四、多变量线性回归

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

4.1 多维特征

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$(x_1, x_2, \ldots, x_n)$。

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：$h_\theta(x)=\theta_0+\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n$，
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：
$h_\theta(x)=\theta_0x_0+\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n$
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：$h_\theta(x)=\theta^TX$，其中上标T代表矩阵转置。

4.2 多变量梯度下降

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：$J(\theta_0,\theta_1\ldots\theta_n)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2$ ，
其中：$h_\theta(x)=\theta^TX=\theta_0+\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n$ ，

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：

$$\text{Repeat } \{$$

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

$$\}$$

即：

$$\text{Repeat } \{$$

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

$$\}$$

求导数后得到：

$$\text{Repeat } \{$$

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

$$(\text{ simultaneously update } \theta_j$$

$$\text{for } j=0,1,\ldots,n \ )$$

$$\}$$

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：$h_\theta(x) = \theta^T X = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

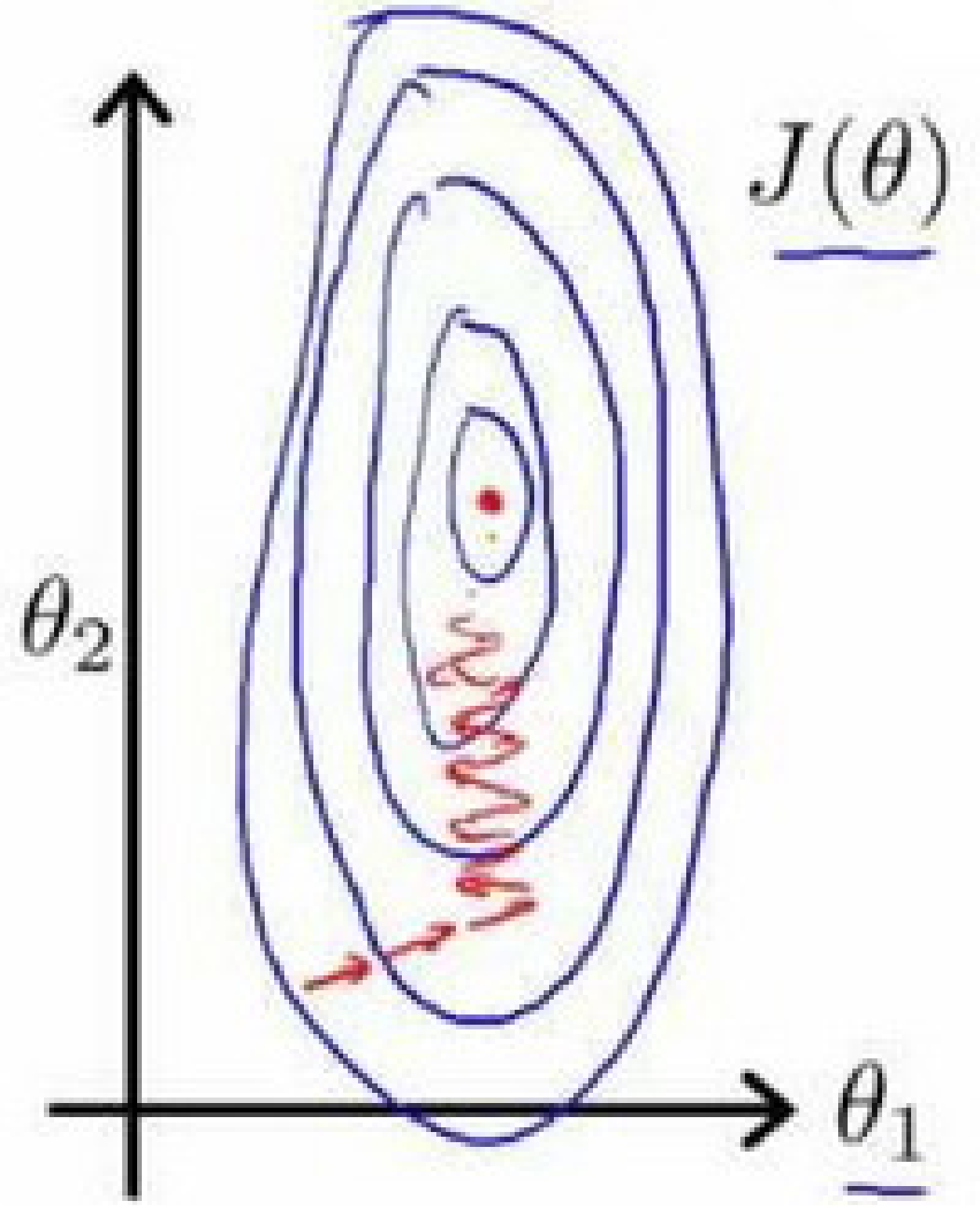
Python 代码：

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

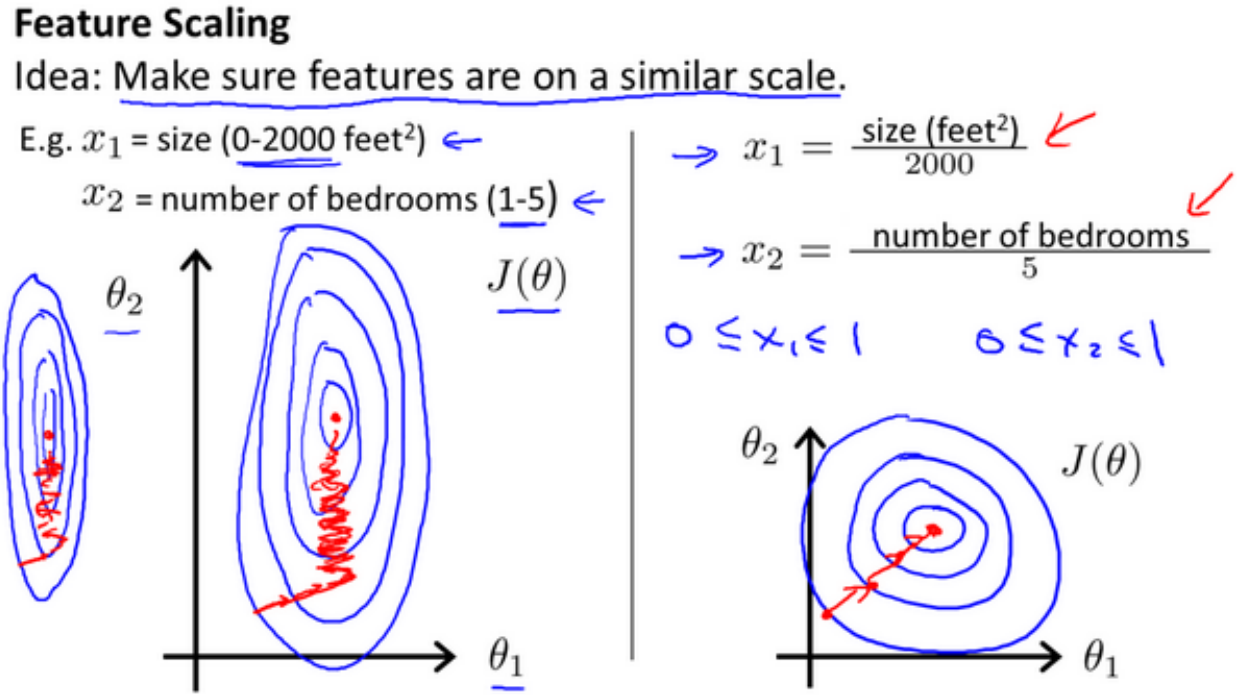
4.3 梯度下降法实践1-特征缩放

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。

问题

为什么等高线的圆心（或椭圆中心）是函数的最小值?
考虑一个二次函数：$f(x, y) = ax^2 + by^2$,其中 $(a > 0, b > 0)$。

（1）它的等高线是怎么来的?

令函数值为常数 (c)：$ax^2 + by^2 = c$
这正是一个**椭圆方程**（当 $(a \neq b)$）或**圆方程**（当 $(a = b)$)。

（2）函数最小值在哪里?

要找最小值，就看梯度：

$$\nabla f(x, y) = \left[\frac{\partial f}{\partial x} \ \frac{\partial f}{\partial y}\right] = [2ax \ 2by]$$

令梯度为 0（极值点条件）：$2ax = 0, \quad 2by = 0$
解得： $x = 0, \quad y = 0$
因此，函数在 ((0,0)) 处取得极值。
再看二阶导： $\frac{\partial^2 f}{\partial x^2} = 2a > 0, \quad \frac{\partial^2 f}{\partial y^2} = 2b > 0$
说明它是**最小值**。

结论

等高线都是围绕点 ((0,0)) 的椭圆或圆。
这个点就是函数值最小的地方，也就是**最小值点（minimum）**。
因此，**圆心（或椭圆中心）= 最小值点**。

为什么等高线图是椭圆形的情况下，优化算法（尤其是梯度下降）会难收敛。

1. 背景：等高线图与收敛性

假设我们要最小化一个二次函数： $f(x, y) = ax^2 + by^2$,其中 (a, b > 0)。
它的等高线图是： $ax^2 + by^2 = c$
如果 $(a = b)$，等高线就是**圆形**；
如果 $(a \neq b)$，等高线就是**椭圆形**。

2. 为什么椭圆形会难收敛

情况一：圆形等高线（a = b）

- 梯度方向总是**直接指向最小值（圆心）**。
- 每一步都朝着正确的方向前进，路径“笔直”。
- 收敛非常快。

情况二：椭圆形等高线（a ≠ b）

- 梯度方向**不是**指向椭圆中心（最优点）的直线方向。
- 梯度下降每一步都会“**偏离最优路径**”，出现“**之字形震荡**”。
- 因此，需要非常小的学习率才能保持稳定，不然会在椭圆两侧来回反弹。

3. 从数学上看（条件数）

对于二次函数 $(f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x})$，
其中 (A) 是正定矩阵。

- 等高线的形状由矩阵 (A) 的**特征值**决定。
- 椭圆的长短轴比 = 特征值之比。
- 定义条件数： $\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$
  若$(\kappa(A))$很大（椭圆非常“扁”），则问题**病态（ill-conditioned）**。

在这种情况下：

- 梯度方向主要受最大特征值方向影响；
- 不同方向的曲率差异很大；
- 导致：
  - 步长难以选择；
  - 收敛速度受最小特征值限制；
  - 整体收敛变慢。

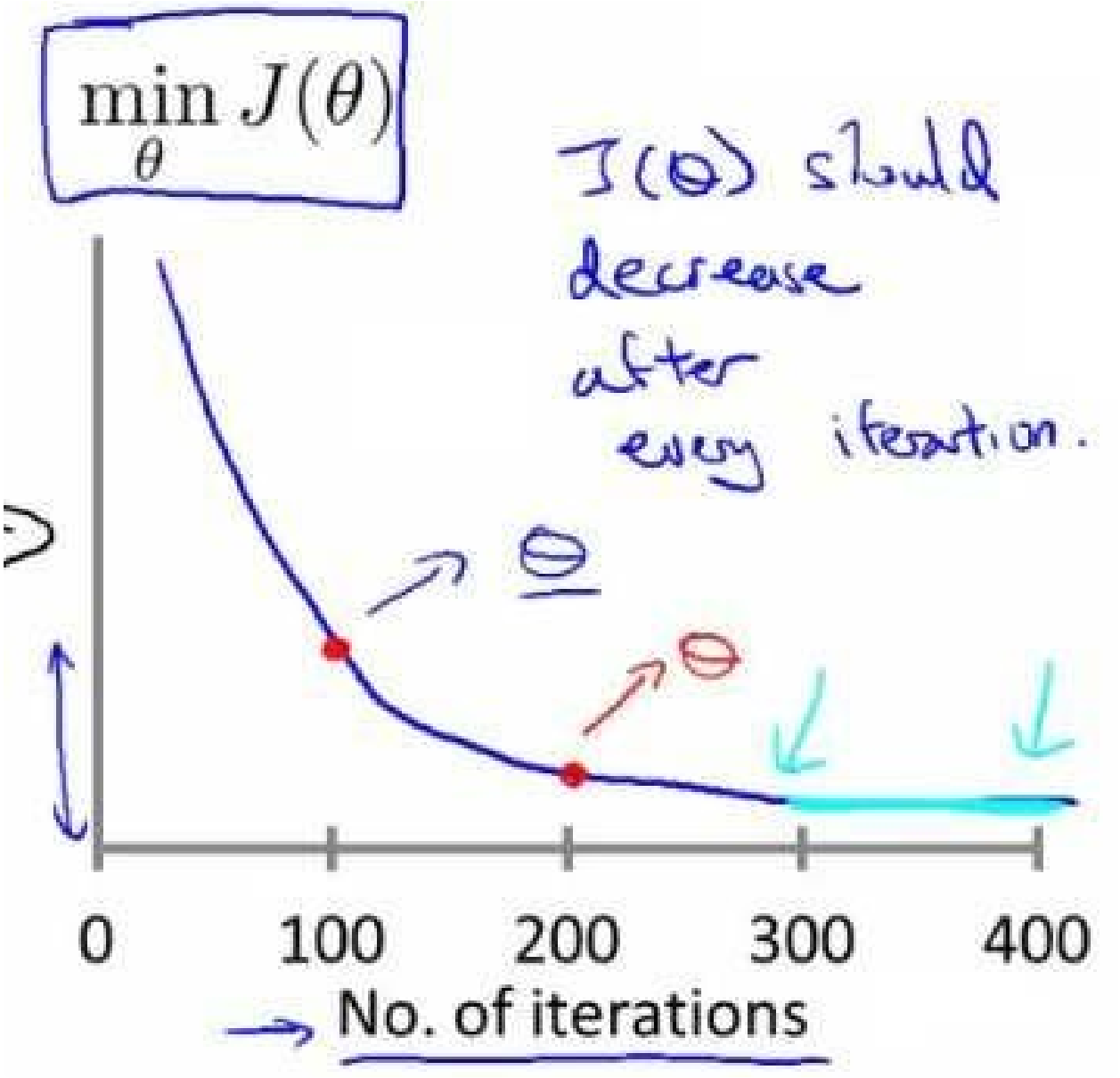
4. 解决方法

要让“椭圆”变得更“圆”，我们可以：

1. **特征缩放（Feature Scaling）**
   - 对输入变量做标准化或归一化；
   - 让所有方向的梯度变化幅度相似。
2. **预条件（Preconditioning）**
   - 使用 Hessian 近似来调整更新方向；
   - 比如 Newton 法、共轭梯度法。
3. **自适应学习率方法**
   - 如 Adam、RMSProp；
   - 自动调节不同方向的学习率。

4.4 梯度下降法实践2-学习率

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。



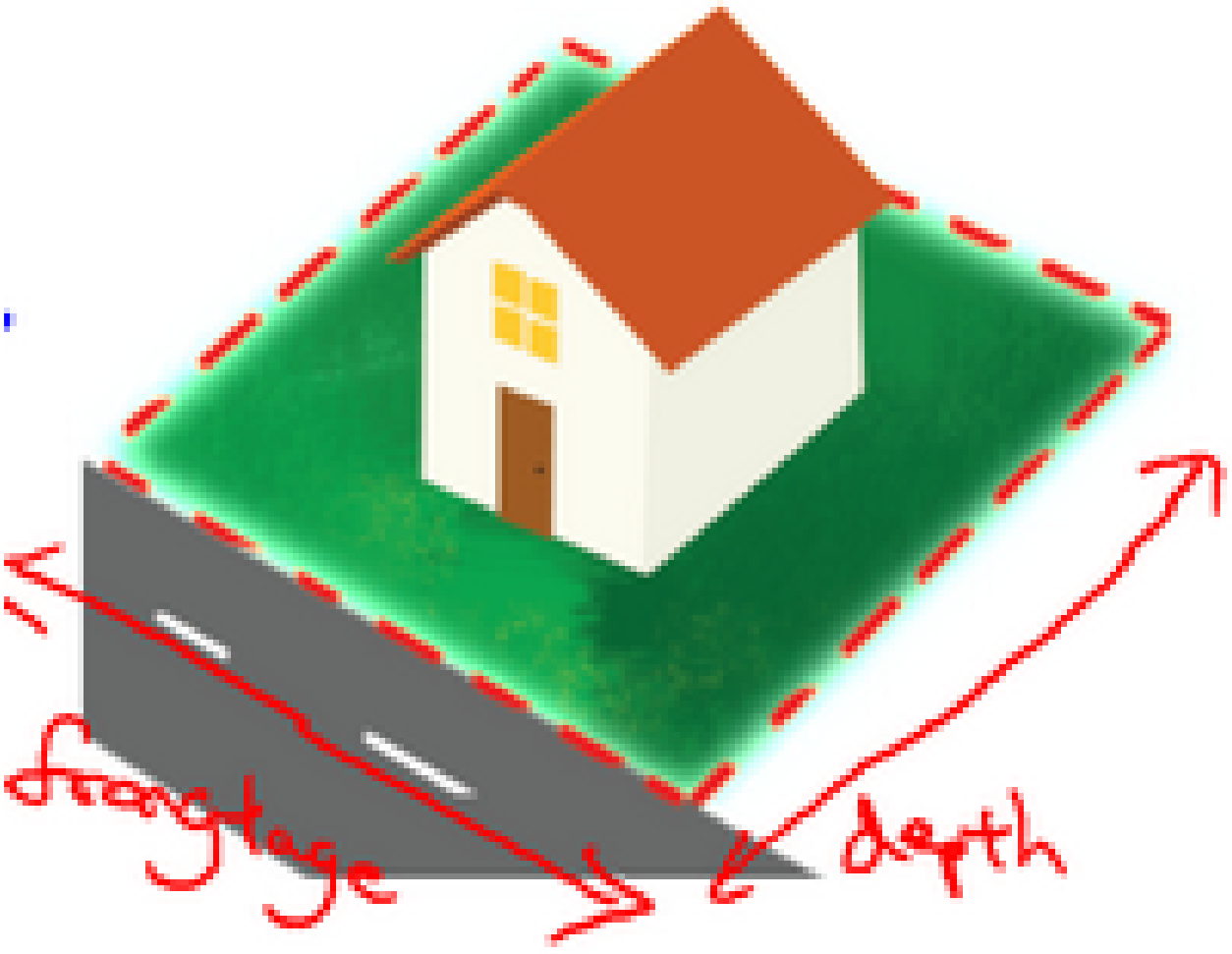
也有一些自动测试是否收敛的方法，例如将代价函数的变化值与某个阀值（例如0.001）进行比较，但通常看上面这样的图表更好。

梯度下降算法的每次迭代受到学习率的影响:

- 学习率a过小，则达到收敛所需的迭代次数会非常高；
- 如果学习率a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

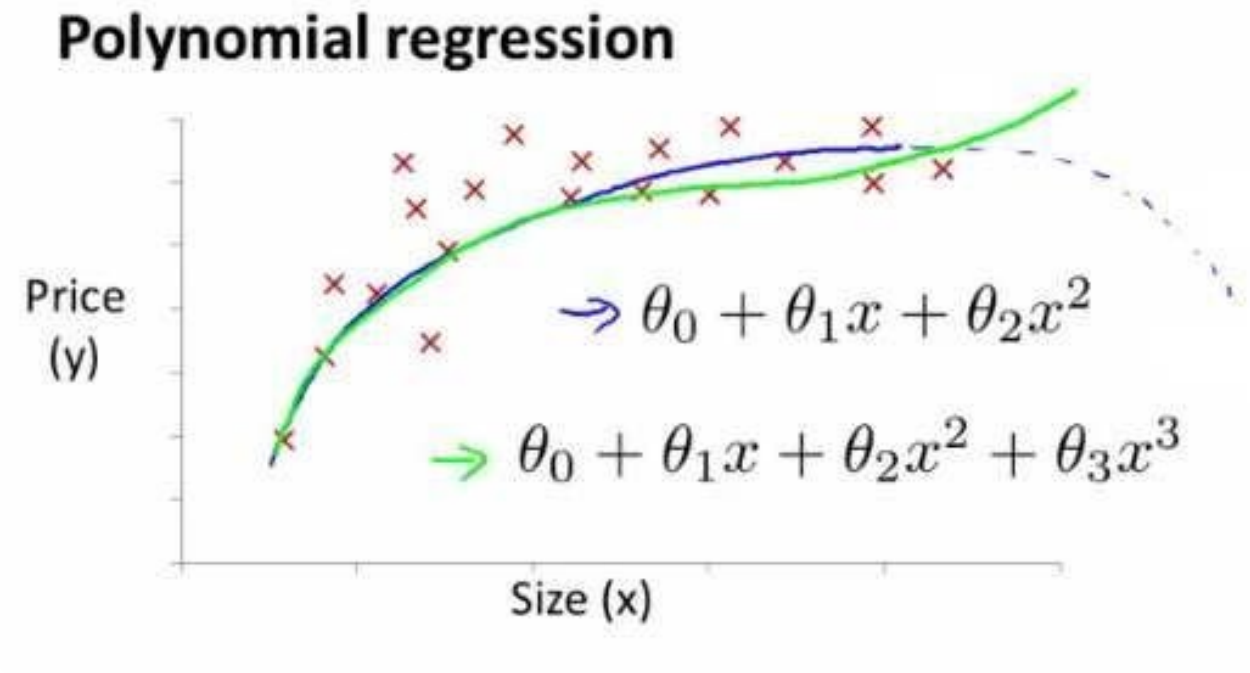
如房价预测问题，



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

$x_1 = frontage$（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：
$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$
或者:
$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{size}$
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

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：

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$\left(X^{T} X\right)^{-1}$ 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$O\left(n^{3}\right)$，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。

推导过程

$\theta=\left(X^{T} X\right)^{-1} X^{T} y$ 的推导过程：

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

其中：$h_{\theta}\left(x\right)=\theta^{T}X=\theta_{0}x_{0}+\theta_{1}x_{1}+\theta_{2}x_{2}+\ldots+\theta_{n}x_{n}$

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

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

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

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

接下来对$J(\theta)$偏导，需要用到以下几个矩阵的求导法则:

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

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

所以有:

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

$=\frac{1}{2}\left(2X^{T}X\theta-X^{T}y-X^{T}y-0\right)$

$=X^{T}X\theta-X^{T}y$

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

则有$\theta=\left(X^{T}X\right)^{-1}X^{T}y$

4.7 正规方程及不可逆性

1. 正规方程的基本形式

在线性回归中，我们通过**正规方程**（Normal Equation）求参数向量： $\theta = (X^T X)^{-1} X^T y$
其中：

- (X)：样本特征矩阵（含截距项）
- (y)：目标值向量
- (θ)：待求参数向量

2. 不可逆矩阵的问题

在计算过程中，可能出现 ($X^T X$) **不可逆（singular 或 degenerate）** 的情况。
不可逆矩阵即无法求其标准逆矩阵。

3. 为什么 $(X^T X)$ 会不可逆?

常见原因有两个：

（1）特征线性相关

当某些特征是**线性相关**的，例如：

- (x_1) 表示房屋面积（平方英尺）
- (x_2) 表示房屋面积（平方米）
  且 $(x_1 = x_2 \times (3.28)^2)$
  那么这两个特征之间存在确定的线性关系，导致 $(X^T X)$ 不可逆。

（2）特征数量过多（($m \leq n$)）

当训练样本数 (m) 小于或等于特征数量 (n) 时，例如：

- (m = 10)（样本数）
- (n = 100)（特征数）
  此时要从 10 个样本中拟合 101 个参数（含截距项），信息不足，矩阵也会退化为不可逆。

4. 解决不可逆问题的方法

方法 1：删除冗余特征

- 检查特征间是否**线性相关或重复**
- 删除冗余或重复的特征，直到剩下的特征相互独立

方法 2：使用伪逆（pseudo-inverse）

- 在 **Octave / MATLAB** 中：

    - `inv()`：计算矩阵的标准逆
    - `pinv()`：计算矩阵的**伪逆**
- 当(X^TX) 不可逆时，使用 `pinv(X' * X) * X' * y` 仍可得到有效的解
  伪逆的数学原理能在不可逆的情况下给出最小二乘意义下的最优解。

方法 3：使用正则化（Regularization）

- 当特征数多、样本少时，可以通过在代价函数中添加惩罚项（如岭回归）来避免不可逆问题。
- 例如： $\theta = (X^TX + \lambda I)^{-1}X^Ty$
  其中 $(\lambda > 0)$保证矩阵可逆。