# 问题描述

一个房屋估价系统，需要得到房屋的属性（面积，年份，...）与房屋的价格之间的联系，并且已知可以找到一个近似线性的解，这个解如何来求得。

这里将问题简化，假设房屋只有一个属性面积，



根据以上数据，解决线性回归问题。

设x代表面积，y代表价格，这个问题就能抽象为y=w\*x+b

为了统一符号的需要，改写为y=w0\*x0+w1\*x1

其中w0=b,x0=1

目标就是找到能够很好拟合数据的w0,w1

# 二、问题符号化

符号定义：

 我们的权值。 训练样本。

需要拟合的线性方程形如：



即一种加权和的形式。

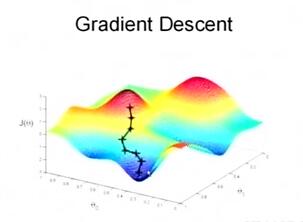
误差函数(loss function)：

欧几里得距离来度量误差。

梯度(gradient)：

这里需要理解梯度的物理含义。

使用梯度下降拟合线性方程：



图中每一段黑色轨迹即那个点梯度的方向。



其实就是求个偏导。

进一步展开公式，我们的误差方程转化为：

 这是展开的最终形式。

对它进行求导的时候，平方会提取出来和之前的1/2抵消，所以之前定义误差函数的时候乘以1/2。

求导结果即， 。。。。。。。。。。。易得

# 三、算法描述

结论：

过程：

什么是梯度下降？即每个参数都从偏导的方向进行更新。

所以，



学习率是啥？ 学习的步长，选取的时候根据情况而定。

何时可以判定为convergence：权值不怎么变化，或准确率到达一个阈值或迭代次数到达一定次数。

# 四、伪代码

初始化 

读取数据 

While(true){

求出当前的所有样本的估计值：mat\_pred = 

求出当前的误差：mat\_loss = mat\_pred - Y

求出每个参数的梯度: 

mat\_loss\_repeat = np.tile(mat\_loss,(1,dim))

mat\_gradient = np.multiply(mat\_loss\_repeat , X)

Gradient = np.sum(arr\_gradient,axis=0)

更新权值：W= W - learning\_rate\*Gradient

如果收敛，退出

}

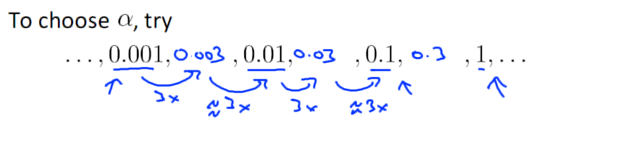
# 延伸知识

学习率的选取

学习率过大：步长过大容易错过最优解，导致不收敛

学习率过小：收敛非常慢

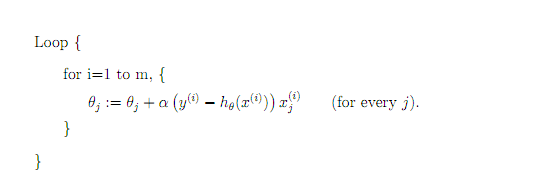
在选取学习率时需要不断尝试。



可以使用这种方法，逐步找到合适的学习率。

随机梯度下降

在上文提到的优化算法又称为**批量梯度下降算法**，在实际应用时因为要遍历每一个样本求得loss的总和，往往十分耗时，因此，有时会采用**随机梯度下降算法**，即利用一个样本，用这个样本的loss来更新权重，并不断循环，直至收敛。



# 总结

线性回归问题不能解决生活中的大部分问题，因此本文仅仅为了后面的内容做铺垫，并复习一部分求导和线性代数的知识，后面会提到非线性问题的解决方案。