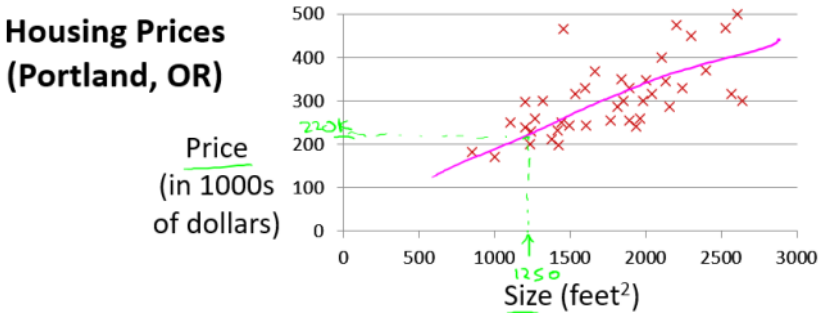
2 一元线性回归

2022年10月17日 15:50

2.1模型描述

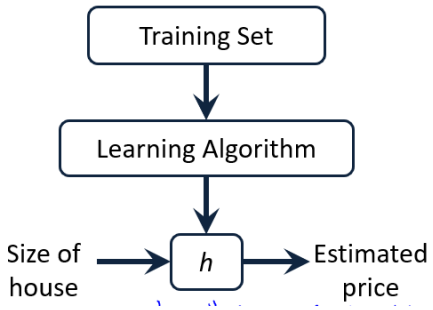
预测住房价格的例子：



这是一个监督学习的例子，因为每一个例子他都给出了“正确答案”，且属于一个回归问题。

常见的符号表示：
m:训练样本个数或者说样本容量
x:输入变量
y:输出变量
(x,y)一个训练样本
$(x^{(i)}, y^{(i)})$:一个特定的第i个训练样本

假设函数(hypothesis)：
将输入映射到输出的函数。
数学表示：$h_\theta(x) = \theta_0 + \theta_1 x$，$\theta_0$和$\theta_1$为模型参数
(针对本例的一元线性回归)



不同的模型参数构成不同的假设函数，选择最佳的模型参数很关键。

2.2代价函数

线性回归的整体目标函数：$\underset{\theta_0\theta_1}{minimize} \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

寻找使整个函数值最小的θ_0和θ_1

定义**代价函数**(Cost Funcation, Squared error function)：

记$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$，则代价函数为$\text{minimize}_{\theta_0,\theta_1} J(\theta_0, \theta_1)$.

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

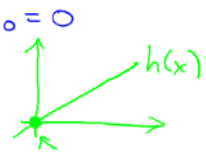
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

对上述函数进行简化，令$\theta_0 = 0$：

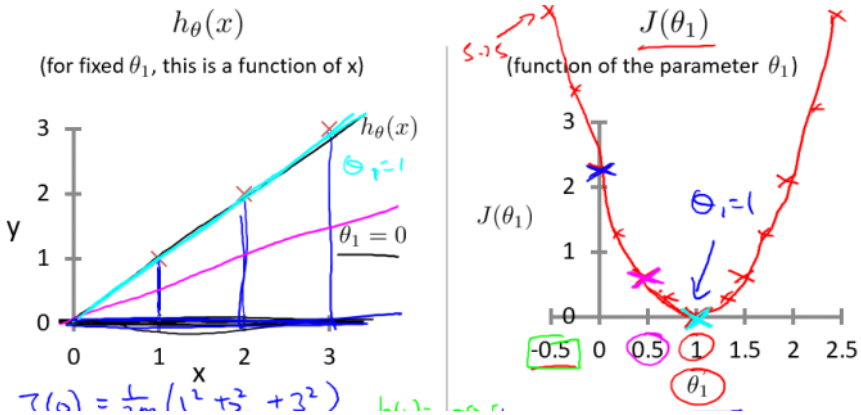
$$h_\theta(x) = \theta_1 x$$

θ_1



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

$$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$$

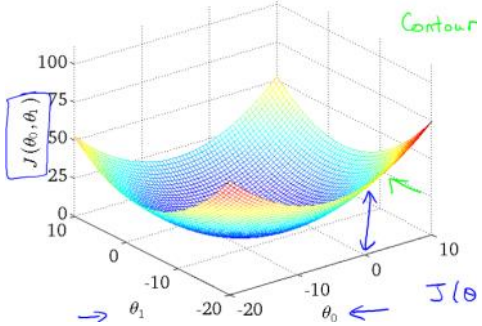


每一个θ_1对应一个不同的假设函数，也即对应一个不同的代价函数的值。

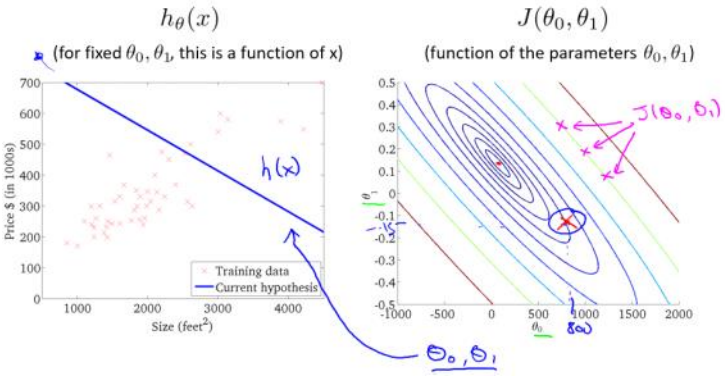
学习算法的优化目标是通过选择合适的模型参数获得最小的代价函数值。

在本例中$\theta_1 = 1$对应的回归函数就是最符合数据的直线，从而完美地拟合了数据集。

当θ_0和θ_1都变化时，可以得到一个二变量的代价函数。



一般可以用等高线对三维图像进行表示：



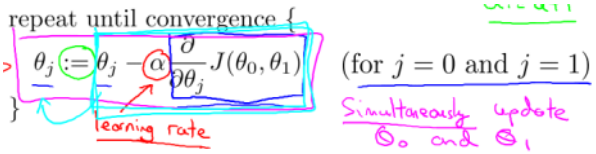
2.3梯度下降

梯度下降：一种求函数最小化方法的算法。

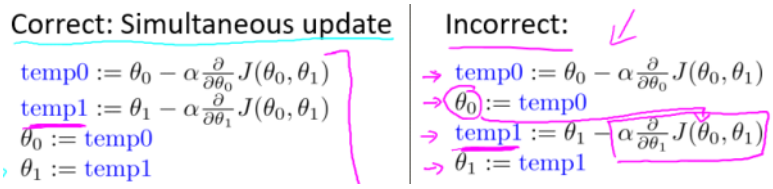
以二元函数为例，梯度下降的一般方法为：

- 给定θ_0和θ_1的初始值，一般都设为0
- 改变两个参数值来减小函数大小，直到找到最小值(局部最小值)后停止算法

算法的底层数学原理：

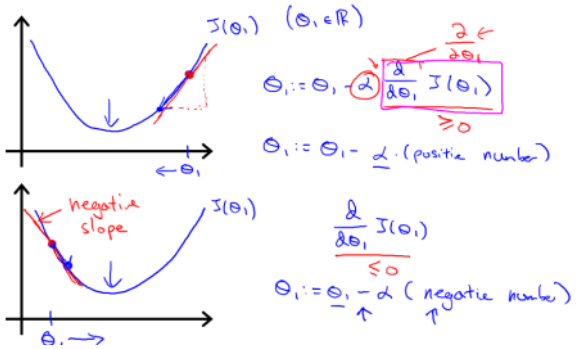


需要注意的是，**应当保证两个参数的同时更新**，即对同一个代价函数的运算：



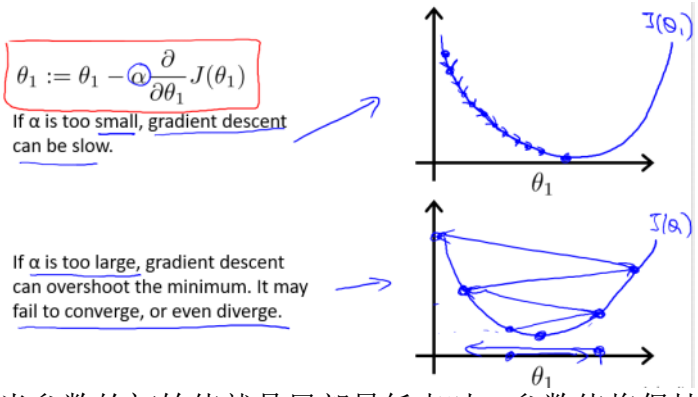
对于偏导的理解：

以一元函数为例



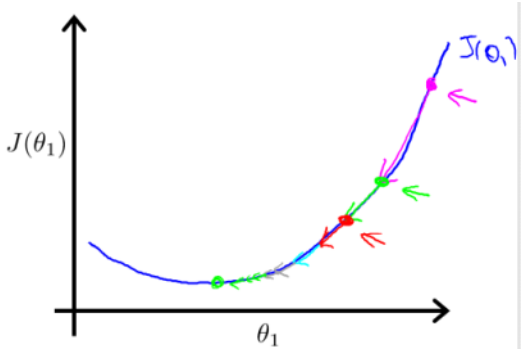
参数减去学习率乘以该点的偏导值使参数逼近于使函数最小的点对应的变量值。

学习率反映了从“山坡”往下走时的“步长”的大小，学习率的设置很重要。
学习率太小，向最低点迈进的“步长”就会很小，从而计算更多次；学习率太大，梯度下降可能会越过最低点，最后甚至无法收敛甚至发散。



对于特殊的情况：当参数的初始值就是局部最低点时，参数值将保持原值不再更新。

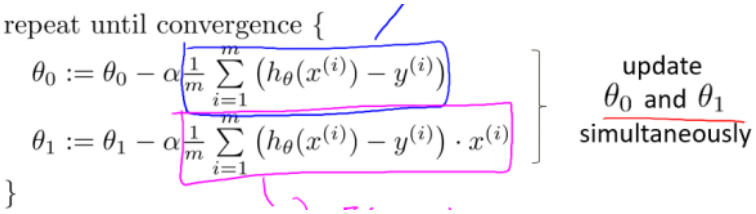
随着参数所对应的代价函数值越来越接近局部最低点，其导数也会越来越接近0，那么“步长”也会越来越小，所以额外减小学习率似乎并不是必要的。



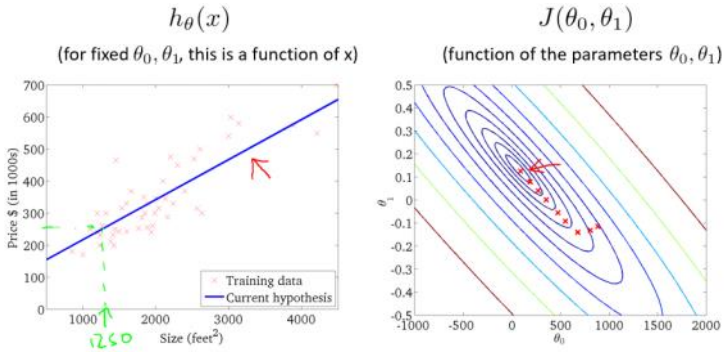
2.4线性回归的梯度下降

将梯度下降运用到最小化平方差代价函数中，来拟合线性回归直线。

将假设函数和代价函数带入梯度下降的公式，从而得到：



梯度下降的一个问题就是它容易陷入局部最优解中。而一元线性回归的代价函数为一个凸函数，也即一个弓形的函数，它没有局部最优，而是全局最优的。从而对它进行梯度下降的计算只能得到全局最优解。



这种梯度下降又叫“Batch梯度下降”，它意味着每一步下降的过程遍历了整个训练集样本，也就是说每一步下降都计算了m个样本。

正规方程组的方法能直接计算代价函数的最小值，而梯度下降更适用于更大的数据集。