线性回归与逻辑回归

1. 监督式学习

部分数据集已经有正确答案，预测另一些数据集的答案。

回归问题：给出十年的房价，得到明年的房价。回归问题是监督的一种。回归的意思是预测一个连续值的输出。

分类问题：通过肿瘤的大小预测是否恶性肿瘤，给定一系列（大小，是否恶性）的数据集，通过这些数据，预测一个给定大小的肿瘤是否为恶性的概率。给定的特征可以上没有上限的，支持向量机(SVM)算法可以支持无穷多的特征。

2. 无监督学习

聚类算法是无监督学习中的一种，让算法自行对给出的数据集进行聚类。

应用场景：计算机集群中，哪些计算机趋于协同工作，可以把他们放得近一点来提高效率；好友系统中，哪些是很亲密的好友组，哪些仅是普通朋友的好友组；客户信息划分，自行将不同的客户划分到不同的市场部分。

另一种无监督学习是非聚类的（non-clustering？？），比如鸡尾酒算法可以把两个重叠的音源分别分离出来。

**初学机器学习，最好使用Octave作为开发环境。**

3. 线性回归算法

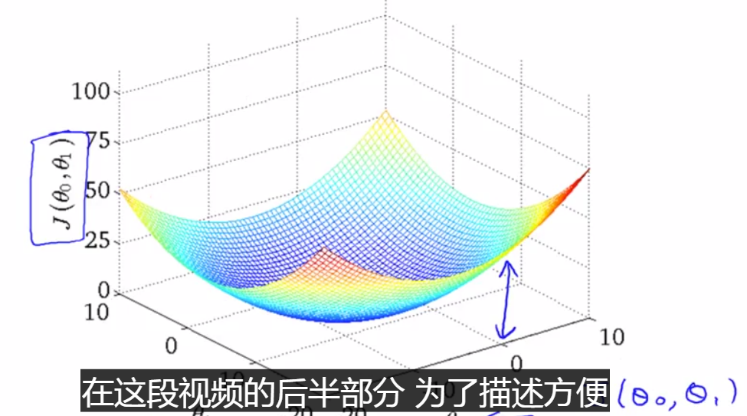
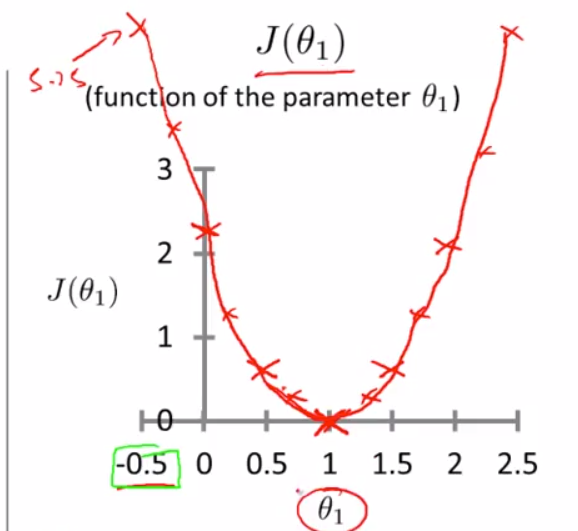
最简单的例子：通过历史数据预测特定面积的房子的价值，将历史数据输入到机器学习算法中，最终得到一个拟合出来的线性函数h(x) = kx + b

最后预测出来的直线，应该尽可能地让历史数据中的点落在或者靠近它，那么很自然的想法就是定义一个代价函数，可以量化地计算出一条直线与一群点之间的相似程度，对于这个问题，假设各个历史数据点为(xi, yi)。

那么单个点和直线的距离为；所有的点距离之和为 ；为了不受到历史数据数目的影响，因此最终的代价函数为。

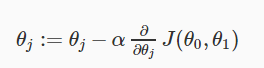
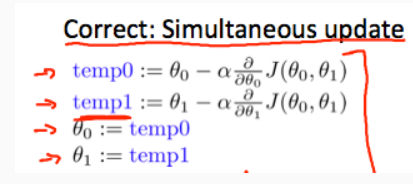
**所以问题转变为求出能够使得代价函数 最小的k,b，就可以确定预测结果h(x)**

如果假设b=0的话，h(x)=kx，那么代价函数就是关于k的二次曲线，可以很容易地找到它的最小值；如果b≠0的话，h(x)=kx+b，那么代价函数是关于k,b的曲面，找出其中的最小点就有点困难了。



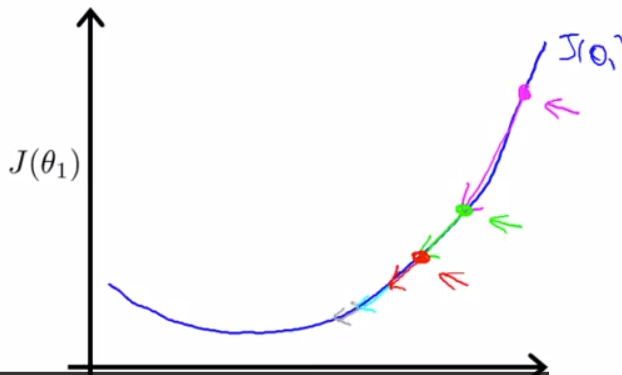
4. 使用梯度下降算法找出最优的代价函数

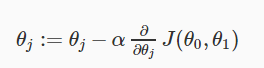
对于代价函数H(k,b)，给出一组特定的(ki,bi)，使用梯度下降算法，可以得到在(ki,bi,H(ki,bi))点，使得H(k,b)的值下降最快的方向，所以持续的进行梯度下降，可以最终得到一个局部的最小值，但不一定是绝对的最小值。

梯度下降公式：，其中α代表学习率，也可以说是每次下降幅度，而因为是以360°的范围来选择下降的方向，所以(k,b)一定是同时更新的，因此有这样的公式：，循环进行直到收敛。

5. 为什么梯度下降算法会起作用

首先考虑b=0的情况，那么代价函数就是关于k的二次曲线，那么整个梯度下降的过程如下图，会慢慢地逼近局部的最优点。

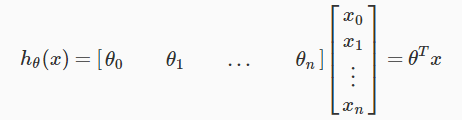


接着考虑b≠0的情况，那么代价函数是关于k,b的二元函数，，公式中分别对k和b求偏导代表着k，b各自的变化率，同时更新k,b的值，可以接近局部最小值。又因为线性回归方法的代价函数总是一个弓形的凸函数，所以使用梯度下降的方法，得到的总是全局最优解。

我觉得梯度下降算法更像是一种启发式算法，通过计算各个特征的偏导可以保证代价函数的值一定是下降的，然后就是设置一个学习率，尝试往降低代价函数值的趋势上走，最后走到局部的最低处。

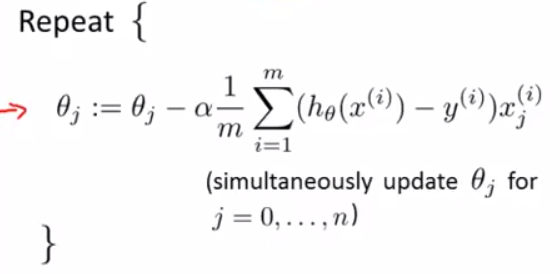
6. 多元线性回归

现在考虑特征不止一个的情况，h(x) = θ0+θ1x1+θ2x2 + …….

这样的公式可以写成矩阵的形式：

其中θ = [θ0，θ1，θ2，θ3……，θn]；x = [x0，x1，x2，x3……，xn]

仍然使用梯度下降的方法，可以简单地得到：



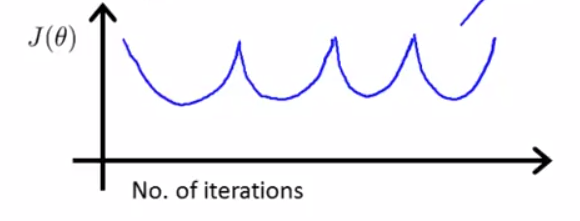
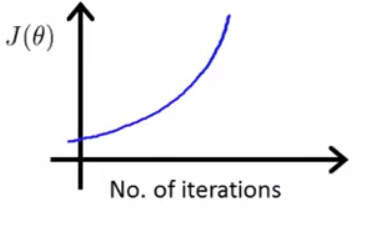
7. 梯度下降方法的使用技巧

a. 特征值范围的选取：

当多个特征之间的取值范围比例差距非常大的时候，比如x1∈[0, 5]，x2∈[0, 2000]时，通过代价函数画出的轮廓图会非常狭长，在进行梯度下降算法的时候，也会需要大量的时间才能收敛。

为了让梯度下降算法能够更快地工作，各个特征值最好是在同一个数量级范围内的，对于x2∈[0, 2000]，我们可以简单的把x2的值都除以2000，也可以使用其他的数学上的标准化方法。

b. 学习率α：没有什么好的方法可以提前确定最合适的学习率，一般来说先给定一个值，然后进行梯度下降算法，根据代价函数的值与迭代的次数形成的二维图来判断之前设定的学习率是大了，还是小了：



比如说在上面两张图的情况下，显然是学习率设置的太大了导致无法收敛，应该降低学习率α。

8. 多项式回归

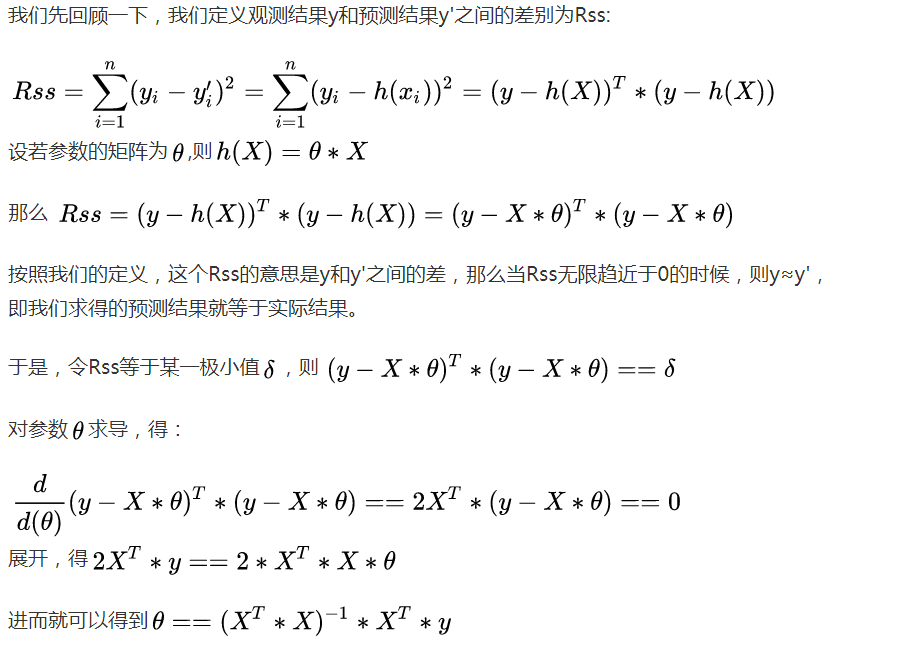
假设现在只有一个特征，如果想用二次曲线去拟合数据，那么就会有x的平方项，比如：

，这种情况下就将x^2作为第二个特征，转化成多元线性回归的方法来进行计算，同时要注意特征值范围的问题，最好统一在一个数量级里，以保证梯度下降算法正常工作。

9. 正规方程

梯度下降的算法可以让我们逐步接近最优解，而正规方程能让我们通过计算一步就得到最优解。

下图为正规方程的推导：



使用正规方程，通过矩阵的计算，一步就能得到最优解的值。

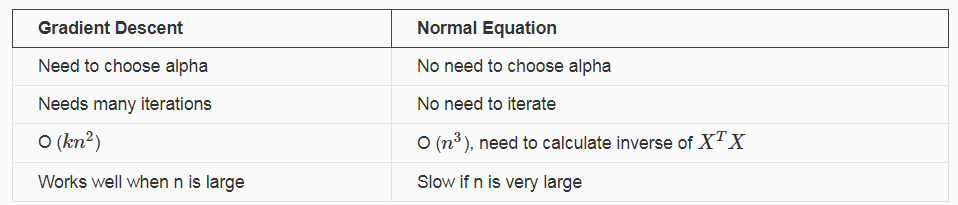
如果XTX不可逆怎么办？？有待仔细看？？？

一般来说，当XTX不可逆的时候，要么是特征的维度远大于训练的样本数目；要么是特征矩阵中含有多余的特征，比如说两个线性相关的值，被分成了两个特征。(经验之谈)

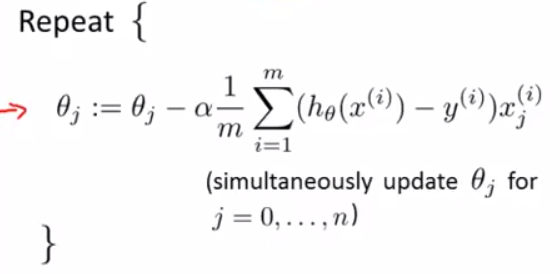
10. 线性回归中梯度下降与正规方程的选择

梯度下降：学习率α需要自行取值，在学习过程中还需要人工地调整；但是，在特征矩阵的维度特别大的情况下，只要学习率合适，梯度下降算法仍然能保证他的效率，总共要进行n\*m\*r次计算，n是特征维度，m是学习样本数目，r是迭代次数。

正规方程：由于不需要人工调整学习率，只需要进行矩阵运算就能得到最优解，因此在矩阵的维度比较小的情况下，正规方程无疑比梯度下降更有优势。但是，求逆矩阵的计算复杂度大约是O(n^3)，所以在特征向量维度特别大的情况下(n>10000)，正规方程的性能非常低下，这时梯度下降的方法就能更快的得到最优解。

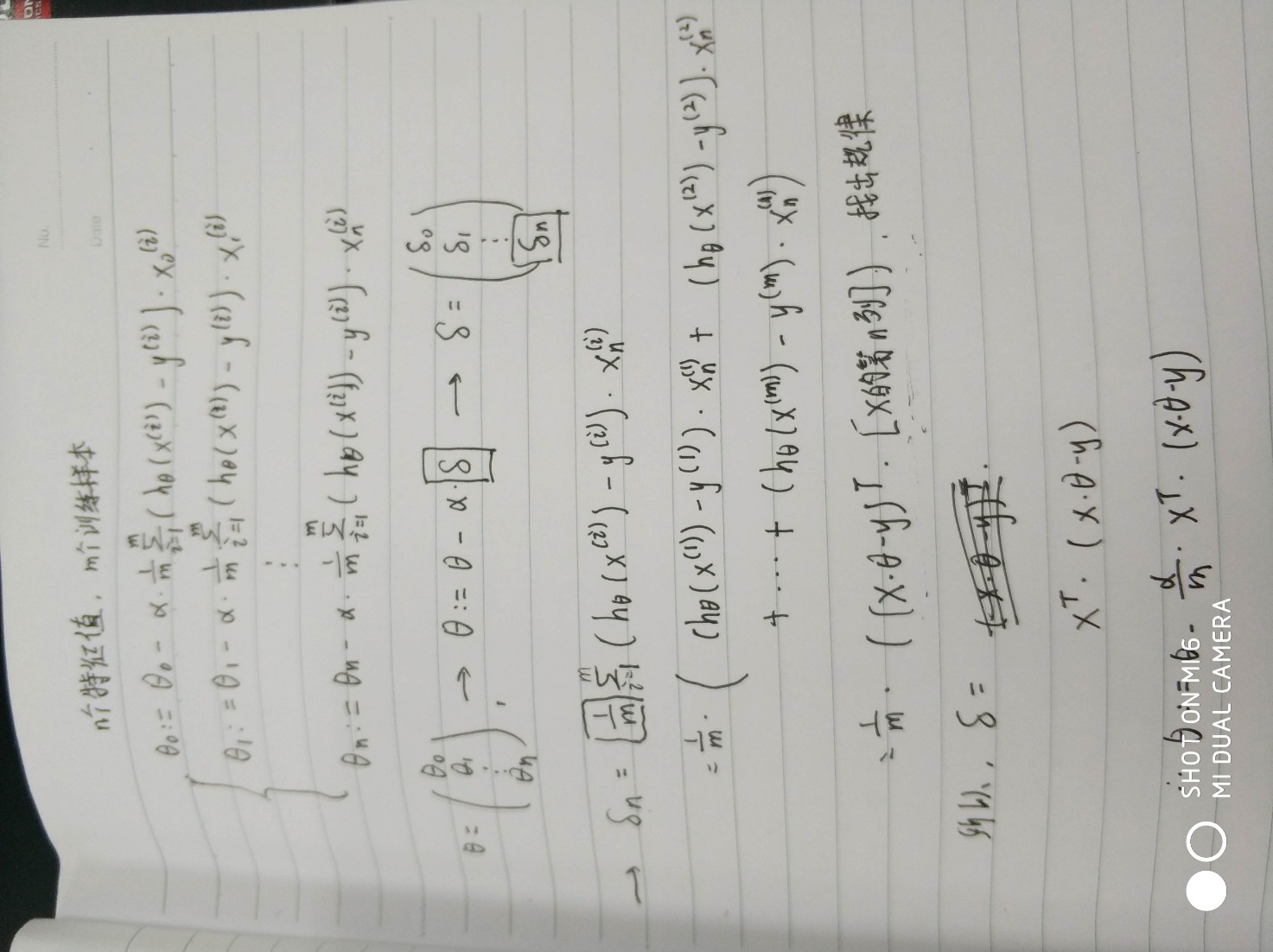


11. 向量化的思想

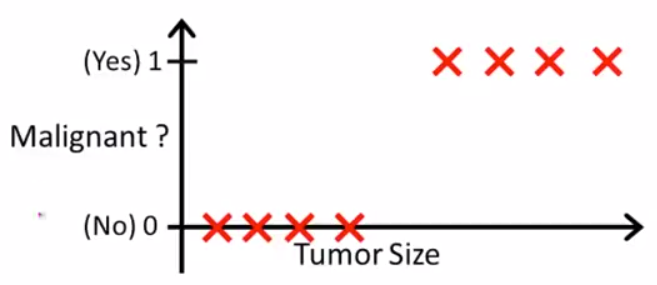


在计算上述公式时，直接对公式进行编码显得特别麻烦，如果在计算式，把X，theta，y这些数据作为矩阵考虑，那么就可以直接对矩阵进行计算，可以极大地减少代码量，也可以使用很多已有的矩阵计算类库，开发效率会提升很多。

下面是推导梯度下降的矩阵计算的过程：



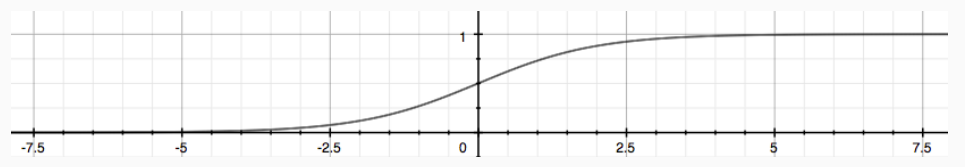
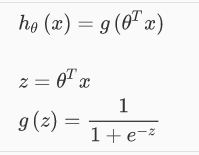
12. 线性回归不能解决的问题

对于分类的问题，比如

如果再使用线性回归的方法，就很有可能会得到在[0, 1]范围之外的值，那结果就很令人迷惑了，所以我们需要另一个算法，能保证输出的值就是0或者1，或者输出的值在[0, 1]这个范围内。

13. 逻辑回归

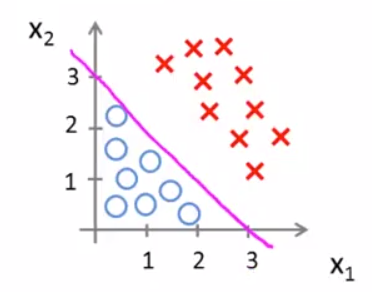
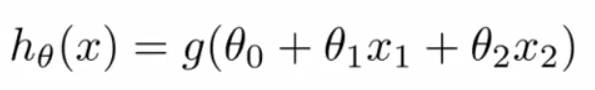
首先考虑二分类问题：为了保证函数的取值在[0, 1]之间，在线性回归的基础上，再套一层逻辑函数



g(z)作为拟合的函数，输出的值的含义是：在某个给出的条件下，该数据属于1这个类别的概率。

14. 逻辑回归在做什么？

逻辑回归的得到的结果一般是这样的：

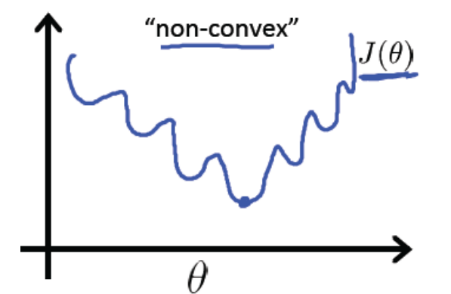


这里拟合得到的θ0+θ1x1+θ2x2，被叫做决策边界，也就是上图中紫色的分界线，然后使用逻辑函数将h(x)的函数值压缩到[0, 1]的范围内，(x1, x2)的点在越上方的地方，那么他的分类属于1的概率就越大；(x1, x2)的点在越下方的地方，他的分类属于1的概率就越小。

15. 逻辑回归的代价函数

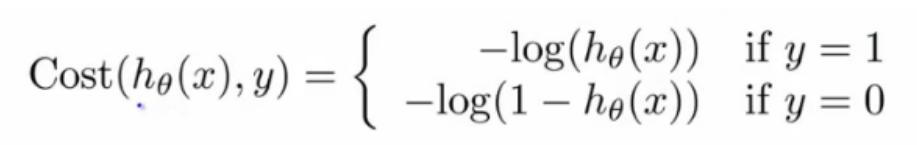
现在我们确定了想要拟合什么样的函数h(x)，那么如同线性回归一样，我们需要一个代价函数来评判某一种拟合结果的代价，然后选出代价最小的结果。

首先，很自然地使用原来的线性回归的代价函数，但是这种情况下，发现最终的代价函数与θ的关系图：



J(θ)并非是凸函数，如果在它的基础上，使用梯度下降的方法，很难得到令人满意的最优值。

所以我们需要选择另一个代价函数来准确地评判每一种拟合，这里我们选择“对数似然损失函数”，来作为新的代价函数：

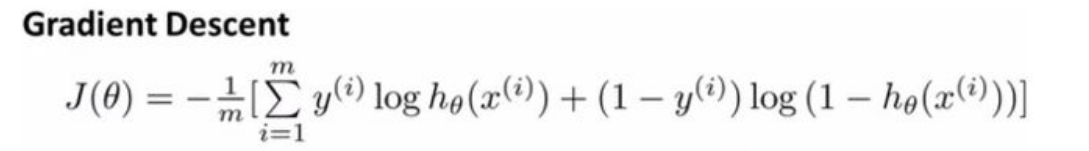


对于这个代价函数，我的理解是这样的：h(x)代表的是，当特征为X的时，该数据属于类别1的概率。

那么对于已经做过标记的数据，比如：在类别为1的情况下，有h(x)使得数据类别为1的概率是0.7，实际上概率是100%，因为y=1，所以这里损失的就是0.3，代表有0.3的不准确度。

那么对于多个给定的数据，他们的事件同时成立的概率应该是概率的乘积，而代价函数一般都是每个数据的代价之和，所以使用-log，来保证总和是乘积，以及在预测离实际值偏差越大的情况下，产生的代价也越大。

确定了如何计算代价后，就得到了逻辑回归的代价函数J（θ）：



使用梯度下降的算法后，实际上对于θ的更新和线性回归是一样的，只是假设函数h(x)有所不同。

通过如同线性回归中相似的推导过程，我们可以得到如下的矩阵形式的运算进行每一步的梯度下降：

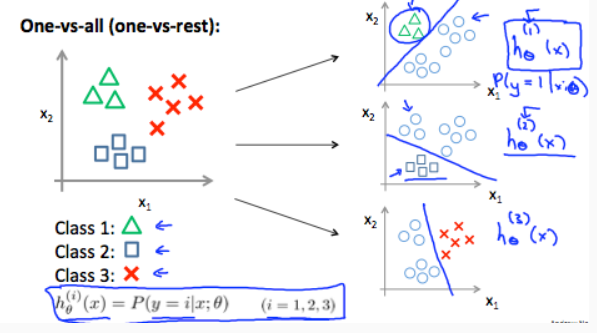
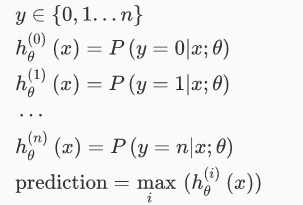


16. 优化使用梯度下降进行逻辑回归的算法

相比于简单的梯度下降算法，有很多在它上面进行优化的算法可供开发者选择，比如共轭梯度，BFGS,L-BFGS算法等等，这里并不解释这些算法的具体实现（它们做的其中一件事就是自动合理地选择学习率α，但实际上它们非常复杂）。在大规模的机器学习项目中，最好使用这些优化的算法，它们的收敛速度往往远超普通的梯度下降算法，在Octave中，也有现成的函数库可以供我们使用。

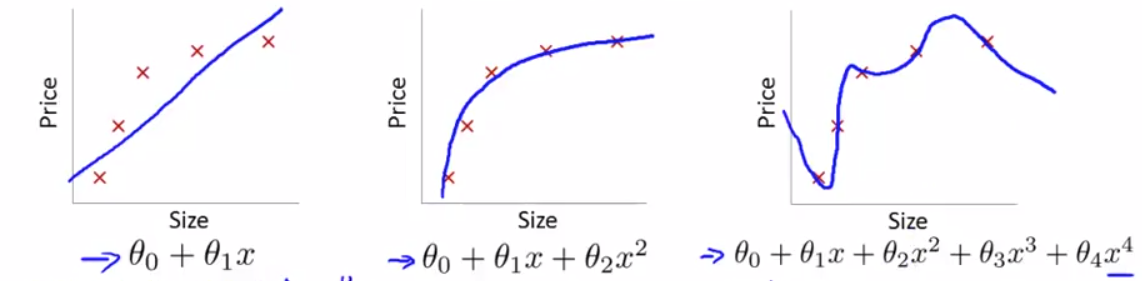
17. 使用逻辑回归解决多分类问题

对于多分类问题，实际上就是把整个问题分成若干个二分类问题，比如：训练数据集将数据分为1~n类，对于一个新的数据，可以通过二分类的逻辑回归计算出，分别计算出它属于1,2,3…类的概率，然后选取其中的最大值，决定这个新的数据属于哪一类。



18. 过度拟合

在进行回归预测时，如果训练样本的特征变量太多，同时训练数据的数量又不能满足要求，就会发生过度拟合的问题。

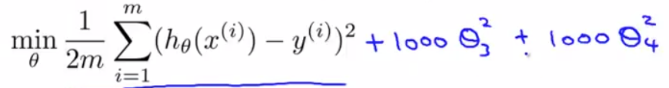
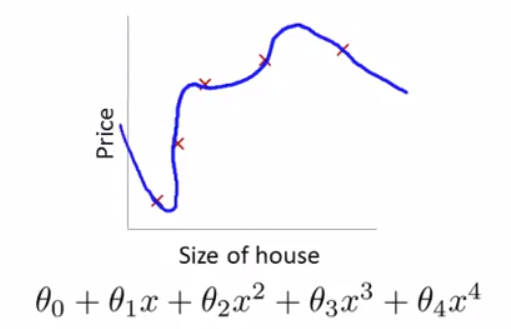


在拟合图像时，使用大量的多项式，可以达到准确匹配每一个数据的目的（如图3），就从代价函数上来看，图3似乎是更优的拟合结果，但是从预测房价的场景来看，图3这样的扭曲的曲线显然不符合常识。所以，过拟合就是说为了强行匹配每一个训练样本，使用大量的多项式，最终得出了复杂的，明显不符合常理的、缺少一般性的拟合结果。

解决过拟合的方法有两种：一种是手工地减少特征变量的数据，或者使用相关的算法，自动地减少变量数目；另一种是使用正则化的方式，对各个特征的取值等作出调整。

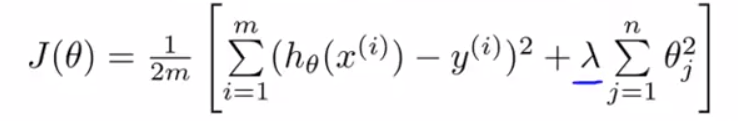
19. 正则化

过拟合的例子中，一般都是使用了太多的特征变量，在拟合的函数中有太多的多项式，对于下图中过拟合的例子，可以使用正则化的方法来避免过拟合：



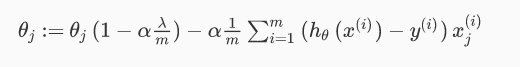
如图，在代价函数后加上1000倍的θ3和1000倍的θ4，这样的话，在最终的结果中，θ3和θ4都会趋向于0，那么几乎等于消除了θ3和θ4这两个多项式，也就不会有过拟合的情况了。

更一般的，我们其实并不知道需要将哪些特征正则化，减小他们的影响，所以通常会对所有的参数进行惩罚，以此来达到避免过拟合的效果。（多项式的系数越小，整个曲线就会越平滑，纵观各种过拟合的图象，都是曲线非常扭曲的，所以使曲线平滑可以避免过拟合的问题），由此我们可以得到新的代价函数：



20. 正则化线性回归

使用我们之前得到的新的代价函数，进行梯度下降，可以得到这样的更新公式：



直观地可以看到，改变就是在每次更新θj的时候，都会将原来的θj乘以一个系数，所以，如果惩罚项的系数设置的太大的话，会导致整个梯度下降无法工作。一般来说每次更新时，会将θj乘以一个0~1之间的系数，结果就是在最终收敛时，θj会变得比原来更小。

对于正规方程来说，引入正则化后，可以得到这样的公式（正则化的逻辑回归也是同理）：

