第一章：

1.策略：平均期望风险最小化、结构风险最小化（SRM）

前者追求对模型的拟合，损失函数中只定义均分误差

后者对模型的复杂度，加入惩罚项，模型拟合、模型复杂程度并重， 越简单越好

对多个模型，选择一个拟合得好，且更倾向于简单的模型

这样，训练误差小，同时，测试误差小（更看重，体现了模型的泛化能力）

SRM的一般方法：正则化 ；或者，交叉验证

正则化：比如取 参数向量（模型要求的参数θ）的L2范数，如0.5\*||W|^2|

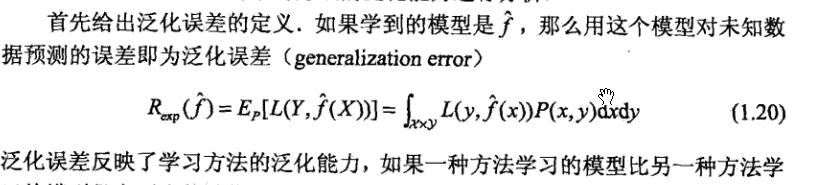
交叉验证（cross validation）

将已知数据划分为（按给定比例，随机划分）训练集（train）/验证集（validation）/测试集（test）

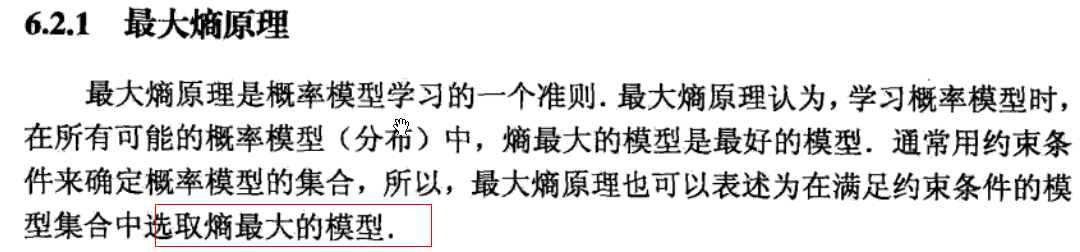
train:用于训练模型——分类器的不同参数设置，会拟合出不同的模型

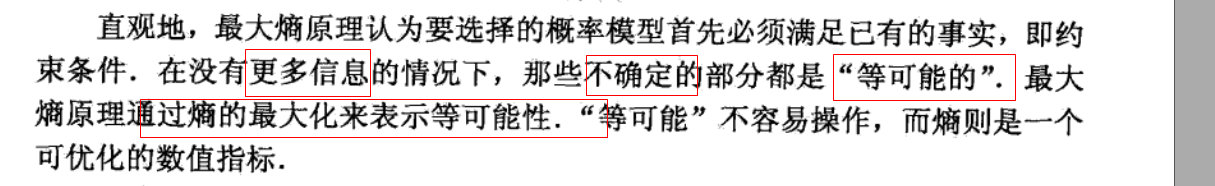
validation:用于选择模型——选择测试误差最小的模型

test:用于评估



最大熵模型：





简单说，当一切信息未知时，我们认为各种随机事件是“等可能”发生的。

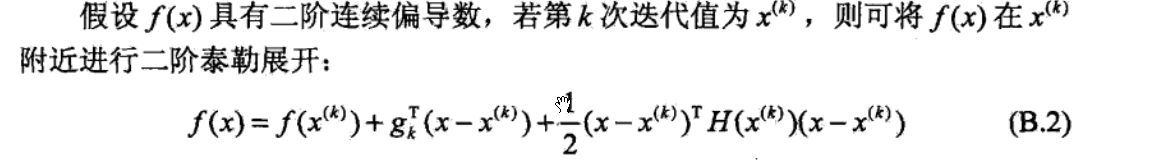
已知部分信息时，剩余未知信息，应该也是等可能发生的。

附录B：

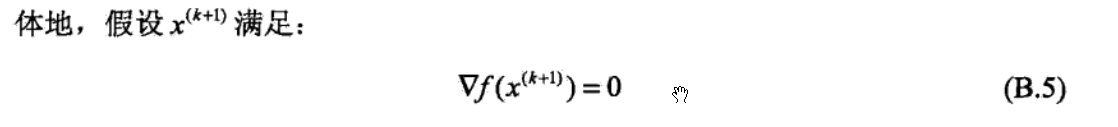
牛顿法：

是根据二阶泰勒展开式，近似得到函数f(x)的表达式，当x非常接近于某个点x0时，f(x)也会非常接近f(x0)

展开公式如下：

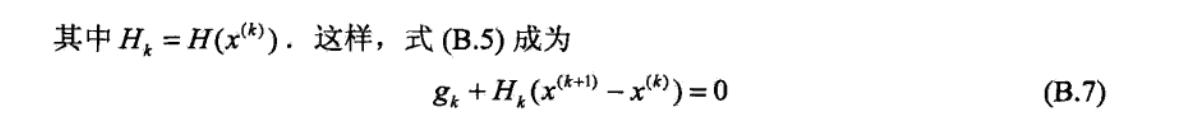


假定经过k+1次迭代，x(k+1)就是极值点，那么必然满足 x(k+1)点的梯度向量为0，即

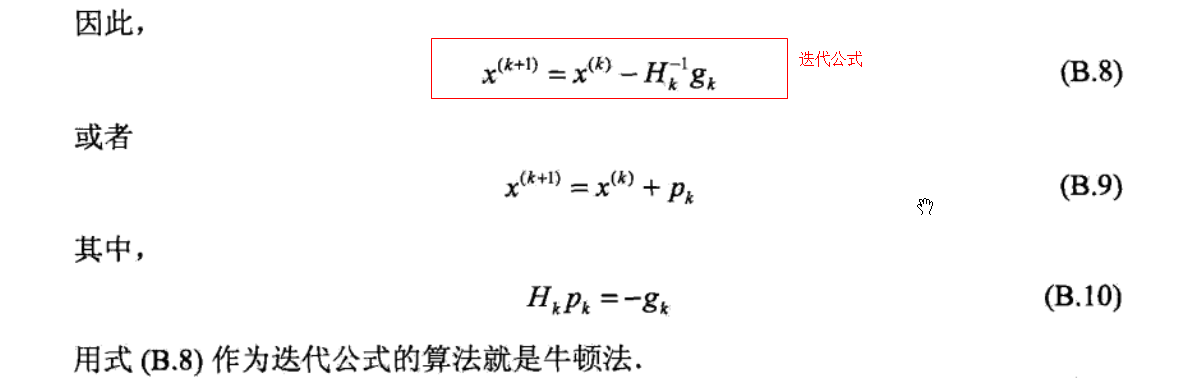


那么带入公式B.2，令x=x(k+1)，得到f(x(k+1)) = …

在x(k+1)处对f( x(k+1) ) 求导，值为0，可得如下公式：



变形为：



在迭代公式中，我们只要求出B.8式子中的Hk(-1)gk ,即可， 然后也知道第k次迭代的点xk,就能求出x(k+1)，

这里关键是求Hk(-1)， 即f(x)在点xk处的二阶导矩阵（何塞矩阵，准确的说xk应该是一个向量（xa,xb,xc….） ）

拟牛顿法公式结束：

更新算法——————求出Hk(-1)gk后，运用公式B.8，更新x(k+1)的值，即可。

直到某个条件为止：

（这个条件，基本就是————在向量x（k+1）处的梯度相当接近于0，比如小于某个阀值时，认为是找到了极值点，再找下去无意义了）

**牛顿法，缺陷在于：每次更新，都要计算Hk，这个成本巨大。**

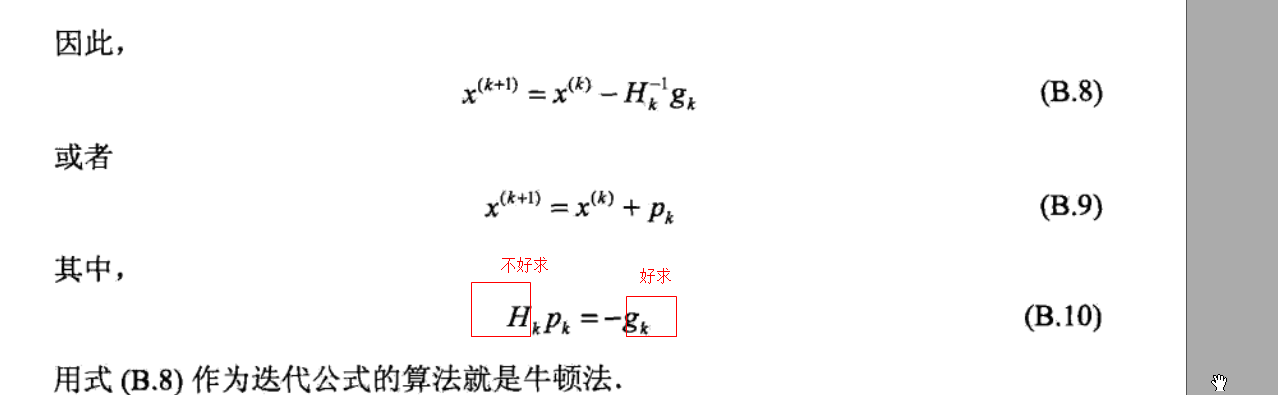
有没有办法找出1个近似的Hk（不需要太大计算量的），该替代的Hk能满足搜索方向是向下的（往极小值点方向前进的），从而出现了“拟牛顿法”

**拟牛顿法————就是围绕：找出一个上述的近似Hk（比较容易计算出该近似Hk的算法）**

**如何生成满足“拟牛顿”条件的近似Hk矩阵**

**最流行的拟牛顿法：BFGS**

**其余还有DFP、Broyden**



5.决策树分类：

一般3步：

特征选择（1.过滤掉和随机分类效果差不多的分类特征，2.构建树时，优先选择分类能力强的特征——例如增益信息比高的分类特征，或者纯度高Gini系数小的特征）

决策树生成（构建树，对分类特征排序，先用哪个分类，再用哪个。。）

决策树修剪（特别复杂的树会“过拟合”，对训练集好，对验证集不好，需要将树剪枝，部分叶节点和父节点合并）

特征选择——使用“信息增益”或“信息增益比”算法

决策树生成——使用ID3（使用“信息增益”算法进行“特征选择”）或使用C4.5算法（使用“信息增益比”进行“特征选择”）

剪枝——

特征选择：

1. 使用“信息增益”法。

对于某个特征Xi，和标签列Y（目标变量）

可以计算，这两个变量的“信息增益”

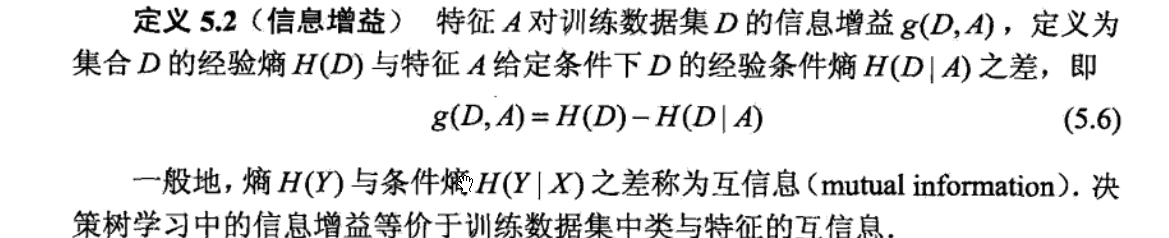
公式为：

信息增益 = H（Y） -H(Y|Xi)

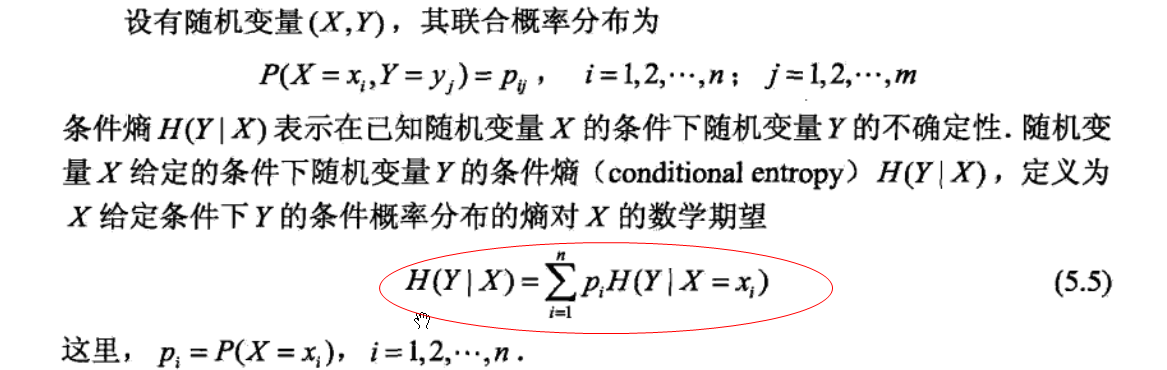
这里H(Y)就是 Y的信息熵， H(Y|Xi) 是 在Xi发生的条件下，Y的条件熵。

**注意：实际计算都是用的“经验熵”——即根据已知样本求的**

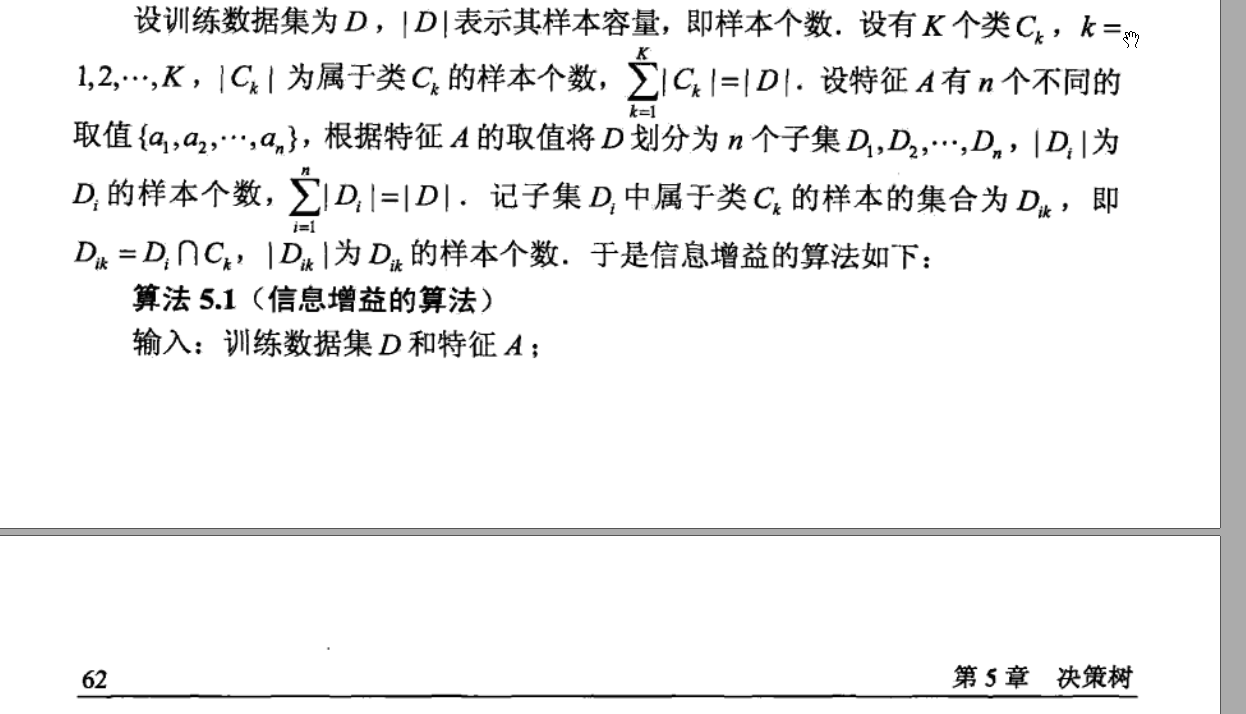
定义如下：



条件熵公式：

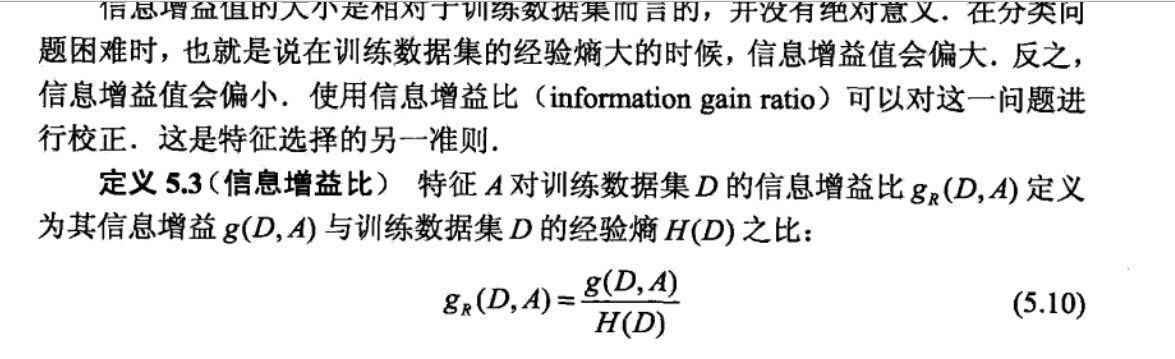


具体计算，见下图部分



为了去除“经验熵”大，导致信息增益也大。引入**“信息增益比”，**

即 = 信息增益/经验熵值 =（H(D) – H(D|X)）/H(D) = 1 – H（D|X）/H(D)



所以，特征选择*，可以根据****“信息增益”或“信息增益比”，***来进行

**条件熵的计算公式及推导：**

**https://blog.csdn.net/zhangyongzhen1991/article/details/59057469**

另有python实现的各种“熵”的代码（交叉熵，条件熵）

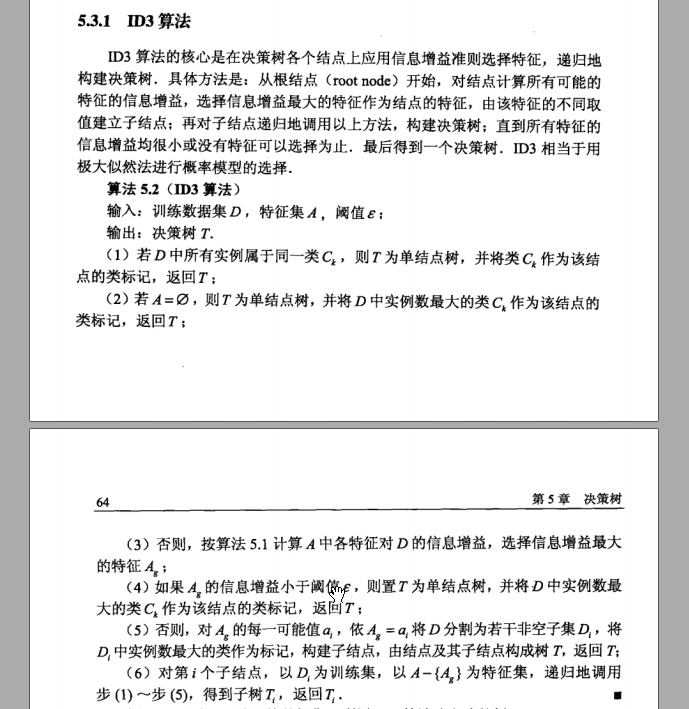
不要将“条件熵”和传递熵 搞“混淆”了。

**互信息————即“信息增益”**

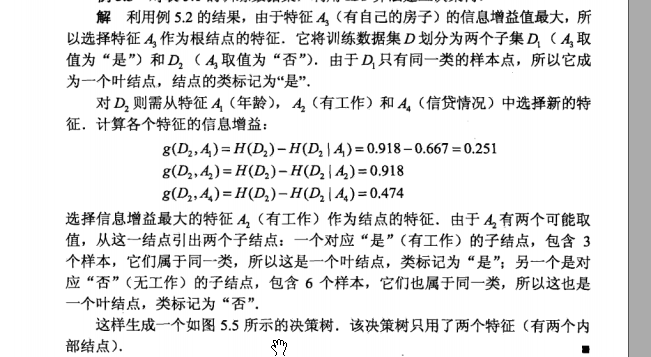
继续接5，决策树分类————————

数的生成：

ID3算法：



示例：



树的生成，就是“递归”思想：

对于每次递归的子数据集（只有第一次根节点为“全部数据集”，其余都是已经剔除被分类过的数据集）Di ,以及剩下的特征集{A1,A2,。。。Ak,。。。}.

1. 先计算各Ak对Di的“信息增益”g（Di,Ak）（有时也用“信息增益比”）

选择 max(g(Di,Ak)) 对应的 Ak作为特征，去构建

1. 依据Ak对Di进行分组，成为不同的数据集G = { D11,D12 ,…. Dki,…}，

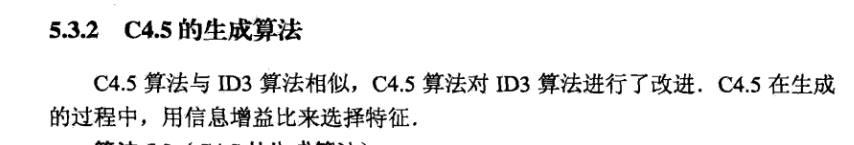
if LD = Dki 属于同一类（不用分类），则将其标记为“叶节点”， 并给出对应的“分类标签” or 不属于同一类(有很多属性xj)，但信息增益比<阀值ξ ，也停止对该LD分类，并将（对应类标计数最多的特征属性xj, 对应其标签，比如1, 即 xj ->1）

else 对“剔除”过“已分类”的Dki， 对剩余的“需继续分类”的数据集G-LD 进行1~2步的递归操作（即又开始找最大信息增益）

注意：每一步的“信息增益”计算g（Di,Ak）,其中H(Di) 都是指剩余数据集Di的“标签列”的信息熵

ID3的缺陷——————————容易过拟合

C4.5的生成算法————就是将ID3中的特征继续分裂的依据：“信息增益” 替换成“信息增益比”————过程都是一样的：



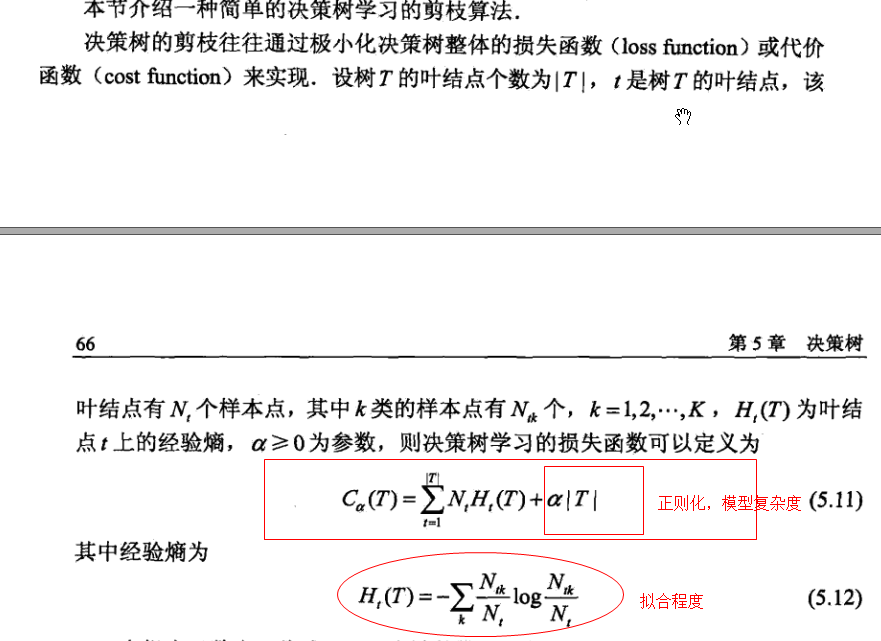
决策树剪枝pruning:

损失函数的定义——也是正则化后的损失程度。

|T|表示叶节点的个数， 叶节点越多， 树越大，越复杂。意味着模型越复杂。

下面的损失函数，Nt项不太明白为什么要加入。

但是损失函数的意义是： 每个叶节点



另外一个单独的算法————CART算法（分类和回归树）

可用于——分类、回归

关于拉格朗日求极值（无约束，有约束下）的问题,有张图画得很明白：

<https://blog.csdn.net/saltriver/article/details/52901203>

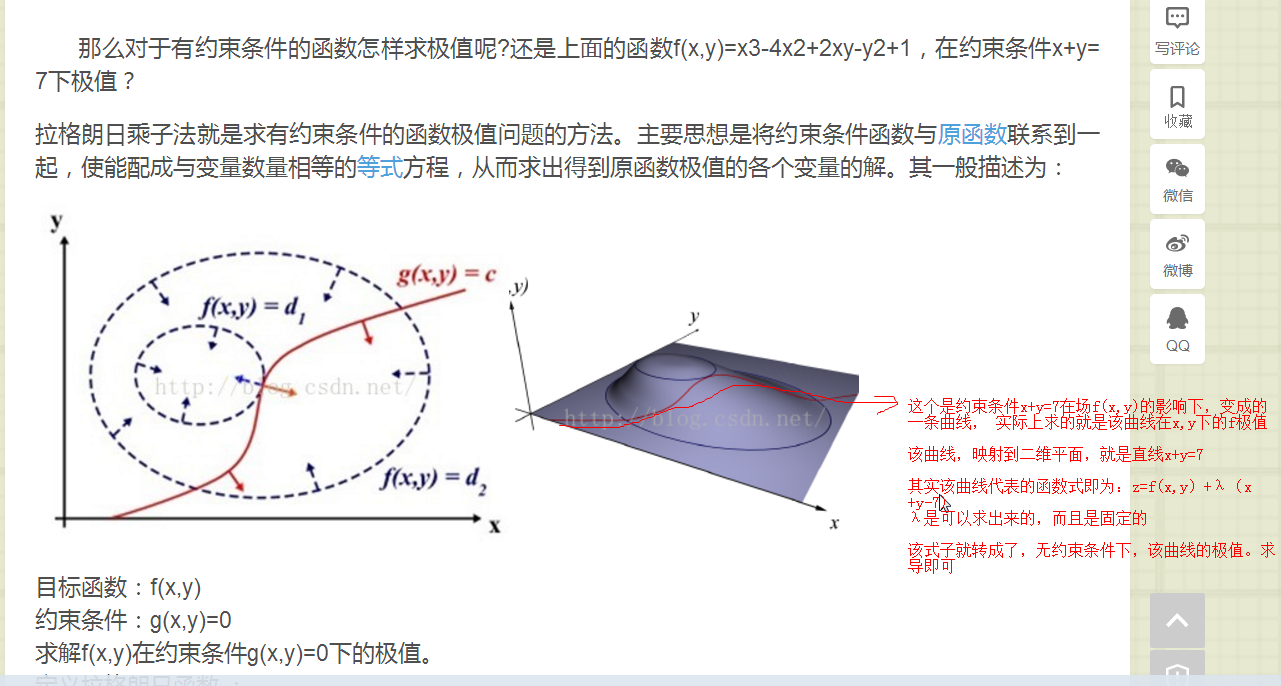
实际上，有约束的对平面求极值， 就是对“约束平面和原始平面相交，得到的曲线”，求无条件极值。

若求y= min f(x,y)

约束条件为：g(x,y)=0

则，求极值即为， 对曲线 z = f(x,y) +λg(x,y) 求“无约束”极值

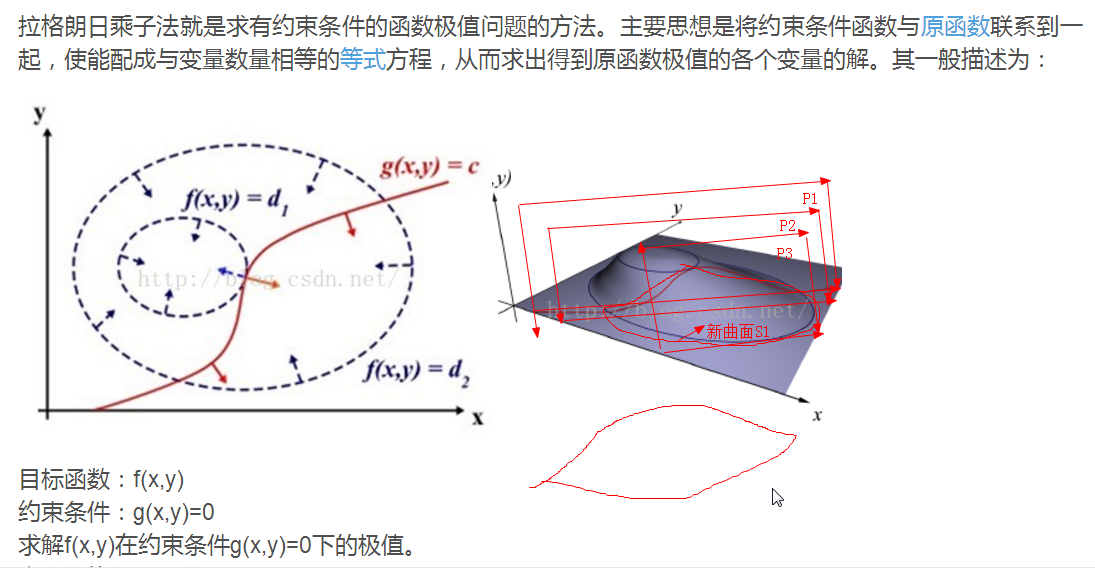
很显然，有约束的条件极值，一般来说，不可能



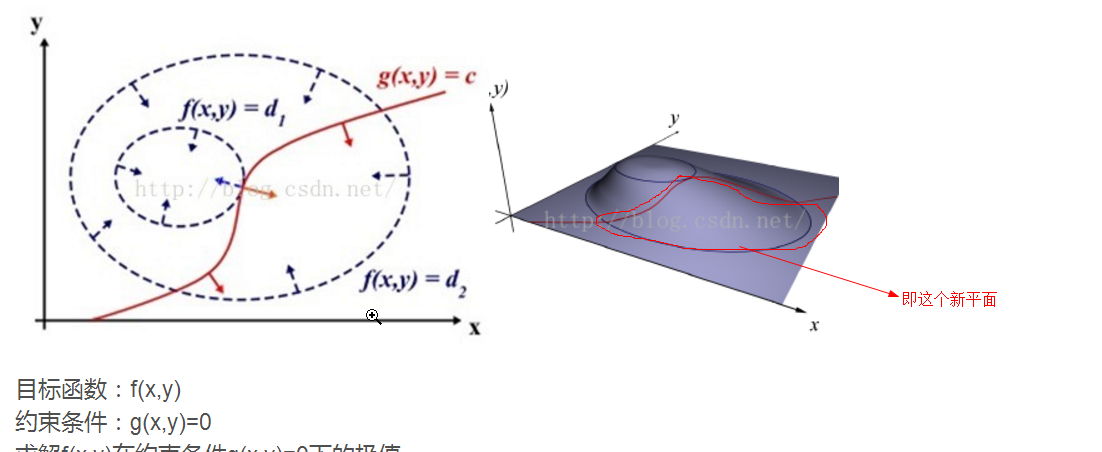
那如果约束变成是x+y-7>=0,

那也就是对平面右边的区域、和曲面相交， 得到的新曲面，对该新曲面，求（无约束极值）

新曲面如下图：



即下面的这个曲面：



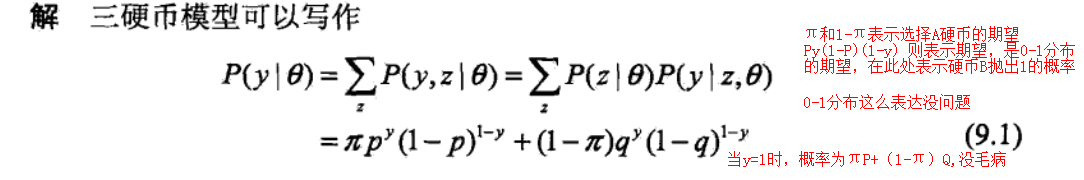
新的约束条件可转成： x+y-7 -（ξ+）=0

这个ξ+就是个松弛变量， 其>0,如果直接看成变量，

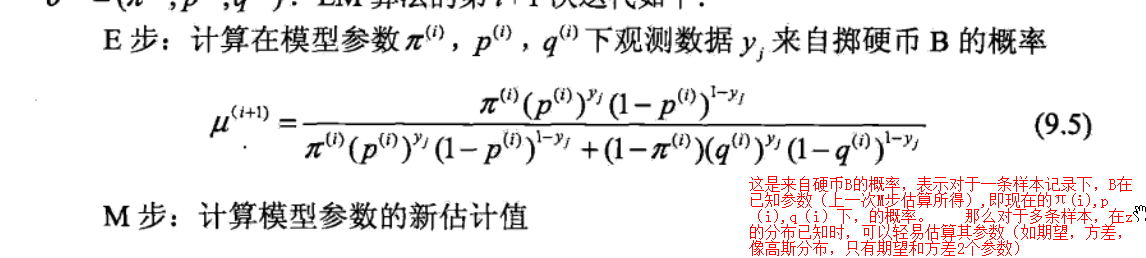
该新的约束条件， 实际上是1个3维空间， 该空间是由无限个平面构成，这些平面都在（x+y-7=0）的上边。 空间与平面，生成“曲面”

所以，最后就是新曲面z: z=h(x,y) 的无约束极值问题。

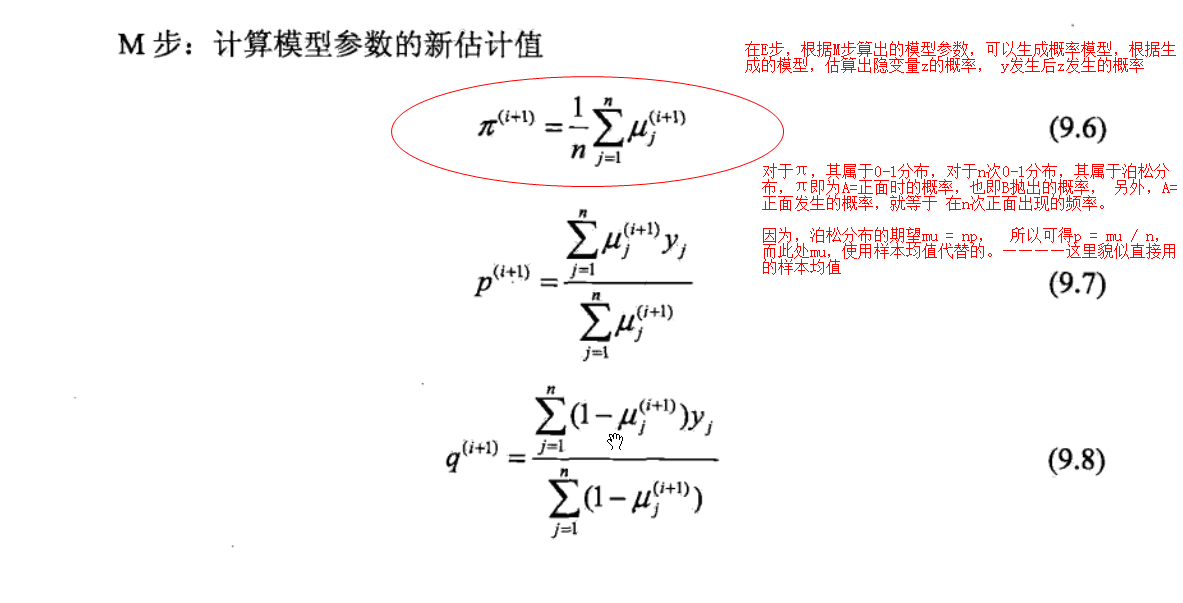
## EM算法：



E步：

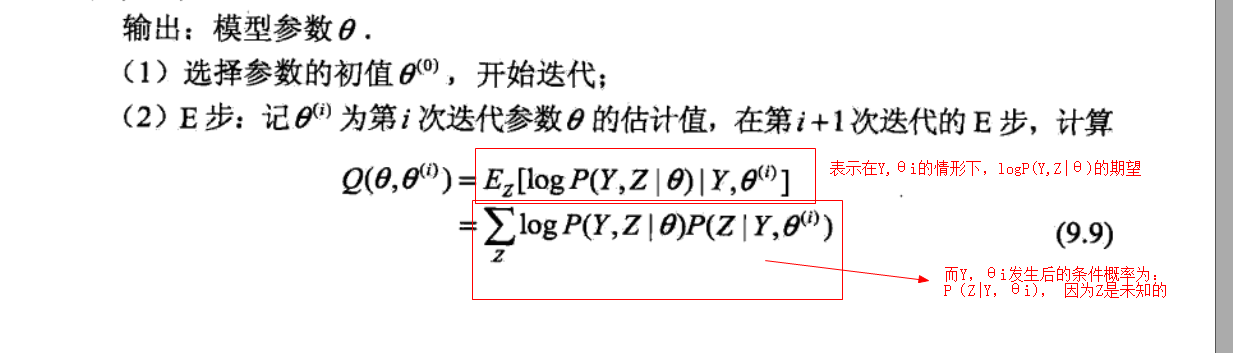


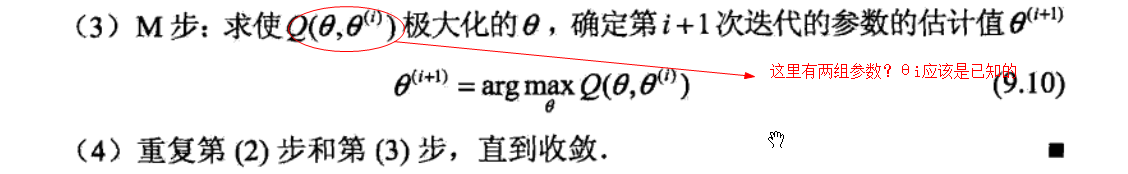
M步：



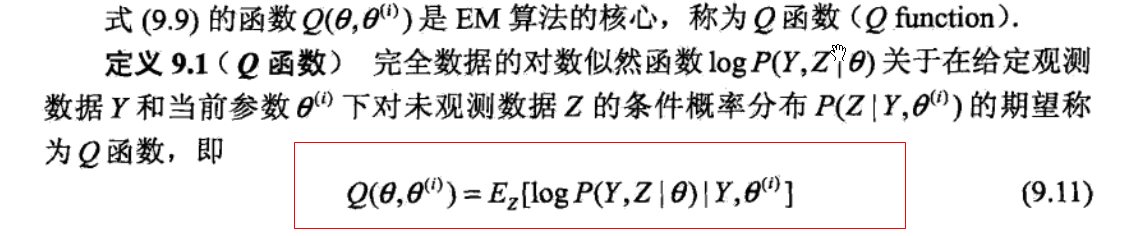
上面的M步解释不太到位：

看下面的解释：

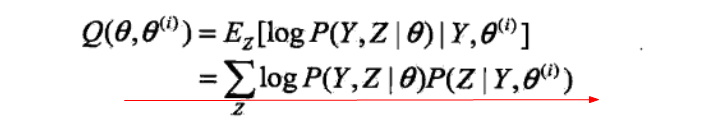




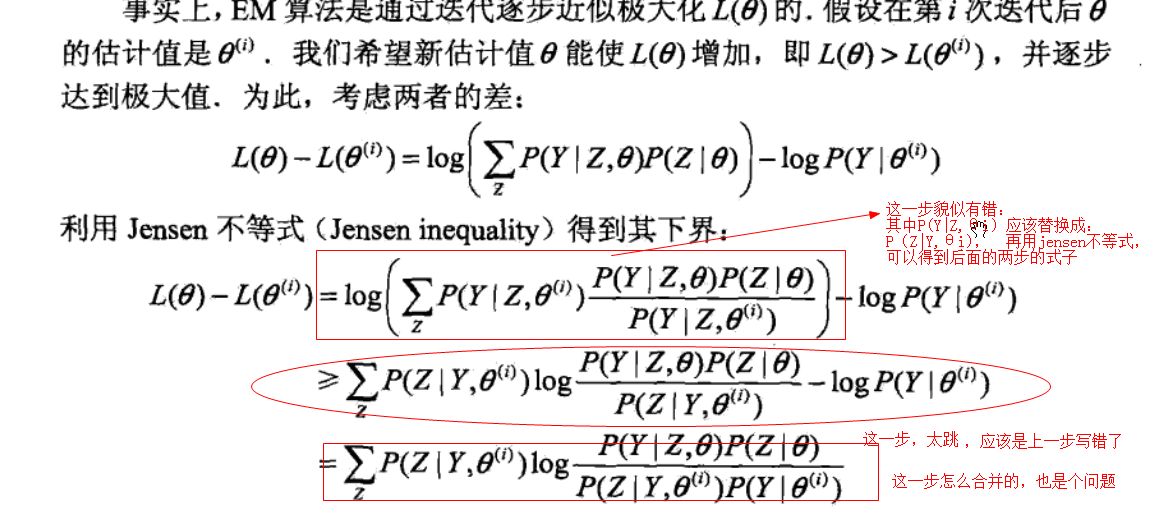
Q函数，EM的核心算法——即怎么计算Q函数：



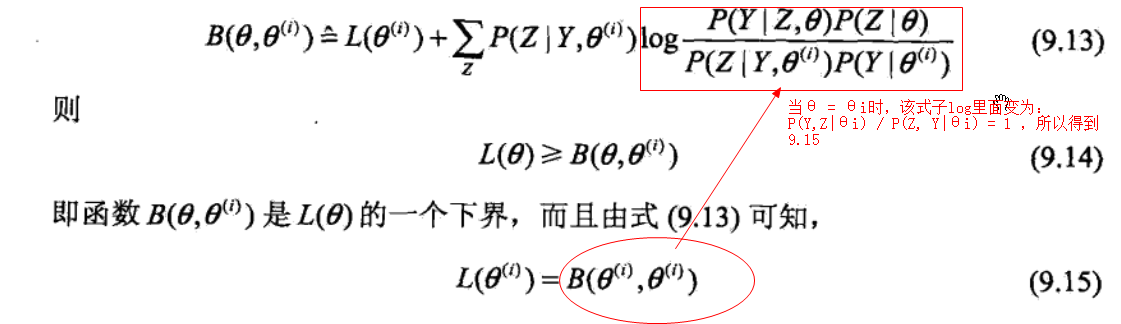
也就是上面的：



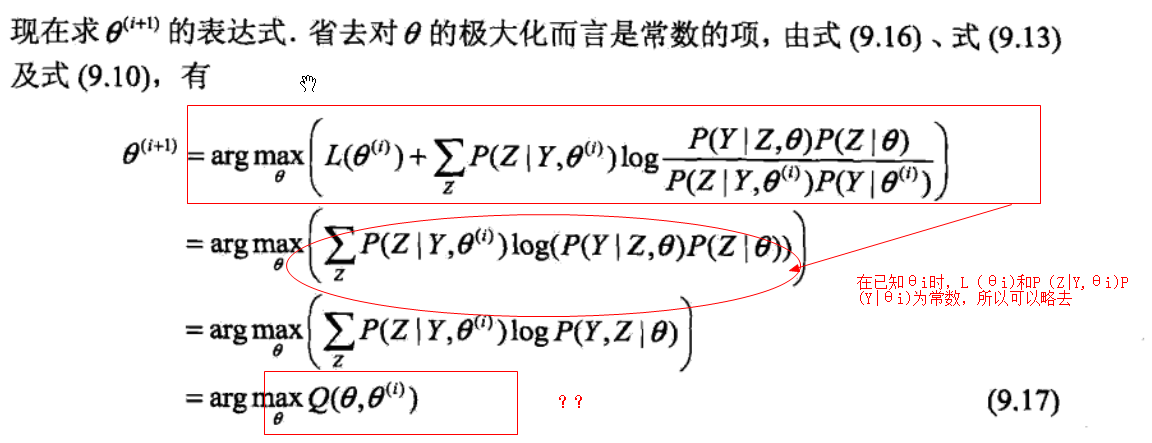
两个有疑问的推导？？



关于B函数：



进一步推导：



最后一步的问号可以去掉，就是9.11式子中的Q函数

那么9.12到9.17的推导，其实就是：

为了最大化9.12式子， 其等价于求9.17， 即等价与最大化Q函数：

9.12式子好理解：就是最大化“完全数据”的似然函数——很朴素的思想

那总体一次迭代的步骤：

1. E步： 计算出Q函数（第一次计算，是随机初始化参数，或者某种方法给定初始θ0, 此时的参数必然不是最优，所以需要后面迭代更新）
2. M步：最大化Q函数，从而得到新的参数θ（i+1）

循环E和M

高斯混合计算：

