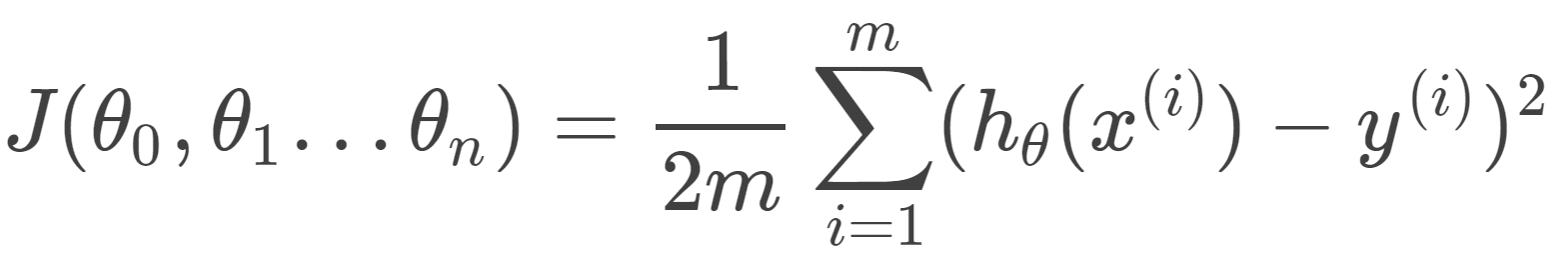
# 吴恩达机器学习总结

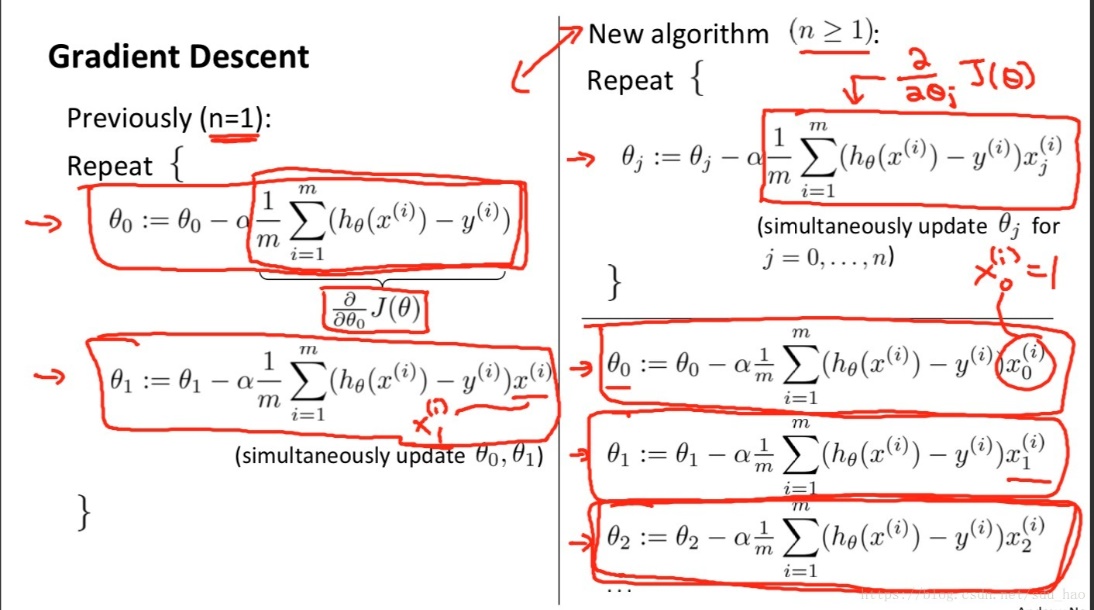
第一、二周

线性回归：单元线性回归,多元线性回归：

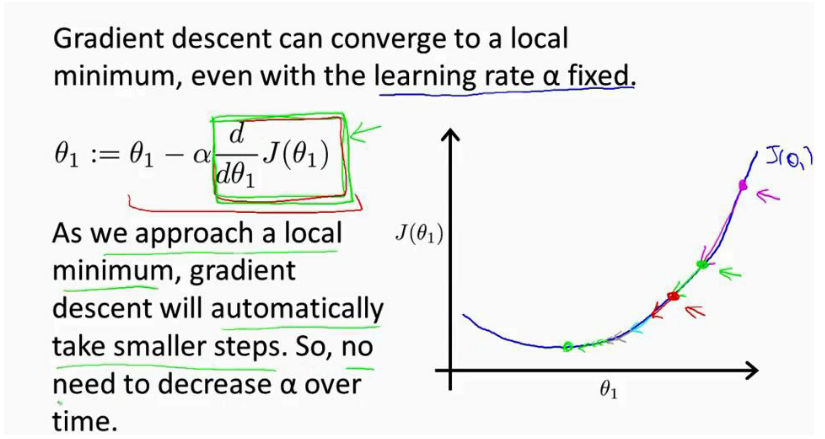


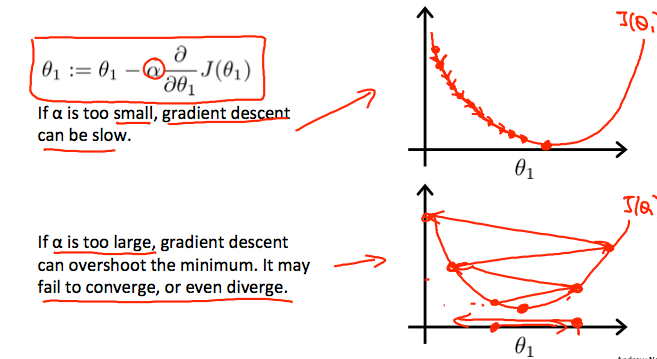
上述公式是平方误差代价函数，它通过目的函数和各个实际值误差的平方建立新的函数。为了使这个值不受个别极端数据影响而产生巨大波动，采用类似平方差再取二分之一的方式来减少个别数据的影响，其中预测函数h中引入的目的是用来表示未被自变量解释且长期存在（非随机）的部分，即信息残留。

梯度下降：每次梯度下降要同步更新每个θ，下图中α是学习速率



α太小梯度下降速度会很慢，α太大会跨过最小值点，甚至无法收敛，见下方第二张图，解决方法是乘偏导数：





特征缩放：

多元线性回归中由于存在多个特征，若能保证这些特征都在一个相近、的范围(即保证不同特征的取值在相近的范围内)这种情况下梯度下降法的收敛速度会很快。一般采用平均值归一化对特征进行缩放:

其中为输入变量第i个特征，为输入变量第i个特征所有取值的平均值，为输入变量第i个特征的取值范围(最大取值减去最小取值），注意i，第0个特征是用来解决信息残留添加的，默认取值为1，不用归一化。

学习率：

两种判断梯度下降法是否收敛的方法

1.做出代价函数随参数更新迭代次数的变化曲线

如果梯度下降法正常工作的话，每次迭代后，代价函数都会下降。

这条曲线作用就是判断梯度下降法是否收敛，如上图所示，当迭代次数超过300次，在300-400之间时，我们会发现曲线比较平坦，也就是说此时并未下降多少，那么说明此时梯度下降法基本收敛。

2.自动收敛测试

若在某次迭代后，代价函数的变化范围小于某个阈值，是一个很小的值，一般取，则认为梯度下降法基本收敛。

实际上多用方法1，而不是方法2.

1. 方法2很难确定一个合适的阈值。
2. 方法1不仅能够判断梯度下降法是否有效，而且还能在梯度下降法不正常工作时，提前警告。

特征和多项式回归：

（1）创造新特征：

拿到训练样本时，我们不一定要使用所有原始特征，可以对原始特征进行一定的处理，创造出一个新的特征。他可以是两个原始特征之间相乘相除等等。

（2）多项式回归：

比如房价预测，可以一直用线性函数来拟合，也可以尝试用曲线来拟合，比如采用二次或三次多项式来拟合等等。

正规方程：

利用正规方程来计算预测函数的参数：

一般来说，梯度下降不仅适用于线性回归也适用于其他机器学习算法；而正规方程法一般仅适用于线性回归问题。

梯度下降法：

优点：

当特征数n非常大的时仍然适用，一般当n大于时，使用梯度下降法更好。

缺点：

（1）需要选择一个合适的学习率

（2）需要很多循环迭代来更新参数

（3）当各个特征取值范围差异较大时，需要归一化，以加快收敛速度

正规方程法：

优点：

（1）不需要选择一个合适的学习率

（2）不需要迭代

（3）不需要归一化特征

缺点：

需要计算，是一个n×n方阵，计算其逆矩阵的复杂度大约是,也就是说，当n非常大时，求解速度非常慢，一般适用于n小于时。

正规方程的可逆性：

正规方程中当不可逆即是奇异矩阵，首先，不可逆的原因大致有以下两种：

1. 存在冗余特征，不同特征间线性相关(即不同特征存在线性关系，比如两个特征都表示面积，一个使用平方米为单位，另一个使用平方分米为单位，我们知道二者存在100的倍数关系)。根据线性代数的知识，若矩阵中的列向量线性相关，则矩阵不可逆。此时可以删除一个线性相关特征。
2. 特征太多而训练样本太少，即m远小于n，此时可删除一些特征或者使用正则化。

第三周

分类问题：

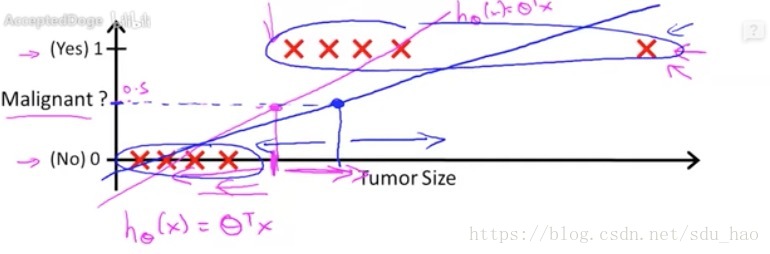
判断邮件是否为垃圾邮件

判断一个在线交易是否为诈骗

判断肿瘤是良性的还是恶性的

以上三个例子都是二分类问题，它们的输出变量y只有两个取值{0,1}，0代表负类，1代表正类

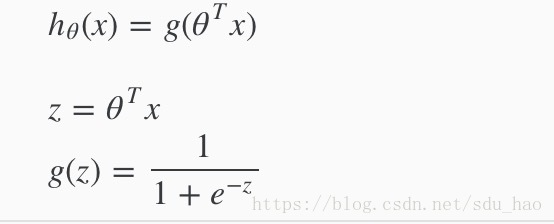
一般用线性回归无法很好的解决二分类问题，见下图：



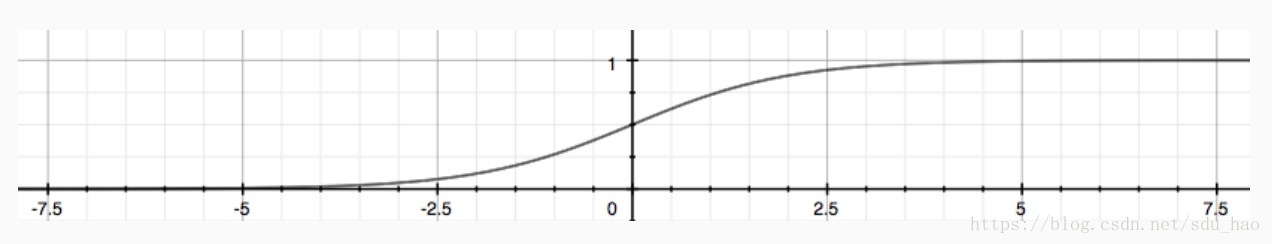
此外，当你使用线性回归解决二分类问题时，远大于1，即使训练样本标签全为0，所有接下来介绍逻辑回归算法

逻辑回归：

逻辑回归算法函数表达式：

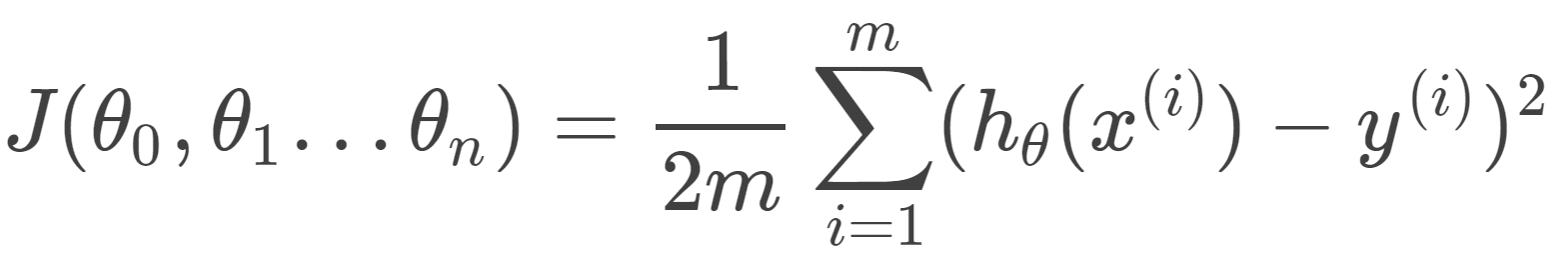


其中g(z)为sigmoid函数，也叫logistic函数，图像如下：



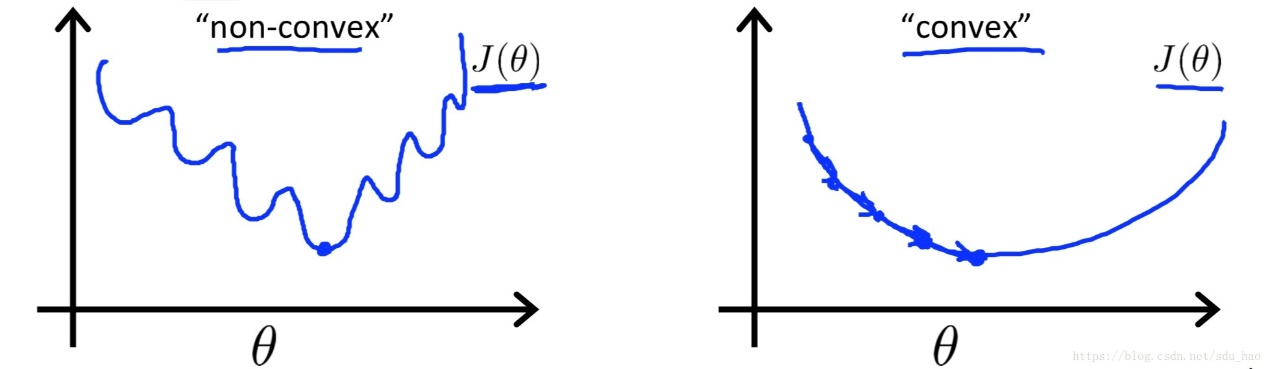
逻辑回归代价函数：

首先回顾线性回归代价函数：



Cost(，y) = 1/2( - y)^2

可以看到之前求解线性回归模型参数时，我们使用的代价函数为平方代价函数，然而逻辑回归的代价函数不可这样使用，见下图：



图中我们绘制两幅代价函数J()随模型参数的变化曲线，左图我们称为非凸函数，很明显我们可以发现很多局部最小值，之前有提到过，梯度下降法只能找到局部最优解，不保证找到全局最优解，也就是说如果代价函数随参数的变化曲线是非凸函数的话，我们不能保证得到全局最小值，我们希望的是右面这种凸函数的情况，有利于我们找到全局最小值。

因此定义代价函数时，应满足其随参数的变化是一个凸函数，不同于线性回归，逻辑回归引入sigmoid函数，我们知道这是一个非线性函数,所以如果采用和线性回归一样的平方代价函数，此时会得到一个非凸函数，因此，我们要重新定义逻辑回归的代价函数，使其为凸函数，便于我们得到全局最小值。

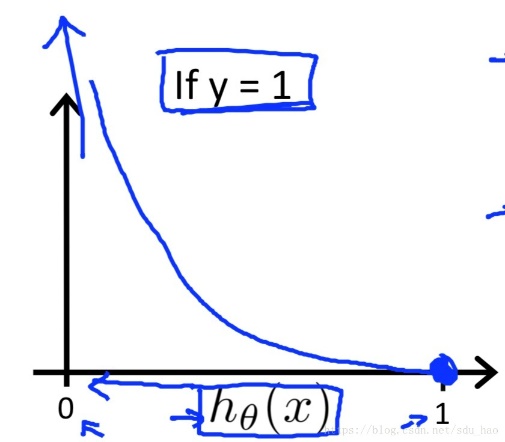
逻辑回归代价函数

Cost(，y) =

如上所示，逻辑回归代价函数是一个分段函数。

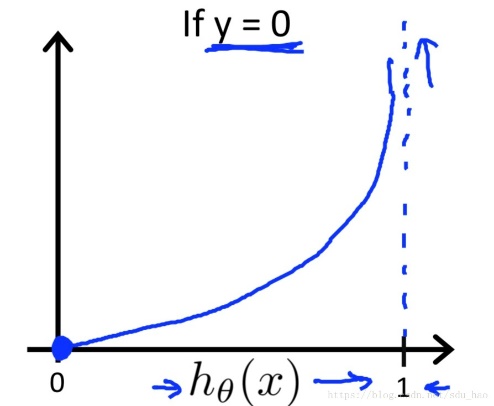
当训练样本的标签（输出变量）y = 1时：

Cost(，y) = ，绘制图像如下：



当 = 1，y = 1时，此时代价函数为0，说明此时的参数拟合最优；但当 = 0，y = 1 时，此时代价函数趋近于∞，说明此时参数拟合最差，这时我们以一个最大的Cost来惩罚我们的学习算法，继续迭代求解最优的参数。当训练样本的标签(输出变量)y = 0时，绘制图像如下：

Cost(，y)



当 = 0，y = 0.此时代价函数为0，说明此时的参数拟合最优；但当 = 1，y = 0时，此时代价函数趋近于∞，说明此时参数拟合很差，这时我们以一个最大的Cost惩罚我们的学习算法，继续迭代求解最优参数。

我们将分段的代价函数合并得到如下公式：

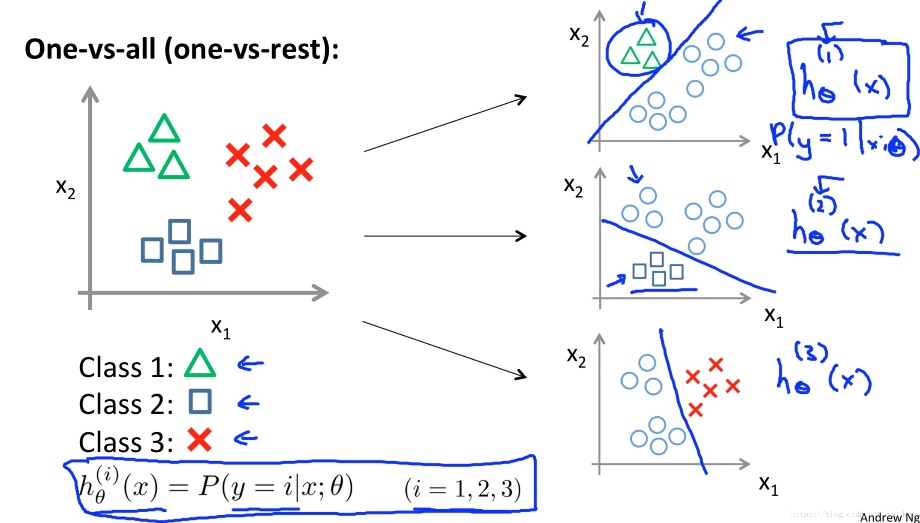
Cost() = -ylog()–(1 - y)log(1 - )

所以简化后的代价函数为：

其最优化仍可通过梯度下降来求得，当然也有高级优化方法，例如Conjugate gradient、BFGS、L-BFGS。这几种算法相较于梯度下降的优点是不需要人工选择一个合适的学习率，而且通常比梯度下降更快，缺点是更复杂。

多分类问题：

比如一个三分类问题，见下图，它可以被看作3个二分类问题。首先把三角看作正类，其余两类统一看作负类，定义一个逻辑回归的分类器,通过这个训练集，利用相关优化算法，最小化代价函数，求解出模型参数，得到分类器；其余的两个2分类问题也一样。这样我们会得到三个分类器，其决策边界如右边3幅图的蓝线所示。接下来对于一个新的输入变量x，分别代入训练好的三个分类器，得到它在三个分类器中属于其正类的概率，取最大的概率，那么这个新的输入变量x就属于这个类别。



过拟合：

如果我们有很多特征，学习到的假设函数在训练集上拟合效果非常好，代价函数的值趋近于0,但该模型无法泛化到新的样本数据中

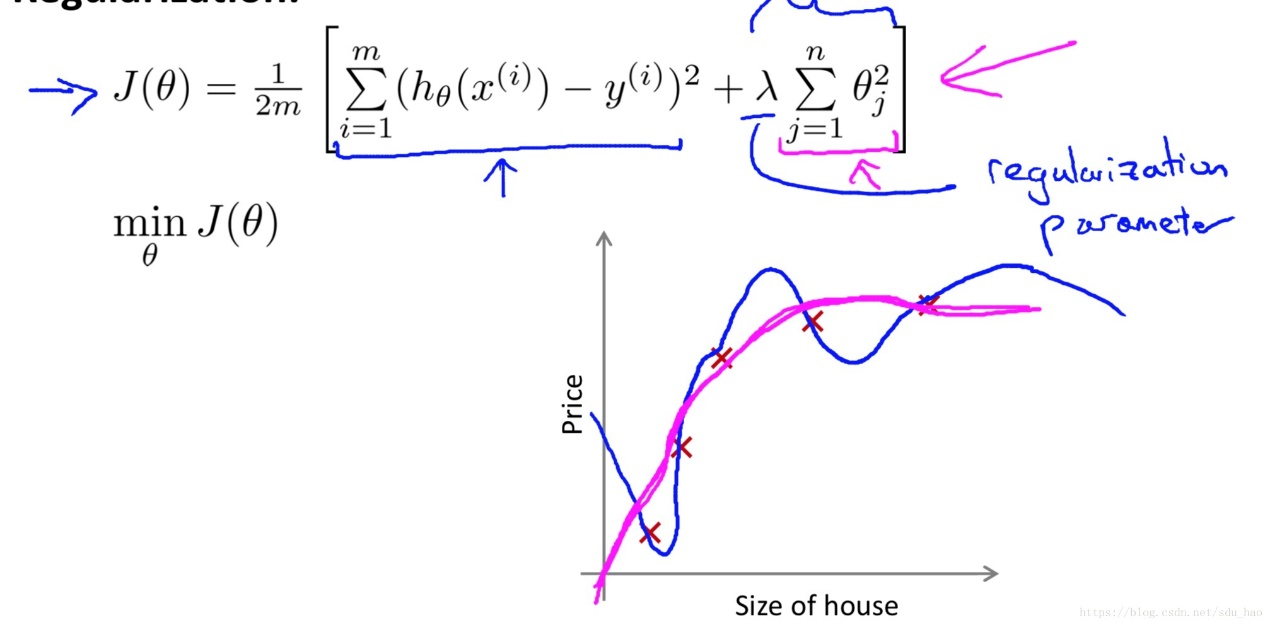
泛化：假设模型能够应用到新样本的能力。

解决过拟合的方法：

1. 减少输入特征的数量：可以人工选择一些比较重要的特征，或者使用模型特征选择算法，自动选择一些特征
2. 正则化：保留所有输入特征，但减少参数的值，当我们有很多输入特征且每个特征都对预测输出值有一些影响时，效果会很好

代价函数（正则化）

正则化思想：使假设模型的参数,,…的值变小，从而让假设模型更加简单，平滑，降低过拟合的风险。和上例不同的是，我们并不知道要减小哪个参数，给哪个参数添加惩罚项。实际上，在做正则化时，是对所有的参数(除)添加惩罚项。



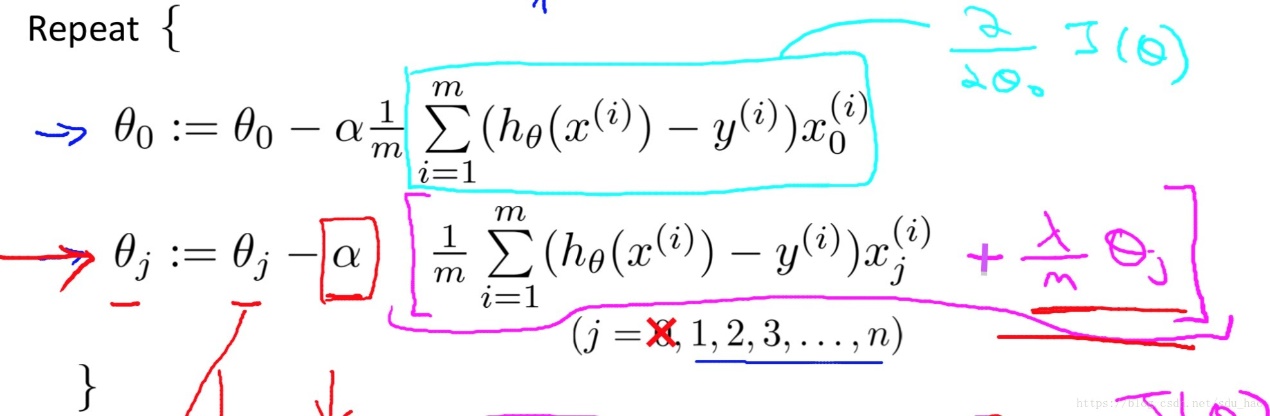
此时的代价函数由两部分组成，第一部分使模型在训练集上的拟合效果尽可能的好；第二部分是惩罚项，它使得模型参数值减小，避免前一部分过拟合。二者做一个平衡。然后就是利用优化算法，最小化这个新的代价函数，得到最优的一组参数。图中的蓝线和红线分别代表正则化前/后的拟合效果。

正则化系数\lambda

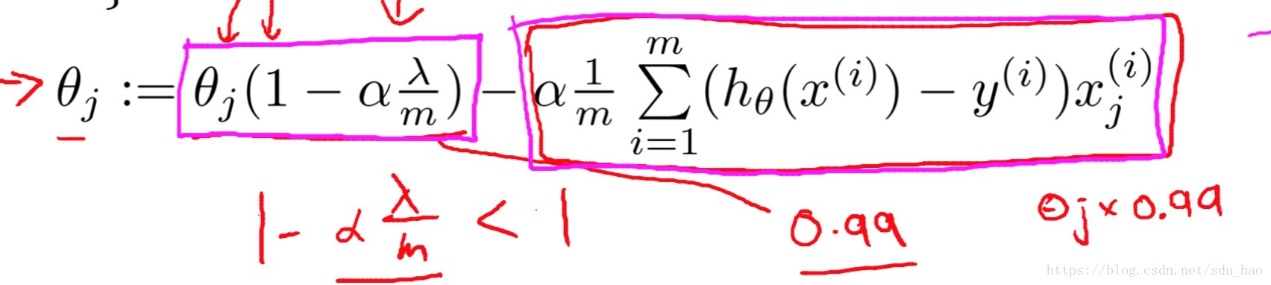
如果\lambda过大时，对参数的惩罚力度过大，此时得到的模型参数基本上是趋于0，那么假设函数,在下图的这个例子中，我们就只能得到一个水平的直线，很显然这是一个欠拟合的例子。因此，为了发挥正则化应有的效果，我们必须要选择一个合适的\lambda值。

对于线性回归模型我们讲过两种优化求参数的方法，一种是梯度下降法，另一种是正规方程法。现在让我们看一下，代价函数加入正则化惩罚项后，这两种方法有什么变化。

梯度下降法：



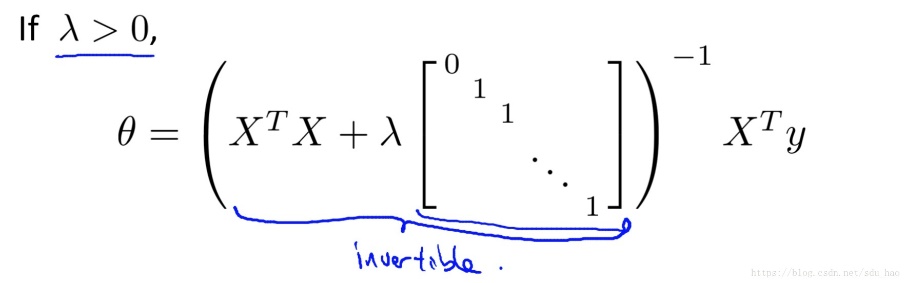
由于没有惩罚项，所以单独列出来，其梯度形式和原来没有区别；由于,…有惩罚项，所以它们的梯度是在原来的基础上加上惩罚项产生的梯度。接下来对,…的迭代形式进行整理，将合并同类项，得到下式：



式子的前一部分可以看作是先让乘以一个比1小的值，先减小一下，再减去后一部分，而后一部分就是原来不加惩罚项时，参数的梯度。

正规方程法：

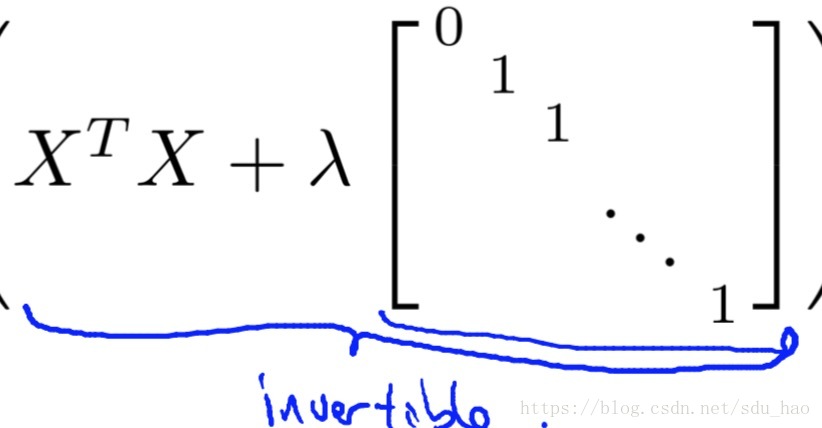
直接加入惩罚项的新代价函数求偏导，使其值为0，求解出模型的最优参数。此时得到的最优参数值为：



其中，矩阵X（m\*（n+1）维）由所有训练样本的输入变量构成，向量y（m维）为所有训练样本的输出变量：



对正规方程使用了正则化，它不仅能够避免过拟合，而且它还能保证下图中的部分一定是可逆的



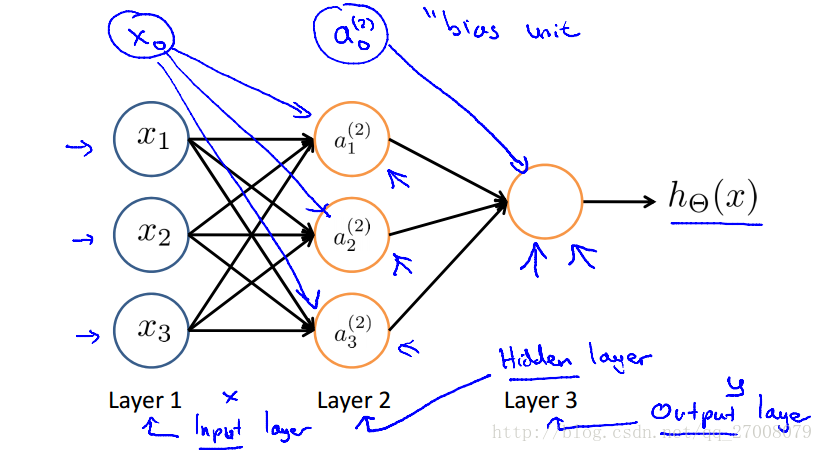
第四、五周

提出神经网络的动机

线性回归、逻辑回归面对一些多特征问题并不能很好的解决，这种时候需要用到多项式回归（非线性的），这种函数画出的曲线可以有任意角度。但是这种函数会因为特征量的增多导致二次项数的剧增。 比如在图像识别中，一个50×50像素的图片，拥有的特征量为2500，那么它的二次项数为2500×2500/2,大约为3百万个，在这种情况下，神经网络在1970年被提出。

神经网络

神经网络是由一个个的神经元组成的网络，下图为一个三层神经网络模型。第一层为输入层，第二层为隐藏层，第三层为输出层。 每条边上有一个权值θ。



:第j层单元i的激励

:第j层到第j+1层单元的权值矩阵

若第j层单元数为，第j+1层单元数为，则是×(+1)的矩阵，其中+1来自偏置节点的中的加法和.

代价函数和反向传播

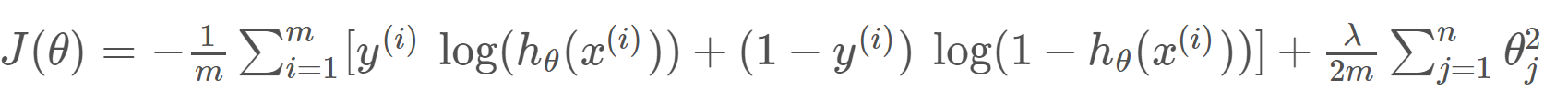
首先定义变量：

L = 网络中的总层数

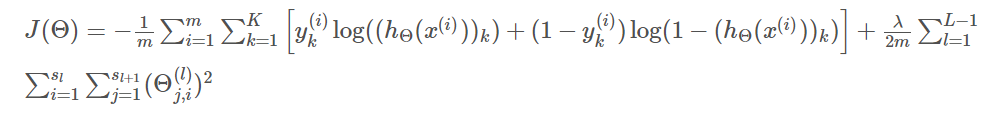
Sl = 第l层的单位数量（不包括偏差单位）

K = 输出单元/类的数量

首先，逻辑回归正则化的成本函数是：



而在神经网络中，我们可能有多个输出节点。我们把表示为导致第k个输出的假设。我们的神经网络的成本函数将会是我们用于逻辑回归的一个综合泛化。神经网络的代价函数为：

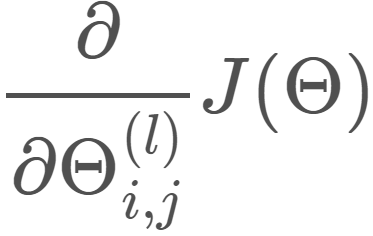


在方程的第一部分，在方括号之前，我们有一个额外的嵌套总和，表示输出节点的数量。

在正则化部分，在方括号后面，我们必须考虑多个theta矩阵。当前theta矩阵中的列数等于当前图层中的节点数（包括偏置单元）。在我们当前theta矩阵中的行数等于下一层中的节点数（不包括偏置单元）。与之前的逻辑回归一样，我们对每一项进行平方。

误差反向传播

就像在逻辑回归和线性回归中使用梯度下降，我们最小化代价函数J，所以通过theta中的一组最优化参数来最小化代价函数J。寻找最小的参数，需要使用梯度下降法；而梯度下降法最重要的是计算梯度



我们使用反向传播算法来计算偏导数：

1. 设:=
2. 执行前向传播以计算（ = 1,2,3…L）
3. 使用计算最后一层的误差： =

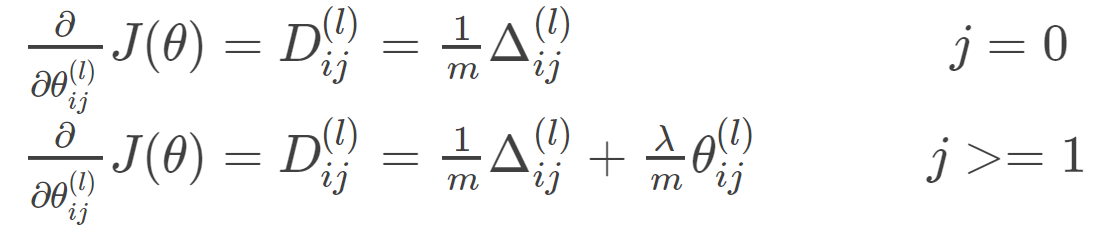
其中L是我们的总层数，是最后一层激活单元的输出向量。

所有我们最后一层的“误差值”就是我们最后一层的实际结果和y中的正确输出的差别。为了得到最后一层之前的图层的增量值，我们可以使用一个从右到左的方程式：

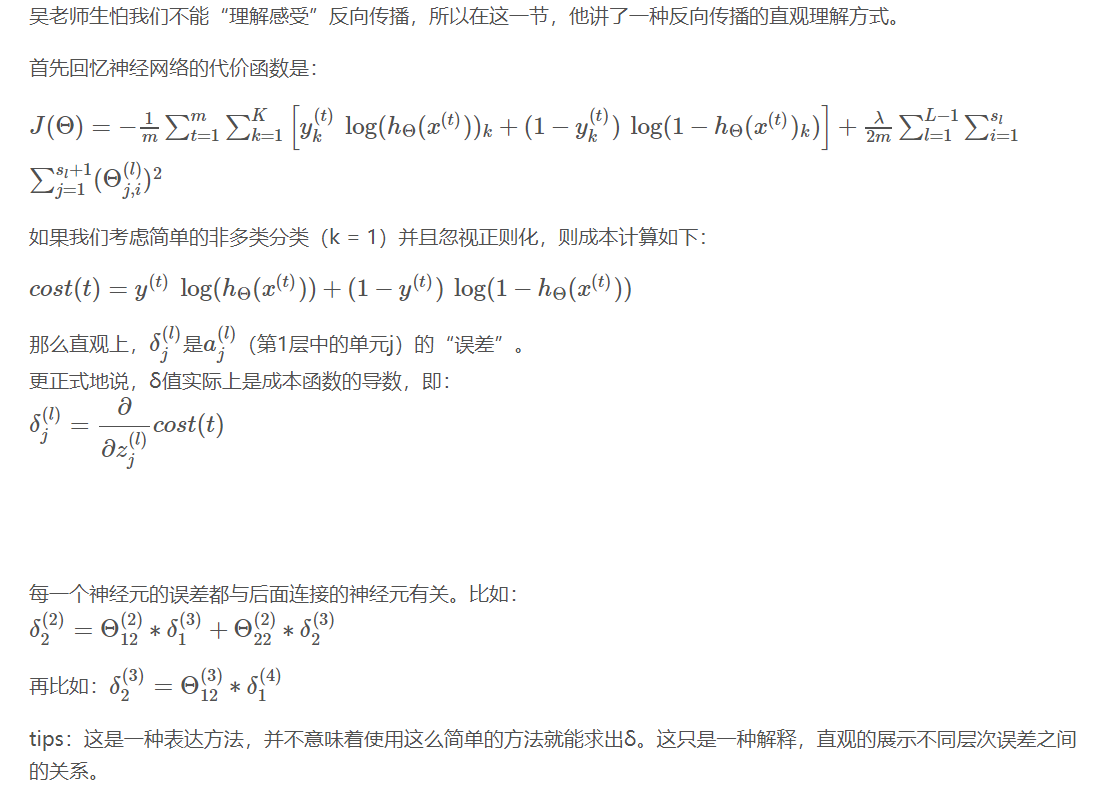
计算之前层次的误差：

使用公式： = (().\*.\*(1-)

反向传播计算所有后计算，其中： = + ,所以偏导数为(考虑正则化项)：



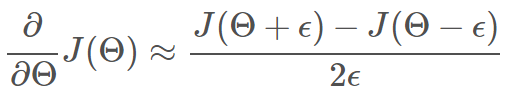
反向传播的理解（见下图）



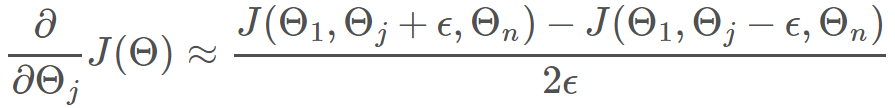


梯度检验

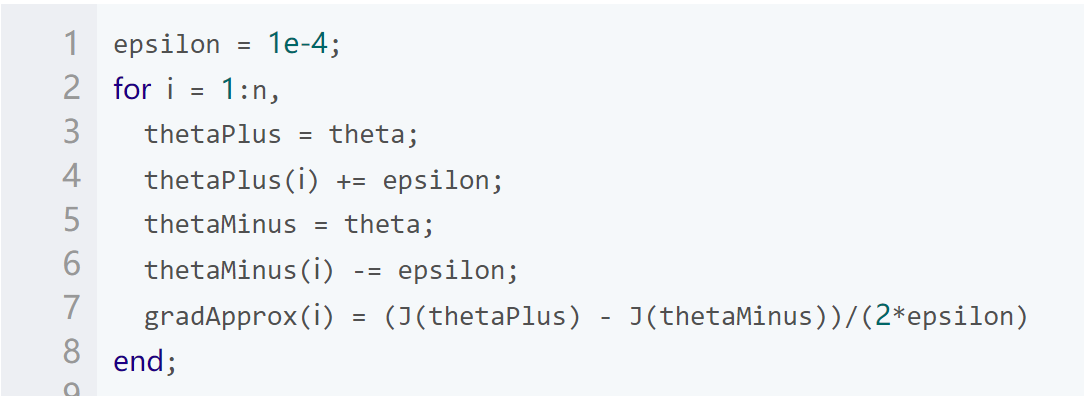
渐变检查将确保我们的反向传播按预期工作。可以用下式近似我们的成本函数的导数



使用多个theta矩阵，我们可以如下近似关于 的导数：



为保证数学运算正确ε（ε）的一个很小的值，比如εϵ 。   
但如果ε的值太小，我们最终会出现数值问题。因此，我们只是在Θj矩阵中加上或减去ε。在octave我们可以做到这一点，如下所示：

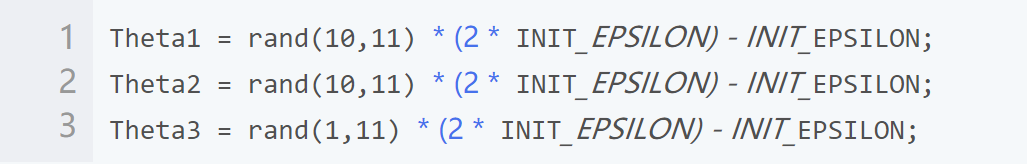


一旦验证了反向传播算法是正确的，就不需要再次计算gradApprox。因为计算gradApprox的代码可能非常慢。

随机初始化参数

将所有的权重初始化为0不适用于神经网络。那样反向传播时，所有节点将重复更新为相同的值。

相反，我们可以使用以下方法随机初始化我们的Θ矩阵的权重：



因此，我们将每个初始化为[-ε,ε]之间的随机值，使用上面公式保证我们得到所需的界限。

神经网络总结

首先，选择一个网络体系结构;选择你的神经网络的布局，包括每层中有多少个隐藏单元，以及你想要的总共有多少层。一些经验：

- 输入单元的数量=特征的维度x（i）   
- 输出单元的数量=类的数量y   
- 每层隐藏单元的数量=通常越多越好（必须与计算成本平衡，因为随着更多隐藏单元的增加而增加）   
- 默认值：1个隐藏层。如果您有多个隐藏层，则建议您在每个隐藏层中具有相同数量的单元。

模型的训练

模型的训练遵照以下步骤：

1. 随机初始化权重

2. 实现向前传播以获得任何x（i）的hΘ(x（i）)，

3. 实施成本函数

4. 实施反向传播以计算偏导数

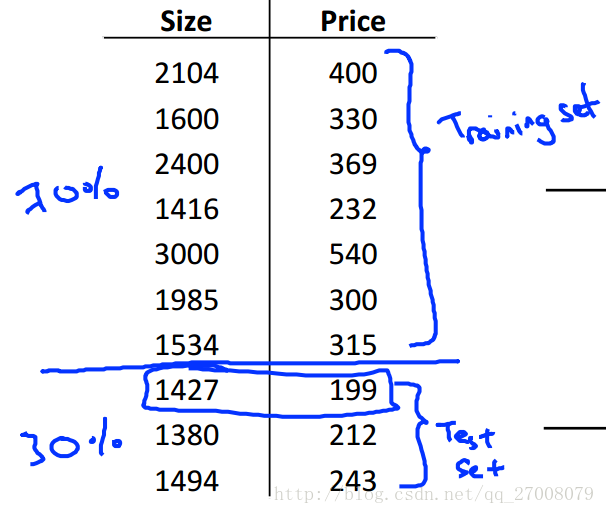
- 循环每个训练的例子for i = 1:m

5. 使用梯度检查来确认您的反向传播的作品。然后禁用梯度检查。

6. 使用梯度下降或内置的优化功能，以theta中的权重最小化成本函数。

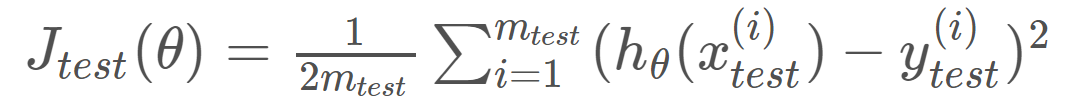
第六周

机器学习诊疗法

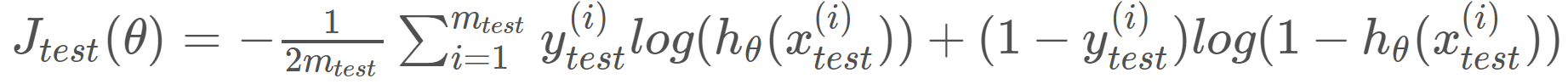


通过将数据分为训练集和测试集，将训练集训练出的参数用测试集数据测试性能。

线性回归时：

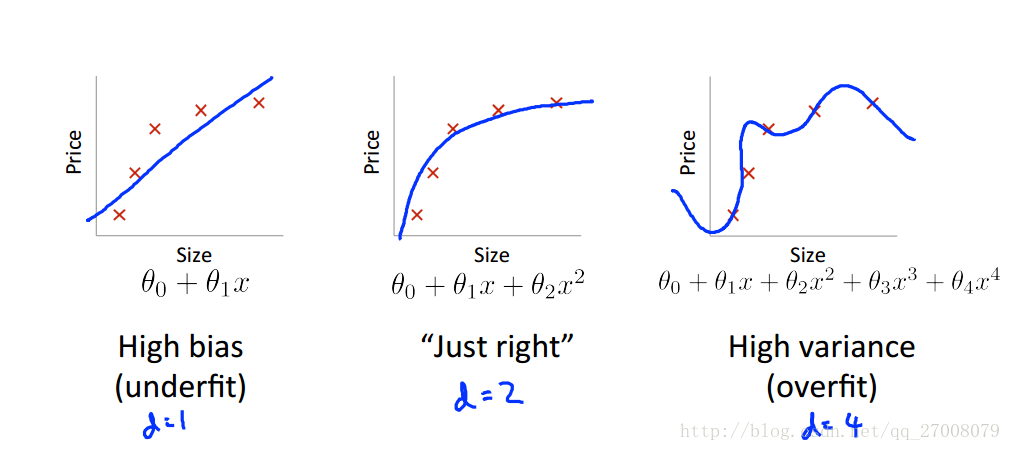


逻辑回归时：

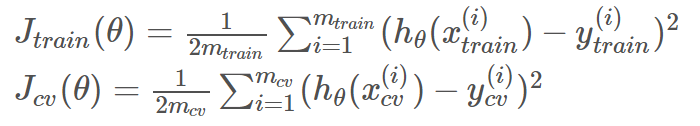


多项式回归中，我们可以把数据集分为三类，训练集，交叉验证集和测试集来确定假设函数的次数，用交叉验证集来作为评判选择的标准，选择合适的模型，而测试集则是作为算法性能的评判。

诊断偏差和方差



上面的图分别表示了高偏差，刚好，高方差



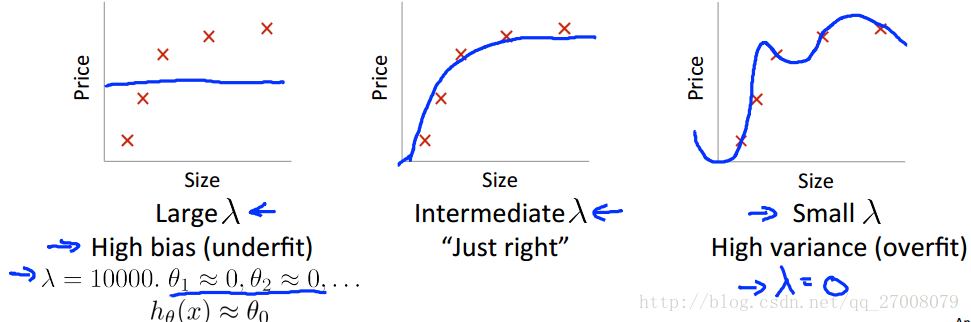


从图中可以看出，随着多项式次数的增大，训练集上的偏差逐渐变小，而交叉验证集上的偏差在减小到一定程度后开始升高

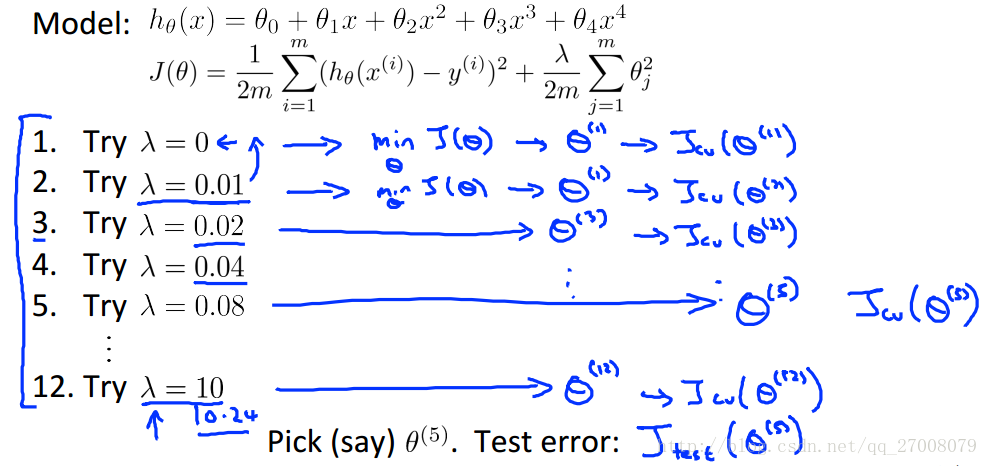
在高偏差(欠拟合中)很高，

在高方差(过拟合中)很低，

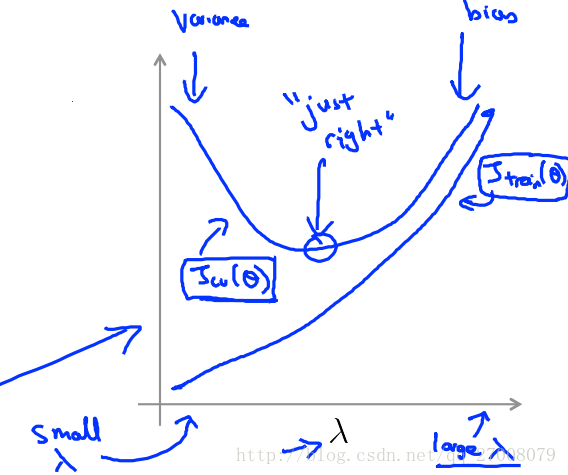
正则化和偏差/方差



可以通过在交叉验证集的测试选择较好的λ值

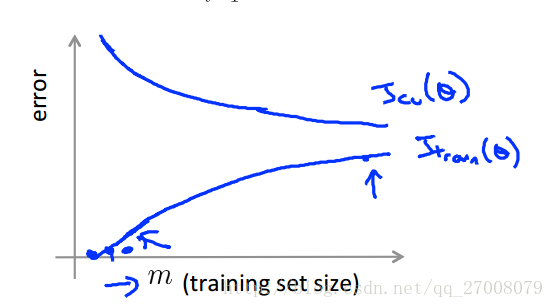


根据λ的大小画出的拟合曲线如下

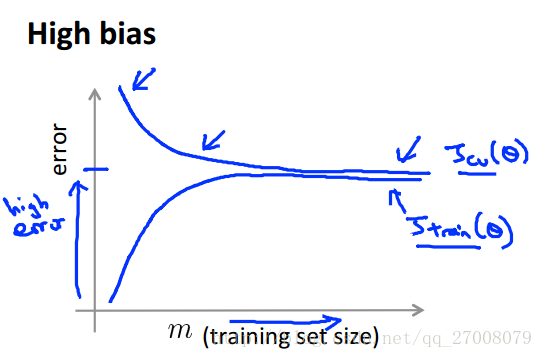


学习曲线

根据样本的大小与误差我们可以画出一般的学习曲线模样

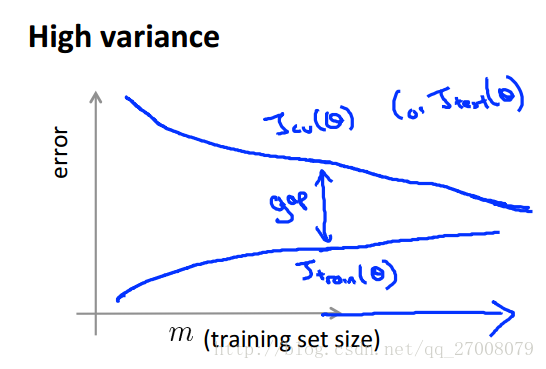


在高偏差的情况下，随着样本数目的增大，训练集上的误差和交叉验证集上的误差逐渐逼近。



也就是说，增大样本的方法对高偏差的模型并不能起到一定作用

而模型处于高方差的情况下，增大样本可能会起到效果。



总结如下：



机器学习系统设计

以垃圾邮件分类器为例，我们需要寻找最频繁出现的n个单词(10000-50000)作为训练集，而不是随意手工寻找100个单词。

1. 收集大量数据。
2. 从邮件信息中找寻复杂的特征（例如从邮件首部）
3. 从邮件体中找寻复杂的特征（discount与discounts是否被对待一致）
4. 使用复杂的算法来检测邮件中的拼写错误

对误差的分析

1. 先开始一个简单算法使你能快速实现它，在交叉验证集上测试它。
2. 画出学习曲线来判断是否更多的数据，更多的特征有助于改进算法。
3. 误差分析，在交叉验证集上检测你的算法，发现错误在某种样本上出现的趋势。

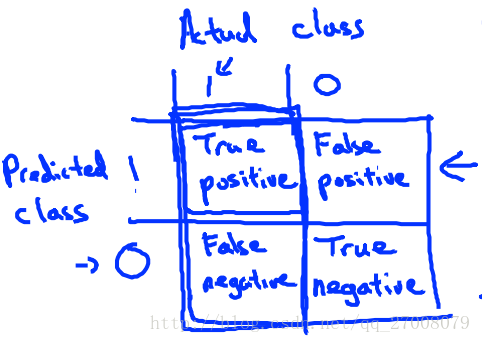
将误差转变为一个单一的数值非常重要，否则很难判断我们所设计的学习算法的表现。在误差分析中我们应使用定量计算来评判算法的表现。

偏斜类问题

以判断癌症的分类器为例

建立逻辑回归模型,y=1表示有癌症，y=0表示没有。

假设算法在测试集只有1%的错误，但测试集中只有0.5%的病人患有癌症，我们通过引入Precision和Recall来提高正确率

。

Precision(准确率)

在我们预测y=1的数据中，真正的癌症的比重。

Recall(召回率)

在真正癌症的数据中，我们预测癌症所占的比重。

在前面我们提到，将误差转变为一个单一的数值非常重要，因为这样我们才能方便的比较不同算法之间的优劣。现在我们有precision和recall两个衡量标准，我们需要权衡两者。如果用Logistic回归模型预测病人是否患癌症，考虑下面的情况：

假设考虑到一个正常人如果误判为癌症，将会承受不必要的心理和生理压力，所以我们要有很大把握才预测一个病人患癌症(y=1)。那么一种方式就是提高阙值(threshold)，不妨设我们将阙值提高到0.7，即：

Predict 1 if: hθ(x)≥0.7

Predict 0 if: hθ(x)<0.7

在这种情况下，我们将会有较高的Precision，但是Recall将会降低。

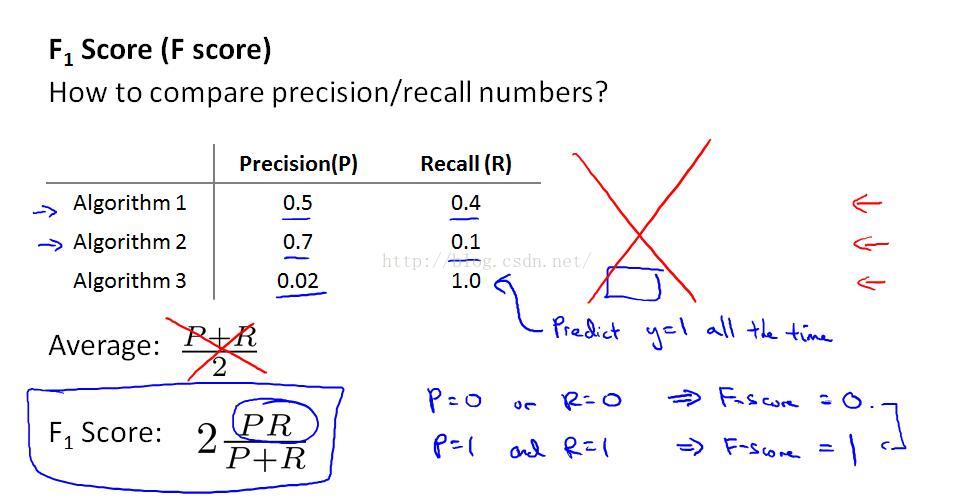
假设考虑到一个已经患癌症的病人如果误判为没有患癌症，那么病人可能将因不能及时治疗而失去宝贵生命，所以我们想要避免错过癌症患者的一种方式就是降低阙值，假设降低到0.3, 即

Predict 1 if: hθ(x)≥0.3

Predict 0 if: hθ(x)<0.3

在这种情况下，将得到较高的Recall,但是Precision将会下降。

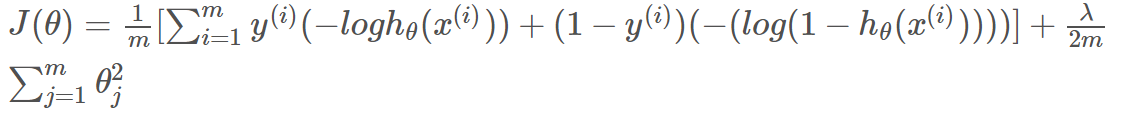
为了将Precision和Recall转变为一个单一的数值，我们引入了F=2



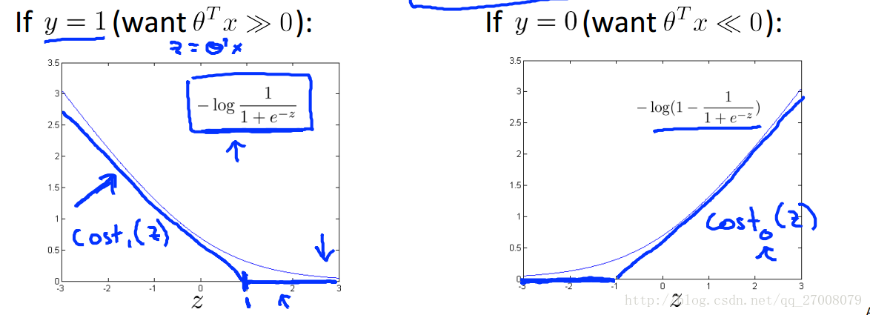
第七周

支持向量机(SVM)

逻辑回归中的代价函数为:

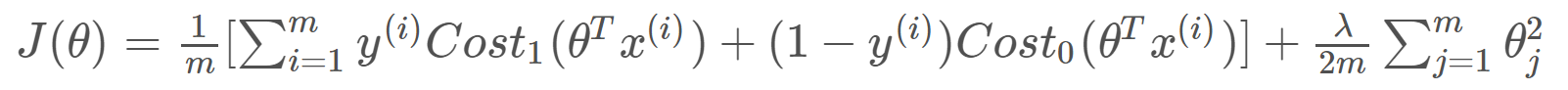


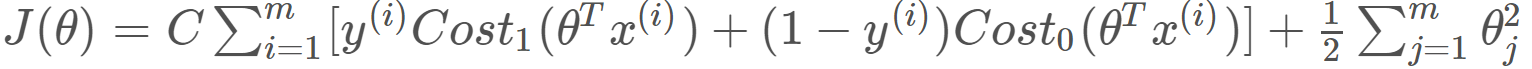
在SVM中对代价函数进行改变：



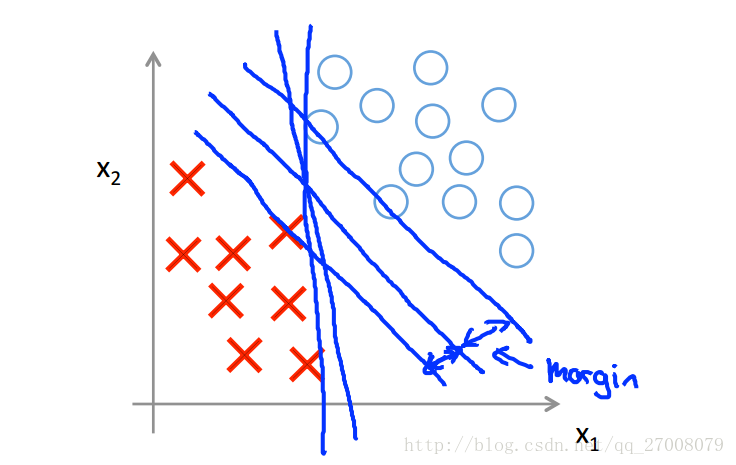
将log函数部分换成了蓝色折线所代表的cost函数。

CostFunction也相应改变为：

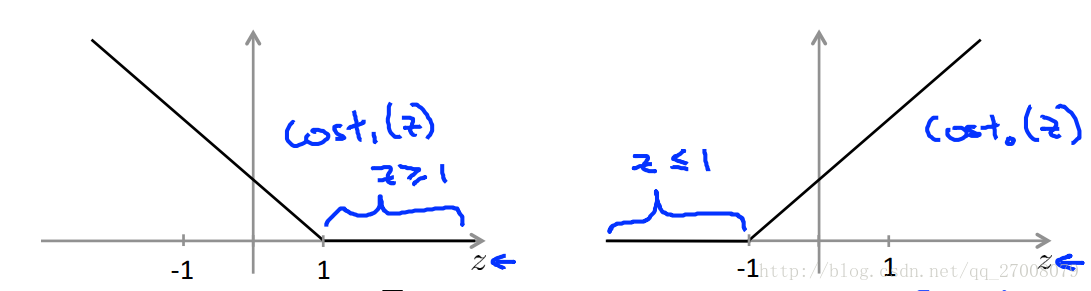
然后在SVM中，令上式同时乘以m，除以λ，我们用C代替1/λ：

上式即为SVM的代价函数

宽边界分界分类器SVM



如图，SVM希望找到最中间的那条分界线(最宽边界)来分割两类。首先观察SVM代价函数的图像：

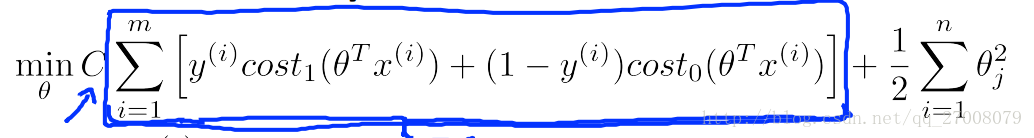


和逻辑回归比较：

IF y=1, we want x≥1 (not just ≥0)

IF y=0, we want x≤−1 (not just ≤0)

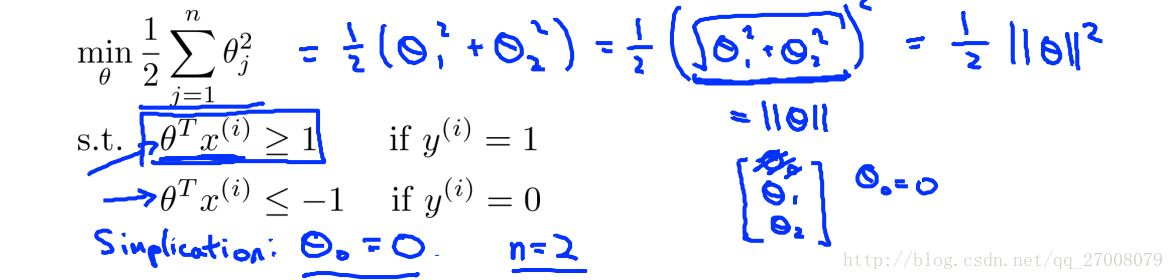
同时，C非常大时，我们使蓝色的这部分为0



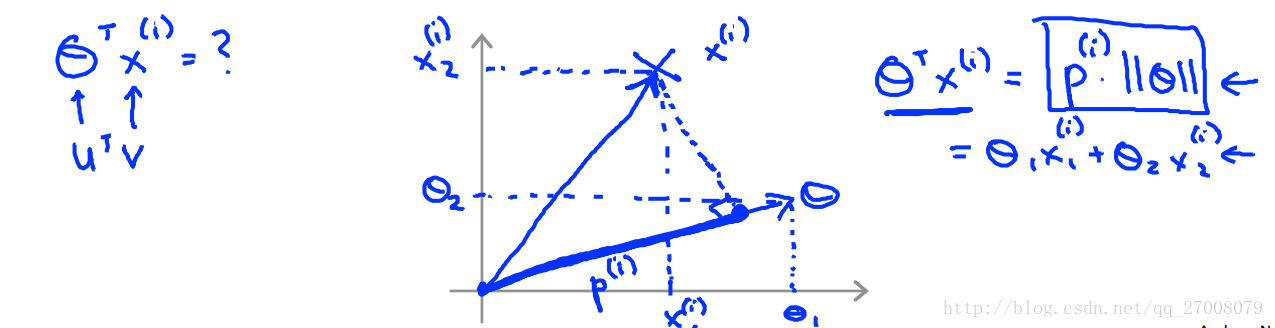
即原式变为min

SVM数学原理

SVM要求的最小值为||||的最小值，即的范数最小值，模型如下



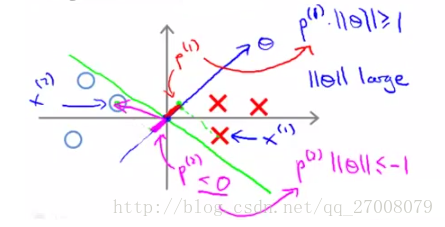
限制条件代表的含义：因为两个向量相乘的内积的几何含义如下：（向量A在另一条向量B上的映射× 向量B的模）



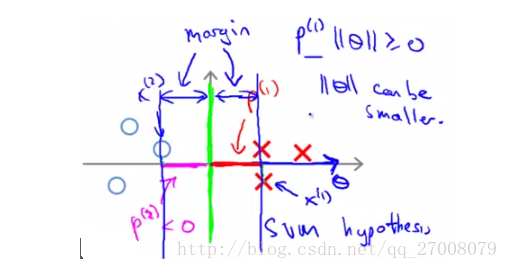
通过上面可知，我们要求||θ||的最小值，因此我们希望p(i)（x在θ上的映射）尽量大。只有这样才能使上面说的约束条件S.T满足。也就是SVM转变为了找到那个x在θ上的最大映射。

例子如下（θ为分界的法线（垂直））：

1. 假如选择了下面图中的绿色线作为边界，我们会发现p(i)比较小，这样不能得出||θ||的最小值



1. 如果选择下面的绿线作为边界，我们可以得较长的映射p、和较小的||θ||值。

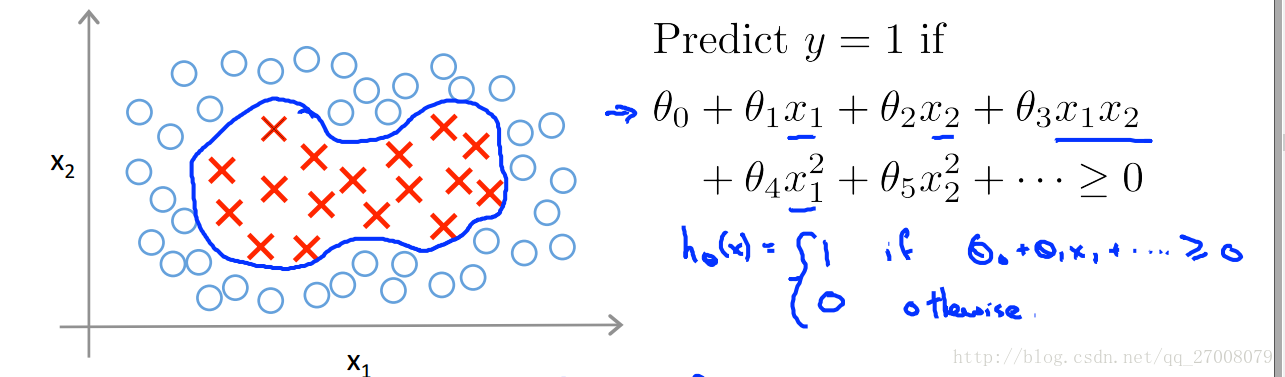


总结：

1. SVM要找到最中间的边界。
2. 所以要找到最长的映射p。
3. 进而可以找到所求参数θ的最小值。

核函数

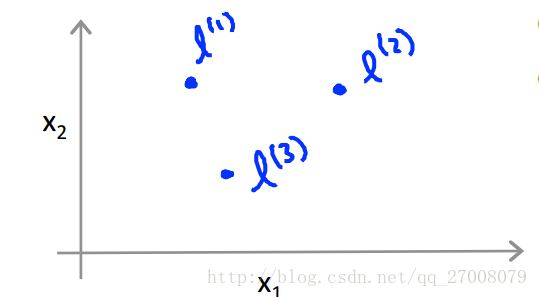
为了解决非线性拟合问题



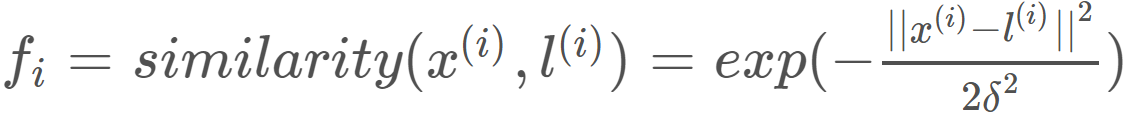
假设函数(用f代替x的参数)这个函数为新的假设函数。

引入：

如果我们给出几个向量作为landmarks

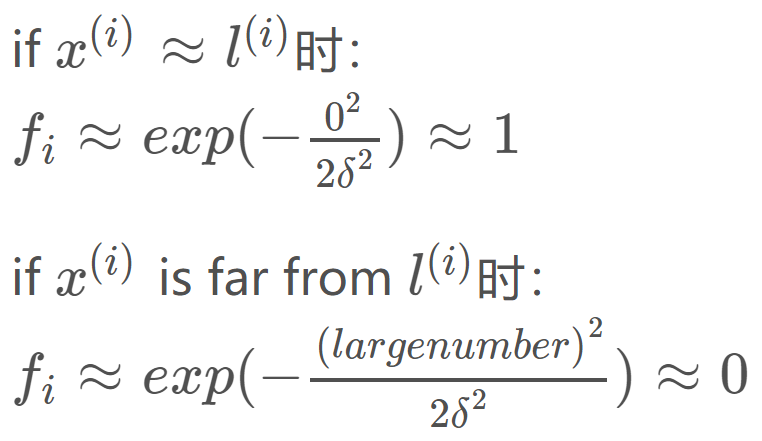


我们设置f函数，衡量标记点和原先样本点的相似性：



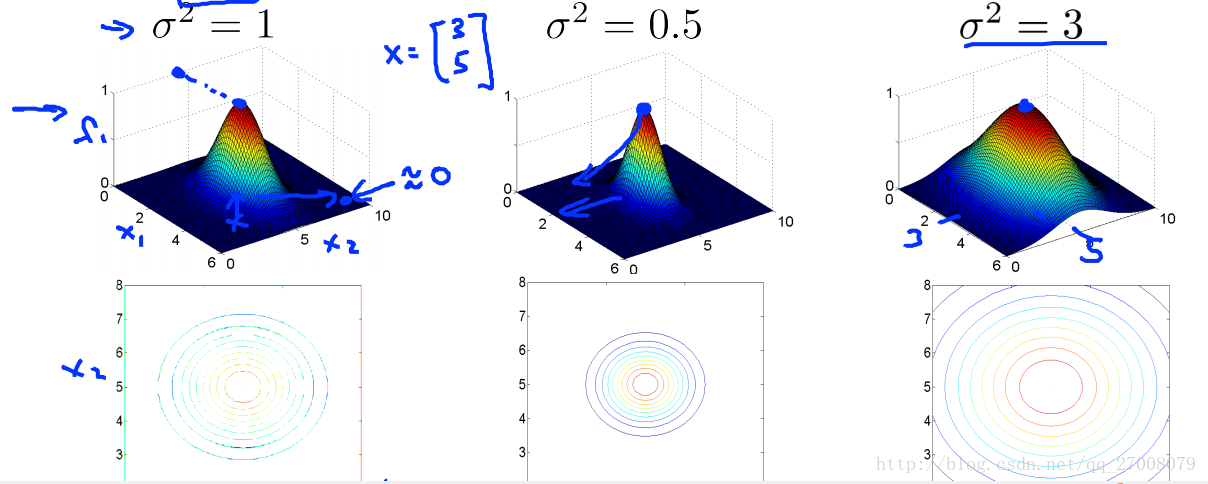
exp中的函数为高斯核函数

在这种情况下，f的取值为：



这样就表示了样本x的一种高维映射，里标记点越近值越高。

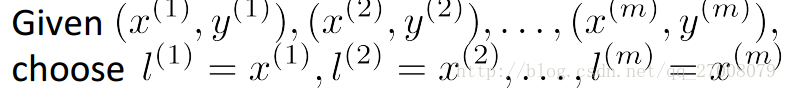
最后，我们需要研究对核函数的影响，通过图片看出，越大收敛慢，越小收敛快。



SVM的计算步骤

Landmark的选取：

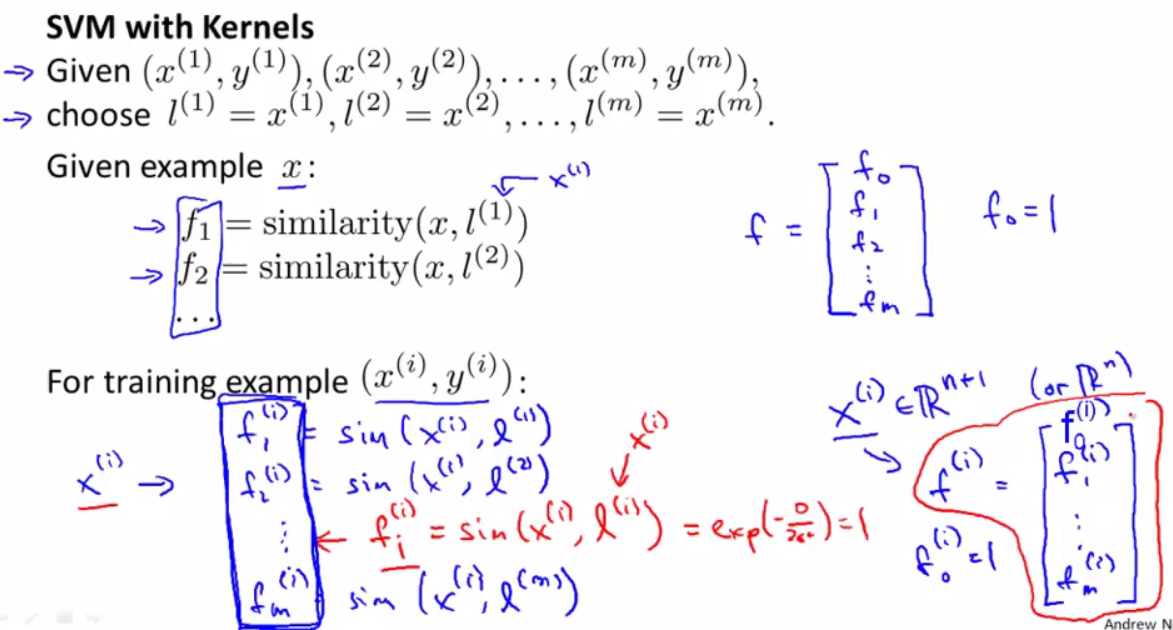
1. 一种比较好的方法是将训练集中的正样本选取为标记点，如图：



这样定义好l后，再定义每一个映射F。

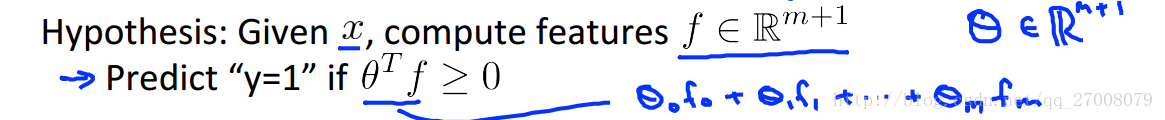
此时，对于每一个训练集中的数据，我们都有一个m+1维向量与之对应（m维训练数据[f1,f2,…,fm]附加一维f0=1）。

1. 每给定一个样本x，计算他与所有l的映射f（这里与之前cost function的区别是用kernel代替了x）：



1. 预测样本点的归属类：

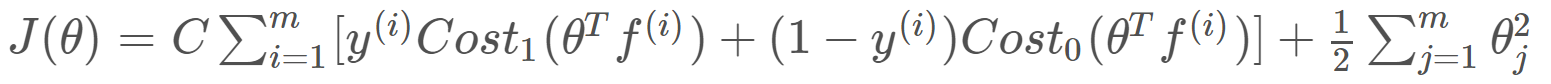
在预测时，使用以下公式：



这个函数等价于：

以上就是已知参数θ时 ，怎么做出预测的过程。那么 怎样得到参数θ呢？

方法具体来说就是要求解这个最小化问题， 你需要求出能使这个式子取最小值的参数θ



下面总结一些SVM中的参数对模型的影响，主要两个方面：C和

1. C

大C：低偏差，高方差（对应低λ，高拟合）因为C约等于λ的倒数

小C：高偏差，低方差（对应高λ）

大：fi分布更平滑，高偏差，低方差

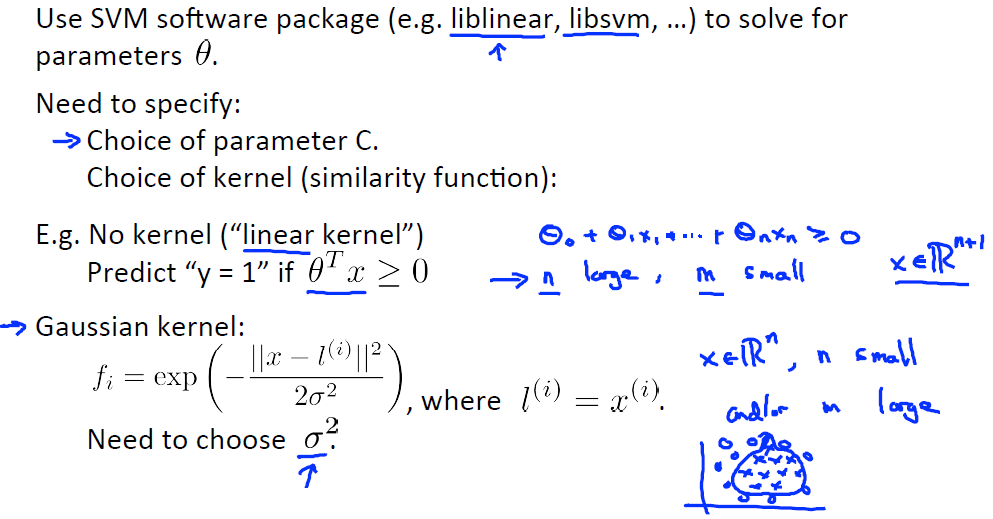
小：fi分布更集中，低偏差，高方差（过拟合）

实践SVM

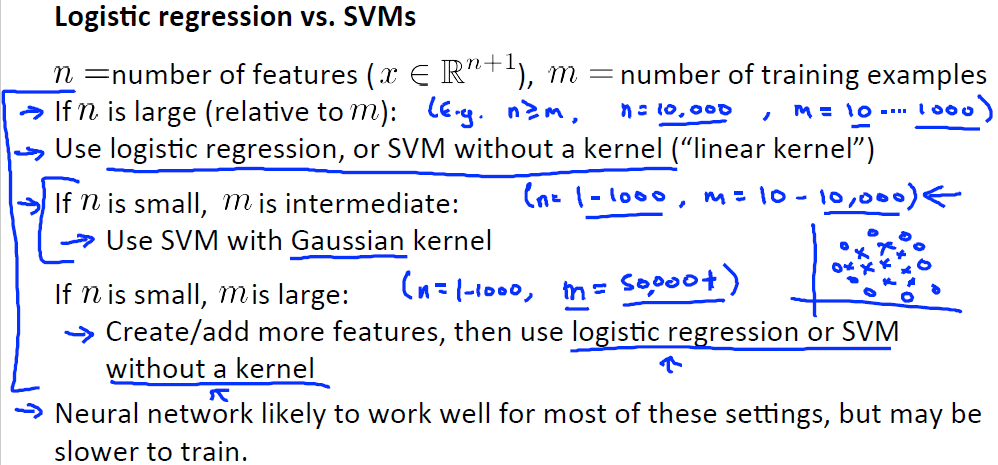
用SVM进行机器学习的过程就是一个optimize参数的过程。

SVM进行学习，主要分为：

1. No Kernel（Linear Kernel），,predict y=1 if x >= 0
2. Kernel f(e.g. Gaussian Kernel),也叫Radial Basis Function简称RBF，它能够把原始特征映射到无穷维，,这里需要选择参数



逻辑回归 vs SVM分类器



①当n>=m，如n=10000，m=10~1000时，建议用logistic regression, 或者linear kernel的SVM

②如果n小，m不大不小，如n=1~1000，m=10~10000，建议用Gaussian Kernel的SVM

③如果n很小，m很大，如n=1~1000，m>50000，建议增加更多的feature并使用logistic regression, 或者linear kernel的SVM

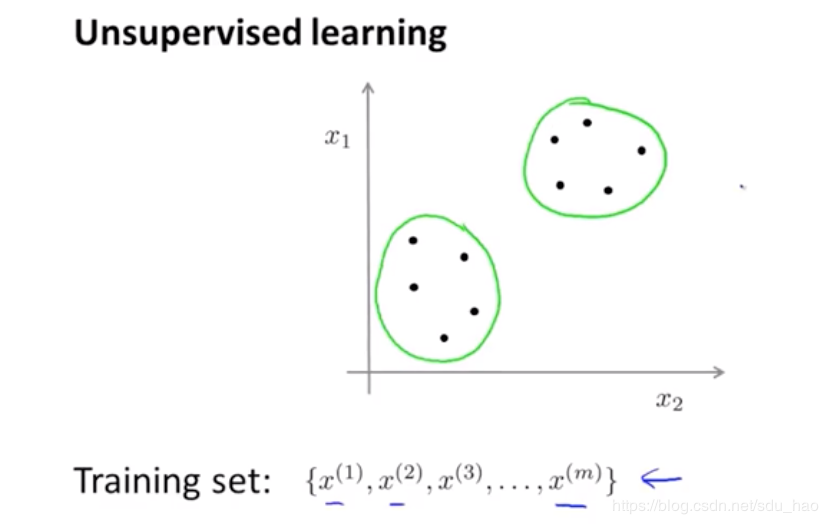
原因，①模型简单即可解决，③如果还用Gaussian kernel会导致很慢，所以还选择logistic regression或者linear kernel

神经网络可以解决以上任何问题，但速度可能会很慢。

第八周

无监督学习

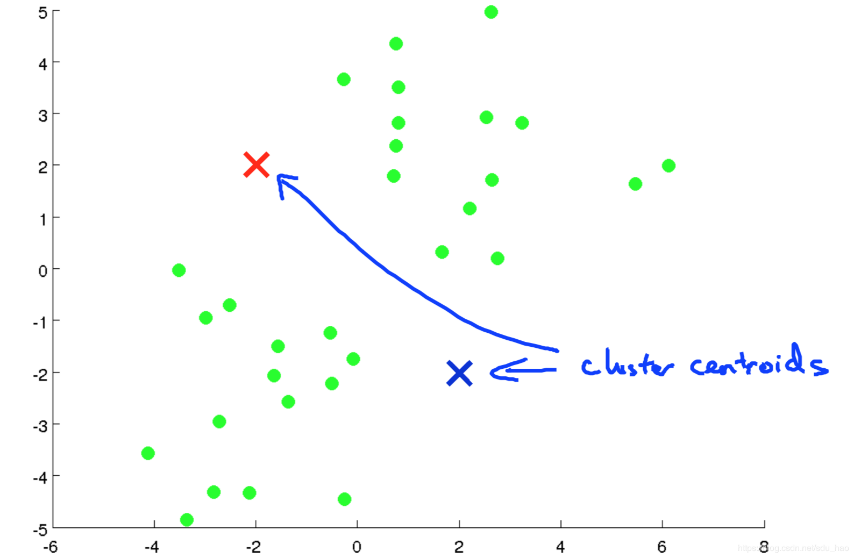
在无监督学习问题中，训练样本没有任何标签信息。我们需要做的是把这些无标签的数据输入到算法中，利用算法找到这些数据中隐藏的结构。对于下图中的数据，我们可以用算法找到一种隐藏结构-簇(这些数据可以分为两组分开的簇),我们称这个算法为聚类算法，这是我们学习的第一个无监督算法。当然无监督算法还有很多，可以发现数据中隐藏的不同结构，不仅仅是簇：



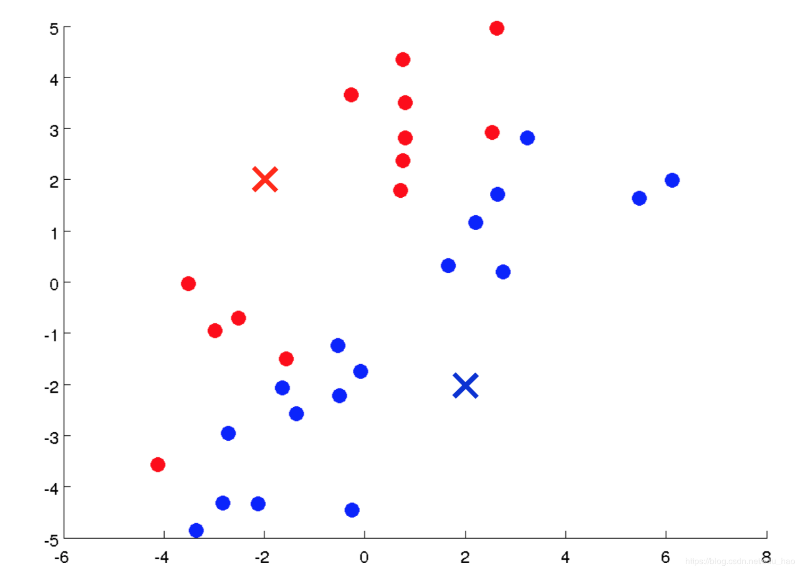
K-Means算法

K-Means原理

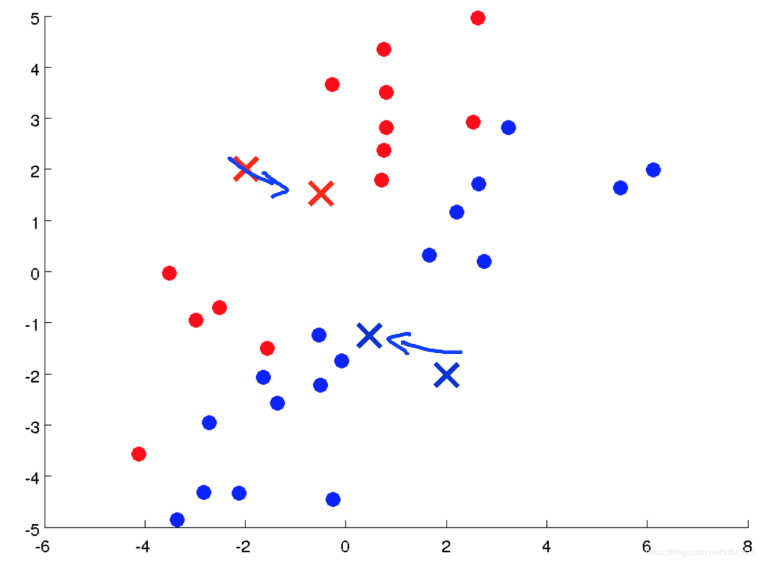
1. 可视化一组无标签训练集(,下图中的绿点),随机选择两个聚类中心(下图中的红、蓝叉）:



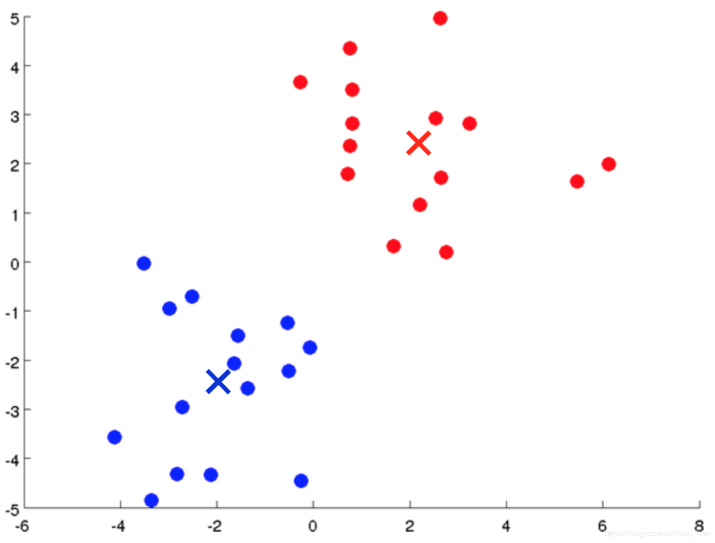
1. 遍历每个样本点，计算其与两个聚类中心的距离，离哪个聚类中心更近，就把该点划分为哪个聚类中心，一次遍历后如下所示：



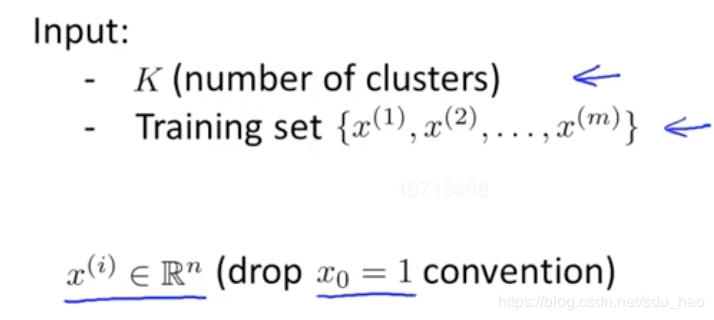
1. 移动聚类中心，分别对上图中的红、蓝色点计算均值，得到新的聚类中心:



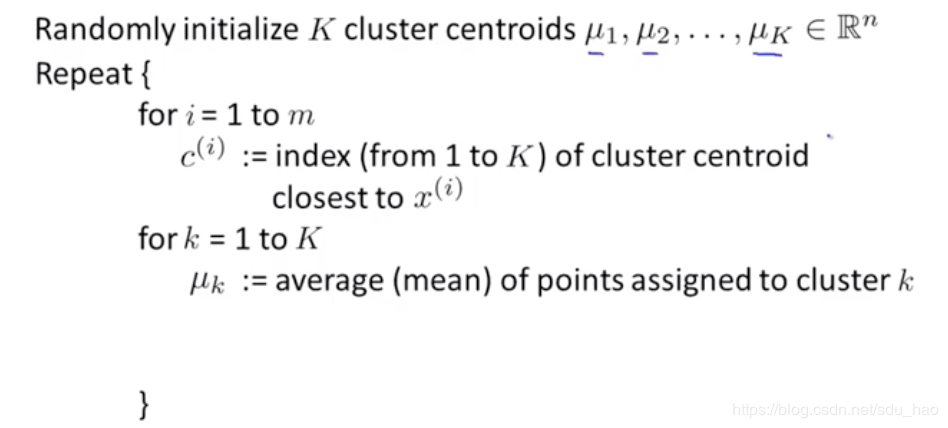
4.重复第二、三步，直到聚类中心不在变化为止，得到的最终聚类效果如下：



K-means算法输入：



K-means算法流程



其中第一个内循环是簇分配，

第二个内循环是更新聚类中心(簇).比如，在第一个内循环结束后，有，,,四个样本点被划分给了聚类中心，即，那么,如果一个聚类中心没有分配到任何样本点时，一般把这个聚类中心去除，此时k个聚类中心就变成了k-1个聚类中心；如果必须把数据分为k个簇，此时可以重新随机初始化k个聚类中心，重新开始。

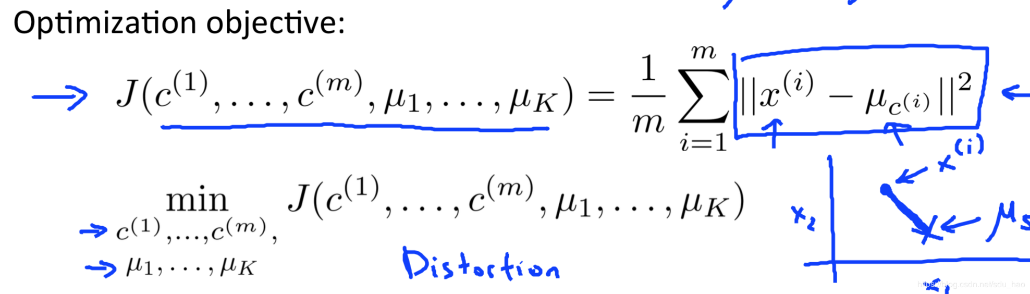
优化目标

符号说明：

：在最近的一次簇分配过程中，样本被分配到簇的聚类中心下标（离最近的聚类中心的下标）

：某个聚类中心(k=1…K,)

：样本被分配到簇的聚类中心（离最近的聚类中心）



K-means的算法流程就是在实现这个优化目标



第一个内循环-簇分配，固定聚类中心不动，关于变量，最小化代价函数J；

第二个内循环-更新聚类中心，关于，最小化代价函数J；再进行下一次迭代

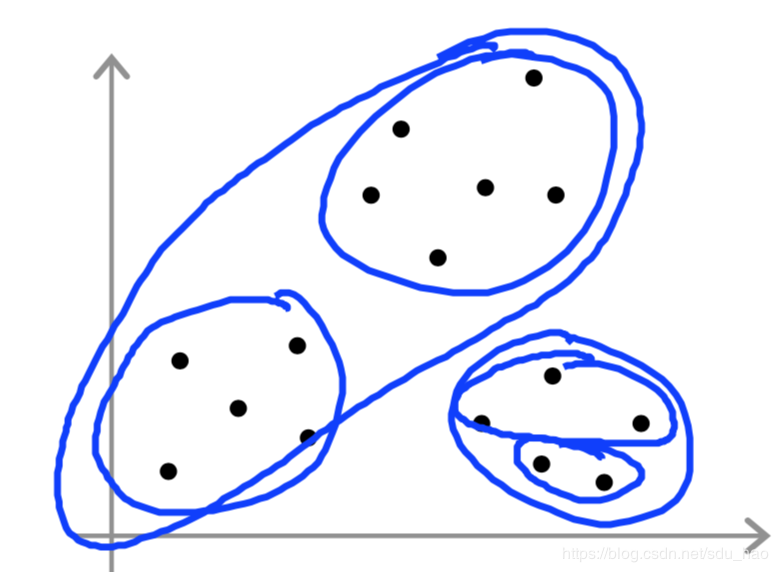
随机初始化

随机初始化聚类中心避免k-means陷入局部最优。

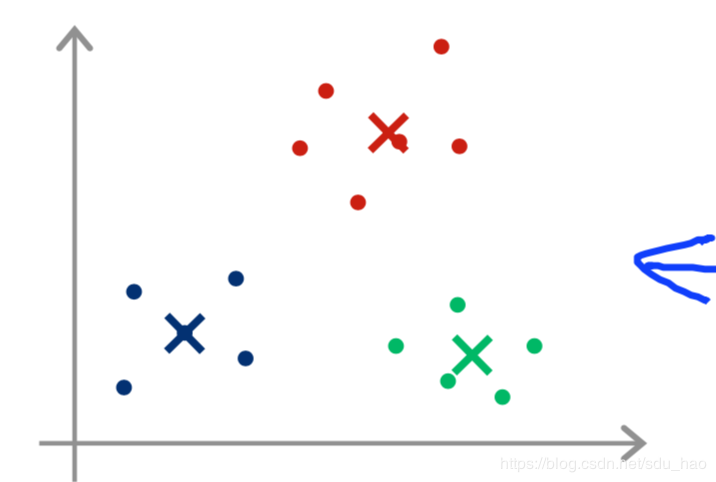
首先选择聚类中心数K应该小于样本数m；

然后随机选择k个样本点作为初始的聚类中心，即，…

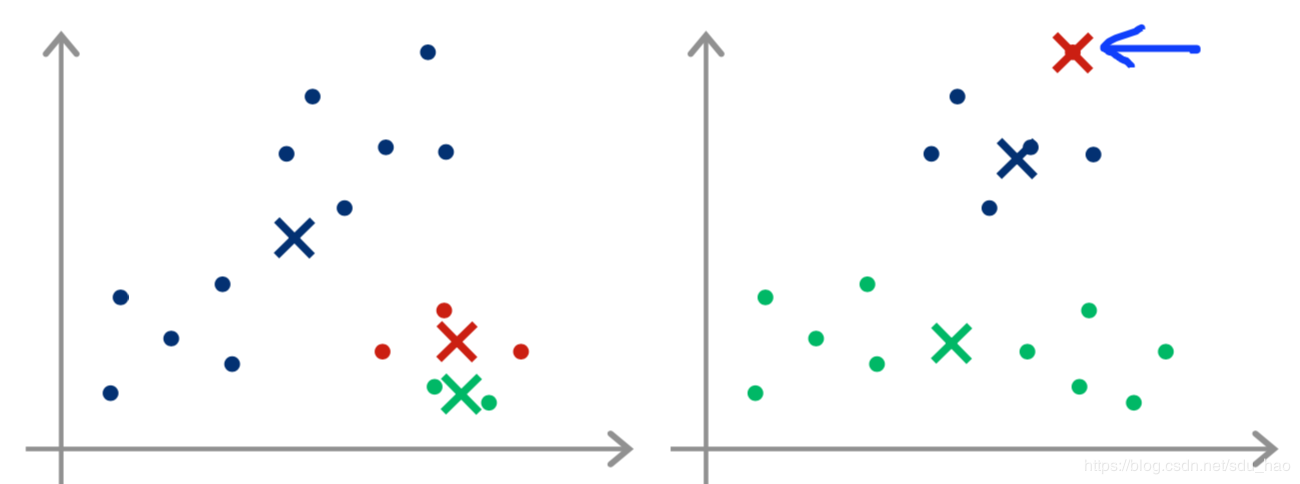
局部最优问题：随机初始化不同，k-means得到的聚类效果也是不同的，有时会陷入局部最优。局部最优指的是代价函数J得到一个局部最优解，考虑下图情况：



对于上图的样本集，最好的聚类效果应该如下图所示：

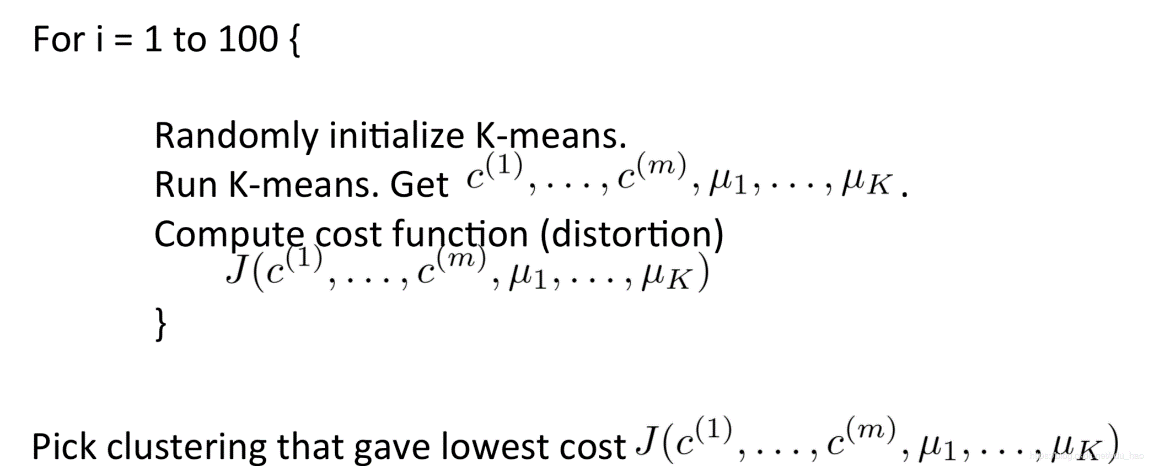


但是，可能由于随机初始化的不同，可能会得到下面的两种不同的聚类效果，称此为k-means陷入局部最优：



解决方法：

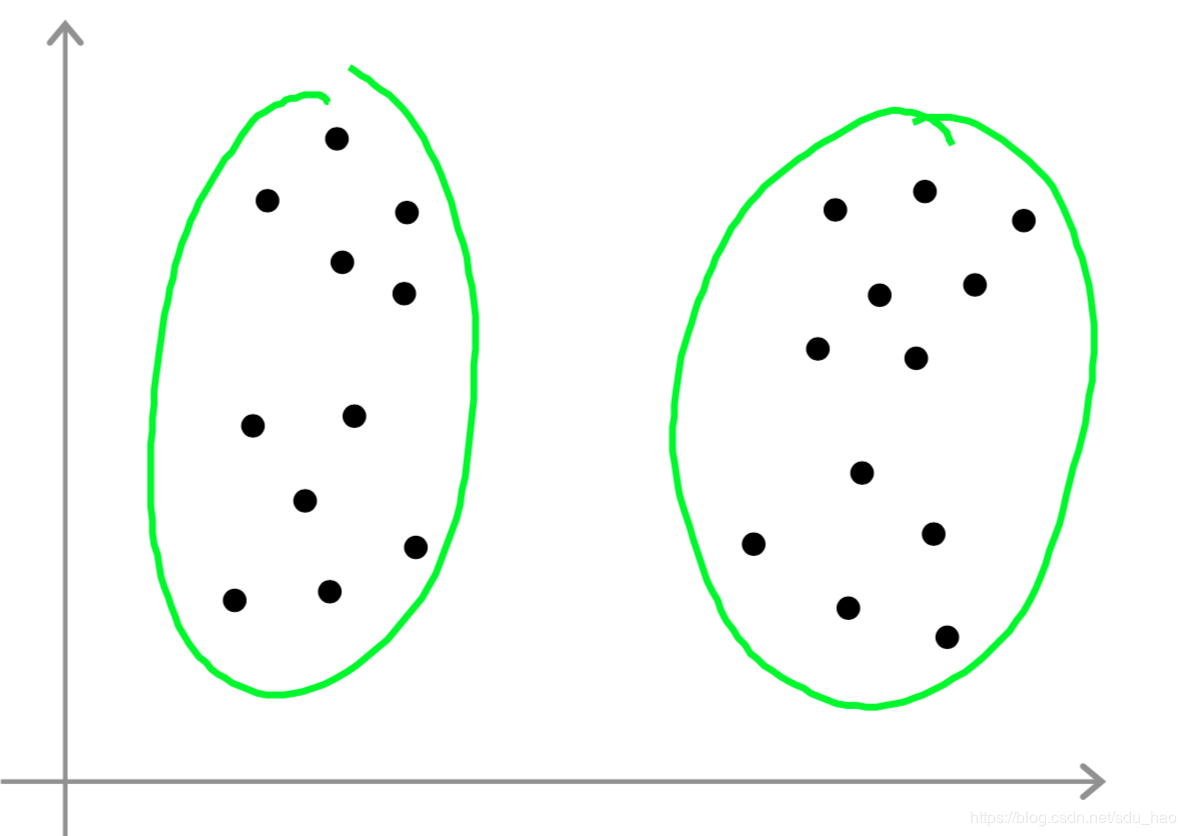
多次使用k-means算法，不同的随机初始化，会得到不同的聚类效果，对应不同的参数。从中找到一组最优的参数，使代价函数J最小，那么该组参数所对应的聚类结果认为是最好的：



这种方法在聚类中心数量K比较小(K=2-10，尤其是K=2-4时)时，效果比较好；当K比较大时，随机初始化对聚类结果的影响就没那么大了，此时这个方法的效果可能就不会很显著。

选择聚类数量

一般通过人工或经验选择聚类数量。但某些可视化的数据聚类数量的选择也是模糊的，见下图：

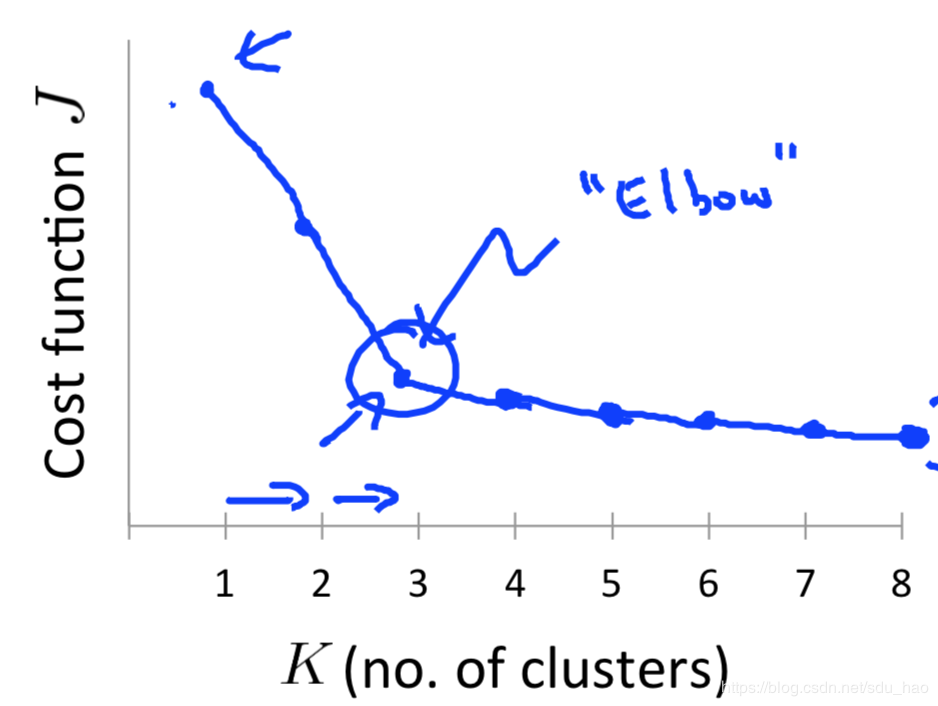


上图中我们既可以选择左右两个聚类中心，也可以选择四个(左右又可以分为上下两部分），而且不同的选择间是没有对错的。

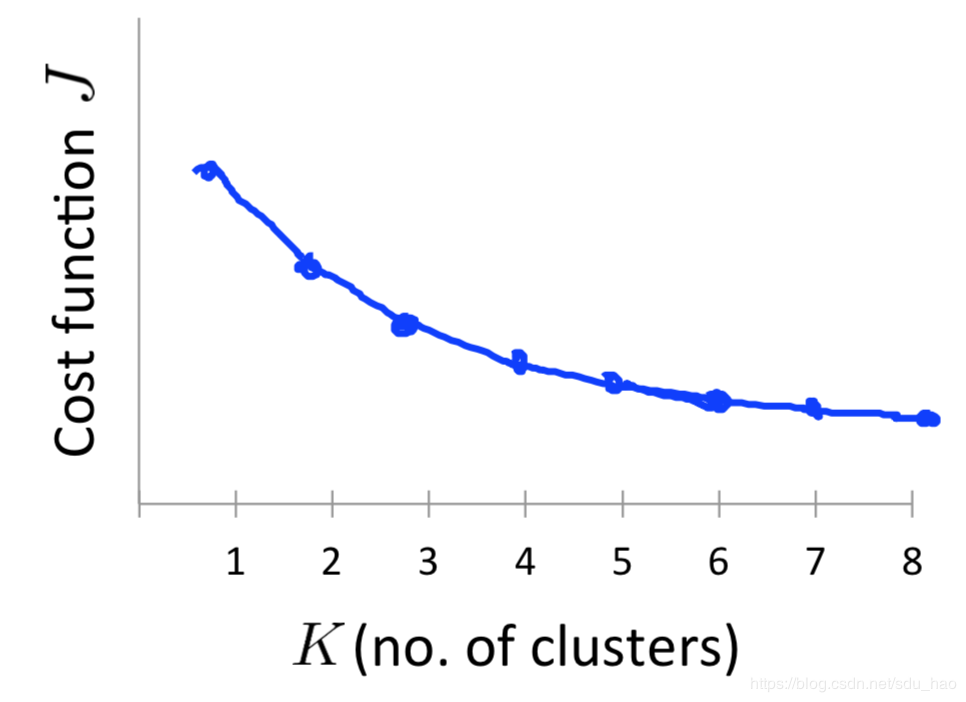
一种自动选择聚类数量的方法：

Elbow method

这种自动选择算法，在某些情况下有用，原理如图：

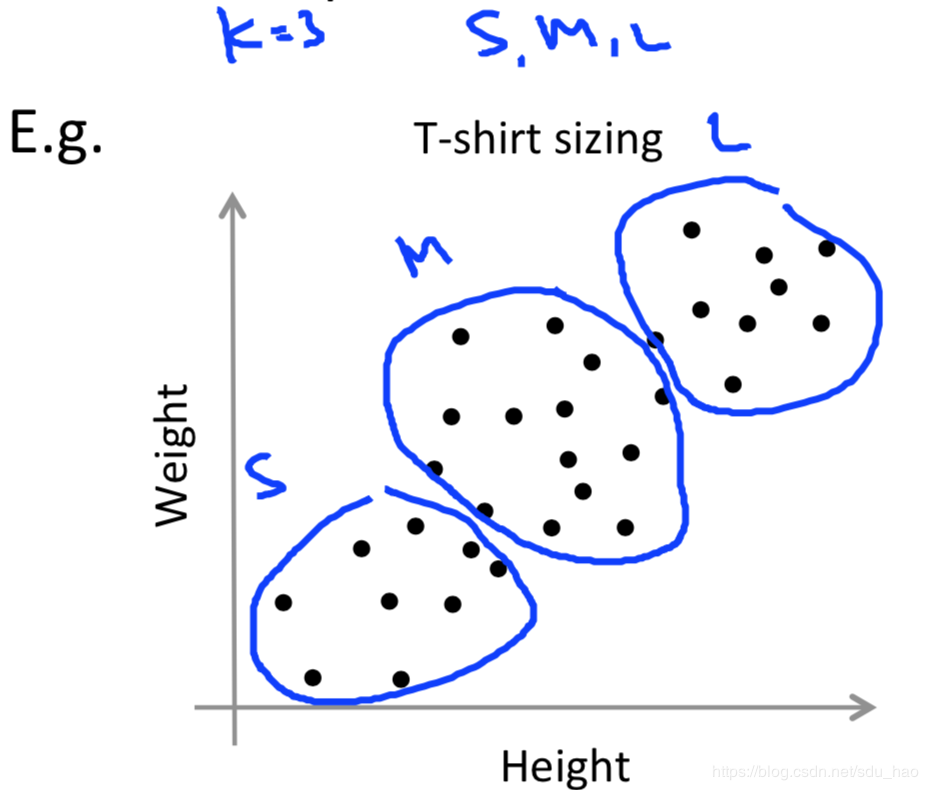


尝试不同的聚类数量，绘制代价函数J关于聚类数量K的曲线，如果曲线的形式类似于人的“胳膊”，那么在拐点(肘部)附近的聚类数量可以认为是最佳的选择。但有时会得到如下曲线：



一种更好的选择是从聚类的目的出发，来选择K，评估在这种情况下的聚类效果是否能满足你的期望和将来的目的。

比如T-shirt型号的设计，如果你希望T-shirt有s,m,l三款，那么就可以对下图的身高、体重数据聚类时，让K=3:



降维