華東師範大學
EAST CHINA NORMAL UNIVERSITY

回归

数据学院
刘小平
2023.9

回归

主要内容

- 回顾：分类与回归
- 线性回归
  - 一元线性回归
  - 多元线性回归
- 逻辑回归

回顾：分类与回归

- 机器学习中的有监督学习包括：
  - 回归——输出是连续型变量，如：预测明天的温度
  - 分类——输出是离散型变量，如：预测明天天气是晴、阴还是雨

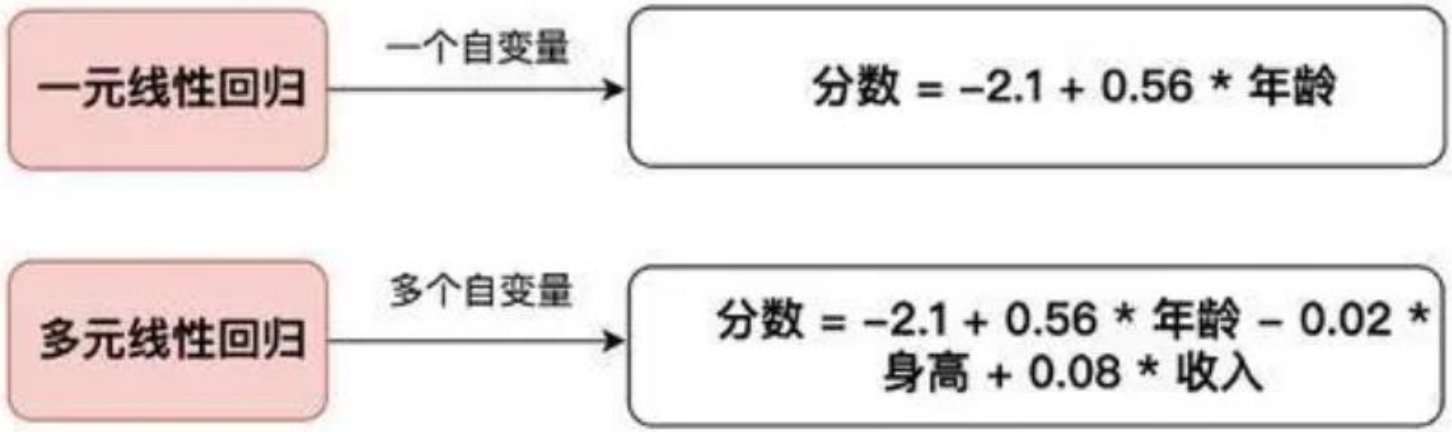
- 回归的定义：
  - 寻找自变量x和因变量y之间的关系，通常称为拟合。

线性回归

- 线性回归：用线性的模型去拟合x,y之间的关系。（注意：机器学习中的线性回归模型并不是只能拟合线性的关系，它也能拟合非线性关系。）
- 什么是线性？
  - 二维空间：某一条直线
  - 三维空间：某一个平面
  - N维空间：某一个超平面
    - 公式：$$h_\theta(x)=\sum_{j=1}^{m}\theta_j x_j+\theta_0$$
    - m ——特征维度
    - θ_j和θ_0 ——超平面系数
    - 注意：超平面所在空间为 $m+1$ 维
  - 以上就是线性回归的模型，这些系数就是模型的待估参数。

线性回归

线性回归模型分为一元线性回归与多元线性回归，区别在于自变量的个数。

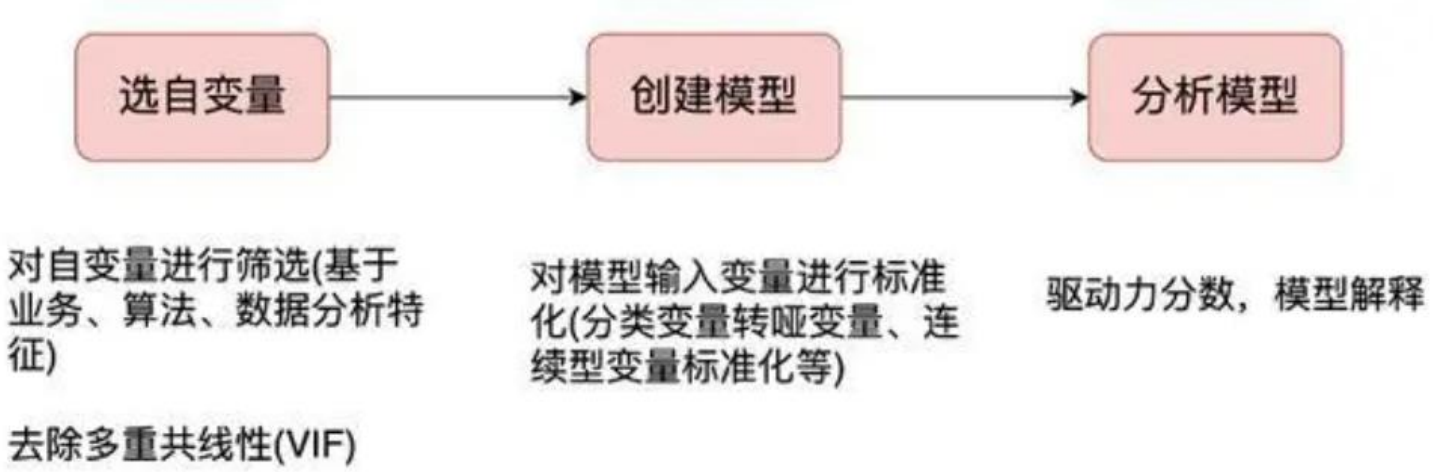


线性回归的应用场景

- 预测：自变量与因变量呈“线性”(?)关系的预测。
  - 例如股价预测、营收预测、广告效果预测、销售业绩预测等
  - 餐厅根据媒体的营业数据（包括菜谱价格、就餐人数、预订人数、特价菜折扣等）预测就餐规模或营业额；
  - 网站根据访问的历史数据（包括新用户的注册量、老用户的活跃度、网站内容的更新频率等）预测用户的支付转化率；
  - 医院根据患者的病历数据（如体检指标、药物复用情况、平时的饮食习惯等）预测某种疾病发生的概率。
- 驱动力分析：某个因变量指标受多个因素所影响，分析不同因素对因变量驱动力的强弱（驱动力指相关性，不是因果性）；
  - 共享单车案例展示

共享单车案例分享1

- 以共享单车服务满意度为案例进行模型实战，想要去分析不同的特征对客户满意度的影响，模型过程如下：



共享单车案例分享2

- 1、读数据：

```
# 共享单车满意度分析（使用线性回归进行驱动力分析）
#一、数据获取
#导入库
import numpy as np
import pandas as pd
df=pd.read_csv("Shared Bike Sample Data - ML.csv",encoding="utf-8")
df.head()
```

2]:

	ID	城区	分数	组别	推荐者	年龄	车龄	月收入	掷硬币结果
0	1	西城区	6	对照组	False	25	15	4688	0
1	2	海淀区	3	对照组	False	21	11	6564	0
2	3	东城区	1	对照组	False	20	10	3776	1
3	4	朝阳区	9	对照组	True	37	26	4688	0
4	5	西城区	1	对照组	False	19	9	6642	1

共享单车案例分享3

- 2、切分因变量和自变量、分类变量转换哑变量；
- 3、使用VIF去除多重共线性（多重共线性判断标准通常是：VIF>10）；

```
#二、数据处理
#用VIF值观察是否存在多重共线性，如果有则消除
from statsmodels.stats.outliers_influence import variance_inflation_factor
X=df.loc[:,["城区","年龄"]]
Y=df.loc[:,["分数"]]
X=pd.get_dummies(data=X,columns=["城区"],drop_first=True)  #将城区特征变成哑变量，为输入模型做准备
vif=pd.DataFrame()
vif["features"]=X.columns
vif["VIF Factor"]=[variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
vif
#各特征VIF值均小于10，说明不存在多重共线性
```

]:

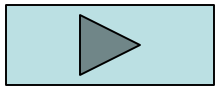
	features	VIF Factor
0	年龄	3.332785
1	城区_朝阳区	1.834868
2	城区_海淀区	1.755736
3	城区_西城区	1.742181

各特征VIF均小于10，说明不存在多重共线性。

哑变量

- 哑变量（Dummy Variable），又称为虚拟变量、虚设变量或名义变量，从名称上看就知道，它是人为虚设的变量，通常取值为0或1，来反映某个变量的不同属性。对于有n个分类属性的自变量，通常需要选取1个分类作为参照，因此可以产生n−1个哑变量。
- 以职业因素为例，假设职业分为学生、农民、工人、公务员、其他共5个分类

	X1	X2	X3	X4
学生	1	0	0	0
农民	0	1	0	0
工人	0	0	1	0
公务员	0	0	0	1
其他	0	0	0	0



什么情况下需要设置哑变量？

- 1．对于无序多分类变量，引入模型时需要转化为哑变量；
- 举一个例子，如血型，一般分为A、B、O、AB四个类型，为无序多分类变量，通常情况下在录入数据的时候，为了使数据量化，我们常会将其赋值为1、2、3、4。从数字的角度来看，赋值为1、2、3、4后，它们是具有从小到大一定的顺序关系的，而实际上，四种血型之间并没有这种大小关系存在，它们之间应该是相互平等独立的关系。如果按照1、2、3、4赋值并带入到回归模型中是不合理的，此时我们就需要将其转化为哑变量。
- 2．对于有序多分类变量，引入模型时需要酌情考虑；
- 例如疾病的严重程度，一般分为轻、中、重度，可认为是有序多分类变量，通常情况下我们也常会将其赋值为1、2、3（等距）或1、2、4（等比）等形式，通过由小到大的数字关系，来体现疾病严重程度之间一定的等级关系。需要注意的是，一旦赋值为上述等距或等比的数值形式，这在某种程度上是认为疾病的严重程度也呈现类似的等距或等比的关系。而事实上由于疾病在临床上的复杂性，不同的严重程度之间并非是严格的等距或等比关系，因此再赋值为上述形式就显得不太合理，此时可以将其转化为哑变量进行量化。
- 3．对于连续性变量，进行变量转化时可以考虑设定为哑变量。
- 例如年龄，以连续性变量带入模型时，其解释为年龄每增加一岁时对于因变量的影响。但往往年龄增加一岁，其效应是很微弱的，并没有太大的实际意义。此时，我们可以将年龄这个连续性变量进行离散化，按照10岁一个年龄段进行划分，如0—10、11—20、21—30、31—40等等，将每一组赋值为1、2、3、4，此时构建模型的回归系数就可以解释为年龄每增加10岁时对因变量的影响。以上赋值方式是基于一个前提，即年龄与因变量之间存在着一定的线性关系。但有时候可能会出现以下情况，例如在年龄段较低和较高的人群中，某种疾病的死亡率较高，而在中青年人群中，死亡率却相对较低，年龄和死亡结局之间呈现一个U字型的关系，此时再将年龄段赋值为1、2、3、4就显得不太合理了。

共享单车案例分享4

4. 计算调整R方：

R2

```
# 将训练好的模型，用来预测原有的（训练）数据
y_pred = lm.predict(X)
# R2值 - 查看模型对数据的拟合情况好坏
from sklearn.metrics import r2_score
r2 = r2_score(y_pred, y)
'R2值为：{}'.format(round(r2, 2))
```

```
'R2值为：0.86'
```

调整R2

```
# 调整r2
n = X.shape[0] # 行数
p = X.shape[1] # 列数
adj_r2 = 1 - (1-r2*r2)*(n-1)/(n-p-1)
'调整R2值为：{}'.format(round(adj_r2, 2))
```

```
'调整R2值为：0.72'
```

由于R2和调整R2的值都>0.7，属于良好的模型拟合

R方/调整R方值区间经验判断：

<0.3	非常弱的模型拟合
0.3—0.5	弱的模型拟合
0.5—0.7	适度的模型拟合
>0.7	较好的模型拟合

R方（一元线性回归）——因变量波动中，被模型拟合的百分比

R方=1 模型完美的拟合数据（100%）

R方>0.7 模型在一定程度较好的拟合数据（70%）

R方<0 拟合直线的趋势与真实因变量相反

调整R方（多元线性回归）——由于模型会因加入无价值的变量导致R方提升，对最终结果产生误导，因此我们引进数据量、自变量个数这两个条件，辅助调整R方的取值，称为调整R方。

调整R方值会因为自变量个数的增加而降低（惩罚），会因为新自变量带来的有价值信息而增加（奖励），可以帮助我们筛选出更多有价值的新自变量。

共享单车案例分享5

5. 数据标准化：使得不同自变量的线性系数，相互之间有可比性，不受它们取值范围影响。

```
#根据模型运行结果，依照四个城区对用户满意分数的影响程度做一个由强到弱的排序。
#数据标准化
from sklearn.preprocessing import StandardScaler
X_standard=StandardScaler().fit_transform(X)
X_standard=pd.DataFrame(data=X_standard,columns=list(X.columns))

X_standard.head()
```

	年龄	城区_东城区	城区_朝阳区	城区_海淀区	城区_西城区
0	-0.410621	-0.577350	-0.577350	-0.577350	1.732051
1	-1.103651	-0.577350	-0.577350	1.732051	-0.577350
2	-1.276909	1.732051	-0.577350	-0.577350	-0.577350
3	1.668471	-0.577350	1.732051	-0.577350	-0.577350
4	-1.450167	-0.577350	-0.577350	-0.577350	1.732051

标准化

1. 先计算得到原始样本各特征值的**均值**和**标准差**。
2. 然后将特征值映射到均值为 0，标准差为 1 的标准**正态分布**上。

$$x' = \frac{x - \mu}{\sigma}$$

均值

标准差

sklearn.preprocessing. StandardScaler 对象可完成标准化任务

共享单车案例分享6

6. 拟合模型，计算回归系数，分析驱动力因素：

```
lm1=LinearRegression()
lm1.fit(X_standard,Y)
coef_df1=pd.DataFrame(lm1.coef_[0],X_standard.columns,columns=["Coefficients"])
coef_df1
```

	Coefficients
年龄	2.669961
城区_东城区	-0.222990
城区_朝阳区	-0.186816
城区_海淀区	0.432013
城区_西城区	-0.022208

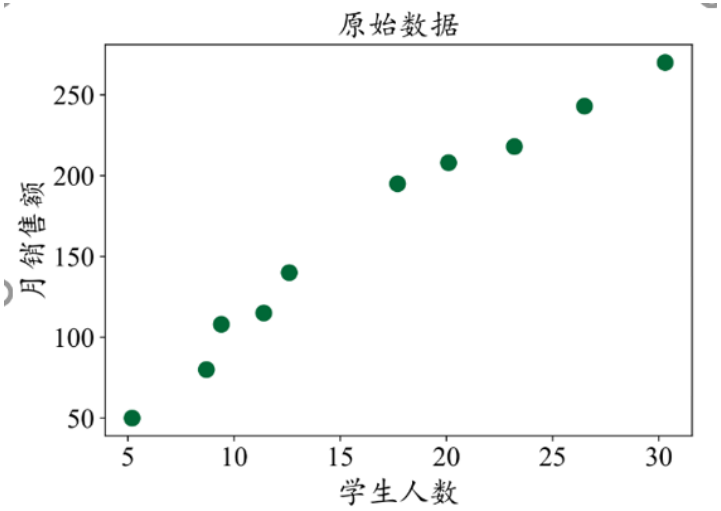
驱动力因素分析：
驱动力分数绝对值越大，表明因素对目标变量的影响力越大；反之越小，驱动力分数为负时，表明此因素对目标变量的影响为负向。

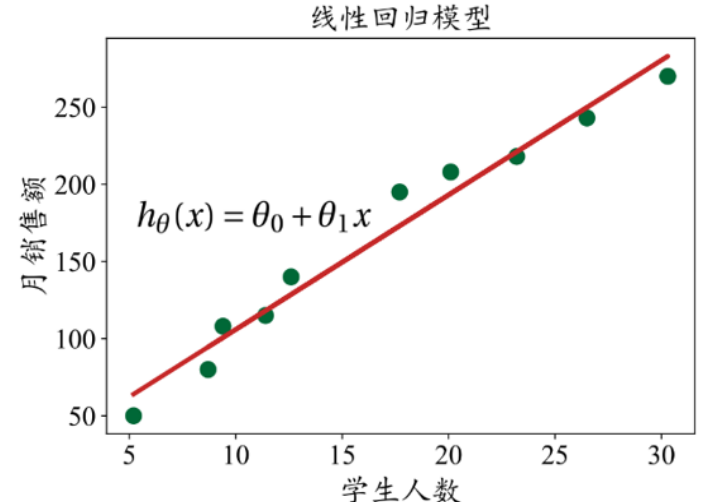
```
# 海淀区>西城区>朝阳区>东城区
```

一元线性回归（或单变量线性回归）举例

- 大学连锁便利店的管理人员认为每家便利店的月销售额是和所在学校的学生人数成正比的，现收集了10家已开连锁店的相关数据（如下表所示），希望能够建立模型并预测新开连锁店的月销售额情况。

代号	学生人数（千）	月销售额（万元）
1	5.2	50
2	20.1	208
3	12.6	140
4	26.5	243
5	17.7	195
6	8.7	80
7	11.4	115
8	23.2	218
9	30.3	270
10	9.4	108





一元线性回归（或单变量线性回归）举例

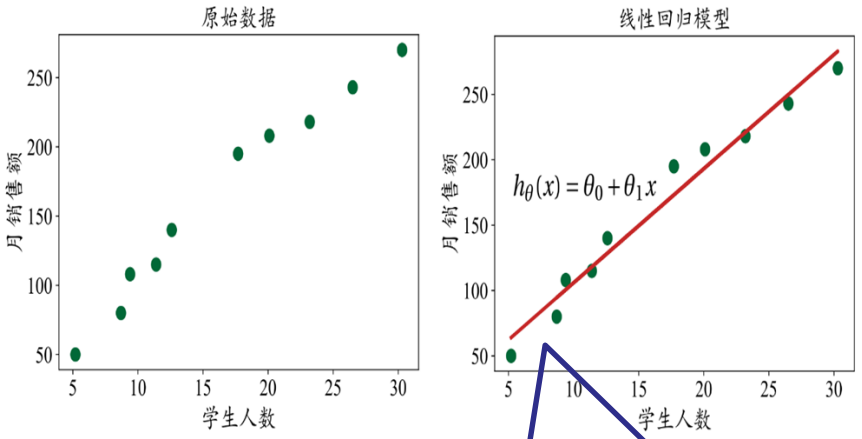


图5-12 大学连锁便利店的线性回归模型

例子中给出了10个点，如何确定一条直线，使得这条直线拟合效果好。？？？

示例1、2

- 学生人数：**自变量 x**
- 月销售额：**因变量 y**
- 可将 y 看作是 x 的一次函数

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

- (x,y) 各点在平面直角坐标系中为一条**直线**。θ_0 为截距或偏差，θ_1 为斜率。
- **找出 θ_0 和 θ_1 的过程：**

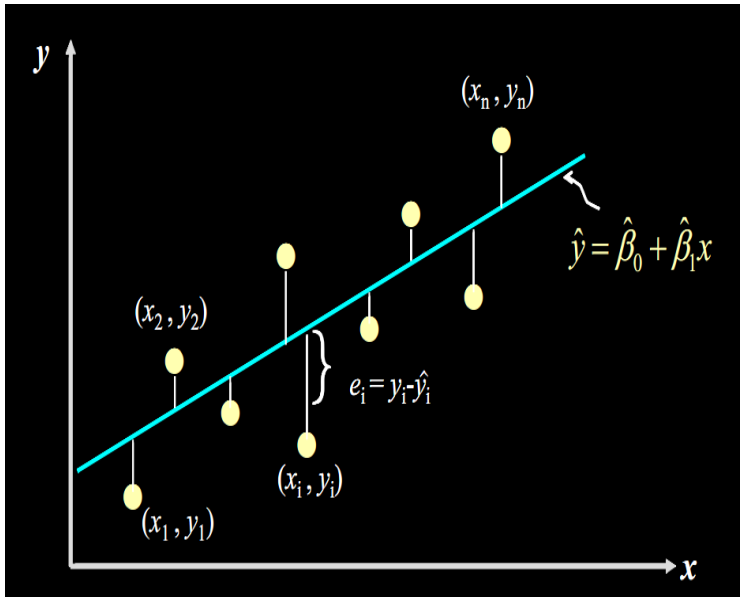
一元线性回归

线性回归

- 回归问题评价指标
  - 均方误差 $$MSE = \frac{1}{n_t}\sum_{i=1}^{n_t}(y_i - \hat{y}_i)^2$$
  - 均方根误差 $$RMSE = \sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}(y_i - \hat{y}_i)^2}$$
  - 平均绝对误差 $$MAE = \frac{1}{n_t}\sum_{i=1}^{n_t}|y_i - \hat{y}_i|$$
  - 百分比误差 $$MAPE = \frac{100}{n_t}\sum_{i=1}^{n_t}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

线性回归损失函数

损失函数又叫代价函数
下面给出的是残差平方和
（RSS）代价函数



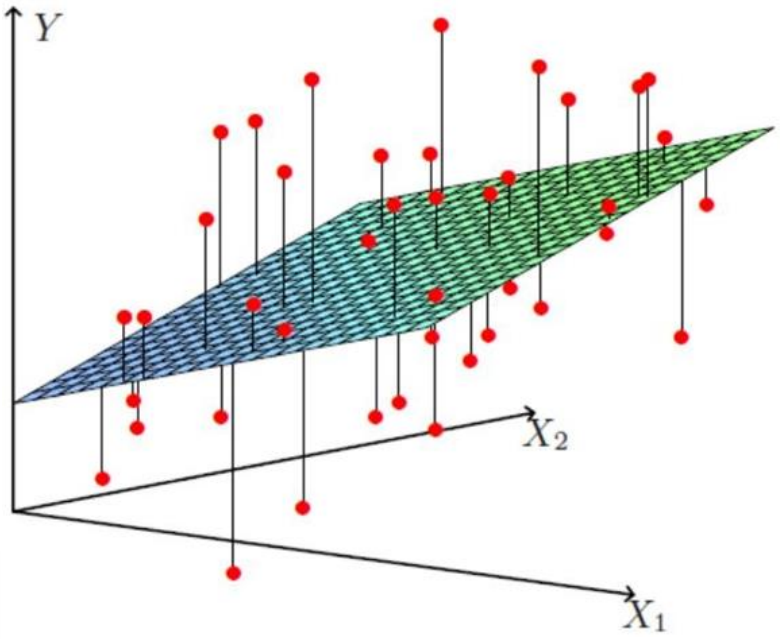
$$J(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2$$

我们期望利用给出的已知点$\{(x_i, y_i)|x_i \in R, y_i \in R\}_{i=1}^{n}$
找θ_0^*和θ_1^*，使得上面值最小，这样有θ_0^*和θ_1^*
决定的直线，拟合效果好。

二元线性回归+

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

两个特征值



三维空间中的二维决定的平面

$$y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

多元线性回归+

n+1 维空间中的 n维 决定的···@#%@···

$$y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots\cdots + \theta_n x_n$$

超平面

损失函数

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

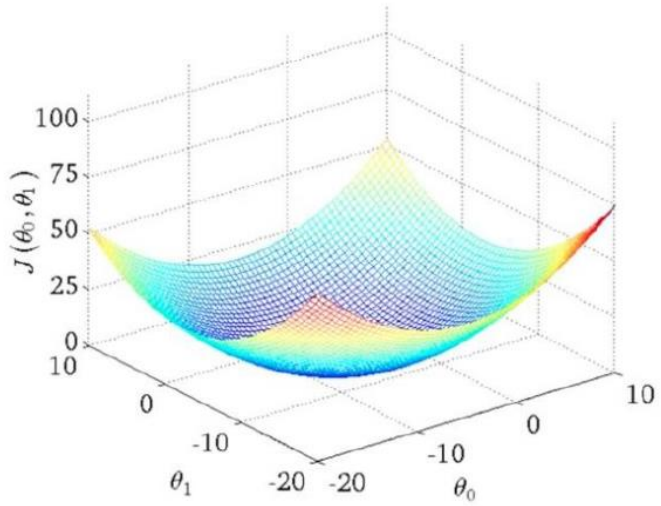
单变量线性回归（一元线性回归）

- 损失函数

Cost Function：$J(\theta_0,\theta_1)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x_i)-y_i)^2$

Goal：$minimize\ J(\theta_0,\theta_1)$

两种求解方法：
一、最小二乘法(正规方程求解)
二、梯度下降法——当样本数量很大时，求解逆矩阵计算量太大，因而，梯度下降法更为合适。



线性回归实现——最小二乘法

- 线性回归模型：　　$h_\theta(\boldsymbol{x}) = \boldsymbol{X\theta}$

  其中，　$\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_n]^T$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

  假设：真实值　$\boldsymbol{Y} = [y_1, y_2, \cdots, y_m]^T$

  那么损失函数：　$J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{X\theta} - \boldsymbol{Y})^T(\boldsymbol{X\theta} - \boldsymbol{Y})$

线性回归实现——最小二乘法公式推导

$$\nabla_{\theta} J(\theta) = \nabla_{\theta}\left(\frac{1}{2}(X\theta - Y)^T(X\theta - Y)\right)$$

$$= \nabla_{\theta}\left(\frac{1}{2}(\theta^T X^T - Y^T)(X\theta - Y)\right)$$

$$= \nabla_{\theta}\left(\frac{1}{2}(\theta^T X^T X\theta - \theta^T X^T Y - Y^T X\theta + Y^T Y)\right)$$

$$= \frac{1}{2}(2X^T X\theta - X^T Y - (Y^T X)^T)$$

$$= X^T X\theta - X^T Y$$

上式为0，则可得：　　$\theta = (X^T X)^{-1} X^T Y$

具体可参考代码：示例3

最小二乘法的问题：

- 1　X^TX不一定可逆
- 2　特征维数大时，求矩阵逆速度慢

- 梯度下降法是最优化方法的基础，是一款经典的求极值的算法
- 在很多机器学习算法中都有应用，比如：线性回归、逻辑回归、支持向量机以及神经网络等等

梯度下降法（Gradient Decent）

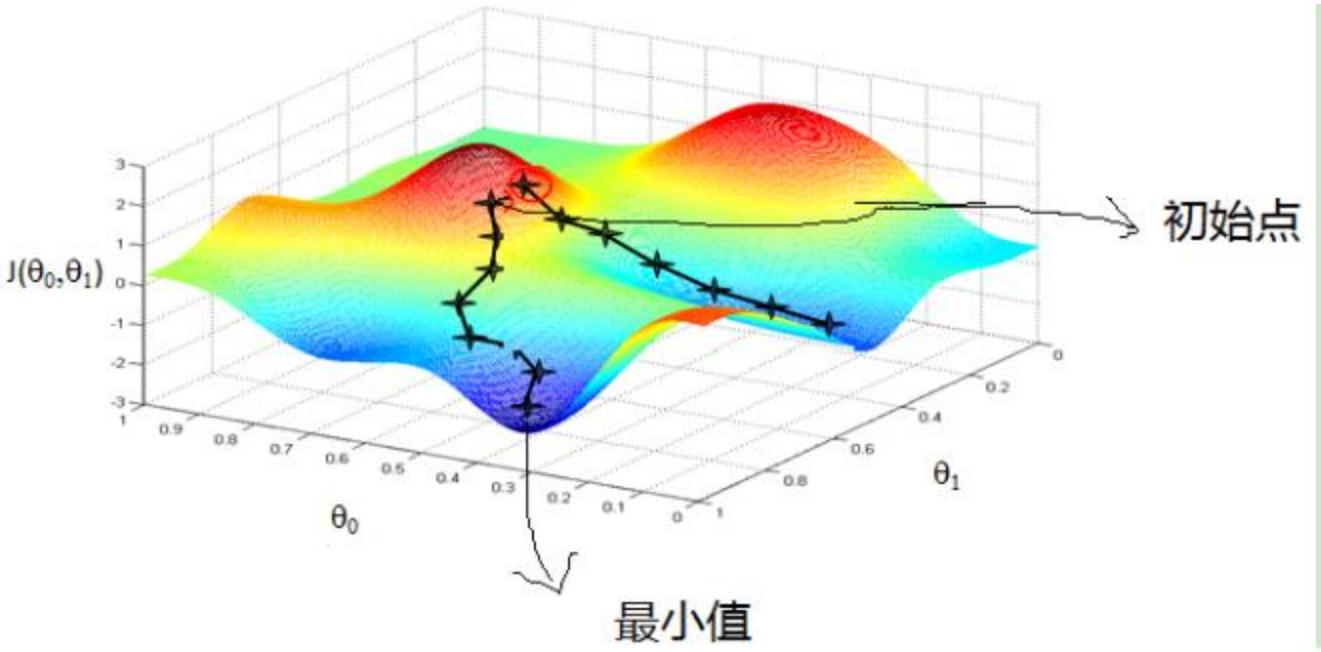
$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial_j}{\partial_{\theta 1}}$$

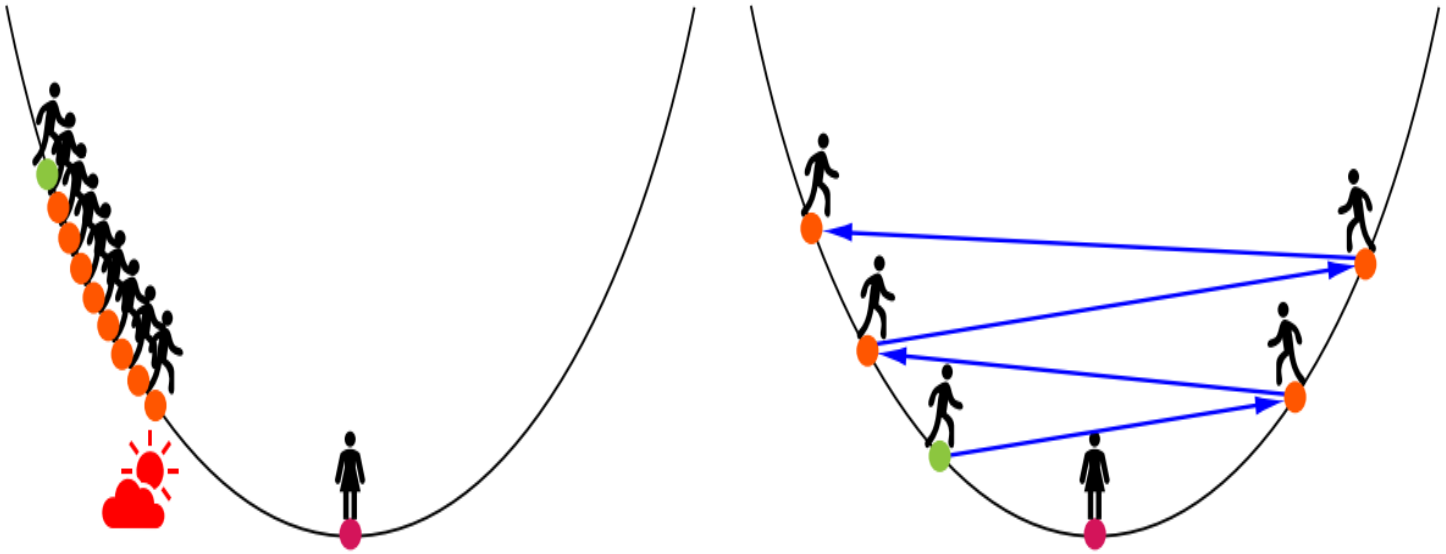
$$\theta_0 = \theta_0 - \alpha \frac{\partial_j}{\partial_{\theta 0}}$$

- 两个概念：
  - 学习率
    - 控制每次移动步伐大小，它就是学习率
  - 梯度
    - 导数

梯度下降寻找最小值的过程



学习率



学习率过小和过大所导致的两种情况

梯度下降的一维情况举例

$$J(\boldsymbol{\theta}) = \boldsymbol{\theta}^2 \qquad J'(\boldsymbol{\theta}) = \boldsymbol{2\theta}$$

初始化θ0 =−1，另α=0.4

θ0 =−1

θ1 = θ0 − α ∗ J′ (θ0) = −1 + 0.4 ∗ 2 = −0.2

θ2 = θ1 − α ∗ J′ (θ1) = −0.2 + 0.4 ∗ 0.4 = −0.04

θ3 = −0.008

θ4 = −0.0016

若θ0 =1，另α=0.4

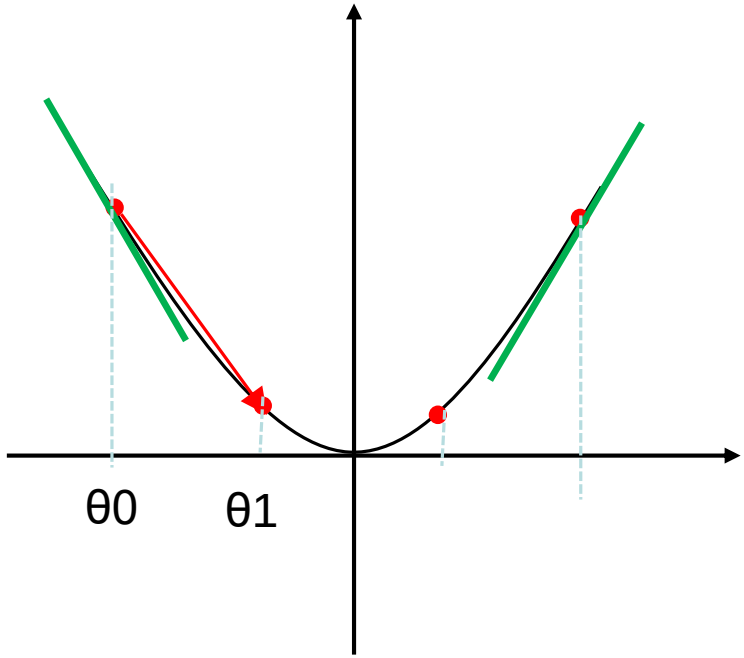
θ0 =1

θ1 = θ0 − α ∗ J′ (θ0) = 1 − 0.4 ∗ 2 = 0.2

θ2 = θ1 − α ∗ J′ (θ1) = 0.2 − 0.4 ∗ 0.4 = 0.04

θ3 = 0.008

θ4 = 0.0016



单变量线性回归实现——梯度下降算法步骤

（1）给定可微分的目标函数$J(\boldsymbol{\theta})$，学习率α，初始值$\boldsymbol{\theta}^t = [\theta_0, \theta_1, \cdots, \theta_n]^T$以及终止条件参数$\varepsilon$，设$t = 0$；

（2）计算目标函数在$\boldsymbol{\theta}^t$处的梯度$\nabla_{\boldsymbol{\theta}^t} J(\boldsymbol{\theta}^t)$；

（3）计算梯度向量的模$\left\| \nabla_{\boldsymbol{\theta}^t} J(\boldsymbol{\theta}^t) \right\|$，如果小于等于$\varepsilon$，算法结束；如果大于，继续下一步；

（4）更新参数$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha \nabla_{\boldsymbol{\theta}^t} J(\boldsymbol{\theta}^t)$；

（5）令$t \coloneqq t + 1$，并返回到第（2）步。

具体可参考代码示例5

单变量线性回归实现——梯度下降

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

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Gradient descent algorithm

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

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

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

}

update θ_0 and θ_1 simultaneously

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

- sklearn建模流程
  - 建立模型
    - lrModel = sklearn.linear_model.LinearRegression()
  - 训练模型
    - lrModel.fit(x,y)
  - 模型评估
    - lrModel.score(x,y)
  - 模型预测
    - lrModel.predict(x)

参考代码：示例4、5

示例4：要求掌握

示例5：手撕代码自行学习

梯度下降法与最小二乘法的对比

梯度下降法	最小二乘法
需要设置学习率	不需要设置学习率
需要反复迭代	一次运算得到结果
样本数量和样本特征维度增大时也能很好适用	需要计算$\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}$，样本数量和样本特征维度增大时运算代价大
适用于其它机器学习模型	只适用于线性回归

线性回归回顾

- 线性模型 $h_\theta(x) = \sum_{j=1}^{m} \theta_j x_j + \theta_0$
- 损失函数

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

- 求解损失函数最小值的两种方法：
  - 最小二乘法
  - 梯度下降
- 线性回归高次项模型（非线性拟合）
- 方差与偏差
- 欠拟合与过拟合
- 正则化

Sklearn 回归分析模型+

sklearn.linear_model.LinearRegression(……)

创建最小二乘法回归分析模型对象。

常用参数：通常可以不指定**超参数**。

返回： LinearRegression 对象实例。

常用属性：

coef_：除截距 θ_0 外的所有系数（数组）。θ_1 ~ θ_n

intercept_：截距 θ_0 。

LinearRegression 对象常用方法+

- fit(X，y，sample_weight=None)：拟合线性模型，模型实例本身已被改变。
  - X：训练集特征值样本(类数组)，二维数据。
  - y：训练集目标值(类数组)。可以一维，也可以是二维数据。
  - 返回：练成后的对象实例，通常无需保存返回值至变量，因为模型实例已经练成。

inplace

- predict(X)：使用线性模型进行预测。
  - X：测试集特征值样本(类数组)，二维数据。
  - 返回：预测值(数值)。
- score(X，y，sample_weight=None)：得分
  - X：测试集特征值样本(类数组)，二维数据。
  - y：测试集中真实目标值(类数组)。可以一维，也可以是二维数据。
  - 返回：R^2 指标，即 $1-\frac{\sum_{i=1}^{n}(y_i-\hat{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$，最佳为 1，差的可能为负数。

LinearRegression 示例——一元+

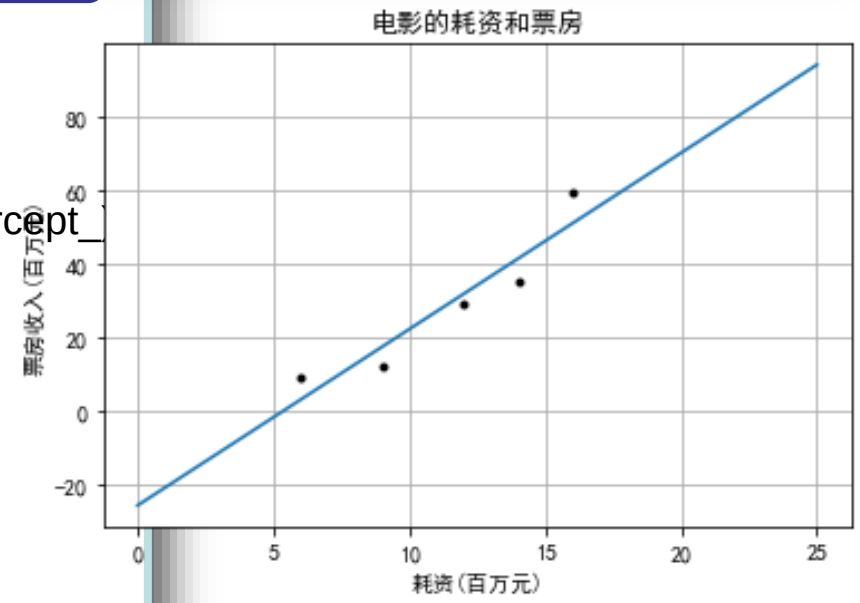
电影投资、收入(百万元)						
耗资	6	9	12	14	16	20
票房	9	12	29	35	59	?

```
import matplotlib.pyplot as plt
from sklearn import linear_model
plt.rcParams['font.sans-serif'] = ['Simhei']
plt.rcParams['axes.unicode_minus'] = False
X = [[6], [9], [12], [14], [16]] #二维数据训练特征数据
y = [9, 12, 29, 35, 59] #也可以是一维的，对模型的属性的维度有影响（降了一维）
model = linear_model.LinearRegression() #创建模型
model.fit(X, y) #训练模型
X_test= [[20]] #测试集特征值
a = model.predict(X_test) #用测试集特征数据预测数据
print("耗资{}百万的电影预计票房收入：{:.2f}百万元".format(X_test[0][0],a[0]))
print("回归模型的系数：",model.coef_ ,type(model.coef_))
print("回归模型的截距：",model.intercept_,type(model.intercept_))
print(f"最佳拟合线: y = {model.coef_[0]:.1f}X{model.intercept_:+.1f}")
plt.title('电影的耗资和票房')
plt.xlabel('耗资(百万元)')
plt.ylabel('票房收入(百万元)')
plt.grid(True)
plt.plot(X, y, 'k.') #绘制无连线的散点图
plt.plot( [0,25], model.predict ( [ [0], [25] ] )) #两点连线
plt.show()
```

训练集和测试集的特征数据维度相同

运行结果：

```
耗资20百万的电影预计票房收入：69.95百万元
回归模型的系数： [4.78481013] <class 'numpy.ndarray'>
回归模型的截距： -25.74683544303797 <class 'numpy.float64'>
最佳拟合线: y = 4.8X-25.7
```



LinearRegression 示例——二元+

```
import numpy as np
from sklearn import linear_model
x = np.array([[6,1,9],[9,3,12],[12,2,29],
[14,3,35],[16,4,59]]) #数据集，包括目标
X = x[:,:-1] #二维特征值(成本,广告)
y = x[:,-1]  #一维目标值(票房)
print('X:',X)
print('y:',y)
# 训练
regr = linear_model.LinearRegression()
regr.fit(X,y)
print('系数(θ1,θ2):',regr.coef_)
print('截距(θ0):',regr.intercept_)
# 预测
X_test= [[10,3]] #测试集二元特征数据
y_pred = regr.predict(X_test)
print(f'成本{X_test[0][0]}百万，推广{X_test[0][1]}百万的电影票房预测：
{y_pred[0]:.1f}百万')
```

注意各类数据的维度

运行结果：

```
X: [[ 6  1]
 [ 9  3]
 [12  2]
 [14  3]
 [16  4]]
y: [ 9 12 29 35 59]
系数(θ1,θ2): [ 4.94890511 -0.70072993]
截距(θ0): -25.79562043795622
成本10百万，推广3百万的电影票房预测：21.6百
万
```

电影投资收入(百万元)		
成本	广告	票房
6	1	9
9	3	12
12	2	29
14	3	35
16	4	59

保存变量至文件、从文件载入变量+

joblib.dump(value, filename, compress=0, ……)

将变量保存至文件中，以后可用 joblib.load() 直接从文件载入变量。

- value：要保持到文件的任何 python 对象。
- filename： 要在其中存储文件的文件对象或文件。如果文件扩展名为（.z、.gz、.bz2、.xz、.lzma）之一，则对应的压缩方法将自动启用。
- compress：压缩级别(0~9)，常用为 3 。
- 返回：文件名。

joblib.load(filename, ……)

从 joblib.dump() 生成的变量文件中载入变量。

- filename： 需载入的、由 joblib.dump() 生成的文件。
- 返回：文件中所保存的变量。

保存变量至文件、从文件载入变量——示例+

```
from joblib import dump,load
其余省略……
# 训练
regr = linear_model.LinearRegression()
regr.fit(X,y)
print('系数(θ1, θ2):',regr.coef_)
print('截距(θ0):',regr.intercept_)
# 预测
X_test= [[10,3]] #测试集二元特征数据
y_pred = regr.predict(X_test)
print(f'成本{X_test[0][0]}百万，推广{X_test[0][1]}百万的电影票房预测：{y_pred[0]:.1f}
百万')
dump(regr, '电影经济.joblib' )
newMovie=load( '电影经济.joblib' )
print('从文件读入的对象：\n系数(θ1, θ2):',newMovie.coef_)
print('截距(θ0):',newMovie.intercept_)
y_pred = newMovie.predict(X_test)
print(f'成本{X_test[0][0]}百万，推广{X_test[0][1]}百万的电影票房预测：{y_pred[0]:.1f}
百万')
```

运行结果：

```
系数(θ1,θ2): [ 4.94890511 -0.70072993]
截距(θ0): -25.79562043795622
成本10百万，推广3百万的电影票房预测：21.6百万
从文件读入的对象：
系数(θ1,θ2): [ 4.94890511 -0.70072993]
截距(θ0): -25.79562043795622
成本10百万，推广3百万的电影票房预测：21.6百万
```

注意各类数据的维度

多元回归、多个目标——示例+

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([[6,1,9,1],[9,3,12,1],[12,2,29,3],
[14,3,35,4],[16,4,59,6]]) #数据集，包括目标
X = x[:,:-2] #二维特征值（成本，广告）
y = x[:,-2:] #二维目标值（票房，周边）
# 训练
regr = LinearRegression()
regr.fit(X,y)
print('系数(θ1,θ2):\n',regr.coef_)
print('截距(θ0):\n',regr.intercept_)
X_test=[[11,2]]
y_pred=regr.predict(X_test)
print('测试集： ',X_test)
print('预测值： ',y_pred)
```

多维目标，多个子模型

sklearn 中的模型有此能力

运行结果：

```
系数(θ1,θ2):
 [[ 4.94890511 -0.70072993]
 [ 0.5729927  -0.28467153]]
截距(θ0):
 [-25.79562044  -2.7919708 ]
测试集：  [[11, 2]]
预测值：  [[27.24087591  2.94160584]]
```

多组系数，二维数组

多个截距，一维数组

多个目标，二维数组

单变量线性回归的二(N)次项模型

- 二次项模型：　$h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x \quad + \theta_2 x^2$

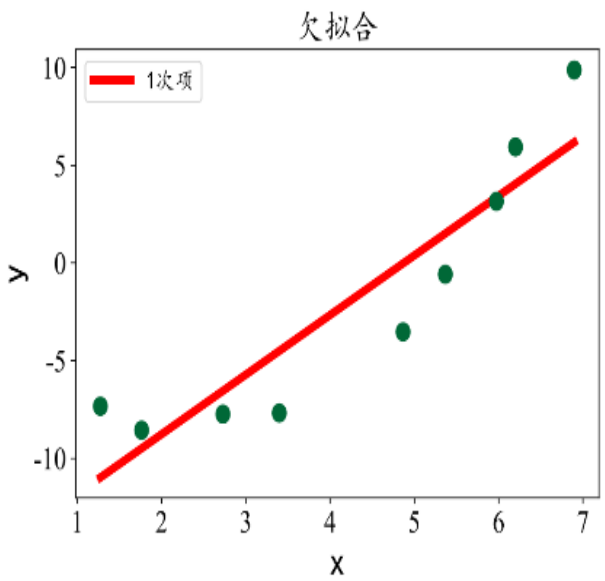
- 本质依然是线性模型：　$h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

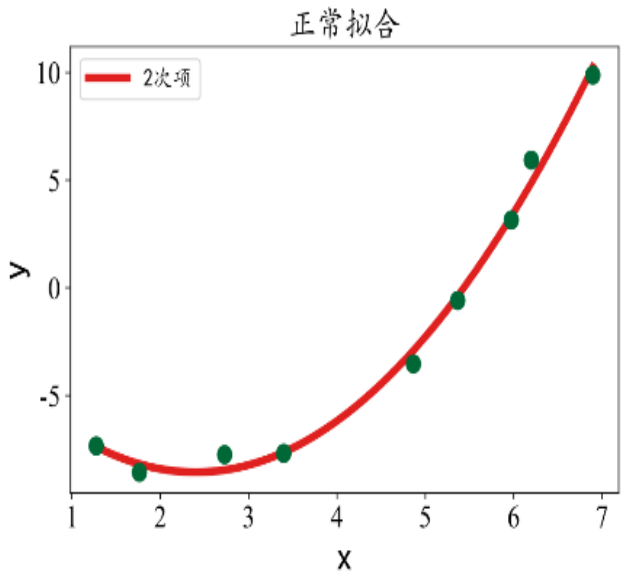
其中　$x_1 = x,$　　$x_2 = x^2$

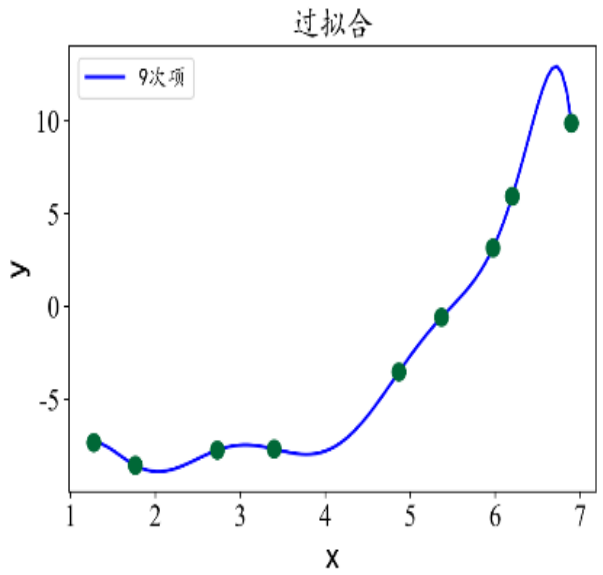
注意：所谓的线性模型是针对代估参数 $\boldsymbol{\theta}$ 而言的。

参看代码示例6、7

欠拟合、正常拟合、过拟合







假如现在有一个测试样本$x = 4$，y=-7;如果用九次项的蓝线的模型预测，结果约为-8.5；而如果用二次项的褐红色的模型预测，结果约为-7。

参考代码：示例8

过拟合现象，如何解决？

一、增加数据

二、正则化

正则化Regularization

- 最常见的正则化技术有一次项正则（L1正则）和二次项正则（L2正则）。

- L1正则：$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(\boldsymbol{x}_i) - y_i)^2 + \lambda\sum_{j=1}^{m}|\theta_j|$

Lasso回归

- L2正则：$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(\boldsymbol{x}_i) - y_i)^2 + \lambda\sum_{j=1}^{m}\theta_j{}^2$

Ridge回归（岭回归）

- 参考代码：示例9

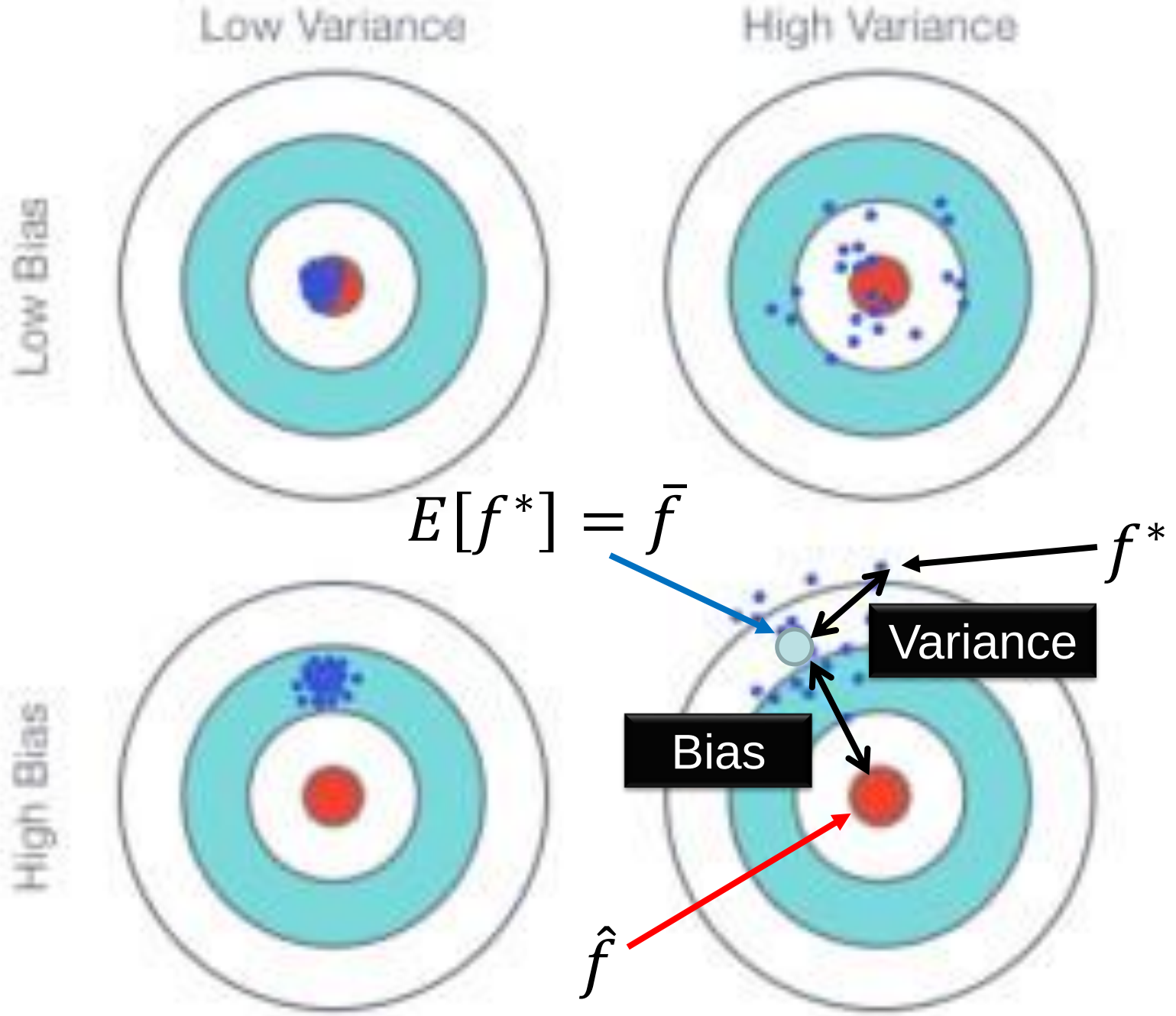
欠拟合与过拟合

训练集验证集上的表现	测试集上的表现	结论
不好	不好	欠拟合（underfitting）
好	不好	过拟合（overfitting）
好	好	适度拟合

- 若一个模型欠拟合，则它在训练集上损失值较大，在测试集上损失值也非常大，那就说明预测值偏离真实数据集，即高偏差。
- 若一个模型过拟合，则它在训练集上损失值较小，但是在测试集上损失值较大，那就说明模型复杂度过高，不同的训练集样本可以训练出较大差异的模型，产生的预测值差异较大，则会产生高方差。

衡量模型好坏的标准

- 误差＝偏差＋方差＋噪声
- 偏差：描述的是预测值（估计值）的期望与真实值之间的差距。偏差越大，越偏离真实数据集。
- 方差：描述的是预测值的变化范围，离散程度，也就是离其期望值的距离。方差越大，预测结果数据的分布越散。


Low Variance
High Variance
Low Bias
High Bias
$E[f^*] = \bar{f}$
f^*
Variance
Bias
$\hat{f}$

欠拟合与过拟合的解决办法

- 欠拟合（高偏差）解决办法：
  - 增加特征
  - 提高模型复杂度
  - 减小正则化系数
- 过拟合（高方差）解决办法：
  - 增加训练数据
  - 降低模型复杂度
  - 增加正则化项
  - 采用集成学习*

模型的选择

- 实际应用中，往往需要在偏差与方差之间权衡，选择误差最小的模型。

- 交叉验证

- 对于一个机器学习系统，需要先确保一个系统是低偏差的。可以通过提高模型复杂度来降低偏差，比如增加神经系统的隐藏层。然后增大数据集，可以利用人工合成数据来扩充数据集，以此来降低方差，提高模型的性能。

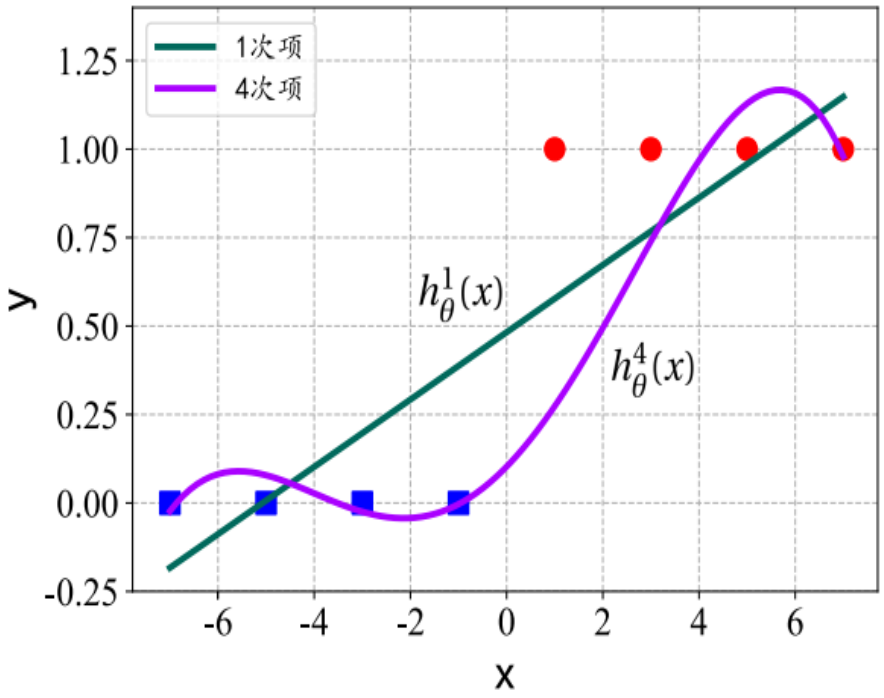
回顾：分类

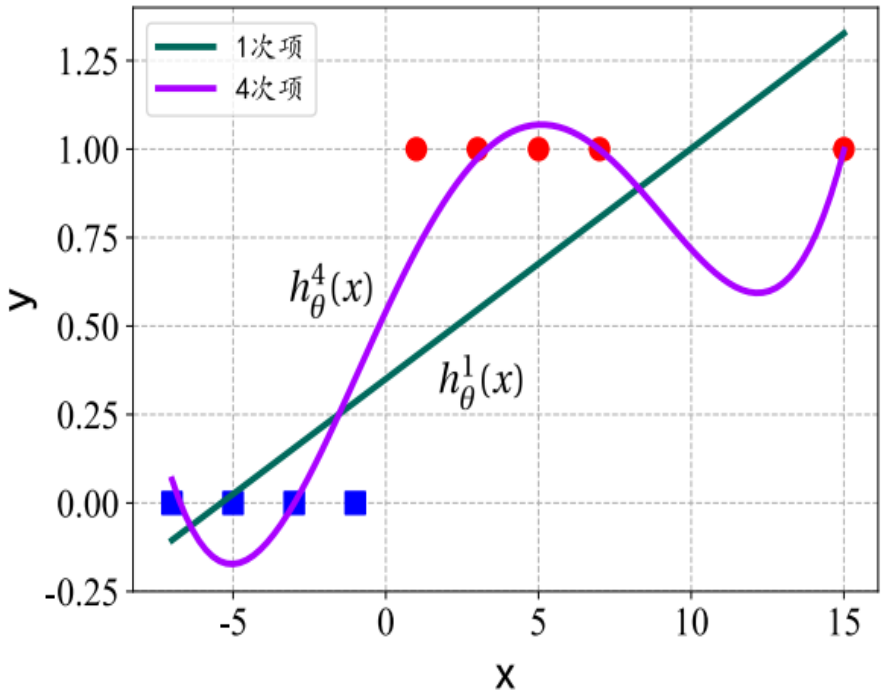
- 信用贷款：
  - 输入：收入、存款、职业、年龄、信用记录
  - 输出：可贷款/拒绝贷款
- 医疗诊断
  - 输入：当前症状、年龄、性别、医疗记录……
  - 输出：某种疾病
- 手写体识别

输入:  输出: **金**

- 人脸识别
  - 输入：人脸图象
  - 输出：某个人

线性回归能分类吗？



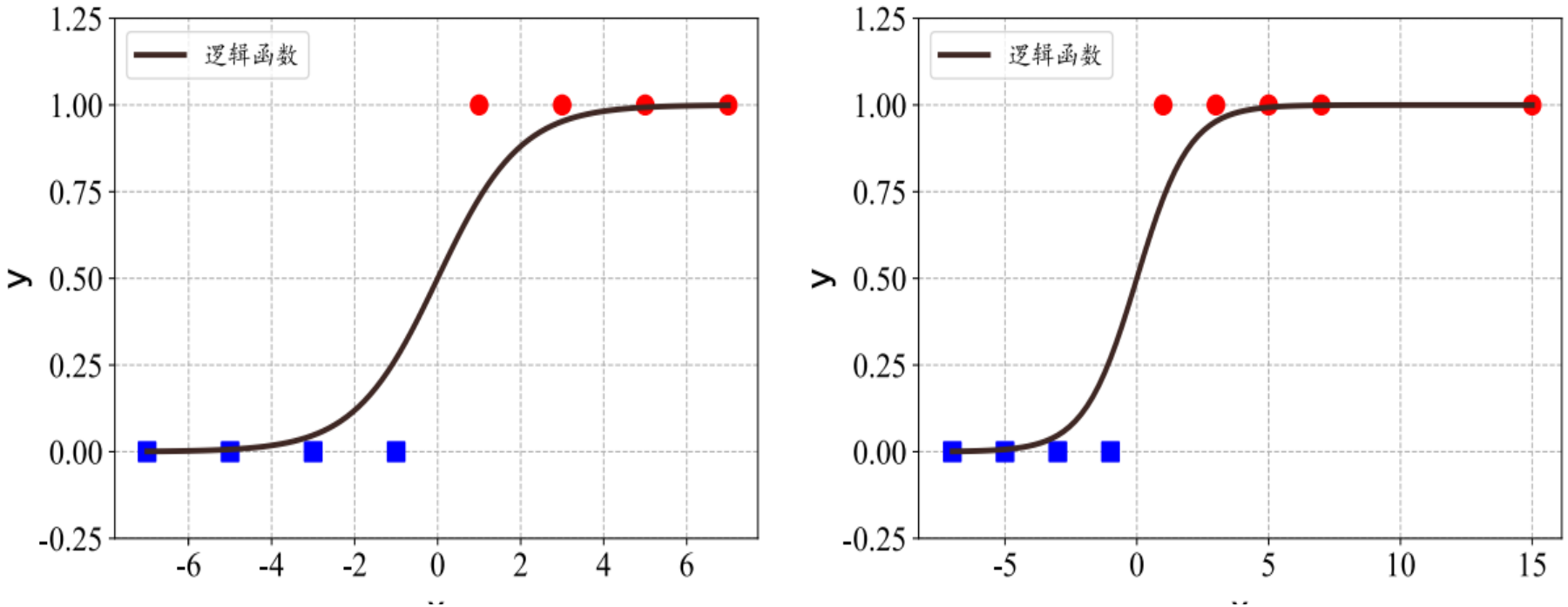


$$f\left(h_\theta^1(x)\right) = \begin{cases} 0, & h_\theta^1(x) < 0.5 \\ 1, & h_\theta^1(x) \geq 0.5 \end{cases}$$

线性回归能分类吗？

- 对于样本不够对称的情况容易出错。
- 采用特征提取或特征构建等手段拟合出来的多次项曲线能够更好地拟合一般的非对称的样本，但是，通过不断地组合多次项以及交叉项来找到合适的曲线，并不容易。
- 拟合出来的曲线，要有一个合适的阈值，才能成功分类，但是阈值设置多少合适，没有通用手段可以确定。
- 有没有更好的办法？？？
  - 有，逻辑回归

逻辑回归



逻辑函数拟合分类样本

- 逻辑函数（sigmoid函数）

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

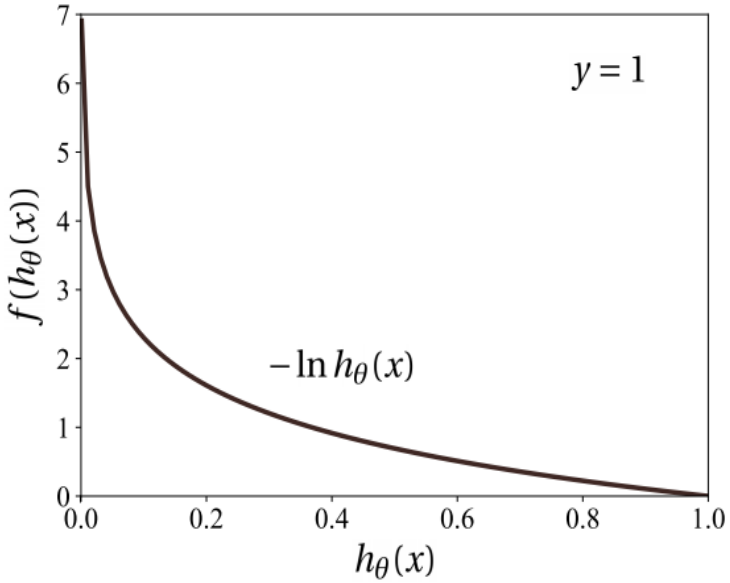
- 逻辑回归模型：

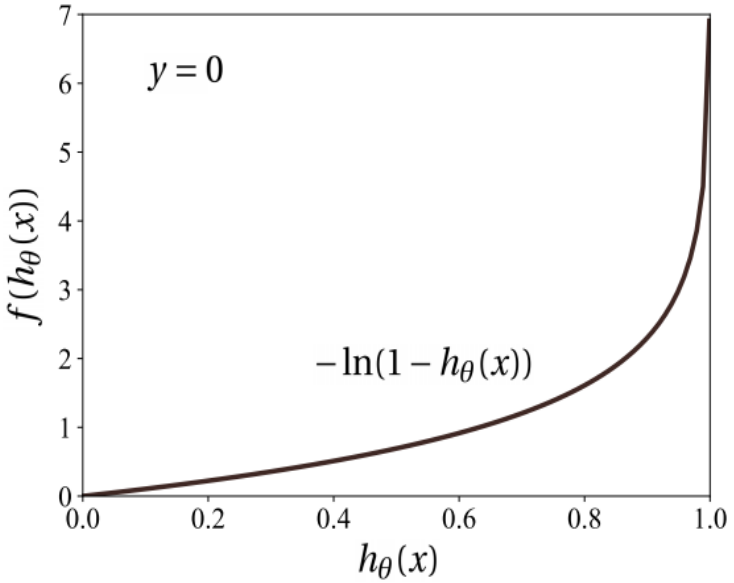
$$h_\theta(\boldsymbol{x}) = \sigma\left(\boldsymbol{\theta}^T\boldsymbol{x}\right) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}}}$$

$$\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_m]^T, \quad \boldsymbol{x} = [1, x_1, \cdots, x_m]^T$$

构造损失函数

- 构造损失函数：$f\big(h_\theta(\boldsymbol{x})\big)=\begin{cases}-\ln h_\theta(\boldsymbol{x}), & y=1\\ -\ln\big(1-h_\theta(\boldsymbol{x})\big), & y=0\end{cases}$





- 损失函数：

$$J(\boldsymbol{\theta})=-\sum_{i=1}^{n}\big(y_i\ln h_\theta(\boldsymbol{x}_i)+(1-y_i)\ln\big(1-h_\theta(\boldsymbol{x}_i)\big)\big)$$

梯度下降法求解

- 对损失函数求偏导可得：

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta}) &= -\sum_{i=1}^{n}\left(\frac{y_i}{h_\theta(\boldsymbol{x}_i)}-\frac{1-y_i}{1-h_\theta(\boldsymbol{x}_i)}\right)\frac{\partial}{\partial\theta_j}h_\theta(\boldsymbol{x}_i)\\ &= -\sum_{i=1}^{n}\left(\frac{y_i}{h_\theta(\boldsymbol{x}_i)}-\frac{1-y_i}{1-h_\theta(\boldsymbol{x}_i)}\right)h_\theta(\boldsymbol{x}_i)\big(1-h_\theta(\boldsymbol{x}_i)\big)\frac{\partial}{\partial\theta_j}\boldsymbol{\theta}^T\boldsymbol{x}_i\\ &= -\sum_{i=1}^{n}\Big(y_i\big(1-h_\theta(\boldsymbol{x}_i)\big)-(1-y_i)h_\theta(\boldsymbol{x}_i)\Big)x_{ij}\\ &= \sum_{i=1}^{n}(h_\theta(\boldsymbol{x}_i)-y_i)x_{ij}\end{aligned}$$

用梯度下降法求解时，θ_j的更新公式：　$\theta_j := \theta_j - \alpha\sum_{i=1}^{n}\big(h_\theta(\boldsymbol{x}_i)-y_i\big)x_{ij}$

- 对损失函数求二阶偏导：　$\frac{\partial^2}{\partial\theta_j^2}J(\boldsymbol{\theta}) = \sum_{i=1}^{n}h_\theta(\boldsymbol{x}_i)\big(1-h_\theta(\boldsymbol{x}_i)\big)x_{ij}\frac{\partial}{\partial\theta_j}\boldsymbol{\theta}^T\boldsymbol{x}_i$

$$= \sum_{i=1}^{n}h_\theta(\boldsymbol{x}_i)\big(1-h_\theta(\boldsymbol{x}_i)\big)x_{ij}^2 > 0$$

  - 二阶偏导恒大于0，说明逻辑回归的损失函数是凸函数，所以，使用梯度下降法就可以得到全局最优解。

线性回归与逻辑回归比较

- 线性回归和逻辑回归的算法框架相似。
- 线性回归一般用来解决回归问题，而逻辑回归通过逻辑函数进行非线性映射能更好地解决分类问题。
- 都可以使用梯度下降法求得问题的最优解，也都可以使用L1正则和L2正则来降低过拟合的风险。不过，逻辑回归无法通过最小二乘法求解。

竞赛及数据集平台介绍

- 【Kaggle竞赛】Kaggle简介
  - https://zhuanlan.zhihu.com/p/480148648
- 华为论坛竞赛：
  - https://bbs.huaweicloud.com/forum/thread-66983-1-1.html
- 阿里云天池大赛
  - https://tianchi.aliyun.com/
- DF,CCF指定专业大数据竞赛平台
  - https://www.datafountain.cn/
- AI Challenger — 全球AI挑战赛
  - https://challenger.ai/
- 竞赛及数据集：
  - https://github.com/HuangCongQing/AI_competitions

谢　谢！