线性模型

一、监督学习和无监督学习

区别

- 在无监督学习中，我们已知的数据。看上去有点不一样，不同于监督学习的数据的样
- 区别就是有无标签

- 聚类属于无监督学习

常见的监督学习

- 回归（预测离散的输出值）
- 分类

二、单变量线性回归

房屋交易问题

- 预测住房价格的，我们要使用一个数据集，数据集包含俄勒冈州波特兰市的住房价格。

我们将要用来描述这个回归问题的标记如下:

m 代表训练集中实例的数量

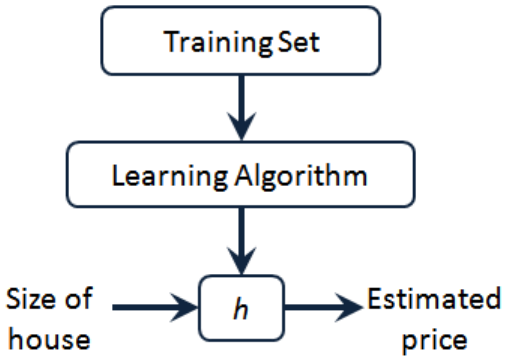
x 代表特征/输入变量

y 代表目标变量/输出变量

(x, y) 代表训练集中的实例

$(x^{(i)}, y^{(i)})$ 代表第i 个观察实例

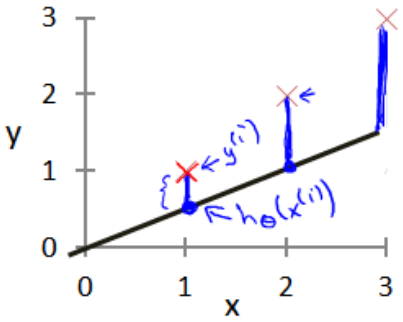
h 代表学习算法的解决方案或函数也称为假设（**hypothesis**）



一种可能的表达方式为：$h_\theta(x) = \theta_0 + \theta_1 x$，因为只含有一个特征/输入变量，因此这样的问题叫作单变量线性回归问题。

2.2 代价函数（Loss）

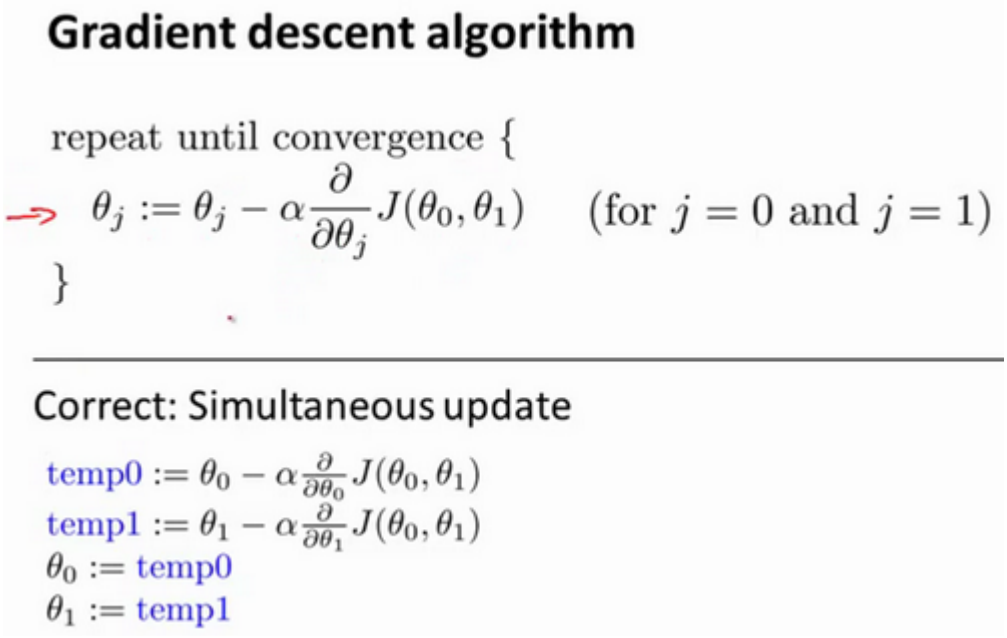
我们选择的参数决定了我们得到的直线相对于我们的训练集的准确程度，模型所预测的值与训练集中实际值之间的差距（下图中蓝线所指）就是**建模误差**（**modeling error**）。



我们的目标便是选择出可以使得建模误差的平方和能够最小的模型参数。 即使得代价函数

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ 最小

2.3 梯度下降



步长α太小或者太大会怎样

- 如果α太小了，即我的学习速率太小，结果就是只能这样像小宝宝一样一点点地挪动，

- 如果α太大，那么梯度下降法可能会越过最低点，甚至可能无法收敛，下一次迭代又移

多种类型的梯度下降

我们刚刚使用的算法，有时也称为批量梯度下降。实际上，在机器学习中，通常不太会纠

三、线性代数

矩阵、向量 相乘啥的

四、多变量的新型回归

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

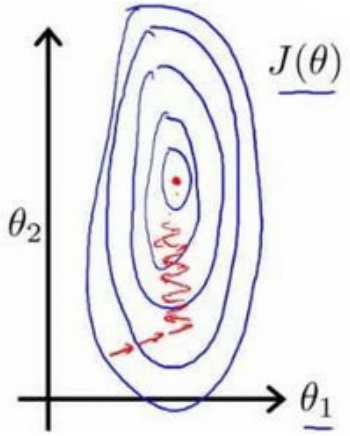
（x1 , x2, … , xn）。

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

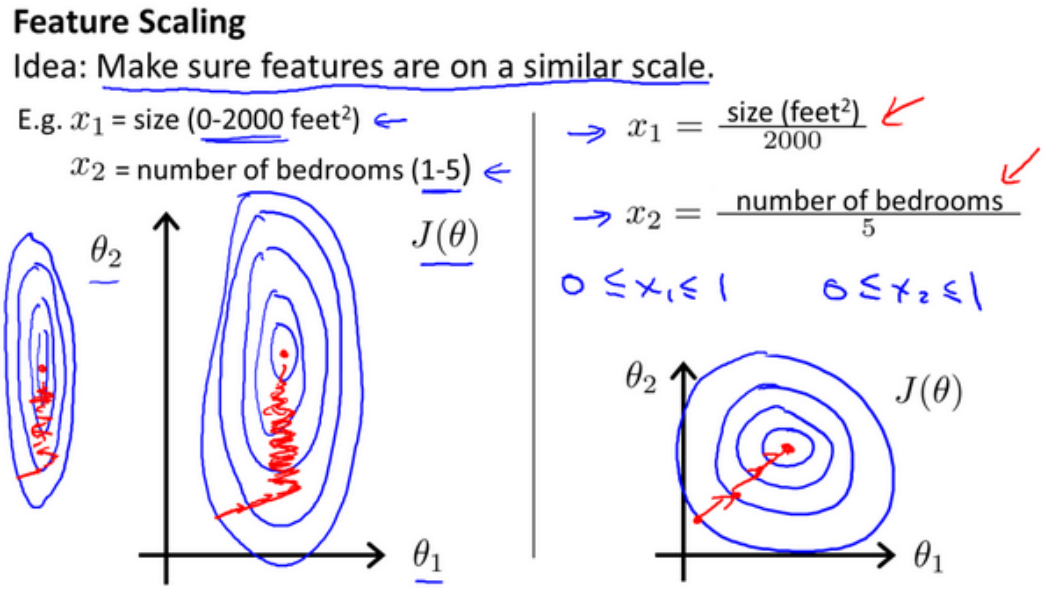
4.1 特征缩放

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：



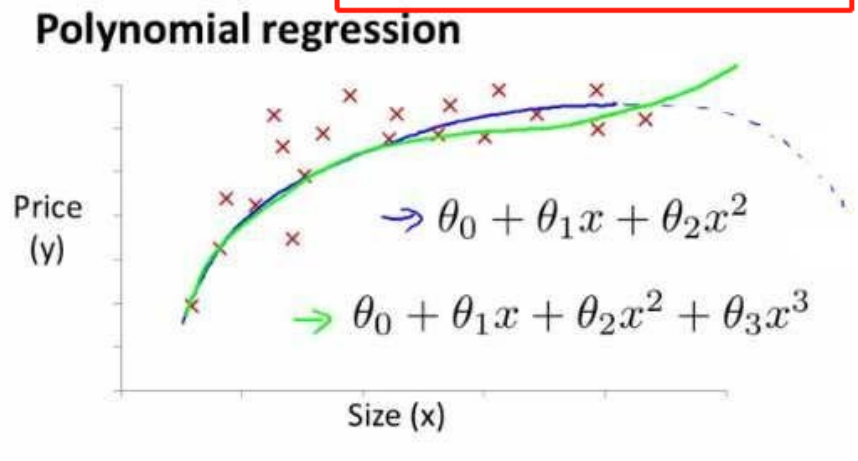
最简单的方法是令：$x_n = \frac{x_n - \mu_n}{s_n}$，其中 μ_n 是平均值，s_n 是标准差。

4.2 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:

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

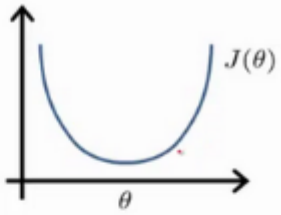
注：如果我们采用多项式回归模型，在运行梯度下降算法前，特征缩放非常有必要。

4.3 正规方程求解最优解

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 ($\theta \in \mathbb{R}$)

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



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：

应用数理统计

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：

梯度下降与正规方程的比较：

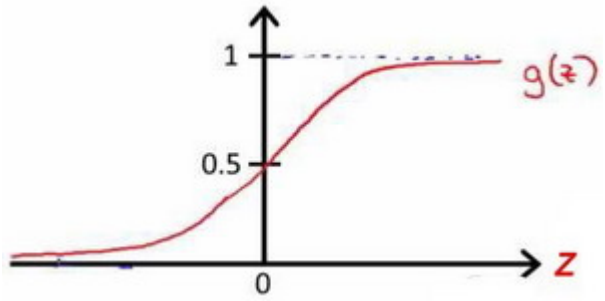
梯度下降	正规方程
需要选择学习率α	不需要
需要多次迭代	一次运算得出
当特征数量n大时也能较好适用	需要计算$(X^T X)^{-1}$ 如果特征数量n较大则运算代价大，因为矩阵逆的计算时间复杂度为$O(n^3)$，通常来说当n小于10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型

五、逻辑回归（对数几率函数Logistic Regression）

如果我们要用线性回归算法来解决一个分类问题，对于分类， y 取值为 0 或者1，但如果你使用的是线性回归，那么假设函数的输出值可能远大于 1，或者远小于0，即使所有训练样本的标签 y 都等于 0 或 1。尽管我们知道标签应该取值0 或者1，但是如果算法得到的值远大于1或者远小于0的话，就会感觉很奇怪。所以我们在接下来的要研究的算法就叫做逻辑回归算法，这个算法的性质是：它的输出值永远在0到 1 之间。

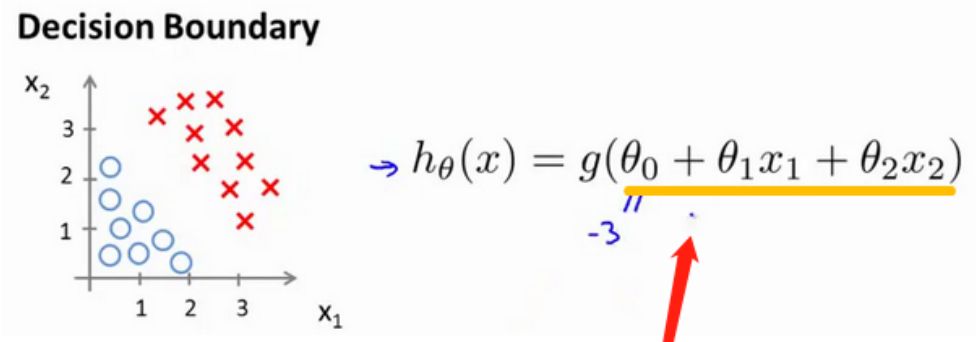
顺便说一下，逻辑回归算法是分类算法，我们将它作为分类算法使用。有时候可能因为这个算法的名字中出现了“回归”使你感到困惑，但逻辑回归算法实际上是一种分类算法，它适用于标签 y 取值离散的情况，如：１００１。

我们引入一个新的模型，逻辑回归，该模型的输出变量范围始终在0和1之间。 逻辑回归模型的假设是：
$h_\theta(x) = g(\theta^T X)$ 其中： X 代表特征向量 g 代表逻辑函数（**logistic function**)是一个常用的逻辑函数为**S**形函数（**Sigmoid function**），公式为：$g(z) = \frac{1}{1+e^{-z}}$。

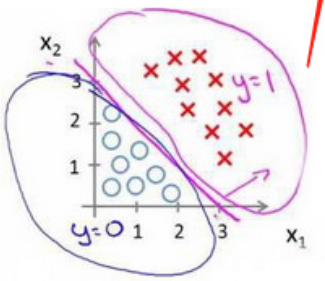


5.2 构建

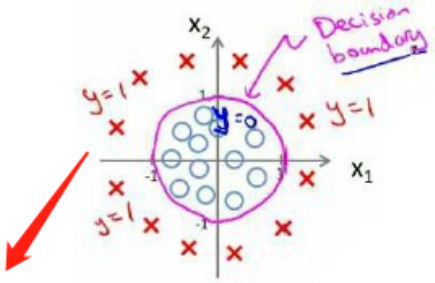
现在假设我们有一个模型：



并且参数θ 是向量[-3 1 1]。 则当$-3 + x_1 + x_2 \geq 0$，即$x_1 + x_2 \geq 3$时，模型将预测 $y = 1$。 我们可以绘制直线$x_1 + x_2 = 3$，这条线便是我们模型的分界线，将预测为1的区域和预测为 0的区域分隔开。



假使我们的数据呈现这样的分布情况，怎样的模型才能适合呢？



因为需要用曲线才能分隔 $y = 0$ 的区域和 $y = 1$ 的区域，我需要二次方特征：
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$是[-1 0 0 1 1]，则我们得到的判定边界恰好是圆点在原点且半径为1的圆形。

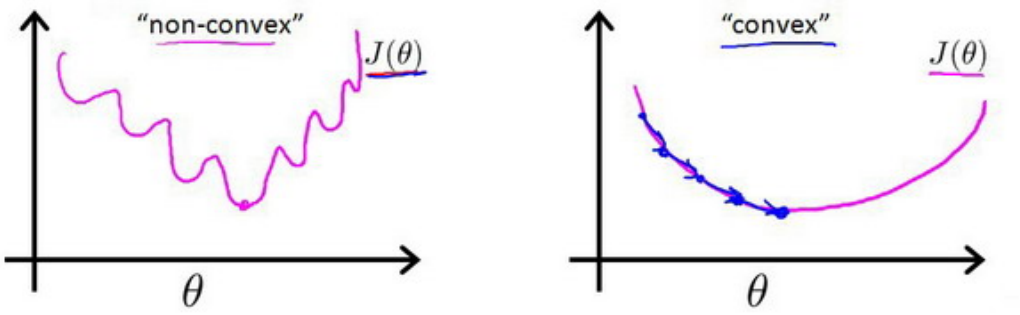
5.3 代价函数（Loss）

- 问题

对于线性回归模型，我们定义的代价函数是所有模型误差的平方和。理论上来说，我们也可以对逻辑回归模型沿用这个定义，但是问题在于，当我们将

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

带入到这样定义了的代价函数中时，我们得到的代价函数将是一个非凸函数（**non-convexfunction**
）。



这意味着我们的代价函数有许多局部最小值，这将影响梯度下降算法寻找全局最小值。

Logistic Regression（逻辑回归）中的损失函数理解

- 理解1（分类讨论）

逻辑回归中sigmoid函数为 $h_\theta(x)=\frac{1}{1+e^{(-\theta^T x)}}$ (其中 $\theta^T x=\sum_{i=0}^{n}\theta_i x_i$)

可以用sigmoid函数表示0-1中取1的概率。所以我们的损失函数可以定义为

$$当y=0时，\ Cost(h_\theta(x),y)=-log(1-h_\theta(x))$$

$$当y=1时，\ Cost(h_\theta(x),y)=-log(h_\theta(x))$$

当我们把损失函数与0-1分布的分布律对应起来的时候，$p=h_\theta(x)$，损失函数就是在0-1分布的基础上取对数然后再取负数。这也好理解，损失函数的要求就是预测结果与真实结果越相近，函数值越小，所以会在前面加上负号。当y=0时，1-p的概率会比较大，在前面加上负号，Cost值就会很小；当y=1时，p的概率会比较大，在前面加上负号，Cost值就会很小。至于取对数，就是跟最大似然函数有关系，取对数不影响原本函数的单调性，而且会放大概率之间的差异，更好的区分各个样本的类别。

把上面损失函数写成统一的形式：

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}logh_\theta(x^{(i)})+(1-y^{(i)})log(1-h_\theta(x^{(i)}))]$$

好了，至此，我们得到了逻辑回归的损失函数。虽然大家都是这么讲的，但是，总是感觉没有太懂为什么最后得到了这个损失函数。如果想从数学的角度推导，可以继续往下看。

- 理解2（似然函数求解）

对于逻辑回归的似然函数

逻辑回归中sigmoid函数为 $h_\theta(x)=\frac{1}{1+e^{(-\theta^T x)}}$，可以用sigmoid函数表示0-1中取1的概率，在这里用于表示逻辑回归中的概率。逻辑回归中的样本值为 $((x^1,y^1),(x^2,y^2)\ldots,(x^m,y^m))$，样本中的 x^i 是用来求概率 $h_\theta(x)$ 的，y^i 是样本的真实值，也就是真实类别。在机器学习中，习惯称 x^i 为特征值，y^i 为标签。

$h_\theta(x)$ 对应于0-1分布中的概率 p，y^i 对应于0-1分布中的 x_i，也就是样本值。这样我们就把逻辑回归和0-1分布对应起来了。我们用逻辑回归来作为分类模型，需要用最大似然估计的方法来评判模型的好坏。让总体分布尽量与样本的分布趋同，就是总体的分布与样本分布具有最大的相似性，然后再来求取模型中的参数 θ，这样就可以得到比较符合最大似然估计的模型。这个模型其实就是 $h_\theta(x)$。

根据0-1分布的似然函数，我们可以写出逻辑回归的似然函数

$$L(p)=\prod_{i=1}^{m}h_\theta(x^{(i)})^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}}$$

对数形式为

$$logL(p)=\sum_{i=1}^{m}y^{(i)}log\,h_\theta(x^{(i)})+\sum_{i=1}^{m}(1-y^{(i)})log(1-h_\theta(x^{(i)}))$$

逻辑回归的损失函数为

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}logh_\theta(x^{(i)})+(1-y^{(i)})log(1-h_\theta(x^{(i)}))]$$

$$J(\theta) = -\frac{1}{m} logL(p)$$

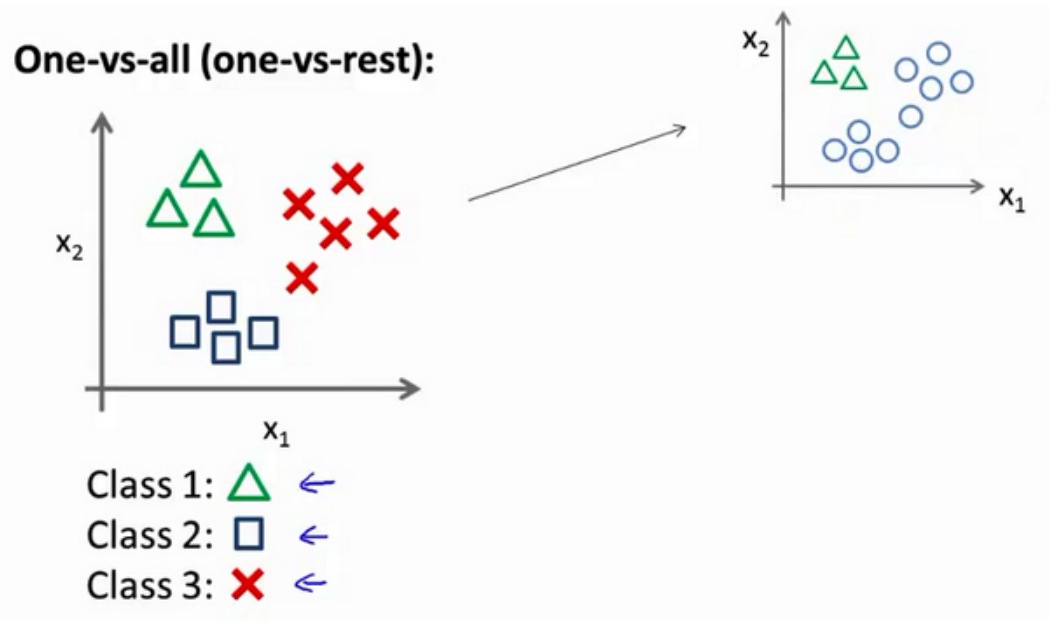
损失函数跟对数形式的似然函数很像，只是在前面乘以 $-\frac{1}{m}$ 。最大似然估计的方法要求 $logL(p)$ 的最大值，损失函数在其前面加上负号，就是求最小值，这个跟损失函数的特性刚好吻合。1/m是用来对m个样本值的损失函数值取平均，不会影响函数功能。
因此，逻辑回归的损失函数求最小值，就是根据最大似然估计的方法来的。

- 代价函数

$$J_{(\theta)} = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} log(h_\theta(x^{(i)})) + (1-y^{(i)}) log(1-h_\theta(x^{(i)}))]$$

- 使用梯度下降法来求得使loss最小的参数
- 优化后的求解逻辑
  - 共轭梯度法 BFGS (变尺度法) 、
  - L-BFGS (限制变尺度法)和
  - 线性搜索(line search)

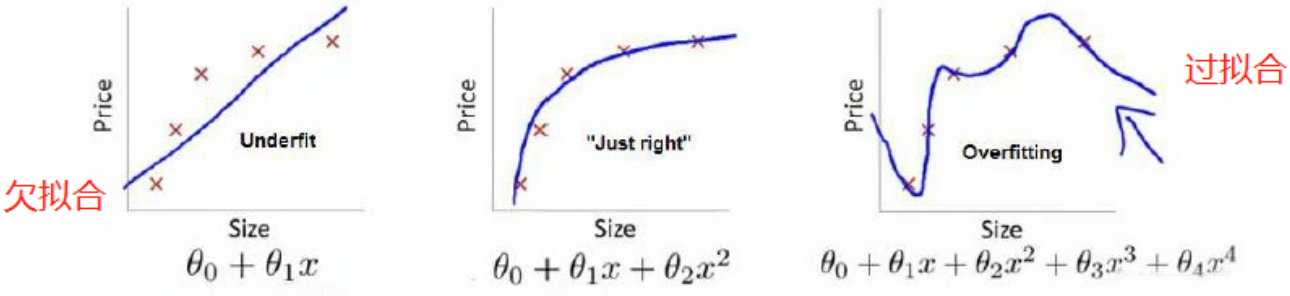
5.4 多分类分类问题（1对多）



六、正则化

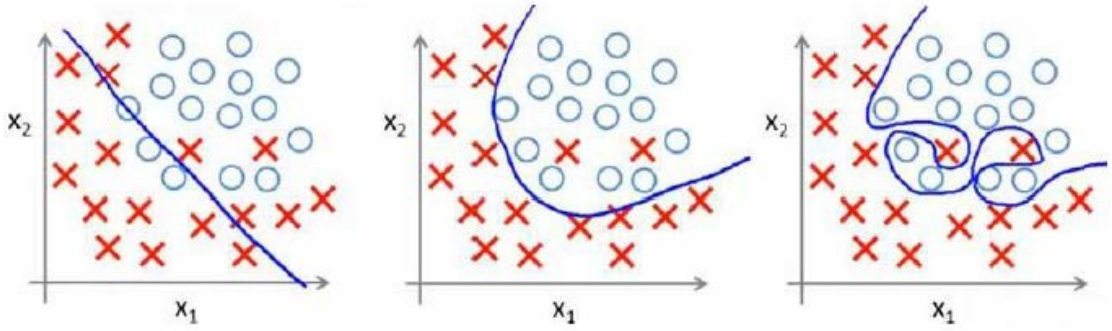
6.1 过拟合问题

下图是一个回归问题的例子：



第一个模型是一个线性模型，欠拟合，不能很好地适应我们的训练集；第三个模型是一个四次方的模型，过于强调拟合原始数据，而丢失了算法的本质：预测新数据。我们可以看出，若给出一个新的值使之预测，它将表现的很差，是过拟合，虽然能非常好地适应我们的训练集但在新输入变量进行预测时可能会效果不好；而中间的模型似乎最合适。

分类问题中也存在这样的问题：



6.2 代价函数

从之前的事例中看出，正是那些高次项导致了过拟合的产生，所以如果我们能让这些高次项的系数接近于0的话，我们就能很好的拟合了。

合了。 所以我们要做的就是在一定程度上减小这些参数θ 的值，这就是正则化的基本方法。我们决定要减少θ_3和θ_4的大小，我们要做的便是修改代价函数，在其中θ_3和θ_4 设置一点惩罚。这样做的话，我们在尝试最小化代价时也需要将这个惩罚纳入考虑中，并最终导致选择较小一些的θ_3和θ_4。 修改后的代价函数如下：

$$\min_\theta \frac{1}{2m}[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2+1000\theta_3^2+10000\theta_4^2]$$

正则项　　高次项

$$J(\theta)=\frac{1}{2m}[\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2+\lambda\sum_{j=1}^{n}\theta_j^2]$$

其中λ又称为正则化参数（**Regularization Parameter**）

6.3 线性回归的正则化

对于线性回归的求解，我们之前推导了两种学习算法：一种基于梯度下降，一种基于正规方程。

正则化线性回归的代价函数为：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}[((h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2)]$$

如果我们要使用梯度下降法令这个代价函数最小化，因为我们未对θ_0进行正则化，所以梯度下降算法将分两种情形：

$Repeat\ until\ convergence\{$

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

$$\theta_j := \theta_j - a[\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j]$$

$for\ j = 1, 2, \ldots n$

$\}$

对上面的算法中$j = 1, 2, \ldots, n$ 时的更新式子进行调整可得：

$\theta_j := \theta_j(1 - a\frac{\lambda}{m}) - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$ 可以看出，正则化线性回归的梯度下降算法的变化在于，每次都在原有算法更新规则的基础上令θ值减少了一个额外的值。

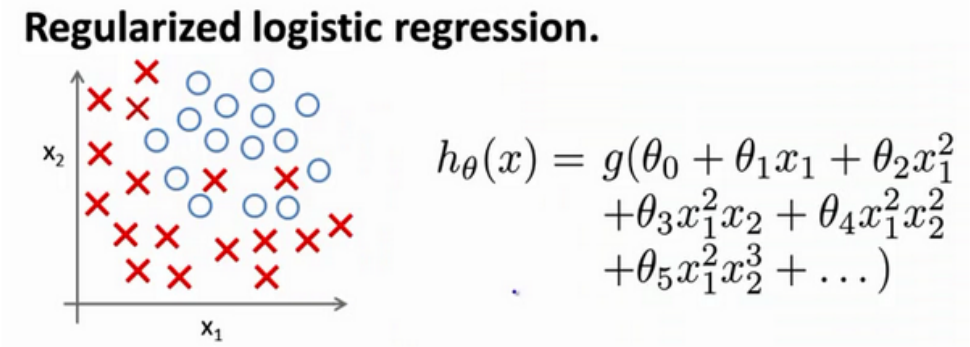
我们同样也可以利用正规方程来求解正则化线性回归模型，方法如下所示：

$$\theta = \left(X^TX + \lambda\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix}\right)^{-1}X^Ty$$

图中的矩阵尺寸为 $(n+1)*(n+1)$。

6.4 逻辑回归的正则化

针对逻辑回归问题，我们在之前的课程已经学习过两种优化算法：我们首先学习了使用梯度下降法来优化代价函数$J(\theta)$，接下来学习了更高级的优化算法，这些高级优化算法需要你自己设计代价函数$J(\theta)$。



自己计算导数同样对于逻辑回归，我们也给代价函数增加一个正则化的表达式，得到代价函数：

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

要最小化该代价函数，通过求导，得出梯度下降算法为：

$Repeat\ until\ convergence\{$

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

$$\theta_j := \theta_j - a[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j]$$

$for\ j = 1, 2, \dots n$

$\}$

注：看上去同线性回归一样，但是知道$h_\theta(x) = g\left(\theta^T X\right)$，所以与线性回归不同。 **Octave** 中，我们依旧可以用`fminuc`函数来求解代价函数最小化的参数，值得注意的是参数θ_0的更新规则与其他情况不同。 注意：

1. 虽然正则化的逻辑回归中的梯度下降和正则化的线性回归中的表达式看起来一样，但由于两者的$h_\theta(x)$不同所以还是有很大差别。
2. θ_0不参与其中的任何一个正则化。

目前大家对机器学习算法可能还只是略懂，但是一旦你精通了线性回归、高级优化算法和正则化技术，坦率地说，你对机器学习的理解可能已经比许多工程师深入了。现在，你已经有了丰富的机器学习知识，目测比那些硅谷工程师还厉害，或者用机器学习算法来做产品。

接下来的课程中，我们将学习一个非常强大的非线性分类器，无论是线性回归问题，还是逻辑回归问题，都可以构造多项式来解决。你将逐渐发现还有更强大的非线性分类器，可以用来解决多项式回归问题。我们接下来将将学会，比现在解决问题的方法强大N倍的学习算法。