

法律声明

■ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，北风网和讲师拥有完全知识产权；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或者机构不得盗版、复制、仿造其中的创意和内容，我们保留一切通过法律手段追究违反者的权利。

■ 课程详情请咨询

◆ 微信公众号：北风教育

◆ 官方网址：<http://www.ibeifeng.com/>



人工智能之机器学习

回归算法

主讲人：Gerry

上海育创网络科技有限公司



课程要求

■ 课上课下 “九字” 真言

- ◆ 认真听，善摘录，勤思考
- ◆ **多温故，乐实践**，再发散

■ 四不原则

- ◆ **不懒散惰性，不迟到早退**
- ◆ **不请假旷课，不拖延作业**

■ 一点注意事项

- ◆ 违反 “四不原则” ， 不包就业和推荐就业

严格是大爱



寄语



做别人不愿做的事，
做别人不敢做的事，
做别人做不到的事。

课程内容

- 线性回归算法
- 多项式回归算法
- 正则化
- Logistic回归算法
- Softmax回归算法
- 梯度下降
- 特征抽取
- 线性回归案例

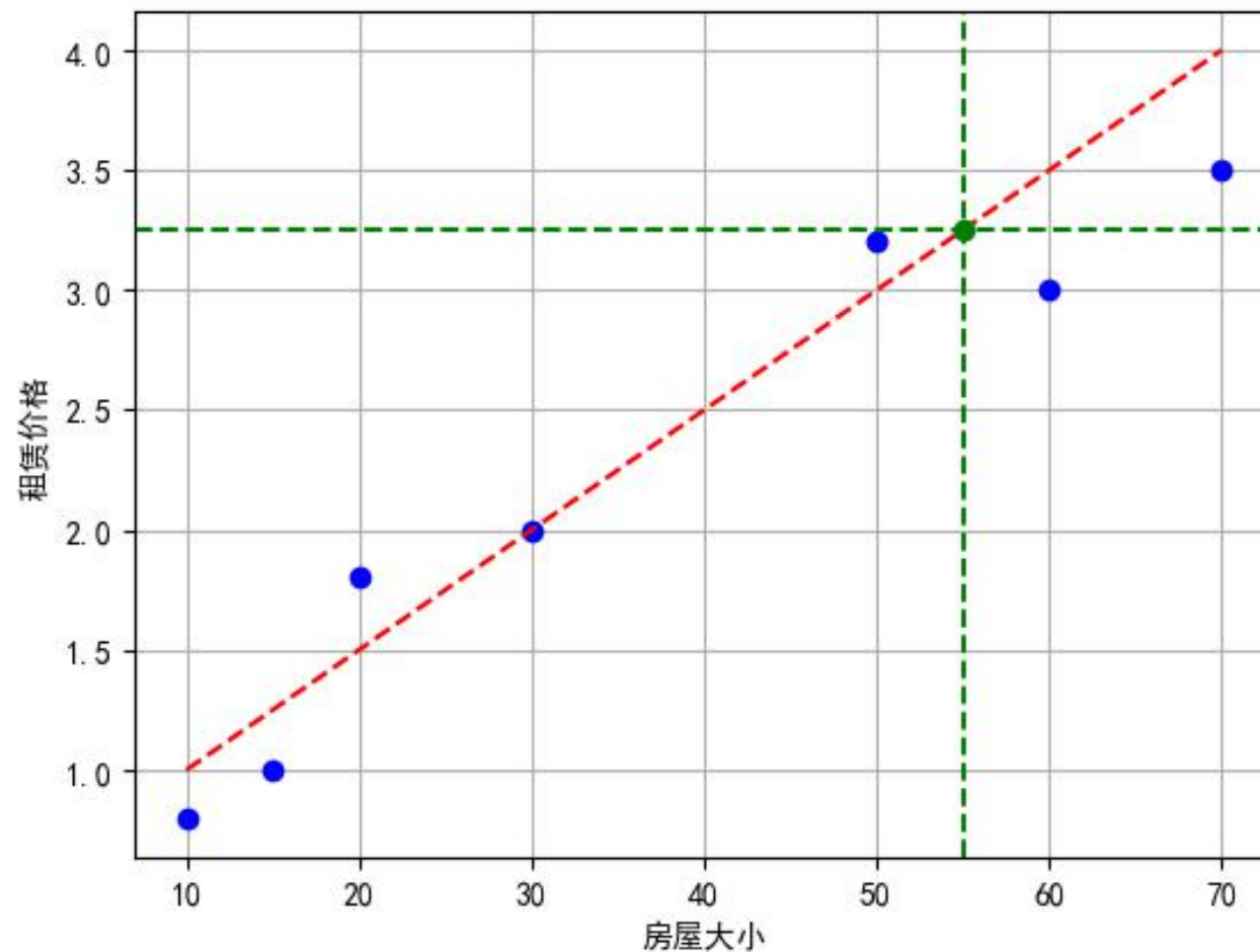
什么是回归算法

- 回归算法是一种有监督算法
- 回归算法是一种比较常用的机器学习算法，用来建立“解释”变量(自变量X)和观测值(因变量Y)之间的关系；从机器学习的角度来讲，用于构建一个算法模型(函数)来做属性(X)与标签(Y)之间的映射关系，在算法的学习过程中，试图寻找一个函数 $h: R^d \rightarrow R$ 使得参数之间的关系**拟合性**最好。
- 回归算法中算法(函数)的最终结果是一个**连续**的数据值，输入值(属性值)是一个d维度的属性/数值向量

线性回归

■ $y=ax+b$

房屋面积(m ²)	租赁价格(1000¥)
10	0.8
15	1
20	1.8
30	2
50	3.2
60	3
60	3.1
70	3.5



回归算法理性认知

■ 房价的预测

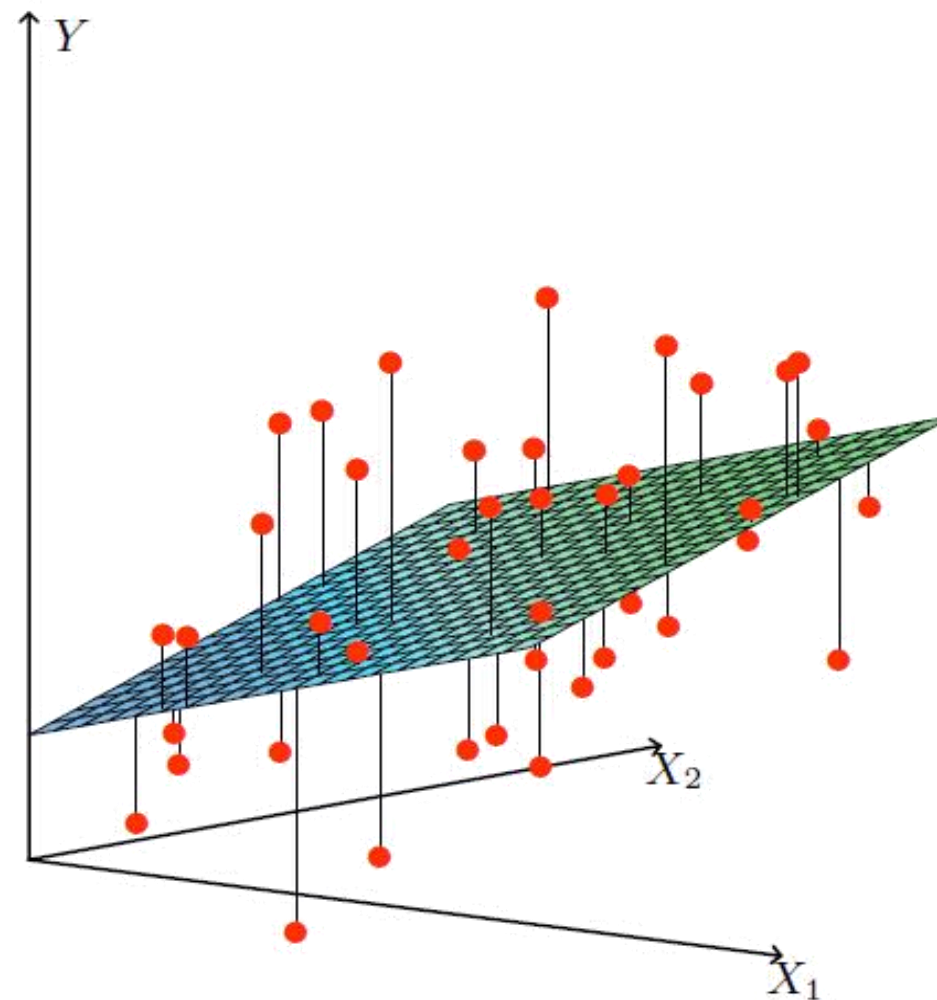
房屋面积(m ²)	租赁价格(1000¥)
10	0.8
15	1
20	1.8
30	2
50	3.2
60	3
60	3.1
70	3.5

请问，如果现在有一个房屋面积为55平，请问最终的租赁价格是多少比较合适？

线性回归

$$\blacksquare h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

房屋面积	房间数量	租赁价格
10	1	0.8
20	1	1.8
30	1	2.2
30	2	2.5
70	3	5.5
70	2	5.2
.....



线性回归

$$\begin{aligned}
 h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \\
 &= \theta_0 1 + \theta_1 x_1 + \cdots + \theta_n x_n \\
 &= \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \\
 &= \sum_{i=0}^n \theta_i x_i = \theta^T x
 \end{aligned}$$

最终要求是计算出 θ 的值，并选择最优的 θ 值构成算法公式

线性回归、最大似然估计及二乘法

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$$

- 误差 $\varepsilon^{(i)} (1 \leq i \leq n)$ 是独立同分布的，服从均值为0，方差为某定值 σ^2 的**高斯分布**。
 - ◆ 原因：**中心极限定理**
- 实际问题中，很多随机现象可以看做**众多因素**的独立影响的综合反应，往往服从正态分布

似然函数

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)} \quad p(\varepsilon^{(i)}) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)}$$

$$p(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$L(\theta) = \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta)$$

$$= \prod_{i=1}^m \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

对数似然、目标函数及最小二乘

$$\begin{aligned}
 \ell(\theta) &= \log L(\theta) \\
 &= \log \prod_{i=1}^m \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
 &= \sum_{i=1}^m \log \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
 &= m \log \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2
 \end{aligned}$$

$$\text{loss}(y_j, \hat{y}_j) = J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

θ的求解过程

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

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

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

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

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

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

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

最小二乘法的参数最优解

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

■ 参数解析式

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

- 最小二乘法的使用要求矩阵 $X^T X$ 是**可逆**的；为了防止不可逆或者过拟合的问题存在，可以增加额外数据影响，导致最终的矩阵是可逆的：

$$\theta = (X^T X + \lambda I)^{-1} X^T y$$

- 最小二乘法直接求解的难点：矩阵逆的求解是一个难处

普通最小二乘法线性回归案例

■ 现有一批描述家庭用电情况的数据，对数据进行算法模型预测，并最终得到预测模型（每天各个时间段和功率之间的关系、功率与电流之间的关系等）

◆ 数据来源: [Individual household electric power consumption Data Set](#)

◆ 建议：使用python的sklearn库的linear_model中LinearRegression来获取算法

Individual household electric power consumption Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available.

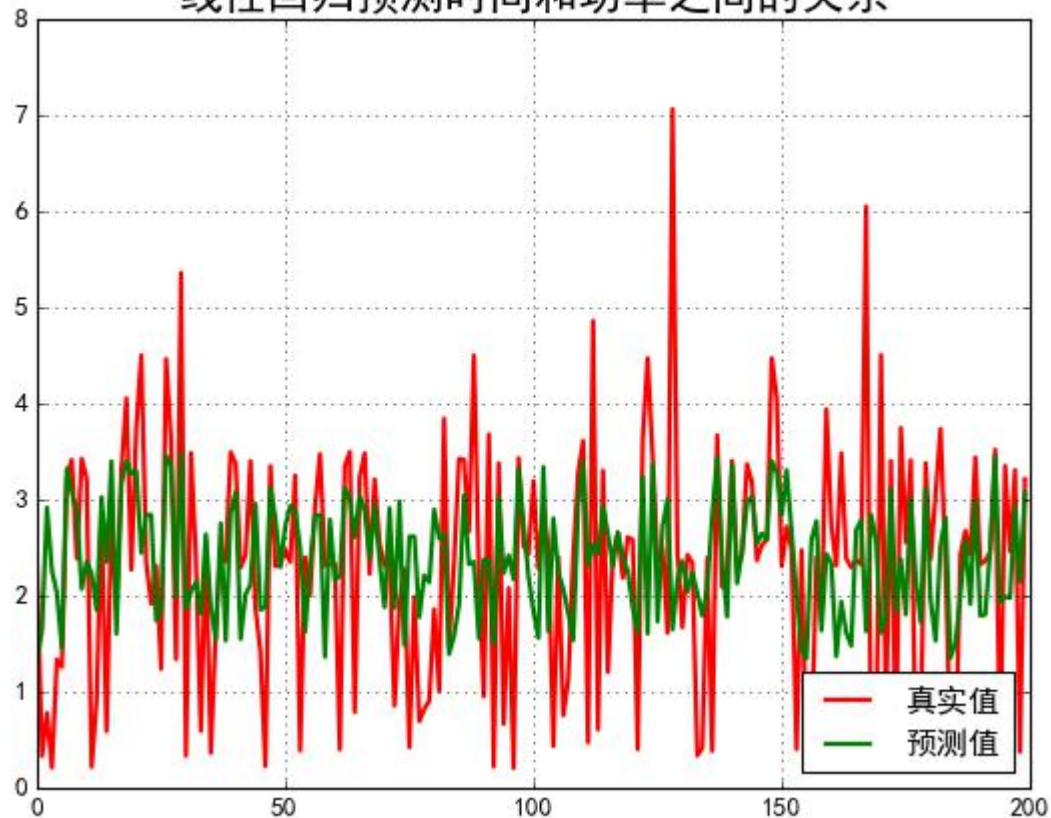
Data Set Characteristics:	Multivariate, Time-Series	Number of Instances:	2075259	Area:	Physical
Attribute Characteristics:	Real	Number of Attributes:	9	Date Donated	2012-08-30
Associated Tasks:	Regression, Clustering	Missing Values?	Yes	Number of Web Hits:	135342

Attribute Information:

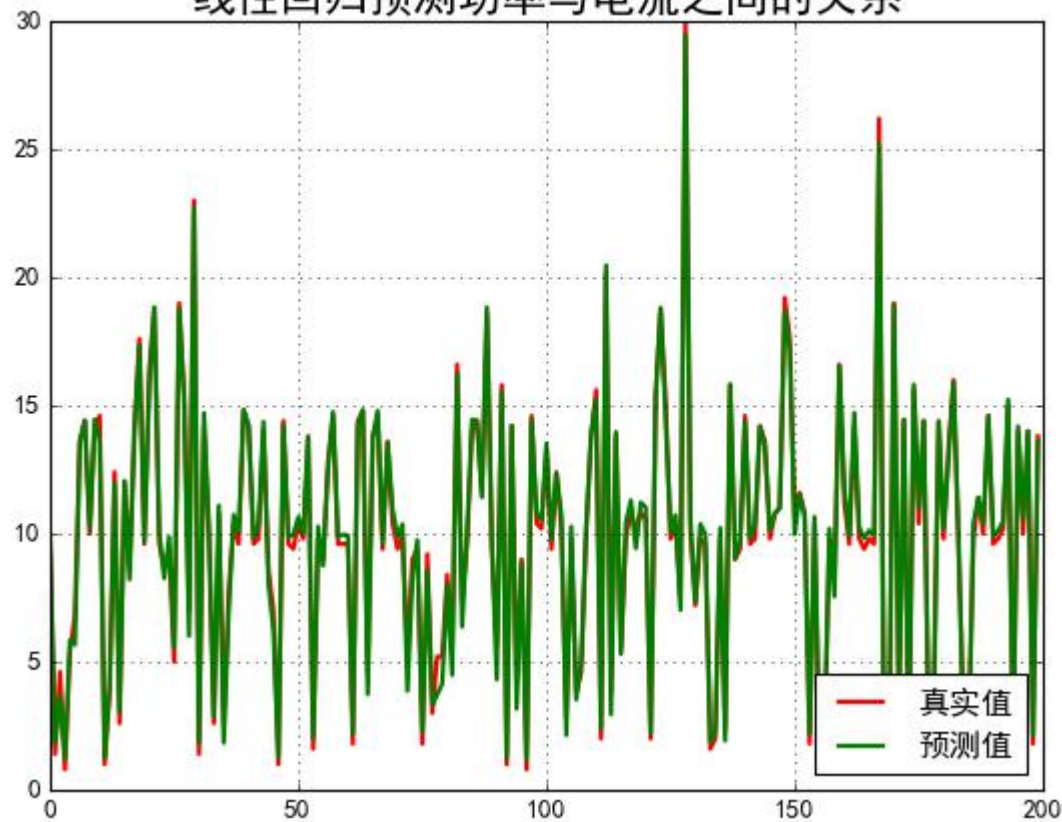
- 1.date: Date in format dd/mm/yyyy
- 2.time: time in format hh:mm:ss
- 3.global_active_power: household global minute-averaged active power (in kilowatt)
- 4.global_reactive_power: household global minute-averaged reactive power (in kilowatt)
- 5.voltage: minute-averaged voltage (in volt)
- 6.global_intensity: household global minute-averaged current intensity (in ampere)
- 7.sub_metering_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
- 8.sub_metering_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
- 9.sub_metering_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

普通最小二乘法线性回归案例

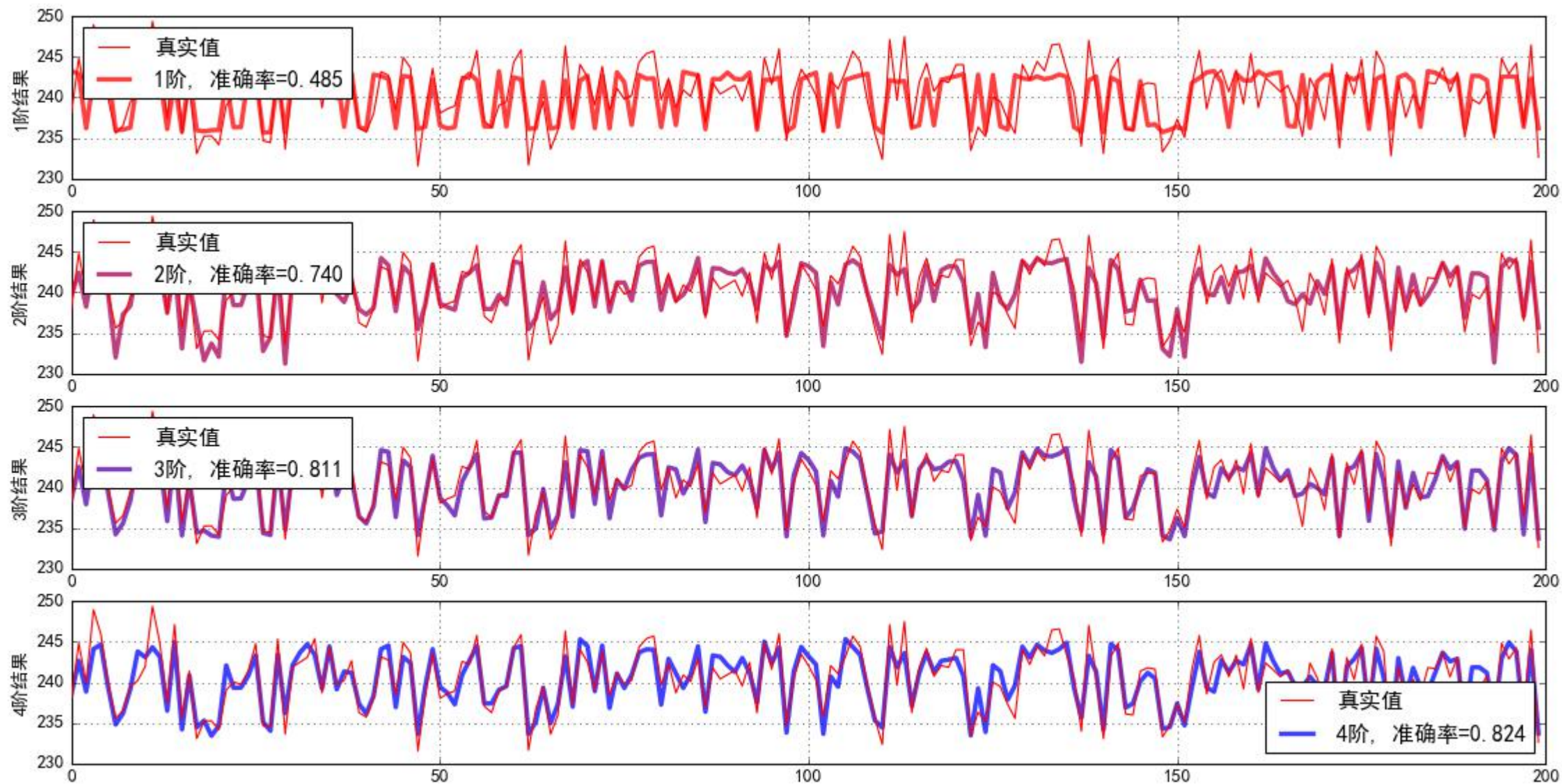
线性回归预测时间和功率之间的关系



线性回归预测功率与电流之间的关系



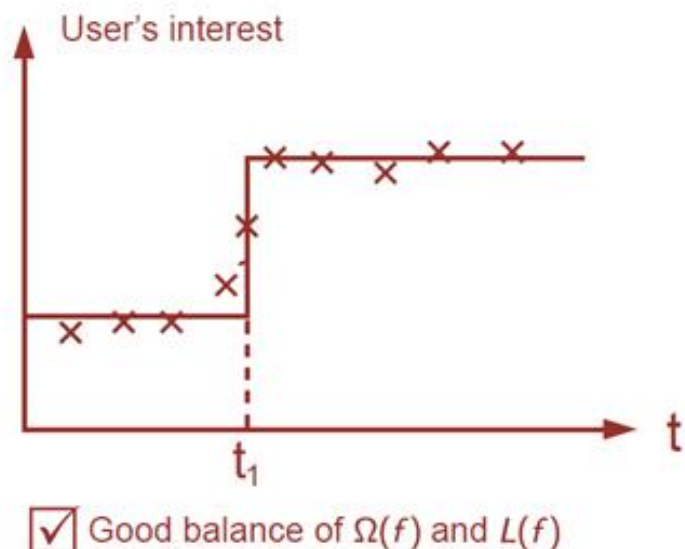
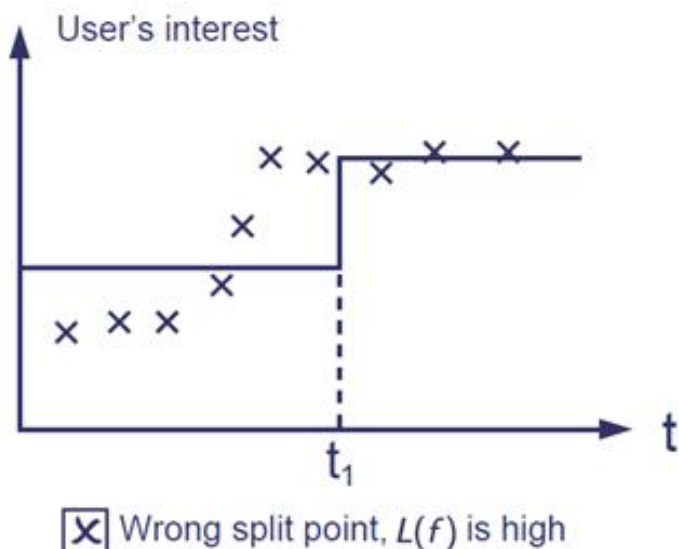
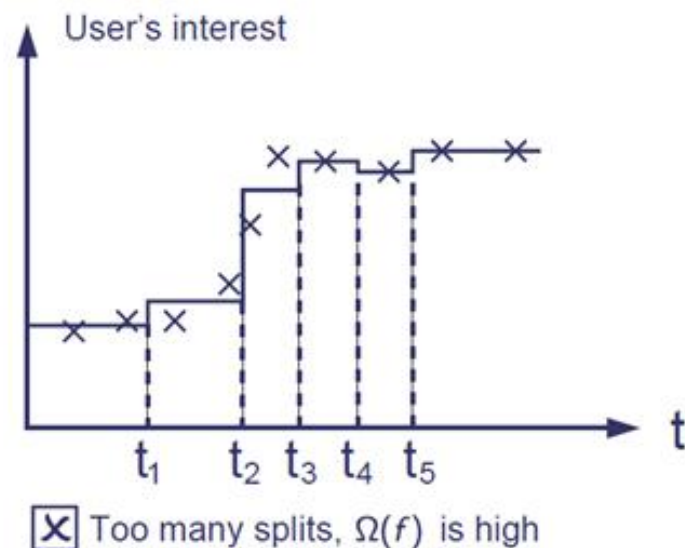
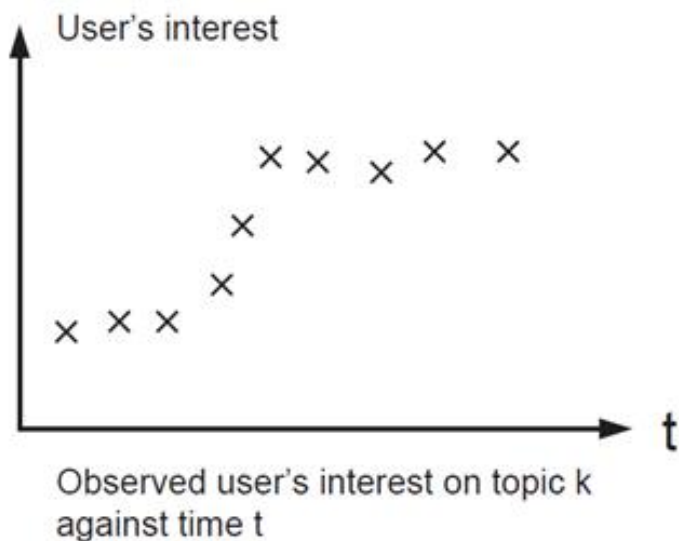
普通最小二乘法线性回归案例



目标函数(loss/cost function)

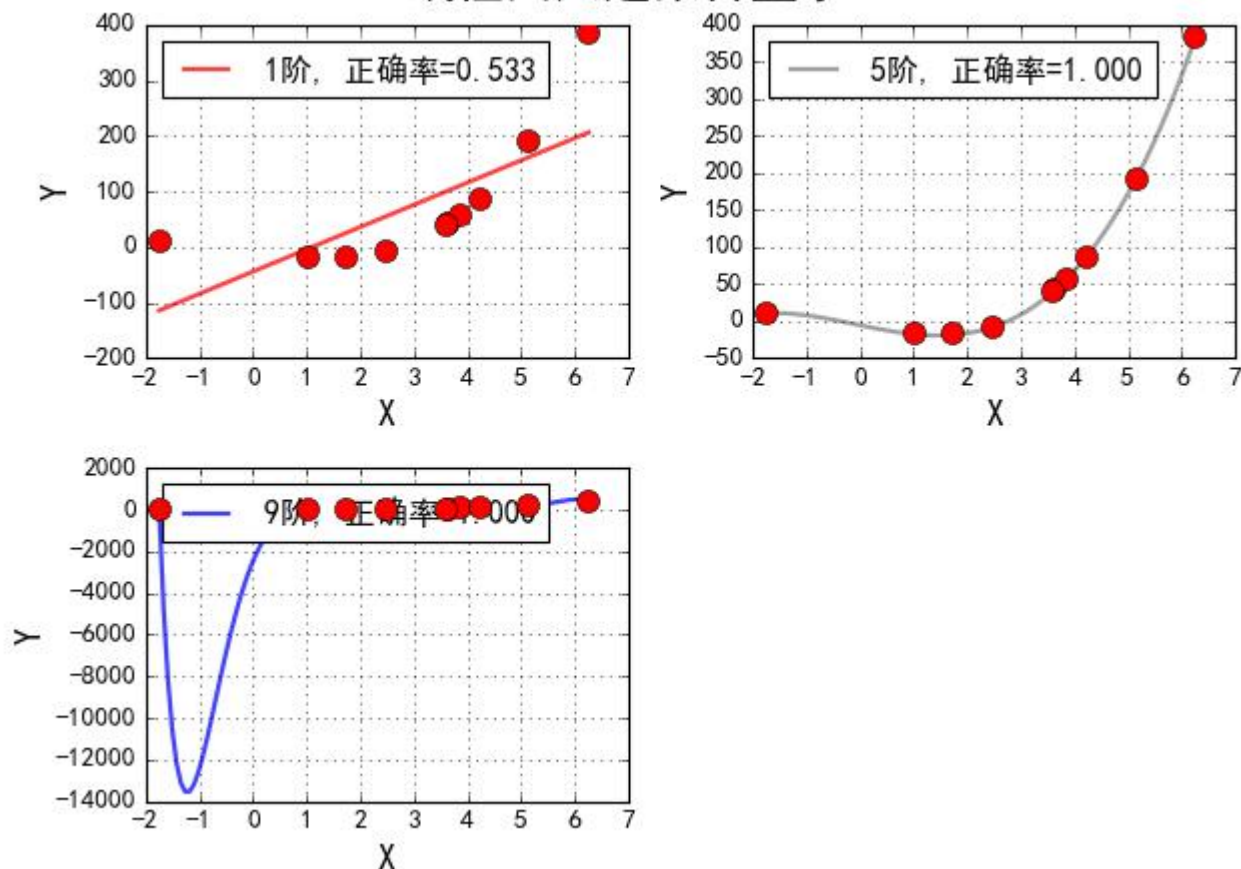
- 0-1损失函数 $J(\theta) = \begin{cases} 1, Y \neq f(X) \\ 0, Y = f(X) \end{cases}$
- 感知损失函数 $J(\theta) = \begin{cases} 1, |Y - f(X)| > t \\ 0, |Y - f(X)| \leq t \end{cases}$
- 平方和损失函数 $J(\theta) = \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- 绝对值损失函数 $J(\theta) = \sum_{i=1}^m |h_{\theta}(x^{(i)}) - y^{(i)}|$
- 对数损失函数 $J(\theta) = -\sum_{i=1}^m (y^{(i)} \log h_{\theta}(x^{(i)}))$

模型



过拟合

线性回归过拟合显示



结论：不是阶数越高效果越好

1阶, 系数为: [-44.14102611 40.05964256]

5阶, 系数为: [-5.60899679 -14.80109301 0.75014858 2.11170671 -0.07724668 0.00566633]

9阶, 系数为: [-2465.58378507 6108.6381056 -5111.99327317 974.74973548 1078.89648247 -829.50276827 266.13230319 -45.71741527 4.11582735 -0.15281063]

线性回归的过拟合

- 目标函数: $J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- 为了防止数据过拟合, 也就是的 θ 值在样本空间中不能过大/过小, 可以在目标函数之上增加一个平方和损失:

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

- 正则项(norm): $\lambda \sum_{j=1}^n \theta_j^2$; 这里这个正则项叫做L2-norm

过拟合和正则项

■ L2-norm:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^n \theta_j^2 \quad \lambda > 0$$

■ L1-norm:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^n |\theta_j| \quad \lambda > 0$$

Ridge回归

- 使用L2正则的线性回归模型就称为Ridge回归(岭回归)

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^n \theta_j^2 \quad \lambda > 0$$

LASSO回归

- 使用L1正则的线性回归模型就称为LASSO回归(Least Absolute Shrinkage and Selection Operator)

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^n |\theta_j| \quad \lambda > 0$$

Ridge(L2-norm)和LASSO(L1-norm)比较

- L2-norm中，由于对于各个维度的参数缩放是在一个圆内缩放的，不可能导致有维度参数变为0的情况，那么也就不会产生稀疏解；实际应用中，数据的维度中是存在**噪音**和**冗余**的，稀疏的解可以找到有用的维度并且减少冗余，提高回归预测的**准确性**和**鲁棒性**（减少了overfitting）（L1-norm可以达到最终解的**稀疏性**的要求）
- Ridge模型具有较高的准确性、鲁棒性以及稳定性(冗余特征已经被删除了)；LASSO模型具有较高的求解速度。
- 如果既要考虑稳定性也考虑求解的速度，就使用Elastic Net

Ridge(L2-norm)和LASSO(L1-norm)比较

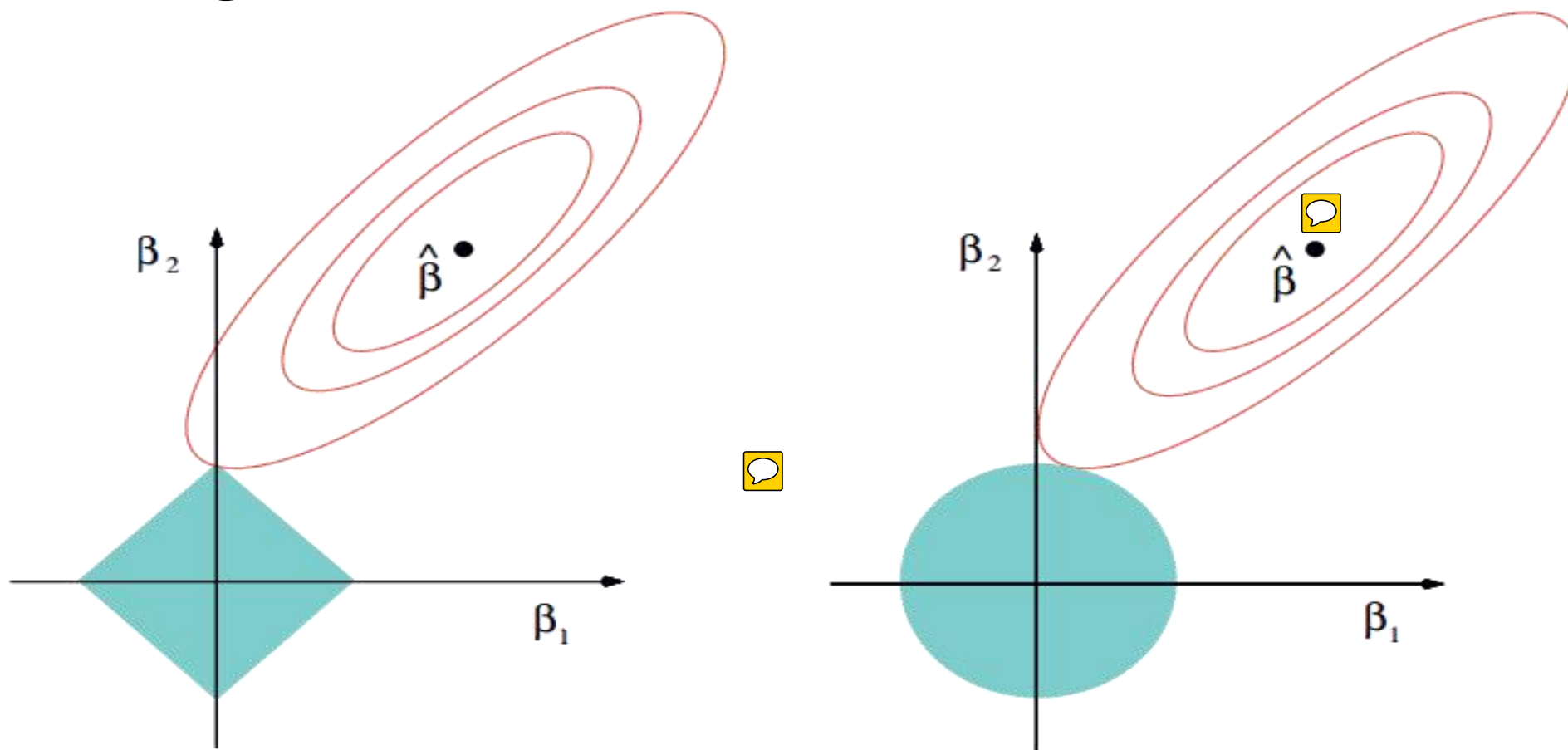


FIGURE 3.11. Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.

Elastic Net

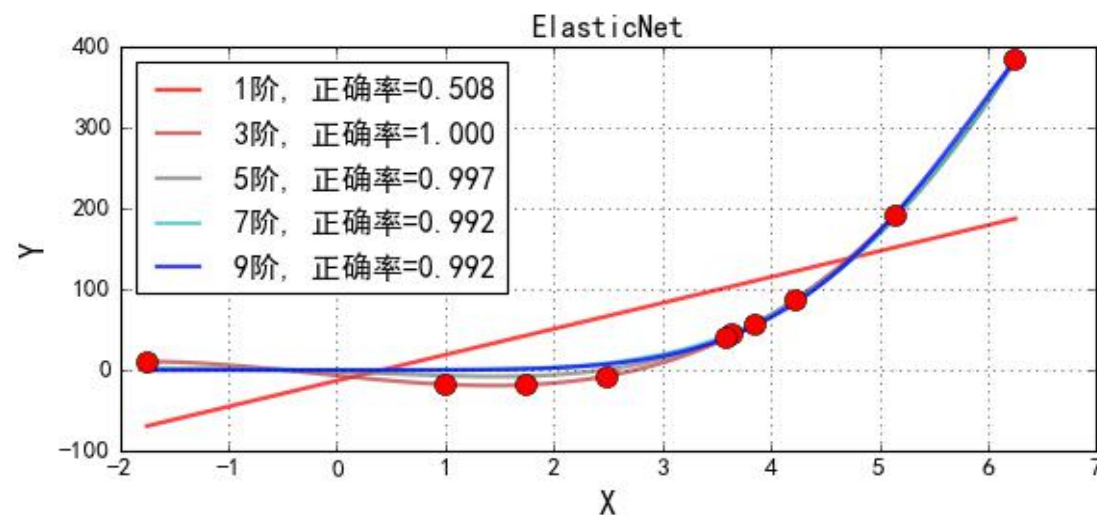
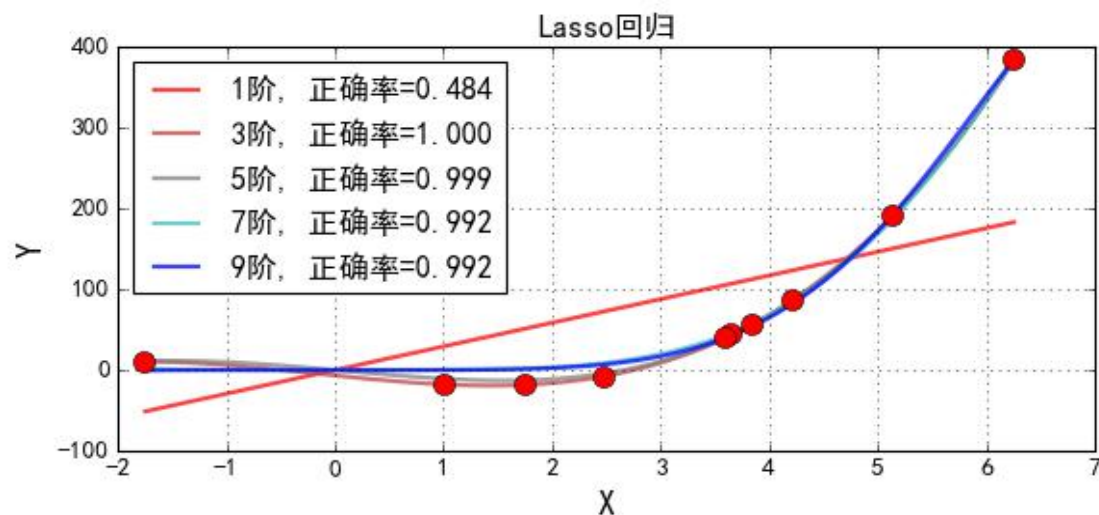
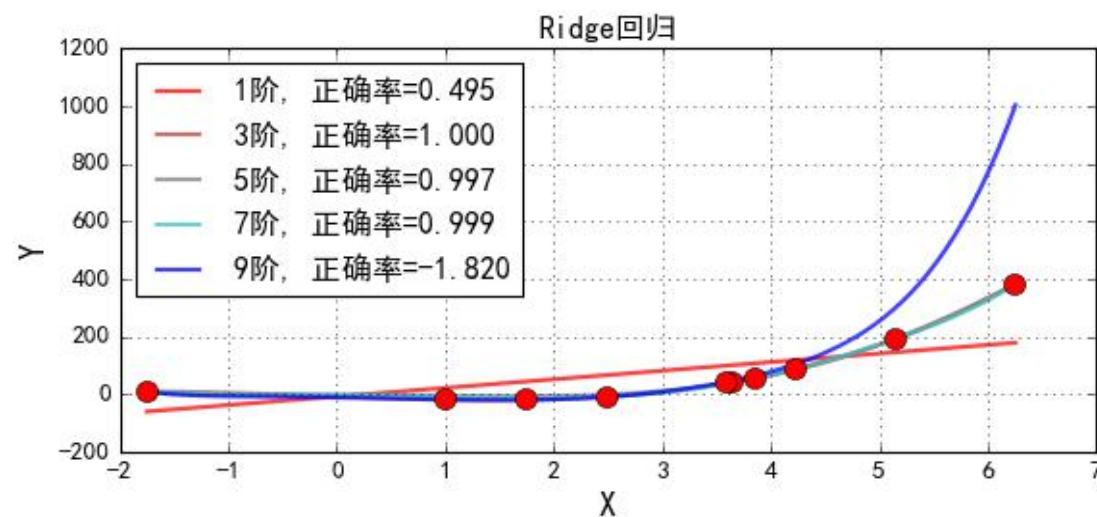
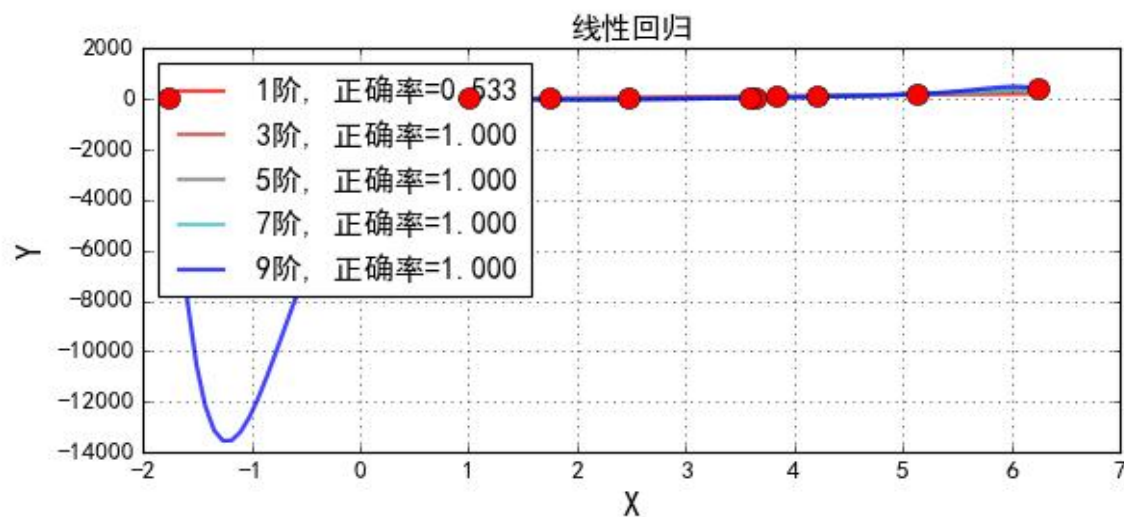
- 同时使用L1正则和L2正则的线性回归模型就称为Elastic Net算法(弹性网络算法)

$$\begin{cases} \lambda > 0 \\ p \in [0, 1] \end{cases}$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \left(p \sum_{j=1}^n |\theta_j| + (1-p) \sum_{j=1}^n \theta_j^2 \right)$$

线性回归算法过拟合比较(一)

各种不同线性回归过拟合显示



线性回归算法过拟合比较(二)

```

线性回归:1阶,系数为: [-44.14102611 40.05964256]
线性回归:3阶,系数为: [-6.80525963 -13.743068 0.93453895 1.79844791]
线性回归:5阶,系数为: [-5.60899679 -14.80109301 0.75014858 2.11170671 -0.07724668 0.00566633]
线性回归:7阶,系数为: [-41.70721172 52.38570529 -29.56451338 -7.66322829 12.07162703 -3.86969096 0.53286096 -0.02725536]
线性回归:9阶,系数为: [-2465.58378507 6108.6381056 -5111.99327317 974.74973548 1078.89648247 -829.50276827 266.13230319 -45.71741
527 4.11582735 -0.15281063]

Ridge回归:1阶,系数为: [-6.71593385 29.79090057]
Ridge回归:3阶,系数为: [-6.7819845 -13.73679293 0.92827639 1.79920954]
Ridge回归:5阶,系数为: [-0.82920155 -1.07244754 -1.41803017 -0.93057536 0.88319116 -0.07073168]
Ridge回归:7阶,系数为: [-1.62586368 -2.18512108 -1.82690987 -2.27495708 0.98685071 0.30551091 -0.10988434 0.00846908]
Ridge回归:9阶,系数为: [-10.50566712 -6.12564342 -1.96421973 0.80200162 0.59148104 -0.23358235 0.20297017 -0.08109698 0.0132585
7 -0.00072184]

Lasso回归:1阶,系数为: [-0. 29.27359177]
Lasso回归:3阶,系数为: [-6.7688595 -13.75928024 0.93989323 1.79778598]
Lasso回归:5阶,系数为: [-0. -12.00109345 -0.50746853 1.74395236 0.07086952 -0.00583605]
Lasso回归:7阶,系数为: [-0. -0. -0. -0.08083315 0.19550746 0.03066137 -0.00020584 -0.00046928]
Lasso回归:9阶,系数为: [-0. -0. -0. -0. 0.04439727 0.05587113 0.00109023 -0.00021498 -0.00004479 -0.0000
0674]

ElasticNet:1阶,系数为: [-13.22089654 32.08359338]
ElasticNet:3阶,系数为: [-6.7688595 -13.75928024 0.93989323 1.79778598]
ElasticNet:5阶,系数为: [-1.65823671 -5.20271875 -1.26488859 0.94503683 0.2605984 -0.01683786]
ElasticNet:7阶,系数为: [-0. -0. -0. -0.15812511 0.22150166 0.02955069 -0.00040066 -0.00046568]
ElasticNet:9阶,系数为: [-0. -0. -0. -0. 0.05255118 0.05364699 0.00111995 -0.00020596 -0.00004365 -0.000
00667]

```

模型效果判断

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

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

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y})^2}$$

$$\bar{y} = \frac{1}{m} \sum_{i=1}^m y_i$$

模型效果判断

- MSE: 误差平方和, 越趋近于0表示模型越拟合训练数据。
- RMSE: MSE的平方根, 作用同MSE
- R^2 : 取值范围(负无穷,1], 值越大表示模型越拟合训练数据; 最优解是1; 当模型预测为随机值的时候, 有可能为负; 若预测值恒为样本期望, R^2 为0
- TSS: 总平方和TSS(Total Sum of Squares), 表示**样本之间**的差异情况, 是伪方差的m倍
- RSS: 残差平方和RSS (Residual Sum of Squares) , 表示预测值和样本值之间的差异情况, 是MSE的m倍

机器学习调参

- 在实际工作中，对于各种算法模型(线性回归)来讲，我们需要获取 θ 、 λ 、 p 的值； θ 的求解其实就是算法模型的求解，一般不需要开发人员参与(算法已经实现)，主要需要求解的是 λ 和 p 的值，这个过程就叫做**调参(超参)**
- 交叉验证：将训练数据分为多份，其中一份进行数据验证并获取最优的超参： λ 和 p ；比如：十折交叉验证、五折交叉验证(scikit-learn中默认)等

训练数据

验证数据

梯度下降案例 $y = f(x) = x^2$

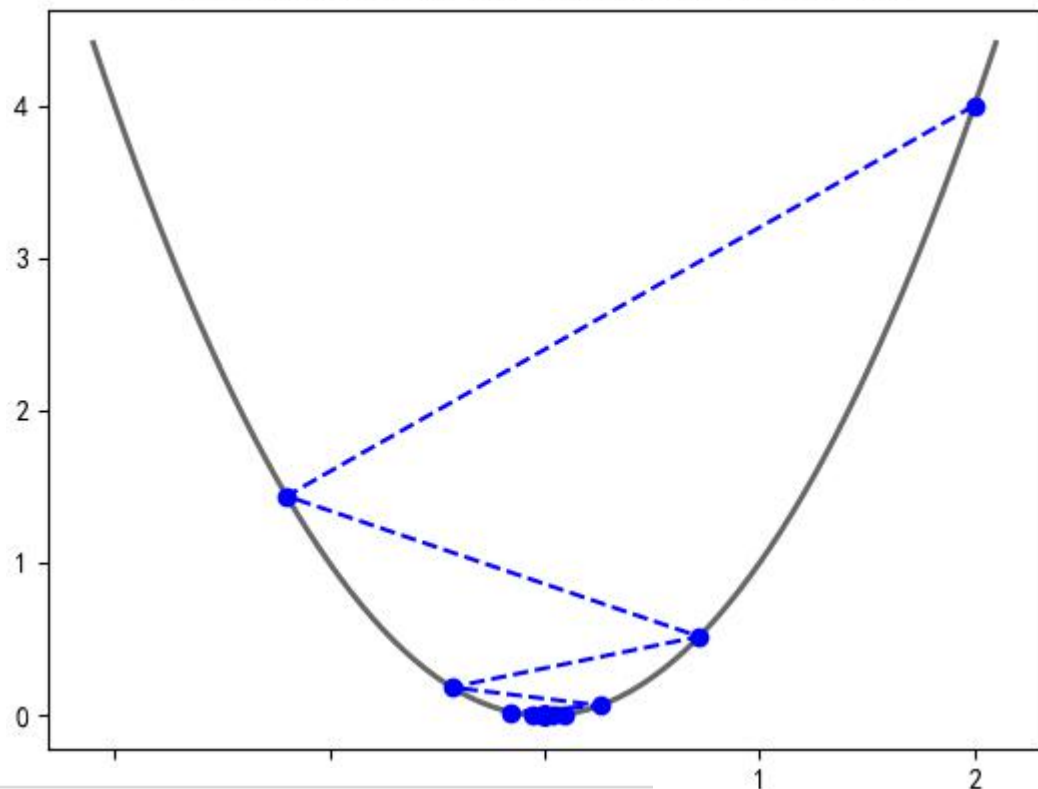
```
## 原函数
def f(x):
    return x ** 2
## 导数
def h(x):
    return 2 * x

X = []
Y = []

x = 2
step = 0.8
f_change = f(x)
f_current = f(x)
X.append(x)
Y.append(f_current)
while f_change > 1e-10:
    x = x - step * h(x)
    tmp = f(x)
    f_change = np.abs(f_current - tmp)
    f_current = tmp
    X.append(x)
    Y.append(f_current)
print u"最终结果为:", (x, f_current)
```

最终结果为: (-5.686057605985963e-06, 3.233125109859082e-11)

$y = x^2$ 函数求解最小值, 最终解为: $x=-0.00, y=0.00$



```
fig = plt.figure()
X2 = np.arange(-2.1, 2.15, 0.05)
Y2 = X2 ** 2

plt.plot(X2, Y2, '- ', color='#666666', linewidth=2)
plt.plot(X, Y, 'bo-')
plt.title(u'$y=x^2$函数求解最小值, 最终解为: x=%.2f, y=%.2f'
        % (x, f_current))
plt.show()
```

梯度下降案例

$$z = f(x, y) = x^2 + y^2$$

梯度下降法求解，最终解为：x=0.00, y=0.00, z=0.00

```
## 原函数
def f(x, y):
    return x ** 2 + y ** 2
## 偏函数
def h(t):
    return 2 * t

X = []
Y = []
Z = []

x = 2
y = 2
f_change = x ** 2 + y ** 2
f_current = f(x, y)
step = 0.1
X.append(x)
Y.append(y)
Z.append(f_current)
while f_change > 1e-10:
    x = x - step * h(x)
    y = y - step * h(y)
    f_change = f_current - f(x, y)
    f_current = f(x, y)
    X.append(x)
    Y.append(y)
    Z.append(f_current)
print u"最终结果为:", (x, y)
```

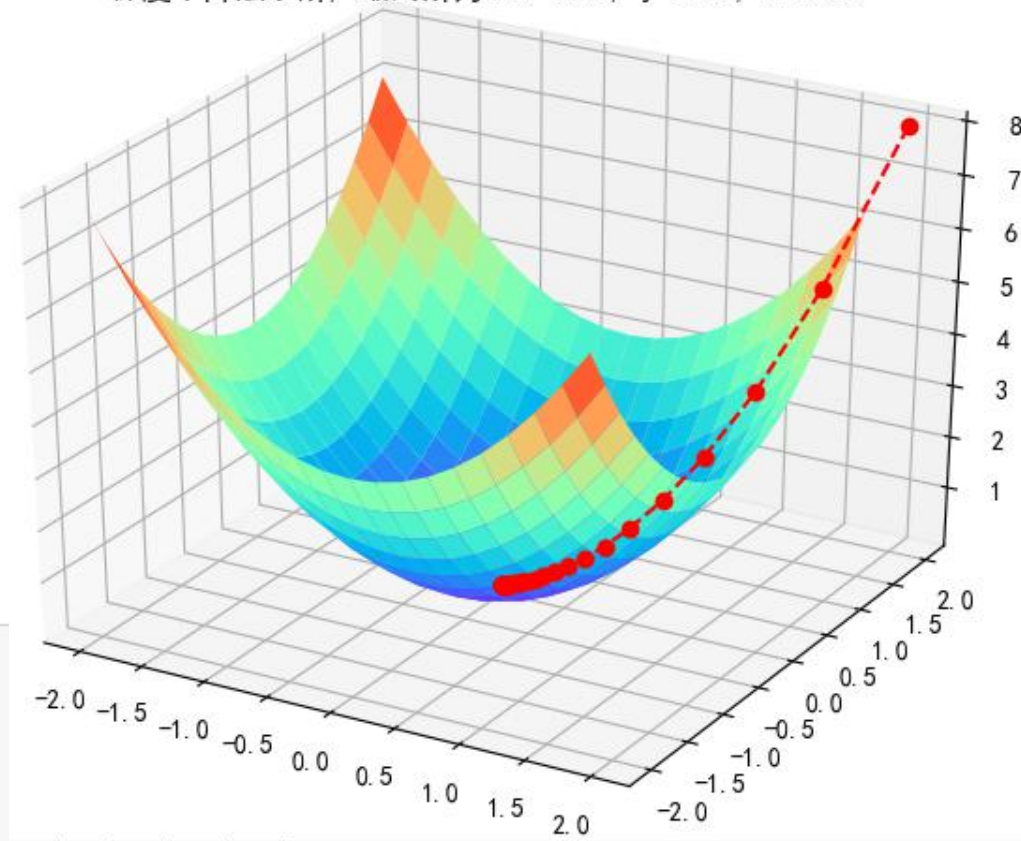
最终结果为: (9.353610478917782e-06, 9

```
fig = plt.figure()
ax = Axes3D(fig)
X2 = np.arange(-2, 2, 0.2)
Y2 = np.arange(-2, 2, 0.2)
X2, Y2 = np.meshgrid(X2, Y2)
Z2 = X2 ** 2 + Y2 ** 2

ax.plot_surface(X2, Y2, Z2, rstride=1, cstride=1, cmap='rainbow')
ax.plot(X, Y, Z, 'ro—')

ax.set_title(u'梯度下降法求解，最终解为：x=%.2f, y=%.2f, z=%.2f' % (x, y, f_current))

plt.show()
```

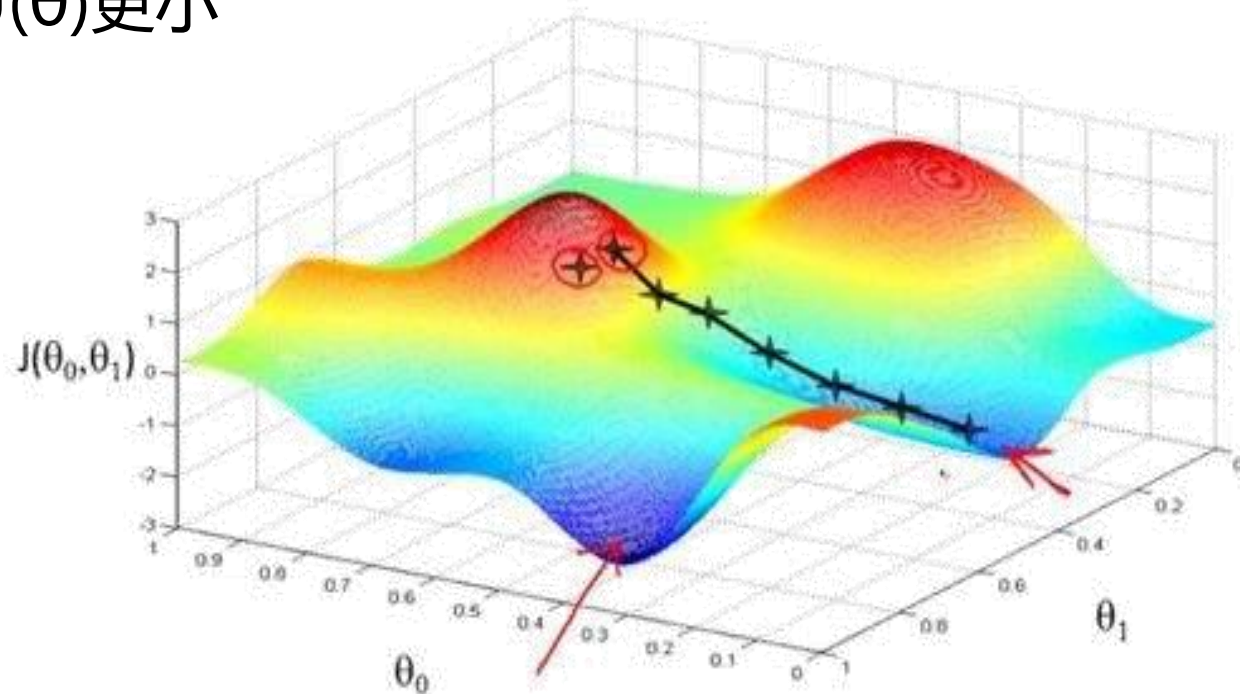


梯度下降算法

- 目标函数 θ 求解 $J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- 初始化 θ (随机初始化, 可以初始为0)
- 沿着负梯度方向迭代, 更新后的 θ 使 $J(\theta)$ 更小

$$\theta = \theta - \alpha \bullet \frac{\partial J(\theta)}{\partial \theta}$$

◆ α : 学习率、步长



梯度方向

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

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_{\theta}(x) - y) x_j \end{aligned}$$

批量梯度下降算法(BGD)

$$\frac{\partial}{\partial \theta_j} J(\theta) = (h_{\theta}(x) - y)x_j$$

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

$$\theta_j = \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)}))x_j^{(i)}$$

随机梯度下降算法(SGD)

$$\frac{\partial}{\partial \theta_j} J(\theta) = (h_{\theta}(x) - y)x_j$$

for i= 1 to m,{

$$\theta_j = \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)}))x_j^{(i)}$$

}

BGD和SGD算法比较

- SGD速度比BGD快(迭代次数少)
- SGD在某些情况下(全局存在多个相对最优解/ $J(\theta)$ 不是一个二次), SGD有可能跳出某些小的局部最优解, 所以不会比BGD坏, SGD在收敛的位置会存在 $J(\theta)$ 函数波动的情况。
- BGD一定能够得到一个局部最优解(在线性回归模型中一定是得到一个全局最优解), SGD由于随机性的存在可能导致最终结果比BGD的差
- **注意: 优先选择SGD**

小批量梯度下降法(MBGD)

- 如果即需要保证算法的训练过程比较快，又需要保证最终参数训练的准确率，而这正是小批量梯度下降法（Mini-batch Gradient Descent，简称MBGD）的初衷。MBGD中不是每拿一个样本就更新一次梯度，而且拿b个样本(b一般为10)的平均梯度作为更新方向。

for i= 1 to m/10,{

$$\theta_j = \theta_j + \alpha \sum_{k=i}^{i+10} \left(y^{(k)} - h_{\theta}(x^{(k)}) \right) x_j^{(k)}$$

}

梯度下降法

- 由于梯度下降法中负梯度方向作为变量的变化方向，所以有可能导致最终求解的值是局部最优解，所以在使用梯度下降的时候，一般需要进行一些调优策略：
 - ◆ **学习率的选择**：学习率过大，表示每次迭代更新的时候变化比较大，有可能会跳过最优解；学习率过小，表示每次迭代更新的时候变化比较小，就会导致迭代速度过慢，很长时间都不能结束；
 - ◆ **算法初始参数值的选择**：初始值不同，最终获得的最小值也有可能不同，因为梯度下降法求解的是局部最优解，所以一般情况下，选择多次不同初始值运行算法，并最终返回损失函数最小情况下的结果值；
 - ◆ **标准化**：由于样本不同特征的取值范围不同，可能会导致在各个不同参数上迭代速度不同，为了减少特征取值的影响，可以将特征进行标准化操作。

梯度下降法

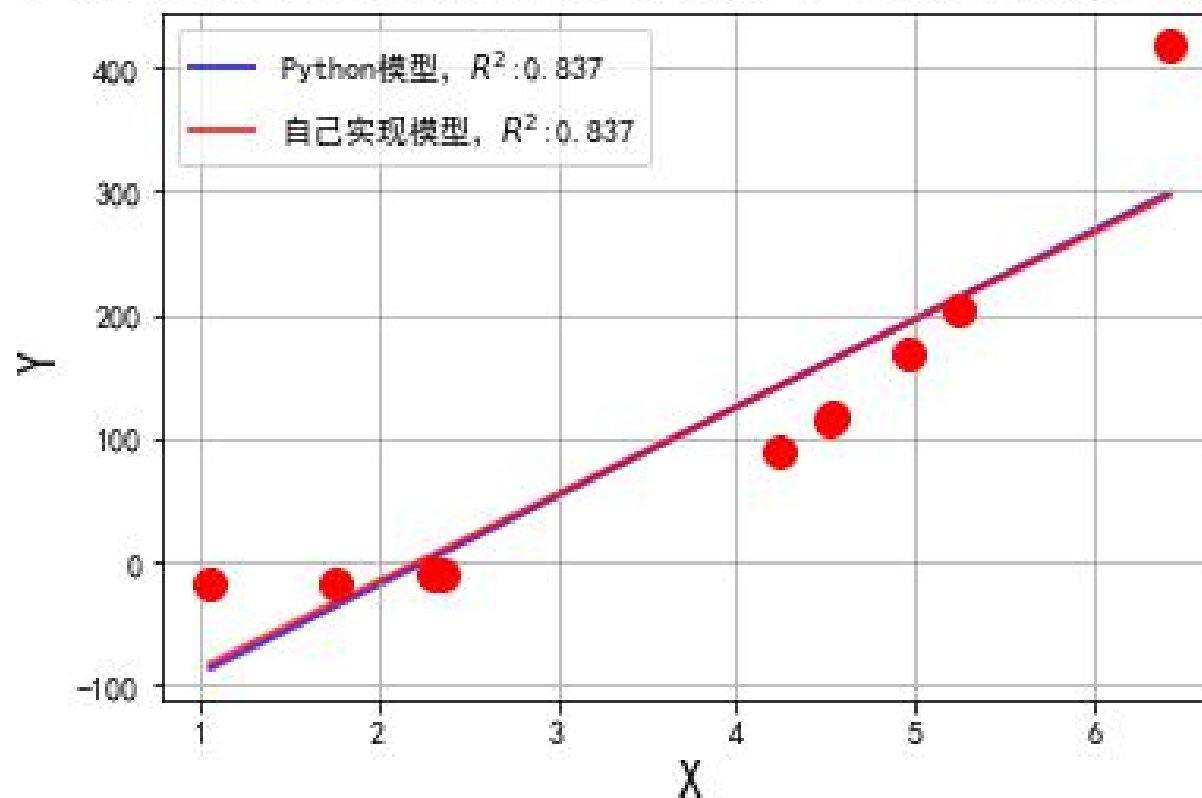
■ BGD、SGD、MBGD的区别：

- ◆ 当样本量为 m 的时候，每次迭代BGD算法中对于参数值更新一次，SGD算法中对于参数值更新 m 次，MBGD算法中对于参数值更新 m/n 次，相对来讲SGD算法的更新速度最快；
- ◆ SGD算法中对于每个样本都需要更新参数值，当样本值不太正常的时候，就有可能导致本次的参数更新会产生相反的影响，也就是说SGD算法的结果并不是完全收敛的，而是在收敛结果处波动的；
- ◆ SGD算法是每个样本都更新一次参数值，所以SGD算法特别适合样本数据量大的情况以及在线机器学习(Online ML)。

回归算法案例：基于梯度下降法实现线性回归算法(作业)

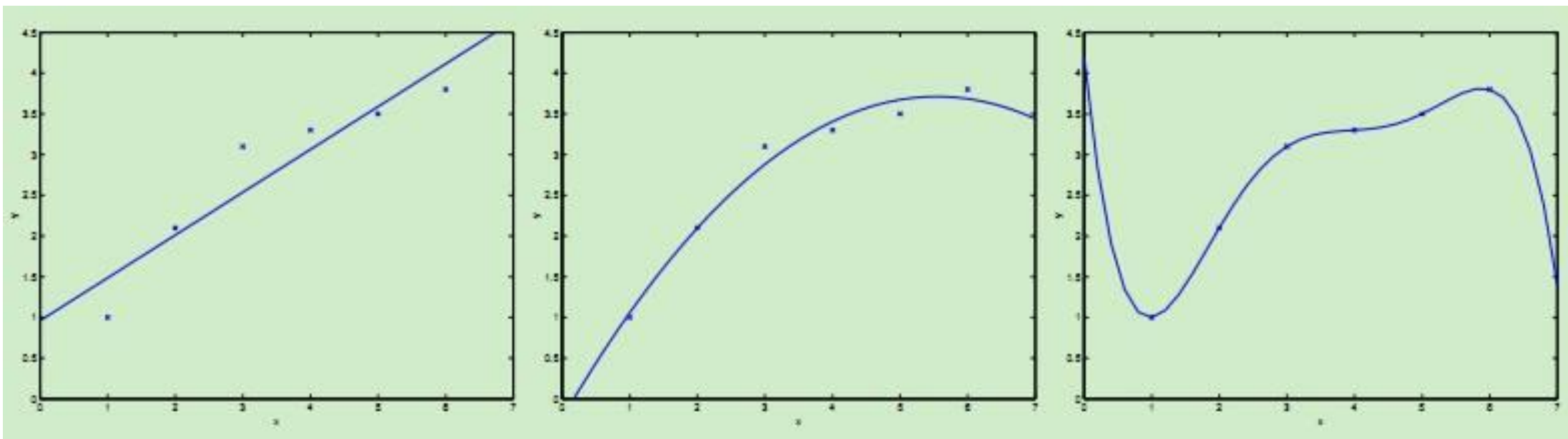
- 基于梯度下降法编写程序实现回归算法，并自行使用模拟数据进行测试，同时对同样的模拟数据进行两种算法的比较(python sklearn LinearRegression和自己实现的线性回归算法)

自定义的线性模型和模块中的线性模型比较



线性回归的扩展

- 线性回归针对的是 θ 而言是一种，对于样本本身而言，样本可以是非线性的
- 也就是说最终得到的函数 $f: x \rightarrow y$ ；函数 $f(x)$ 可以是非线性的，比如：曲线等



$$y = \theta_0 + \theta_1 x$$

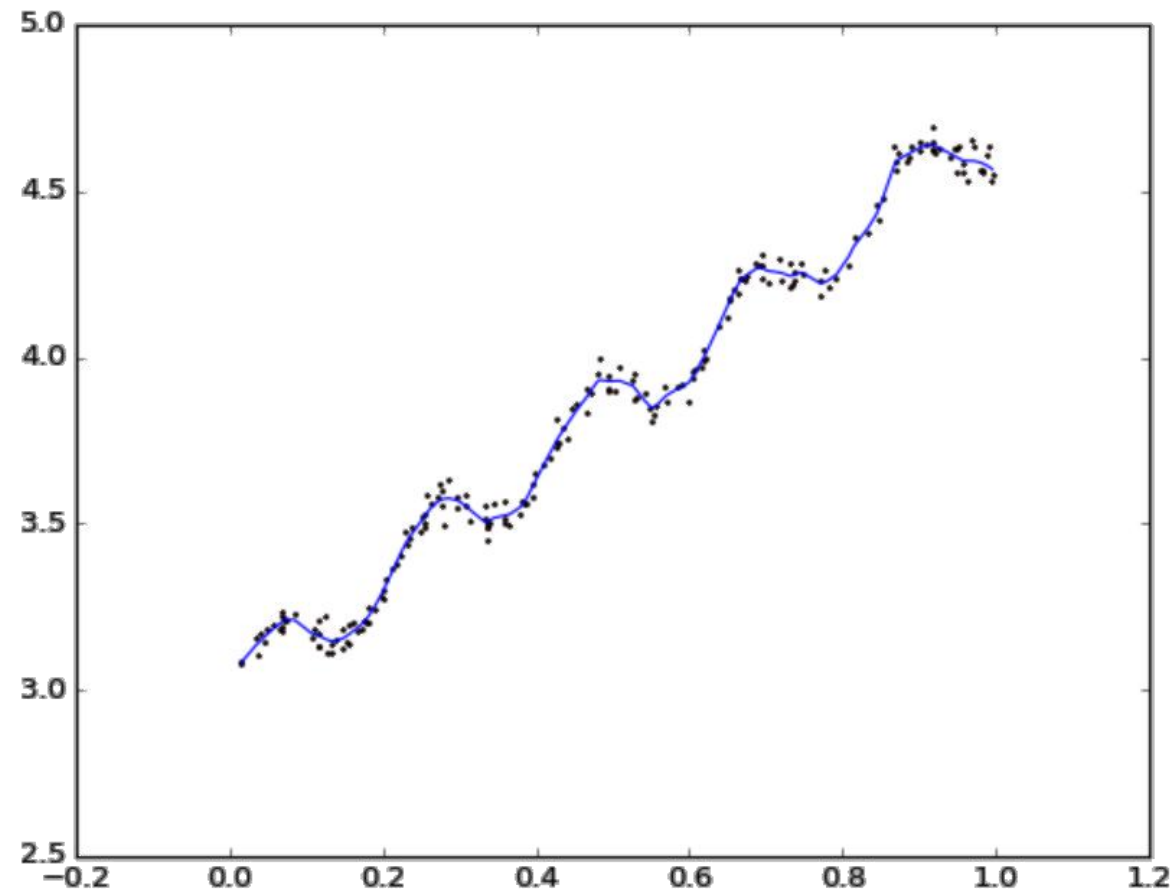
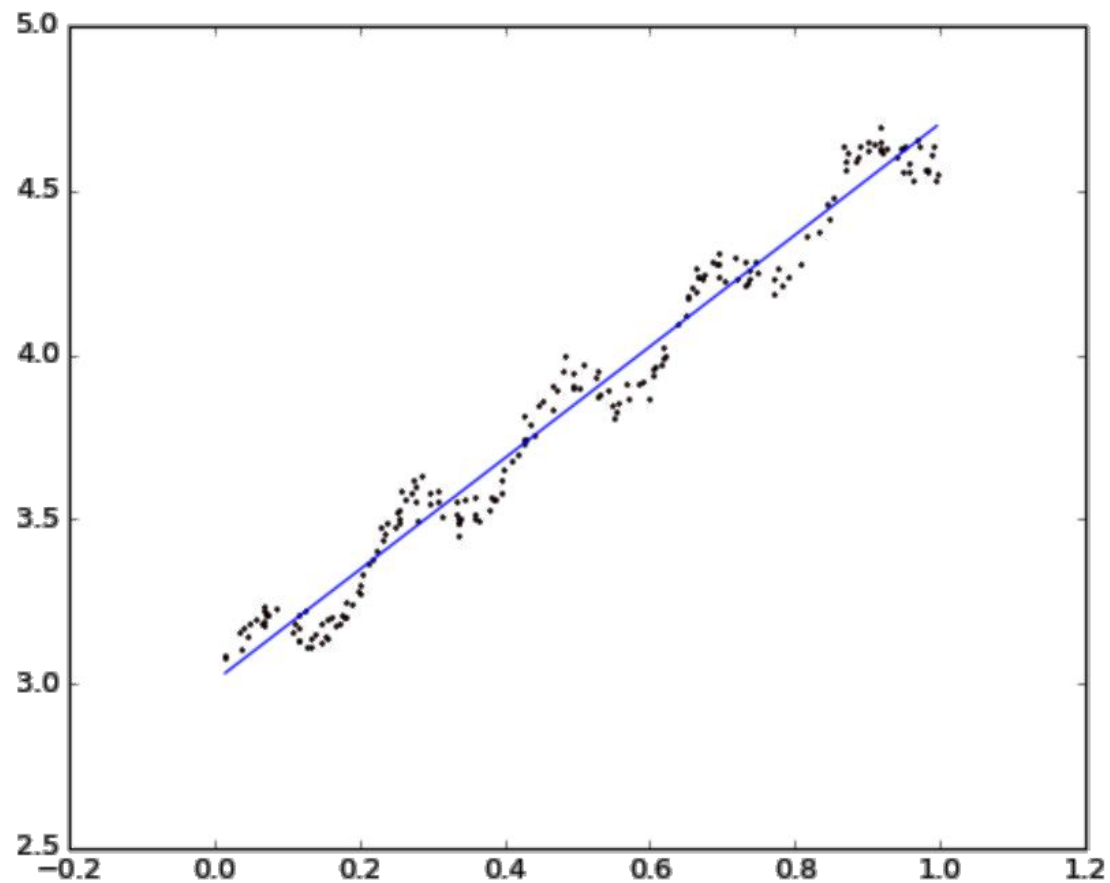
$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

线性回归总结

- 算法模型：线性回归(Linear)、岭回归(Ridge)、LASSO回归、Elastic Net
- 正则化：L1-norm、L2-norm
- 损失函数/目标函数： $J(\theta) = \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \rightarrow \min_{\theta} J(\theta)$
- θ 求解方式：最小二乘法(直接计算，目标函数是平方和损失函数)、梯度下降(BGD\SGD\MBGD)

局部加权回归-直观理解



局部加权回归-损失函数

■ 普通线性回归损失函数：

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

■ 局部加权回归损失函数：

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

局部加权回归-权重值设置

- $w^{(i)}$ 是权重，它根据要预测的点与数据集中的点的距离来为数据集中的点赋权值。

当某点离要预测的点越远，其权重越小，否则越大。常用值选择公式为：

$$w^{(i)} = \exp \left(- \frac{(y^{(i)} - \hat{y})^2}{2k^2} \right)$$

- 该函数称为指数衰减函数，其中k为波长参数，它控制了权值随距离下降的速率
- 注意：使用该方式主要应用到样本之间的相似性考虑，主要内容在SVM中再考虑(核函数)

回归算法综合案例(二): 波士顿房屋租赁价格预测(作业)

- 基于波士顿房屋租赁数据进行房屋租赁价格预测模型构建, 分别使用Lasso回归、Ridge回两种回归算法构建模型, 并分别构建1/2/3阶算法中的最优算法(参数), 并比较这两种回归算法的效果; 另外使用lasso回归算法做特征选择(选择特征参数不为0的属性数据作为最终的特征属性, 用这个选择出来的特征属性矩阵做Ridge回归)
- ◆ 数据下载url: <http://archive.ics.uci.edu/ml/datasets/Housing>(现在没法下载啦)

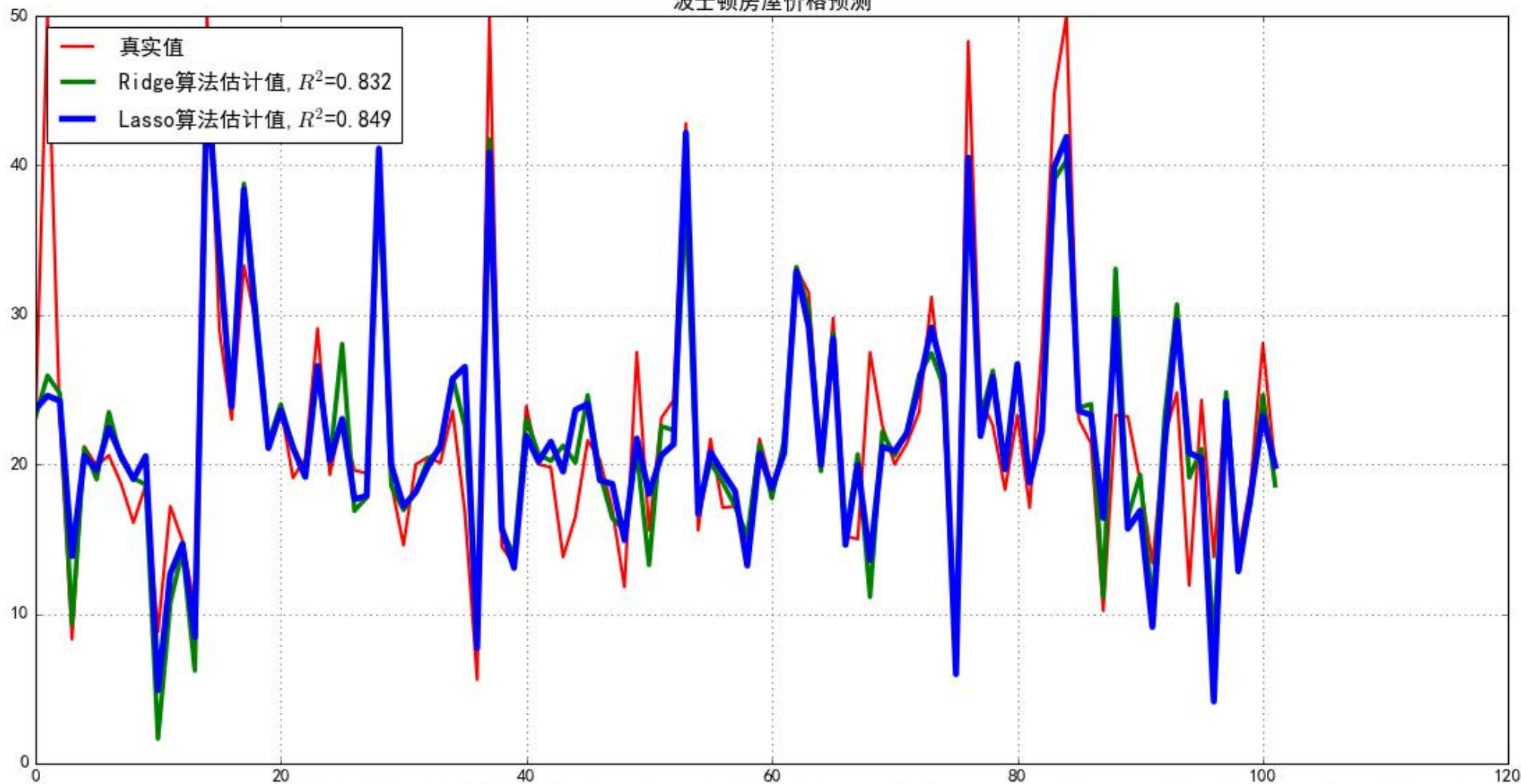
Attribute Information:

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per \$10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in \$1000's

0.31533	0.00	6.200	0	0.5040	8.2660	78.30	2.8944	8	307.0	17.40	385.05	4.14	44.80
0.52693	0.00	6.200	0	0.5040	8.7250	83.00	2.8944	8	307.0	17.40	382.00	4.63	50.00
0.38214	0.00	6.200	0	0.5040	8.0400	86.50	3.2157	8	307.0	17.40	387.38	3.13	37.60
0.41238	0.00	6.200	0	0.5040	7.1630	79.90	3.2157	8	307.0	17.40	372.08	6.36	31.60
0.29819	0.00	6.200	0	0.5040	7.6860	17.00	3.3751	8	307.0	17.40	377.51	3.92	46.70
0.44178	0.00	6.200	0	0.5040	6.5520	21.40	3.3751	8	307.0	17.40	380.34	3.76	31.50
0.53700	0.00	6.200	0	0.5040	5.9810	68.10	3.6715	8	307.0	17.40	378.35	11.65	24.30
0.46296	0.00	6.200	0	0.5040	7.4120	76.90	3.6715	8	307.0	17.40	376.14	5.25	31.70
0.57529	0.00	6.200	0	0.5070	8.3370	73.30	3.8384	8	307.0	17.40	385.91	2.47	41.70
0.33147	0.00	6.200	0	0.5070	8.2470	70.40	3.6519	8	307.0	17.40	378.95	3.95	48.30
0.44791	0.00	6.200	1	0.5070	6.7260	66.50	3.6519	8	307.0	17.40	360.20	8.05	29.00
0.33045	0.00	6.200	0	0.5070	6.0860	61.50	3.6519	8	307.0	17.40	376.75	10.88	24.00
0.52058	0.00	6.200	1	0.5070	6.6310	76.50	4.1480	8	307.0	17.40	388.45	9.54	25.10
0.51183	0.00	6.200	0	0.5070	7.3580	71.60	4.1480	8	307.0	17.40	390.07	4.73	31.50
0.08244	30.00	4.930	0	0.4280	6.4810	18.50	6.1899	6	300.0	16.60	379.41	6.36	23.70
0.09252	30.00	4.930	0	0.4280	6.6060	42.20	6.1899	6	300.0	16.60	383.78	7.37	23.30
0.11220	20.00	4.020	0	0.4280	6.9070	54.20	6.2261	6	200.0	16.60	201.25	11.20	22.00

回归算法综合案例(二): 波士顿房屋租赁价格预测

波士顿房屋价格预测



Ridge算法:最优参数: {'poly_degree': 3, 'linear_fit_intercept': True, 'poly_include_bias': True, 'poly_interaction_only': True}
 Ridge算法:R值=0.832
 Lasso算法:最优参数: {'poly_degree': 3, 'linear_fit_intercept': False, 'poly_include_bias': True, 'poly_interaction_only': True}
 Lasso算法:R值=0.849

回归算法综合案例(二): 波士顿房屋租赁价格预测

```
## 模型训练 ==> 单个Lasso模型 (一阶特征选择) (2参数给定1阶情况的最优参数)
model = Pipeline([
    ('ss', StandardScaler()),
    ('poly', PolynomialFeatures(degree=1, include_bias=True, interaction_only=True)),
    ('linear', LassoCV(alphas=np.logspace(-3, 1, 20), fit_intercept=False))
])
# 模型训练
model.fit(x_train, y_train)

# 模型评测
## 数据输出
print "参数:", zip(names, model.get_params('linear')['linear'].coef_)
print "截距:", model.get_params('linear')['linear'].intercept_
```

参数: [('CRIM', 22.600592809201991), ('ZN', -0.93534557687414488), ('INDUS', 1.0202352850146854), ('CHAS', -0.0), ('NOX', 0.5948313841546149), ('RM', -1.8002644875942369), ('AGE', 2.5861907995357281), ('DIS', -0.064956108249539249), ('RAD', -2.8017533936656509), ('TAX', 1.9343329692037559), ('PTRATIO', -1.7218677875512203), ('B', -2.2762334623842988), ('LSTAT', 0.70288003005515387)]
截距: 0.0

CHAS列的数据对于LassoCV模型而言无用，所以在进行实际模型构建的时候，可以不考虑该特征

回归算法综合案例(三): 葡萄酒质量预测

- 基于葡萄酒数据进行葡萄酒质量预测模型构建, 分别使用线性回归、Lasso回归、Ridge回归、Elastic Net四类回归算法构建模型(并分别测试1/2/3阶), 并比较这些回归算法的效果

◆ 数据下载url: <http://archive.ics.uci.edu/ml/datasets/Wine+Quality>

Attribute Information:

For more information, read [Cortez et al., 2009].

Input variables (based on physicochemical tests):

- 1 - fixed acidity
- 2 - volatile acidity
- 3 - citric acid
- 4 - residual sugar
- 5 - chlorides
- 6 - free sulfur dioxide
- 7 - total sulfur dioxide
- 8 - density
- 9 - pH
- 10 - sulphates
- 11 - alcohol

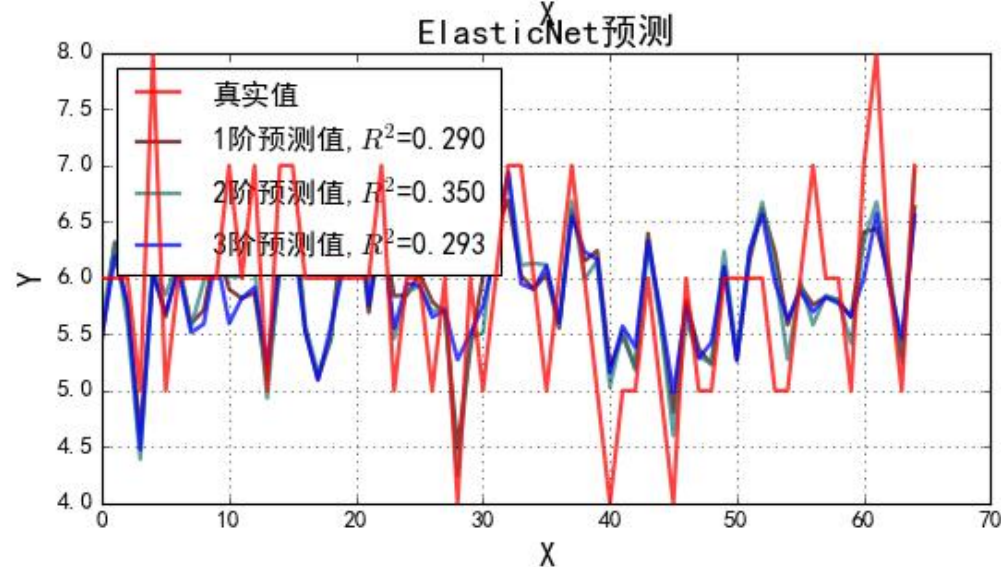
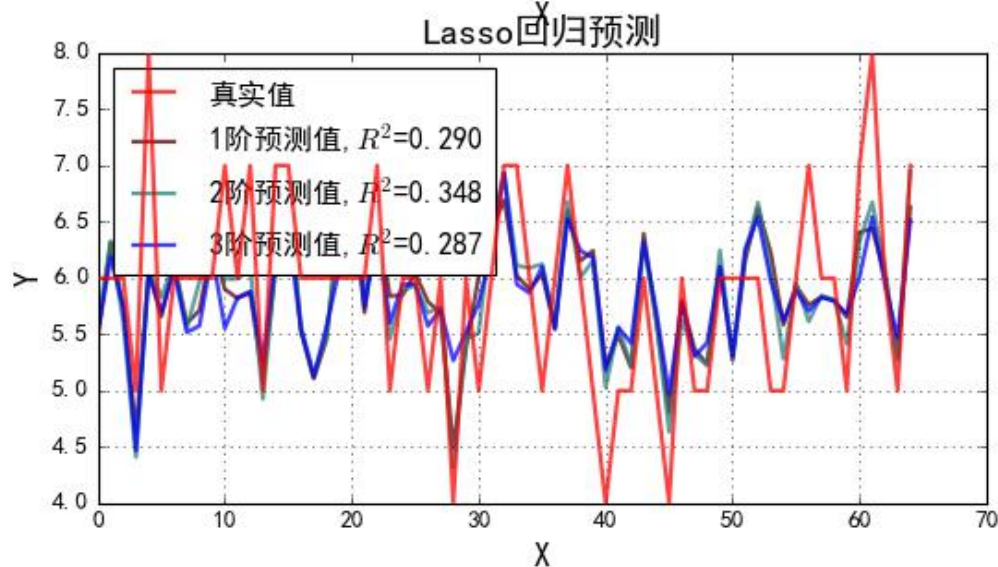
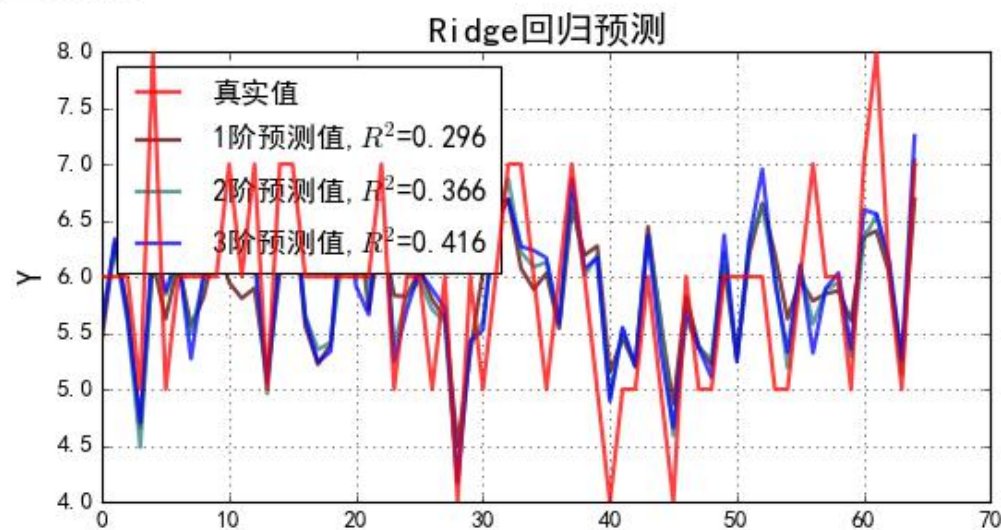
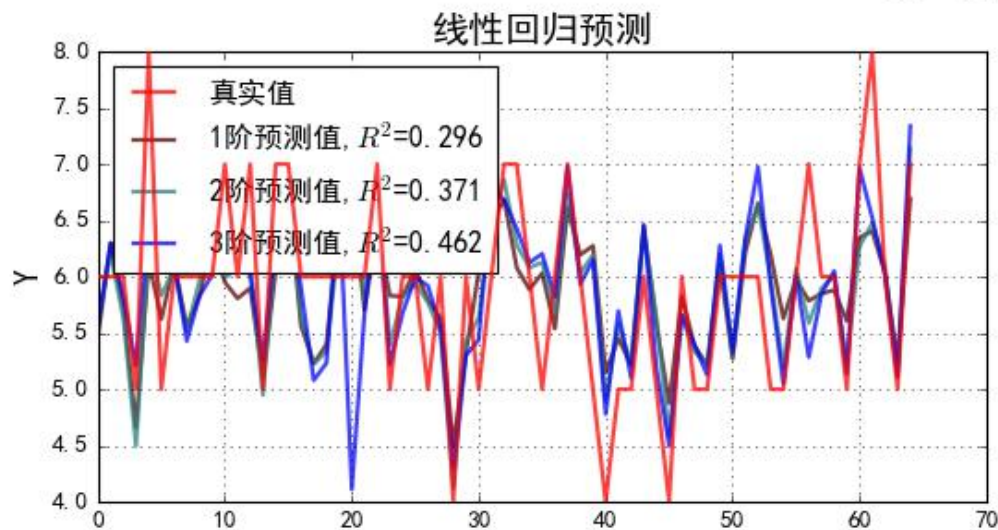
Output variable (based on sensory data):

- 12 - quality (score between 0 and 10)

2	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
3	7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5
4	7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;9.8;5
5	11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58;9.8;6
6	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7	7.4;0.66;0;1.8;0.075;13;40;0.9978;3.51;0.56;9.4;5
8	7.9;0.6;0.06;1.6;0.069;15;59;0.9964;3.3;0.46;9.4;5
9	7.3;0.65;0;1.2;0.065;15;21;0.9946;3.39;0.47;10;7
10	7.8;0.58;0.02;2;0.073;9;18;0.9968;3.36;0.57;9.5;7
11	7.5;0.5;0.36;6.1;0.071;17;102;0.9978;3.35;0.8;10.5;5

回归算法综合案例(三): 葡萄酒质量预测

葡萄酒质量预测



Logistic回归

■ Logistic/sigmoid函数 $p = h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$

$$g(z) = \frac{1}{1 + e^{-z}}$$

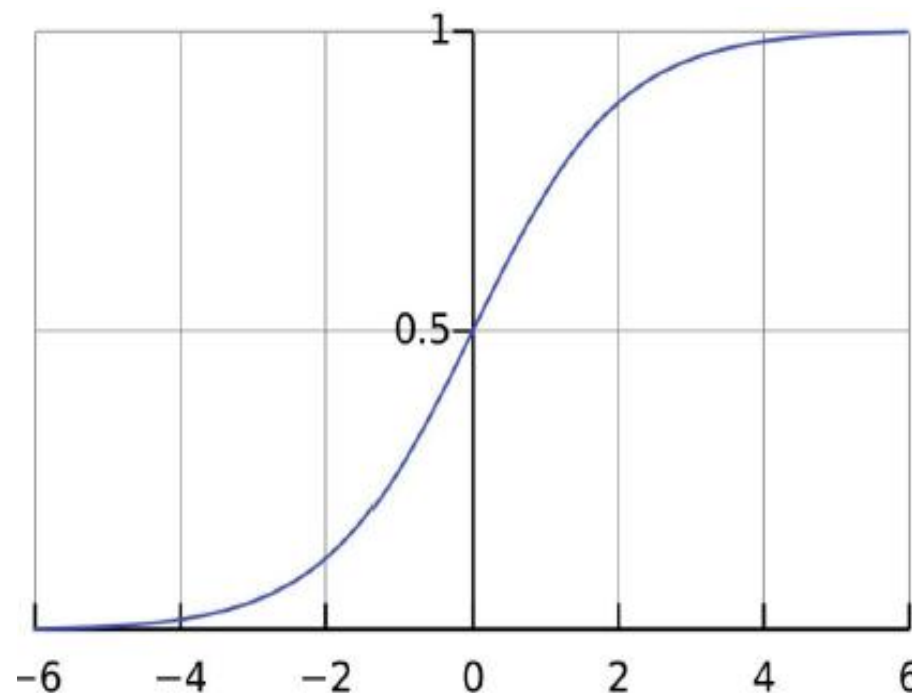
$$g'(z) = \left(\frac{1}{1 + e^{-z}} \right)' = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$= \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} = \frac{1}{1 + e^{-z}} \cdot \left(1 - \frac{1}{1 + e^{-z}} \right)$$

$$= g(z) \cdot (1 - g(z))$$

$$y = \begin{cases} 0 \\ 1 \end{cases}$$

$$\hat{y} = \begin{cases} 1, P(\hat{y} = 1) > p \\ 0, P(\hat{y} = 0) > p \end{cases}$$



Logistic回归及似然函数

- 假设: $P(y = 1 | x; \theta) = h_{\theta}(x)$
 $P(y = 0 | x; \theta) = 1 - h_{\theta}(x)$

	y=1	y=0
p(y x)	θ	$1-\theta$

$$P(y | x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{(1-y)}$$

- 似然函数: $L(\theta) = p(\vec{y} | X; \theta) = \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta)$
$$= \prod_{i=1}^m (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{(1-y^{(i)})}$$

- 对数似然函数: $\ell(\theta) = \log L(\theta) = \sum_{i=1}^m (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$

最大似然/极大似然函数的随机梯度

■ 对数似然函数: $\ell(\theta) = \log L(\theta) = \sum_{i=1}^m (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})))$

$$\frac{\partial \ell(\theta)}{\partial \theta_j} = \sum_{i=1}^m \left(\frac{y^{(i)}}{h_{\theta}(x^{(i)})} - \frac{1 - y^{(i)}}{1 - h_{\theta}(x^{(i)})} \right) \cdot \frac{\partial h_{\theta}(x^{(i)})}{\partial \theta_j}$$

$$= \sum_{i=1}^m \left(\frac{y^{(i)}}{g(\theta^T x^{(i)})} - \frac{1 - y^{(i)}}{1 - g(\theta^T x^{(i)})} \right) \cdot \frac{\partial g(\theta^T x^{(i)})}{\partial \theta_j}$$

$$= \sum_{i=1}^m \left(\frac{y^{(i)}}{g(\theta^T x^{(i)})} - \frac{1 - y^{(i)}}{1 - g(\theta^T x^{(i)})} \right) \cdot g(\theta^T x^{(i)}) (1 - g(\theta^T x^{(i)})) \cdot \frac{\partial \theta^T x^{(i)}}{\partial \theta_j}$$

$$= \sum_{i=1}^m (y^{(i)} (1 - g(\theta^T x^{(i)})) - (1 - y^{(i)}) g(\theta^T x^{(i)})) \cdot x_j^{(i)} = \sum_{i=1}^m (y^{(i)} - g(\theta^T x^{(i)})) \cdot x_j^{(i)}$$

θ参数求解

- Logistic回归θ参数的求解过程为(类似梯度下降方法):

$$\theta_j = \theta_j + \alpha \sum_{i=1}^m \left(y^{(i)} - h_{\theta} \left(x^{(i)} \right) \right) x_j^{(i)}$$

$$\theta_j = \theta_j + \alpha \left(y^{(i)} - h_{\theta} \left(x^{(i)} \right) \right) x_j^{(i)}$$

极大似然估计与Logistic回归损失函数

$$L(\theta) = \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta) = \prod_{i=1}^m p_i^{y^{(i)}} (1 - p_i)^{1-y^{(i)}} \quad p_i = h_{\theta}(x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

$$\ell(\theta) = \ln L(\theta) = \sum_{i=1}^m \ln \left[p_i^{y^{(i)}} (1 - p_i)^{1-y^{(i)}} \right]$$

$$loss = -\ell(\theta)$$

$$= -\sum_{i=1}^m [y^{(i)} \ln(p_i) + (1 - y^{(i)}) \ln(1 - p_i)]$$

$$= \sum_{i=1}^m [-y^{(i)} \ln(h_{\theta}(x^{(i)})) - (1 - y^{(i)}) \ln(1 - h_{\theta}(x^{(i)}))]$$

Logistic案例(一): 乳腺癌分类

■ 基于病理数据进行乳腺癌预测(复发4/正常2), 使用Logistic算法构建模型

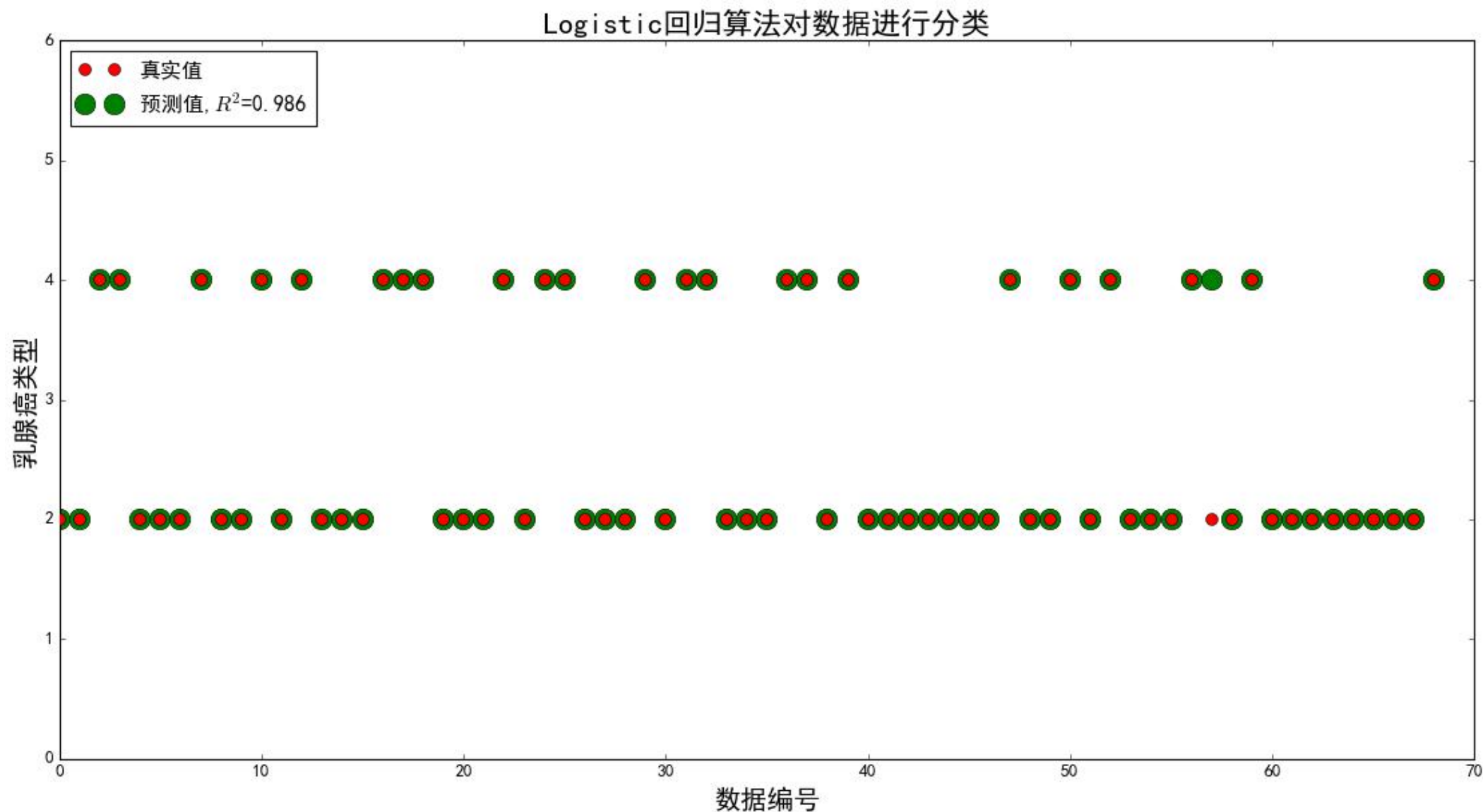
◆ 数据来源:

<http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29>

#	Attribute	Domain
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 - 10
11.	Class:	(2 for benign, 4 for malignant)

```
1000025, 5, 1, 1, 1, 2, 1, 3, 1, 1, 2
1002945, 5, 4, 4, 5, 7, 10, 3, 2, 1, 2
1015425, 3, 1, 1, 1, 2, 2, 3, 1, 1, 2
1016277, 6, 8, 8, 1, 3, 4, 3, 7, 1, 2
1017023, 4, 1, 1, 3, 2, 1, 3, 1, 1, 2
1017122, 8, 10, 10, 8, 7, 10, 9, 7, 1, 4
1018099, 1, 1, 1, 1, 2, 10, 3, 1, 1, 2
1018561, 2, 1, 2, 1, 2, 1, 3, 1, 1, 2
1033078, 2, 1, 1, 1, 2, 1, 1, 1, 5, 2
1033078, 4, 2, 1, 1, 2, 1, 2, 1, 1, 2
1035283, 1, 1, 1, 1, 1, 1, 3, 1, 1, 2
```

Logistic案例(一): 乳腺癌分类



Softmax回归

- softmax回归是logistic回归的一般化，适用于K分类的问题，第k类的参数为向量 θ_k ，组成的二维矩阵为 $\theta_{k \times n}$ ；
- softmax函数的本质就是将一个K维的任意实数向量压缩（映射）成另一个K维的实数向量，其中向量中的每个元素取值都介于（0，1）之间。
- softmax回归概率函数为：

$$p(y = k | x; \theta) = \frac{e^{\theta_k^T x}}{\sum_{l=1}^K e^{\theta_l^T x}}, k = 1, 2, \dots, K$$

Softmax算法原理

$$p(y = k | x; \theta) = \frac{e^{\theta_k^T x}}{\sum_{l=1}^K e^{\theta_l^T x}}, k = 1, 2, \dots, K$$

$$h_{\theta}(x) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \dots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \\ \dots \\ e^{\theta_k^T x} \end{bmatrix} \Rightarrow \theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2n} \\ \dots & \dots & \dots & \dots \\ \theta_{k1} & \theta_{k2} & \dots & \theta_{kn} \end{bmatrix}$$

Softmax算法损失函数

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k I(y^{(i)} = j) \ln \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right) \quad I(y^{(i)} = j) = \begin{cases} 1, & y^{(i)} = j \\ 0, & y^{(i)} \neq j \end{cases}$$

Softmax算法梯度下降法求解

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k I(y^{(i)} = j) \ln \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right)$$

$$I(y^{(i)} = j) = \begin{cases} 1, & y^{(i)} = j \\ 0, & y^{(i)} \neq j \end{cases}$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} - I(y^{(i)} = j) \ln \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right)$$

$$= \frac{\partial}{\partial \theta_j} - I(y^{(i)} = j) \left(\theta_j^T x^{(i)} - \ln \left(\sum_{l=1}^k e^{\theta_l^T x^{(i)}} \right) \right)$$

$$= -I(y^{(i)} = j) \left(1 - \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right) x^{(i)}$$

Softmax算法梯度下降法求解

$$\frac{\partial}{\partial \theta_j} J(\theta) = -I(y^{(i)} = j) \left(1 - \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right) x^{(i)}$$

$$\theta_j = \theta_j + \alpha \sum_{i=1}^m I(y^{(i)} = j) \left(1 - p(y^{(i)} = j | x^{(i)}; \theta) \right) x^{(i)}$$

$$\theta_j = \theta_j + \alpha I(y^{(i)} = j) \left(1 - p(y^{(i)} = j | x^{(i)}; \theta) \right) x^{(i)}$$

Softmax案例(一): 葡萄酒质量分类

- 基于葡萄酒数据进行葡萄酒质量预测模型构建, 使用Softmax算法构建模型, 并获取Softmax算法构建的模型效果(注意: 分成11类)

◆ 数据来源: <http://archive.ics.uci.edu/ml/datasets/Wine+Quality>

Attribute Information:

For more information, read [Cortez et al., 2009].

Input variables (based on physicochemical tests):

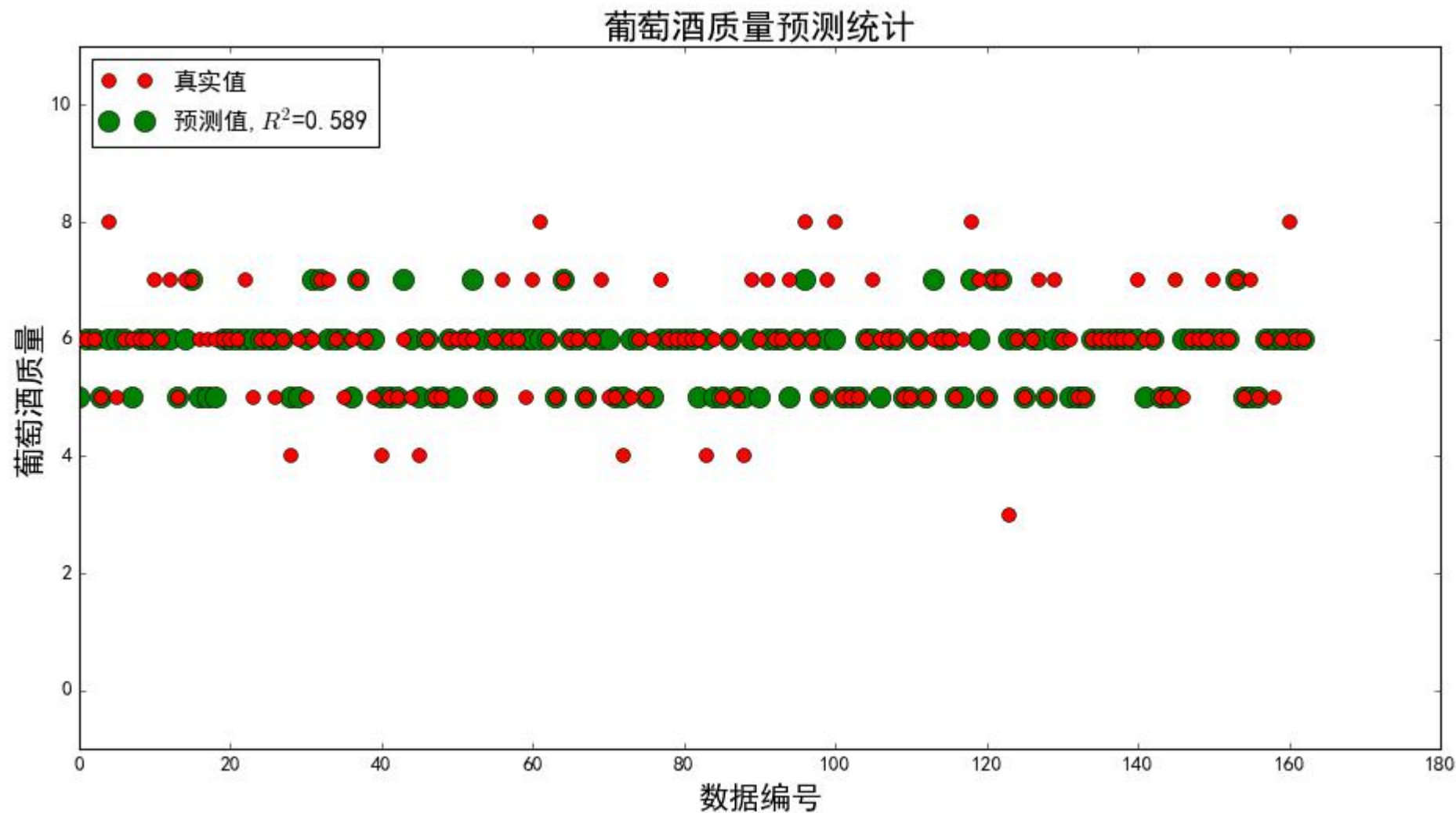
- 1 - fixed acidity
- 2 - volatile acidity
- 3 - citric acid
- 4 - residual sugar
- 5 - chlorides
- 6 - free sulfur dioxide
- 7 - total sulfur dioxide
- 8 - density
- 9 - pH
- 10 - sulphates
- 11 - alcohol

Output variable (based on sensory data):

- 12 - quality (score between 0 and 10)

2	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
3	7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5
4	7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;9.8;5
5	11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58;9.8;6
6	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7	7.4;0.66;0;1.8;0.075;13;40;0.9978;3.51;0.56;9.4;5
8	7.9;0.6;0.06;1.6;0.069;15;59;0.9964;3.3;0.46;9.4;5
9	7.3;0.65;0;1.2;0.065;15;21;0.9946;3.39;0.47;10;7
10	7.8;0.58;0.02;2;0.073;9;18;0.9968;3.36;0.57;9.5;7
11	7.5;0.5;0.36;6.1;0.071;17;102;0.9978;3.35;0.8;10.5;5

Softmax案例(一): 葡萄酒质量分类



分类问题综合案例(一): 信贷审批

- 基于信贷数据进行用户信贷分类, 使用Logistic算法和KNN算法构建模型, 并比较这两大类算法的效果

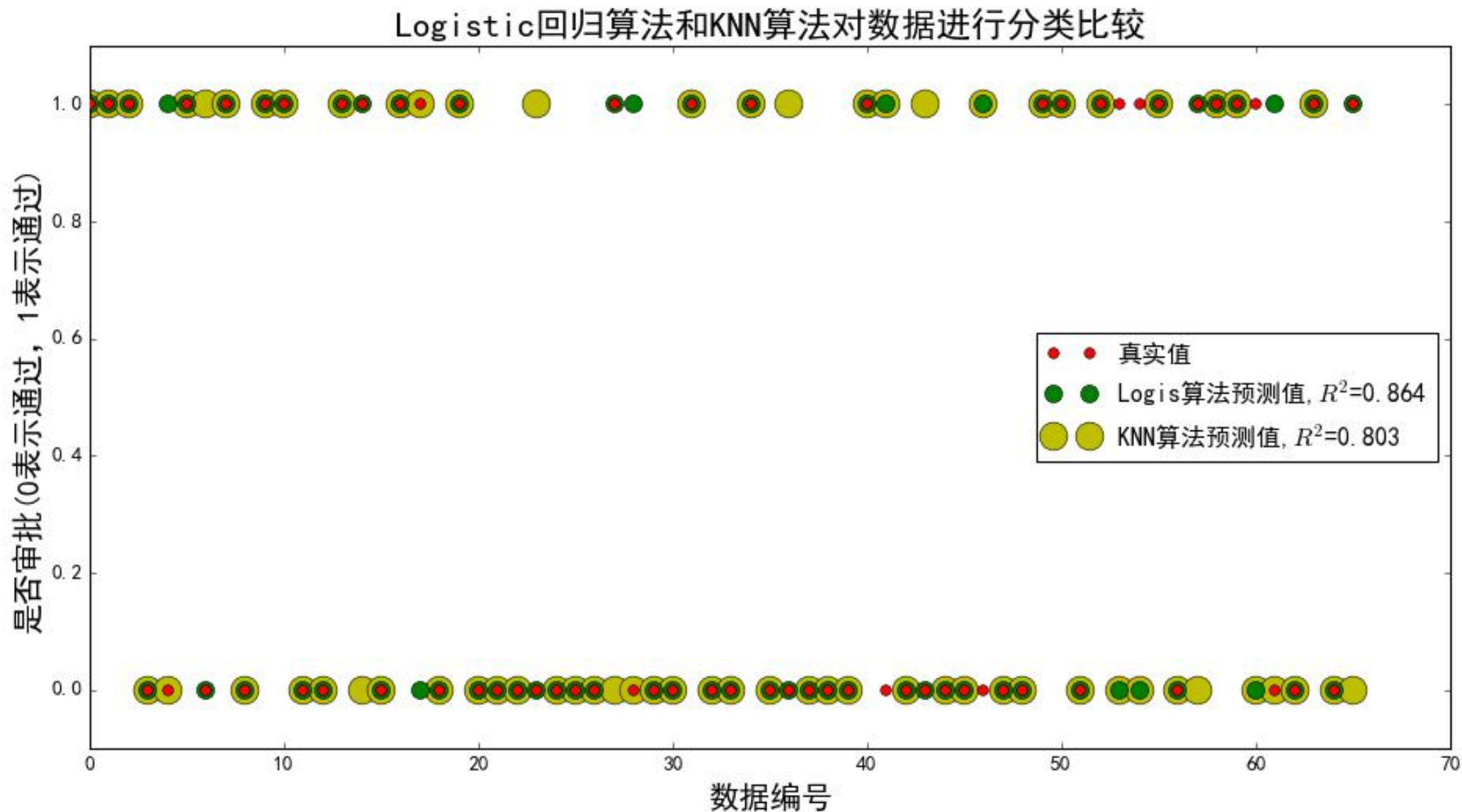
◆ 数据来源: <http://archive.ics.uci.edu/ml/datasets/Credit+Approval>

Attribute Information:

A1: b, a.
A2: continuous.
A3: continuous.
A4: u, y, l, t.
A5: g, p, gg.
A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7: v, h, bb, j, n, z, dd, ff, o.
A8: continuous.
A9: t, f.
A10: t, f.
A11: continuous.
A12: t, f.
A13: g, p, s.
A14: continuous.
A15: continuous.
A16: +, - (class attribute)

```
a, 40.83, 10, u, g, q, h, 1.7b, t, f, 0, f, g, 00029, 837, +
b, 19.33, 9.5, u, g, q, v, 1, t, f, 0, t, g, 00060, 400, +
a, 32.33, 0.54, u, g, cc, v, 0.04, t, f, 0, f, g, 00440, 11177, +
b, 36.67, 3.25, u, g, q, h, 9, t, f, 0, t, g, 00102, 639, +
b, 37.50, 1.125, y, p, d, v, 1.5, f, f, 0, t, g, 00431, 0, +
a, 25.08, 2.54, y, p, aa, v, 0.25, t, f, 0, t, g, 00370, 0, +
b, 41.33, 0, u, g, c, bb, 15, t, f, 0, f, g, 00000, 0, +
b, 56.00, 12.5, u, g, k, h, 8, t, f, 0, t, g, 00024, 2028, +
a, 49.83, 13.585, u, g, k, h, 8.5, t, f, 0, t, g, 00000, 0, +
```


分类问题综合案例(一): 信贷审批



分类问题综合案例(二): 鸢尾花数据分类

- 基于鸢尾花数据进行分类模型构建，使用logistics算法和KNN算法进行构建，并计算两种算法的AOC值，以及画出对应的ROC曲线

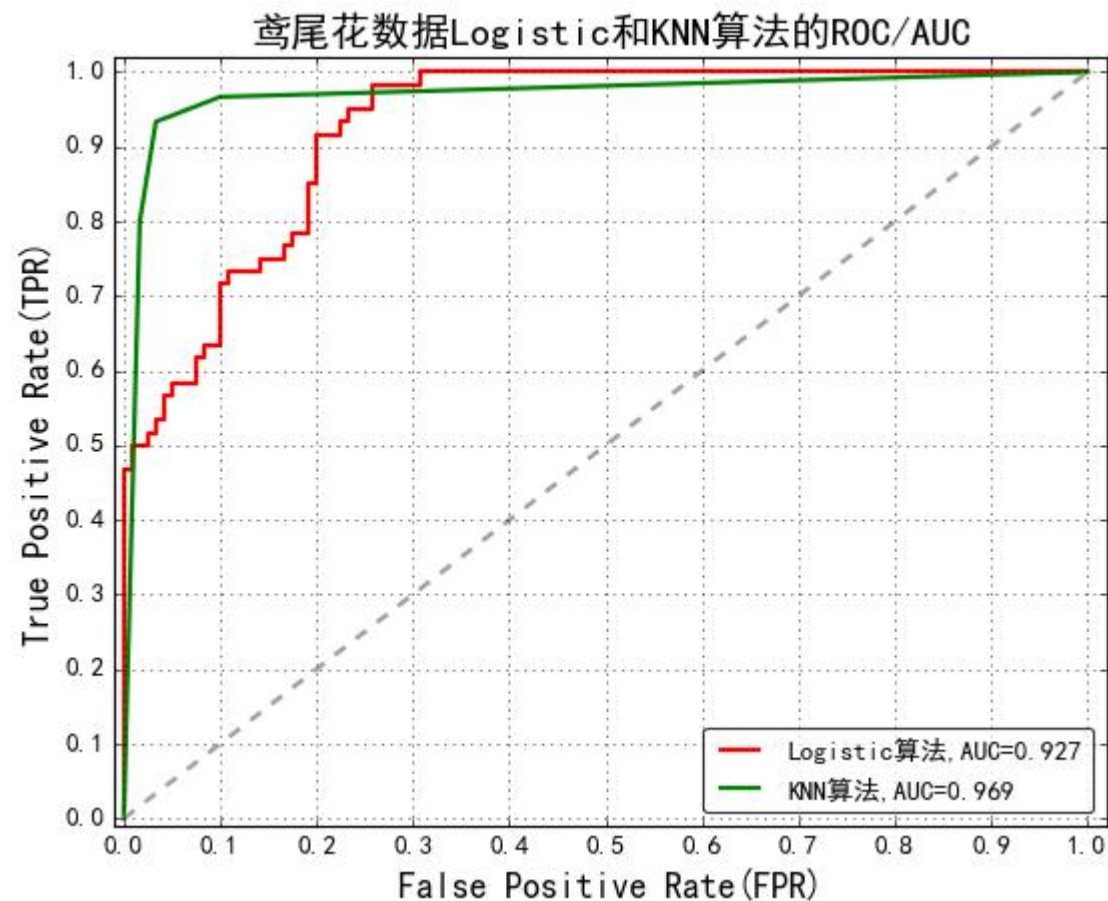
◆ 数据来源: <http://archive.ics.uci.edu/ml/datasets/Iris>

Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	1319181

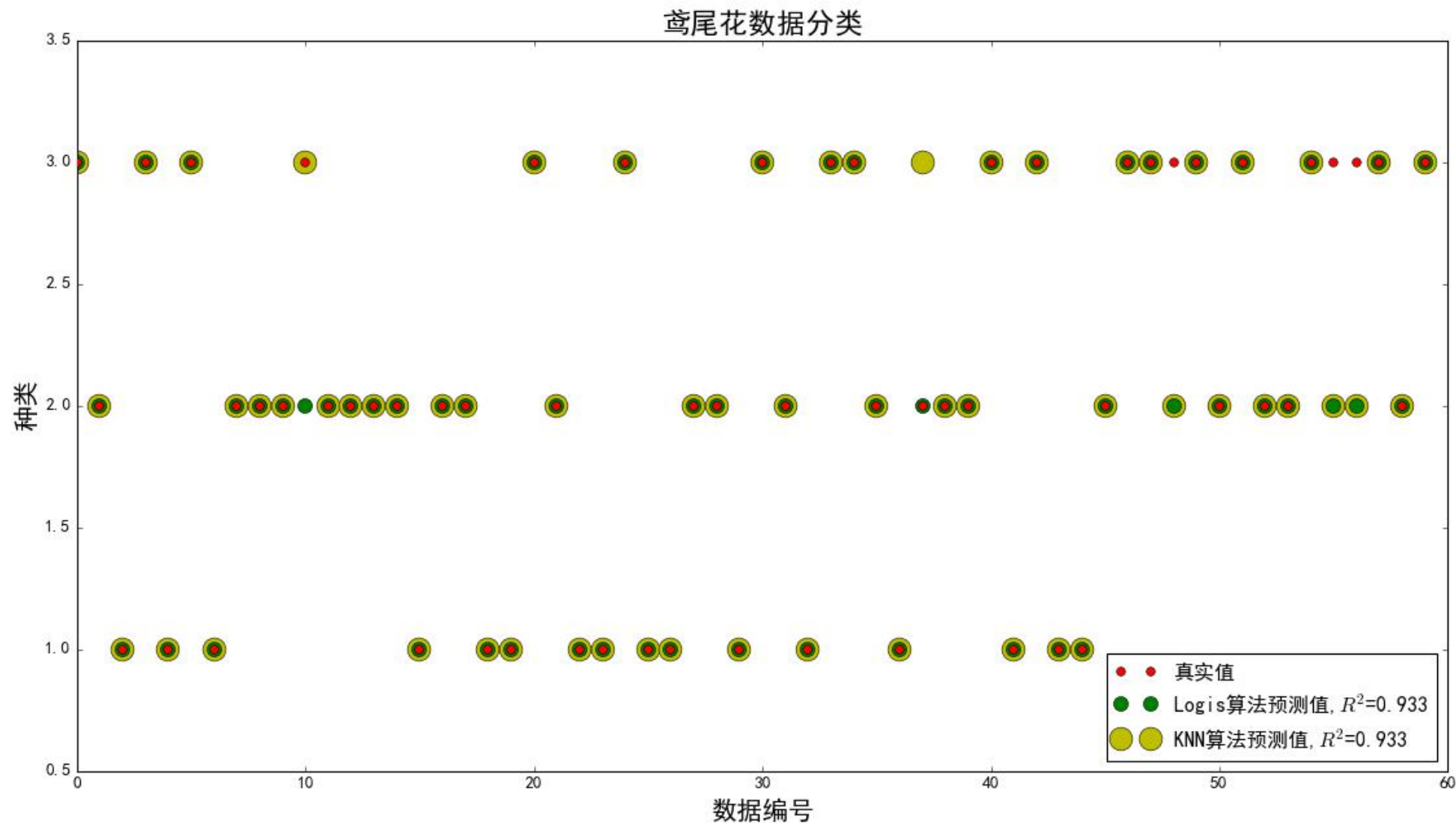
Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
 - Iris Setosa
 - Iris Versicolour
 - Iris Virginica

分类问题综合案例(二): 鸢尾花数据分类



分类问题综合案例(二): 鸢尾花数据分类



总结

- 线性模型一般用于回归问题，Logistic和Softmax模型一般用于分类问题
- 求 θ 的主要方式是梯度下降算法，梯度下降算法是参数优化的重要手段，主要是SGD，适用于在线学习以及跳出局部极小值
- Logistic/Softmax回归是实践中解决分类问题的最重要的方法
- 广义线性模型对样本要求不必要服从正态分布、只需要服从指数分布簇(二项分布、泊松分布、伯努利分布、指数分布等)即可；广义线性模型的自变量可以是连续的也可以是离散的。



THANK YOU

上海育创网络科技有限公司