4、多变量线性回归(Linear Regression with Multiple Variables)

4.1 多维特征

参考视频: 4 - 1 - Multiple Features (8 min).mkv

目前为止，我们探讨了单变量/特征的回归模型，现在我们对房价模型增加更多的特征，例如房间数楼层等，构成一个含有多个变量的模型，模型中的特征为$(x_1, x_1, \ldots, x_n)$。

Size (feet2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

增添更多特征后，我们引入一系列新的注释：

n 代表特征的数量

$x^{(i)}$代表第 i 个训练实例，是特征矩阵中的第i行，是一个**向量**（**vector**）。

比方说，上图的

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix},$$

$x_j^{(i)}$代表特征矩阵中第 i 行的第 j 个特征，也就是第 i 个训练实例的第 j 个特征。

如上图的$x_2^{(2)} = 3, x_3^{(2)} = 2$，

支持多变量的假设 h 表示为：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$，

这个公式中有$n+1$个参数和n个变量，为了使得公式能够简化一些，引入$x_0 = 1$，则公式转化为：$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

此时模型中的参数是一个$n+1$维的向量，任何一个训练实例也都是$n+1$维的向量，特征矩阵X的维度是 $m * (n+1)$。 因此公式可以简化为：$h_\theta(x) = \theta^T X$，其中上标T代表矩阵转置。

4.2 多变量梯度下降

参考视频: 4 - 2 - Gradient Descent for Multiple Variables (5 min).mkv

与单变量线性回归类似，在多变量线性回归中，我们也构建一个代价函数，则这个代价函数是所有建模误差的平方和，即：$J(\theta_0,\theta_1\dots\theta_n)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2$ ，

其中：$h_\theta(x)=\theta^TX=\theta_0+\theta_1x_1+\theta_2x_2+\dots+\theta_nx_n$ ，

我们的目标和单变量线性回归问题中一样，是要找出使得代价函数最小的一系列参数。多变量线性回归的批量梯度下降算法为：

Repeat {

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1,\dots,\theta_n)$$

}

即：

Repeat {

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

}

求导数后得到：

Repeat {

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})\cdot x_j^{(i)})$$

(simultaneously update θ_j for j=0,1,...,n)

}

当$n>=1$时，

$$\theta_0:=\theta_0-a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\,x_0^{(i)}$$

$$\theta_1:=\theta_1-a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\,x_1^{(i)}$$

$$\theta_2:=\theta_2-a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\,x_2^{(i)}$$

我们开始随机选择一系列的参数值，计算所有的预测结果后，再给所有的参数一个新的

值，如此循环直到收敛。

代码示例：

计算代价函数 $J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2$ 其中：$h_\theta(x)=\theta^T X=\theta_0 x_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n$

Python 代码：

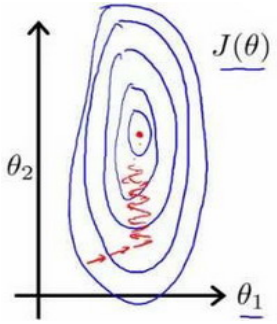
```
def computeCost(X, y, theta):
    inner = np.power(((X * theta.T) - y), 2)
    return np.sum(inner) / (2 * len(X))
```

4.3 梯度下降法实践 1-特征缩放

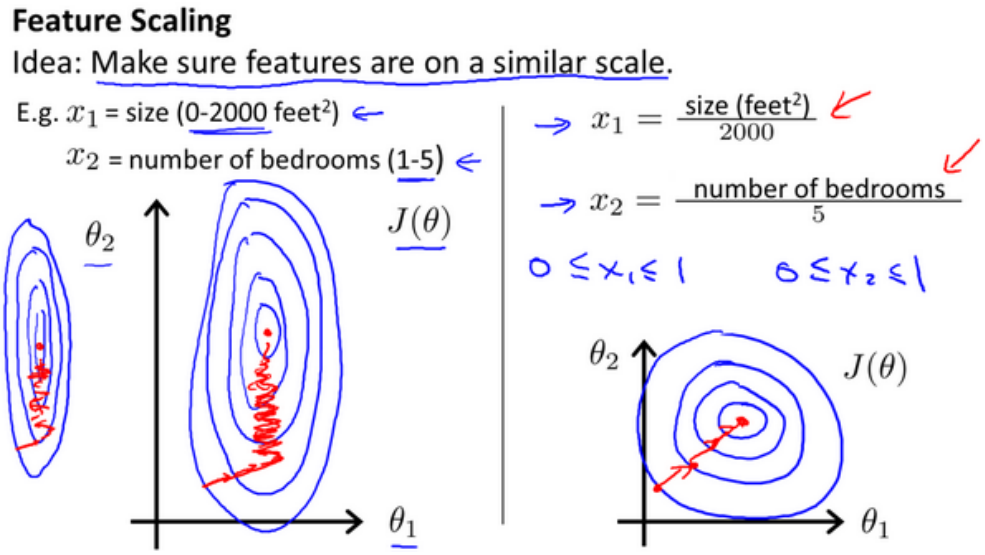
参考视频: 4 - 3 - Gradient Descent in Practice I - Feature Scaling (9 min).mkv

在我们面对多维特征问题的时候，我们要保证这些特征都具有相近的尺度，这将帮助梯度下降算法更快地收敛。

以房价问题为例，假设我们使用两个特征，房屋的尺寸和房间的数量，尺寸的值为 0-2000 平方英尺，而房间数量的值则是 0-5，以两个参数分别为横纵坐标，绘制代价函数的等高线图能，看出图像会显得很扁，梯度下降算法需要非常多次的迭代才能收敛。



解决的方法是尝试将所有特征的尺度都尽量缩放到-1 到 1 之间。如图：

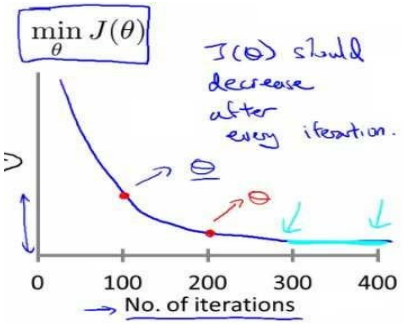


最简单的方法是令：$x_n = \frac{x_n - \mu_n}{s_n}$，其中 μ_n是平均值，s_n是标准差。

4.4 梯度下降法实践 2-学习率

参考视频: 4 - 4 - Gradient Descent in Practice II - Learning Rate (9 min).mkv

梯度下降算法收敛所需要的迭代次数根据模型的不同而不同，我们不能提前预知，我们可以绘制迭代次数和代价函数的图表来观测算法在何时趋于收敛。



也有一些自动测试是否收敛的方法，例如将代价函数的变化值与某个阀值（例如 0.001）进行比较，但通常看上面这样的图表更好。

梯度下降算法的每次迭代受到学习率的影响，如果学习率α过小，则达到收敛所需的迭代次数会非常高；如果学习率α过大，每次迭代可能不会减小代价函数，可能会越过局部最小值导致无法收敛。

通常可以考虑尝试些学习率：

$\alpha = 0.01$，0.03，0.1，0.3，1，3，10

4.5 特征和多项式回归

参考视频: 4 - 5 - Features and Polynomial Regression (8 min).mkv

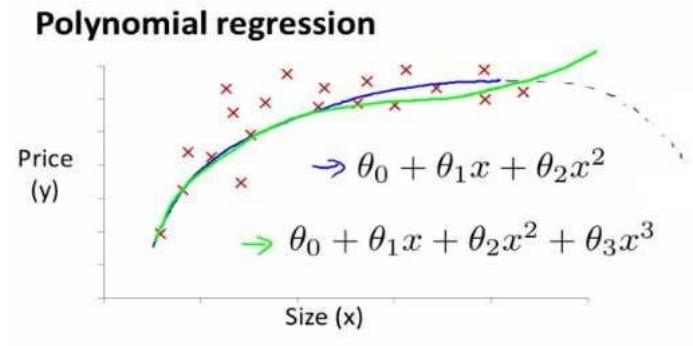
如房价预测问题，

$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$

$x_1 = frontage$（临街宽度），$x_2 = depth$（纵向深度），$x = frontage * depth = area$（面积），则：$h_\theta(x) = \theta_0 + \theta_1 x$。

线性回归并不适用于所有数据，有时我们需要曲线来适应我们的数据，比如一个二次方模型：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$

或者三次方模型： $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$



通常我们需要先观察数据然后再决定准备尝试怎样的模型。 另外，我们可以令：

$x_2 = x_2^2, x_3 = x_3^3$，从而将模型转化为线性回归模型。

根据函数图形特性，我们还可以使：

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_1(size)^2$$

或者:

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_1\sqrt{size}$$

注：如果我们采用多项式回归模型，在运行梯度下降算法前，特征缩放非常有必要。

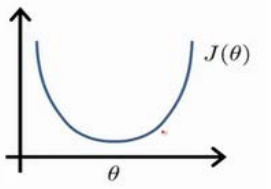
4.6 正规方程

参考视频: 4 - 6 - Normal Equation (16 min).mkv

到目前为止，我们都在使用梯度下降算法，但是对于某些线性回归问题，正规方程方法是更好的解决方案。如：

Intuition: If 1D ($\theta \in \mathbb{R}$)

$J(\theta) = a\theta^2 + b\theta + c$



正规方程是通过求解下面的方程来找出使得代价函数最小的参数的：$\frac{\partial}{\partial \theta_j} J(\theta_j) = 0$ 。假设我们的训练集特征矩阵为 X（包含了 $x_0 = 1$）并且我们的训练集结果为向量 y，则利用正规方程解出向量 $\theta = (X^T X)^{-1} X^T y$ 。

上标 **T** 代表矩阵转置，上标-1 代表矩阵的逆。设矩阵$A = X^T X$，则：$(X^T X)^{-1} = A^{-1}$

以下表示数据为例：

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

即：

X(0)	X(1)	X(2)	X(3)	X(4)	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

运用正规方程方法求解参数：

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2104 & 1416 & 1534 & 852 \\ 5 & 3 & 3 & 2 \\ 1 & 2 & 2 & 1 \\ 45 & 40 & 30 & 36 \end{bmatrix} X \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \right)^{-1} X \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2104 & 1416 & 1534 & 852 \\ 5 & 3 & 3 & 2 \\ 1 & 2 & 2 & 1 \\ 45 & 40 & 30 & 36 \end{bmatrix} X \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

在 **Octave** 中，正规方程写作：

```
pinv(X'*X)*X'*y
```

注：对于那些不可逆的矩阵（通常是因为特征之间不独立，如同时包含英尺为单位的尺寸和米为单位的尺寸两个特征，也有可能是特征数量大于训练集的数量），正规方程方法是不能用的。

梯度下降与正规方程的比较：

梯度下降	正规方程
需要选择学习率α	不需要
需要多次迭代	一次运算得出
当特征数量n大时也能较好适用	需要计算$(X^TX)^{-1}$ 如果特征数量n较大则运算代价大，因为矩阵逆的计算时间复杂度为$O(n^3)$，通常来说当n小于 10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型

总结一下，只要特征变量的数目并不大，标准方程是一个很好的计算参数θ的替代方法。具体地说，只要特征变量数量小于一万，我通常使用标准方程法，而不使用梯度下降法。

随着我们要讲的学习算法越来越复杂，例如，当我们讲到分类算法，像逻辑回归算法，我们会看到，实际上对于那些算法，并不能使用标准方程法。对于那些更复杂的学习算法，我们将不得不仍然使用梯度下降法。因此，梯度下降法是一个非常有用的算法，可以用在有大量特征变量的线性回归问题。或者我们以后在课程中，会讲到的一些其他的算法，因为标准方程法不适合或者不能用在它们上。但对于这个特定的线性回归模型，标准方程法是一个比梯度下降法更快的替代算法。所以，根据具体的问题，以及你的特征变量的数量，这两种算法都是值得学习的。

正规方程的 **python** 实现：

```python
import numpy as np
def normalEqn(X, y):
   theta = np.linalg.inv(X.T@X)@X.T@y #X.T@X 等价于X.T.dot(X)
   return theta
```

4.7 正规方程及不可逆性（选修）

参考视频: 4 - 7 - Normal Equation Noninvertibility (Optional) (6 min).mkv

在这段视频中谈谈正规方程（**normal equation**），以及它们的不可逆性。 由于这是一种较为深入的概念，并且总有人问我有关这方面的问题，因此，我想在这里来讨论它，由于概念较为深入，所以对这段可选材料大家放轻松吧，也许你可能会深入地探索下去，并且会觉得理解以后会非常有用。但即使你没有理解正规方程和线性回归的关系，也没有关系。

我们要讲的问题如下：$\theta = (X^T X)^{-1} X^T y$

备注：本节最后我把推导过程写下。

有些同学曾经问过我，当计算 θ=inv(X'X) X'y ，那对于矩阵$X'X$的结果是不可逆的情况咋办呢?

如果你懂一点线性代数的知识，你或许会知道，有些矩阵可逆，而有些矩阵不可逆。我们称那些不可逆矩阵为奇异或退化矩阵。

问题的重点在于$X'X$的不可逆的问题很少发生，在 **Octave** 里，如果你用它来实现θ的计算，你将会得到一个正常的解。在 **Octave** 里，有两个函数可以求解矩阵的逆，一个被称为 pinv()，另一个是 inv()，这两者之间的差异是些许计算过程上的，一个是所谓的伪逆，另一个被称为逆。使用 pinv() 函数可以展现数学上的过程，这将计算出θ的值，即便矩阵$X'X$是不可逆的。

在 pinv() 和 inv() 之间，又有哪些具体区别呢 ?

其中 inv() 引入了先进的数值计算的概念。例如，在预测住房价格时，如果x_1是以英尺为尺寸规格计算的房子，x_2是以平方米为尺寸规格计算的房子，同时，你也知道 1 米等于 3.28 英尺（四舍五入到两位小数），这样，你的这两个特征值将始终满足约束：$x_1 = x_2 * (3.28)^2$。

实际上，你可以用这样的一个线性方程，来展示那两个相关联的特征值，矩阵$X'X$将是不可逆的。

第二个原因是，在你想用大量的特征值，尝试实践你的学习算法的时候，可能会导致矩阵$X'X$的结果是不可逆的。 具体地说，在m小于或等于 n 的时候，例如，有m等于 10 个的训练样本也有n等于 100 的特征数量。要找到适合的$(n+1)$ 维参数矢量θ，这将会变成一个 101 维的矢量，尝试从 10 个训练样本中找到满足 101 个参数的值，这工作可能会让你花上

一阵子时间，但这并不总是一个好主意。因为，正如我们所看到你只有 10 个样本，以适应这 100 或 101 个参数，数据还是有些少。

稍后我们将看到，如何使用小数据样本以得到这 100 或 101 个参数，通常，我们会使用一种叫做正则化的线性代数方法，通过删除某些特征或者是使用某些技术，来解决当m比n小的时候的问题。即使你有一个相对较小的训练集，也可使用很多的特征来找到很多合适的参数。 总之当你发现的矩阵$X'X$的结果是奇异矩阵，或者找到的其它矩阵是不可逆的，我会建议你这么做。

首先，看特征值里是否有一些多余的特征，像这些x_1和x_2是线性相关的，互为线性函数。同时，当有一些多余的特征时，可以删除这两个重复特征里的其中一个，无须两个特征同时保留，将解决不可逆性的问题。因此，首先应该通过观察所有特征检查是否有多余的特征，如果有多余的就删除掉，直到他们不再是多余的为止，如果特征数量实在太多，我会删除些用较少的特征来反映尽可能多内容，否则我会考虑使用正规化方法。 如果矩阵$X'X$是不可逆的，（通常来说，不会出现这种情况），如果在 **Octave** 里，可以用伪逆函数 `pinv()` 来实现。这种使用不同的线性代数库的方法被称为伪逆。即使$X'X$的结果是不可逆的，但算法执行的流程是正确的。总之，出现不可逆矩阵的情况极少发生，所以在大多数实现线性回归中，出现不可逆的问题不应该过多的关注X^TX是不可逆的。

增加内容：

$\theta = (X^TX)^{-1}X^Ty$ 的推导过程：

$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ 其中：$h_\theta(x) = \theta^TX = \theta_0x_0 + \theta_1x_1 + \theta_2x_2 + \ldots + \theta_nx_n$

将向量表达形式转为矩阵表达形式，则有$J(\theta) = \frac{1}{2}(X\theta - y)^2$ ，其中X为m行n列的矩阵（m为样本个数，n为特征个数），θ为n行 1 列的矩阵，y为m行 1 列的矩阵，对$J(\theta)$进行如下变换:

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$= \frac{1}{2}(\theta^TX^T - y^T)(X\theta - y)$$

$$= \frac{1}{2}(\theta^TX^TX\theta - \theta^TX^Ty - y^TX\theta - y^Ty)$$

接下来对$J(\theta)$偏导，需要用到以下几个矩阵的求导法则:

$\frac{dAB}{dB} = A^T$

$\frac{dX^TAX}{dX} = 2AX$

所以有:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{2}(2X^TX\theta - X^Ty - (y^TX)^T - 0)$$

$$= \frac{1}{2}(2X^TX\theta - X^Ty - X^Ty - 0)$$

$$= X^TX\theta - X^Ty$$

令$\frac{\partial J(\theta)}{\partial \theta} = 0,$

则有$\theta = (X^TX)^{-1}X^Ty$