



线性回归模型及其求解方法

(Linear Regression Model and Its Solution)

刘远超

哈尔滨工业大学

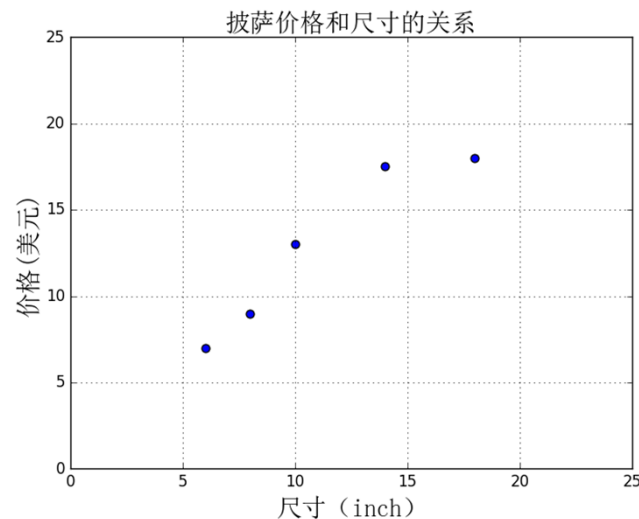
计算机科学与技术学院

什么是回归？

- 给定一组数据

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ \vdots & & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix}$$

研究 X 和 Y 之间关系的统计分析方法称之为**回归**。 X 是自变量， Y 是因变量。



- 利用训练数据，使用回归模型（如线性模型）去拟合变量之间的关系。因此训练任务就是利用数据，来学习模型中的参数 **parameter**（如线性模型中的斜率和截距）。
- 在本例中，训练完毕后，输入一个披萨的尺寸，模型就可以预测出其价格应该为多少。

回归和分类的区别和联系

- 区别:

- 分类: 使用训练集推断输入 x 所对应的离散类别 (如: $+1$, -1)。
- 回归: 使用训练集推断输入 x 所对应的输出值, 为连续实数。

- 联系:

- 利用回归模型进行分类: 可将回归模型的输出离散化以进行分类, 即 $y = \text{sign}(f(x))$ 。
- 利用分类模型进行回归: 也可利用分类模型的特点, 输出其连续化的数值。

线性模型

- 狭义线性（**linear**）模型：

- 通常指自变量与因变量之间按比例、成直线的关系，在数学上可理解为一阶导数为常数的函数，如 $y = \theta^T x$ ；
- 线性通常表现为一次曲线。

- 广义线性（**generalized linear model, GLM**）：

- 是线性模型的扩展，主要通过联结函数 $g()$ (link function)，使预测值落在响应变量的变幅内。例如逻辑回归

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad (\text{括号内为线性函数})$$

非线性模型

- 非线性non-linear模型:

- 非线性一般指不按比例、不成直线的关系，一阶导数不为常数

- 常见的非线性模型

- 2次以上的多项式 $y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$

- 幂函数模型 $y = \beta_0 x_1^{\beta_1} x_2^{\beta_2} \dots x_k^{\beta_k}$

- 指数函数模型 $y = \beta_0 e^{\beta_1 x}$

- 对数函数模型: $y = \beta_0 + \beta_1 \ln x$

- 等等

线性回归

- 线性回归模型中，假设自变量和因变量满足如下形式：

$$y = h_{\theta}(x) = \theta^T x$$

- 问题：已知一些数据，如何求里面的未知参数，给出一个最优解。
- 因此通常将参数求解问题转化为求最小误差问题。一般采用模型预测结果与真实结果的差的平方和作为损失函数：

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

$$\text{即求 } \theta = \min_{\theta} (J(\theta))$$

概率解释

- 设预测结果 $\theta^T x^{(i)}$ 与真实结果 $y^{(i)}$ 之间误差为 $\epsilon^{(i)}$ ，即 $y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$
- 通常误差满足平均值为0的高斯分布，即正态分布。那么在一个样本 i 上 x 和 y 的概率密度公式为 $p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2})$
- 模型在全部样本上预测的最大似然估计为

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

$$l(\theta) = \ln L(\theta) = -m \ln(\sqrt{2\pi}\sigma) - \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

从而，需要 $\frac{1}{2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$ 最小

求解参数

- 接下来，就是求解使得 $\frac{1}{2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$ 最小的参数 θ 。
- 解法有：
 - 矩阵解法。scikit-learn中的LinearRegression类使用的是矩阵解法（有时也称为最小二乘法）。可以解出线性回归系数 θ 。
 - 梯度下降法。梯度下降（Gradient descent）是利用一阶的梯度信息找到函数局部最优解的一种方法。

参数的矩阵解法

例如，设 $Y_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_i + \varepsilon_i$ ，即为线性关系 $\Rightarrow \varepsilon_i = Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_i$

$$Q = \sum_{i=1}^m \varepsilon_i^2 = \sum_{i=1}^m (Y_i - \widehat{Y}_i)^2 = \sum_{i=1}^m (Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_i)^2$$

通过使 Q 最小，即可确定 $\widehat{\beta}_0$ ， $\widehat{\beta}_1$ 。

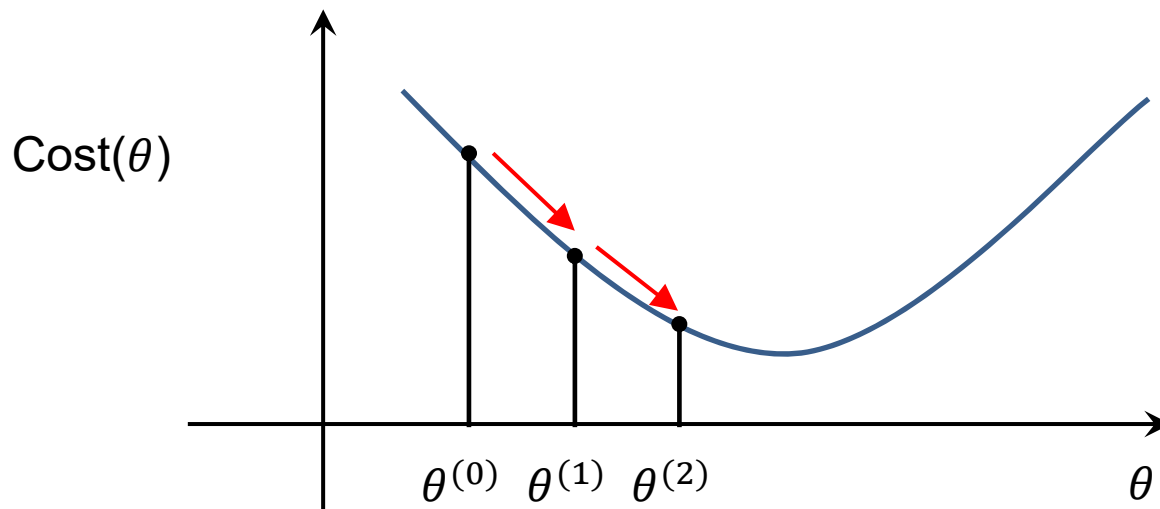
根据数学知识我们知道，函数的极值点为偏导为0的点，即

$$\begin{cases} \frac{\partial Q}{\partial \widehat{\beta}_0} = 2 \sum_{i=1}^m (Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_i)(-1) = 0 \\ \frac{\partial Q}{\partial \widehat{\beta}_1} = 2 \sum_{i=1}^m (Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_i)(-X_i) = 0 \end{cases}$$

$$\Rightarrow \beta_0 = \frac{n \sum X_i Y_i - \sum X_i \sum Y_i}{n \sum X_i^2 - (\sum X_i)^2}$$

$$\beta_1 = \frac{\sum X_i^2 \sum Y_i - \sum X_i \sum X_i Y_i}{n \sum X_i^2 - (\sum X_i)^2}$$

参数的梯度下降求解法



- 梯度下降（Gradient descent）是利用一阶的梯度信息找到函数局部最优解的一种方法，也是机器学习里面常用的一种优化方法。
- 其基本思想是，要找代价函数最小值，只需要每一步都往下走，也就是每一步都可以让误差损失函数小一点。
- 对于线性回归，参数的更新方法一般为：

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

- 如何求梯度？ →

线性回归的梯度下降

（接上文） $\theta'_j = \theta_j - L \frac{\partial J(\theta)}{\partial \theta_j} = \theta_j - \frac{1}{m} L \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$ ，是如何得出的？

- 对于某个实例 i ：

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

- 先初始化一组 θ ，在这个 θ 值之上，用梯度下降法去求出下一组 θ 的值。当迭代到一定程度， $J(\theta)$ 的值趋于稳定，此时的 θ 即为要求得的值。

Thanks!





多元回归与多项式回归

(Multivariate Regression and Polynomial Regression)

刘远超

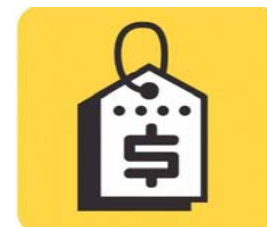
哈尔滨工业大学

计算机科学与技术学院

Sklearn的一元线性回归

- 在scikit-learn中，所有的估计器都带有`fit()`和`predict()`方法。
 - `fit()`用来拟合模型，`predict()`利用拟合出来的模型对样本进行预测。
- 用scikit-learn来构建一元线性回归的例子：

```
from sklearn.linear_model import LinearRegression
# 创建并拟合模型
model = LinearRegression()
X = [[6], [8], [10], [14], [18]]
y = [[7], [9], [13], [17.5], [18]]
model.fit(X, y)
print('预测12英寸匹萨价格：$%.2f' % model.predict([[12]]))
```



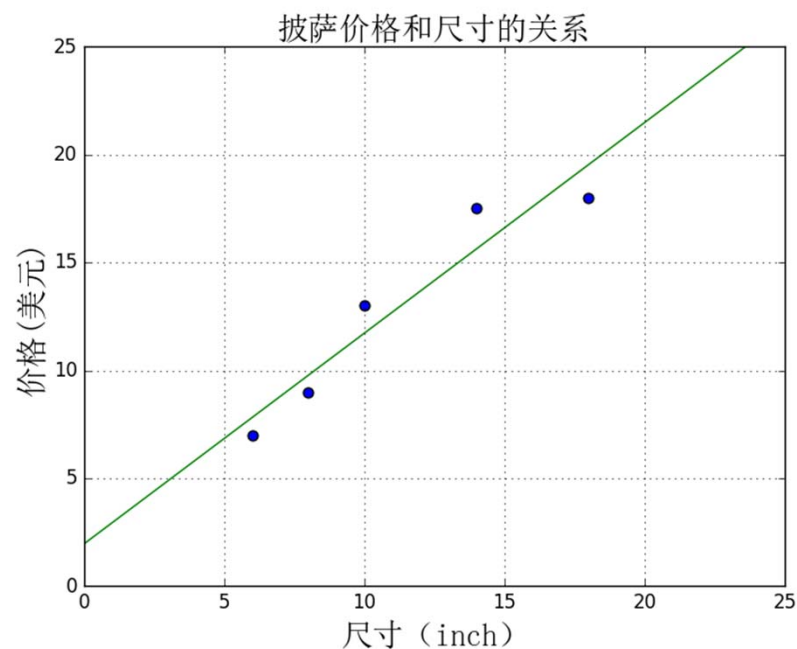
本例中根据模型，预测12英寸披萨的价格为\$13.68

线性回归的参数

- 对于刚才的例子，LinearRegression类的fit()方法学习线性回归模型：

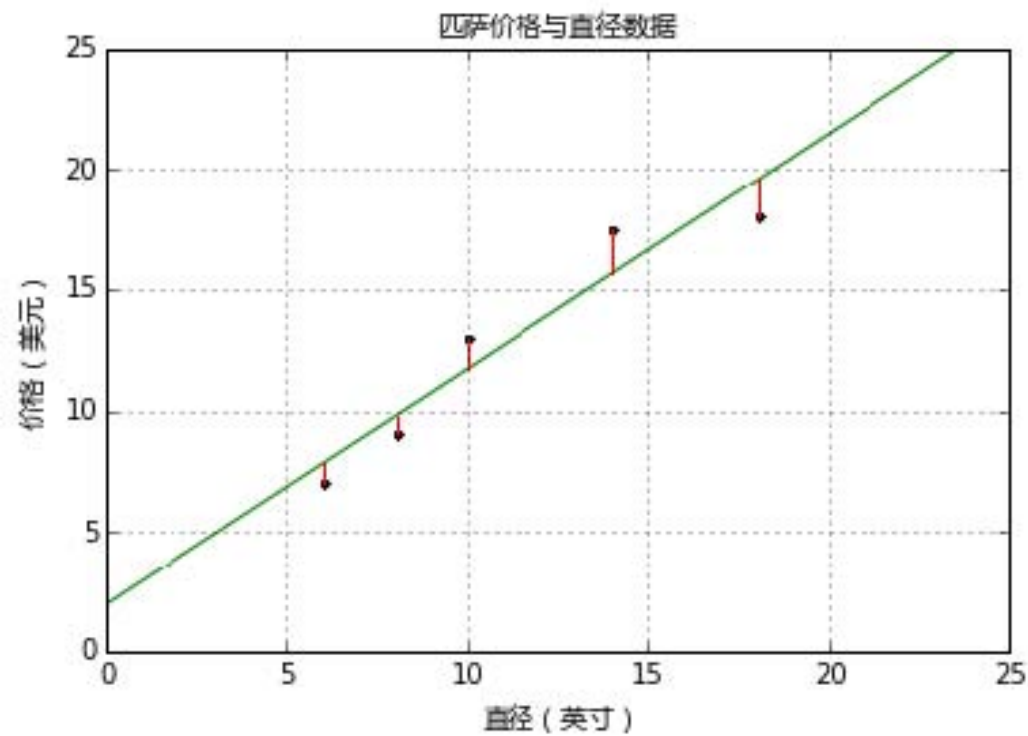
$$y = w_0 + w_1x$$

- 线性回归模型学习到的参数是截距和权重系数。下图中的直线就是匹萨直径与价格的线性关系。



残差 (residual)

- 残差：估计值 (拟合值) 与实际观察值之间的差。



多元线性回归

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ \vdots & & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix}$$

```
from sklearn.linear_model import LinearRegression
X = [[6, 2], [8, 1], [10, 0], [14, 2], [18, 0]]
y = [[7], [9], [13], [17.5], [18]]
model = LinearRegression()
model.fit(X, y)

X_test = [[8, 2], [9, 0], [11, 2], [16, 2], [12, 0]]
y_test = [[11], [8.5], [15], [18], [11]]
predictions = model.predict(X_test)
for i, prediction in enumerate(predictions):
    print('Predicted: %s, Target: %s' %
          (prediction, y_test[i]))
print('R-squared: %.2f' % model.score(X_test, y_test))
```

输出结果:

```
Predicted: [ 10.06250019], Target: [11]
Predicted: [ 10.28125019], Target: [8.5]
Predicted: [ 13.09375019], Target: [15]
Predicted: [ 18.14583353], Target: [18]
Predicted: [ 13.31250019], Target: [11]
R-squared: 0.77
```

R方计算步骤

R方可以用于评估回归模型对现实数据拟合的程度。

测试样本	真实价格 y_i	预测价格 $f(x_i)$
1	11	9.775
2	8.5	10.75
3	15	12.70
4	18	17.58
5	11	13.68

设 y_i 是测试集第 i 个样本的价格， \bar{y} 是真实价格的均值， $f(x_i)$ 是模型对第 i 个样本的预测价格， n 是样本数量。则R方计算步骤为：

- 1). 计算残差平方和： $SS_{res} = \sum_{i=1}^n (y_i - f(x_i))^2$
 $= (11 - 9.775)^2 + (8.5 - 10.75)^2 + \dots + (11 - 13.68)^2 = 19.19$
- 2). 计算样本总离差平方和： $SS_{tss} = \sum_{i=1}^n (y_i - \bar{y})^2$
 $= (11 - 12.7)^2 + (8.5 - 12.7)^2 + \dots + (11 - 12.7)^2 = 56.8$
- 3). 最后得到R方： $R^2 = 1 - \frac{SS_{res}}{SS_{tss}} = 1 - \frac{19.19}{56.8} = 0.66$

R方是0.66说明测试集中过半数的价格都可以通过模型解释。

多项式回归

- 用适当幂次的多项式来近似反映因变量与自变量之间的关系。

- 什么是多项式模型？

- 它是由常数与自变量 x 经过有限次乘法与加法运算得到。例如 $p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ 就是多项式函数。

- 多项式回归的步骤：

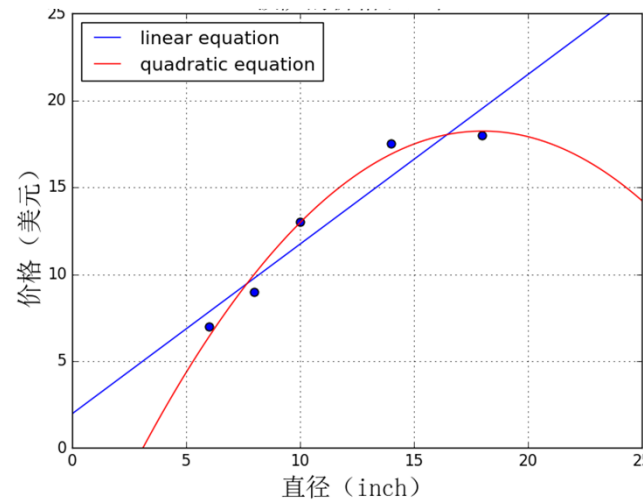
1. **生成多项式特征。**例如输入样本是 2 维的如 $[a, b]$ ，则二阶多项式的特征集为 $[1, a, b, a^2, ab, b^2]$ 。在 sklearn 中可以使用 ***PolynomialFeatures*** 生成多项式特征。
2. 利用得到的多项式特征，使用线性分类器处理。

二次回归 (Quadratic Regression)

- 最高次为2次。
- 仍以披萨的直径和价格之间的关系为例。
则此时二次回归的表达式为 $y = \beta_0 + \beta_1 x + \beta_2 x^2$ 。

#训练集数据

```
X_train = [[6], [8], [10], [14], [18]]  
y_train = [[7], [9], [13], [17.5], [18]]
```



#测试集数据

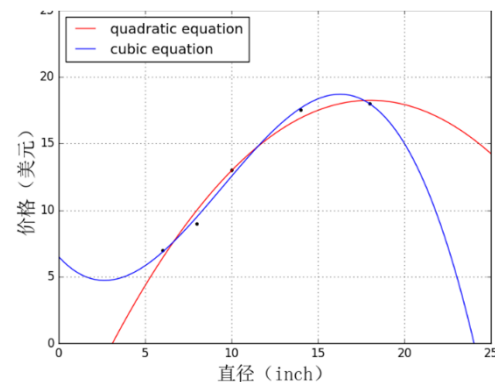
```
X_test = [[6], [8], [11], [16]]  
y_test = [[8], [12], [15], [18]]
```



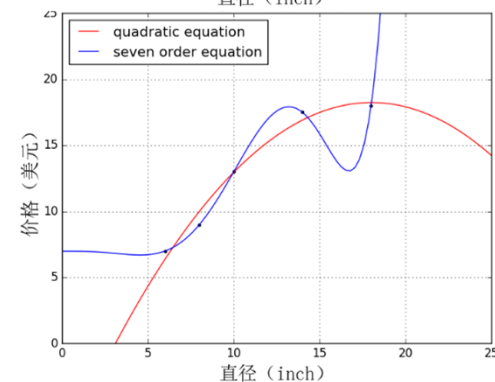
线性回归 (R方=0.81)
二次回归 (R方=0.87)

更高次的多项式回归

- 三次回归：拟合效果如右图：
 - 二次回归 $r\text{-squared}$ 0.87
 - 三次回归 $r\text{-squared}$ **0.84**



- 七次回归：拟合效果如右图：
 - 二次回归 $r\text{-squared}$ 0.87
 - 七次回归 $r\text{-squared}$ **0.49**



结论：高次多项式容易发生**过拟合 (overfitting)**，即模型在训练集上效果较好，但在测试集上效果较差。

Thanks!





损失函数的正则化

(Regularization of loss function)

刘远超

哈尔滨工业大学

计算机科学与技术学院

向量范数

向量范数，假设向量有 N 个元素。 $[\dots, \dots, \dots]$

- L1-范数：即向量元素绝对值之和。

$$\|x\|_1 = \sum_{i=1}^N |x_i|$$

- L2-范数：Euclid范数（欧几里得范数，常用计算向量长度），即向量元素绝对值的平方和再开方。

$$\|x\|_2 = \left(\sum_{i=1}^N |x_i|^2 \right)^{\frac{1}{2}}$$

- ∞ -范数：即所有向量元素绝对值中的最大值。

$$\|x\|_{\infty} = \max_i |x_i|$$

- $-\infty$ -范数：即所有向量元素绝对值中的最小值。

$$\|x\|_{-\infty} = \min_i |x_i|$$

- p-范数：即向量元素绝对值的p次方和的1/p次幂。

$$\|x\|_p = \left(\sum_{i=1}^N |x_i|^p \right)^{\frac{1}{p}}$$

矩阵范数

假设矩阵A为m*n, 即m行, n列。

$$\begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix}$$

- L1-范数: 列和范数, 即矩阵的所有列向量元素绝对值之和的最大值。

$$\|A\|_1 = \max_j \sum_{i=1}^m |a_{ij}|$$

- L2-范数: 谱范数, 即 $A^T A$ 矩阵的最大特征值的开平方。

$$\|A\|_2 = \sqrt{\lambda_1}, \quad \lambda_1 \text{ 为 } A^T A \text{ 的最大特征值}$$

- ∞ -范数: 行和范数, 即矩阵的所有行向量元素绝对值之和的最大值。

$$\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$$

- F-范数: Frobenius范数, 即矩阵元素绝对值的平方和再开平方。

$$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}}$$

线性回归的正则化

- 应对过拟合(Overfitting)。因为在某些情况下，学习得到的模型在训练集上也许误差较小。但是对于测试集中之前未见样本的预测却未必有效。为此可以在损失函数中加入正则化项。以线性回归为例，

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

正则化项

其中 α 是正则化参数（regularization parameter），用于控制两个不同的目标的平衡。

- I. 第一个目标是使假设更好地拟合训练数据。
- II. 第二个目标是要正则化处理，使得模型不要太复杂。

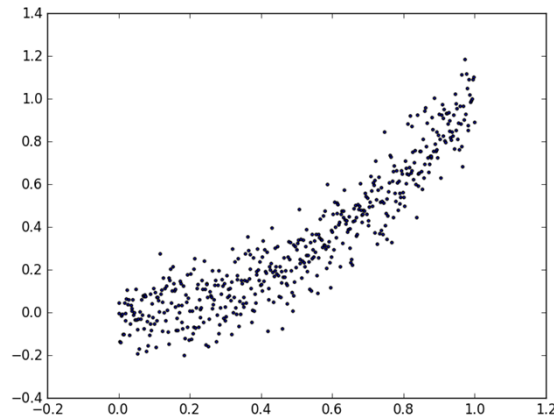
线性回归正则化后的梯度更新方法

- 新的损失函数 $J(\theta) = \frac{1}{2m} [\sum_{i=1}^m (h_{\theta}(x^i) - y^i)^2 + \alpha \sum_{k=1}^n \theta_k^2]$
- 新的梯度更新公式:

$$\begin{aligned}\theta_j' &= \theta_j - L \frac{\partial J(\theta)}{\partial \theta_j} \\ &= \theta_j - L \cdot \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i) x_j^i - \frac{\alpha}{2m} \cdot 2 \cdot \theta_j \right] \\ &= \theta_j - L \cdot \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i) x_j^i - \frac{\alpha}{2m} \cdot 2 \cdot \theta_j \right] \\ &= \theta_j (1 - L \frac{\alpha}{m}) - L \cdot \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i) x_j^i\end{aligned}$$

举例：为什么回归要正则化？(1)

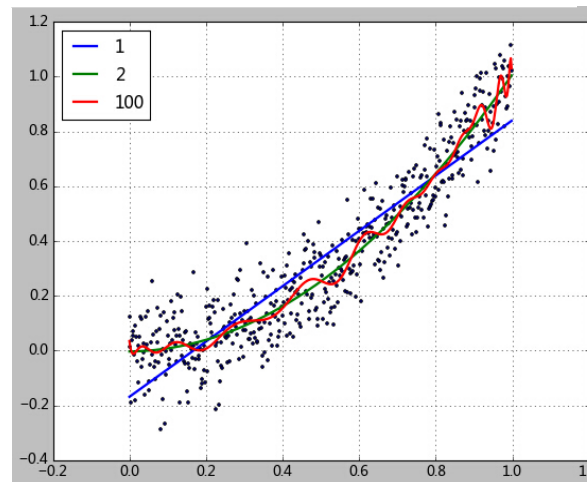
- 本例中使用一个2次函数加上随机的扰动来生成500个点。（其中300个作为训练集，后200个作为测试集）



- 然后尝试用1、2、100次方的多项式对该数据进行拟合。

拟合的性能如下：

- 一次方： **$R^2=0.82$**
- 二次方： **$R^2=0.88$**
- 100次方： **$R^2=0.89$**



举例：为什么回归要正则化？ (2)

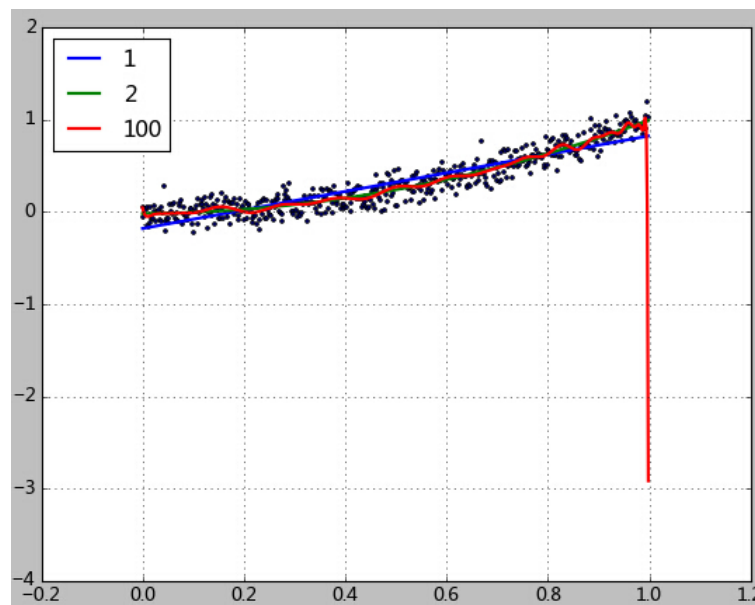
高次多项式的过拟合现象的观察：

●而在测试集上的评价结果为：

一次： **$R^2=0.85$**

二次： **$R^2=0.90$**

100次： **$R^2=0.57$**



●结果表明，高次多项式过度拟合了训练数据。

举例：为什么回归要正则化？(3)

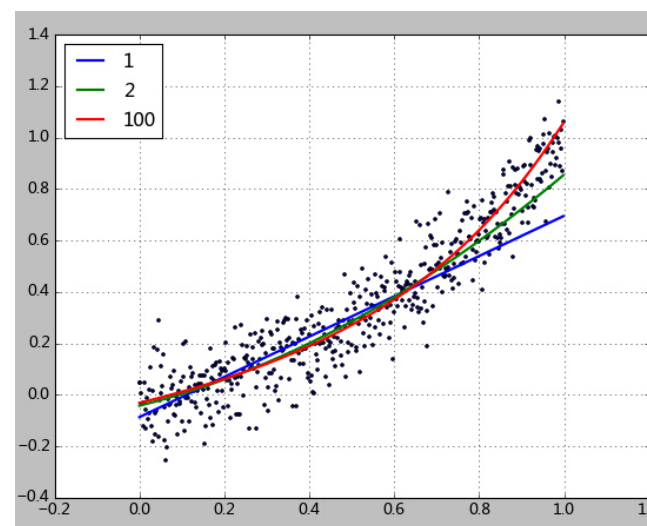
- 通过查看，会发现100次多项式拟合出的系数数值通常很大。因此在拟合过程中可以限制这些系数数值的大小来避免生成畸形的拟合函数。
- 具体做法就是在损失函数中加入正则化项，以抑制这些系数的数值。
 - 如：Lasso回归（使用L1正则化）、岭（Ridge）回归（使用L2正则化）、弹性网（Elastic net，使用L1+L2正则化）等回归。

- 下面以岭回归为例看看100次多项式的拟合是否有效。

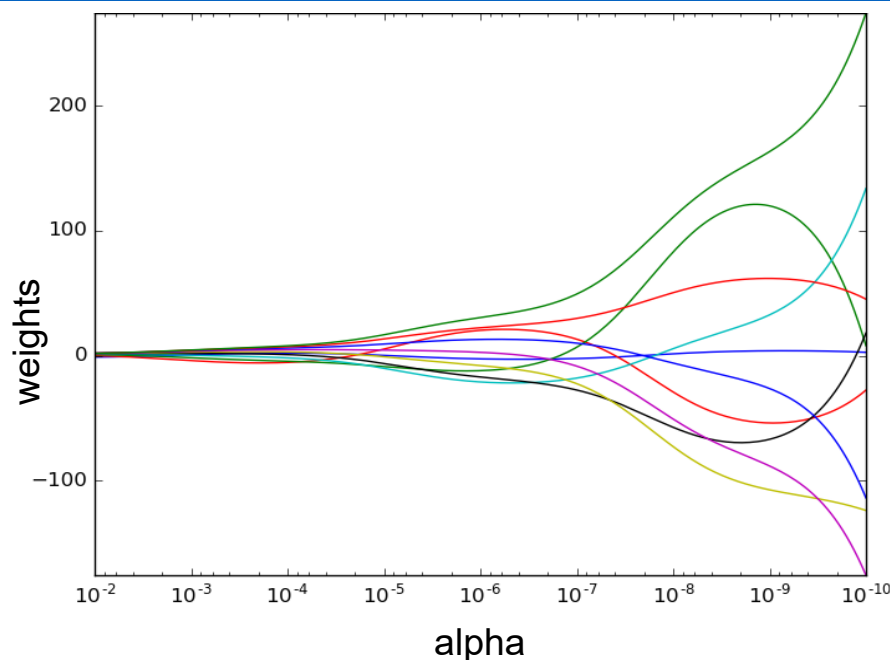
一次： **R2=0.78**

二次： **R2=0.87**

100次： R2=0.90



岭回归中正则权重的作用



岭回归中权重系数weights（纵轴）的取值与正则强度alpha（横轴）的关系

- alpha值越大（即越偏左），weights（即回归权重参数）变得较小，即对权重的约束越大。
- alpha值越小（即越偏右），权重系数weights的取值表现出较大的震荡。特别是当接近最右侧（ $10^{-10} \approx 0$ ）时，相当于没有对权重参数进行约束。

通过交叉验证找到最佳超参数alpha

- 可以通过交叉验证的方式设置调整超参数alpha。
- **Sklearn提供了RidgeCV**，可以允许输入一系列的alphas，然后sklearn通过交叉验证得到适合的alpha。

举例程序如下：

➤ `from sklearn import linear_model`

➤ `clf = linear_model.RidgeCV(alphas=[0.1, 1.0, 10.0])`

➤ `clf.fit([[0, 0], [0, 0], [1, 1]], [0, .1, 1])`

`RidgeCV(alphas=[0.1, 1.0, 10.0], cv=None, fit_intercept=True,
scoring=None, normalize=False)`

➤ `clf.alpha_`

0.1

Thanks!





逻辑回归

(Logistic Regression)

刘远超

哈尔滨工业大学

计算机科学与技术学院

逻辑回归 (Logistic Regression)

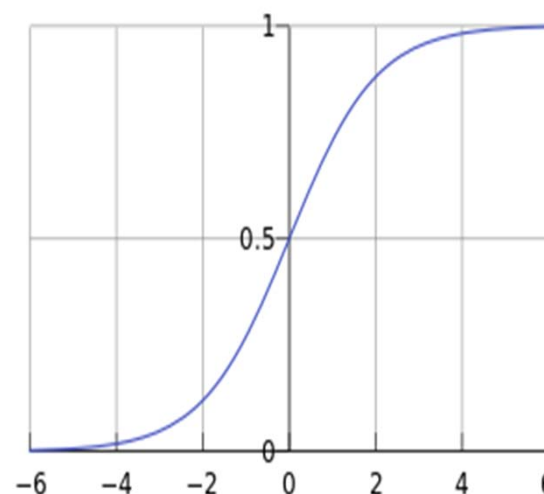
- 线性回归的输出值的范围通常是无法限定的。
- 逻辑回归通过使用 $logistic$ 函数（或称为 $sigmoid$ 函数）将其转化为 $(0, 1)$ 区间的数值。

- $logistic$ 函数形式为：

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

其中 $z = h_{\theta}(x) = \theta^T x$

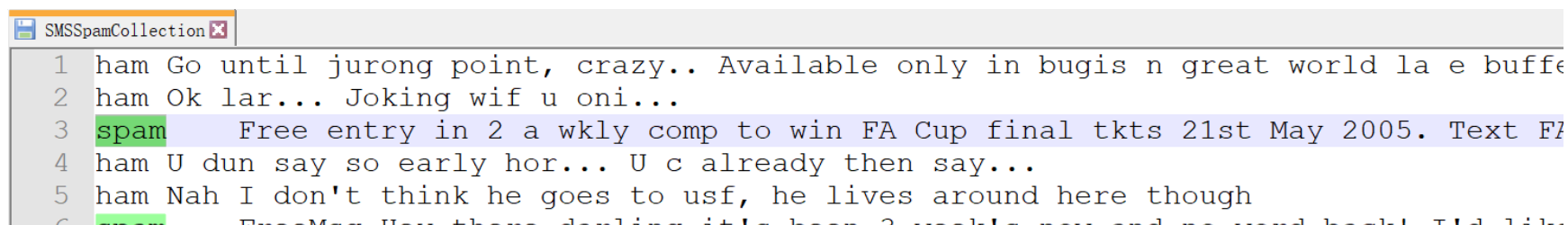
- z 变量的范围为 $-\infty$ 到 $+\infty$,
- 值域限制在 $0 - 1$ 之间。



- 逻辑回归可以被理解为一个被 $logistic$ 函数归一化后的线性回归。也可以被视为一种广义线性模型。

逻辑回归应用举例：垃圾短信分类

- 垃圾短信的数据集：UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>)。总共5574条，其中4827条是ham，747条是spam。



The screenshot shows a text file named 'SMSSpamCollection'. It contains a list of SMS messages, each preceded by a line number and a label ('ham' or 'spam'). The messages are as follows:

Line	Label	Message
1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffe
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text F2
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives around here though
6	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like

- 可以用TF-IDF来抽取短信的特征向量，然后用逻辑回归分类：
 - 如果逻辑回归响应变量等于或超过指定的临界值（如0.5），预测结果就是正例，否则为反例。

逻辑回归中的损失函数优化方法

即如何找到能使损失函数取值尽可能小的模型参数。

- `class sklearn.linear_model.LogisticRegression (solver='liblinear')`。其中的solver参数决定了逻辑回归损失函数的优化方法。

- Solver的常见取值包括：

- **sag**：即随机平均梯度下降，是**梯度下降法**的变种，和普通梯度下降法的区别是每次迭代仅仅用一部分的样本来计算梯度，适合于样本数据多的情况。

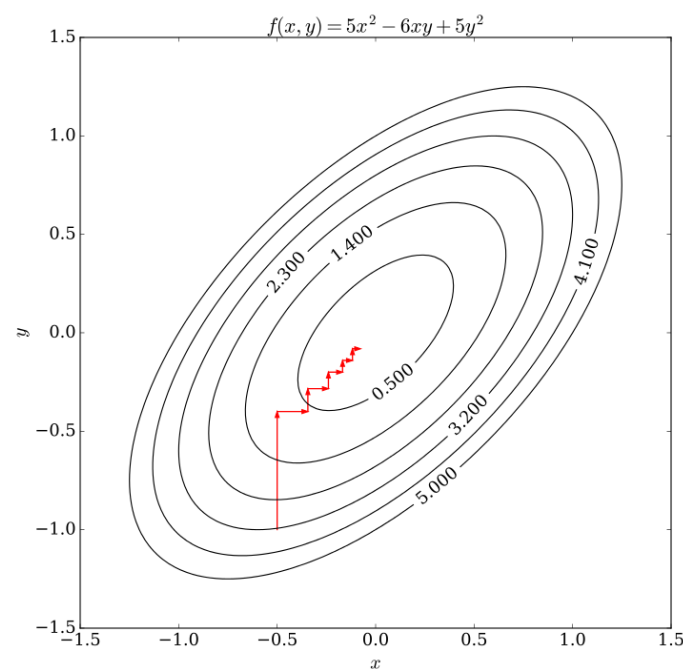
- **liblinear**：使用了开源的liblinear库实现，内部使用了**坐标轴下降法CD**来迭代优化损失函数。

- 其他取值，如**lbfgs**、**newton-cg**等

坐标下降法

- Sklearn中逻辑回归的优化解决方案“liblinear”使用基于Liblinear的**coordinate descent** CD（坐标下降）算法。
- CD是一种非梯度优化算法。在每次迭代中，在当前点处沿一个坐标方向进行一维搜索以求得一个函数的局部极小值。在整个过程中循环使用不同的坐标方向。
- 如果在某次迭代中，函数得不到优化，说明一个驻点已经达到。

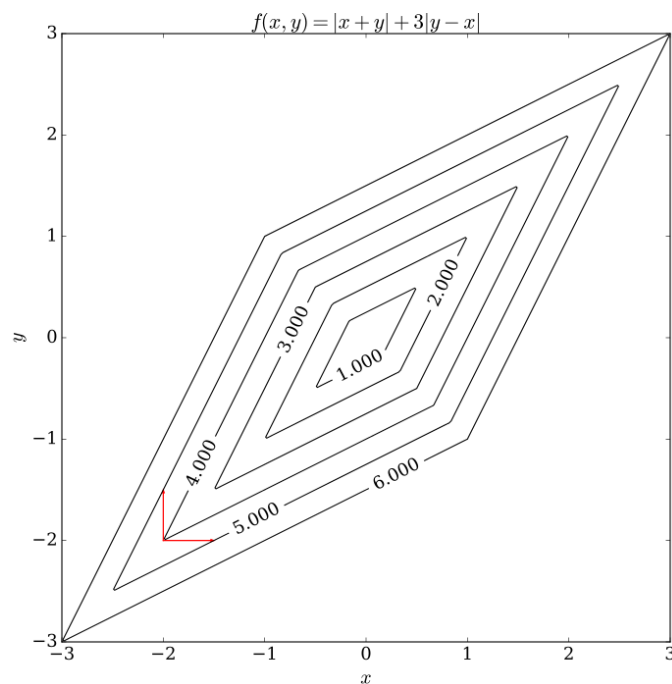
举例：如何找到能使 $f(x, y) = 5x^2 - 6xy + 5y^2$ 较小值的 x 和 y 取值对？



https://en.wikipedia.org/wiki/Coordinate_descent

坐标下降法(续)

- 对于非平滑函数，CD法可能会遇到问题。即函数等高线非平滑时，算法可能在非驻点中断执行。



https://en.wikipedia.org/wiki/Coordinate_descent

Thanks!

