作者：成清清 徐唱 肖凯 吕迪迪

根据维基百科[1]的定义，梯度下降(Gradient Descendent，GD) 法是一阶迭代式优化算法(First-Order Iterative Optimization Algorithm)。

我们首先举一个机器学习的问题：某地的房价与房地面积和卧室的数量之间成如下表的关系：

房子面积（平方英尺）	卧室数	价格（千美元）
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

根据这个已知数据，我们要通过分析上面的数据学习出一个模型（即价格和房子面积+卧室数之间的关系），用于预测其它情况（比如面积 2000,卧室数 5）的房价。这个问题我们可以看作是一个概率里的回归问题，属于个机器学习的问题。

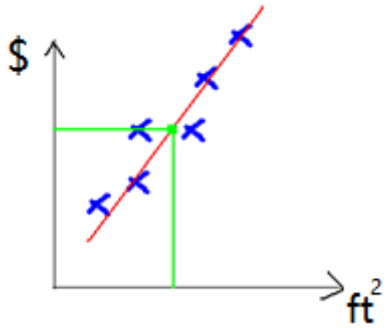

Fig.1:一维输入数据拟合示例

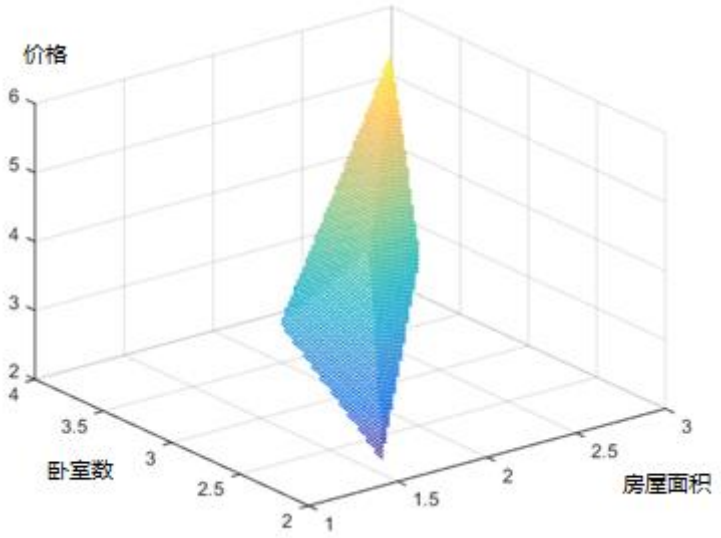

Fig.2: 表中数据 plot 后一个视角

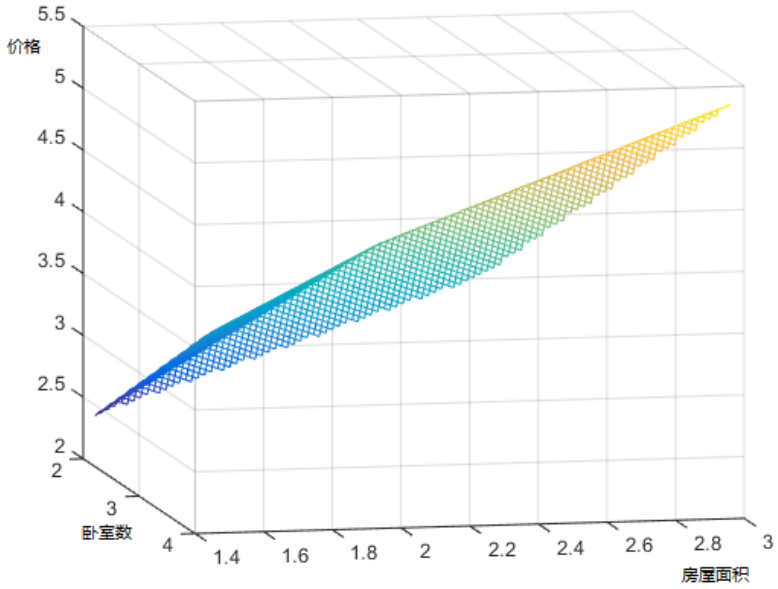

Fig.3: 表中数据 plot 后另一个视角

对于一维的输入数据，如 Fig.1 所示，我们需要的拟合函数实际上表现为一条直线。而对于二维的输入数据，我们需要得到的拟合函数实际上是一个平面。但从 Fig.2 和 Fig.3 能看出，直接 plot 出的数据并不是一个平面（Fig.3 中能看出有点接近平面），也可以说输入的三个向量数据构造成了一个接近平面（线性）的函数。我们的目的实际上是要从数据中进行优化，得到一个平面。

由于我们已经有了已知的数据，我们实际上是要从已知数据对未知数据的值进行预测，因此属于有监督学习范畴。

问题先放一下，我们现在来看看一些机器学习中的一些概念：

x_k表示“输入”变量（’input’ variables），也称为特征(features)

我们不使用特征值这样的说法，为了避免和 eigenvalue(中文翻译为特征值)这个概念混淆。

y_k表示“输出”变量(‘output’ variables)，也称为目标值(target)

一对(x_k, y_k)成为一个训练样本(training example)，用作训练的数据集就是一组m个训练样本$(x_k, y_k); k = 1, \ldots, m$ ，被称为训练集(training set)

为避免混淆，本文开始启用k代表样本数量，i代表维度。

$\boldsymbol{X}$表示输入变量的取值空间，$\boldsymbol{Y}$表示输出变量的取值空间。那么$\boldsymbol{h}: \boldsymbol{X} \to \boldsymbol{Y}$就是训练得到的映射函数，对于每个取值空间$\boldsymbol{X}$的取值，都能给出取值空间$\boldsymbol{Y}$上的一个预测值。函数$\boldsymbol{h}$这里实际做出了假设（hypothesis），即假设函数$\boldsymbol{h}$能够对$\boldsymbol{X} \to \boldsymbol{Y}$做出有效映射。

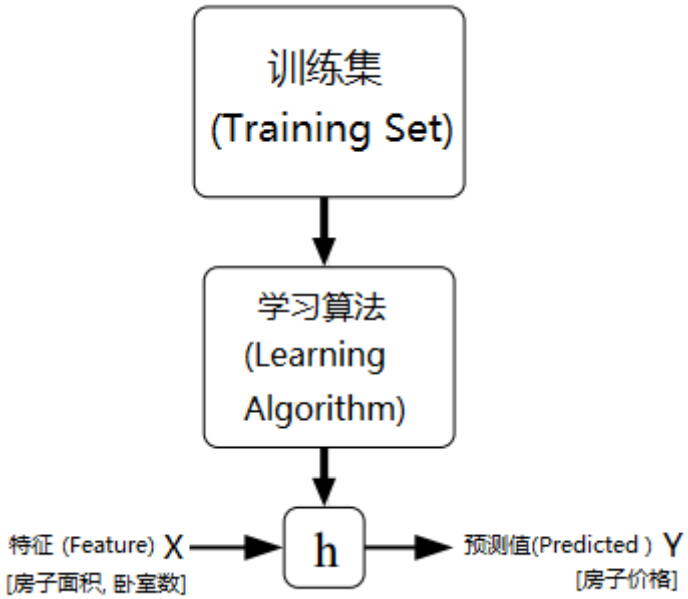

Fig.4: 机器学习基本框架

对于这个问题，当预测值$\boldsymbol{Y}$为连续值时，则有监督学习问题是回归(regression)问题；预测值$\boldsymbol{Y}$为离散值时，则为分类(Classification)问题。

回归及问题：

对于线性回归问题，先简单将$\boldsymbol{Y}$表示为$\boldsymbol{X}$的线性函数。我们用x_1，x_2.. x_n 去描述特征里面的分量，比如x_1=房子面积，x_2=卧室数量，等等，我们可以做出一个估计函数：

$$h(x) = h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

其中，θ称为参数(parameters) ，也叫做权重(weights)，参数决定了$\boldsymbol{X}$到$\boldsymbol{Y}$的射映空间。在这儿的意思是调整 feature 中每个分量的影响力，就是到底是房子面积更重要还是卧室数量更重要。用$x_0 = 1$来表示截距项(intercept term)，我们就可以用向量的方式来表示了：

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

注意这里的i代表特征的维度数量，上例中房子面积和卧室数各为一个维度。我们还应该再注意一点，i的取值范围从 0 开始，如果我们将此公式做出转换，利用上面的例子中二维输入数据，我们实际得到：$h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$ ，如果令$x_0 = 1$

有了训练集，如何通过学习得到参数θ?

常用的一种方法，让预测值$h(x)$尽量接近真实值$\boldsymbol{Y}$，定义成本函数(cost function)：

$$J(\theta) = \frac{1}{2}\sum_{k=1}^{m}(h_\theta(x_k) - y_k)^2$$

*有的文章在此公式中直接使用h，并没有θ下标。本文在h_θ中加入θ下标的目的是表明我们需要求得的函数$\boldsymbol{h}$是由权重θ决定的。在回归问题中，我们现在较多采用的是$h_\theta = \theta^T x$这种线性模型，但实际上还有一些其他的模型(指数，幂，对数形式)，其中θ^T作为决定函数的参量，x作为变量，两者维数一致，实际问题中将θ称为权重，实则是它描述了x中每个分量对最后估计的$\mathbf{y}$所起的决定作用的大小。(见解提供人：@熊均达@致远学院@SJTU)。

这个错误估计函数是去对x_k的估计值与真实值y_k差的平方和作为错误估计函数，前面乘上的$\frac{1}{2}$是为了在求导的时候，这个系数就不见了。

这实际上就是最小二乘成本函数，我们把这个回归模型叫做普通最小二乘回归模型(ordinary least squares regression model)。我们要尽量选择最优的θ，使$J(\theta)$得到全局最小值。而 $J(\theta)$到了全局最小值，也意味这我们的预测值$h(x)$和实际测试数据误差最小。

分类问题：

分类问题在这里实际上就是LMS算法的实现。这里依然需要利用刚才在线性回归中求得的成本函数$J(\theta)$。为了找到使成本函数$J(\theta)$最小的参数θ，采用搜索算法：给定一个θ的初值，然后不断改进，每次改进都使$J(\theta)$更小，直到最小化$J(\theta)$的θ的值收敛。这其实就是梯度下降法的基本思想。

梯度下降法：

梯度下降法是按下面的流程进行的：

1） 首先对θ赋值，这个值可以是随机的，也可以让θ是一个全零的向量。

*但是这里要注意，对于非凸问题，初始值的选取非常重要，因为梯度下降对初始值选取非常敏感，也就是说初始值选取直接影响着实际问题的表现。

2）改变θ的值，使得$J(\theta)$按梯度下降的方向进行减少。

为了更清楚，给出下面的Fig.5：注意为了这里方便显示，我们又回到一维的输入变量：θ_0变成了截距项，θ_1是斜率了。我们在一维和二维的输入数据上绕来绕去，是希望大家不要把输入数据局限在直线的拟合上，机器学习要处理的维度经常很大。大家可以看英文的参考文献【3】，作者把直线的拟合介绍得很清晰，但并没有上升到多维输入空间。

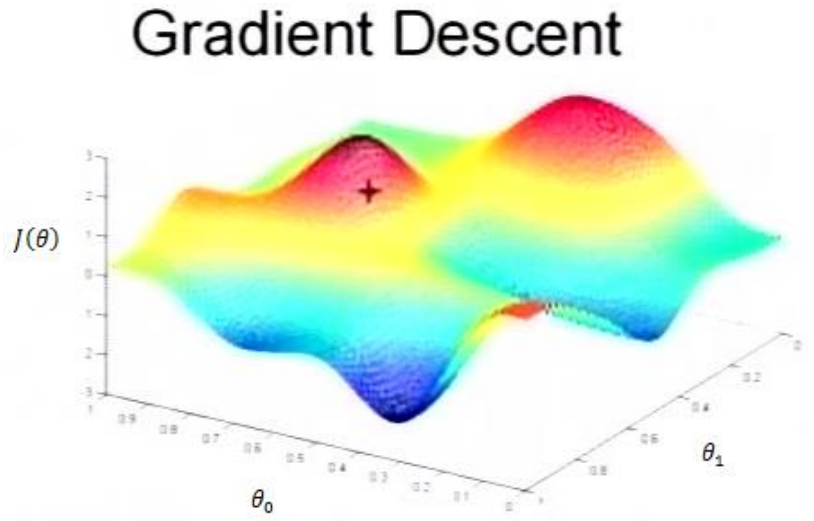

Fig.5: 参数与成本函数关系图

Fig.5是一个表示参数θ与成本函数$J(\theta)$的关系图，红色的部分是表示$J(\theta)$有着比较高的取值，我们需要的是，能够让$J(\theta)$的值尽量的低。也就是深蓝色的部分。θ_0，θ_1表示θ向量的两个

维度。

在上面提到梯度下降法的第一步是给θ给一个初值，假设随机给的初值是在图上的十字点。然后我们将θ按照梯度下降的方向进行调整，就会使得$J(\theta)$往更低的方向进行变化，如图所示，算法的结束将是在θ下降到无法继续下降为止。

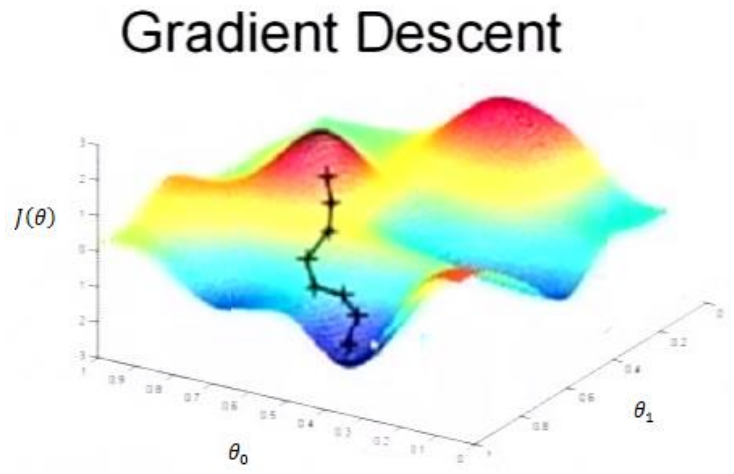

Fig.6：梯度下降法示意图

当然，可能梯度下降的最终点并非是全局最小点，可能是一个局部最小点，可能是下面的情况：

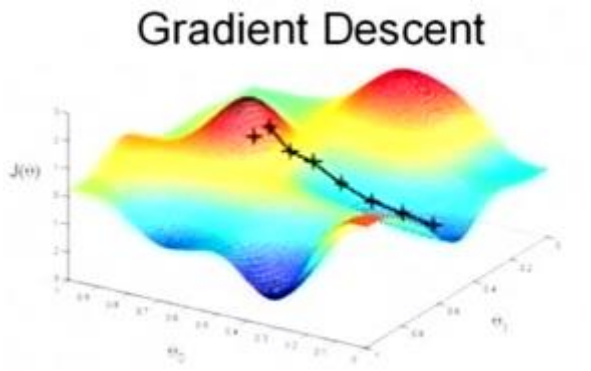

Fig.7：梯度下降法陷入局部最优示例

上面这张图就是描述的一个局部最小点，这是我们重新选择了一个初始点得到的，看来我们这个算法将会在很大的程度上被初始点的选择影响而陷入局部最小点

我们用数学方式描述梯度下降法从初始θ开始，不断更新：

$$\theta_i = \theta_i + \alpha \frac{\delta J(\theta_i)}{\delta \theta_i}$$

注意，更新是对所有$i = 1, \ldots, n$的θ_i值，即各个维度同时进行。α被称作学习率(learning rate)，也是梯度下降的长度，若α取值较小，则收敛的时间较长；相反，若α取值较大，则可能错过最优值。

假设我们只有一个训练样本(x, y)（注意，这里x代表输入特征，是个向量，在我们例子中有房子面积和卧室数两个维度；y代表输出，一般是个数值），此时$J(\theta) = \frac{1}{2}(h_\theta(x) - y)^2$，求偏导项得到：

$$\begin{aligned}\frac{\delta}{\delta\theta_i}J(\theta) &= \frac{\delta}{\delta\theta_i}\frac{1}{2}(h_\theta(x) - y)^2 \\ &= (h_\theta(x) - y) * \frac{\delta}{\delta\theta_i}(h_\theta(x) - y) \\ &= \left((h_\theta(x) - y)\right) * \frac{\delta}{\delta\theta_i}(\sum_{l=0}^{n}\theta_l x_l - y) \\ &= (h_\theta(x) - y) * x_i\end{aligned}$$

下面是更新的过程，也就是θ_i会向着梯度最小的方向进行减少。θ_i表示更新之前的值，后面的部分表示按梯度方向减少的量，学习率α表示每次按照梯度减少的方向变化多少。按照以下式子更新θ_i的值：

$$\theta_i = \theta_i + \alpha\left(y_k - h_\theta(x_k)\right) * x_{k,i}$$

一个很重要的地方值得注意的是，梯度是有方向的，对于一个向量θ，每一维分量θ_i都可以求出一个梯度的方向，我们就可以找到一个整体的方向，在变化的时候，我们就朝着下降最多的方向进行变化就可以达到一个最小点，不管它是局部的还是全局的。

我们再用更简单的语言来描述这个过程：

$$\nabla_\theta J = \begin{bmatrix}\frac{\delta}{\delta\theta_0}J \\ \vdots \\ \frac{\delta}{\delta\theta_n}J\end{bmatrix}$$

$$\theta = \theta - \alpha\nabla_\theta J$$

倒三角形表示梯度，按这种方式来表示，θ_i就不见了，看看用好向量和矩阵，真的会大大的简化数学的描述啊。

一些思考：

1. 如何解决局部最优问题？也就是说，非凸问题有没有好的解决方法？
2. 随机梯度下降和批量随机下降到底是怎么玩的？

参考文献：

1. https://en.wikipedia.org/wiki/Gradient_descent
2. http://www.cnblogs.com/LeftNotEasy/archive/2010/12/05/mathmatic_in_machine_learning_1_regression_and_gradient_descent.html
3. https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/