Machine Learning - Week 2

赵燕

目录

1 Multivariate Linear Regression

1.1 Multiple Features

根据之前的例子，我们对房价模型增加更多的特征，例如房间数楼层等，构成了一个含有多个变量的模型，模型中的特征值为（x_1,x_2,...,x_n）：

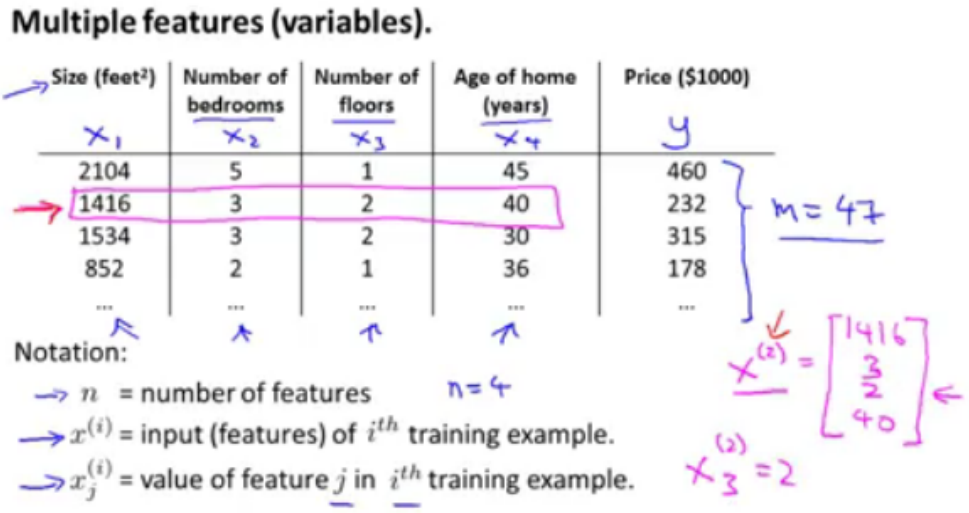

图 1: 多变量房价模型

注释：

（1）n：特征的数量

（2）$x^{(i)}$：第i个训练样本，是特征矩阵中的第i行，是一个向量（vector）； 例如：上图中的

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \tag{1}$$

（3）$x_j^{(i)}$：特征矩阵中第i行的第j个特征值，也就是第i个训练样本实例的第j个特征，如上图的$x_3^{(2)}$=2。

多变量的假设函数：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{2}$$

这个公式中有你n+1个参数和n个变量，为了能使公式简化，引入x_0=1，则公式转化为：

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{3}$$

此时模型中的参数是一个n+1维的变量，任何一个训练实例都是n+1维的向量，特征矩阵X的维度是m*(n+1)。 公式可以简化为：

$$h_\theta(x) = \begin{bmatrix}\theta_0 & \theta_1 & . & . & . & \theta_n\end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \theta^T X \tag{4}$$

$$h_\theta(x) = \theta^T X \tag{5}$$

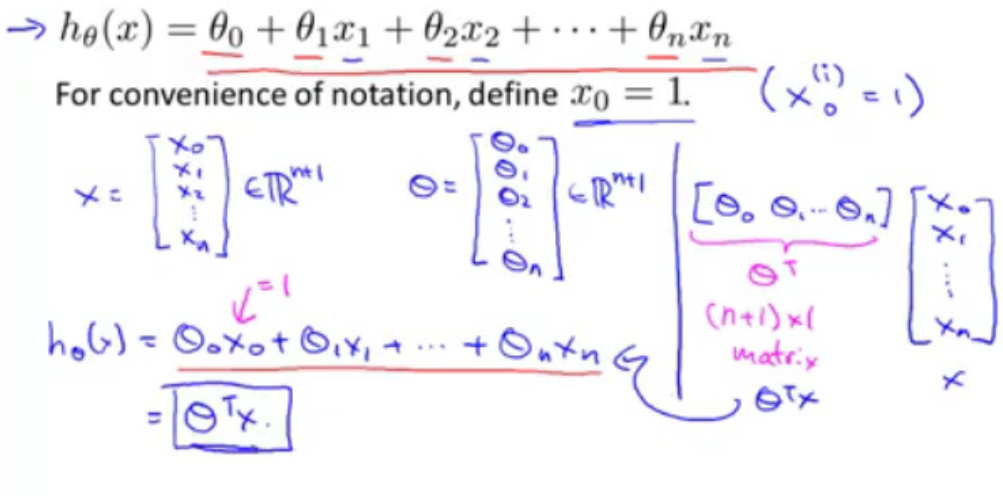

图 2: 多变量假设函数

1.2 Gradient Descent for Multiple Variables

与单变量线性回归类似，在多变量线性回归中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：

$$J(\theta_1, \theta_2, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{6}$$

其中：

$$h_\theta(x) = \theta^T X = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{7}$$

多变量梯度算法定义：n=1和n¿=1的情况其实是一样的

repeat until convergence:{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \tag{8}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \tag{9}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \tag{10}$$

.....

}

In others words:

repeat until convergence:{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{11}$$

$$for j := 0, 1, 2, ..., n \tag{12}$$

}

单变量与多变量的比较： The following image compares gradient descent with one variable to gradient descent with multiple variables:

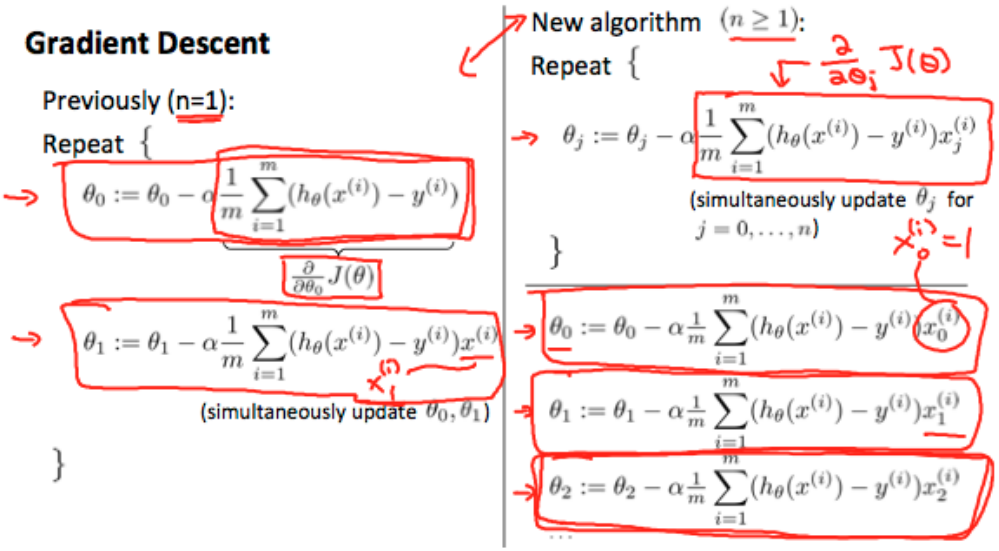

图 3: 单变量与多变量的梯度下降算法对比

1.3 Gradient Descent in Practice I-Feature Scaling

特征缩放可以将梯度下降的速度加快，将梯度下降收敛所需的循环次数减少。

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，如图所示，代价函数的等高线图很扁，又细又长，梯度下降算法需要非常多次的迭代才能收敛。

解决的额方法是尝试将所有特征的尺度都尽量缩放到-1到1之间。

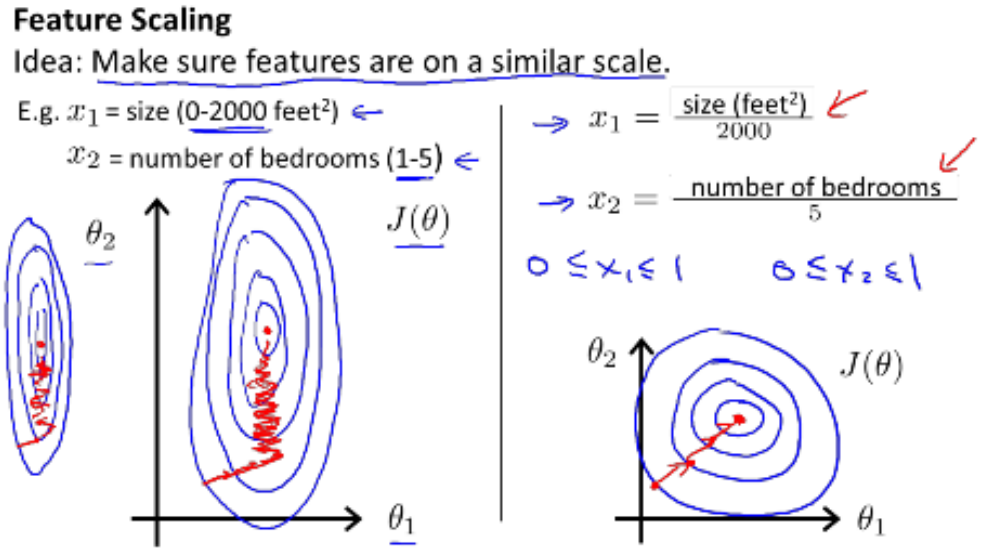

图 4: 多变量假设函数

Two techniques to help with this are feature scaling and mean normalization. Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. Mean normalization involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:

$$x_i = \frac{x_i - \mu_i}{s_i} \tag{13}$$

μ_i :is the average of all the values for feature (i);

s_i :is the range of values (max - min), or si is the standard deviation(标准差)

Note that dividing by the range, or dividing by the standard deviation, give different results. The quizzes in this course use range - the programming exercises use standard deviation.

For example: if xi represents housing prices with a range of 100 to 2000 and a mean value of 1000, then:

$$x_i = \frac{price - 1000}{900} \tag{14}$$

1.4 Gradient Descent in Practice II-Learning Rate

梯度下降算法所需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个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阈值（例如0.001）进行比较，但是通常看如下的图表更好。

Debugging gradient descent: Make a plot with number of iterations on the x-axis. Now plot the cost function, J(θ) over the number of iterations of gradient descent. If J(θ) ever increases, then you probably need to decrease α.

Automatic convergence test：Declare convergence if J(θ) decreases by less than E in one iteration, where E is some small value such as 103. However in practice it's difficult to choose this threshold value.

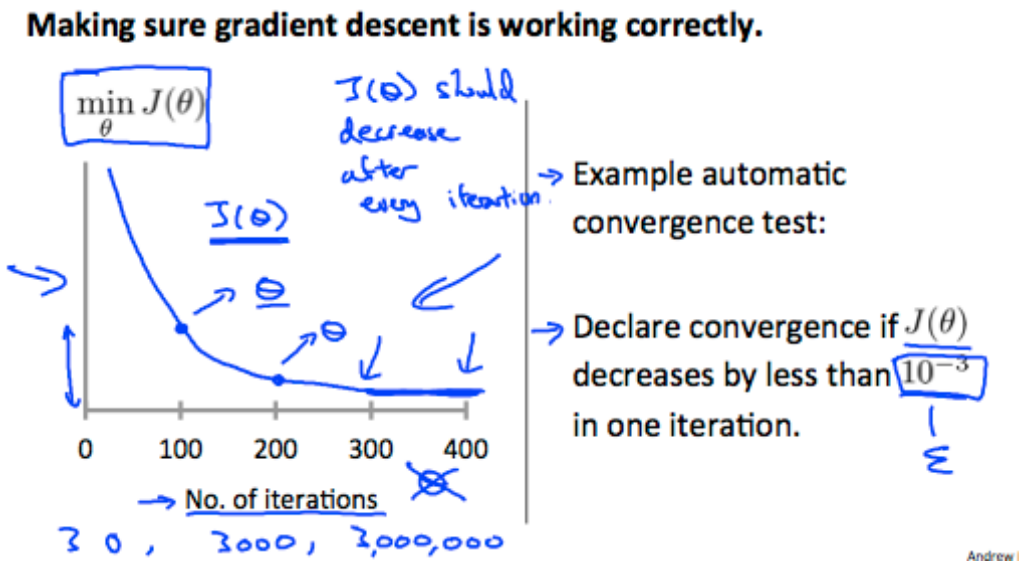

图 5: 迭代次数与代价函数

梯度下降算法的每次迭代受到学习率α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。

It has been proven that if learning rate α is sufficiently small, then J(θ) will decrease on every iteration.

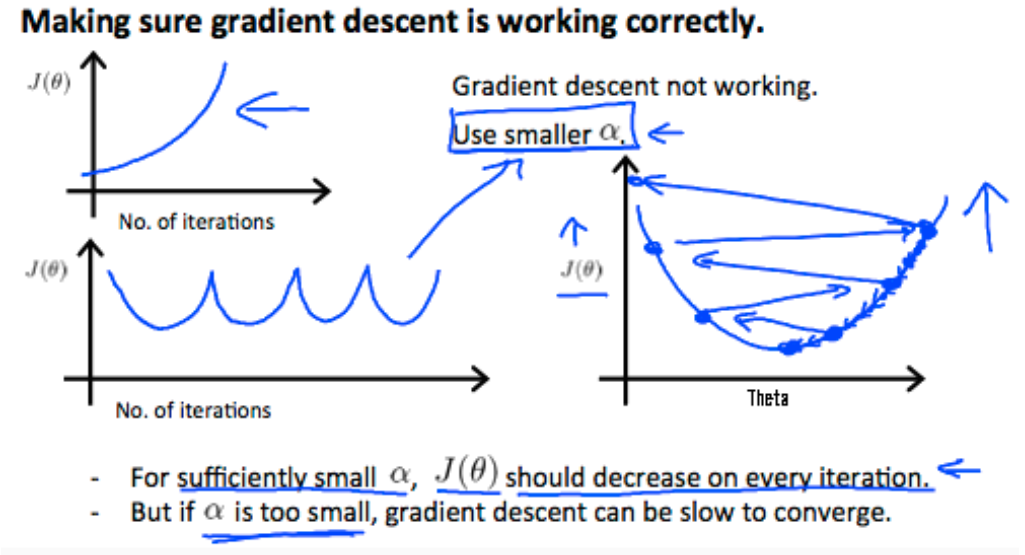

图 6: α的影响

通常可以考虑尝试这些学习率α:

α =0.01，0.03，0.1，0.3，1，3，10

To summarize:

If α is too small: slow convergence.

If α is too large: may not decrease on every iteration and thus may not converge.

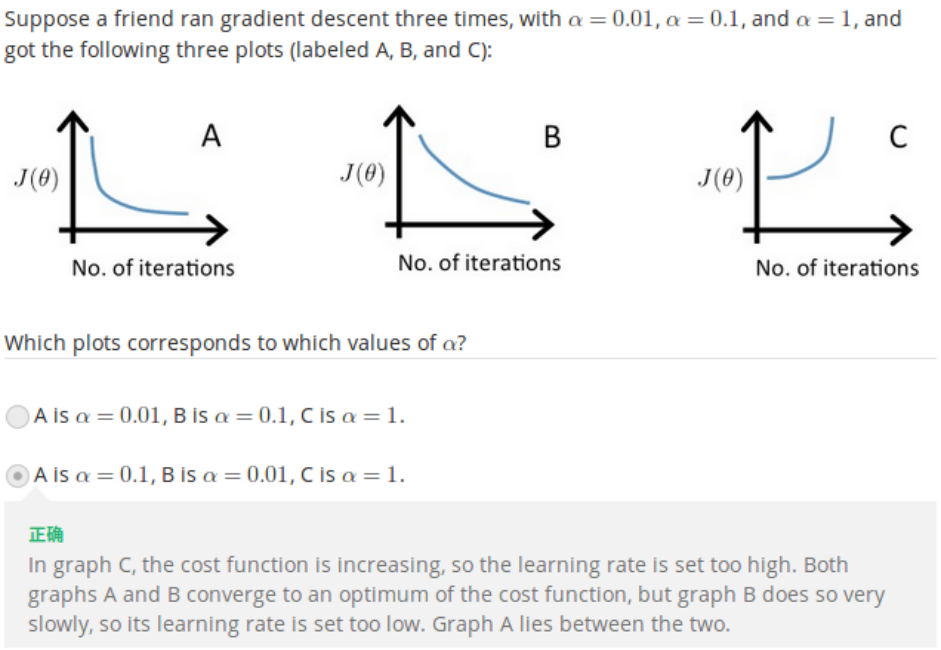

图 7: Learning Rate

1.5 Features and Polynomial Regression

举例：房价预测问题

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth \tag{15}$$

注释：

x_1:frontage(临街宽度)

x_2:depth(纵向深度)

x:area=frontage*depth(面积) 则：

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{16}$$

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：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 \tag{17}$$

或者三次方模型：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 \tag{18}$$

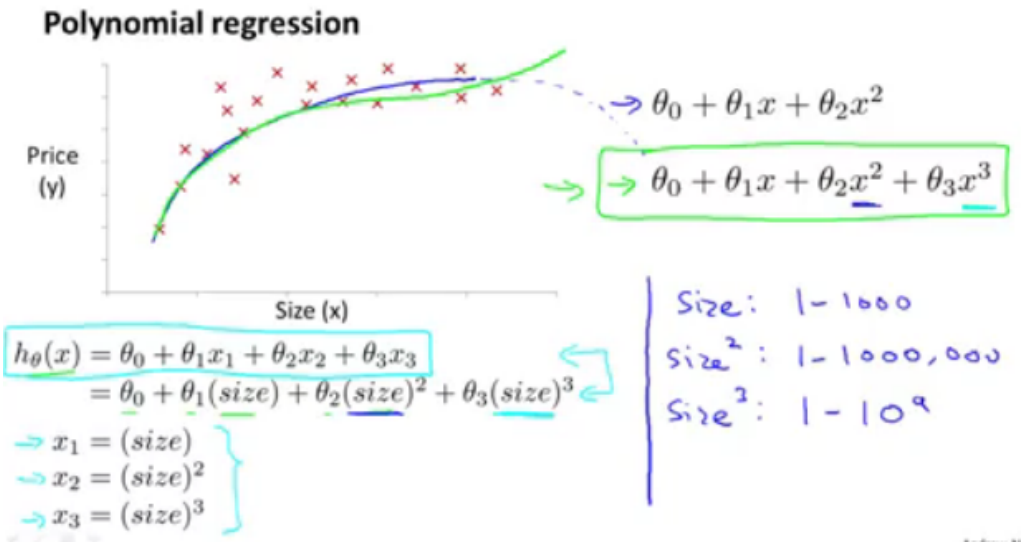

图 8: 房价预测：多项式回归

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。另外我们可以另：

$$x_2 = x_2^2 \tag{19}$$

$$x_3 = x_3^3 \tag{20}$$

从而将模型转化为线性回归模型。

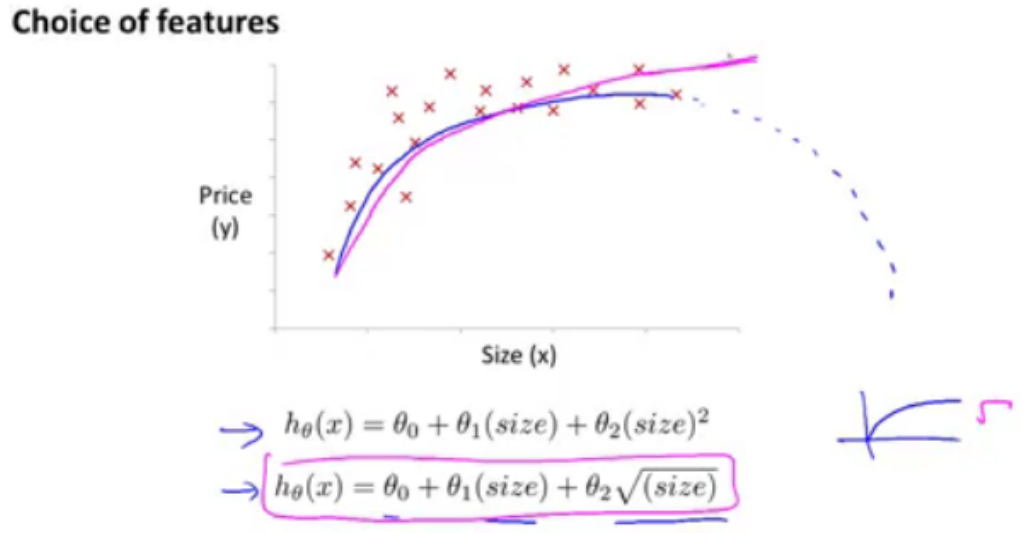

图 9: Choice of features

根据函数的图形特征，我们可以使：

$$h_\theta(x) = \theta_0 + \theta_1 size + \theta_2 (size)^2 \tag{21}$$

或者：

$$h_\theta(x) = \theta_0 + \theta_1 size + \theta_2 \sqrt{(size)} \tag{22}$$

Polynomial Regression(多项式回归):

Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can change the behavior or curve of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

For example, if our hypothesis function is

$$h_\theta(x) = \theta_0 + \theta_1 x_1 \tag{23}$$

then we can create additional features based on x1, to get the quadratic function

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \tag{24}$$

or the cubic function

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 \tag{25}$$

In the cubic version, we have created new features x_2 and x_3 where $x_2 = x_1^2$ and $x_3 = x_1^3$.

To make it a square root function, we could do: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

One important thing to keep in mind is, if you choose your features this way then feature scaling(特征缩放) becomes very important.

注意：如果我们采用多项式回归，在运行梯度下降算法前，特征缩放非常有必要。

eg. if x1 has range 1 - 1000 then range of x_1^2 becomes 1 - 1000000 and that of x_1^3 becomes 1 - 1000000000

多项式回归特征缩放练习题:

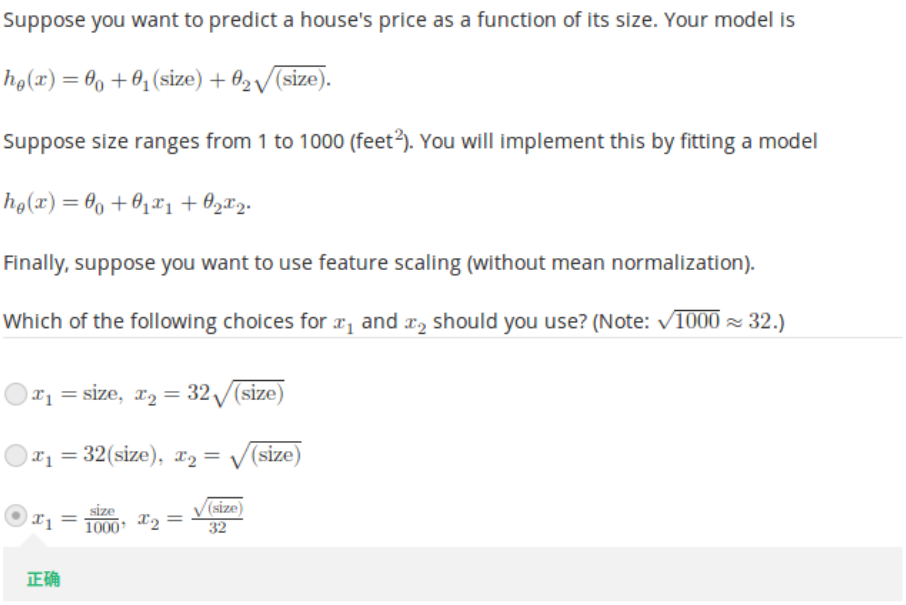

图 10: 特征缩放例题

2 Computer Parameters Analytically

2.1 Normal Equation

正规方程：针对某些问题，正规方程有更好的解决方案。

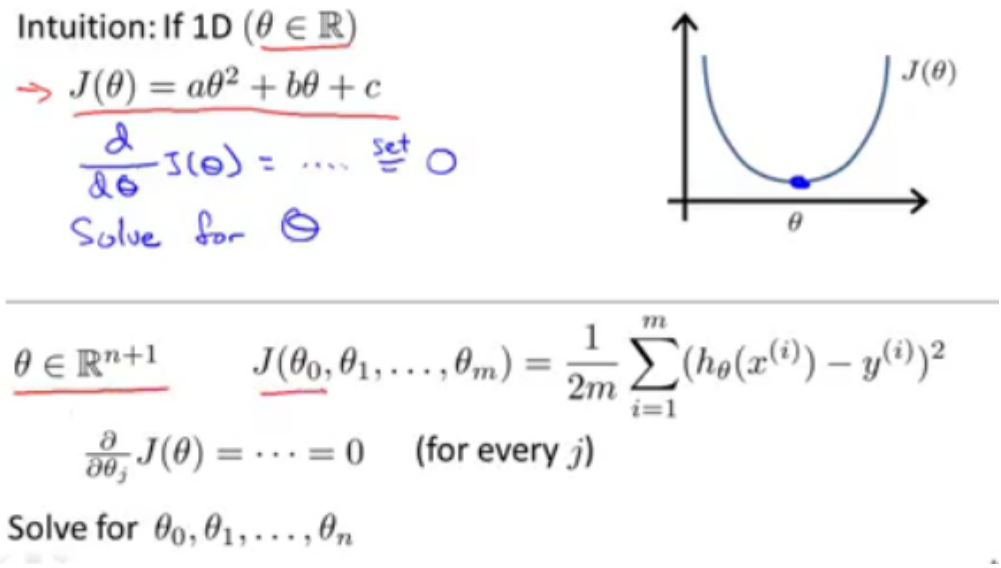

图 11: Intuition

正规方程通过求解下面的方程来找出使得代价函数最小的参数的θ:

$$\frac{\partial}{\partial \theta_j} J(\theta_j) = 0 \tag{26}$$

假使我们的训练集特征为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

$$\theta = (X_T X)^{-1} X^T y \tag{27}$$

注释： 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：$(X_T X)^{-1} = A^{-1}$

以下数据为例：

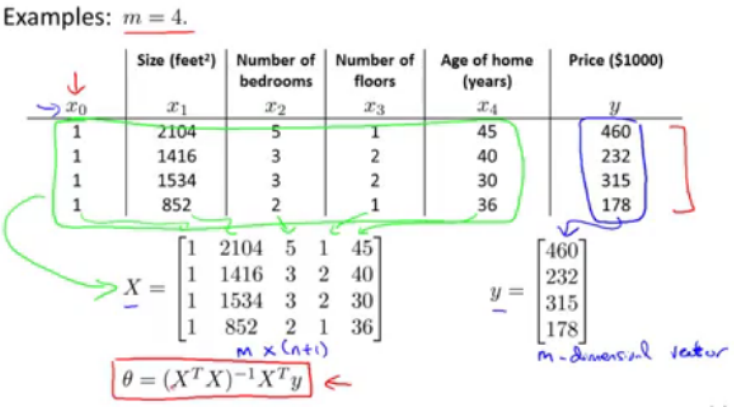

图 12: 房价问题：正规方程

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

图 13: 数据表格

运用正规方程求解参数:

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

图 14: 运用正规方程求解参数

在Octave中，正规方程

$$\theta = (X_T X)^{-1} X^T y \tag{28}$$

写作:

$$pinv(X' * X) * X' * y \tag{29}$$

注意：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

图 15: 梯度下降与正规方程的比较

总结: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。

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of X^TX
Works well when n is large	Slow if n is very large

图 16: Gradient Descent or Normal Equation

With the normal equation, computing the inversion has complexity
$bigcirc(n^3)$. So if we have a very large number of features, the normal equation will be slow. In practice, when n exceeds 10,000 it might be a good time to go from a normal solution to an iterative process.

2.2 Normal Equation Noninvertibility

When implementing the normal equation in octave we want to use the 'pinv' function rather than 'inv.' The 'pinv' function will give you a value of θ even if X^TX is not invertible.

If X^TX is noninvertible, the common causes might be having :

(1)Redundant features, where two features are very closely related (i.e. they are linearly dependent)

(2)Too many features (e.g. m $\leqslant$ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

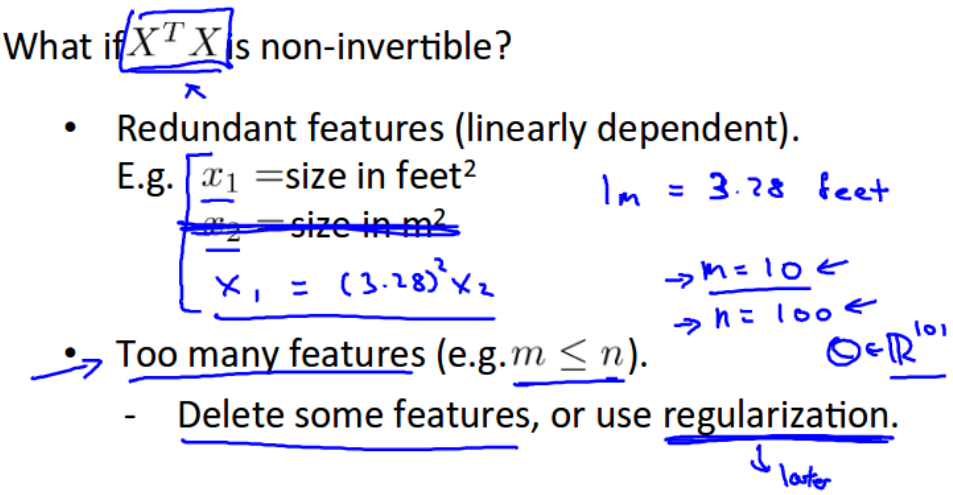

图 17: Non-invertible

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.