回归：多元线性回归（解析解与梯度下降算法）

覃雄派

回归：多元线性回归(解析解与梯度下降算法)

提纲

- 多元线性回归
- 增加常数特征x_0
- 多元线性回归的矩阵形式及其求解
- 多元线性回归的梯度下降法求解
- 梯度下降法的矩阵形式
- 一元/多元线性回归的评价

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归（Multiple Linear Regression, MLR）
  - 多元线性回归
  - 在简单（一元）线性回归SLR模型基础上添加**更多的独立变量**
- 多元线性回归的一般形式

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_d x_d = \theta_0 + \sum_{j=1}^{d} \theta_d x_d$$

- 基本概念：
  - 输入变量$x_1, x_2, \ldots, x_d$也称：特征（Feature）、协变量（Covariate)、解释变量（Explanatory Variable）、回归量（Regressor）
  - 参数$\theta_1, \theta_2, \ldots, \theta_d$度量了输入变量对预测值的**权重**
  - 参数θ_0为**截距项**

回归：多元线性回归（解析解与梯度下降算法）

- 向量点积
- 给定两个向量$\boldsymbol{a}$和$\boldsymbol{b}$

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- 它们的点积操作定义为

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots a_n b_n = \boldsymbol{a}^T \boldsymbol{b}$$

  - 两个向量的点积是一个标量，而非向量
  - 点积操作只能定义在两个相同长度的向量上
- 练习：
  - 将$a + bx_i$表示为点积的形式$[a \quad b]\begin{bmatrix} 1 \\ x_i \end{bmatrix}$

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归（Multiple Linear Regression, MLR）
  - 在简单（一元）线性回归SLR模型基础上添加更多的独立变量
- 针对每个数据点，添加一个常数特征$x_0 = 1$，得到

$$\hat{y} = \sum_{j=0}^{d} \theta_d x_d$$

- MLR模型举例：波士顿房价数据集
  - 输入变量
    - RM: average number of rooms per dwelling
    - LSTAT: % lower status of the population
  - 输出变量
    - Price: price of house

应该如何建立MLR模型?

回归：多元线性回归（解析解与梯度下降算法）

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归（Multiple Linear Regression, MLR）

$$\hat{y} = \sum_{j=0}^{d} \theta_d x_d$$

- 引入两个向量：

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \Rightarrow \hat{y} = [1 \quad x_1 \quad x_2 \quad \dots \quad x_d] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} = \boldsymbol{x}^T \boldsymbol{\theta}$$

  - 注：我们使用粗体表示向量及矩阵

请问上述公式中$\hat{y}$是

A. d*1的向量　B. 标量

回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵（Design Matrix）
- 给定训练集，我们可以定义设计矩阵

$$\mathbb{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & x_3^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & x_3^2 & \cdots & x_d^2 \\ 1 & x_1^3 & x_2^3 & x_3^3 & \cdots & x_d^3 \\ 1 & x_1^4 & x_2^4 & x_3^4 & \cdots & x_d^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^n & x_2^n & x_3^n & \cdots & x_d^n \end{bmatrix}$$

每行代表一个数据实例（数据点）；如数据点1的特征

样本数量

每列代表一个数据特征（输入变量）
如所有点在特征1上的取值

上标：样本编号
下标：维度编号

回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵（Design Matrix）
- 给定训练集，我们可以定义设计矩阵
- 波士顿房价的例子

	BIAS	RM	LSTAT
0	1	6.575	4.98
1	1	6.421	9.14
2	1	7.185	4.03
3	1	6.998	2.94
4	1	7.147	5.33
...	...	...	...
501	1	6.593	9.67
502	1	6.120	9.08
503	1	6.976	5.64
504	1	6.794	6.48
505	1	6.030	7.88

506 rows × 3 columns

途中对应的设计矩阵维数
A. 506*1 **B. 506*3** C. 3*506 D. 3*1

回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵（Design Matrix）
- 给定训练集，我们可以定义设计矩阵
- 波士顿房价的例子
- 基于设计矩阵，MLR模型表示为

	BIAS	RM	LSTAT
0	1	6.575	4.98
1	1	6.421	9.14
2	1	7.185	4.03
3	1	6.998	2.94
4	1	7.147	5.33
...	...	...	...
501	1	6.593	9.67
502	1	6.120	9.08
503	1	6.976	5.64
504	1	6.794	6.48
505	1	6.030	7.88

506 rows × 3 columns

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\hat{\mathbb{Y}} = \mathbb{X}\boldsymbol{\theta}$$

请结合左图分别说出$\hat{\mathbb{Y}}$、$\mathbb{X}$和$\boldsymbol{\theta}$的维数

A. 506*1　B. 506*3　C. 3*506　D. 3*1

回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵（Design Matrix）
- 基于设计矩阵，MLR模型表示为矩阵形式

$$\hat{\mathbb{Y}} = \mathbb{X}\boldsymbol{\theta}$$

$$\begin{bmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \hat{y}^3 \\ \hat{y}^4 \\ \vdots \\ \hat{y}^n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & x_3^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & x_3^2 & \cdots & x_d^2 \\ 1 & x_1^3 & x_2^3 & x_3^3 & \cdots & x_d^3 \\ 1 & x_1^4 & x_2^4 & x_3^4 & \cdots & x_d^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^n & x_2^n & x_3^n & \cdots & x_d^n \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \vdots \\ \theta_d \end{bmatrix}$$

- 上标表示样本编号，$\hat{y}^2$表示第二个样本的预测的y，为了和平方区分，有时记为$\hat{y}^{(2)}$
- 下标表示分量（第几个变量）

$$\hat{y}^{(2)} = \mathbb{X}_2\boldsymbol{\theta} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$$

回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵（Design Matrix）
- 基于设计矩阵，MLR模型表示为矩阵形式
- 请计算$\hat{y}_4$

	BIAS	RM	LSTAT
0	1	6.575	4.98
1	1	6.421	9.14
2	1	7.185	4.03
3	1	6.998	2.94
4	1	7.147	5.33
...	...	...	...
501	1	6.593	9.67
502	1	6.120	9.08
503	1	6.976	5.64
504	1	6.794	6.48
505	1	6.030	7.88

506 rows × 3 columns

$$\boldsymbol{\theta} = \begin{bmatrix} -1.36 \\ 5.09 \\ -0.64 \end{bmatrix}$$

1*(-1.36) +7.147*5.09 +5.33*(-0.64)

回归：多元线性回归（解析解与梯度下降算法）

- **针对单一数据点**
  - 模型表示

$$\hat{y} = \boldsymbol{x}^T \boldsymbol{\theta}$$

    - $\boldsymbol{x}$是长度为$d+1$的向量
    - $\hat{y}$是标量（一个y值）
    - $\boldsymbol{\theta}$是长度为$d+1$的向量
- **针对整个训练集**
  - 模型表示

$$\hat{\mathbb{Y}} = \mathbb{X}\boldsymbol{\theta}$$

    - $\mathbb{X}$是$n \times (d+1)$的矩阵
    - $\mathbb{Y}$是长度为n的向量
    - $\boldsymbol{\theta}$是长度为$d+1$的向量

注：为了表示方便，在不引起混淆的情况下，我们直接考虑d维

回归：一元回归（解析解与梯度下降算法）

- MSE目标函数的矩阵形式
- 给定SLR模型，均方误差MSE可以写为

$$R(a,b)=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-(a+bx_i)\right)^2$$

应该如何优化$R(\boldsymbol{\theta})$？

- 矩阵求导
- 利用几何含义求解

- 给定线性回归的矩阵形式，我们有

$$R(\boldsymbol{\theta})=\|\mathbb{Y}-\mathbb{X}\boldsymbol{\theta}\|_2^2=(y^1-x^1\boldsymbol{\theta})^2+(y^2-x^2\boldsymbol{\theta})^2+\cdots+(y^n-x^n\boldsymbol{\theta})^2$$

- 注：
  - 上式省略了常数项$\frac{1}{n}$
  - $\|\boldsymbol{x}\|_2=\sqrt{\sum_{i=1}^{n}x_i^2}$ 称为向量x的**L2范数（L2 Norm）**

注：矩阵求导计算超出本课的范围；单提供一些辅助材料帮助大家掌握；对后续深入理解机器学习，包括深度学习很有帮助。

回归：多元线性回归（解析解与梯度下降算法）

- 线性回归的矩阵表示可以看成d个特征向量的线性组合
- 预测值$\hat{\mathbb{Y}}$可以看为$\mathbb{X}$中各列的线性组合

- $\hat{\mathbb{Y}} = \mathbb{X}\boldsymbol{\theta} = [1 \quad \mathbb{X}^1 \quad \mathbb{X}^2 \quad \dots \quad \mathbb{X}^{\mathrm{d}}]\begin{bmatrix}\boldsymbol{\theta}_0 \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \dots \\ \boldsymbol{\theta}_{\mathrm{d}}\end{bmatrix}$
- $= \boldsymbol{\theta}_0 1 + \boldsymbol{\theta}_1 \mathbb{X}^1 + \boldsymbol{\theta}_2 \mathbb{X}^2 + \cdots + \boldsymbol{\theta}_d \mathbb{X}^d$

分量都是1的列

第1个变量的列

第2个变量的列

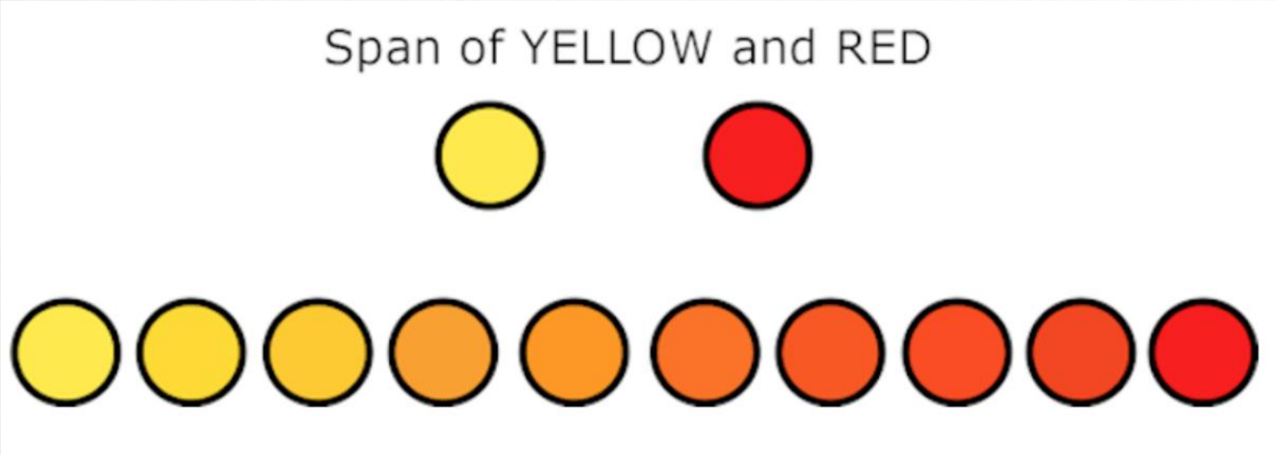
第d个变量的列

回归：多元线性回归（解析解与梯度下降算法）

- 线性代数：张成空间
- 张成空间，也称线性生成（Linear Span）的定义如下
  - 给定线性空间V，用V的若干向量构成集合 $S=\{\boldsymbol{v}_\alpha\}$，其中$S$中向量可以线性相关，也可线性无关。从$S$任意选择有限个向量进行线性组合所得到的集合$\{\sum_i^N c_i\boldsymbol{v}_i \mid N\in\mathbb{Z}, c_i\in\mathbb{R}\}$称为$S$张成（span）的空间，记为$\mathrm{span}(S)$
- 张成空间最直观的例子就是子空间（subspace）
  - 给定三维空间中三个向量$\boldsymbol{u}$, $\boldsymbol{v}$和$\boldsymbol{w}$，它们张成子空间$\mathrm{span}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w})$
  - 请回答以下线性组合是否属于$\mathrm{span}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w})$
    - $\boldsymbol{u}$
    - $3\boldsymbol{u}+0.5\boldsymbol{v}+10\boldsymbol{w}$
    - $\boldsymbol{u}\times\boldsymbol{v}+\boldsymbol{w}$
    - $2\boldsymbol{u}+5\boldsymbol{v}-6\boldsymbol{w}+10$

回归：多元线性回归（解析解与梯度下降算法）

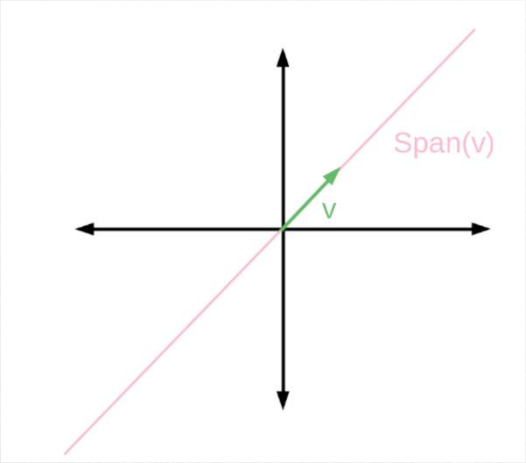
- 张成空间、及其直观解释
- 假设只给你黄色和红色颜料进行绘画，能够调制出的颜色集合，就可以形象地理解为黄色和红色的张成空间



向量空间里，是向量的线性组合

回归：多元线性回归（解析解与梯度下降算法）

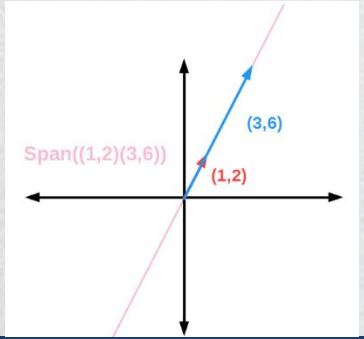
- 线性代数：张成空间
- 在二维空间（记作$\mathbb{R}^2$）的一个向量$\boldsymbol{v}$张成空间$\text{span}(\boldsymbol{v})$的直观含义
  - 例： $\boldsymbol{v} = (1,1)$

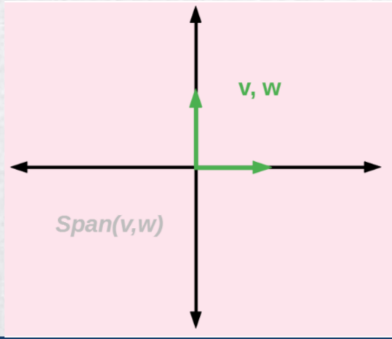


二维空间$\mathbb{R}^2$上通过向量$\boldsymbol{v}$的一条无限长的直线

回归：多元线性回归（解析解与梯度下降算法）

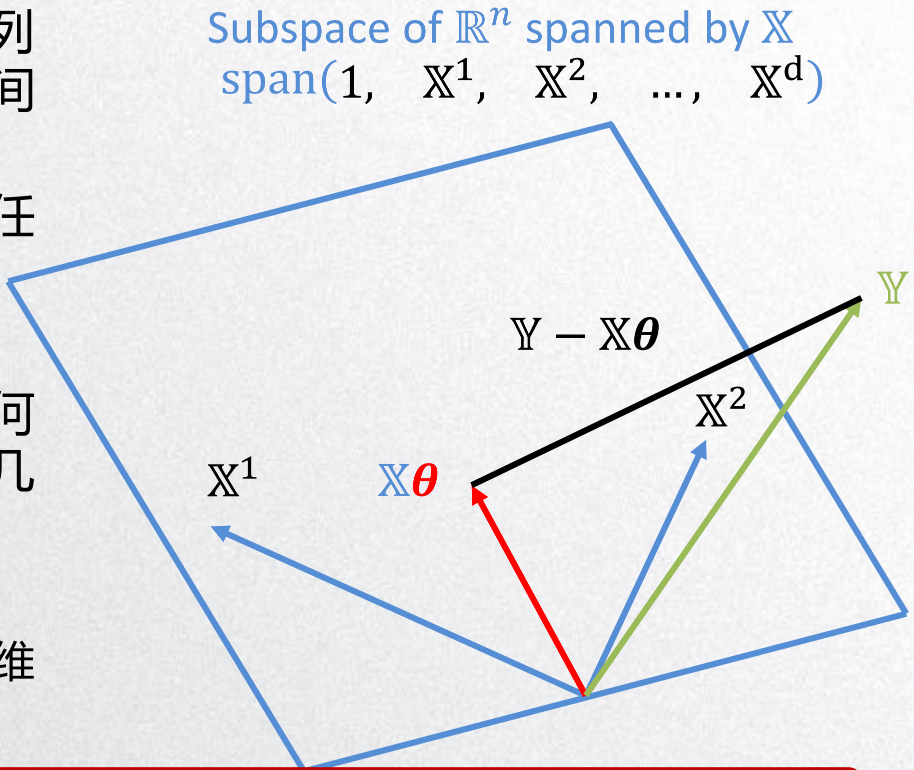
- 线性代数：张成空间
- 思考：
  - 二维空间（记作$\mathbb{R}^2$）中两个向量$\boldsymbol{v}_1$和$\boldsymbol{v}_2$的张成空间$\text{span}(\boldsymbol{v}_1, v_2)$的直观几何含义是什么？
  - 是一个二维的平面吗？不一定！
- 取决于两个向量是否线性无关





回归：多元线性回归（解析解与梯度下降算法）

- 设计矩阵 d +1 个列 $1, \mathbb{X}^1, \mathbb{X}^2, \ldots, \mathbb{X}^d$张成的子空间 $\mathrm{span}(1, \mathbb{X}^1, \mathbb{X}^2, \ldots, \mathbb{X}^d)$
- 预测模型$\mathbb{X}\boldsymbol{\theta}$是张成空间上的一个任意的向量（红色）
- 观测值$\mathbb{Y}$表示为另一向量（绿色）
- 目标函数$R(\boldsymbol{\theta}) = \|\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}\|_2^2$的几何含义是向量$\mathbb{Y}$和向量$\mathbb{X}\boldsymbol{\theta}$之间的欧几里得距离
- 令$R(\boldsymbol{\theta})$最小化的条件：
  - 向量$\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}$与设计矩阵$\mathbb{X}$张成的$n$维子空间正交（Orthogonal）



这里是两个向量的张成空间，以便查看

回归：多元线性回归（解析解与梯度下降算法）

- 利用几何含义解释MSE目标函数优化
- 令$R(\boldsymbol{\theta})$最小化的条件：
  - 向量$\mathbb{Y}-\mathbb{X}\boldsymbol{\theta}$与设计矩阵$\mathbb{X}$张成的$d$维子空间正交（Orthogonal）

- 根据**正交**的定义，我们得到

$$\mathbb{X}^T(\mathbb{Y}-\mathbb{X}\boldsymbol{\theta})=0$$

$$\Rightarrow \mathbb{X}^T\mathbb{Y}-\mathbb{X}^T\mathbb{X}\boldsymbol{\theta}=0$$

- 根据上式得到最优的参数估计

$$\boxed{\hat{\boldsymbol{\theta}}=\left(\mathbb{X}^T\mathbb{X}\right)^{-1}\mathbb{X}^T\mathbb{Y}}$$

回归：多元线性回归（解析解与梯度下降算法）

- 课堂练习
- 给定一组训练数据
  - (2,4)
  - (5,1)
  - (8,9)
- 请按照以下公式计算线性回归模型的最优参数$\hat{\boldsymbol{\theta}}$
  - 注：可以使用python辅助计算在此处键入公式。

$$\hat{\theta} = \left(\mathbb{X}^T \mathbb{X}\right)^{-1} \mathbb{X}^T \mathbb{Y}$$

把实际数据代入

$$\mathbb{X} = \begin{bmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 8 \end{bmatrix} \qquad \mathbb{Y} = \begin{bmatrix} 4 \\ 1 \\ 9 \end{bmatrix}$$

注意加上1这一列

回归：多元线性回归（解析解与梯度下降算法）

- 验证

```python
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([2,5,8]).reshape((-1, 1))
y = np.array([4,1,9])

model = LinearRegression()
model.fit(x, y)

r_sq = model.score(x, y)

print('intercept_:', model.intercept_)
print('coef_:', model.coef_)

# ----------
x = np.array([[1,2], [1,5],[1,8]])
y= np.array([4,1,9]).reshape(-1, 1)

print("x", x)
print("y", y)

t1 = np.dot(x.T,x)
print("t1",t1)

t1_inv = np.linalg.inv(t1)
print("t1_inv",t1_inv)

t2 = np.dot(t1_inv, x.T)
final = np.dot(t2, y)
print("final",final)
```

用LinearRegression模型

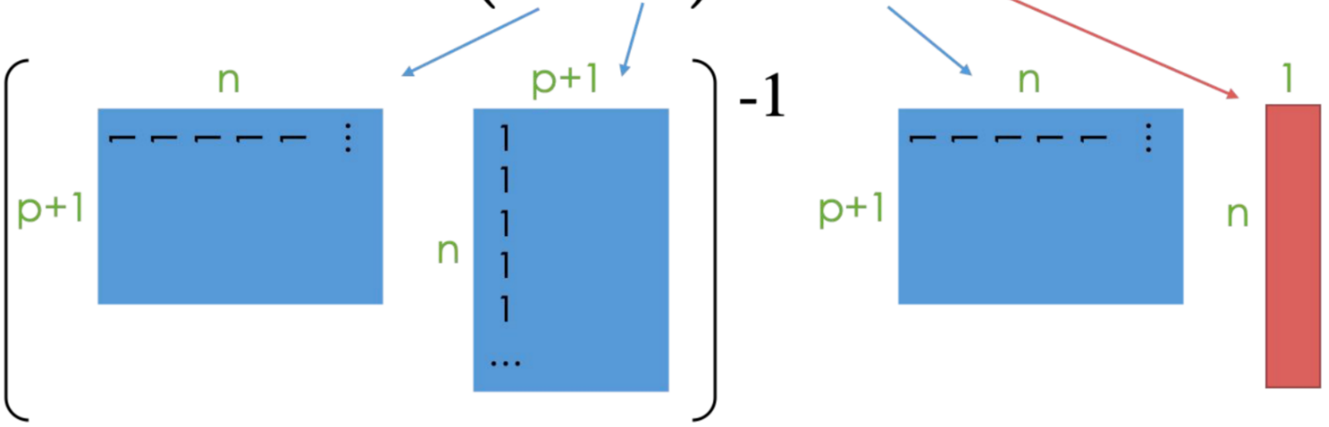
矩阵运算解析解

```
intercept_: 0.5000000000000009
coef_: [0.83333333]
x [[1 2]
 [1 5]
 [1 8]]
y [[4]
 [1]
 [9]]
t1 [[ 3 15]
 [15 93]]
t1_inv [[ 1.72222222 -0.27777778]
 [-0.27777778  0.05555556]]
final [[0.5       ]
 [0.83333333]]
```

回归：多元线性回归（解析解与梯度下降算法）

- 深入理解最优参数估计：最优的参数估计
  - 当维数d远小于数据量n时

$$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$$



回归：多元线性回归（解析解与梯度下降算法）

- 线性代数背景知识：矩阵的秩
- 一个矩阵$A_{m\times n}$的秩，记为$\mathrm{rank}(A)$
  - 定义为A包含的线性无关的列（或行）的最大个数

- 一些基本的性质：
  - 矩阵A的秩$\mathrm{rank}(A) \le \min(m, n)$
  - 在方块矩阵（$m = n$）的情况下，A是可逆的
  - 当且仅当$\mathrm{rank}(A) = n$，即矩阵A有满秩

回归：多元线性回归（解析解与梯度下降算法）

- 深入理解最优参数估计
- 上式中$\hat{\boldsymbol{\theta}}$有唯一解的条件是$\mathbb{X}^T\mathbb{X}$可逆
- $\mathbb{X}^T\mathbb{X}$可逆的充分必要条件是$\mathbb{X}^T\mathbb{X}$满秩，即秩为$d+1$
  - 矩阵秩的含义：包含线性独立的列（或行）的个数
- 由于$\mathbb{X}^T\mathbb{X}$和$\mathbb{X}$的秩相等，上述条件等价于$\mathbb{X}$的秩为$d+1$
- 因此$\hat{\boldsymbol{\theta}}$有唯一解的条件是，**$d$个输入变量彼此线性独立！**

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归（Multiple Linear Regression, MLR）
  - 在简单（一元）线性回归SLR模型基础上添加**更多的独立变量**

- 思考：
  - 上面为什么强调独立变量？
  - 给定MSE损失函数，SLR模型有唯一解，MLR有唯一解吗？
  - 如果希望MLR满足在MSE损失函数下有唯一解的条件是什么？

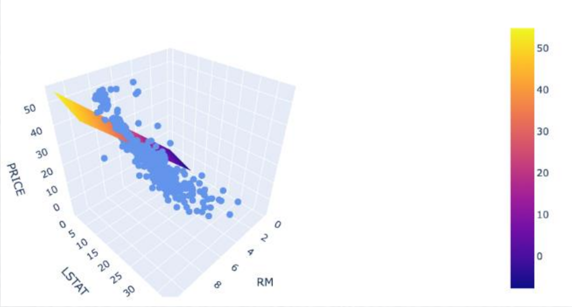
> 因此$\hat{\boldsymbol{\theta}}$有唯一解的条件是，***d*个输入变量彼此线性独立！**

回归：多元线性回归（解析解与梯度下降算法）

- 建立MLR模型
- 利用sklearn库的linear_model包，得到模型

```python
import sklearn.linear_model as lm

model = lm.LinearRegression(fit_intercept = True)
model.fit(boston_df[['RM', 'LSTAT']], boston_df['PRICE']);
```

$$\mathrm{PRICE} = -1.36 + 5.09 * \mathrm{RM} + (-0.64) * \mathrm{LSTAT}$$

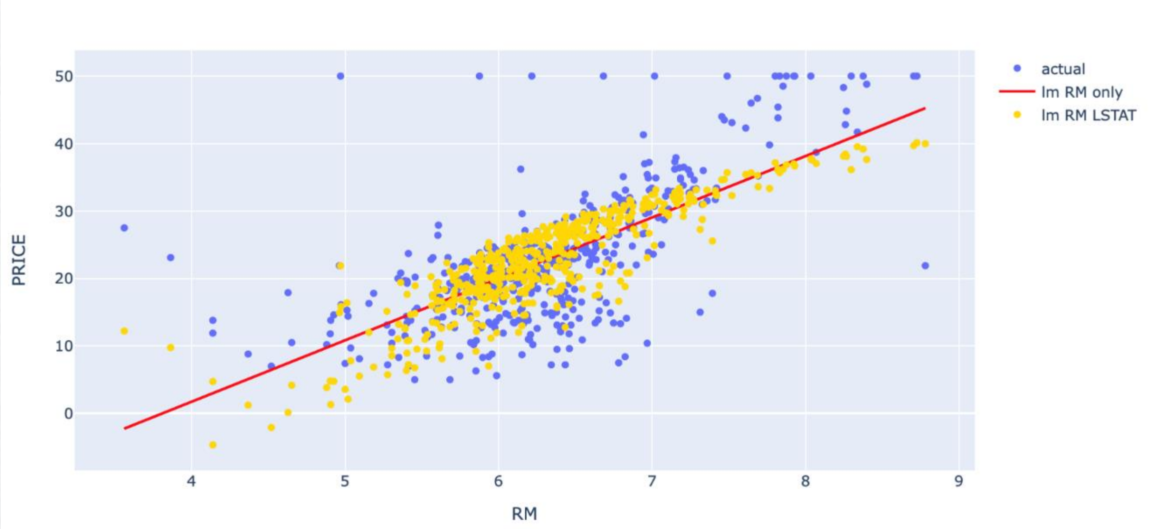


给定两个输入变量，如RM和LSTAT，线性回归计算的结果是三维中的一个平面

名称	类型	大小	修改日期
04SLR_MLR_boston_house_price.ipynb	IPYNB 文件	5,313 KB	2021/10/16 14:15

回归：多元线性回归（解析解与梯度下降算法）

- 将求得的MLR模型可视化在二维平面上



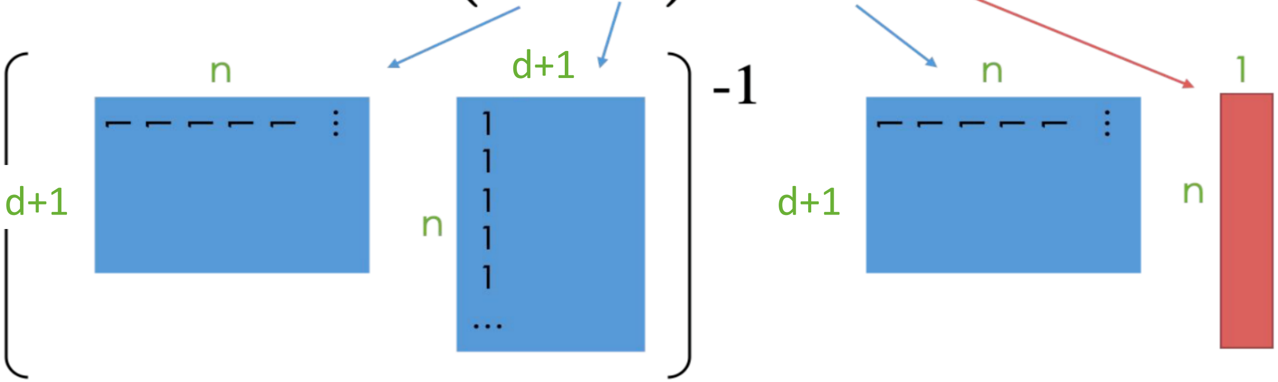
名称	类型	大小	修改日期
04SLR_MLR_boston_house_price.ipynb	IPYNB 文件	5,313 KB	2021/10/16 14:15

回归：多元线性回归（解析解与梯度下降算法）

回归：多元线性回归（解析解与梯度下降算法）

- 如何求解最优参数估计？
- 方法1：计算解析解

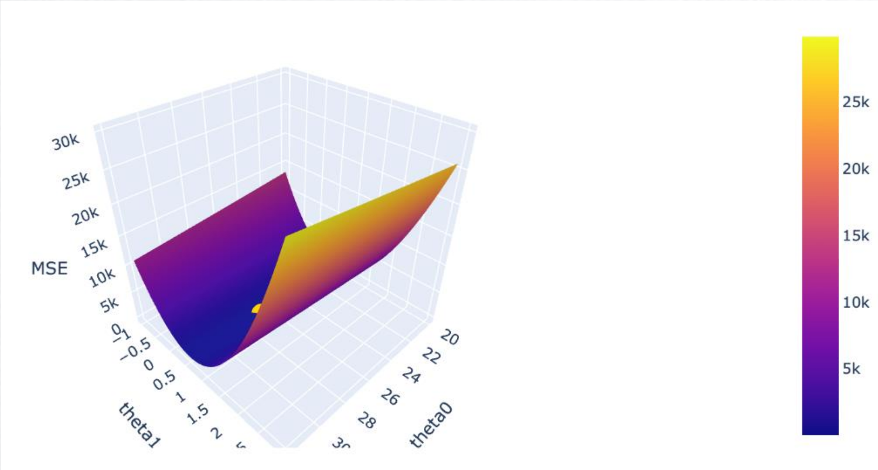
$$\hat{\theta} = \left(\mathbb{X}^T\mathbb{X}\right)^{-1}\mathbb{X}^T\mathbb{Y}$$



时间复杂度高！

回归：多元线性回归（解析解与梯度下降算法）

- 如何求解最优参数估计?
- 方法2：暴力搜索方法，枚举可能的参数值θ，计算MSE



枚举复杂度高！

回归：多元线性回归（解析解与梯度下降算法）

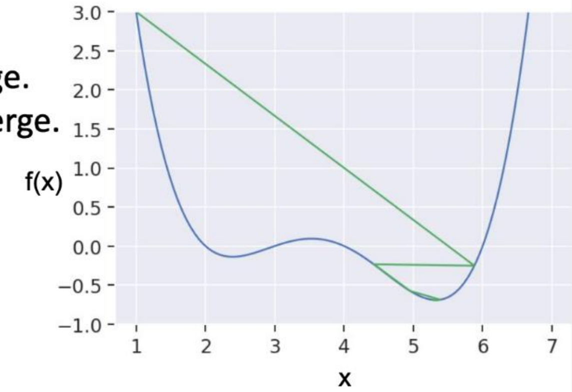
- 如何求解最优参数估计?
- 方法3：梯度下降法（Gradient Decent, GD）

The gradient descent algorithm is shown below:

- alpha is known as the “learning rate”.
  - Too large and algorithm fails to converge.
  - Too small and it takes too long to converge.

$$x^{(t+1)} = x^{(t)} - \alpha \frac{d}{dx} f(x)$$

```python
def gradient_descent(df, initial_guess, alpha, n):
    guesses = [initial_guess]
    guess = initial_guess
    while len(guesses) < n:
        guess = guess - alpha * df(guess)
        guesses.append(guess)
    return np.array(guesses)
```



接下来探讨梯度下降法求解

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解

$$\hat{y}^{\mathrm{i}} = \mathrm{h}_\theta(x^i) = \begin{bmatrix} 1 & x_1^i & x_2^i & \dots & x_d^i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \vdots \\ \theta_d \end{bmatrix}$$

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
- 目标函数
  - 注意：这里把样本数量记为m
  - 现在有m个样本，每个样本都有误差
  - 均方差：作为损失函数

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

分母里的2，是为了求导以后，销项容易

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 目标函数对θ_i求编导

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

$$\frac{\partial}{\partial\ \theta_0}J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\cdot x_0^{(i)}$$

$$\frac{\partial}{\partial\ \theta_1}J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\cdot x_1^{(i)}$$

$$\frac{\partial}{\partial\ \theta_2}J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\cdot x_2^{(i)}$$

$$\cdots$$

$$\hat{y}^{\mathrm{i}}=\begin{bmatrix}1 & x_1^i & x_2^i & \dots & x_d^i\end{bmatrix}\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \vdots\\ \theta_d\end{bmatrix}$$

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 目标函数对θ_i求编导
  - 梯度下降算法

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

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

- 采用向量运算，一次计算所有参数的梯度
- 后文介绍

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

名称	类型	大小	修改日期
01MLR_multivariate_housing_prices_in_portlans_oregon.py	Python File	7 KB	2021/10/15 16:09
multivariate_housing_prices_in_portlans_oregon.csv	Microsoft Excel...	1 KB	2021/10/15 16:00

参考文献https://satishgunjal.com/multivariate_lr/

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('./multivariate_housing_prices_in_portlans_oregon.csv')
print(df.head() )

X = df.values[:, 0:2]  # get input values from first two columns
y = df.values[:, 2]  # get output values from last coulmn
m = len(y) # Number of training examples

print('Total no of training examples (m) = %s \n' %(m))

```

- Import 库/模块/类
- 读取文件
- 显示信息

```
   size(in square feet)   number of bedrooms   price
0                  2104                    3  399900
1                  1600                    3  329900
2                  2400                    3  369000
3                  1416                    2  232000
4                  3000                    4  539900
Total no of training examples (m) = 47
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法
  - 运行代码、分析代码

- X的规范化

```
mu=  [2000.68085106    3.17021277]
sigma=  [7.94702354e+02 7.60981887e-01]
X_norm=  [[ 0.13000987 -0.22367519]
 [-0.50418984 -0.22367519]
 [ 0.50247636 -0.22367519]
 [-0.73572306 -1.53776691]
 [ 1.25747602  1.09041654]]
mu_testing [3.77948264e-17 2.74603035e-16]
sigma_testing [1. 1.]
```

```python
def feature_normalize(X):
    """
    Normalizes the features(input variables) in X.

    Parameters
    ----------
    X : n dimensional array (matrix), shape (n_samples, n_features)
        Features(input varibale) to be normalized.

    Returns
    -------
    X_norm : n dimensional array (matrix), shape (n_samples, n_features)
        A normalized version of X.
    mu : n dimensional array (matrix), shape (n_features,)
        The mean value.
    sigma : n dimensional array (matrix), shape (n_features,)
        The standard deviation.
    """
    #Note here we need mean of indivdual column here, hence axis = 0
    mu = np.mean(X, axis = 0)
    # Notice the parameter ddof (Delta Degrees of Freedom)  value is 1
    sigma = np.std(X, axis= 0, ddof = 1)  # Standard deviation (can also use range)
    X_norm = (X - mu)/sigma
    return X_norm, mu, sigma

X, mu, sigma = feature_normalize(X)

print('mu= ', mu)
print('sigma= ', sigma)
print('X_norm= ', X[:5])

mu_testing = np.mean(X, axis = 0) # mean
print( "mu_testing", mu_testing)
sigma_testing = np.std(X, axis = 0, ddof = 1) # mean
print( "sigma_testing", sigma_testing)
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

- 加上分量1

```python
# Lets use hstack() function from numpy to add column of ones to X feature
# This will be our final X matrix (feature matrix)
X = np.hstack((np.ones((m,1)), X))
print("X[:5]", X[:5])
```

```
X[:5] [[ 1.          0.13000987 -0.22367519]
 [ 1.         -0.50418984 -0.22367519]
 [ 1.          0.50247636 -0.22367519]
 [ 1.         -0.73572306 -1.53776691]
 [ 1.          1.25747602  1.09041654]]
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

- 损失函数

```python
def compute_cost(X, y, theta):
    """
    Compute the cost of a particular choice of theta for linear regression.

    Input Parameters
    ----------------
    X : 2D array where each row represent the training example and each column represent the
        m= number of training examples
        n= number of features (including X_0 column of ones)
    y : 1D array of labels/target value for each traing example. dimension(1 x m)

    theta : 1D array of fitting parameters or weights. Dimension (1 x n)

    Output Parameters
    -----------------
    J : Scalar value.
    """
    predictions = X.dot(theta)
    #print('predictions= ', predictions[:5])
    errors = np.subtract(predictions, y)
    #print('errors= ', errors[:5])
    sqrErrors = np.square(errors)
    #print('sqrErrors= ', sqrErrors[:5])
    #J = 1 / (2 * m) * np.sum(sqrErrors)
    # OR
    # We can merge 'square' and 'sum' into one by taking the transpose of matrix 'errors' and
    # If your confuse about this try to do this with few values for better understanding
    J = 1/(2 * m) * errors.T.dot(errors)

    return J
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

- 梯度下降函数

- 这里采用向量运算，一次计算所有参数的梯度
- 后文介绍梯度向量的计算方法

```
def gradient_descent(X, y, theta, alpha, iterations):
  """
  Compute cost for linear regression.

  Input Parameters
  ----------------
  X : 2D array where each row represent the training example and each column represent the
      m= number of training examples
      n= number of features (including X_0 column of ones)
  y : 1D array of labels/target value for each traing example. dimension(m x 1)
  theta : 1D array of fitting parameters or weights. Dimension (1 x n)
  alpha : Learning rate. Scalar value
  iterations: No of iterations. Scalar value.

  Output Parameters
  -----------------
  theta : Final Value. 1D array of fitting parameters or weights. Dimension (1 x n)
  cost_history: Conatins value of cost for each iteration. 1D array. Dimansion(m x 1)
  """
  cost_history = np.zeros(iterations)

  for i in range(iterations):
    predictions = X.dot(theta)
    #print('predictions= ', predictions[:5])
    errors = np.subtract(predictions, y)
    #print('errors= ', errors[:5])
    sum_delta = (alpha / m) * X.transpose().dot(errors);
    #print('sum_delta= ', sum_delta[:5])
    theta = theta - sum_delta;

    cost_history[i] = compute_cost(X, y, theta)

  return theta, cost_history
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

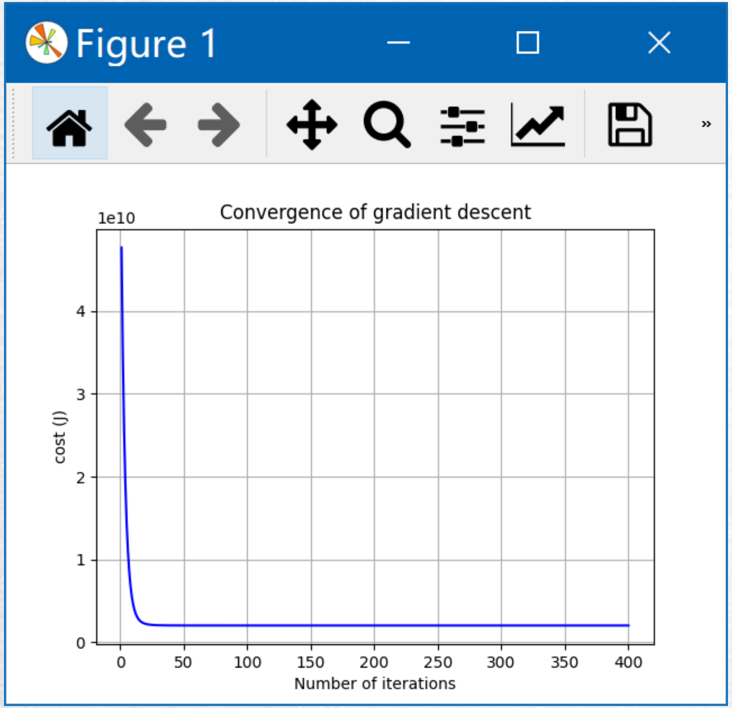
- 训练与预测

```python
# We need theta parameter for every input variable. since we have three input variable
theta = np.zeros(3)
iterations = 400;
alpha = 0.15;

theta, cost_history = gradient_descent(X, y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('First 5 values from cost_history =', cost_history[:5])
print('Last 5 values from cost_history =', cost_history[-5 :])



normalize_test_data = ((np.array([1650, 3]) - mu) / sigma)
normalize_test_data = np.hstack((np.ones(1), normalize_test_data))
price = normalize_test_data.dot(theta)
print('Predicted price of a 1650 sq-ft, 3 br house:', price)
```

回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  - 运行代码、分析代码

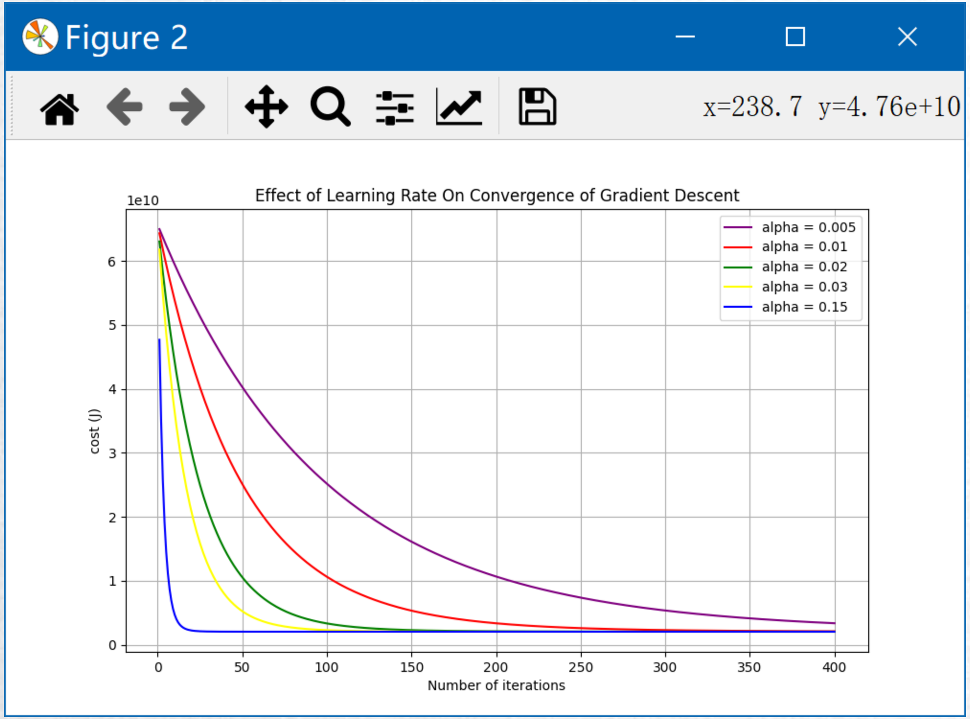
- 损失函数的变化



回归：多元线性回归（解析解与梯度下降算法）

- 多元线性回归：梯度下降算法求解
  – 运行代码、分析代码

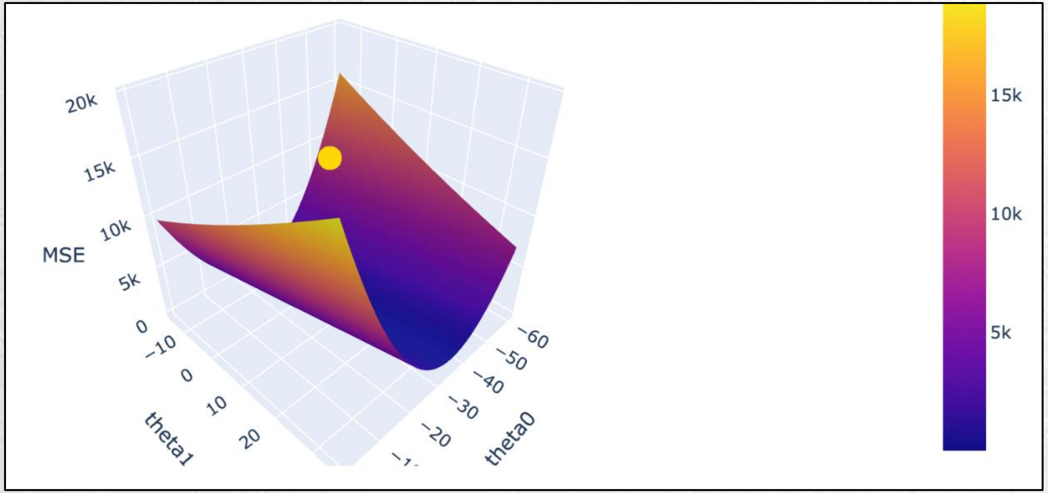
- 不同学习率的影响



回归：多元线性回归（解析解与梯度下降算法）

回归：多元线性回归（解析解与梯度下降算法）

- 给定线性回归的矩阵形式，我们有
- 损失函数为$R(\boldsymbol{\theta}) = \frac{1}{2n}\|\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}\|_2^2$



比如，Boston house price数据集考虑变量RM和截距的情况

回归：多元线性回归（解析解与梯度下降算法）

- 给定线性回归的矩阵形式，我们有
  - $R(\boldsymbol{\theta}) = \frac{1}{2n}\|\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}\|_2^2 = \frac{1}{2n}(\mathbb{Y} - \mathbb{X}\boldsymbol{\theta})^T\,(\mathbb{Y} - \mathbb{X}\boldsymbol{\theta})$
  - $= \frac{1}{2n}\,(\mathbb{Y}^T - \boldsymbol{\theta}^{\mathrm{T}}\,\mathbb{X}^{\boldsymbol{T}})(\mathbb{Y} - \mathbb{X}\boldsymbol{\theta})$
- 对$\boldsymbol{\theta}$求倒导数，得到梯度向量 $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$
  - $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$

- 梯度向量，应该是一个(d+1) *1的向量
- 如何求导？

回归：多元线性回归（解析解与梯度下降算法）

- 给定线性回归的矩阵形式，我们有
  - $R(\boldsymbol{\theta}) = \frac{1}{2n}\|\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}\|_2^2 = \frac{1}{2n}(\mathbb{Y} - \mathbb{X}\boldsymbol{\theta})^T\ (\mathbb{Y} - \mathbb{X}\boldsymbol{\theta})$
  - $= \frac{1}{2n}\left(\mathbb{Y}^T - \boldsymbol{\theta}^{\mathrm{T}}\ \mathbb{X}^{\boldsymbol{T}}\right)(\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}) = \frac{1}{2n}(\mathbb{Y}^T\mathbb{Y} - \mathbb{Y}^T\mathbb{X}\boldsymbol{\theta} - \boldsymbol{\theta}^{\mathrm{T}}\ \mathbb{X}^{\boldsymbol{T}}\mathbb{Y} + \boldsymbol{\theta}^{\mathrm{T}}\ \mathbb{X}^{\boldsymbol{T}}\mathbb{X}\boldsymbol{\theta})$

对$\boldsymbol{\theta}$求导数

分别对两个$\boldsymbol{\theta}$求导

$$= \frac{1}{2n}(0 - \mathbb{X}^T\mathbb{Y} - \mathbb{X}^T\mathbb{Y} + \mathbb{X}^T\mathbb{X}\boldsymbol{\theta} + \mathbb{X}^T\mathbb{X}\boldsymbol{\theta})$$

- 对$\boldsymbol{\theta}$求倒导数，具体如右侧所示
  - 最后得到梯度向量 $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$
  - $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$

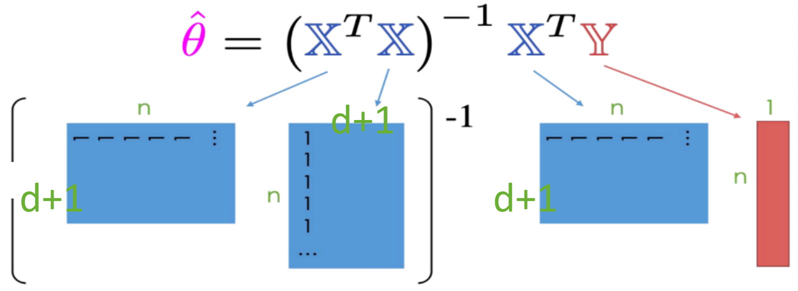
各个成分维度 (d+1)*n * (n*(d+1) *(d+1)*1 – n*1)

- 梯度向量，应该是一个(d+1) *1的向量

补充材料初步介绍矩阵求导

回归：多元线性回归（解析解与梯度下降算法）

- 回顾解析解
  - 梯度为$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$
  - 令梯度为0，可以计算解析解
  - $\mathbb{X}^T\mathbb{X}\boldsymbol{\theta} - \mathbb{X}^T\mathbb{Y} = 0$
  - $\mathbb{X}^T\mathbb{X}\boldsymbol{\theta} = \mathbb{X}^T\mathbb{Y}$
  - $\widehat{\boldsymbol{\theta}} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$
  - 对照前文
  - 和利用张成空间推导的式子是一样的



参考文献
https://zhuanlan.zhihu.com/p/91095053

回顾完成，继续梯度下降的矩阵形式

回归：多元线性回归（解析解与梯度下降算法）

- 梯度下降算法GD

$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...)

For τ from 0 to convergence:

$$\vec{\theta}^{(t+1)} = \vec{\theta}^{(t)} - \alpha \boxed{\nabla_{\vec{\theta}} L(\vec{\theta}, \mathbb{X}, \vec{y})}$$

- α is the learning rate
- Converges when gradient is ≈ 0 (or we run out of patience)

思考：上面哪一步的计算最耗时，为什么？

回归：多元线性回归（解析解与梯度下降算法）

- 课堂练习
  - 给定一组训练数据：(2,4), (5,1), (8,9)
  - 请计算$\boldsymbol{\theta} = [\frac{1}{2}, \frac{5}{6}]$时的梯度向量$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$

x	y
2	4
5	1
8	9

X	Y
1 2	4
1 5	1
1 8	9

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$$

$$\frac{1}{3}\begin{bmatrix}1 & 1 & 1\\ 2 & 5 & 8\end{bmatrix}\left(\begin{bmatrix}1 & 2\\ 1 & 5\\ 1 & 8\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\ \boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}4\\ 1\\ 9\end{bmatrix}\right)$$

$\boldsymbol{\theta} = [\frac{1}{2}, \frac{5}{6}]$代入

使用梯度下降求解多元线性回归模型

- Gradient Descent
  - 每次迭代，全部样本，
  - 计算梯度
  - 参数更新
    - 实际应用中n比较大，比如500

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n} \mathbb{X}^T (\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$$

x	y
2	4
5	1
8	9

$$\frac{1}{3}\begin{bmatrix}1 & 1 & 1\\ 2 & 5 & 8\end{bmatrix}\left(\begin{bmatrix}1 & 2\\ 1 & 5\\ 1 & 8\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\ \boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}4\\ 1\\ 9\end{bmatrix}\right)$$

使用梯度下降求解多元线性回归模型

- Stochastic Gradient Descent
  - 每次迭代，1个样本，
  - 计算梯度
  - 参数更新

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n} \mathbb{X}^T (\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$$

x	y
2	4
5	1
8	9

$$\frac{1}{3}\begin{bmatrix}1\\2\end{bmatrix}\left(\begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\\boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}4\end{bmatrix}\right)$$

$$\frac{1}{3}\begin{bmatrix}1\\5\end{bmatrix}\left(\begin{bmatrix}1 & 5\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\\boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}1\end{bmatrix}\right)$$

$$\frac{1}{3}\begin{bmatrix}1\\8\end{bmatrix}\left(\begin{bmatrix}1 & 8\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\\boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}9\end{bmatrix}\right)$$

使用梯度下降求解多元线性回归模型

- Min bath Gradient Descent
  - 小批量样本，
  - 计算梯度
  - 参数更新
    - Batch size =2
    - 最后一批只有1个样本

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$$

$$\frac{1}{3}\begin{bmatrix}1 & 1\\ 2 & 5\end{bmatrix}\left(\begin{bmatrix}1 & 2\\ 1 & 5\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\ \boldsymbol{\theta}_1\end{bmatrix} - \begin{bmatrix}4\\ 1\end{bmatrix}\right)$$

$$\frac{1}{3}\begin{bmatrix}1\\ 8\end{bmatrix}\left(\begin{bmatrix}1 & 8\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_0\\ \boldsymbol{\theta}_1\end{bmatrix} - [9]\right)$$

x	y
2	4
5	1
8	9

回归：多元线性回归（解析解与梯度下降算法）

- Min bath Gradient Descent
  - Shuffle: 将数据做一次随机化排序
  - $(\mathbb{X}_b, \mathbb{Y}_b)$: 大小为B(batch size)的一个mini-batch

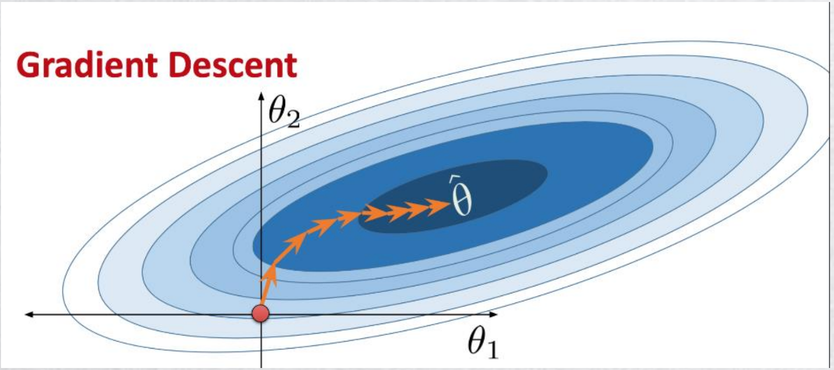
一个epoch

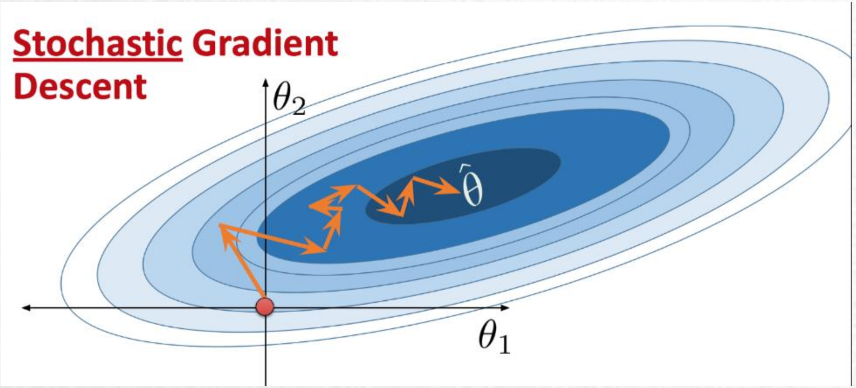
1. Initialize $\boldsymbol{\theta}^{(0)} \leftarrow$ Zero, Random, etc.
2. For t from 0 to convergence:
3. Shuffle $(\mathbb{X}, \mathbb{Y})$
4. For each B-sized minibatch $(\mathbb{X}_b, \mathbb{Y}_b)$ in $(\mathbb{X}, \mathbb{Y})$
5. $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \alpha$ Gradient $(\mathbb{X}_b, \mathbb{Y}_b, \boldsymbol{\theta}^{(t)})$
6. End For
7. End For

主流机器学习（深度学习）的参数优化方法

回归：多元线性回归（解析解与梯度下降算法）

- GD（每次迭代全量样本一次更新参数）
- 与SGD（每次迭代一个样本，每个样本更新参数）的比较





回归：多元线性回归（解析解与梯度下降算法）

- 梯度下降算法
- 示例代码分析

$$\frac{d(mse_{loss})}{d\theta} = gradient\_mse$$

```python
def mse_loss_lr(theta_vec, x, y):
    res_squred = (y - X@theta_vec)**2
    return 1 / 2 * np.mean(res_squred)
```

```python
def gradient_mse_lr (x, y, theta_vec):
    res = x@theta_vec - y
    grad = x.T@res / y.shape[0]
    return grad
```

$$R(\boldsymbol{\theta}) = \frac{1}{2n}\|\mathbb{Y} - \mathbb{X}\boldsymbol{\theta}\|_2^2$$

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \frac{1}{n}\mathbb{X}^T(\mathbb{X}\boldsymbol{\theta} - \mathbb{Y})$$

名称	类型	大小	修改日期
05SGD_simple_boston_house_price.ipynb	IPYNB 文件	5,392 KB	2021/10/16 15:37

回归：多元线性回归（解析解与梯度下降算法）

- 梯度下降算法
- 示例代码分析

- 几个关键点：
  - 添加了tolerance参数
  - 添加了异常判断
  - 梯度函数的输入和输出均为向量

```python
def gradient_descent(
    gradient, x, y, start, learn_rate=0.1, n_iter=50, tolerance=1e-06,
    dtype="float64"
):
    # Checking if the gradient is callable
    if not callable(gradient):
        raise TypeError("'gradient' must be callable")

    # Setting up the data type for NumPy arrays
    dtype_ = np.dtype(dtype)

    # Converting x and y to NumPy arrays
    x, y = np.array(x, dtype=dtype_), np.array(y, dtype=dtype_)
    if x.shape[0] != y.shape[0]:
        raise ValueError("'x' and 'y' lengths do not match")

    # Initializing the values of the variables
    vector = np.array(start, dtype=dtype_)
    # Setting up and checking the learning rate
    learn_rate = np.array(learn_rate, dtype=dtype_).T
    if np.any(learn_rate <= 0):
        raise ValueError("'learn_rate' must be greater than zero")

    # Setting up and checking the maximal number of iterations
    n_iter = int(n_iter)
    if n_iter <= 0:
        raise ValueError("'n_iter' must be greater than zero")

    # Setting up and checking the tolerance
    tolerance = np.array(tolerance, dtype=dtype_)
    if np.any(tolerance <= 0):
        raise ValueError("'tolerance' must be greater than zero")

    # Performing the gradient descent loop
    for _ in range(n_iter):
        # Recalculating the difference
        diff = -learn_rate * np.array(gradient(x, y, vector), dtype_)

        # Checking if the absolute difference is small enough
        if np.all(np.abs(diff) <= tolerance):
            break
        # Updating the values of the variables
        vector = vector + diff
        #print(vector)

    return vector if vector.shape else vector.item()
```

名称	类型	大小	修改日期
05SGD_simple_boston_house_price.ipynb	IPYNB 文件	5,392 KB	2021/10/16 15:37

回归：多元线性回归（解析解与梯度下降算法）

回归：多元线性回归（解析解与梯度下降算法）

- 一元/多元线性回归的评价：如何评判SLR/MLR两个模型的优劣
- 基本想法：度量观测值y与预测值$\hat{y}$之间的差异
- **均方根误差**（Root Mean Squared Error）

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  - 均方根误差RMSE是MSE损失函数的平方根
  - 均方根误差RMSE与观测值y与预测值$\hat{y}$的量纲相同
  - 均方根误差RMSE越小，说明模型越准确

回归：多元线性回归（解析解与梯度下降算法）

- 一元/多元线性回归的评价：如何评判SLR/MLR两个模型的优劣
- **均方根误差**
- **代码分析：**
  - 请在notebook中编写rmse函数计算均方根误差

```python
def rmse(y, yhat):
    # Write down your code here
    return 0
```

  - 请计算以下两个模型的均方根误差，并比较大小

$$\text{PRICE} = -34.67 + 9.1 \,*\, \text{RM}$$

$$\text{PRICE} = -1.36 + 5.09 * \text{RM} + (-0.64) * \text{LSTAT}$$

名称	类型	大小	修改日期
04SLR_MLR_boston_house_price.ipynb	IPYNB 文件	5,313 KB	2021/10/16 14:15

回归：多元线性回归（解析解与梯度下降算法）

- 拟合优度（Multiple R²）度量预测值$\hat{y}$对观测值y的**拟合程度**
- 拟合优度R²定义**$\hat{y}$与y皮尔森相关系数的平方**

$$R^2 = [r(y, \hat{y})]^2$$

  - 拟合优度的取值范围在[0, 1]，越高说明模型准确性越好
  - 拟合优度的含义：模型在多大程度上解释了观测值的变化

针对包含截距的线性回归模型，拟合优度也可如下计算

$$R^2 = \frac{\text{variance of fitted values}}{\text{variance of true values}} = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2}$$

回归：多元线性回归（解析解与梯度下降算法）

- 拟合优度（Multiple R^2）
- 代码分析：
  - 请在notebook中编写multiple-r-squared函数计算拟合优度

```python
def MultipleRSqured(y, yhat):
    # Write down your code here
    return 0
```

  - 请计算以下两个模型的拟合优度，并比较大小

$$\text{PRICE} = -34.67 + 9.1 * \text{RM}$$

$$\text{PRICE} = -1.36 + 5.09 * \text{RM} + (-0.64) * \text{LSTAT}$$

名称	类型	大小	修改日期
04SLR_MLR_boston_house_price.ipynb	IPYNB 文件	5,313 KB	2021/10/16 14:15