监督学习

监督学习是已经知道数据的label（数据的意义），例如预测房价问题，给出了房子的面积和价格。

- **回归**问题（regression）是预测连续值的输出
  - 比如通过房子面积（x轴）来预测房价（y轴）。
- **分类**（classification问题是预测离散值输出
  - 比如给定肿瘤大小（x轴），年龄（y轴）来判断肿瘤是良性还是恶性。

无监督学习

无监督学习是不知道数据具体的含义，比如给定一些数据但不知道它们具体的信息，对于分类问题无监督学习可以得到多个不同的聚类，从而实现预测的功能。

> 聚类：是按照某个特定标准(如距离)把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇中的数据对象的差异性也尽可能地大。

线性回归

线性回归是拟合一条线，将训练数据尽可能分布到线上。另外还有多变量的线性回归称为多元线性回归。

代价函数

- cost function，也称为损失函数。用来量化数据偏离程度（误差），一般使用最小均方差来评估参数的好坏。
- $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
  - m代表训练样本的数量，除以1/m，是为了消除样本数量m对 J 的影响
  - 与均方差公式相比，除以1/2m是为了抵消对 J 求偏导数时的"2"，使得计算更加方便
- 图中$y = mx$, 就是预测函数(hypothesis)，斜率w即参数。
  - 由预测函数推导误差公式，得到代价函数。在本图当中，改变w，利用均方差公式得出代价函数。
  - 改变w，使得左侧代价函数达到最低点，右侧实现了最佳拟合。

- 当我们考虑到多参数时候，绘制出代价函数即为一个三维曲面。
- 同时可以使用等高线图、等高图像表示代价函数。

- 对于三个具有相同的 J 值（等高线），最小值就是同心椭圆的中心（拟合点

梯度下降

梯度下降，首先为每个参数赋一个初值，通过代价函数的梯度，然后不断地调整参数，最终得到一个**局部最优解**。初值的不同可能会得到两个不同的结果，即梯度下降不一定得到全局最优解。

梯度下降公式：$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

> 赋值：a:=b
>
> 判断真假：a=b

梯度下降公式主要包括学习率a、偏导数。

- 细节注意：参数的变化是**同时**的
- 从左或者右边逼近，学习率a是一个整数，×对应的偏导数（有正有负），不断向最小值逼近。
- 学习率a的选择
  - 如果α太小，学习速率太小，收敛很慢，因为它会一点点挪动，它会需要很多步才能到达全局最低点。
  - 如果α太大，那么梯度下降法可能会越过最低点，下一次迭代又移动了一大步，越过一次，又越过一次，一次次越过最低点，直到你发现实际上离最低点越来越远，最终会导致无法收敛，甚至发散。

在梯度下降法中，当接近局部最低点时，梯度下降法会自动采取更小的幅度。这是因为当我们接近局部最低时，导数值会自动变得越来越小，在局部最低时导数等于零。所以梯度下降将自动采取较小的幅度，所以实际上没有必要再另外减小α。

线性回归中的梯度下降

梯度下降算法和线性回归算法，如图：

Gradient descent algorithm	Linear Regression Model

repeat until convergence {
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
(for $j = 1$ and $j = 0$)
}
	$h_\theta(x) = \theta_0 + \theta_1 x$
	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

对我们之前的线性回归问题运用梯度下降法，关键在于求出代价函数的导数，即：

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

$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x$

当$j = 0$时：$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

当$j = 1$时：$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) * x^{(i)})$

则算法改写成：

Gradient descent algorithm

repeat until convergence {

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

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

} update θ_0 and θ_1 simultaneously

我们刚刚使用的算法，被称为**批量梯度下降（Batch）**。它指的是在梯度下降的每一步中，都用到了所有的训练样本，在计算微分求导项时，需要进行求和运算。所以，在每一个单独的梯度下降中，都需要对所有m个训练样本求和。而事实上，也有其他类型的梯度下降法每次只关注训练集中的一些小的子集。

线性代数中，还有一种计算代价函数J最小值的数值解法，**正规方程(normal equations)**，不需要梯度下降这种迭代算法，但是处理大型数据集时，梯度下降是更好的方法。

多参数线性回归

定义

- 引入多种特征后的假设h模型，例如对于（面积，价格）的房价模型，现在我们对房价模型增加更多的特征 (房间数量, 楼层数量, 房屋年龄)

Multiple features (variables).

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
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：

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

  - 增加了$x_0 = 1$，使得假设矩阵可以用一个行向量与列向量表示假设矩阵。

多变量梯度下降

- 模型如下图：

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

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

Repeat $\{$

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

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

- 对梯度下降求导数后得到：

Gradient Descent

Previously (n=1):

Repeat $\{$

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

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

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

$\}$

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

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

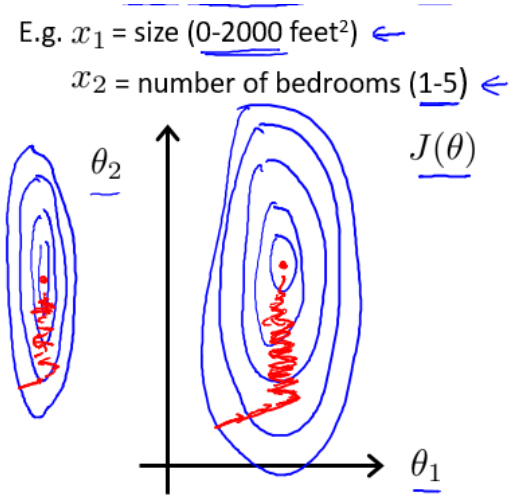
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

- 理解这边的时候，带入假设函数，并且注意多变量，这里变量指的是θ，不是x

特征缩放

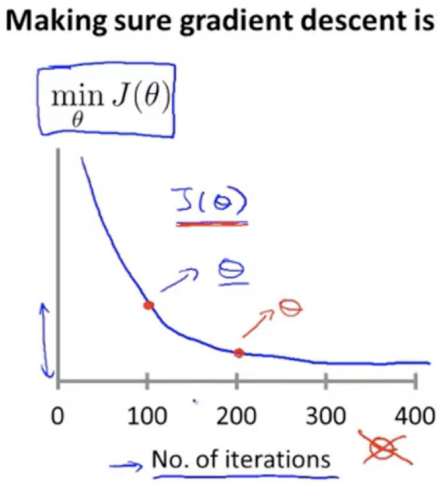
- 梯度下降中的特征缩放: 确保特征具有相近的尺度。这可以帮助梯度下降算法更快地收敛。
- 以房价问题为例，假设我们使用两个特征，房屋尺寸的值为 0-2000 平方英尺，而房间数量的值则是 0-5，以两个参数分别为横纵坐标，绘制代价函数的等高线图，能看出图像会显得很扁，梯度下降算法需要非常多次的迭代才能收敛。



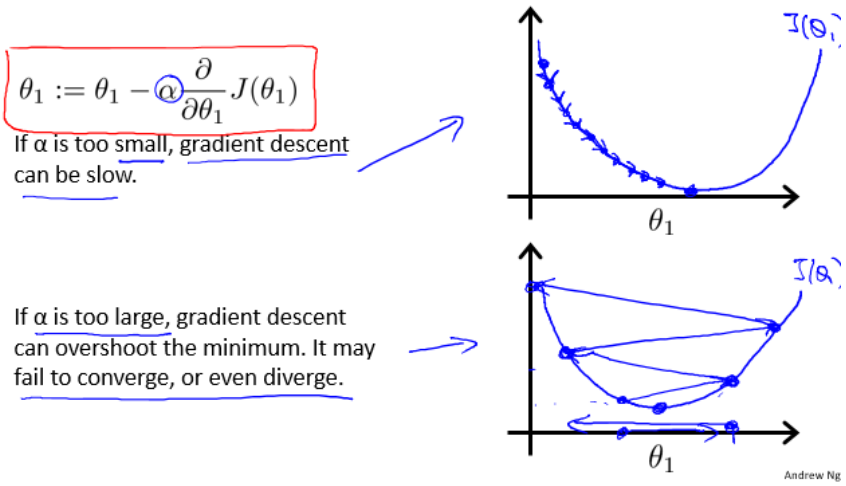
- 一般最简单的方法是令：$x_n = \frac{x_n - \mu_n}{s_n}$，参数$\mu_n$、$s_n$分别为平均值和标准差，$s_n$同时还可以用最大值和最小值进行缩放。
- 缩放的区间一般为-3~+3，-1/3~1/3，在-1~1是广义上的，稍大稍小都能接受。

学习率 a

- 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，因此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**自动测试**是否收敛地方法。
- 梯度下降算法的每次迭代受到学习率的影响:
  - 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。



- 根据实验需求选择合适的学习率，通常可以考虑尝试些学习率： 0.01，0.03，0.1，0.3，1，3，10。

特征与多项式回归

- 面对多参数问题时，可以自己创建一个特征，特征数据为x1=frontage（横向宽度），x2=depth（纵向深度），可以转换为房子面积的特征。
- 也可以观察数据尝试转换模型，例如$(size)^2$转为x_2, 将模型转换为线性模型。

正规方程

- 正规方程是通过求解下面的方程，求解代价函数的梯度等于 0，来找出使得代价函数最小的参数的：$\frac{\partial}{\partial \theta_j} J(\theta_j) = 0$
- 举个栗子

Examples: $m = 4$.

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

- 注意x_0为1，所以有n+1个参数
- X为设计矩阵，y输出
- 使用正规方程不需要特征缩放

梯度下降与正规方程比较

m training examples, n features.

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
$n = 10^6$	• Slow if n is very large.
	$n = 100$
	$n = 1000$
	$n = 10000$

- 特征变量n<10^4 使用正规方程，计算机可以很快计算设计矩阵，特征变量较大时使用梯度下降法。
- 但不是说梯度下降应用不如正规方程，随着更复杂的学习算法，不得不使用梯度下降，对于特定问题正规方程是很好的方法。

正规方程可逆讨论

- 对于Octave可以使用pinv、inv计算逆矩阵。（伪逆）
- X不可逆的原因
  - n较大的情况
  - 特征冗余的情况，不同特征之间存在线性关系

- 解决：
    - 删除重复、线性相关的特征
    - 如果特征数量实在太多，可以删除些用较少的特征来反映尽可能多内容，或者考虑使用正则化方法。
    - 对于不可逆X^TX，使用伪逆函数pinv仍可以计算出结果。

总之，出现不可逆矩阵的情况极少发生，所以在大多数实现线性回归中，不应该过多的关注X^TX

正规方程解的推导过程

设计矩阵的求导法则，只是了解了下，参考以下文章

- 正规方程不可逆的情况 (momodel.github.io)
- 正规方程（含推导过程）

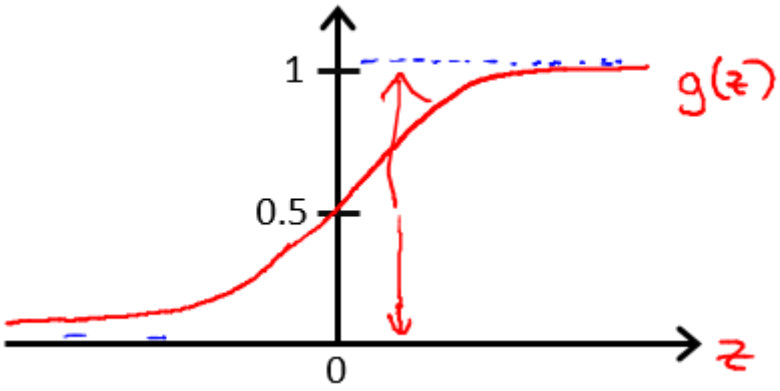
逻辑回归

定义

- 是分类问题的一个算法
- 对比线性回归，假设函数不同
    - 线性回归用一个直线去拟合
- 采用逻辑回归的分类算法，这个算法的性质是：它的输出值永远在 0 到 1 之间。 它适用于标签 y 取值离散的情况，如：1 0 0 1。
- 正类负类是什么

假设函数

- 回忆线性回归的假设函数是什么
- 回归模型的假设是： $h_\theta(x)=g(\theta^TX)=\frac{1}{1+e^{-\theta^TX}}$
    - X 代表特征向量
    - g 代表逻辑函数（logistic function），下图是一个常用的逻辑函数为 S 形函数（Sigmoid function），公式为：$g(z)=\frac{1}{1+e^{-z}}$
    - 该函数的图像为：

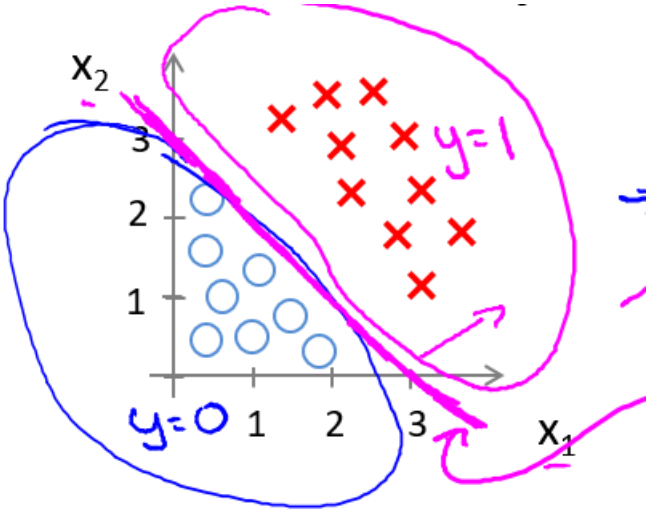


    - 函数的含义，对于给定的输入变量，根据选择的参数计算输出变量为1 的可能性（estimated probablity），即：$h_\theta(x)=P(y=1|x;\theta)$

决策边界

- 是假设函数的一个属性
- 参数θ是向量[-3 1 1]。 则当$-3+x_1+x_2$大于等于 0，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。此时界限可以称为决策边界。

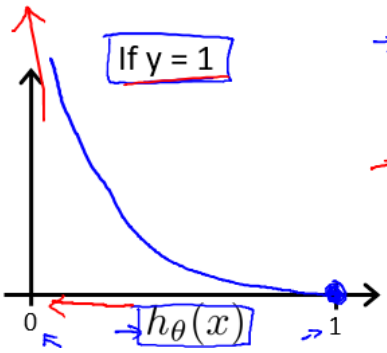


代价函数

- 代价函数是用来拟合逻辑回归的参数，这便是监督学习问题中的逻辑回归模型的拟合问题。
- 对比线性回归的代价函数，逻辑回归的代价函数$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}Cost(h_\theta(x^{(i)}),y^{(i)})$

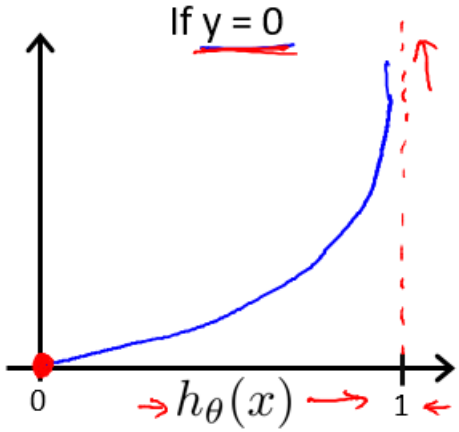
$$\text{Cost}(h_\theta(x),y)=\begin{cases}-\log(h_\theta(x)) & \text{if } y=1\\ -\log(1-h_\theta(x)) & \text{if } y=0\end{cases}$$

- $h_\theta(x)$与$Cost(h_\theta(x),y)$之间的关系图：



$\text{Cost}=0$ 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.



- 从上述代价函数可以看出，在y=1，hx=1时候，误差为0，hx≠1时候，随着hx减小，误差不断增大，同理可以看到在y=0的情况。

简化代价函数与梯度下降

- 合并简化代价函数为：

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

  我们可以看到当y=1或者y=0时候分别可以抵消另一项。

- 最小化代价函数的方法，是使用梯度下降法(gradient descent)。这是通常用的梯度下降法的模板：

Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

$\}$ (simultaneously update all θ_j)

反复更新每个参数，用这个式子减去学习率 α 乘以后面的微分项。求导后得到：

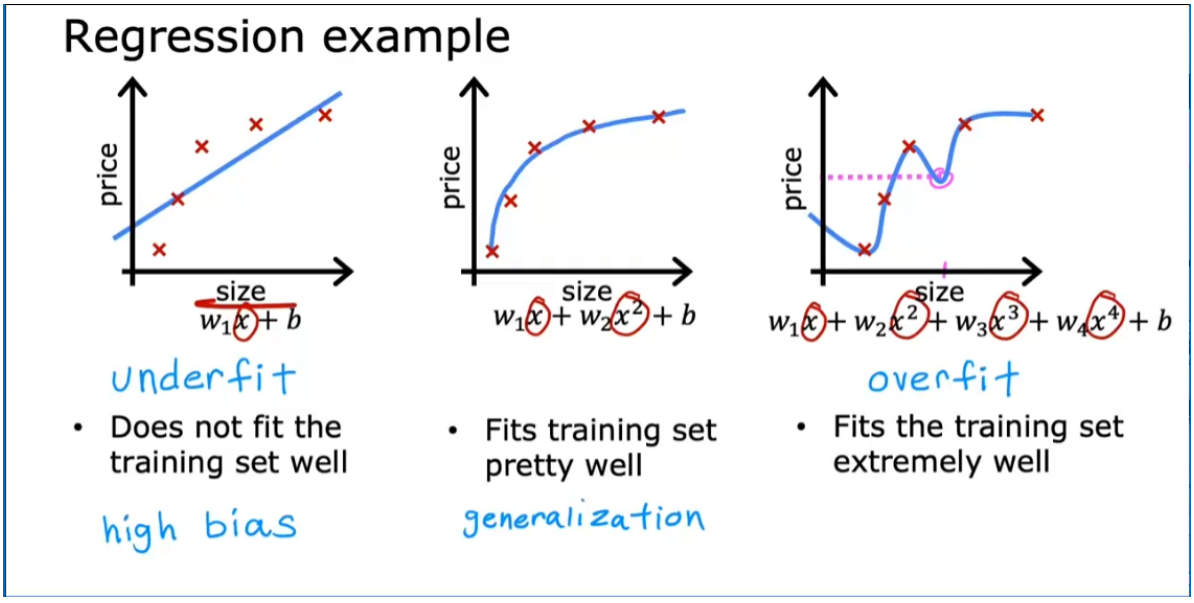
$$
\begin{aligned}
l(\theta) &= log(\prod_{i=1}^{m} h_\theta(x^i)^{y^{(i)}}(1-h_\theta(x^{(i)})^{1-y^{(i)}}) \\
&= \sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^i)) + (1-y^{(i)})(1-h_\theta(x^{(i)})]
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta} &= \sum_{i=1}^{m}[\frac{y^{(i)}}{h_\theta(x^i)}\frac{\partial h_\theta(x^i)}{\partial \theta} + \frac{1-y^{(i)}}{1-h_\theta(x^i)}\frac{\partial h_\theta(x^i)}{\partial \theta}(-1)] \\
&= \sum_{i=1}^{m}[\frac{\partial h_\theta(x^i)}{\partial \theta}(\frac{y^{(i)}}{h_\theta(x^i)} - \frac{1-y^{(i)}}{1-h_\theta(x^i)})] \\
&= \sum_{i=1}^{m}[\frac{\partial h_\theta(x^i)}{\partial \theta}(\frac{y^{(i)}(1-h_\theta(x^i)) + (y^{(i)}-1)h_\theta(x^i)}{h_\theta(x^i)(1-h_\theta(x^i))})] \\
&= \sum_{i=1}^{m}[\frac{\partial h_\theta(x^i)}{\partial \theta}(\frac{y^{(i)}-h_\theta(x^i)}{h_\theta(x^i)(1-h_\theta(x^i))})]
\end{aligned}
$$

其中$\frac{\partial h_\theta(x^i)}{\partial \theta} = h_\theta(x^i)(1-h_\theta(x^i))x^{(j)}$,先对$h_\theta(x^i)$求导，然后回带$h_\theta(x^i)$。最后化简结果为$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$。这也是等价于线性回归的函数

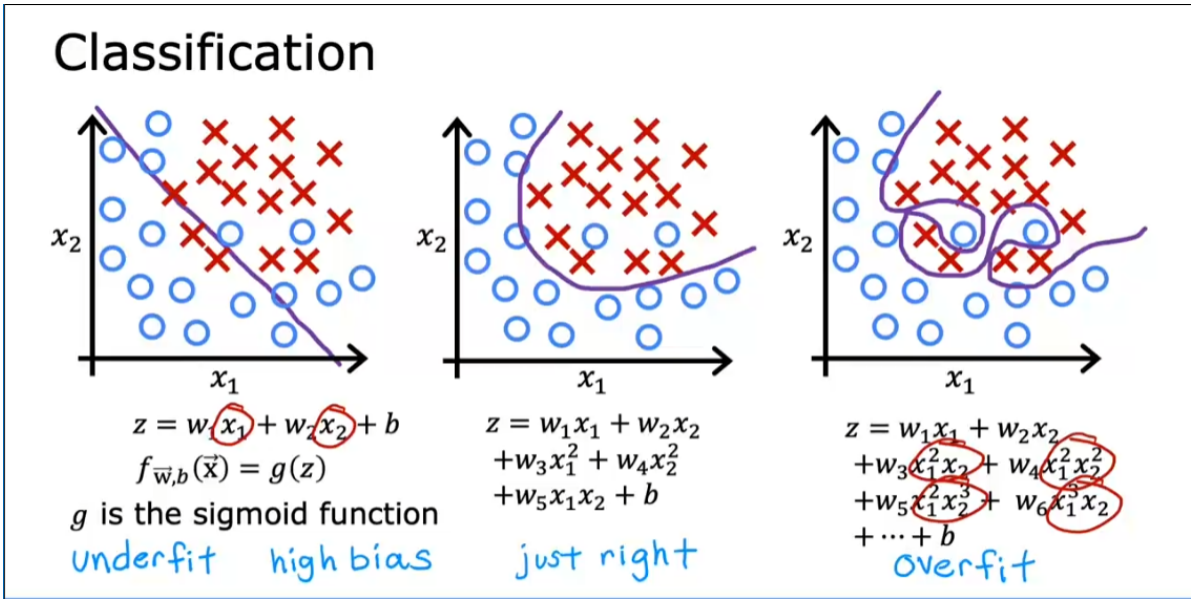
正则化

过拟合问题



- 上述图分别对应着欠拟合、拟合、过拟合
  - 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。

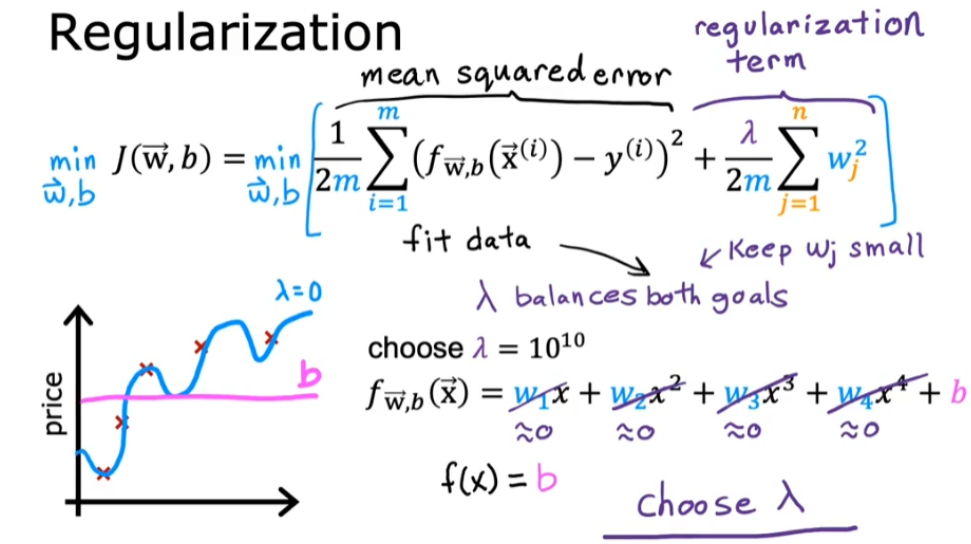


就以多项式理解，x 的次数越高，拟合的越好，但相应的预测的能力就可能变差。

- 如何解决过拟合问题
  - 特征选择。可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如 PCA, LDA），缺点是丢弃特征的同时，也丢弃了这些相应的信息。
  - 正则化。 保留所有的特征，但是减少参数的大小（magnitude），当我们有大量的特征，每个特征都对目标值有一点贡献的时候，比较有效。
  - 增加数据集。因为过拟合导致的原因就过度拟合测试数据集, 那么增加数据集就很大程度提高了泛化性了。

正则化代价函数

- 高次项导致了过拟合的产生。
- 正则化的基本方法：对高次项添加惩罚值，让高次项的系数接近于0。
- 假如我们有非常多的特征，我们并不知道其中哪些特征我们要惩罚，我们将对所有的特征进行惩罚，并且让代价函数最优化的软件来选择这些惩罚的程度。这样的结果是得到了一个较为简单的能防止过拟合问题的假设： $J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2]$
  - 其中λ又称为正则化参数（Regularization Parameter）λ>0,第二项成为正则项。 注：根据惯例，我们不对θ进行惩罚。



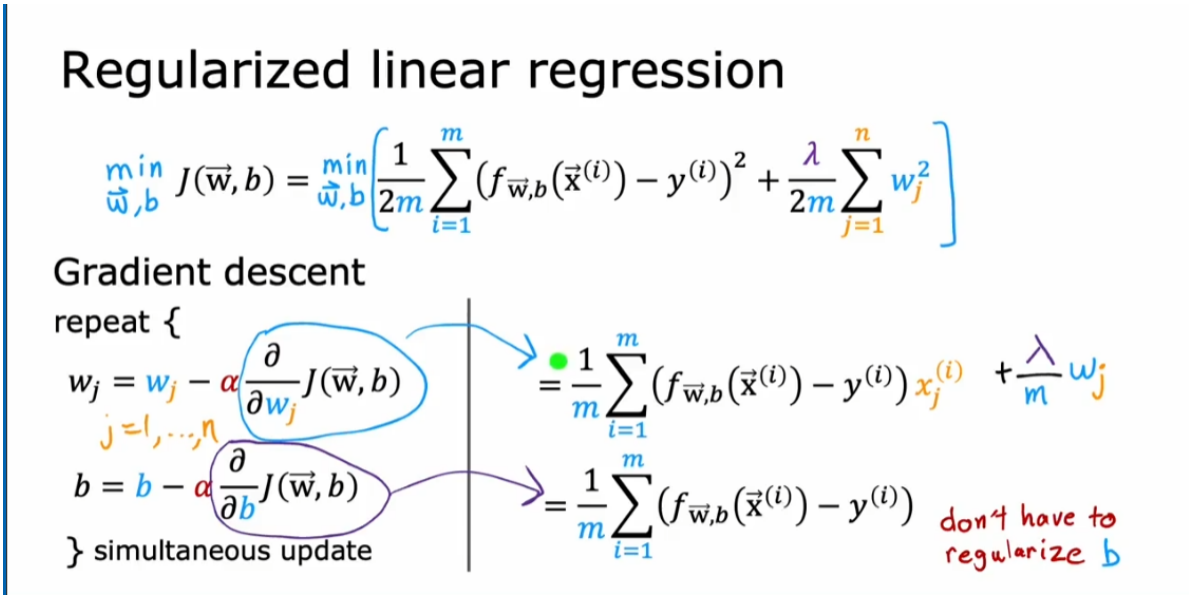
正则化参数选择

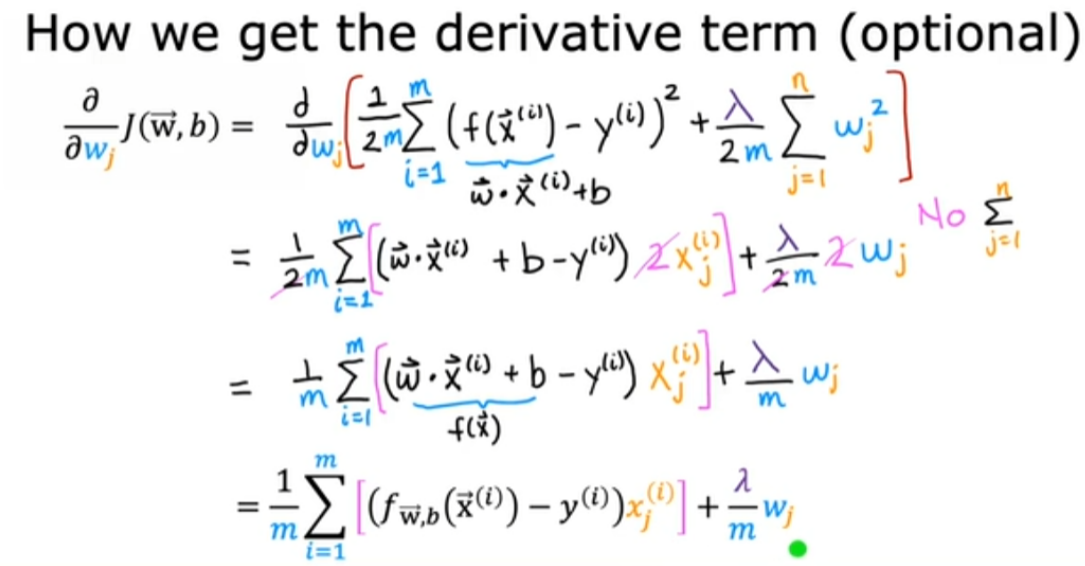
如果选择的正则化参数λ过大，则会把所有的参数都最小化了，导致模型变成 $f(x) = b$，造成欠拟合。

原因是：增加λ $\sum_{j=1}^{n} w_j^2$后，如果令λ的值很大的话，为了使 Cost Function 尽可能的小，所有的 w的值（不包括w_0）都会在一定程度上减小。 但若λ的值太大了，那么w的值（不包括w_0）都会趋近于 0，这样我们所得到的只能是一条平行于 x 轴的直线。

所以对于正则化，我们要取一个合理的λ的值，这样才能更好的应用正则化。

正则化与线性回归





Q：如何理解No求和那边

正则化与逻辑回归

参考资料

- momodel笔记
- 吴恩达课程