(X, theta):
    return np.dot(X,theta.T)

def computeCost(X, theta, y):
    return 0.5 * np.mean(np.square(h(X, theta) - y))

def gradientDescent(X, theta, y, iterations, alpha):
    CostL = []
    CostL.append(computeCost(X, theta, y))
    for i in range(iterations):
        grad0 = np.mean(h(X, theta) - y)
        grad1 = np.mean((h(X, theta) - y) * (X[:,1].T))
        theta[0] = theta[0] - alpha * grad0
        theta[1] = theta[1] - alpha * grad1
        CostL.append(computeCost(X, theta, y))
    return theta, CostL
```

# 目录

# 最小二乘法 Ordinary Least Square，OLS

给定一个输入向量 $X = (X_1, X_2, ..., X_p)$ ，通过以下模型来预测输出 Y:

$$\hat{Y} = \theta_0 + \sum_{j=1}^{p} X_j \theta_j$$

可以写成内积 (X 是列向量)：

$$\hat{Y} = X^T \theta$$

**最小二乘法 (Least Square)**

选择系数 $\theta$，使得残差的平方和最小

$$RSS(\theta) = \sum_{i=1}^{N} (y_i - x_i^T \theta)^2$$

# 最小二乘法

$$RSS(\theta) = (y - X\theta)^T(y - X\theta)$$

其中 X 是 $N \times p$ 的矩阵，每行是一个输入向量，而 y 是训练数据集中标签向量，为求极值点，对上式关于 $\theta$ 求微分 =0，得到标准方差 (normal equation)

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

如果 $X^TX$ 是非奇异的，则有唯一解

**最小二乘法 (Least Square)**

损失函数：$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x^{(i)}) - y^{(i)})^2$

$$\hat{\theta} = (X^TX)^{-1}X^Ty$$

# 目录

# 岭回归 (Ridge Regression)

- 损失函数：$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{m}\theta_j^2$

$$\hat{\theta}^{ridge} = \arg\min_{\theta}\{\sum_{i=1}^{N}(y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j)^2 + \lambda\sum_{j=1}^{p}\theta_j^2\}$$

- 等价于

$$\hat{\theta}^{ridge} = \arg\min_{\theta}\{\sum_{i=1}^{N}(y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j)^2\}$$

s.t. $\sum_{j=1}^{p}\theta_j^2 \leq s$

- $\lambda$ 或 s 控制了模型复杂度

# 岭回归 (Ridge Regression)

- 残差平方和 (Residual sum of squares ,RSS)

$$RSS = (y - X\theta)^T(y - X\theta) + \lambda\theta^T\theta$$

$$\hat{\theta}^{ridge} = (X^TX + \lambda I)^{-1}X^Ty$$

- 当 $X^TX$ 为奇异矩阵时，解也存在，更具鲁棒性

**证明.**

RSS 对 $\theta$ 求导，最小值，导数为 0

$$\frac{\partial RSS}{\partial \theta} = 2X^TX\theta - 2X^Ty + 2\lambda\theta = 0$$

$$(X^TX + \lambda I)\theta = X^Ty$$

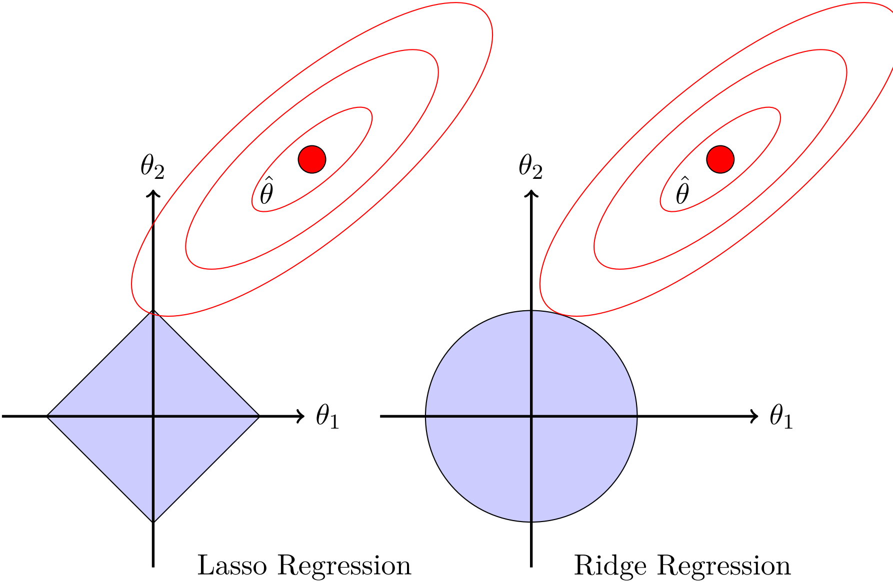
$$\theta = (X^TX + \lambda I)^{-1}X^Ty$$

□

# Lasso Regression

- 损失函数：$J(\theta)=\frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{m}|\theta_j|$
- $$\hat{\theta}^{lasso}=\arg\min_{\theta}\{\sum_{i=1}^{N}(y_i-\theta_0-\sum_{j=1}^{p}x_{ij}\theta_j)^2\}$$
  $$s.t.\sum_{j=1}^{p}|\theta_j|\leq s$$
- 与岭回归相比：惩罚项替换为使用 $\sum_{j=1}^{p}|\theta_j|$
- Lasso 回归能够使得损失函数中的许多 $\theta$ 均变成 0，这点要优于岭回归，因为岭回归是要所有的 $\theta$ 均存在的，这样计算量 Lasso 回归将远远小于岭回归。

# Lasso Regression



# 目录

1. 回归分析
   - 线性回归
   - 最小二乘法
   - 岭回归 (Ridge Regression)
   - 多变量回归
   - 逻辑回归
   - 随机抽样一致 RANSAC

# 多变量回归

**例**

若你是一个房屋租赁中介，手里已经有一些房屋的面积和租赁价格数据，对于新进的房屋已知面积如何确定它的租赁价格?

```
房屋面积x1, 房间数量x2, 楼层x3, 房龄x4,    租赁价格Y
[ 325,          5,          2,      5,      3185],
[ 306,          5,          4,      6,      2500],
[ 278,          4,          8,      7,      1750],
[ 208,          3,          6,      8,      1500],
[ 262,          3,          5,      15,     1923]
```

# 多变量回归

- 单变量线性回归:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- 多变量线性回归:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

- 通用表达

$$h_\theta(x) = \theta^T x$$

# 多变量回归

- 预测假设:

$$h_\theta(x) = \theta^T x$$

- 模型参数:

$$\theta = \{\theta_0, \theta_1., ...\}$$

- 损失函数:

$$L(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- 学习方式: 梯度下降

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}L(\theta_0, \theta_1 ... \theta_n)$$

# 多变量回归

## 正规方程的表示 $\theta = (X^TX)^{-1}X^Ty$

**证明.**

1. 损失函数：$L(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
2.L 对 $\theta$ 求导，最小值，导数为 0

$$\frac{\partial L}{\partial \theta} = 2X^TX\theta - 2X^Ty = 0$$

3. 解方程：(需要 $X^TX$ 可逆)

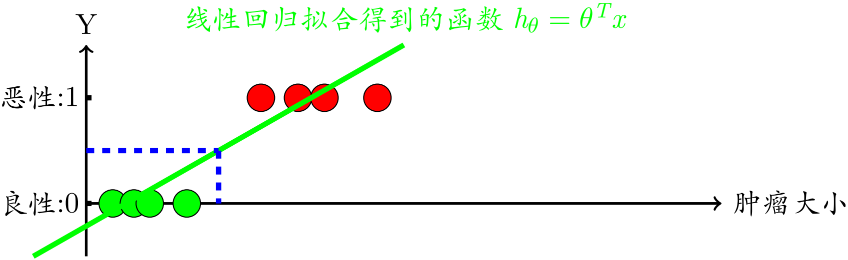
$$X^TX\theta = X^Ty \rightarrow \theta = (X^TX)^{-1}X^Ty$$

$\square$

# 目录

# 逻辑回归

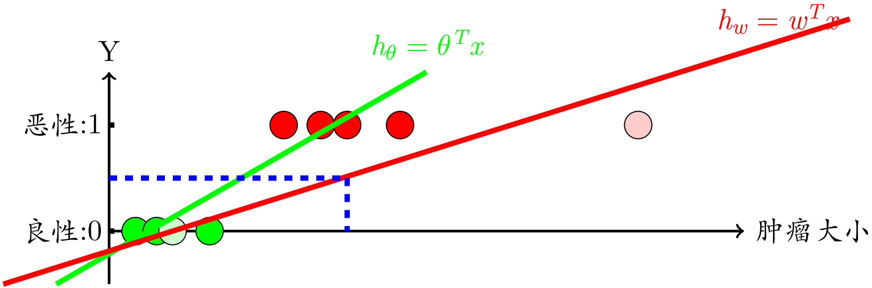
- 线性回归能解决分类问题吗?




$$
\begin{cases} \text{预测 } 1 & if \quad h_\theta(x) \geq 0.5 \\ \text{预测 } 0 & if \quad h_\theta(x) < 0.5 \end{cases}
$$

# 逻辑回归

- 线性回归能解决分类问题吗?




$$\begin{cases} \text{预测 } 1 & \text{if} \quad h_w(x) = w^T x \geq 0.5 \\ \text{预测 } 0 & \text{if} \quad h_w(x) = w^T x < 0.5 \end{cases}$$

# 逻辑回归

- 线性回归不能解决分类问题
- 因此我们引入逻辑回归（Logistic Regression）：

$$0 \le h_\theta(x) \le 1$$

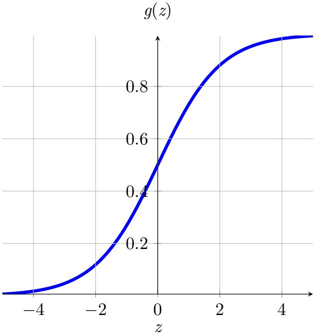
- 逻辑回归可以作为分类算法

$$h_\theta(x) = g(\theta^T x) : g(z) = \frac{1}{1 + e^{-z}}$$

- $h_\theta(x) = P(y = 1 | x, \theta)$

# 逻辑回归

- $z = \theta^T x$
- 如果 $\theta^T x \geq 0$ 则预测 y=1
- 如果 $\theta^T x < 0$ 则预测 y=0
- L 损失函数是一个非凸函数，需要另外定义



# 逻辑回归

## 损失函数

$$L(\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

$$\log L(h_\theta(x), y) = \begin{cases} \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x)) & if \quad y = 1 \\ \sum_{i=1}^{m} (1 - y^{(i)}) log(1 - h_\theta(x)) & if \quad y = 0 \end{cases}$$

把分段函数写成一个表达式:

$$\log L(h_\theta(x), y) = \sum_{i=1}^{m} [y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))]$$

该函数是凸函数，因为 Hessian 矩阵恒大于 0，因此可用梯度下降法求解最优值。

# 逻辑回归 I

## 损失函数

**定义**

给定一个大小为 $n \times n$ 的实对称矩阵 A ，若对于任意长度为 n 的非零向量 x ，有 $x^T Ax > 0$ 恒成立，则矩阵 A 是一个正定矩阵。

$$H = \begin{bmatrix} \frac{\partial^2 f}{(\partial x_1)(\partial x_1)} & \cdots & \frac{\partial^2 f}{(\partial x_1)(\partial x_n)} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{(\partial x_n)(\partial x_1)} & \cdots & \frac{\partial^2 f}{(\partial x_n)(\partial x_n)} \end{bmatrix} \tag{1}$$

**定义**

若 $f(x)$ 的 Hessian 是正定矩阵, 则目标函数是凸函数

可证明逻辑回归损失函数的 Hessian 矩阵是正定的

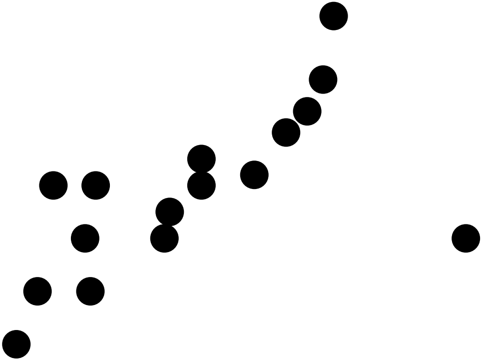
```
theta= np.zeros(X.shape[1])
def g(x):
    return 1 / (1 + np.exp(-x))

def cost(theta, X, y):
    return np.mean(- y * np.log(g(np.dot(X, theta)))
        - (1 - y) * np.log(1 - g(np.dot(X, theta))))
def gradient(theta, X, y):
    M=X.shape[0]
    return (1/M) * np.dot(X.T, g(np.dot(X, theta)) - y)

import scipy.optimize as opt
res = opt.minimize(fun=cost, x0=theta, args=(X, y),
          jac=gradient, method='Newton-CG')
theta_result = res.x
```

# 目录

# RANSAC 回归

- RANSAC 算法，Random Sample Consensus（随机抽样一致）[Fischler & Bolles 1981]

思路 我们想避免外点集对于拟合的影响，因此寻找内点集，并且仅用它们来进行函数拟合



# RANSAC 回归

算法

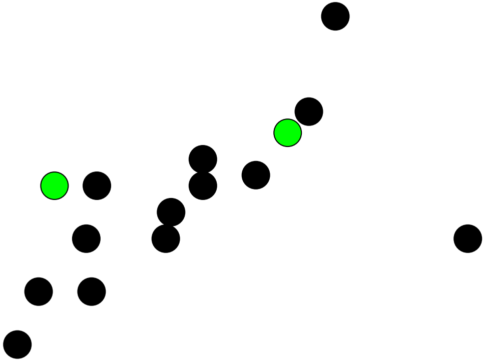
- 随机选取种子点集, 并基于它们进行变换估计
- 计算种子点集的变换
- 找出内点集
- 若内点集的数量足够大，用最小二乘法估计找出所有内点

$$RSS = \min \sum_{i=1}^{M} (h_\theta(x_i) - y_i)^2$$

- 始终保持在最大内点集上进行变换

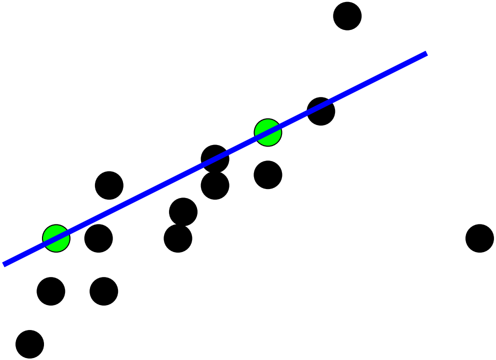
# RANSAC 线性拟合

思路 随机选取种子点集，采样两个点



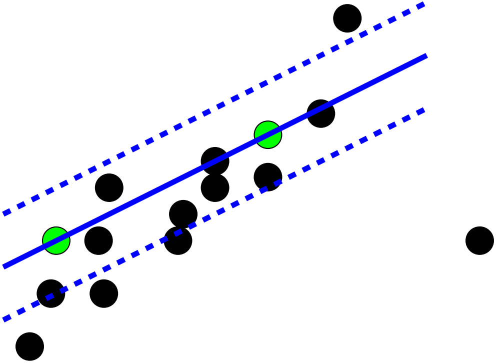
# RANSAC 线性拟合

思路 计算种子点集的变换: 用最小二乘法估计构建直线方程



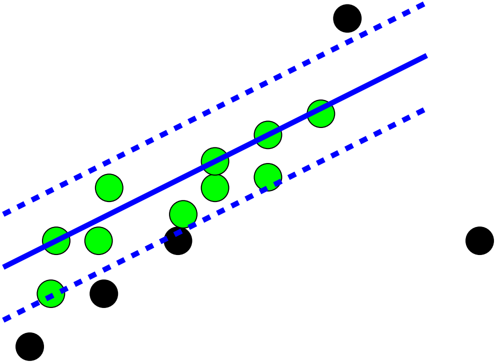
# RANSAC 线性拟合

思路 给定阈值，计算在阈值区间内点集



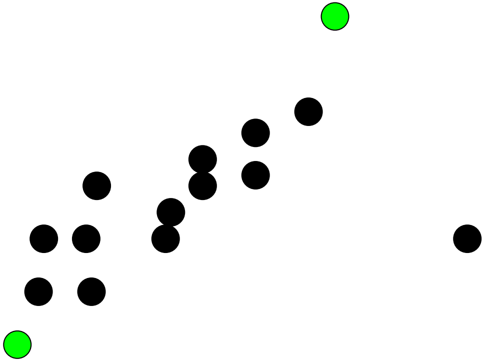
# RANSAC 线性拟合

思路 计算在阈值区间内点集，有 10 个内点



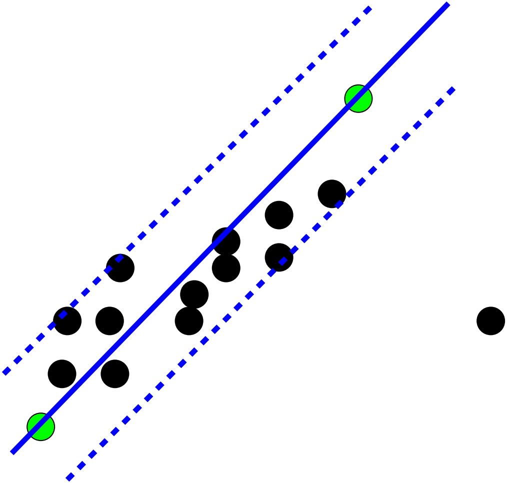
# RANSAC 线性拟合

思路 重新采样，重复上述计算过程，得到最好的拟合直线（内点集最大）



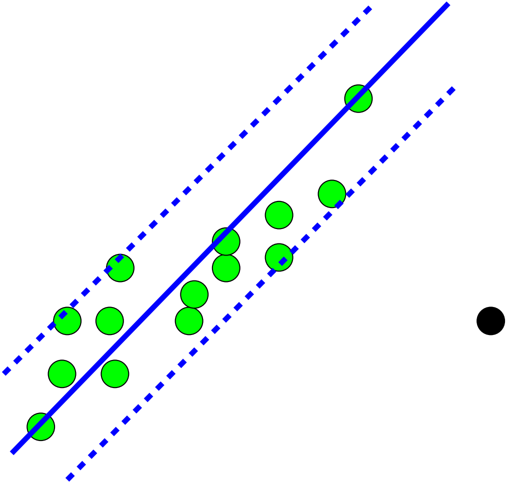
# RANSAC 线性拟合

思路 重新采样，重复上述计算过程，得到最好的拟合直线（内点集最大）



# RANSAC 线性拟合

思路 这次有 13 个内点（内点集最大）



# RANSAC 线性拟合 I

习题 1：给出 RANSAC 线性拟合的 python 实现