回归和梯度下降

1、基本概念

在回归中，具有代表性的是线性回归和逻辑回归

- 逻辑回归：$y=sigmoid(w^Tx+b)$
- 线性回归：$y=w^Tx+b$

也就是逻辑回归比线性回归多了一个 sigmoid 函数，sigmoid(x)=1/(1+exp(-x))，其实就是对 x 进行归一化操作，使得 sigmoid(x)位于 0~1,逻辑回归通常用于二分类模型，目标函数是二类交叉熵，y 的值表示属于第 1 类的概率，用户可以自己设置一个分类阈值。线性回归用来拟合数据，目标函数是平法和误差。

2、线性回归

2、1 单变量线性回归

方法：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；

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；

我们能够给出单变量线性回归的模型：

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

我们常称 x 为 feature，h(x)为 hypothesis；

从上面“方法”中，我们肯定有一个疑问，怎么样能够看出线性函数拟合的好不好呢？

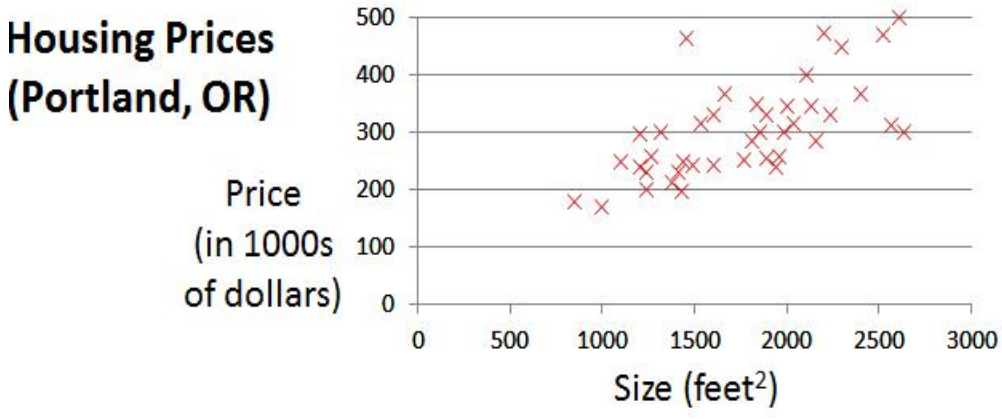
我们需要使用到 Cost Function（代价函数），代价函数越小，说明线性回归地越好（和训练集拟合地越好），当然最小就是 0，即完全拟合；

举个实际的例子：

我们想要根据房子的大小，预测房子的价格，给定如下数据集：

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

根据以上的数据集画在图上，如下图所示：



我们需要根据这些点拟合出一条直线，使得 cost Function 最小；

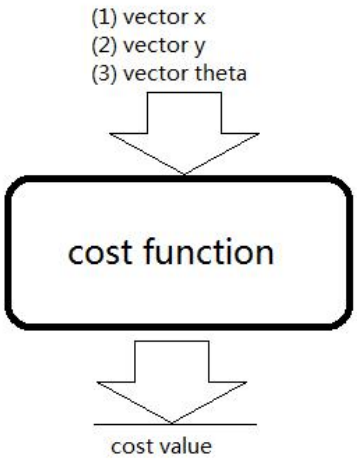
虽然我们现在还不知道 Cost Function 内部到底是什么样的，但是我们的目标

是：给定输入向量 x，输出向量 y，theta 向量，输出 Cost 值；

2、2 损失函数（代价函数）

Cost Function 的用途：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；

下图详细说明了当 cost function 为黑盒的时候，cost function 的作用；



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

但是我们肯定想知道 cost Function 的内部构造是什么？因此我们下面给出公式：

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

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；

比如给定数据集(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）

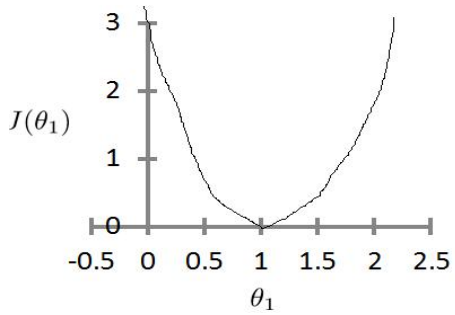
如果我们预测 theta0 = 0，theta1 = 1，则 h(x) = x，则 cost function：

J(0,1) = 1/(2*3) * [(h(1)-1)^2+(h(2)-2)^2+(h(3)-3)^2] = 0；

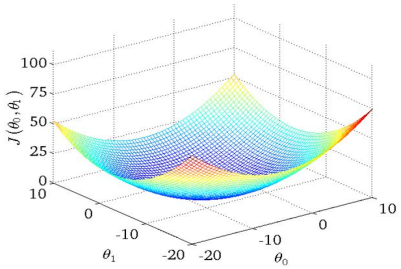
如果我们预测 theta0 = 0，theta1 = 0.5，则 h(x) = 0.5x，则 cost function：

J(0,0.5) = 1/(2*3) * [(h(1)-1)^2+(h(2)-2)^2+(h(3)-3)^2] = 0.58；

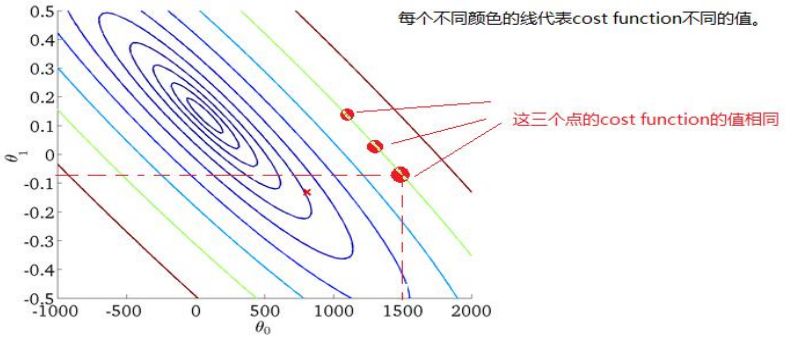
如果 theta0 一直为 0， 则 theta1 与 J 的函数为：



如果有 theta0 与 theta1 都不固定，则 theta0、theta1、J 的函数为：



当然我们也能够用二维的图来表示，即等高线图；



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

2、3 梯度下降

但是又一个问题引出了，虽然给定一个函数，我们能够根据 cost function 知道这个函数拟合的好不好，但是毕竟函数有这么多，总不可能一个一个试吧？因此我们引出了梯度下降：能够找出 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；

当然解决问题的方法有很多，梯度下降只是其中一个，还有一种方法叫 Normal Equation；

方法：

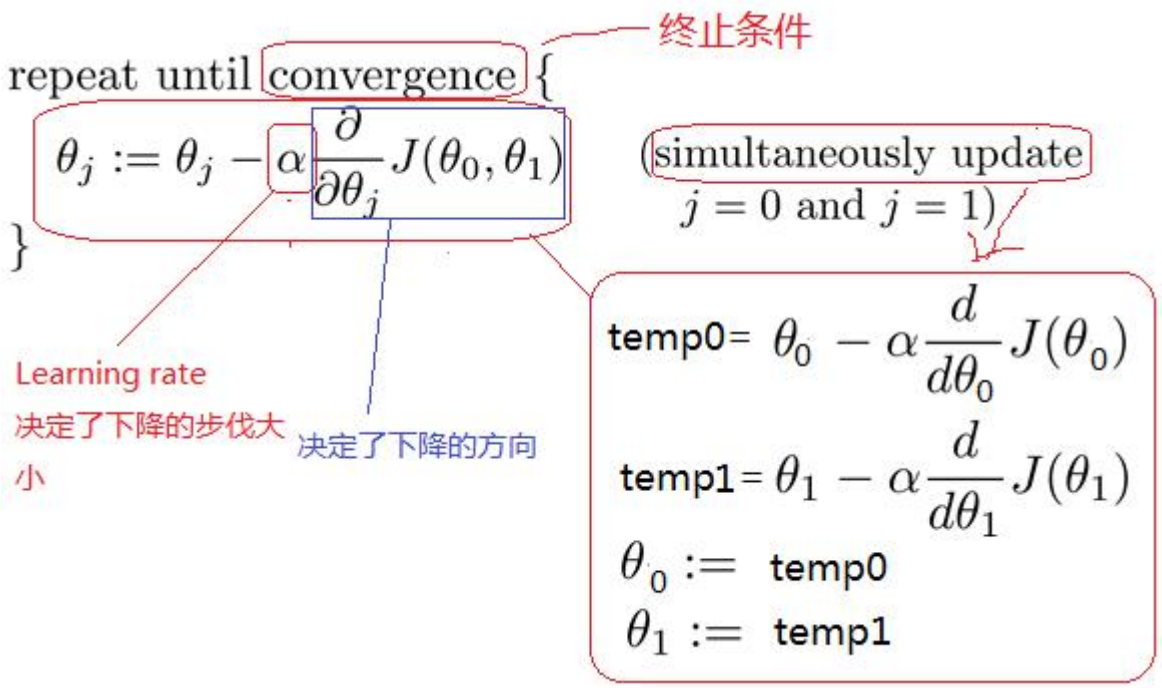
(1)先确定向下一步的步伐大小，我们称为 Learning rate；

(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；

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；

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？

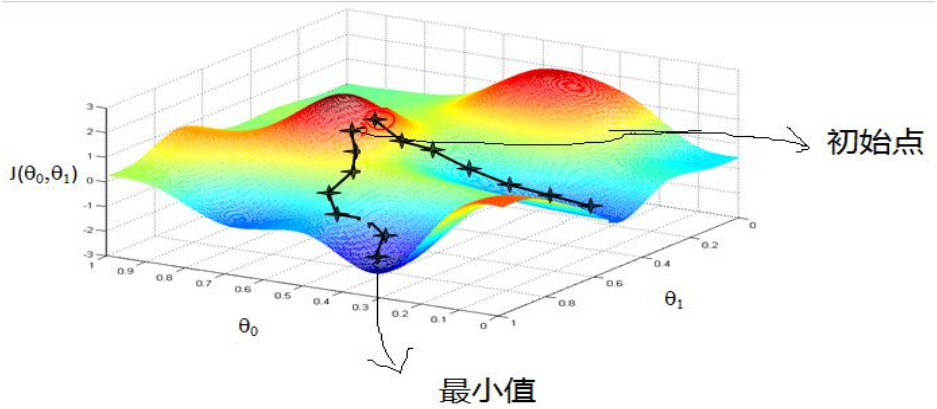
答：因为 θ_0 θ_1 已经在 local minimum 位置，所以 derivative 肯定是 0，因此 θ_0 θ_1 不会变化；

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；

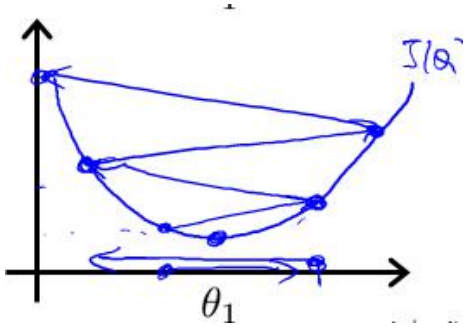
下图就详细的说明了梯度下降的过程：



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；

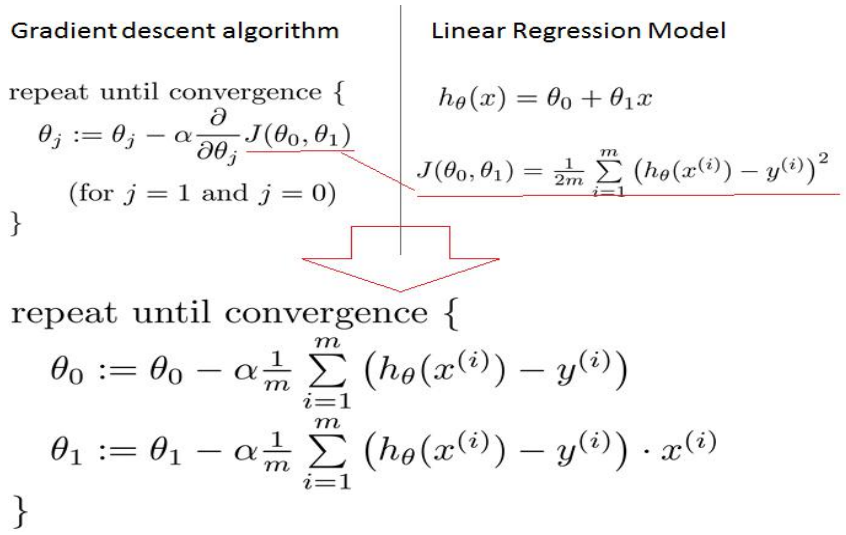
注意：下降的步伐大小非常重要，因为如果太小，则找到函数最小值的速度就很慢，如果太大，则可能会出现 overshoot the minimum 的现象；

下图就是 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；

梯度下降能够求出一个函数的最小值；线性回归需要求出使得得 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：



梯度下降是通过不停的迭代，而我们比较关注迭代的次数，因为这关系到梯度下降的执行速度，为了减少迭代次数，因此引入了 Feature Scaling；此种方法应用于梯度下降，为了加快梯度下降的执行速度；

思想：将各个 feature 的值标准化，使得取值范围大致都在-1<=x<=1 之间；

常用的方法是 Mean Normalization，即

$\frac{x-\bar{x}}{max-min}$，其中$\bar{x}$为训练集中当前feature的平均值，max为x能取的最大值，min为x能取的最小值

或者：

[X-mean(X)]/std(X);

举个实际的例子，

有两个 Feature：

(1)size，取值范围 0~2000；

(2)#bedroom，取值范围 0~5；

则通过 feature scaling 后，

$$x_1 = \frac{size-1000}{2000} \quad x_2 = \frac{\#bedrooms-2}{5}$$

2、4 多变量线性回归

前面我们只介绍了单变量的线性回归，即只有一个输入变量，现实世界不可能这么简单，因此此处我们要介绍多变量的线性回归；

举个例子：

房价其实由很多因素决定，比如 size、number of bedrooms、number of floors、age of home 等，这里我们假设房价由 4 个因素决定，如下图所示：

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
…	…	…	…	…

我们前面定义过单变量线性回归的模型：

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

这里我们可以定义出多变量线性回归的模型：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Cost function 如下：

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

如果我们要用梯度下降解决多变量的线性回归，则我们还是可以用传统的梯度下降算法进行计算：

Repeat {

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

代表第i个训练记录的第j个特征

(simultaneously update θ_j for $j = 0, \ldots, n$)

}

举例

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

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

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

例如：

1.我们想要根据一个学生第一年的成绩预测第二年的成绩，x 为第一年得到 A 的数量，y 为第二年得到 A 的数量，给定以下数据集：

x	y
3	4
2	1
4	3
0	1

(1)训练集的个数是多少？ 4 个

(2)J(0,1)的结果是多少？

J(0,1) = 1/(2*4)*[(3-4)^2+(2-1)^2+(4-3)^2+(0-1)^2] = 1/8*(1+1+1+1) = 1/2 = 0.5；

我们也可以通过 vectorization 的方法快速算出 J(0,1)：

$$X = \begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 1 & 4 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 4 \\ 1 \\ 3 \\ 1 \end{bmatrix}, \theta = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow h(x) = X\theta = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 0 \end{bmatrix} \Rightarrow h(x) - Y = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow (h(x) - y)^2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\sum (h(x) - y)^2 = 4 \Rightarrow J(0,1) = \frac{1}{2*4} * 4 = 1/2$$

3、逻辑回归

3、1 逻辑回归模型

回归是一种极易理解的模型，就相当于 y=f(x)，表明自变量 x 与因变量 y 的关系。最常见问题有如医生治病时的望、闻、问、切，之后判定病人是否生病或生了什么病，其中的望闻问切就是获取自变量 x，即特征数据，判断是否生病就相当于获取因变量 y，即预测分类。

最简单的回归是线性回归，在此借用 Andrew NG 的讲义，有如图 1.a 所示，X 为数据点——肿瘤的大小，Y 为观测值——是否是恶性肿瘤。通过构建线性回归模型，如 hθ(x)所示，构建线性回归模型后，即可以根据肿瘤大小，预测是否为恶性肿瘤 hθ(x)≥.05 为恶性，hθ(x)<0.5 为良性。

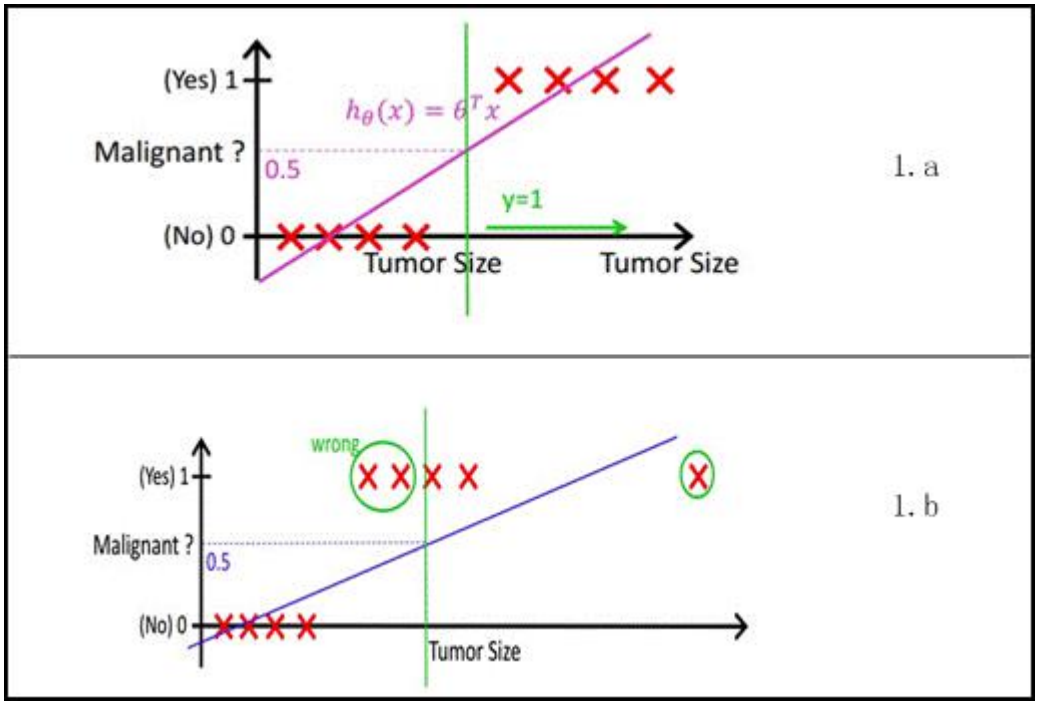


图 1 线性回归示例

然而线性回归的鲁棒性很差，例如在图 1.b 的数据集上建立回归，因最右边噪点的存在，使回归模型在训练集上表现都很差。这主要是由于线性回归在整个实数域内敏感度一致，而分类范围，需要在[0,1]。逻辑回归就是一种减小预测范

围，将预测值限定为[0,1]间的一种回归模型，其回归方程与回归曲线如图 2 所示。逻辑曲线在 z=0 时，十分敏感，在 z>>0 或 z<<0 处，都不敏感，将预测值限定为(0,1)。

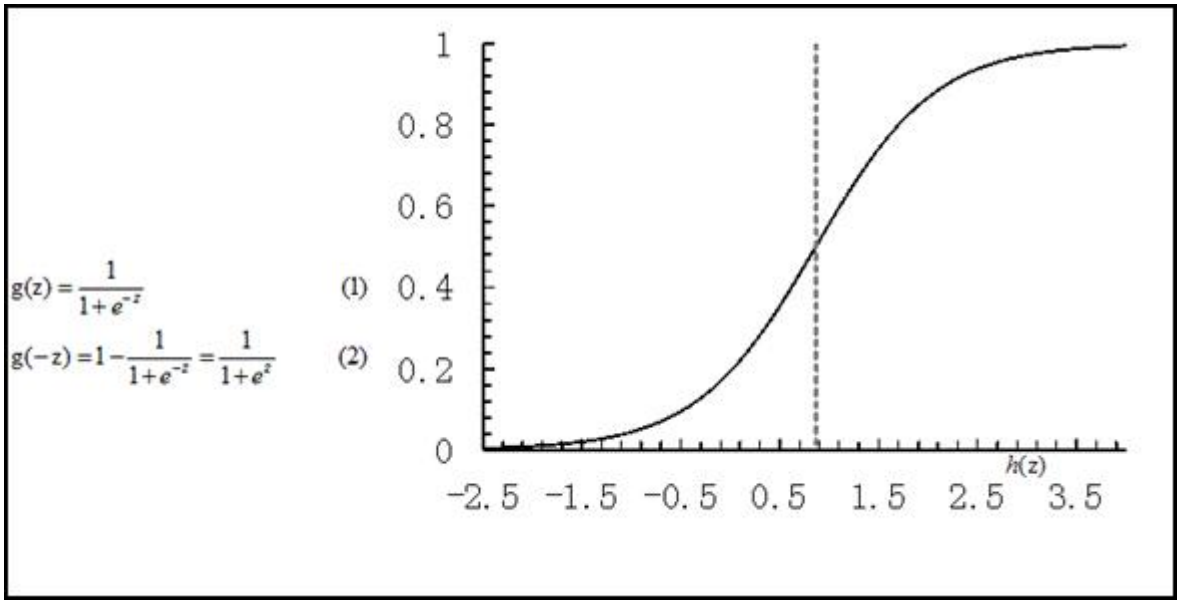


图 2 逻辑方程与逻辑曲线

逻辑回归其实仅为在线性回归的基础上，套用了一个逻辑函数，但也就由于这个逻辑函数，逻辑回归成为了机器学习领域一颗耀眼的明星，更是计算广告学的核心。对于多元逻辑回归，可用如下公式似合分类，其中公式(4)的变换，将在逻辑回归模型参数估计时，化简公式带来很多益处，y={0,1}为分类结果。

$$\begin{cases} p(y=1\,|\,x,\theta)=\dfrac{1}{1+e^{-\theta^T x}} & (3) \\ p(y=0\,|\,x,\theta)=\dfrac{1}{1+e^{-\theta^T x}}=1-p(y=1\,|\,x,\theta)=p(y=1\,|\,x,-\theta) & (4) \end{cases}$$

$$令:h_\theta(x)=g(\theta^T x)=\frac{1}{1+e^{-\theta^T x}};\ g(z)=\frac{1}{1+e^{z}}$$

对于训练数据集，特征数据 x={x1, x2, ... , xm}和对应的分类数据 y={y1, y2, ... , ym}。构建逻辑回归模型 f(θ)，最典型的构建方法便是应用极大似然估计。首先，对于单个样本，其后验概率为：

$$p(y \mid x, \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$
其中$y = 1$(或0)

那么，极大似然函数为：

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

log 似然函数是：

$$l(\theta) = \log(L(\theta \mid x, y))$$
$$= \sum_{i=1}^{m} y^{(i)} \log(h(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

3、2 逻辑回归中的梯度下降

由第 1 节可知，求逻辑回归模型 f(θ)，等价于：

$$\theta^* = \arg\min_{\theta}(l(\theta))$$

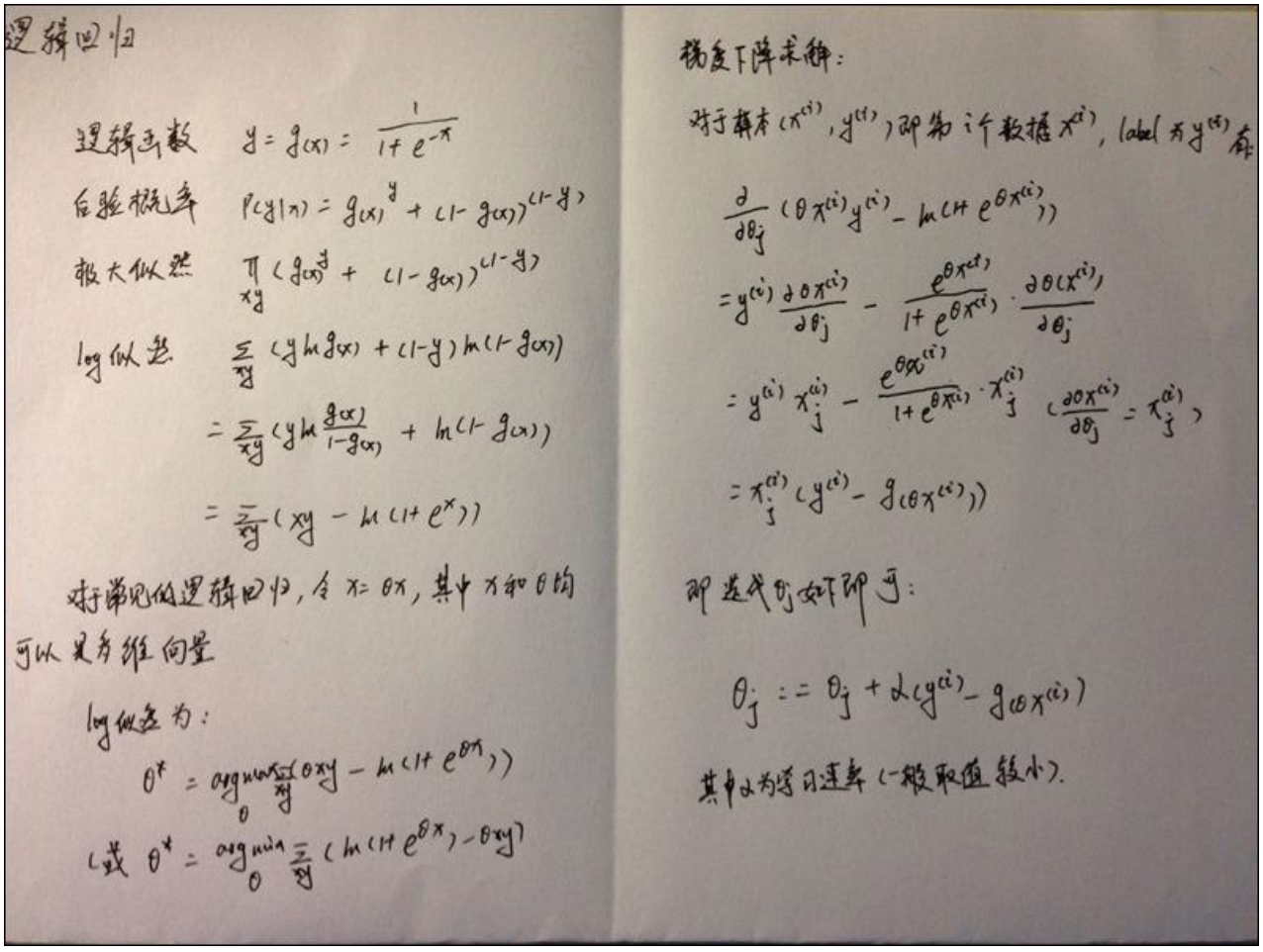
采用梯度下降法：

$$\frac{\partial}{\partial \theta_j}(l(\theta)) = \frac{\partial}{\partial \theta_j}(\sum_{i=1}^{m} y^{(i)} \log(h(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(x^{(i)})))$$
$$= (\frac{y^{(i)}}{h(x^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - h(x^{(i)})}) \frac{\partial}{\partial \theta_j}(h(x^{(i)}))$$
$$= (\frac{y^{(i)}}{g(\theta^T x^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - g(\theta^T x^{(i)})}) \frac{\partial}{\partial \theta_j}(g(\theta^T x^{(i)}))$$
$$= (\frac{y^{(i)}}{g(\theta^T x^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - g(\theta^T x^{(i)})}) g(\theta^T x^{(i)})(1 - g(\theta^T x^{(i)})) \frac{\partial \theta^T x^{(i)}}{\partial \theta_j}$$
$$= (y^{(i)}(1 - g(\theta^T x^{(i)}) - (1 - y^{(i)}) g(\theta^T x^{(i)}))) x_j$$
$$= (y^{(i)} - h_\theta(x^{(i)})) x_j$$

从而迭代θ至收敛即可：

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$$

演算手稿：



3、3 逻辑回归选用 sigmoid 函数的原因

- 在用回归处理二分类问题时，一般会采用单位阶跃（unit-step）函数或者硬限幅（hardlim）函数，但是这两个函数并不是一个连续的函数，在一个很小的狭区间内函数会突然从 0 阶跃到 1，没有过渡状态，在处理支撑超平面的误分类样本点时会发生函数震荡，影响分类精度。
- Logstic 函数虽然不能给出函数的解析解，但是求解其最优解的过程使用了梯度信息和迭代公式，他可以保证收敛的速度以及最优解的存在。
- Logstic 函数的对数最大似然函数 L(θ)是一个指数和对数函数，它是一个凸函数，而凸函数有个最重要的性质就是：局部最优即全局最优。从某种程度上克服了梯度下降的容易导致陷入局部最优的缺点。