监督学习
--回归

万永权

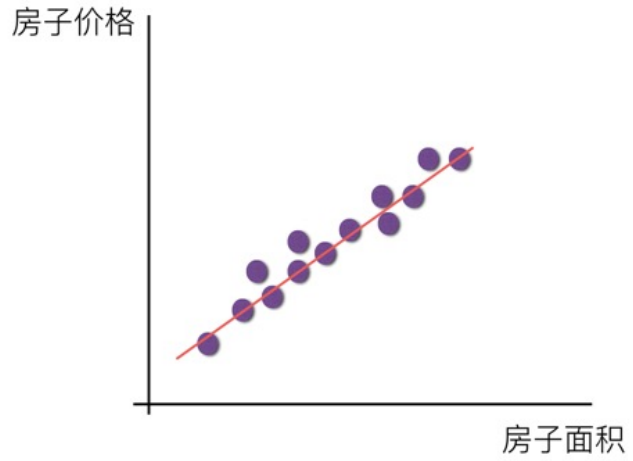
目录

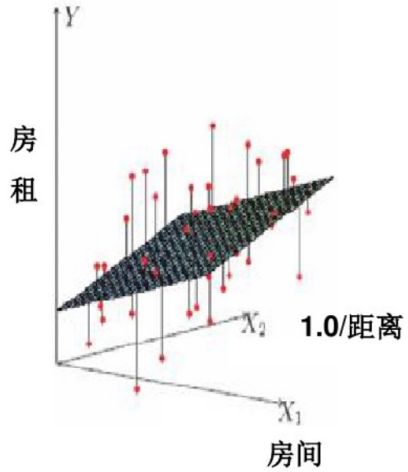
CONTENTS

回归

回归属于监督学习中的一类任务。
该任务的核心思想是从连续型的统计数据中得到数学模型，然后将该数学模型用于预测或者分类。

回归方法处理的数据可以是一维的，也可以是多维的。





回归（Regression）

- 英国人类学家高尔登（Francis Galton）首次在《自然遗传》一书中，提出并阐明了“相关”和“相关系数”两个概念，为相关论奠定了基础。
- 1870年，他和英国统计学家Karl Pearson对上千个家庭的身高、臂长、拃长做了测量，发现：
- 儿子身高（Y）与父亲身高（X）存在线性关系：
  - Y = 0.516X + 33.73
- Galton 将这种身高趋向于种族稳定的现象称为“回归”。

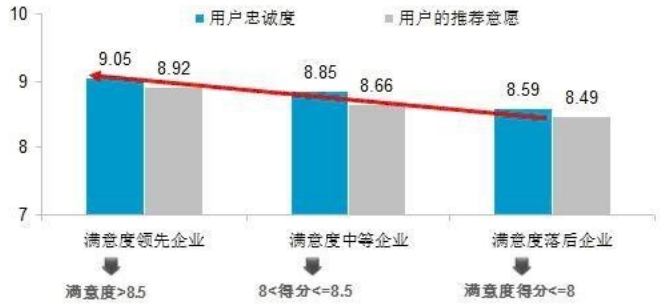
回归

- “回归”成为表示变量之间某种数量依存关系的统计学术语
- 机器学习中的“回归”，借鉴了统计学中的同名术语
- 机器学习中回归任务的实现，采用了机器学习中的一套规则、实现方法和评价标准，这些与统计学有着很大的不同

回归的应用

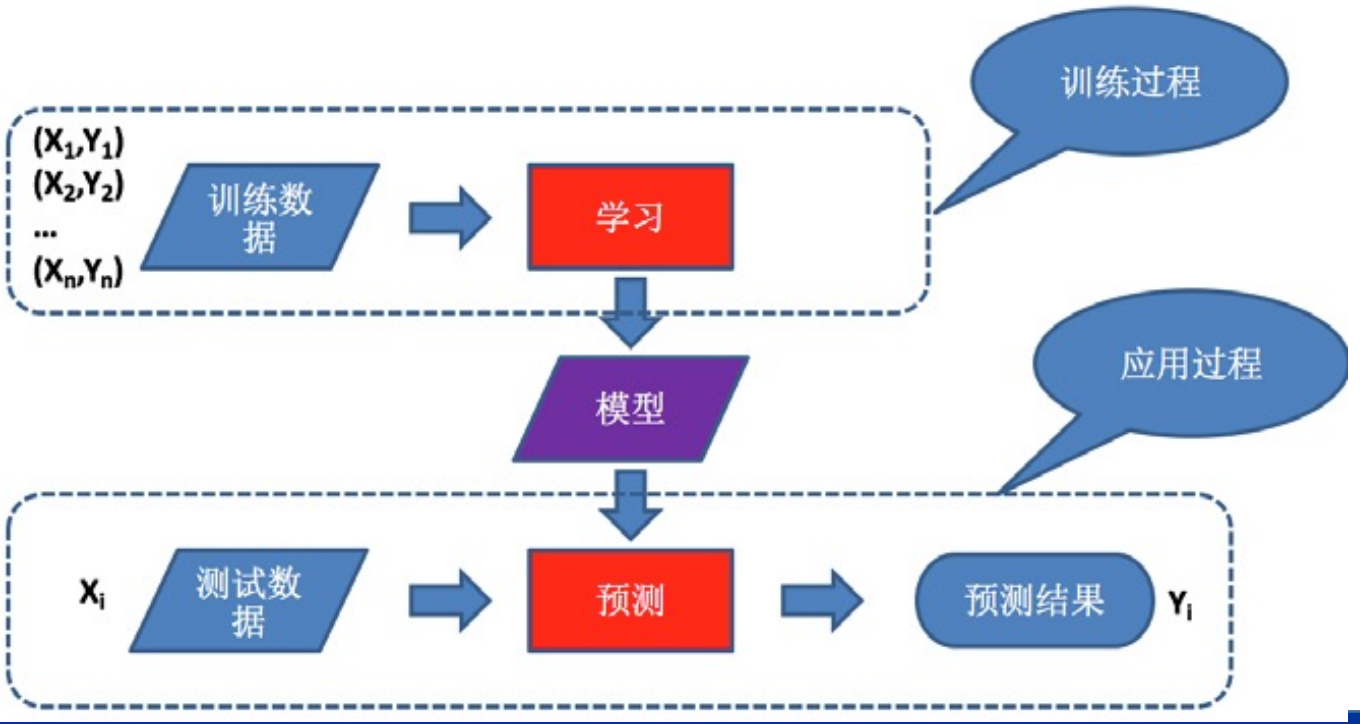
回归分析简单有效，应用十分广泛，包括：

- 广告点击率的预估、
- 大型网站的容量规划、
- 用户满意度的分析等等。



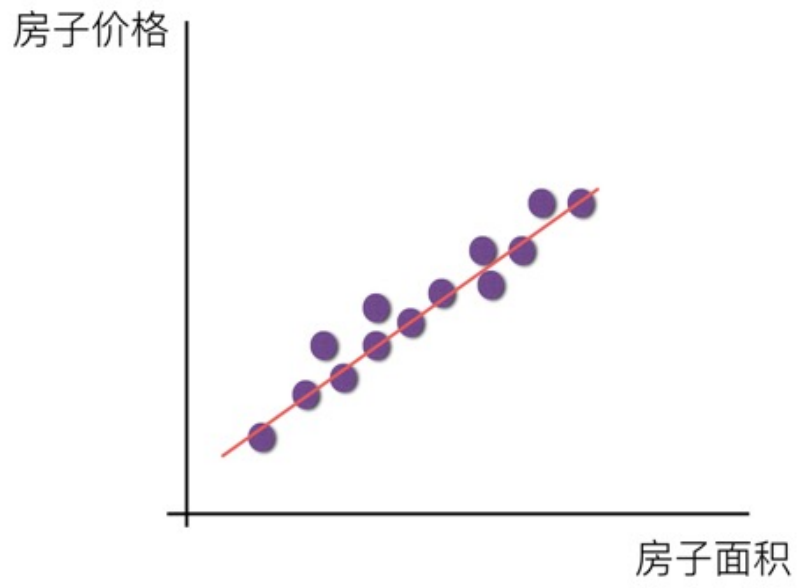
回归

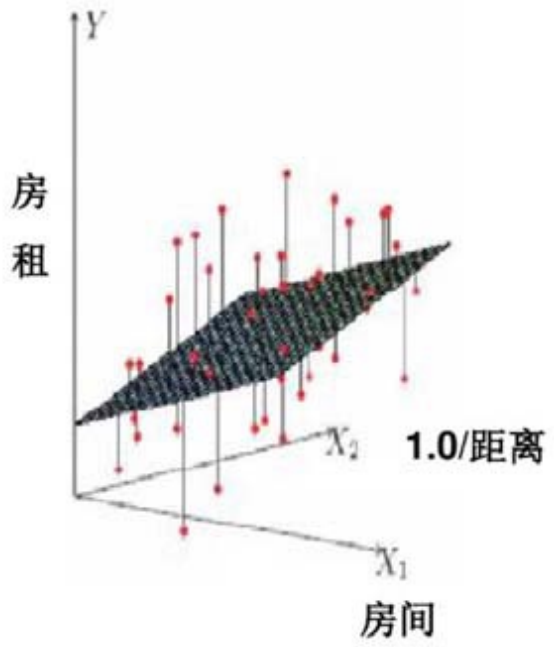
- 回归分析研究的主要是因变量（目标）和自变量（特征）之间的定量依存关系。
- 按数量依存关系的类型，可分为线性回归分析和非线性回归分析。
- 学习过程如下：



回归

- 根据自变量的个数，回归可以分为：
  - 一元回归分析
  - 多元回归分析



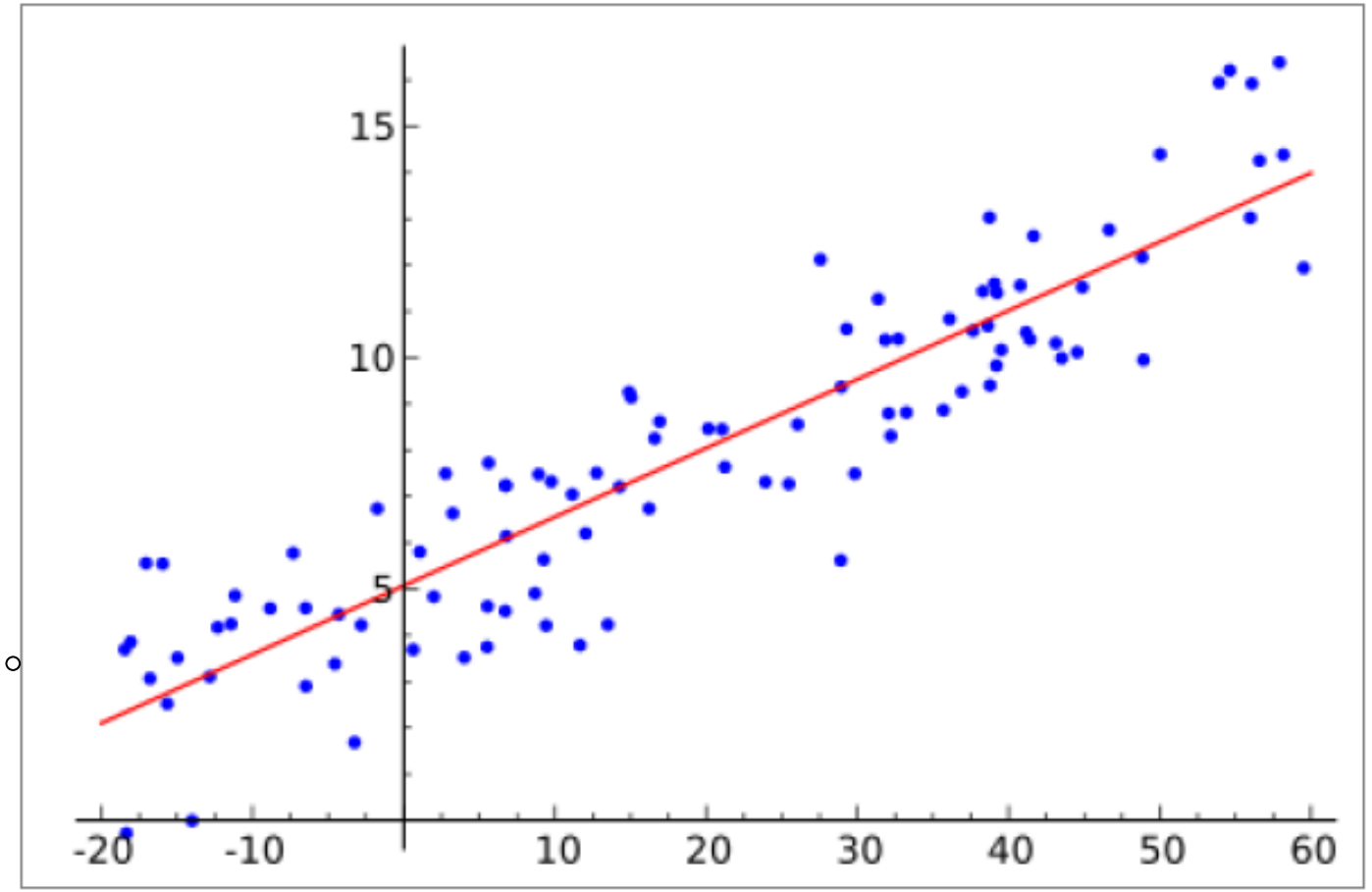


（1）线性回归

- 线性回归模型：$y = w^T x + e$，其中w^T是模型参数向量的转置，e表示误差。
- 研究一个因变量与一个或多个自变量间多项式的回归分析方法，称为**多项式回归**。
- 如果自变量只有一个时，称为**一元多项式回归**；
- 如果自变量有两个或两个以上时，称为**多元多项式回归**。
- 若一个因变量与一个或多个自变量间是非线性关系，例如，$y(x) = w_2 x^2 + w_1 x + e$，则称为**非线性回归**。
- 在一元多项式回归分析中，若一个自变量和一个因变量的关系可用一条直线近似表示，这种回归称为**一元线性回归**，即找一条直线来拟合数据。
- 如果在多元多项式回归分析中，一个因变量和多个自变量之间是线性关系，则称为**多元线性回归**。

线性回归

- 线性回归中，采用具有如下特征的函数对观测数据进行建模：
  - 该函数是模型参数的线性组合；
  - 该函数取决于一个（一元线性回归）或多个独立自变量（多元线性回归）。



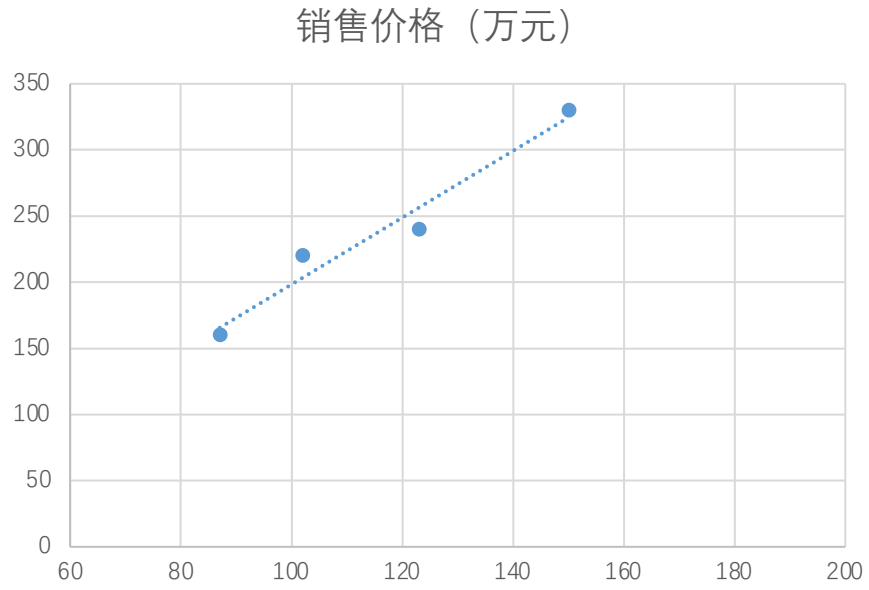
$$y(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b$$

模型表达：$y\,(x,w)=w_1x_1+\ldots+w_nx_n+b$

回归

假设有房屋销售的数据：

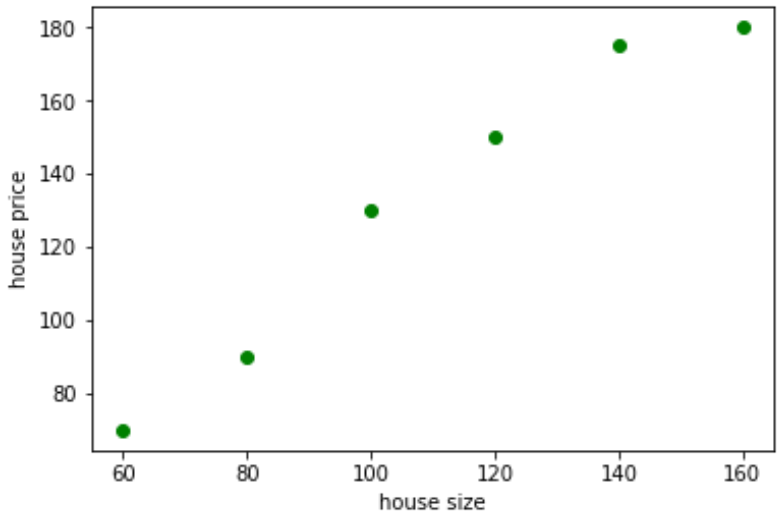
面积（平方米）	销售价格（万元）
123	240
150	330
87	160
102	220

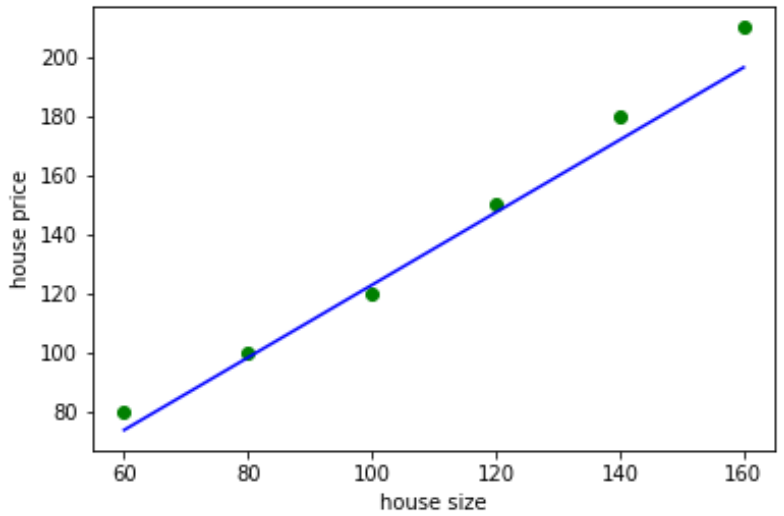


例:

房屋的价格与面积、卧室数目、地铁、电梯等有关系，这些称为特征。假设我们的房屋价格仅仅与面积有关系，那么我们如何找出房屋价格和面积之间的关系呢?

训练样本	房屋面积	价格
1	60	70
2	80	90
3	100	130
4	120	150
5	140	175
6	160	180





一元线性回归

- 寻找 X 和 Y 之间的线性关系
- 其中 X 只有一维数据

$$Y = \theta_0 + \theta_1 * X$$

- θ_0 和 θ_1 均称为 参数
- 一元线性回归的任务就是通过给定的训练数据，来获得最佳的参数值，这个过程称为“学习”，从数据中学习。
- 参数决定了回归直线相对于训练集的准确程度，即，模型预测值与训练集中实际值之间的差距，称为误差。
- 学习的目的是为了获得参数的最优值，从而完成模型的构建。

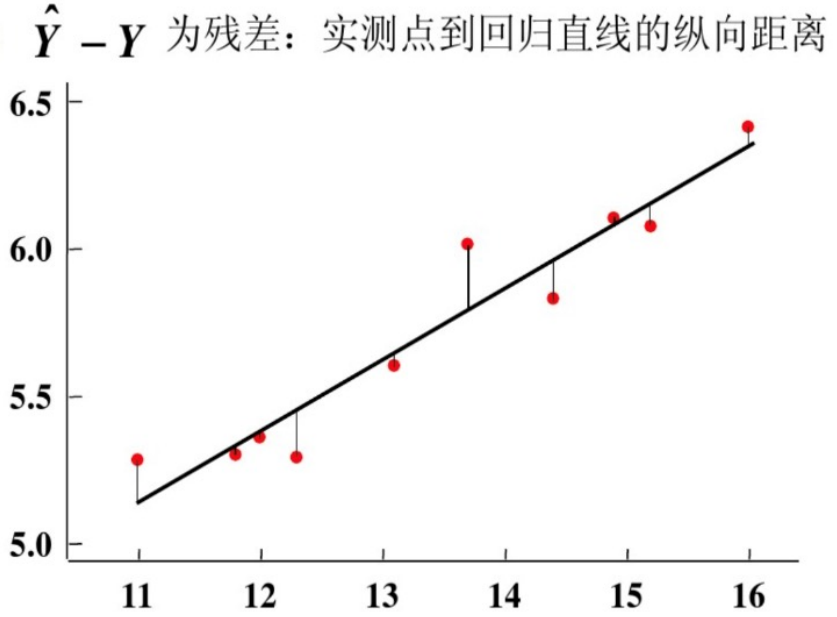
一元线性回归

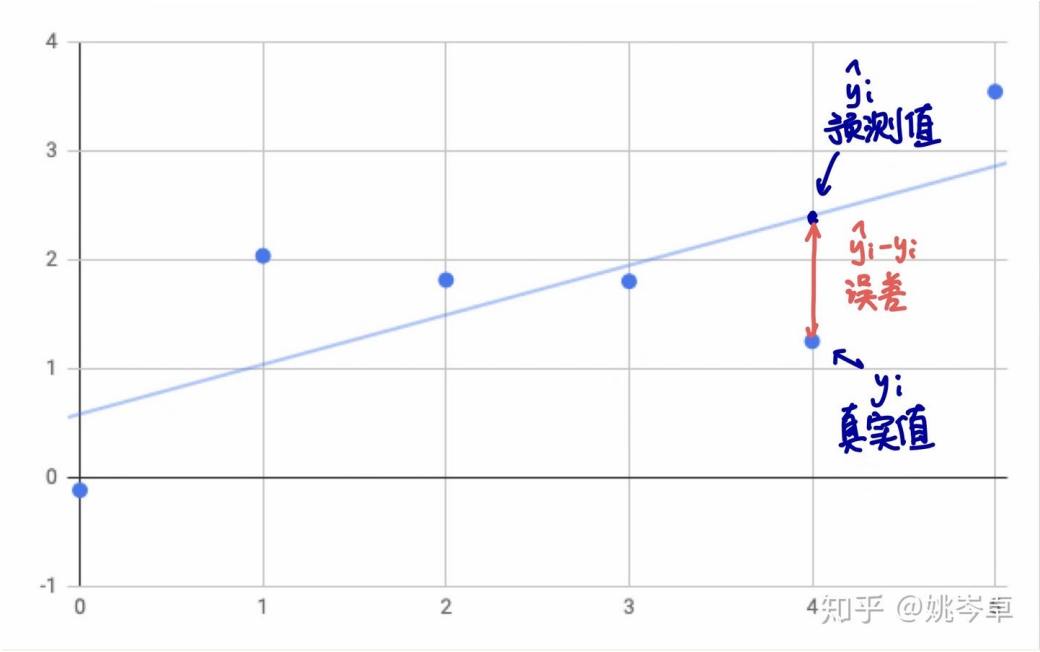
如何确定参数 θ_0 和 θ_1 呢?

常采用的策略是误差平方和最小化准则，即:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(\hat{Y}(\mathrm{i}) - Y(\mathrm{i})\right)^2$$

$$min_\theta \ J_\theta$$





直接计算法

一元线性回归方程

- 设一元线性回归方程为 $\hat{y} = \alpha x + \beta$
- 数据样本点为 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$
- 要使得这 n 个样本点落在在一元线性回归方程附近，不妨假设误差为 ε,使得每个样本点都落在一元线性回归方程上。
- 因此$\hat{y}_i = y_i + \varepsilon_i$ 恒成立。
- 回归直线应满足的条件是：全部观测值与对应的回归估计值的误差平方和最小，即：

$$\arg\min_{\alpha,\beta} \sum_{i=1}^{n} \varepsilon_i^2 = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \arg\min_{\alpha,\beta} \sum_{i=1}^{n} (y_i - \alpha x_i - \beta)^2$$

一元线性回归

- 令 $J(\alpha,\beta)=\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)^2$ ，原问题就转化为求关于α和β二元函数J的极小值的问题。
- 由微积分相关知识可知:

关于α的导数

$$\nabla_\alpha J(\alpha,\beta)=-2\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)x_i$$

$$=-2\sum_{i=1}^{n}x_iy_i+2\alpha\sum_{i=1}^{n}x_i^2+2\beta\sum_{i=1}^{n}x_i$$

关于β的导数

$$\nabla_\beta J(\alpha,\beta)=-2\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)$$

$$=-2\sum_{i=1}^{n}y_i+2\alpha\sum_{i=1}^{n}x_i+2n\beta$$

一元线性回归

◆然后令 $\nabla_\alpha J(\alpha,\beta)=0$ 和 $\nabla_\beta J(\alpha,\beta)=0$ 即可求出α、β的值。

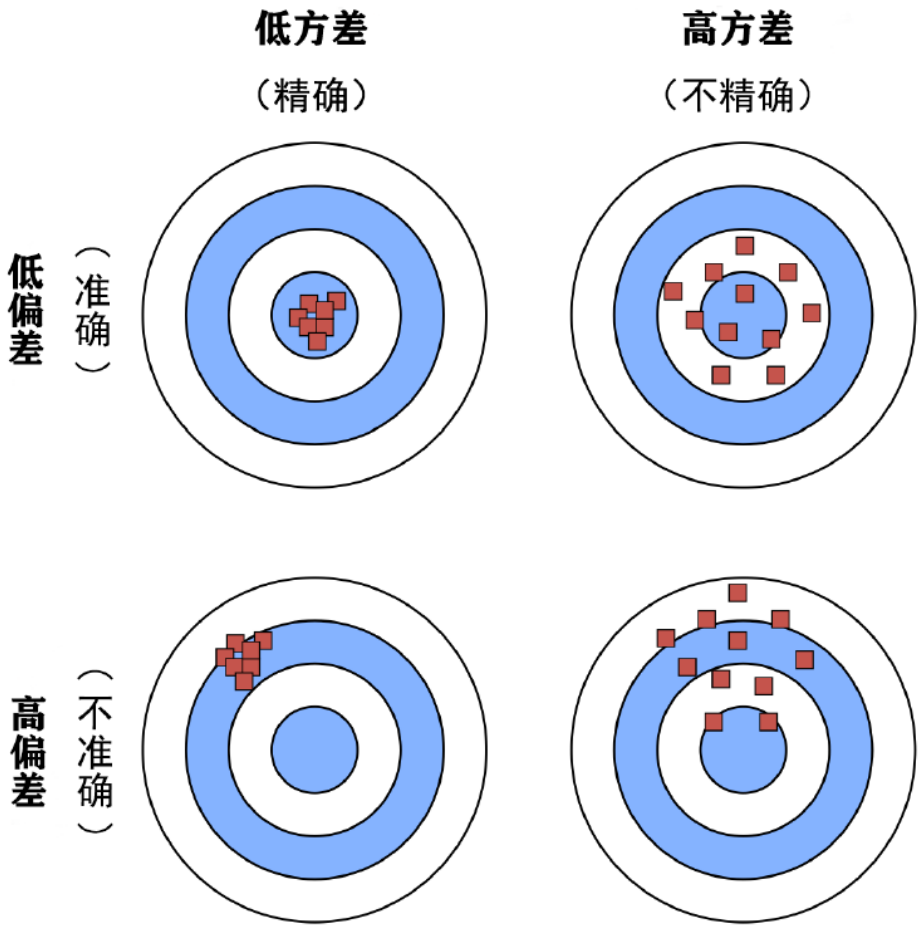
$$\nabla_\beta J(\alpha,\beta)=0$$
$$\Rightarrow \sum_{i=1}^{n} y_i = \alpha\sum_{i=1}^{n} x_i + n\beta$$
$$\Rightarrow \bar{y} = \alpha\bar{x} + \beta$$

$$\nabla_\alpha J(\alpha,\beta)=0$$
$$\Rightarrow \sum_{i=1}^{n} x_i y_i = \alpha\sum_{i=1}^{n} x_i^2 + \beta\sum_{i=1}^{n} x_i$$
$$\Rightarrow \alpha = \frac{\sum_{i=1}^{n}(x_i y_i - \bar{y}x_i)}{\sum_{i=1}^{n}(x_i^2 - \bar{x}x_i)} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} x_i)}$$
$$= \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

教材P155

偏差与方差

教材P163



◆偏差

- 描述模型预测值与实际值之间的偏离关系

◆方差

- 描述模型预测值的变化范围

• 偏差-方差权衡

- 过拟合：对噪声过于敏感，方差大 → 增加表示/有效容量
- 欠拟合：对噪声过于不敏感，偏差大 → 降低表示/有效容量

模型　　训练时间等

在固定数据集上无法同时优化方差和偏差

方差和协方差

- 协方差（Covariance）在概率论和统计学中用于衡量两个变量的总体误差。 而方差是协方差的一种特殊情况，即当两个变量是相同的情况。
- 可以通俗的理解为：两个变量在变化过程中是同方向变化？还是反方向变化？同向或反向程度如何？
  - 你变大，同时我也变大，说明两个变量是同向变化的，这时协方差就是正的。
  - 你变大，同时我变小，说明两个变量是反向变化的，这时协方差就是负的。
  - 从数值来看，协方差的数值越大，两个变量同向程度也就越大。反之亦然。

最小二乘法

- 最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。
- 利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。
- 基本思路是：令

$$f(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_m\varphi_m(x)$$

其中，$\varphi(x)$ 是事先选定的一组线性无关的函数，

α_k 是待定系数，　$(k = 1, 2, \cdots, m, m < n)$

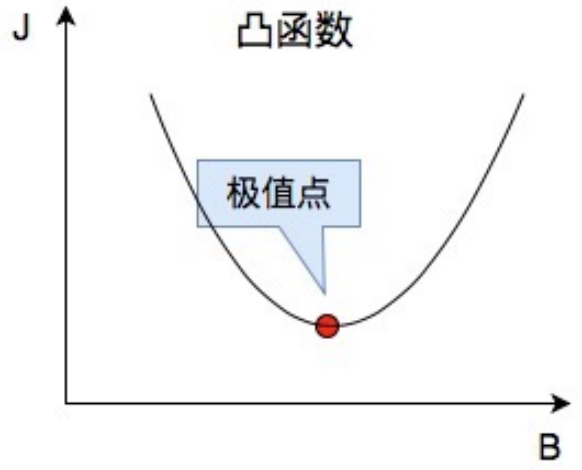
拟合准则是使 $y_i\ (i = 1,2,\cdots,n)$与 $f(x_i)$的距离 δ_i的平方和最小。

从样本数据出发推导损失函数

在样本数据中 $y^{(i)}$ 是实际存在值而 $h_\theta(x^{(i)})$ 对应的是模型预测值，显然如果想要模型预测的效果好，那么对应的误差就要小，假设函数在任意样本点的误差为 $|h_\theta(x^{(i)}) - y^{(i)}|$ 则 m 个样本点的误差和为 $\sum_{i=1}^{m} |h_\theta(x^{(i)}) - y^{(i)}|$ ，因此问题就转化为求解 $\arg\min_\theta \sum_{i=1}^{m} |h_\theta(x^{(i)}) - y^{(i)}|$ 为了后续求解最优值(绝对值函数不好求导)，所以损失函数采用了误差平方和的形式 $\arg\min_\theta \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ 。

回归

- 问题转化为求 J_θ的最小值问题。
- 步骤包括:
- 1. 对目标函数求导;
- 2. 令其导数为0，求得极值。
- 如果该函数是凸函数，极值点就是最值点。
- 这就是著名方法——最小二乘法的基本思想。



梯度下降法

优化算法

在模型的参数没有解析解（闭式解）的情况下，可以通过求损失函数的最小值来获得参数的最优解，这是一个典型的最优化问题。此时就需要使用优化算法，这些算法大多数都是通过多次迭代来逼近参数的最优解。

在这些迭代法中，梯度下降算法是最常用的算法策略之一。

梯度下降

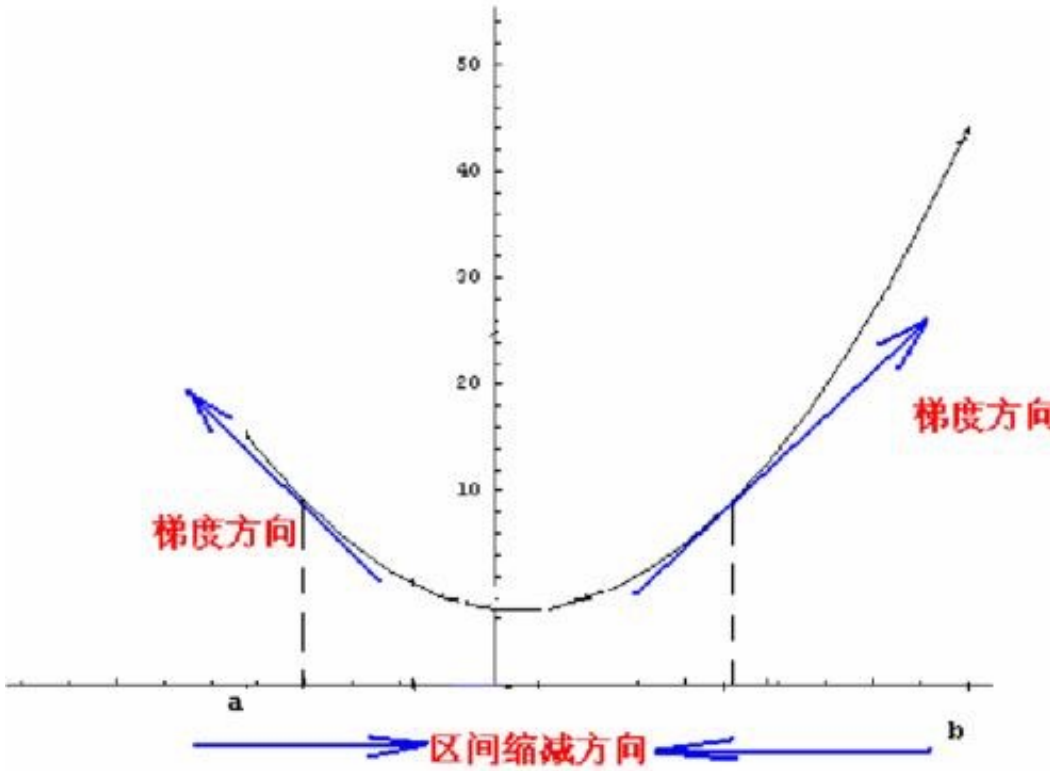
梯度下降（gradient descent）在机器学习中应用十分广泛，其目的是通过迭代找到目标函数的最小值，或者收敛到最小值。在监督学习中，梯度下降算法是找到损失函数的最小值对应的参数，此时的参数就是参数的最优解。

梯度下降的过程和下山的场景很类似，其目的和策略都是尽快达到山脚下。最快的下山的方式就是找到当前位置最陡峭的方向，然后沿着此方向向下走，对应到函数中，就是找到给定点的梯度，然后朝着梯度相反的方向，就能让函数值下降的最快，因为梯度的方向就是函数值变化最快的方向，而梯度相反的方向，就是函数值下降最快的方向。

梯度下降法

- 在单变量的函数中，梯度就是函数的导数，代表着函数在某个给定点的切线的斜率，其梯度是一个一维向量；

- 而在多变量函数中，梯度是一个多维向量。向量有方向，梯度的方向就是函数在给定点的上升最快的方向，因此，负的梯度方向，就是函数值下降最快的方向。



梯度下降

例如，有函数 $f(x) = x^2$，则该函数的梯度为其导数：

$$f'(x) = 2x。$$

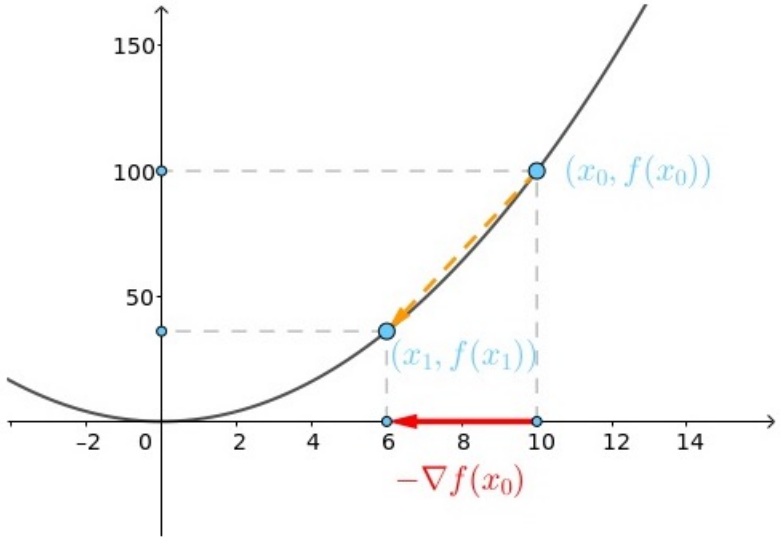
现在需要计算该函数有最小值时对应的x值0，随机给定该参数的初始值x_0为10，在此处其梯度值为一维向量，其值为20，梯度下降的方向是梯度值的反方向，即为-20，该值表示x轴的负方向。在确定了方向之后，还需要控制该方向上的距离，此处引入另外一个值，称为**步长**，一般使用η来表示。此处步长采用0.2，于是得到：

$$x_1 = x_0 - \eta * f'(x) = 10 - 0.2 * 20 = 6$$

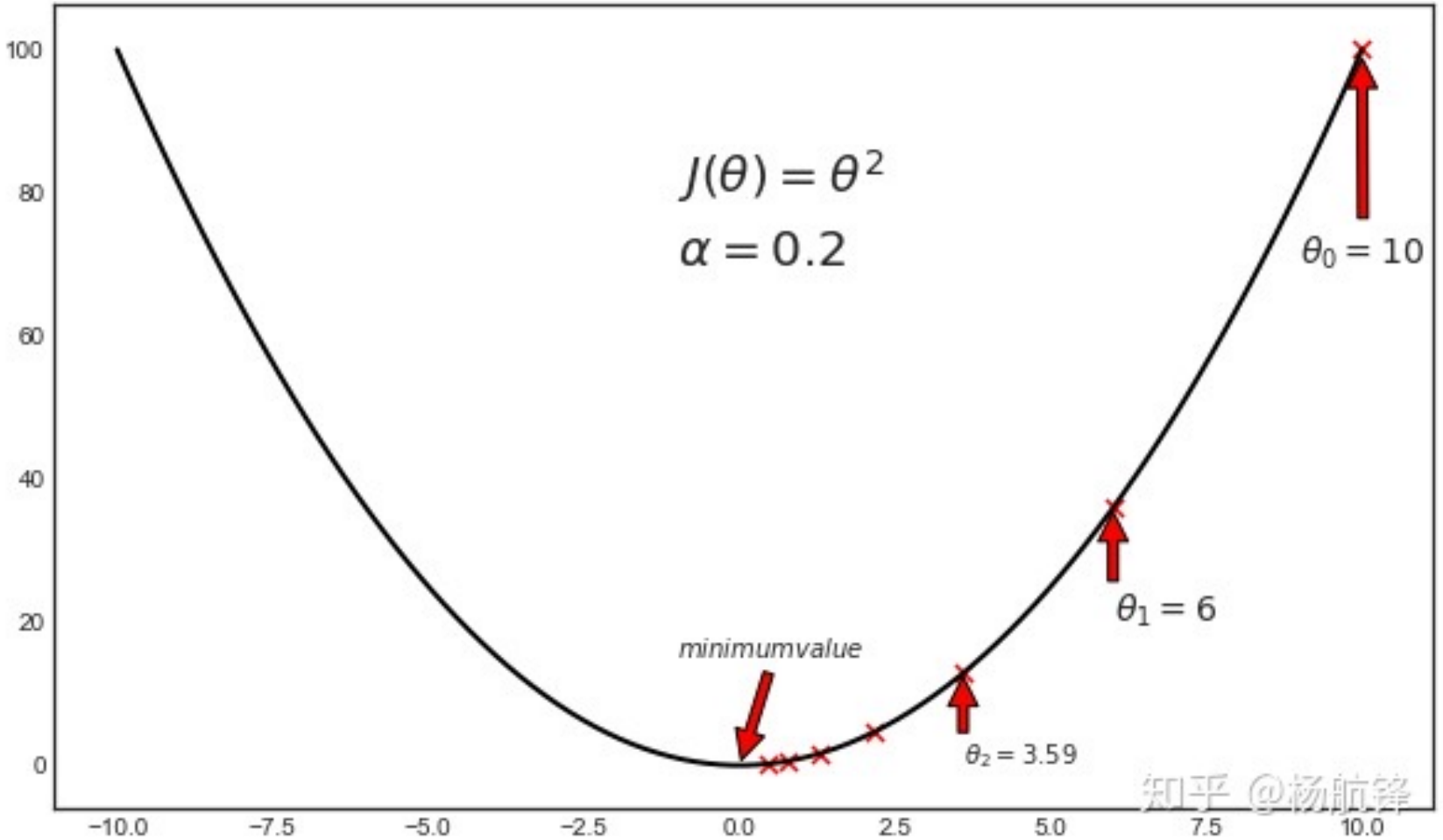
梯度下降

$$x_1 = x_0 - \eta * f'(x) = 10 - 0.2 * 20 = 6$$

其几何意义如下图所示，图中$-\nabla f(x_0)$就表示函数$f(x)$在x_0处的负梯度方向。上式就是采用梯度下降算法进行了1次迭代的情况。进行类似的多次迭代之后，x就会逼近最优解0了。



梯度下降法


$J(\theta) = \theta^2$
$\alpha = 0.2$
$\theta_0 = 10$
$\theta_1 = 6$
$\theta_2 = 3.59$
minimumvalue
100
80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
知乎 @杨航锋

梯度下降

下表中列出了前10次迭代中x的值，可以看出，只经过了10次迭代，x的值已经小于0.1了，距离最优解0已经很接近了。

迭代次数	0	1	2	3	4	5	6	7	8	9	10
x	10	6.0	3.6	2.16	1.296	0.778	0.467	0.28	0.168	0.101	0.06
$f(x)=x^2$	100	36.0	12.96	4.666	1.68	0.605	0.218	0.078	0.028	0.01	0.0036

梯度下降算法

梯度下降法是一种迭代算法，迭代更新参数θ的值，逐步实现目标函数的极小化。

具体的更新过程如下：

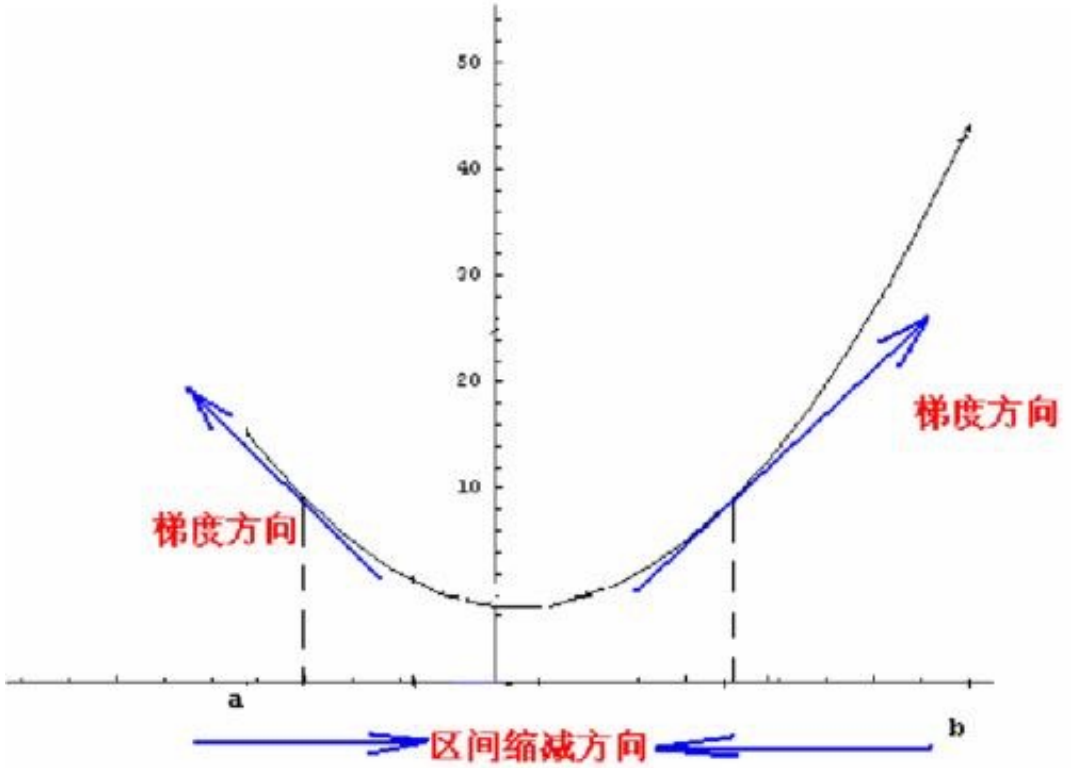
1. 随机初始化参数 θ
2. 迭代新的θ使得 J(θ)更小

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

$$\theta_j := \theta_j - \alpha\frac{\partial J(\Theta)}{\partial \theta_j}$$

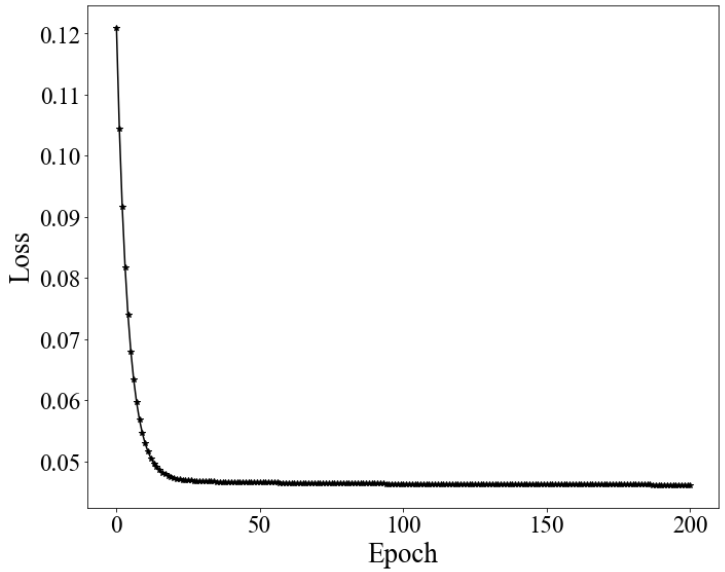
3. 如果 J(θ) 能够继续减少，则返回 2

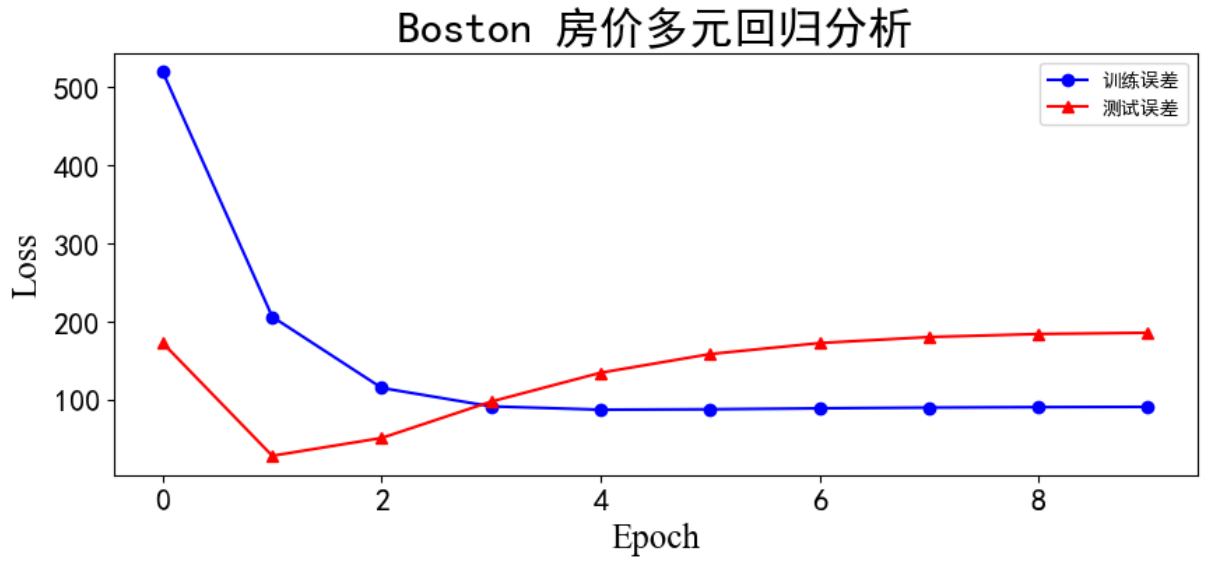
α 为步长（学习速率）：超参数



梯度下降

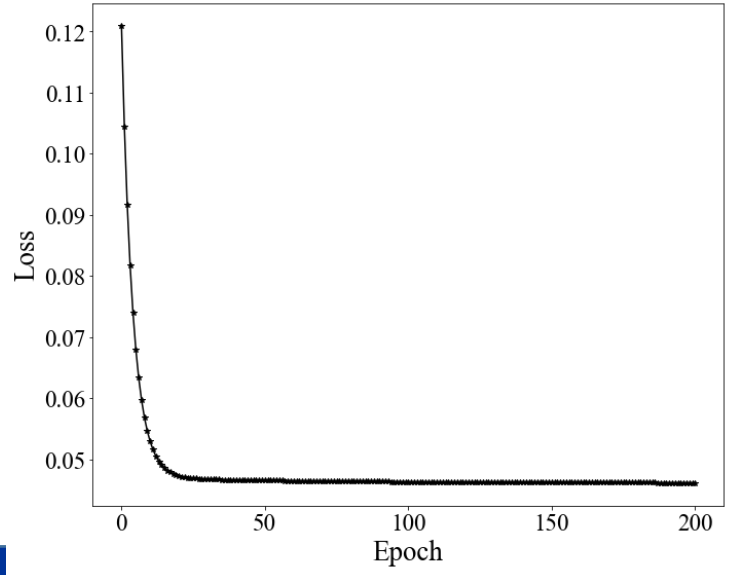
在多元函数中，函数的梯度是一个多维向量，向量的维度等于自变量X的个数，该向量每个维度上的值就是函数在自变量X该分量上的偏导数。
在监督学习中，一般情况下都是使损失函数获得最小值，此时的参数值即为最优解。一元线性回归和多元线性回归（Boston房价数据）的任务，采用梯度下降算法，其运行结果如下图所示。

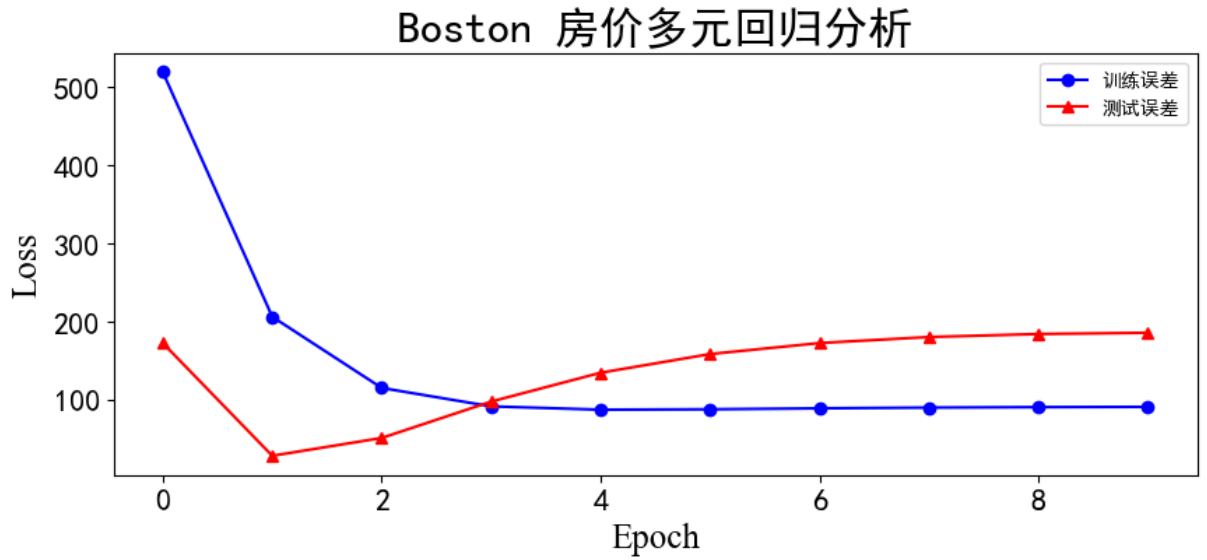




梯度下降

在图中，横坐标是Epoch，它的意思是一代，1个Epoch表示1次梯度下降的迭代，纵坐标是损失函数。从左图可以看出，经过约30次迭代之后，损失函数的值就很接近最后收敛的值了。而在右图中，经过2~3次迭代就达到了损失函数的最小值了，另外还可以看出，随着迭代次数的增加，虽然训练误差越来越小，但是测试误差在缓慢增加，因此，梯度下降算法的迭代次数并不是越多越好，迭代次数过多有可能会导致过拟合现象。





梯度下降

梯度下降法作为机器学习中较常使用的优化算法，具体又有三种不同的形式：

- 批量梯度下降（Batch Gradient Descent，BGD）
- 随机梯度下降（Stochastic Gradient Descent，SGD）
- 小批量梯度下降（Mini-Batch Gradient Descent，MBGD）

其中小批量梯度下降法MBGD也常用在深度学习中进行模型的训练。

梯度下降

批量梯度下降法是梯度下降算法中最原始的形式，它是指在每一次迭代时使用所有样本来进行梯度的更新。在批量梯度下降算法中，用来对参数进行更新的梯度是对所有样本梯度的平均值。在更新参数时需要先对所有样本的梯度值求和再计算平均值，因此需要对所有样本进行计算处理。

批量梯度下降算法的优点：由所有样本数据确定的方向能够更好地代表样本总体，从而更准确地朝向极值所在的方向。当目标函数为凸函数时，使用该算法一定能够得到全局最优解。其缺点是当样本数目很大时，每迭代一步都需要对所有样本计算，训练过程会很慢。

梯度下降

随机梯度下降法不同于批量梯度下降，随机梯度下降是每次迭代随机地使用一个样本来对参数进行更新，使得训练速度加快。

使用随机梯度下降算法，由于不是在全部训练数据上的损失函数，而是在每轮迭代中随机优化某一条训练数据上的损失函数，因此每一轮参数的更新速度大大加快。其缺点包括：准确度下降，由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛；由于单个样本并不能代表全体样本的趋势，可能会收敛到局部最优解而不是全局最优解。

梯度下降

小批量梯度下降，是对批量梯度下降和随机梯度下降的一个折中办法。

其思想是：每次迭代使用若干个样本（称为批量，batch）来对参数进行更新。在MBGD方法中，每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近BGD的效果。例如，如果样本总数为10万，batch的大小设置为512，则1个Epoch内需要进行196次（10万/512）小批量梯度下降。

梯度下降

小批量的梯度下降可以利用矩阵和向量计算进行加速，还可以减少参数更新的方差，得到更稳定的收敛。

在MBGD中，开始阶段步长一般设置的比较大，随着训练不断进行，可以动态地减小步长，这样可以保证一开始算法收敛速度较快。

超参数

在上述梯度下降算法中，使用梯度的负方向确定了下降最快的方向，然后使用步长η来决定在梯度的负方向上下降多大的距离。另外，在迭代之前还需要确定迭代的次数。因此，步长η和迭代次数是运行梯度下降算法之前需要人为来设置的值，之后运行梯度下降算法，该算法用来获得模型中参数（例如一元线性回归中的w和b）的最优解。

一般把机器学习算法运行之前、由人为来设置值的步长η和迭代次数等称为超参数，以和模型中的参数相区别。机器学习（包括人工神经网络和深度学习等）中常说的“调参”，指的就是调节超参数的值。

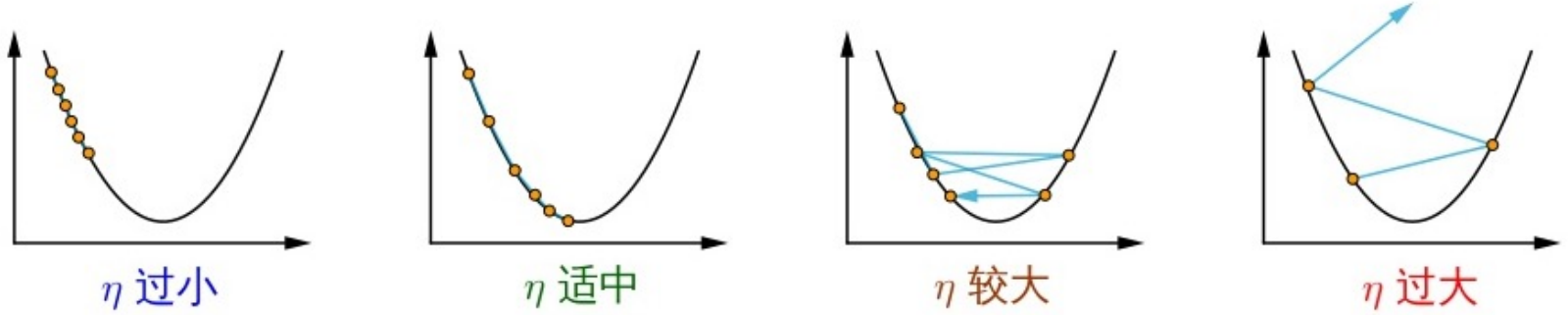
超参数

模型的超参数是模型外部的配置，其值不能从数据估计得到，其特征包括：

① 超参数常应用于估计模型参数的过程中；
② 超参数通常人为地在算法运行之前提前指定；
③ 超参数通常可以使用启发式方法来设置；
④ 超参数通常根据给定的预测建模问题而调整。

超参数

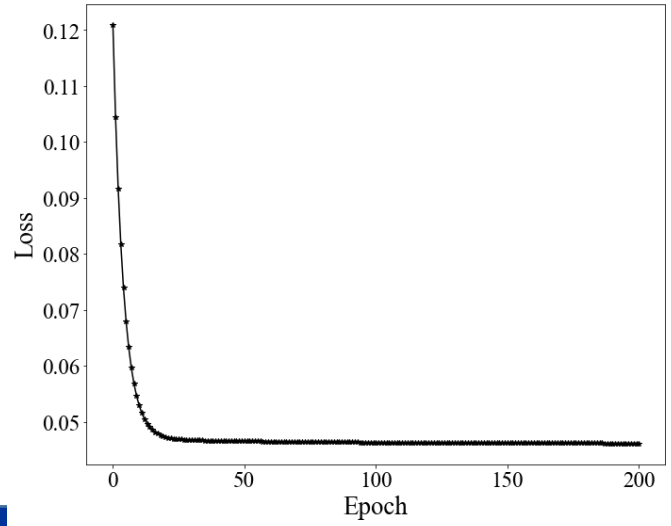
对于给定的机器学习问题，一般无法知道模型超参数的最优值。但可以使用经验法则来探寻其最优值，或复制用于其他问题的值，也可以通过反复试验的方法来获得超参数的最优值。
在上一小节 梯度下降算法中已经讨论了迭代次数设置的情况，此处就步长的设置进行说明。步长，也称为学习率（learning rate），是梯度下降算法中常用的超参数之一。下图说明了步长不同的设置情况下梯度下降算法运行的情况。

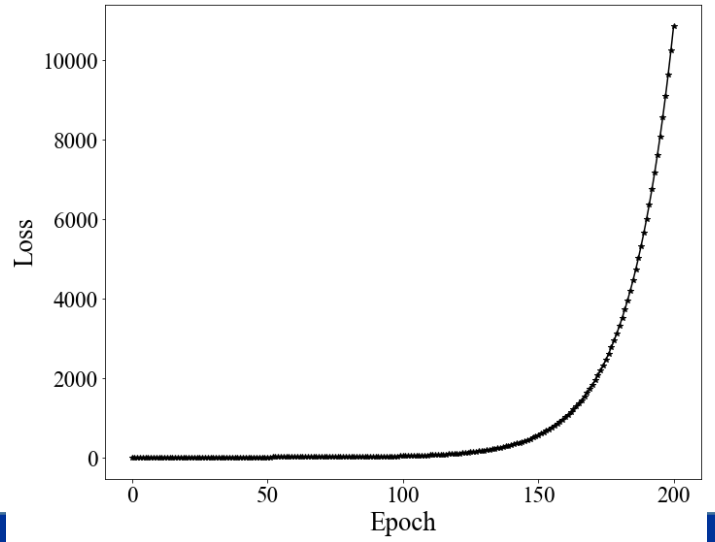


超参数

如果步长设置过小，则目标函数值下降很慢，需要很多次迭代才能完成优化；如果设置较大，则可能会发生震荡而无法达到极值点。步长设置过大也容易导致无法达到极值点。

左图是一元线性回归采用梯度下降算法进行建模时损失函数随迭代次数的增加而逐渐降低，此时的步长值为0.1。如果把步长设置为1.72或更大的值，则随着迭代次数的增加损失函数也逐渐增加，从而优化算法失败，如右图所示。





参数与超参数

参数就是模型内部的配置变量，可以用数据估计它的值。

参数有以下特征：

- 进行模型预测时需要模型参数；
- 模型参数值可以定义模型功能；
- 模型参数用数据估计或数据学习得到；
- 模型参数一般不由实践者手动设置；
- 模型参数通常作为学习模型的一部分保存。

通常使用优化算法估计模型参数，优化算法是对参数的可能值进行的一种有效搜索。

参数与超参数

模型超参数是模型外部的配置，其值不能从数据估计得到。

具体特征有：

- 超参数常应用于估计模型参数的过程中；
- 超参数通常由实践者直接指定；
- 超参数通常可以使用启发式方法来设置；
- 超参数通常根据给定的预测建模问题而调整。

对于给定的问题，无法知道模型超参数的最优值。

但可以使用经验法则来探寻其最优值，或复制用于其它问题的值，也可以通过反复试验的方法。

梯度下降法求解凸优化问题

- 梯度下降算法不一定能够找到全局的最优解，有可能是一个局部最优解。然而，如果损失函数是凸函数，那么梯度下降法得到的解就一定是全局最优解

线性回归算法

岭回归

$$\min_{w} \|Xw-y\|_2^2+\alpha\|w\|_2^2$$

lasso回归

$$\min_{w} \frac{1}{2n}\|Xw-y\|_2^2+\alpha\|w\|_1$$

ElasticNet回归

$$\min_{w} \frac{1}{2n}\|Xw-y\|_2^2+\alpha\rho\|w\|_1+\frac{\alpha(1-\rho)}{2}\|w\|_2^2$$

多项式回归

$$\hat{y}(w,x)=w_0+w_1x_1+w_2x_2$$

$$\hat{y}(w,x)=w_0+w_1x_1+w_2x_2+w_3x_1x_2+w_4x_1^{\ 2}+w_5x_1^{\ 2}$$

$$z=[x_1,x_2,x_1x_2,x_1^2,x_2^2]$$

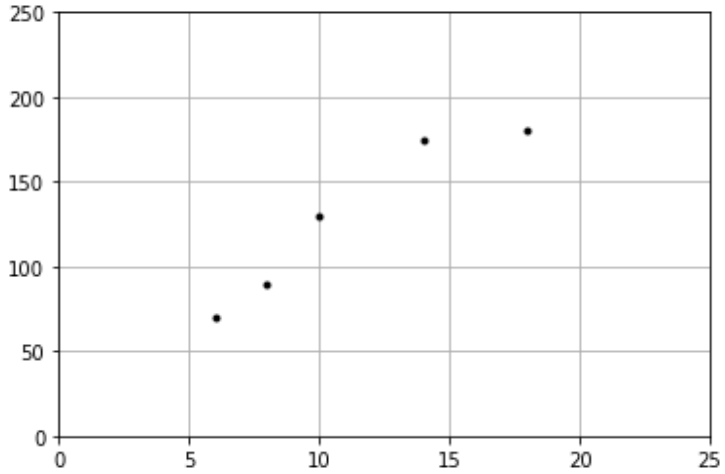
$$\hat{y}(w,x)=w_0+w_1z_1+w_2z_2+w_3z_3+w_4z_4+w_5z_5$$

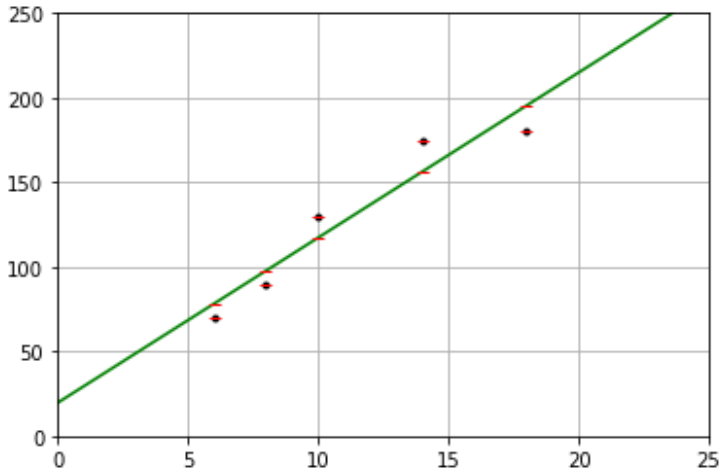
一元线性回归

用给定的数据来建立披萨的价格与其大小之间的线性回归方程。

序号	披萨直径	披萨价格
1	6	70
2	8	90
3	10	130
4	14	175
5	18	180

一元线性回归



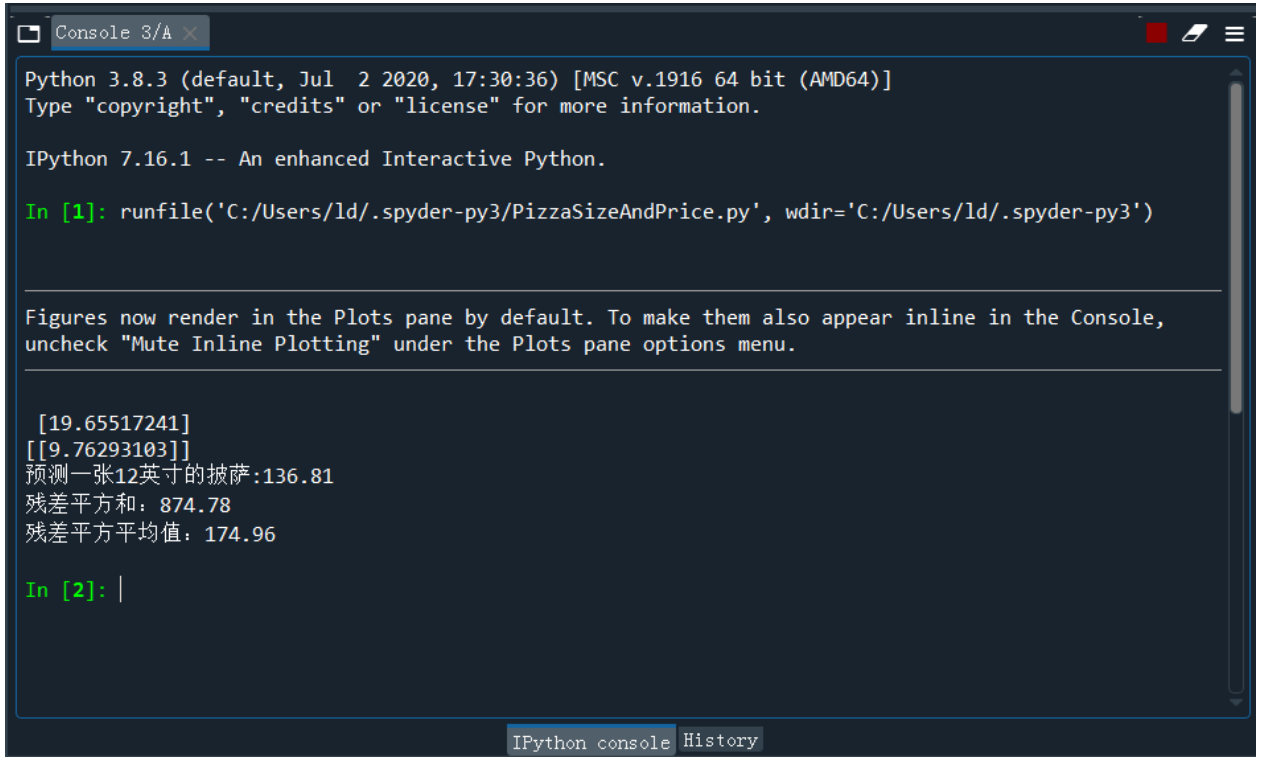


线性回归方程：

$$y = 19.66 + 9.76 * x$$

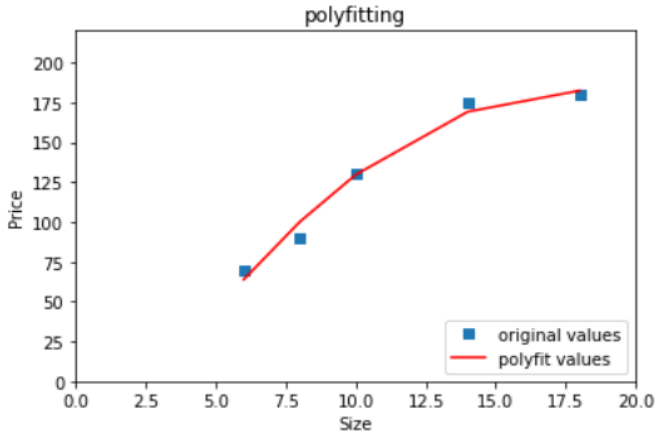
θ_0 θ_1

一元线性回归


Console 3/A
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.16.1 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/ld/.spyder-py3/PizzaSizeAndPrice.py', wdir='C:/Users/ld/.spyder-py3')
Figures now render in the Plots pane by default. To make them also appear inline in the Console,
uncheck "Mute Inline Plotting" under the Plots pane options menu.
[19.65517241]
[[9.76293103]]
预测一张12英寸的披萨:136.81
残差平方和：874.78
残差平方平均值：174.96
In [2]:
IPython console
History

一元线性回归

可以引入 x 的二次项，使用含有3个参数的多项式回归方程（n = 2)




```
Console 1/A

In [32]: runfile('C:/Users/ld/.spyder-py3/PizzaSizeAndPriceQuadratic.py',
x is :
 [ 6  8 10 14 18]
y is :
 [ 70  90 130 175 180]
f1 is :
 [ -0.82022921  29.56156716 -83.97654584]
p1 is :
          2
-0.8202 x + 29.56 x - 83.98
yvals is :
 [ 63.86460554 100.02132196 129.61620469 169.12046908 182.37739872]

In [33]:
```

一元线性回归

- 引入x的二次项之后的回归方程是：

$$y = -0.8202\,x^2 + 29.56\,x - 83.98$$

- 该模型还是线性回归模型吗？

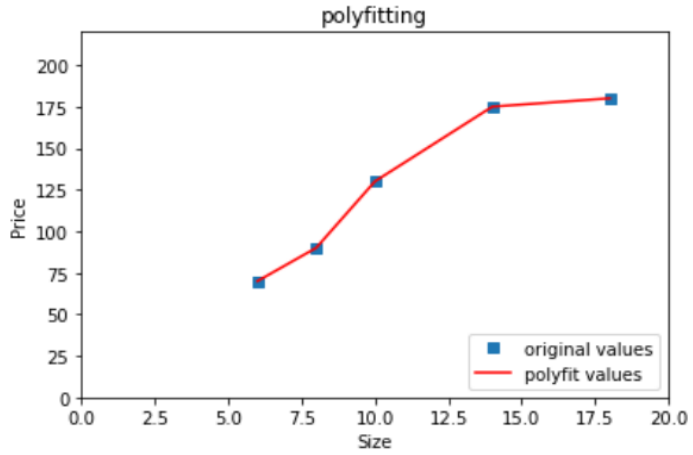
- 一元m次多项式回归方程：

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_m x^m$$

- 多项式回归模型是线性回归模型的一种，此时回归方程关于回归系数是线性的，并不是说只能使用x的一次项。

一元线性回归

继续引入 x 的高次方项（n = 4）
含有 5 个参数




```
Console 1/A
p1 is :
          4         3         2
0.04297 x - 2.128 x + 36.89 x - 251.7 x + 656.2
yvals is :
 [ 70.  90. 130. 175. 180.]

In [36]: runfile('C:/Users/ld/.spyder-py3/PizzaSizeAndPriceQuadratic.py',
x is :
 [ 6  8 10 14 18]
y is :
 [ 70  90 130 175 180]
f1 is :
 [ 4.29687500e-02 -2.12760417e+00  3.68906250e+01 -2.51739583e+02
  6.56250000e+02]
p1 is :
          4         3         2
0.04297 x - 2.128 x + 36.89 x - 251.7 x + 656.2
yvals is :
 [ 70.  90. 130. 175. 180.]

In [37]:
```

（2）逻辑回归

- 逻辑回归又称为**逻辑回归分析**，是通过历史数据的表现对未来结果发生的概率进行预测。
- 尽管逻辑回归输出的是实数值，但本质上它**是一种分类方法，而不是回归方法**。
- 逻辑回归的自变量可以有一个，也可以有多个。
- 逻辑回归的因变量可以是二分类，也可以是多分类。二分类更为常用，也更容易解释。
- 若采用**sigmoid函数**计算概率，令阈值为0.5，则完成**二分类**任务。

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

- 若采用**softmax函数**计算概率，则逻辑回归可完成**多分类**任务。

$$f(x_i)=\frac{e^{x_i}}{\sum_{k=1}^{M} e^{x_k}} \quad i=1,\ldots,M \quad ; \quad M\text{为类别数}$$

输出的实数表示未知样本 x 属于某一类别的概率.

线性回归与逻辑回归的异同点

- 线性回归与逻辑回归的**区别**在于：
  - 线性回归用于**预测连续值**，其输出的值域是实数集，其模型是线性的；
  - 逻辑回归主要用于**解决分类问题**，其输出的值域为[0,1]，其模型是非线性的。
- 线性回归与逻辑回归的**共同点**在于：

  两者的输入数据既可以是连续的值，也可以是离散的值。