第三讲：回归学习

回归学习

- 回归属于有监督学习中的一种方法。该方法的核心思想是从连续型统计数据中得到数学模型，然后将该数学模型用于预测或者分类。该方法处理的数据可以是多维的。
- 回归是由达尔文的表兄弟Francis Galton发明的。Galton于1877年完成了第一次回归预测，目的是根据上一代豌豆的种子（双亲）的尺寸来预测下一代豌豆种子（孩子）的尺寸（身高）。Galton在大量对象上应用了回归分析，甚至包括人的身高。他得到的结论是：如果双亲的高度比平均高度高，他们的子女也倾向于平均身高但尚不及双亲，这里就可以表述为：孩子的身高向着平均身高回归。Galton在多项研究上都注意到了这一点，并将此研究方法称为回归。

回归学习

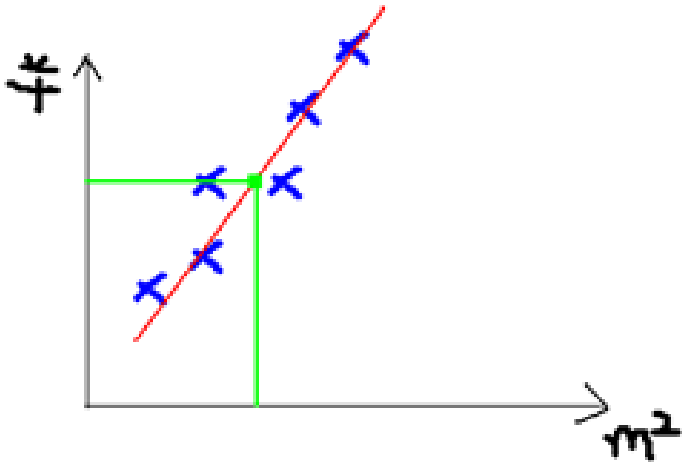
问题引入

假设有一个房屋销售的数据如下：

面积(m^2)	销售价钱（万元）
123	250
150	320
87	160
102	220

如果来了一个新的面积，假设在销售价钱的记录中没有的，怎么处理？

解决方法：用一条曲线去尽量准的拟合这些数据，然后如果有新的输入过来，我们可以在将曲线上这个点对应的值返回。如果用一条直线去拟合，可能是下面的样子：



回归学习

常用概念和符号：

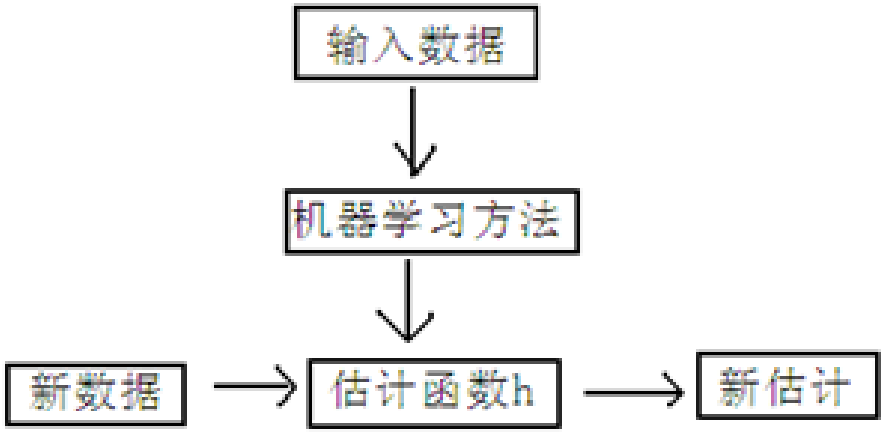
- **房屋销售记录表：**训练集(training set)或者训练数据(training data)，是我们流程中的输入数据，一般称为x
- **房屋销售价钱：**输出数据，一般称为y
- **拟合的函数（或者称为假设或者模型）：**一般写做 y = h(x)
- **训练数据的条目数(#training set)，：**一条训练数据是由一对输入数据和输出数据组成的输入数据的维度n（特征的个数，#features)

这个例子的特征是两维的，结果是一维的。然而回归方法能够解决特征多维，结果是一维多离散值或一维连续值的问题。

回归学习

学习过程

首先给出一个输入数据，算法通过一系列的过程得到一个估计的**函数**，这个函数有能力对没有见过的新数据给出一个新的估计，也被称为构建一个模型。就如同上面的线性回归函数。



一个典型的机器学习的过程

回归学习

- 线性回归（Linear regression）是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析.
- 线性回归属于**监督学习**，因此方法和监督学习应该是一样的，先给定一个训练集，根据这个训练集学习出一个**线性函数**，然后测试这个函数训练的好不好（即此函数是否足够拟合训练集数据），挑选出最好的函数（cost function最小）即可.

线性回归

注意：

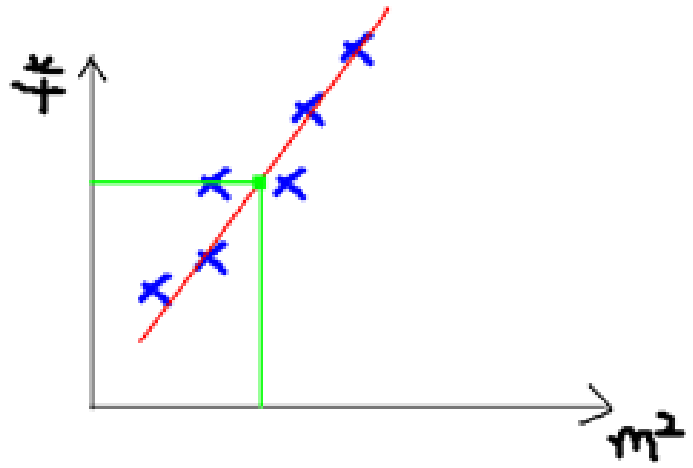
(1)因为是线性回归，所以学习到的函数为线性函数，即直线函数

(2)因为是单变量，因此只有一个x；

单变量线性回归模型：

$$h_\theta(x) = \theta_0 + \theta_1 x$$

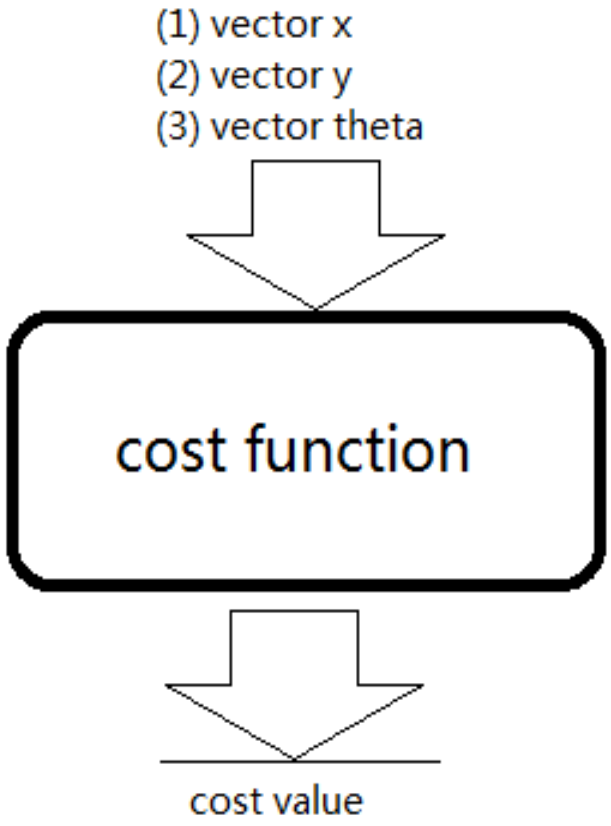
X:feature, h(x):hypothesis;



问题：线性函数拟合的好不好？

线性回归

代价函数（Cost Function）：对假设的函数进行评价，cost function越小的函数，说明拟合训练数据拟合的越好；



解释：

比如存在数据集(1,1)、(2,2)、(3,3)，则x=[1;2;3]，y = [1;2;3]

如果假设预测了函数为y = x，则发现cost value = 0；

如果假设预测了函数为y = 2x，则发现cost value = 1；

如果假设预测了函数为y = 3x，则发现cost value = 2；

则我们发现y = x 的cost value最小，因此选择 y=x 作为我们的hypothesis

线性回归

代价函数（Cost Function）：

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

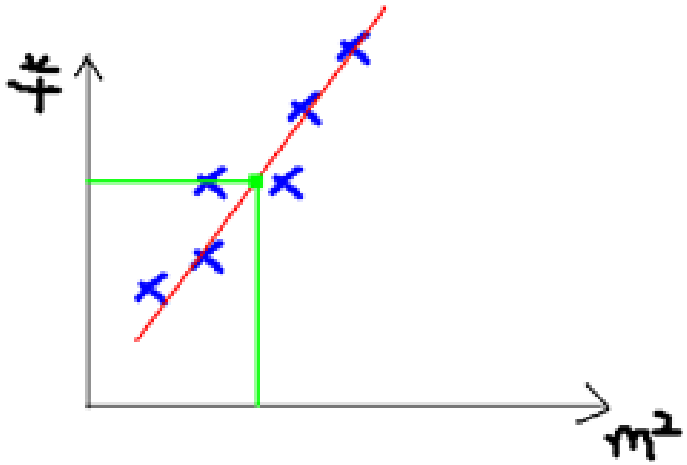
其中：

$x^{(i)}$ 表示向量x中的第i个元素；

$y^{(i)}$ 表示向量y中的第i个元素；

$h_\theta(x^{(i)})$表示已知的假设函数；

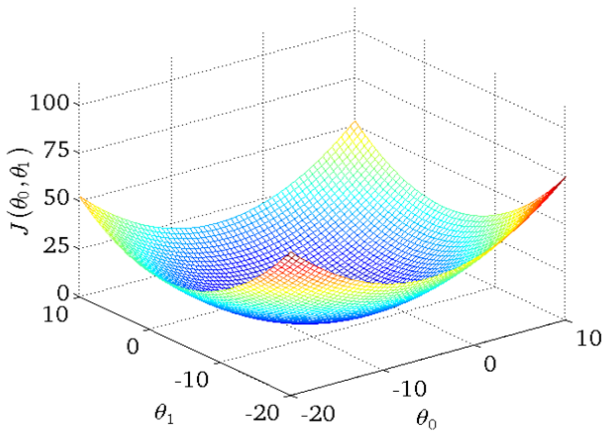
m为训练集的数量；

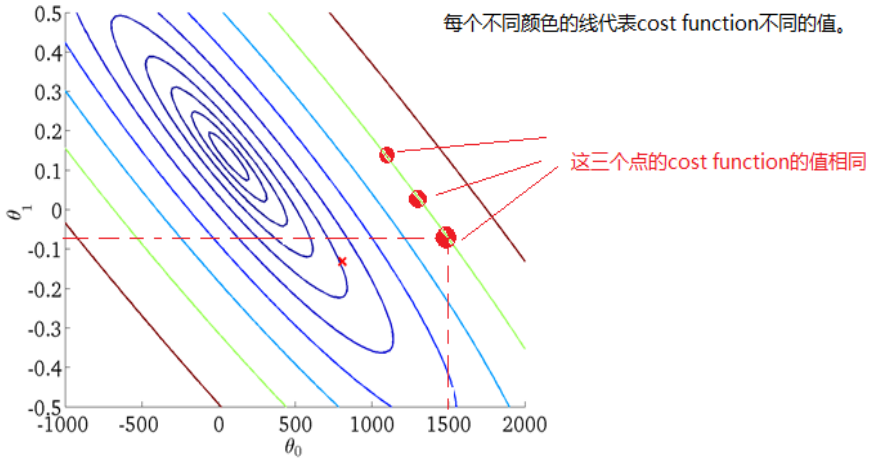


例：给定数据集(1,1)、(2,2)、(3,3)
则x = [1;2;3]， y = [1;2;3]　　（此处的语法为Octave语言的语法，表示3*1的矩阵）
如果我们预测theta0 = 0，theta1 = 1，则h(x) = x，则cost function:
J(0,1) = 1/(2*3) * [(h(1)-1)^2+(h(2)-2)^2+(h(3)-3)^2] = 0;
如果我们预测theta0 = 0，theta1 = 0.5，则h(x) = 0.5x，则cost function:
J(0,0.5) =1/(2*3) * [(h(1)-1)^2+(h(2)-2)^2+(h(3)-3)^2] = 0.58;

线性回归

代价函数与参数的关系： $J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2$





注意：如果是线性回归，则cost function J与theta0与theta1的函数一定是碗状的，即只有一个最小点；

线性回归

一般情况：$h_\theta(x) = \theta_0 + \theta_0 x_1 + \cdots \theta_n x_n = \mathbf{\theta x}$

$x^{(i)} \in R^n$ $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ **最小二乘损失函数**

$\min_\theta J_\theta$

求解：

- 最小二乘法 $\theta = (X^T X)^{-1} X^T \vec{y}.$

是一个直接的数学求解公式，不过它要求X是列满秩的，

- 梯度下降法

线性回归

Gradient Descent（梯度下降）

找出cost function函数的最小值；

梯度下降原理：将函数比作一座山，我们站在某个山坡上，往四周看，从哪个方向向下走一小步，能够下降的最快；

方法：

(1)先确定向下一步的步伐大小，我们称为Learning rate；

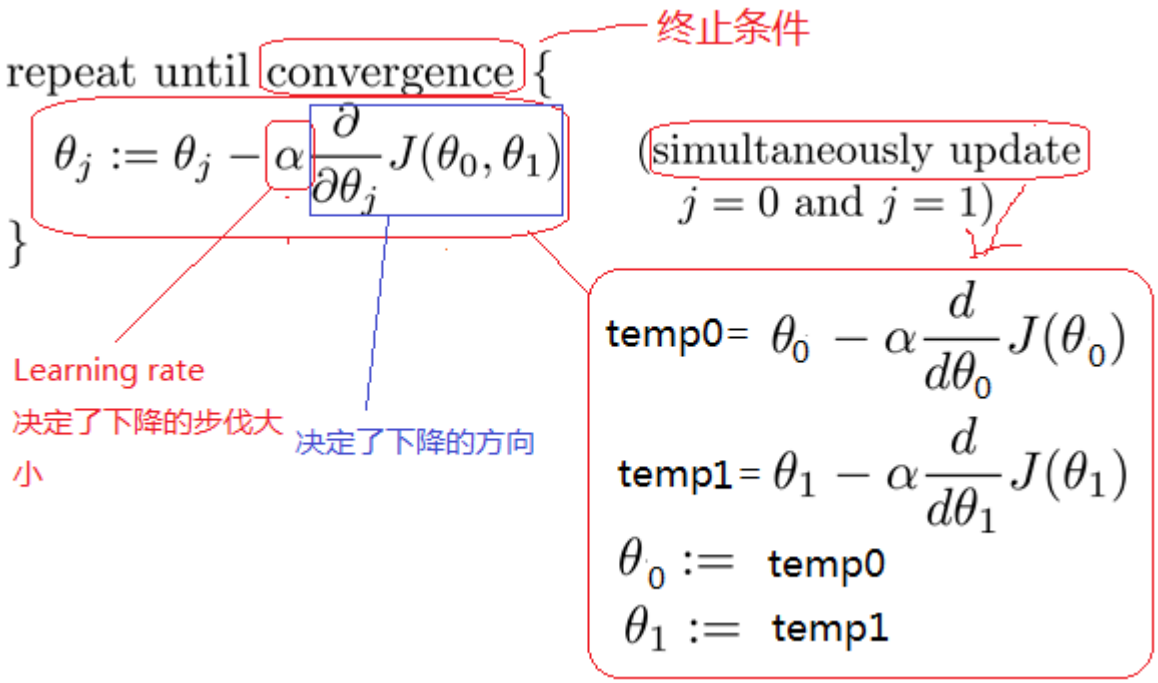
(2)任意给定一个初始值：θ_0 θ_1；

(3)确定一个向下的方向，并向下走预先规定的步伐，并更新θ_0 θ_1；

(4)当下降的高度小于某个定义的值，则停止下降；

线性回归

梯度下降算法：



特点：

(1) 初始点不同，获得的最小值也不同，因此梯度下降求得的只是局部最小值；

(2) 越接近最小值时，下降速度越慢；

线性回归

梯度下降算法：

问题：

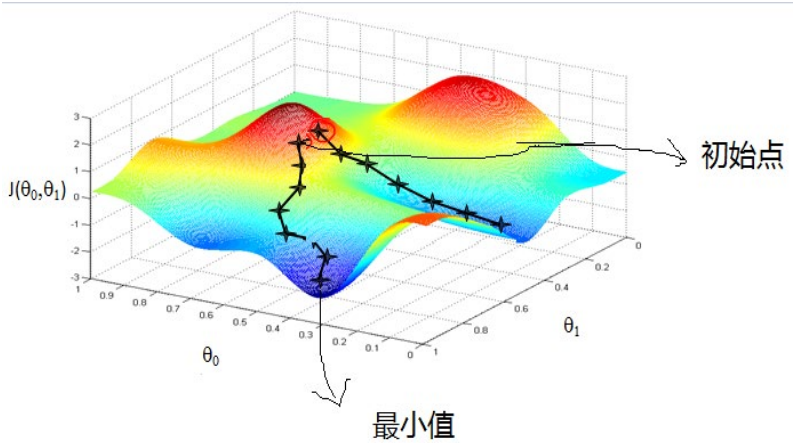
如果 θ_0 θ_1 初始值就在local minimum的位置，则 θ_0 θ_1会如何变化？

答：因为 θ_0 θ_1已经在local minimum位置，所以derivative 肯定是0，因此 θ_0 θ_1不会变化；

如果取到一个正确的 α 值，则cost function应该越来越小；

问题：怎么取 α 值？

答：随时观察 α 值，如果cost function变小了，则ok，反之，则再取一个更小的值；

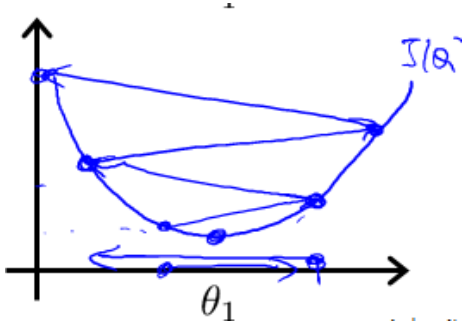


从上面的图可以看出：**初始点不同，获得的最小值也不同**，因此梯度下降求得的只是局部最小值；

线性回归

注意：下降的步伐大小非常重要，因为如果太小，则找到函数最小值的速度就很慢，

如果太大，则可能会出现overshoot the minimum的现象；



overshoot minimum现象：

如果Learning rate取值后发现J function 增长了，则需要减小Learning rate的值；

线性回归

Gradient Descent for Linear Regression

梯度下降能够求出一个函数的最小值；
线性回归需要求出h，使得cost function的最小；

因此我们能够对cost function运用梯度下降，即将梯度下降和线性回归进行整合，如下图所示：

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

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

repeat until convergence {

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

}

逻辑回归

逻辑回归（Logistic Regression）

- 逻辑回归的模型是一个非线性模型，
- sigmoid函数，又称**逻辑回归函数**。但是它本质上又是一个线性回归模型，
- 因为除去sigmoid映射函数关系，其他的步骤，算法都是线性回归的。
- 可以说，逻辑回归，都是以线性回归为理论支持的。
- 只不过，线性模型，无法做到sigmoid的非线性形式，sigmoid可以轻松处理0/1分类问题。

逻辑回归

二分类问题

二分类问题是指预测的y值只有两个取值（0或1），二分类问题可以扩展到多分类问题。例如：我们要做一个垃圾邮件过滤系统，是邮件的特征，预测的y值就是邮件的类别，是垃圾邮件还是正常邮件。对于类别我们通常称为正类（positive class）和负类（negative class），垃圾邮件的例子中，正类就是正常邮件，负类就是垃圾邮件。

- 应用举例：是否垃圾邮件分类？是否肿瘤、癌症诊断？是否金融欺诈？

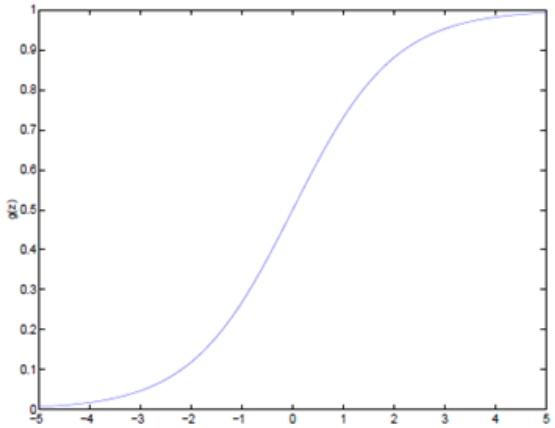
逻辑回归

Logistic函数

如果我们忽略二分类问题中y的取值是一个离散的取值（0或1），我们继续使用线性回归来预测y的取值。这样做会导致y的取值并不为0或1。逻辑回归使用一个函数来归一化y值，使y的取值在区间(0,1)内，这个函数称为Logistic函数(logistic function)，也称为Sigmoid函数(sigmoid function)。函数公式如下：

$$g(z) = \frac{1}{1 + e^{-z}}$$

- Logistic函数当z趋近于无穷大时，g(z)趋近于1；当z趋近于无穷小时，g(z)趋近于0。Logistic函数的图形如下：



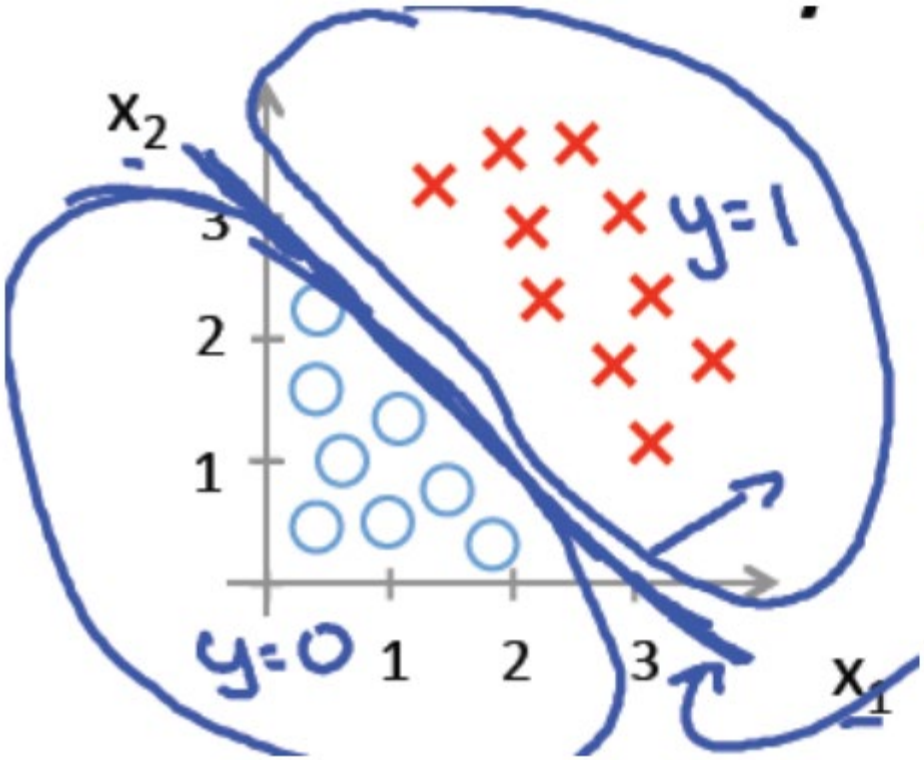
逻辑回归

Sigmoid函数(sigmoid function)的性质：

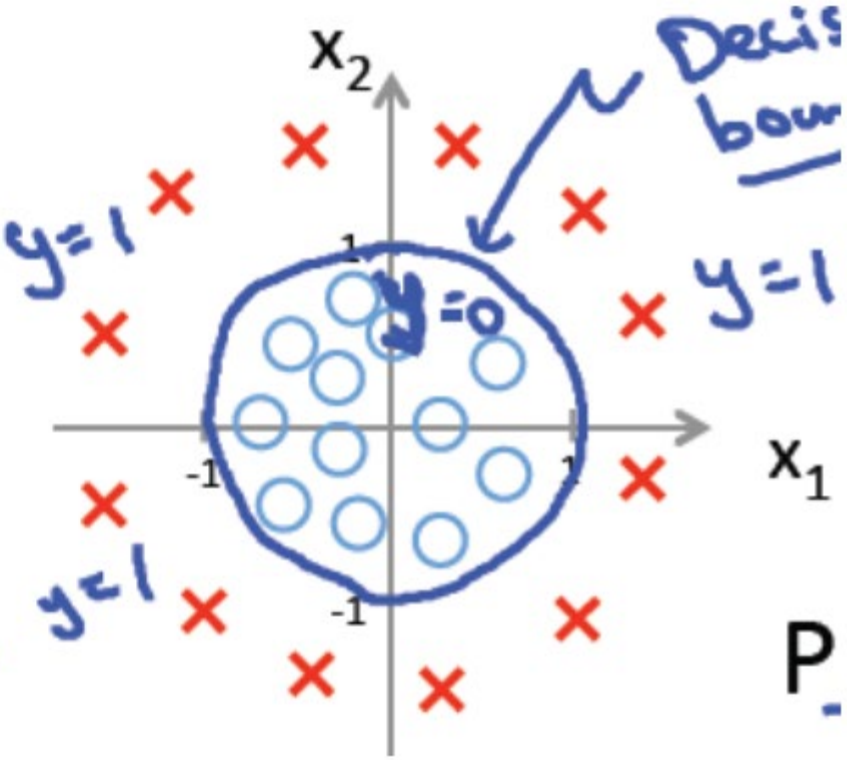
$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

逻辑回归

分类问题



线性决策边界



非线性决策边界

逻辑回归

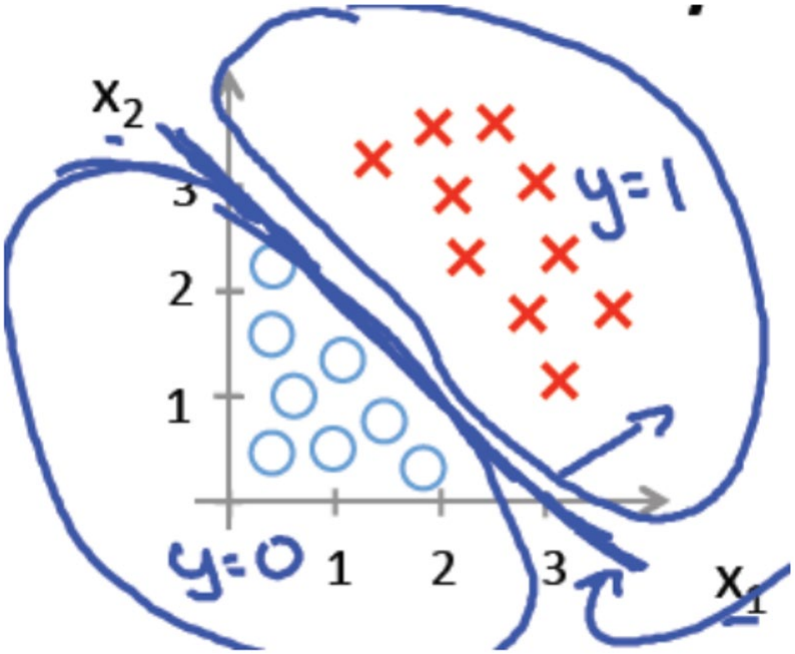
对于线性边界的情况，边界形式如下：

$$\theta_0 + \theta_1 x_1 + ,\ldots, + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$

构造预测函数为：$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

函数 $h_\theta(x)$ 的值有特殊的含义，它表示结果取1的概率，因此对于输入x分类结果为类别1和类别0的概率分别为：

$$P(y=1 \mid x;\theta) = h_\theta(x)$$
$$P(y=0 \mid x;\theta) = 1 - h_\theta(x) \quad (1)$$



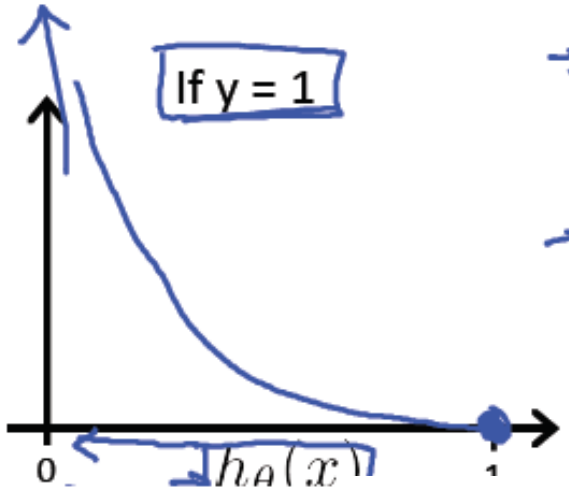
线性决策边界

逻辑回归

构造损失函数J

Cost函数和J函数如下，它们是基于最大似然估计推导得到的。

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



- $\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
  But as $h_\theta(x) \to 0$
  $Cost \to \infty$
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

逻辑回归

梯度下降算法：

θ更新过程

To fit parameters θ :

$$\min_{\theta} J(\theta)$$

Goal θ

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

$$\frac{\partial(J(\theta))}{\partial \theta_j}$$

To make a prediction given new x:

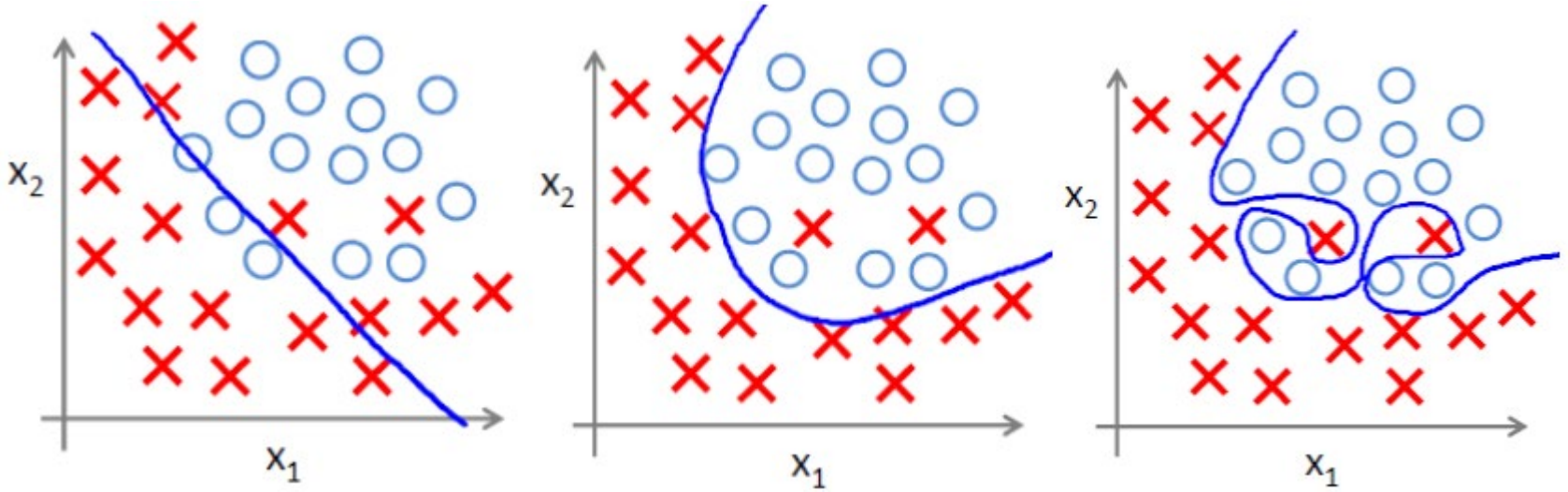
Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$p(y=1 \mid x;\theta)$

正则化Regularization

过拟合问题

对于线性回归或逻辑回归的损失函数构成的模型，可能会有些权重很大，有些权重很小，导致过拟合（就是过分拟合了训练数据），使得模型的复杂度提高，泛化能力较差（对未知数据的预测能力）。



欠拟合　　合适拟合　　过拟合

正则化Regularization

- 问题的主因：

过拟合问题往往源自过多的特征。

- 解决方法

1）减少特征数量（减少特征会失去一些信息，即使特征选的很好）

- 可用人工选择要保留的特征；
- 模型选择算法；

2）正则化（特征较多时比较有效）

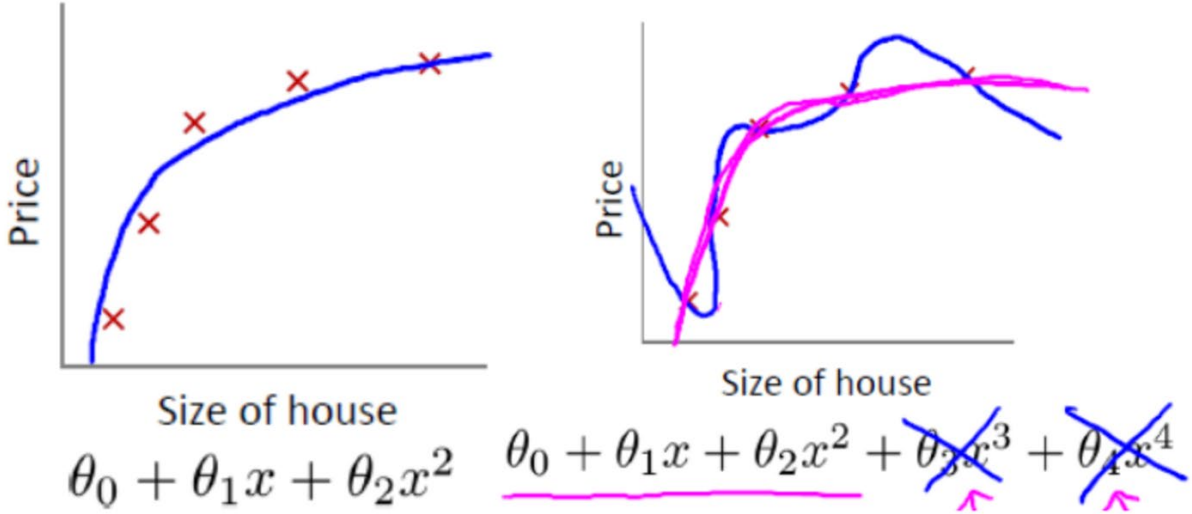
保留所有特征，但减少 θ 的大小

正则化Regularization

正则化方法

正则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项或惩罚项。正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化项就越大。

房价预测问题，多项式回归

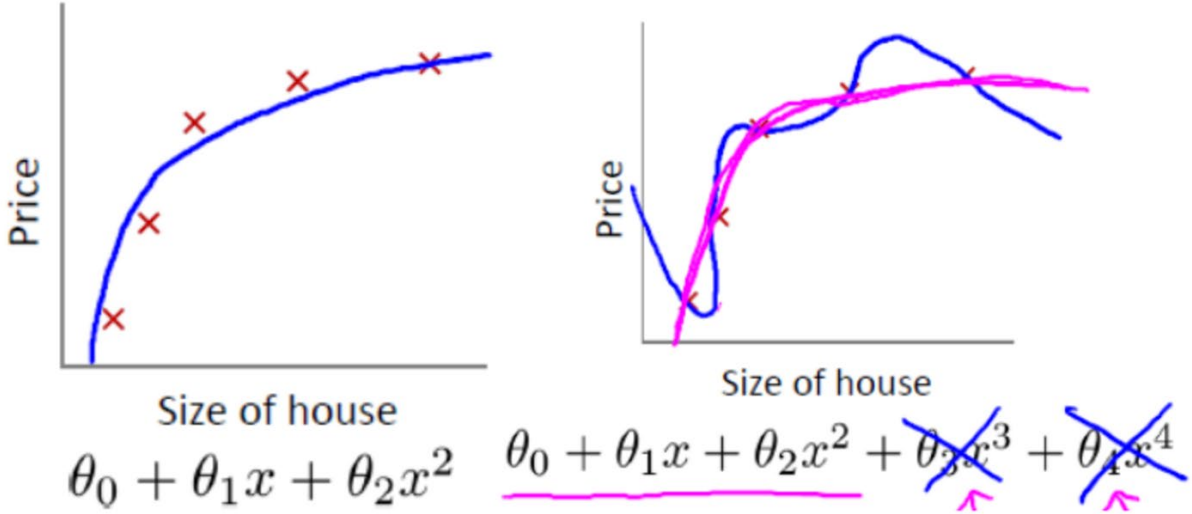


正则化Regularization

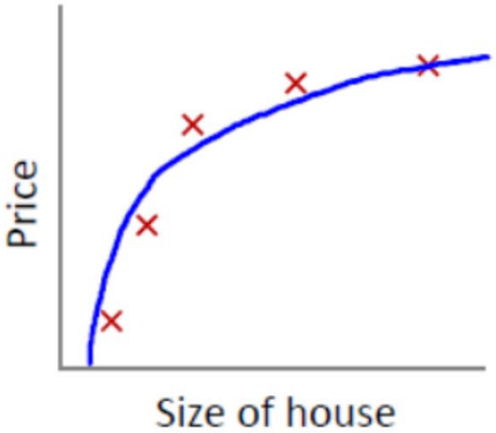
正则化方法

正则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项或惩罚项。正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化项就越大。

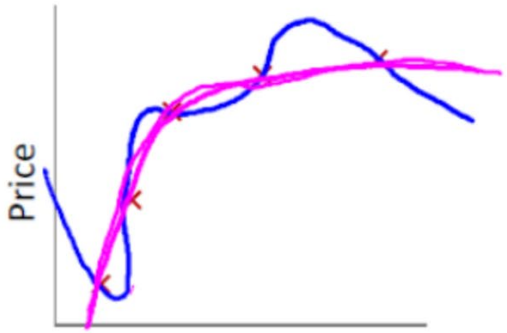
房价预测问题，多项式回归



正则化Regularization



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

直观来看，如果我们想解决这个例子中的过拟合问题，最好能将 x^3, x^4 的影响消除，也就是让 $\theta 3 \approx 0, \theta 4 \approx 0$。假设我们对 θ_3, θ_4 进行惩罚，并且令其很小，一个简单的办法就是给原有的Cost函数加上两个略大惩罚项，例如：

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

这样在最小化Cost函数的时候，$\theta 3 \approx 0, \theta 4 \approx 0$。

正则化Regularization

正则项可以取不同的形式，在回归问题中取平方损失，就是参数的L2范数，也可以取L1范数。取平方损失时，模型的损失函数变为：

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{n}(\mathrm{h}_{\theta}(\mathrm{x}_{i})-y_{i})^{2}+\lambda\sum_{j=1}^{n}\theta_{j}^{2}$$

lambda是正则项系数：

•如果它的值很大，说明对模型的复杂度惩罚大，对拟合数据的损失惩罚小，这样它就不会过分拟合数据，在训练数据上的偏差较大，在未知数据上的方差较小，但是可能出现欠拟合的现象；

•如果它的值很小，说明比较注重对训练数据的拟合，在训练数据上的偏差会小，但是可能会导致过拟合。

Softmax回归

假设预测值y有k种可能，即y∈{1,2,…,k}

比如k=3时，可以看作是要将一封未知邮件分为垃圾邮件、个人邮件还是工作邮件这三类。

$$p(y=i|x;\theta) = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$$

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)}=1|x^{(i)};\theta) \\ p(y^{(i)}=2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)}=k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Softmax回归

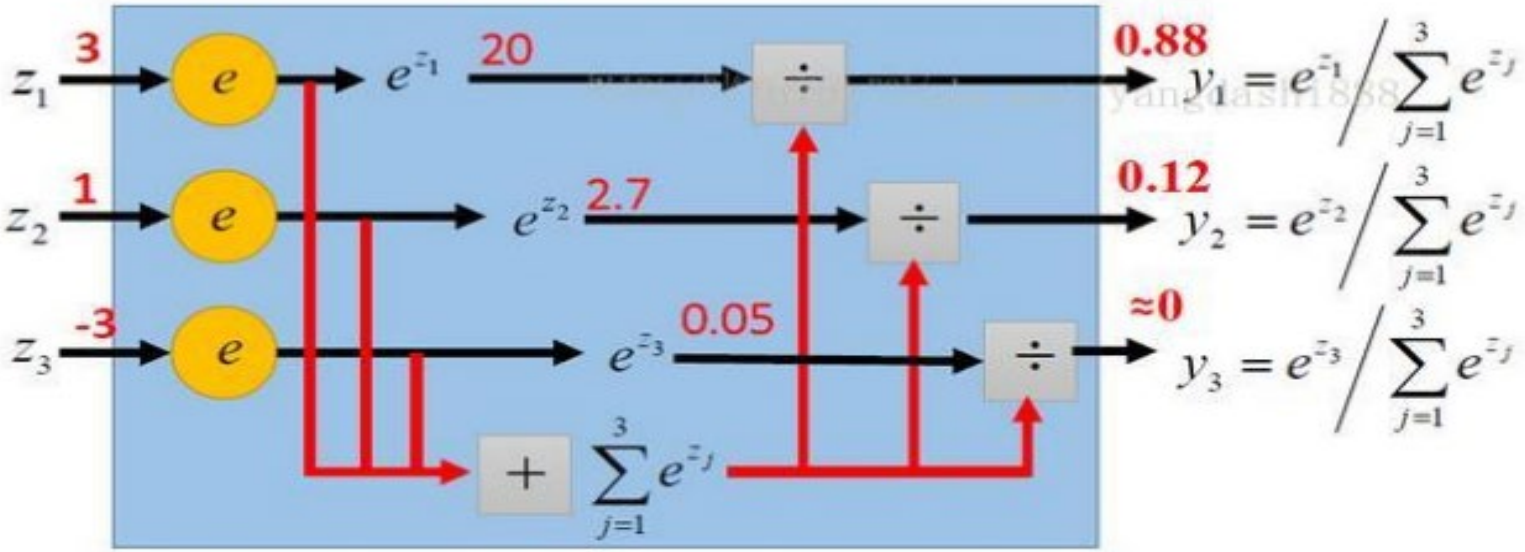
Output Layer (Option)

- Softmax layer as the output layer

Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

Softmax Layer



中间蓝色区域表示一层layer，左边输入右边输出。softmax layer的意思就明白了。

Softmax回归

Softmax回归的代价函数：

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)}=j\right\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$$

参数 $(\theta_1, \theta_2, \ldots, \theta_k)$ 是 $J(\theta)$ 的极小值点：

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}(1\{y^{(i)}=j\} - p(y^{(i)}=j|x^{(i)};\theta))\right] + \lambda\theta_j$$

θ更新过程

$$\theta_j := \theta_j - \alpha\nabla_{\theta_j} J(\theta)$$ （对每个j=1,2,...k）

1{y=k}表示当y=k的时候，1{y=k}=1