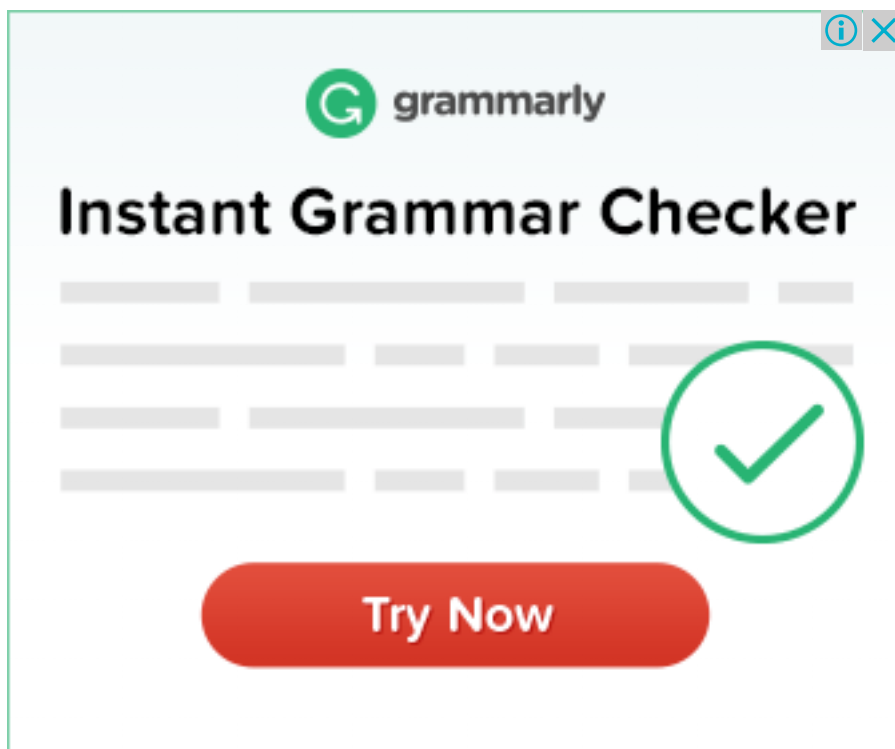


机器学习常用算法总结

 编程技术

机器学习&数据挖掘

前言：

找工作时（IT行业），除了常见的软件开发以外，机器学习岗位也可以当作是一个选择，不少计算机方向的研究生都会接触这个，如果你的研究方向是机器学习/数据挖掘之类，且又对其非常感兴趣的话，可以考虑考虑该岗位，毕竟在机器智能没达到人类水平之前，机器学习可以作为一种重要手段，而随着科技的不断发展，相信这方面的人才需求也会越来越大。

纵观IT行业的招聘岗位，机器学习之类的岗位还是挺少的，国内大点的公司里百度，阿里，腾讯，网易，搜狐，华为（华为的岗位基本都是随机分配，机器学习等岗位基本面向的是博士）等会有相关职位，另外一些国内的中小型企业 and 外企也会招一小部分。当然了，其中大部分还是百度北京要人最多，上百人。阿里的算法岗位很大一部分也是搞机器学习相关的。

下面是本人在找机器学习岗位工作时，总结的常见机器学习算法（主要是一些常规分类器）大概流程和主要思想，希望对大家找机器学习岗位时有点帮助。实际上在面试过程中，懂这些算法的基本思想和大概流程是远远不够的，那些面试官往往问的都是一些公司内部业务中的课题，往往要求你不仅要懂得这些算法的理论过程，而且要非常熟悉怎样使用它，什么场合用它，算法的优缺点，以及调参经验等等。说白了，就是既要会点理论，也要会点应用，既要有点深度，也要有点广度，否则运气不好的话很容易就被刷掉，因为每个面试官爱好不同。

朴素贝叶斯：

有以下几个地方需要注意：

1. 如果给出的特征向量长度可能不同，这是需要归一化为通长度的向量（这里以文本分类为例），比如说是句子单词的话，则长度为整个词汇量的长度，对应位置是该单词出现的次数。

2. 计算公式如下：

$$p(c_i|w) = \frac{p(w|c_i)p(c_i)}{p(w)}$$

其中一项条件概率可以通过朴素贝叶斯条件独立展开。要注意一点就是

$$p(w|c_i)$$

的计算方法，而由朴素贝叶斯的前提假设可知，

$$p(w_0, w_1, w_2, \dots, w_N | c_i)$$

=

$$p(w_0 | c_i) p(w_1 | c_i) p(w_2 | c_i) \dots p(w_N | c_i)$$

，因此一般有两种，一种是在类别为 c_i 的那些样本集中，找到 w_j 出现次数的总和，然后除以该样本的总和；第二种方法是类别为 c_i 的那些样本集中，找到 w_j 出现次数的总和，然后除以该样本中所有特征出现次数的总和。

3. 如果

$$p(w|c_i)$$

中的某一项为0，则其联合概率的乘积也可能为0，即2中公式的分子为0，为了避免这种现象出现，一般情况下会将这一项初始化为1，当然为了保证概率相等，分母应对应初始化为2（这里因为是2类，所以加2，如果是k类就需要加k，术语上叫做laplace光滑，分母加k的原因是使之满足全概率公式）。

朴素贝叶斯的优点：

对小规模的数据表现很好，适合多分类任务，适合增量式训练。

缺点：

对输入数据的表达形式很敏感。

决策树：

决策树中很重要的一点就是选择一个属性进行分枝，因此要注意一下信息增益的计算公式，并深入理解它。

信息熵的计算公式如下：

$$H = -\sum_{i=1}^n p(x_i) \log_2 p(x_i)$$

其中的n代表有n个分类类别（比如假设是2类问题，那么n=2）。分别计算这2类样本在总样本中出现的概率p1和p2，这样就可以计算出未选中属性分枝前的信息熵。

现在选中一个属性xi用来进行分枝，此时分枝规则是：如果xi=vx的话，将样本分到树的一个分支；如果不相等则进入另一个分支。很显然，分支中的样本很有可能包括2个类别，分别计算这2个分支的熵H1和H2,计算出分枝后的总信息熵H' = p1*H1+p2*H2.，则此时的信息增益ΔH=H-H'。以信息增益为原则，把所有的属性都测试一边，选择一个使增益最大的属性作为本次分枝属性。

决策树的优点：

计算量简单，可解释性强，比较适合处理有缺失属性值的样本，能够处理不相关的特征；

缺点：

容易过拟合（后续出现了随机森林，减小了过拟合现象）；

Logistic回归：

Logistic是用来分类的，是一种线性分类器，需要注意的地方有：

1. logistic函数表达式为：

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

where

$$g(z) = \frac{1}{1 + e^{-z}}$$

其导数形式为：

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\ &= \frac{1}{(1 + e^{-z})^2} (e^{-z}) \\ &= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})}\right) \\ &= g(z)(1 - g(z)). \end{aligned}$$

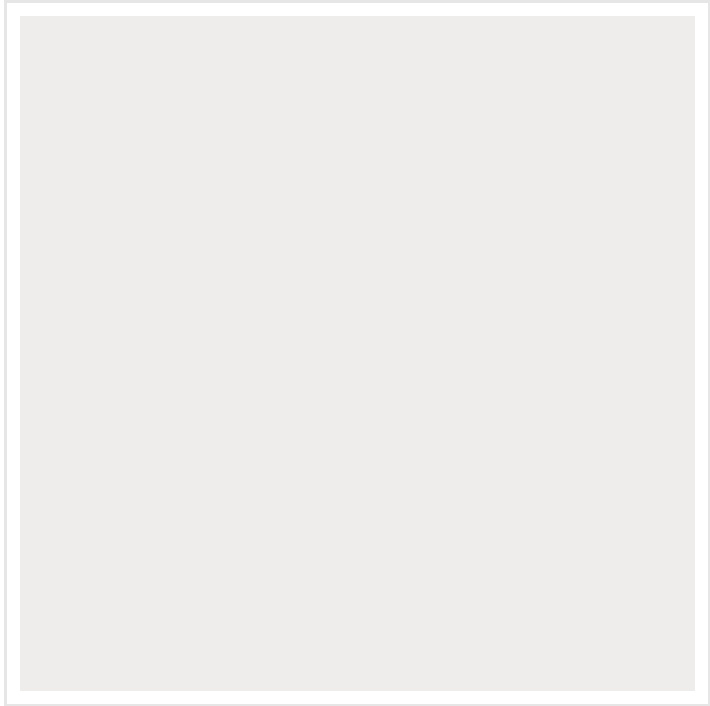
2. logsitc回归方法主要是用最大似然估计来学习的，所以单个样本的后验概率为：

$$p(y | x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

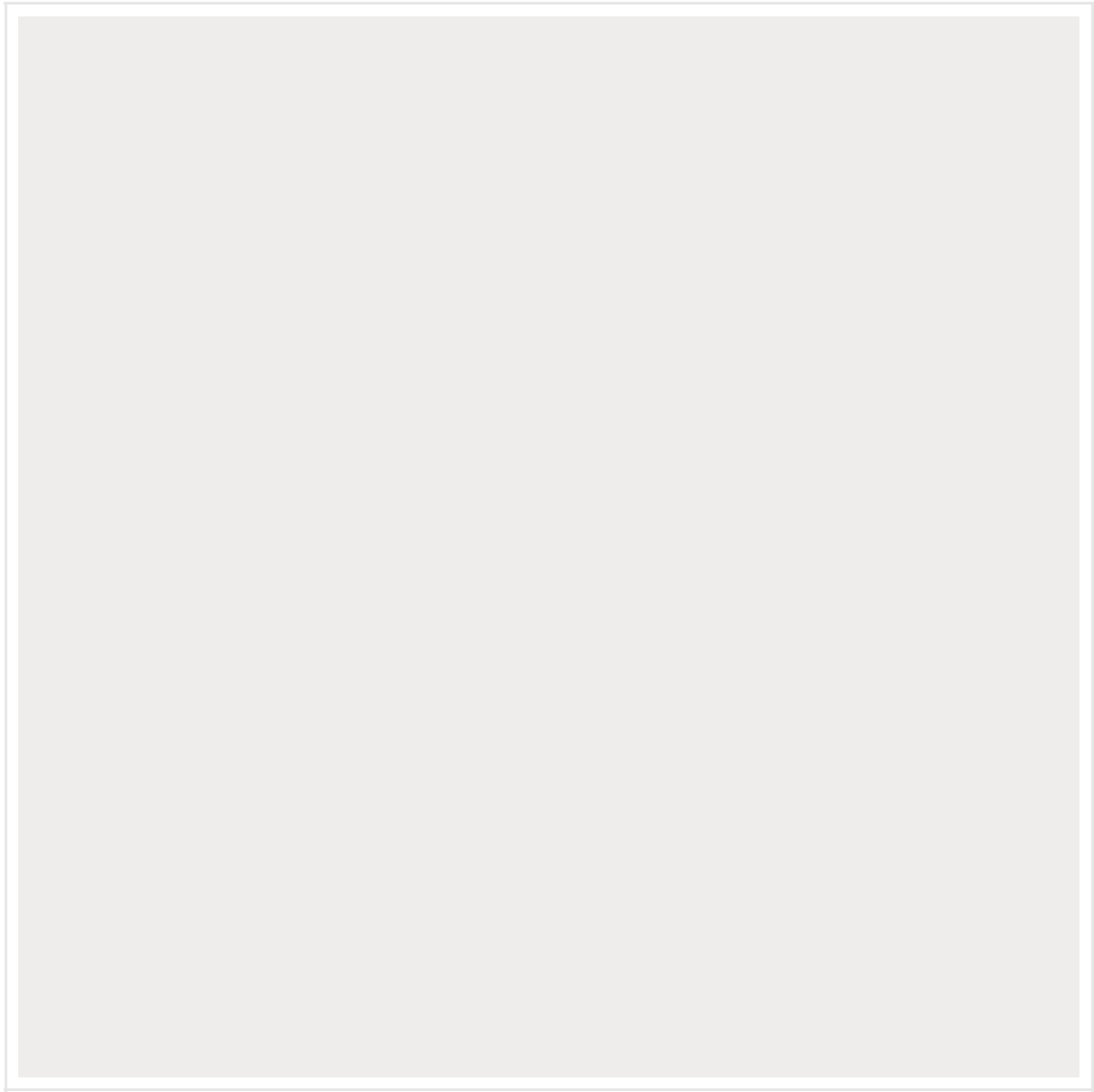
到整个样本的后验概率：

$$\begin{aligned} L(\theta) &= p(\vec{y} | X; \theta) \\ &= \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta) \\ &= \prod_{i=1}^m (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

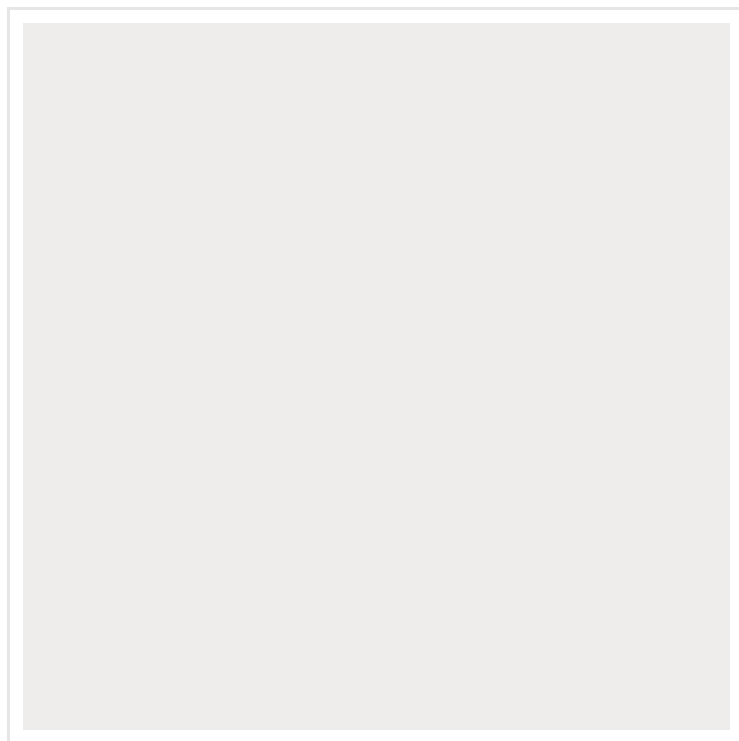
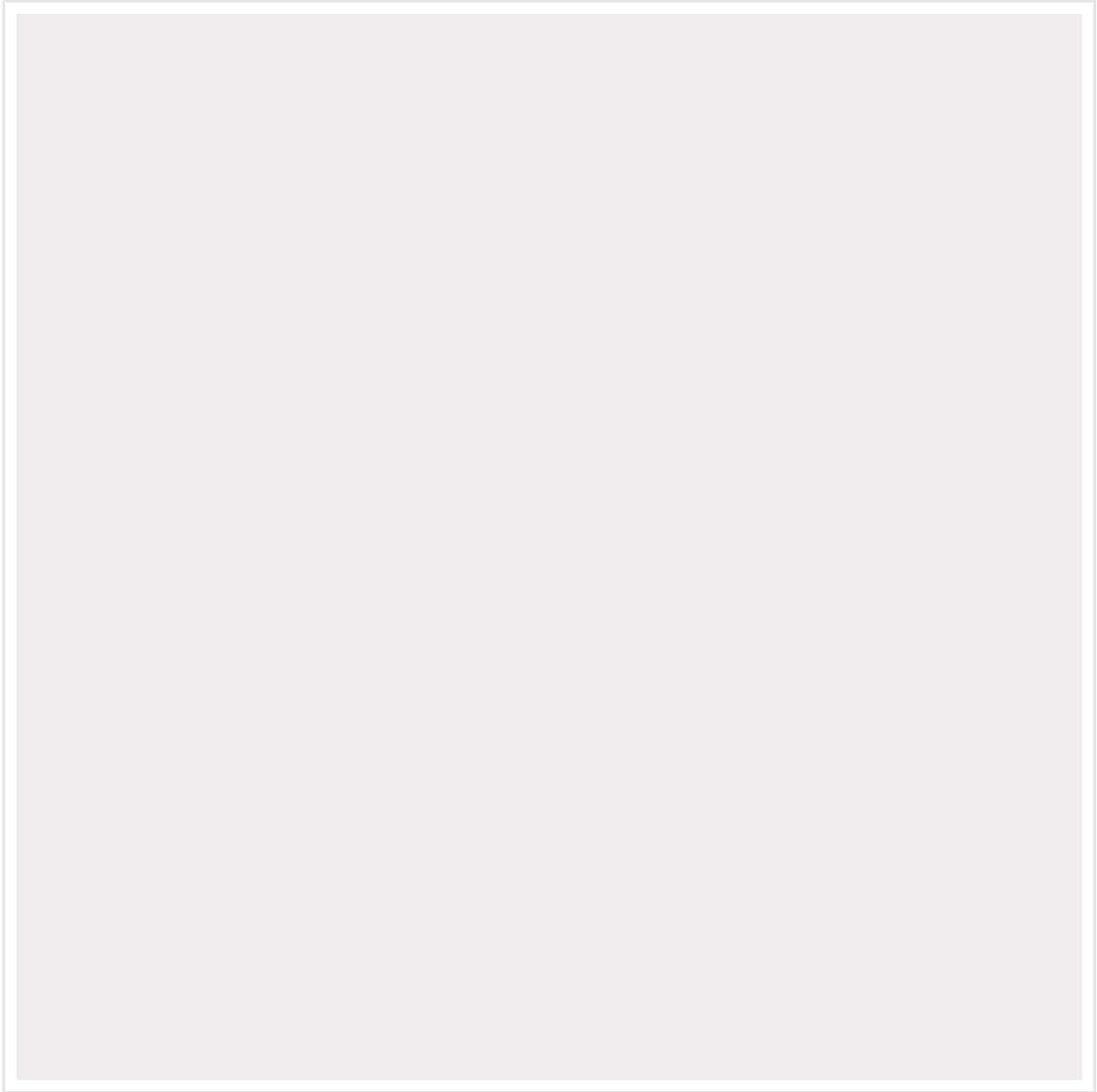
其中：



通过对数进一步化简为：



3. 其实它的loss function为 $-l(\theta)$ ，因此我们需使loss function最小，可采用梯度下降法得到。梯度下降法公式为：



Logistic回归优点：

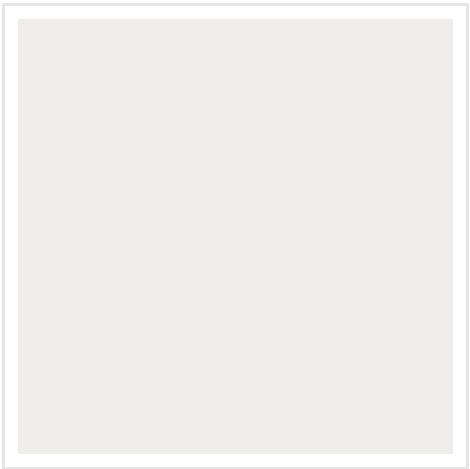
- 1、实现简单；
- 2、分类时计算量非常小，速度很快，存储资源低；

缺点：

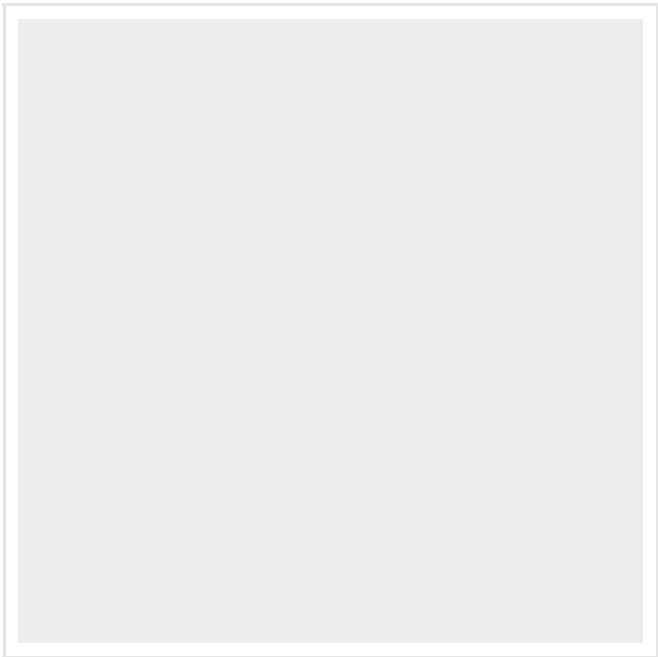
- 1、容易欠拟合，一般准确度不太高
- 2、只能处理两分类问题（在此基础上衍生出来的softmax可以用于多分类），且必须线性可分；

线性回归：

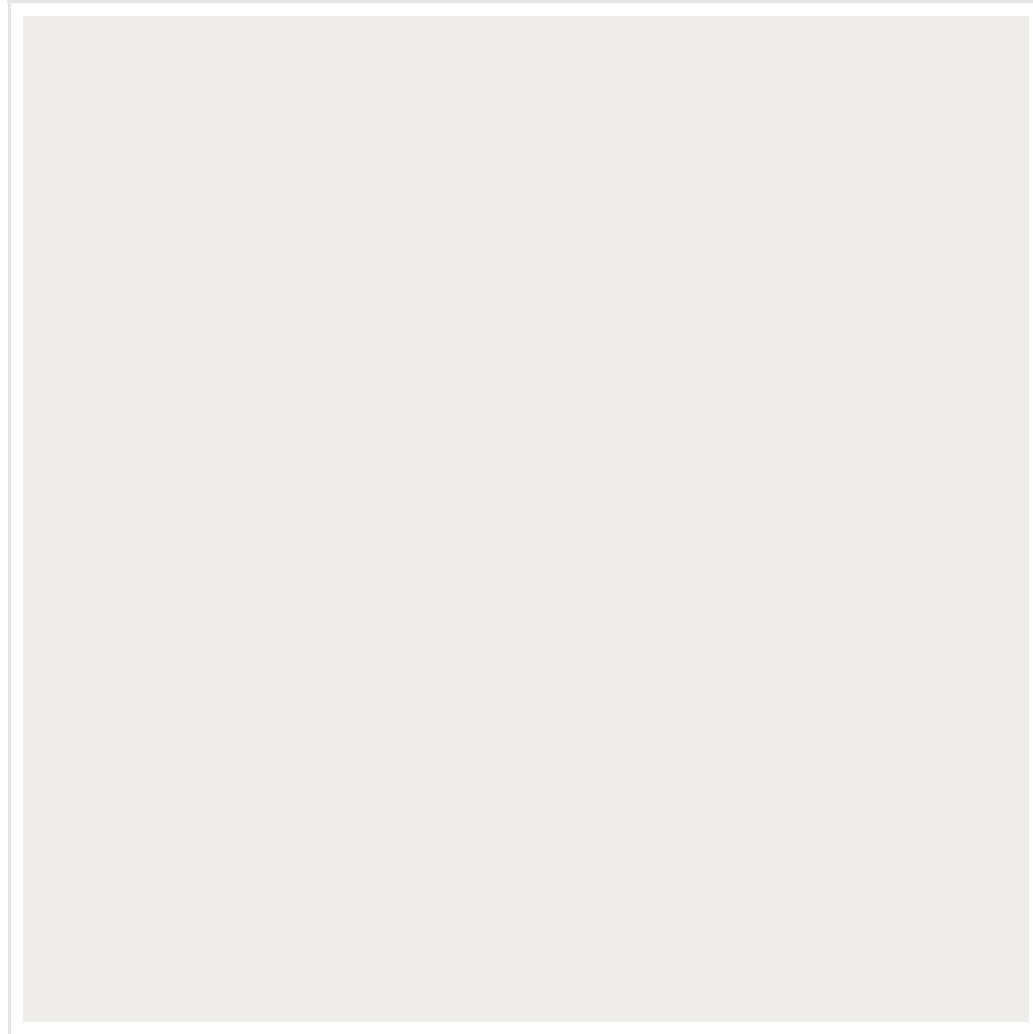
线性回归才是真正用于回归的，而不像logistic回归是用于分类，其基本思想是用梯度下降法对最小二乘法形式的误差函数进行优化，当然也可以用normal equation直接求得参数的解，结果为：



而在LWLR（局部加权线性回归）中，参数的计算表达式为:



因为此时优化的是：



由此可见LWLR与LR不同，LWLR是一个非参数模型，因为每次进行回归计算都要遍历训练样本至少一次。

线性回归优点：

实现简单，计算简单；

缺点：

不能拟合非线性数据；

KNN算法：

KNN即最近邻算法，其主要过程为：

1. 计算训练样本和测试样本中每个样本点的距离（常见的距离度量有欧式距离，马氏距离等）；
2. 对上面所有的距离值进行排序；
3. 选前k个最小距离的样本；
4. 根据这k个样本的标签进行投票，得到最后的分类类别；

如何选择一个最佳的K值，这取决于数据。一般情况下，在分类时较大的K值能够减小噪声的影响

。但会使类别之间的界限变得模糊。一个较好的K值可通过各种启发式技术来获取，比如，交叉验证。另外噪声和非相关性特征向量的存在会使K近邻算法的准确性减小。

近邻算法具有较强的一致性结果。随着数据趋于无限，算法保证错误率不会超过贝叶斯算法错误率的两倍。对于一些好的K值，K近邻保证错误率不会超过贝叶斯理论误差率。

注：马氏距离一定要先给出样本集的统计性质，比如均值向量，协方差矩阵等。关于马氏距离的介绍如下：

马氏距离是由印度统计学家马哈拉诺比斯 (P. C. Mahalanobis) 提出的，表示数据的协方差距离。它是一种有效的计算两个未知样本集的相似度的方法。与欧氏距离不同的是它考虑到各种特性之间的联系（例如：一条关于身高的信息会带来一条关于体重的信息，因为两者是有关联的）并且是尺度无关的 (scale-invariant)，即独立于测量尺度。 对于一个均值为 $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_p)^T$ ，协方差矩阵为 Σ 的多变量向量 $x = (x_1, x_2, x_3, \dots, x_p)^T$ ，其马氏距离为

$$D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

马氏距离也可以定义为两个服从同一分布并且其协方差矩阵为 Σ 的随机变量 \vec{x} 与 \vec{y} 的差异程度：

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

如果协方差矩阵为单位矩阵，马氏距离就简化为欧氏距离；如果协方差矩阵为对角阵，其也可称为 正交化的欧氏距离。

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^p \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

其中 σ_i 是 x_i 的标准差。

机器学习联盟

KNN算法的优点：

1. 思想简单，理论成熟，既可以用来做分类也可以用来做回归；
2. 可用于非线性分类；
3. 训练时间复杂度为O(n)；
4. 准确度高，对数据没有假设，对outlier不敏感；

缺点：

1. 计算量大；
2. 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；
3. 需要大量的内存；


SVM：

要学会如何使用libsvm以及一些参数的调节经验，另外需要理清楚svm算法的一些思路：

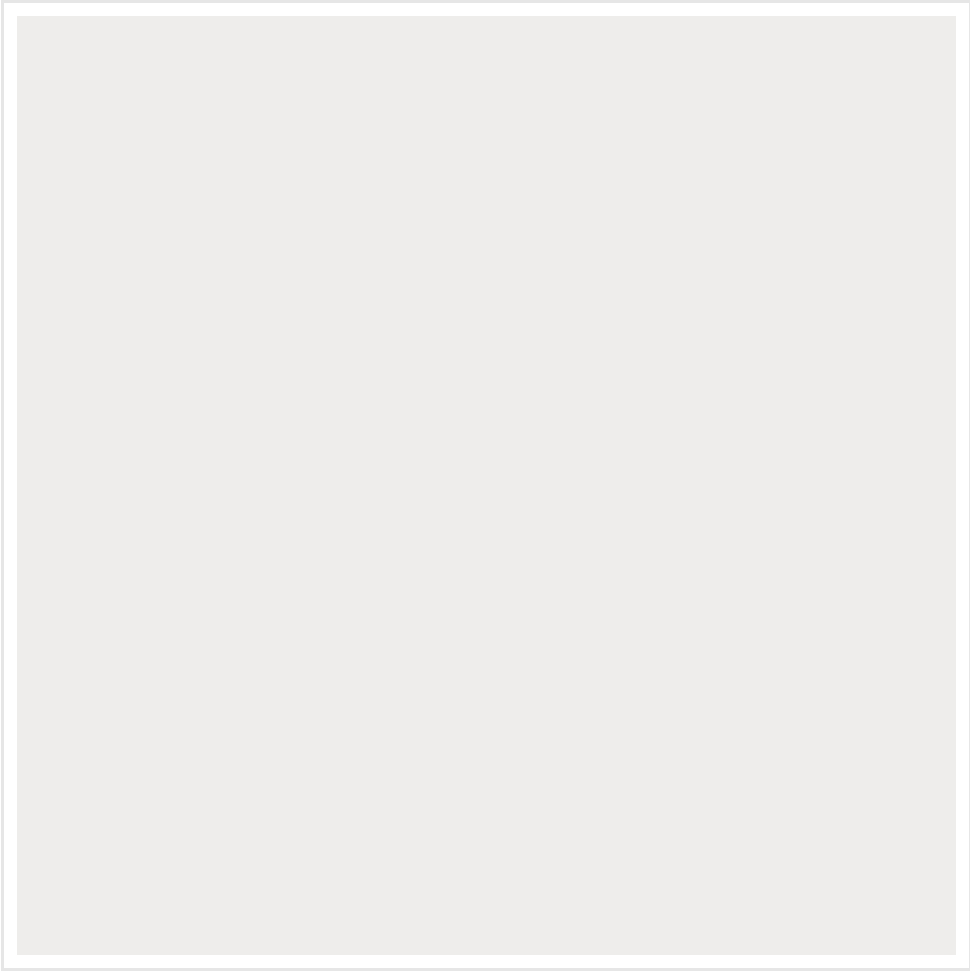
1. svm中的最优分类面是对所有样本的几何裕量最大（为什么要选择最大间隔分类器，请从数学

角度上说明？网易深度学习岗位面试过程中有被问到。答案就是几何间隔与样本的误分次数间存在关系：

误分次数 $\leq \left(\frac{2R}{\delta}\right)^2$

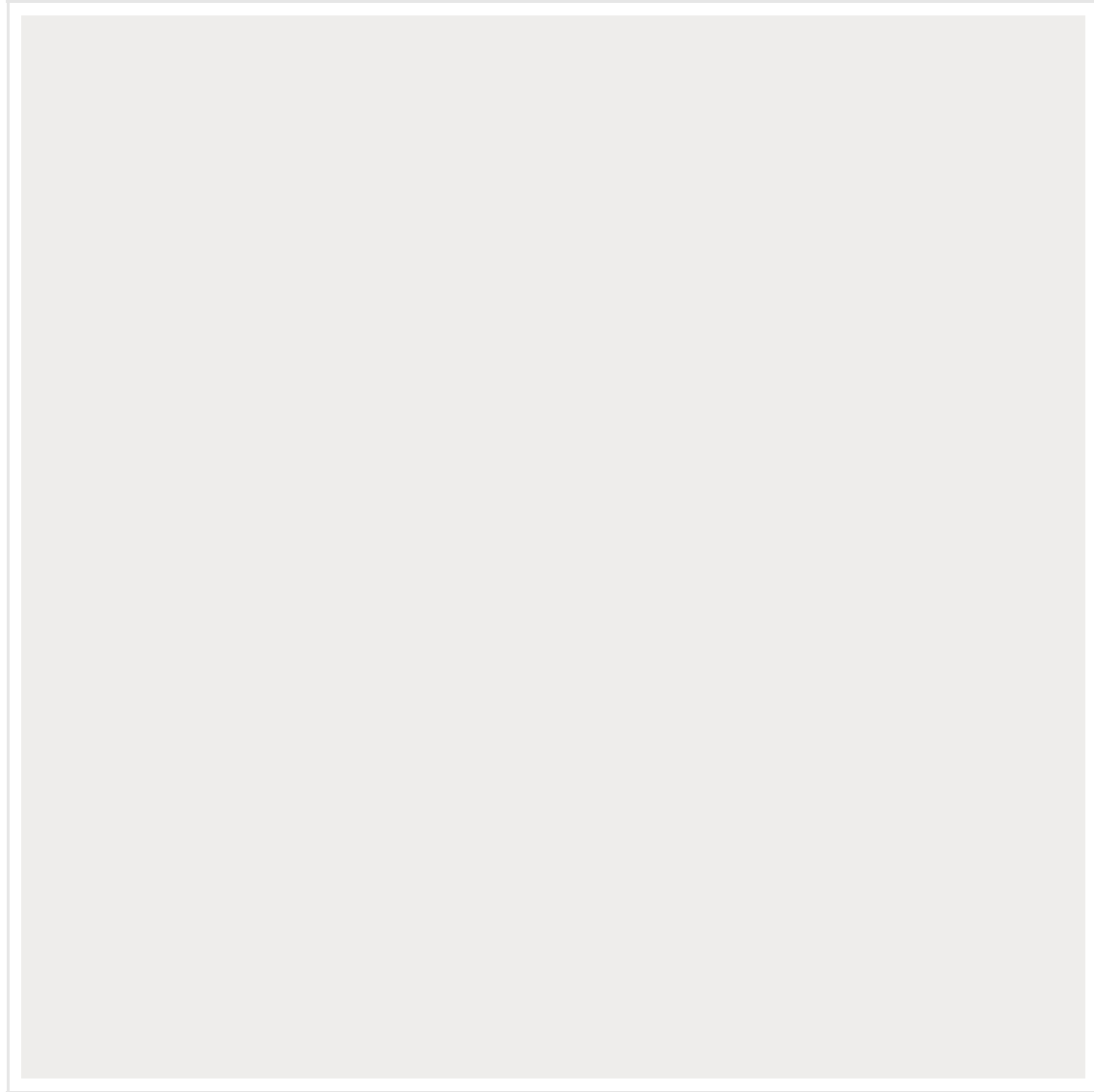
 机器学习联盟

，其中的分母就是样本到分类间隔距离，分子中的R是所有样本中的最长向量值），即：



经过一系列推导可得为优化下面原始目标：

2. 下面来看看拉格朗日理论：



可以将1中的优化目标转换为拉格朗日的形式（通过各种对偶优化，KKD条件），最后目标函数为：

$$\square$$

我们只需要最小化上述目标函数，其中的 α 为原始优化问题中的不等式约束拉格朗日系数。

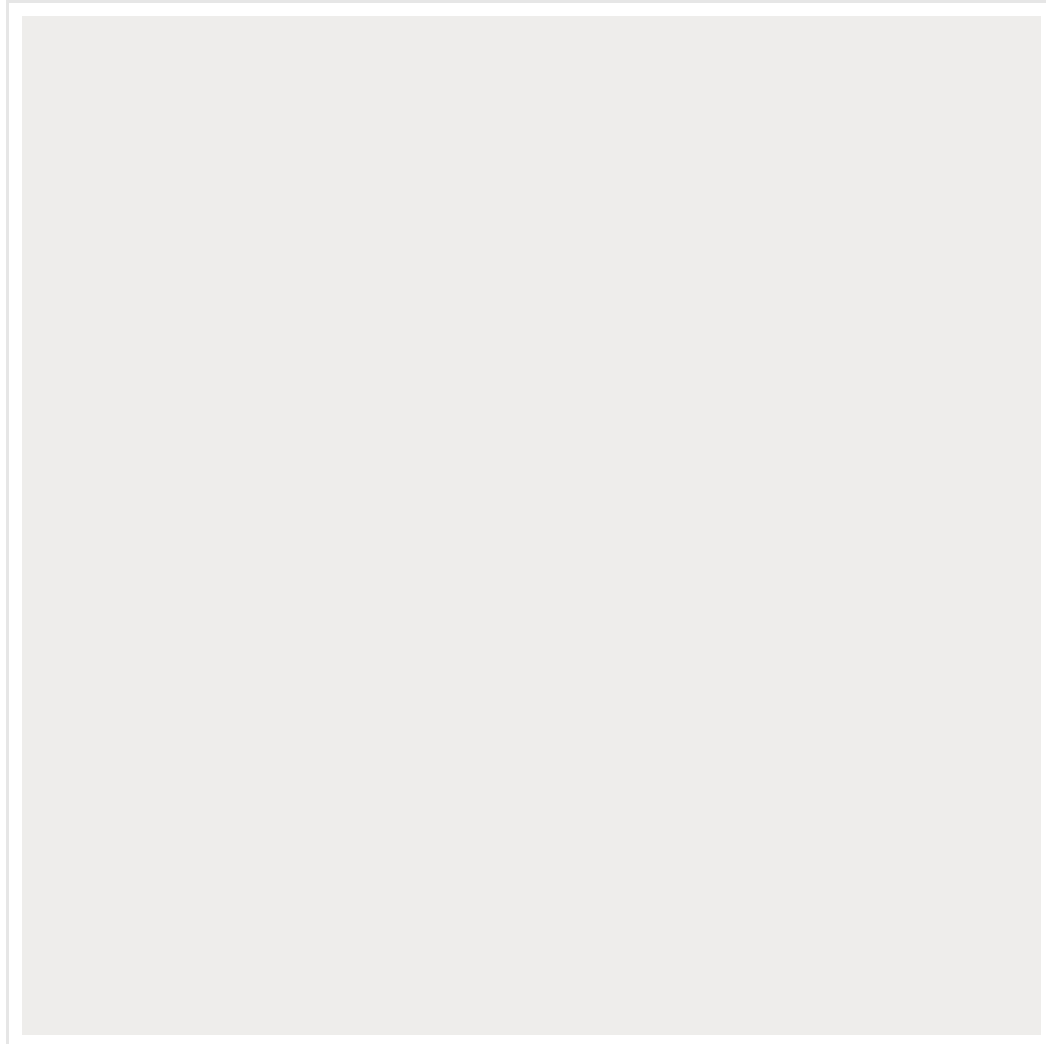
3. 对2中最后的式子分别w和b求导可得：

$$\square$$

$$\square$$

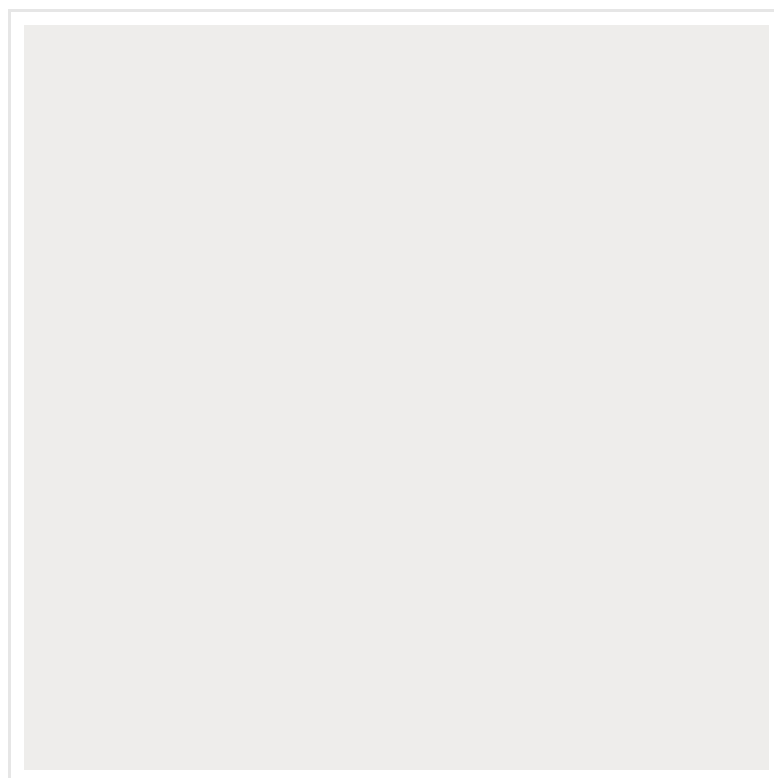
由上面第1式子可以知道，如果我们优化出了 α ，则直接可以求出w了，即模型的参数搞定。而上面第2个式子可以作为后续优化的一个约束条件。

4. 对2中最后一个目标函数用对偶优化理论可以转换为优化下面的目标函数：



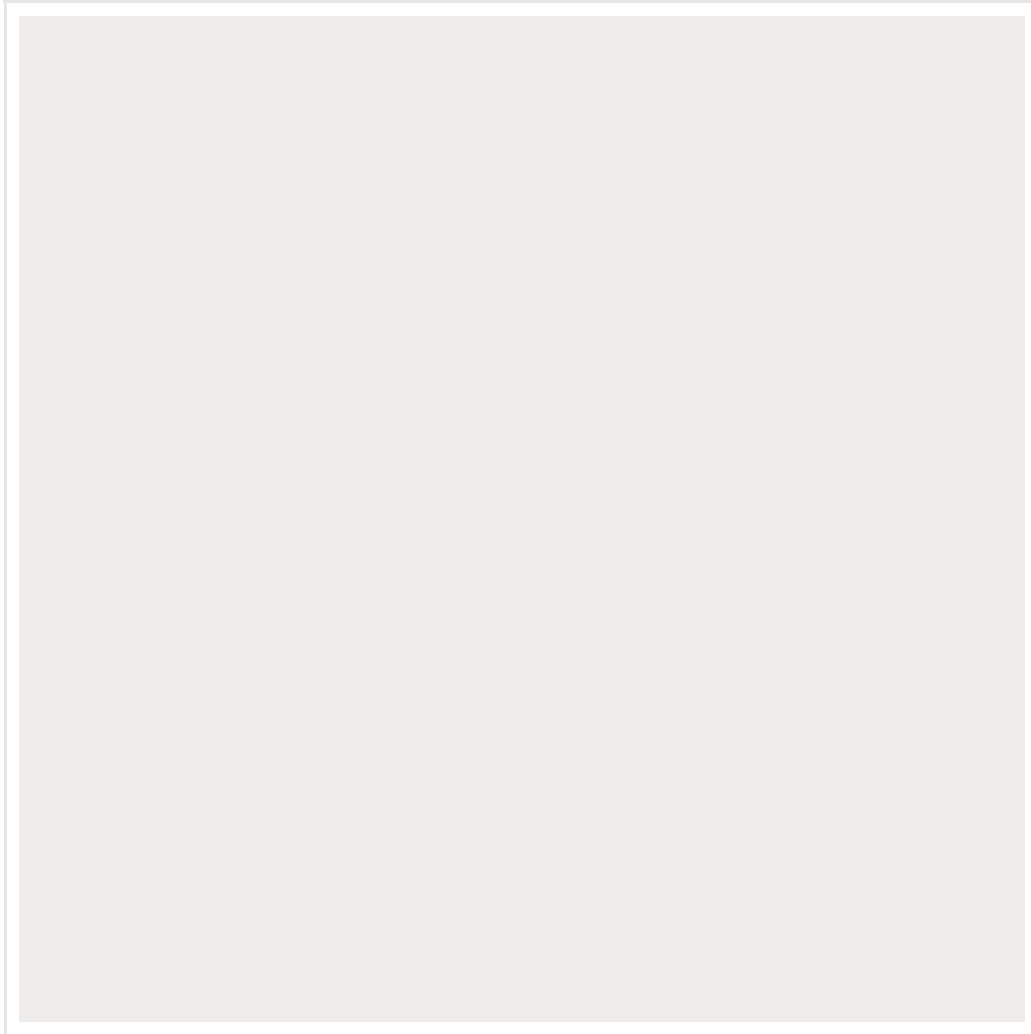
而这个函数可以用常用的优化方法求得 α ，进而求得 w 和 b 。

5. 按照道理，svm简单理论应该到此结束。不过还是要补充一点，即在预测时有：

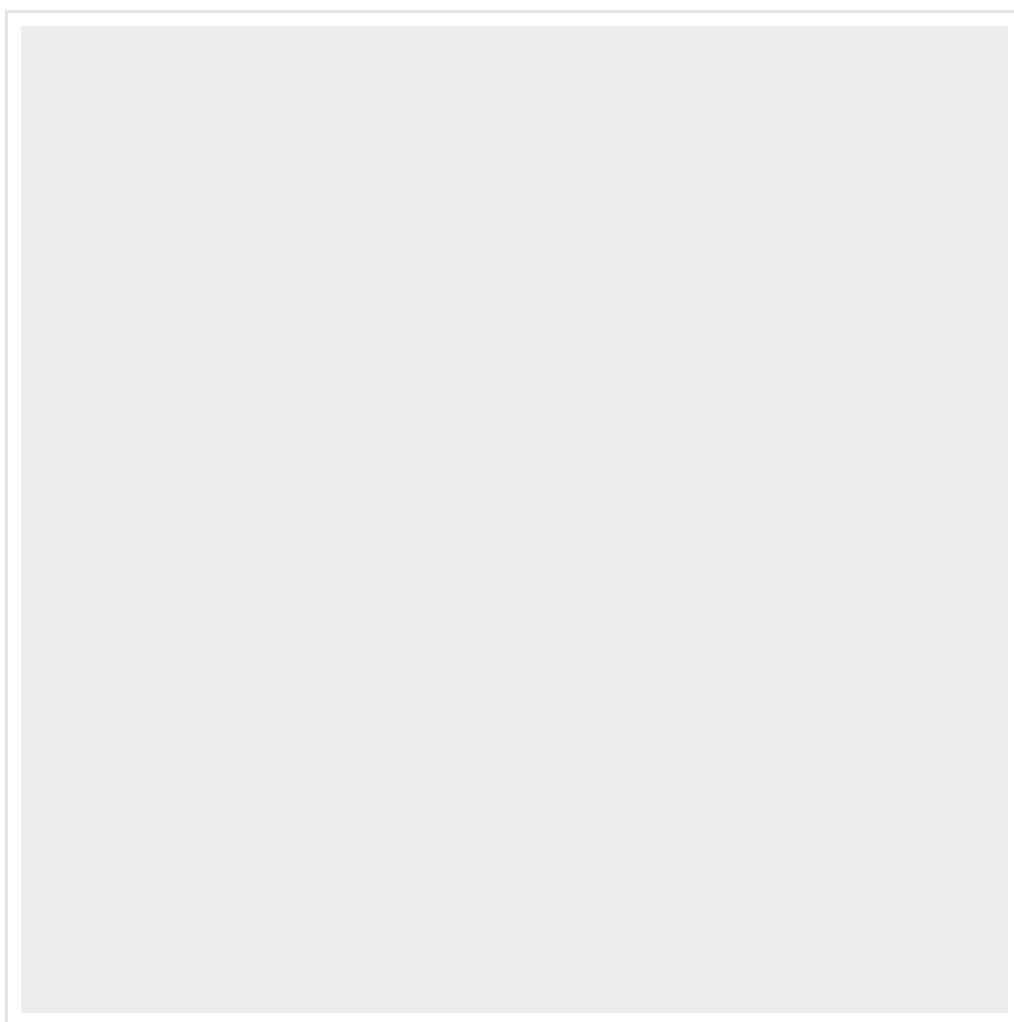


那个尖括号我们可以用核函数代替，这也是svm经常和核函数扯在一起的原因。

6. 最后是关于松弛变量的引入，因此原始的目标优化公式为：



此时对应的对偶优化公式为：



与前面的相比只是 α 多了个上界。

***SVM*算法优点：**

可用于线性/非线性分类，也可以用于回归；

低泛化误差；

容易解释；

计算复杂度较低；

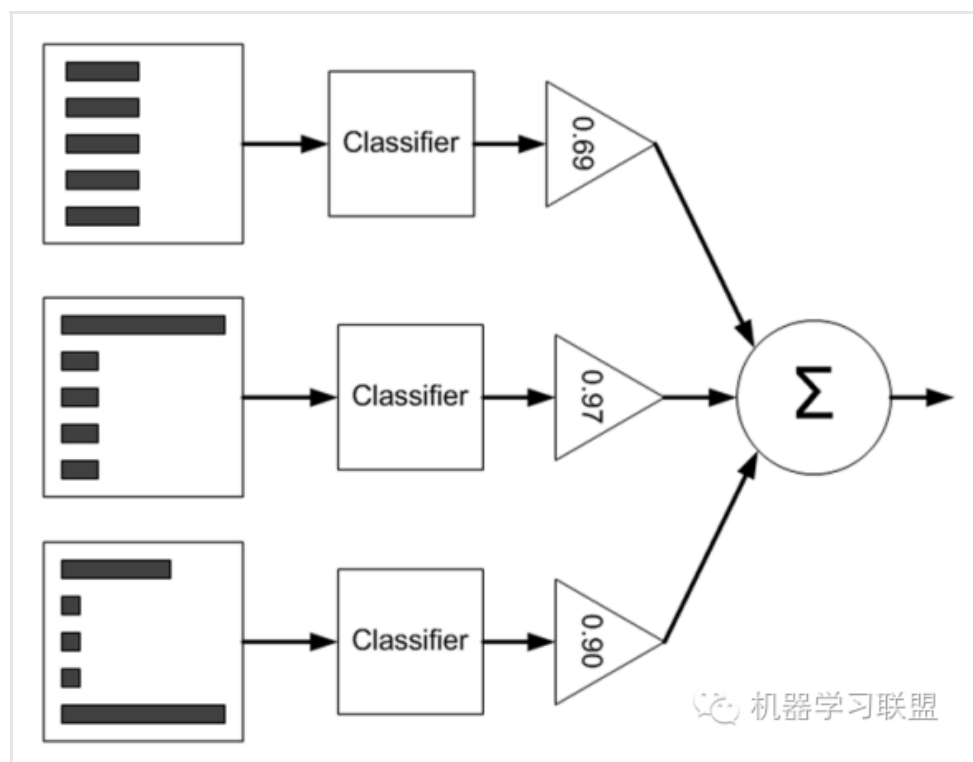
缺点：

对参数和核函数的选择比较敏感；

原始的SVM只比较擅长处理二分类问题；

Boosting：

主要以Adaboost为例，首先来看看Adaboost的流程图，如下：



从图中可以看到，在训练过程中我们需要训练出多个弱分类器（图中为3个），每个弱分类器是由不同权重的样本（图中为5个训练样本）训练得到（其中第一个弱分类器对应输入样本的权值是一样的），而每个弱分类器对最终分类结果的作用也不同，是通过加权平均输出的，权值见上图中三角形里面的数值。那么这些弱分类器和其对应的权值是怎样训练出来的呢？

下面通过一个例子来简单说明。

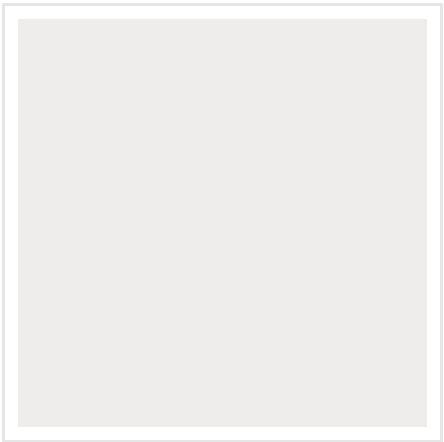
书中（machine learning in action）假设的是5个训练样本，每个训练样本的维度为2，在训练第一个分类器时5个样本的权重各为0.2. 注意这里样本的权值和最终训练的弱分类器组对应的权值 α 是不同的，样本的权重只在训练过程中用到，而 α 在训练过程和测试过程都有用到。

现在假设弱分类器是带一个节点的简单决策树，该决策树会选择2个属性（假设只有2个属性）的一个，然后计算出这个属性中的最佳值用来分类。

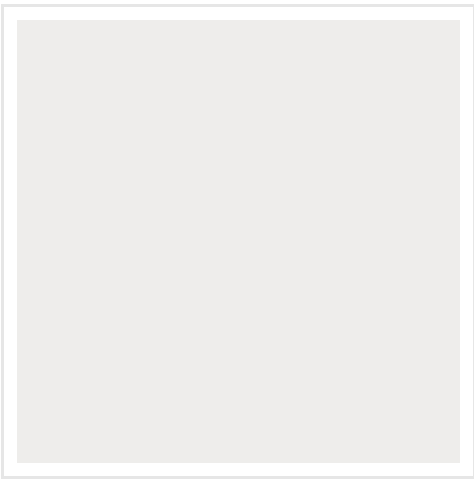
Adaboost的简单版本训练过程如下：

1. 训练第一个分类器，样本的权值D为相同的均值。通过一个弱分类器，得到这5个样本（请对应书中的例子来看，依旧是machine learning in action）的分类预测标签。与给出的样本真实标签对比，就可能出现误差(即错误)。如果某个样本预测错误，则它对应的错误值为该样本的权重，如果分类正确，则错误值为0. 最后累加5个样本的错误率之和，记为 ϵ 。

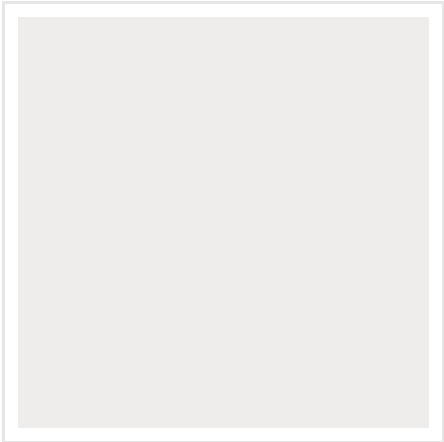
2. 通过 ϵ 来计算该弱分类器的权重 α ，公式如下：



3. 通过 α 来计算训练下一个弱分类器样本的权重D，如果对应样本分类正确，则减小该样本的权重，公式为：



如果样本分类错误，则增加该样本的权重，公式为：



4. 循环步骤1,2,3来继续训练多个分类器，只是其D值不同而已。

测试过程如下：

输入一个样本到训练好的每个弱分类中，则每个弱分类都对应一个输出标签，然后该标签乘以对应的 α ，最后求和得到值的符号即为预测标签值。

Boosting算法的优点：

低泛化误差；

容易实现，分类准确率较高，没有太多参数可以调；

缺点：

对outlier比较敏感；

聚类：

根据聚类思想划分：

1. 基于划分的聚类:

K-means, k-medoids(每一个类别中找一个样本点来代表),CLARANS.

k-means是使下面的表达式值最小：

$$\sum_{i=1}^n \sum_{j=1}^k d_{ij}^2$$

k-means算法的优点：

(1) k-means算法是解决聚类问题的一种经典算法，算法简单、快速。

(2) 对处理大数据集，该算法是相对可伸缩的和高效率的，因为它的复杂度大约是 $O(nkt)$ ，其中 n 是所有对象的数目， k 是簇的数目, t 是迭代的次数。通常 $k \ll n$ 。这个算法通常局部收敛。

(3) 算法尝试找出使平方误差函数值最小的 k 个划分。当簇是密集的、球状或团状的，且簇与簇之间区别明显时，聚类效果较好。

缺点：

(1) k-平均方法只有在簇的平均值被定义的情况下才能使用，且对有些分类属性的数据不适合。

(2) 要求用户必须事先给出要生成的簇的数目k。

(3) 对初值敏感，对于不同的初始值，可能会导致不同的聚类结果。

(4) 不适合于发现非凸面形状的簇，或者大小差别很大的簇。

(5) 对于“ 噪声” 和孤立点数据敏感，少量的该类数据能够对平均值产生极大影响。

2. 基于层次的聚类：

自底向上的凝聚方法，比如AGNES。

自上向下的分裂方法，比如DIANA。

3. 基于密度的聚类：

DBSCAN,OPTICS,BIRCH(CF-Tree),CURE.

4. 基于网格的方法：

STING, WaveCluster.

5. 基于模型的聚类：

EM,SOM,COBWEB.

以上这些算法的简介可参考聚类（ 百度百科 ）。

推荐系统：

推荐系统的实现主要分为两个方面：基于内容的实现和协同滤波的实现。

基于内容的实现：

不同人对不同电影的评分这个例子，可以看做是一个普通的回归问题，因此每部电影都需要提前提取出一个特征向量(即x值)，然后针对每个用户建模，即每个用户打的分值作为y值，利用这些已有的分值y和电影特征值x就可以训练回归模型了(最常见的就是线性回归)。这样就可以预测那些用户没有评分的电影的分数。（ 值得注意的是需对每个用户都建立他自己的回归模型 ）

从另一个角度来看，也可以是先给定每个用户对某种电影的喜好程度（ 即权值 ），然后学出每部电影的特征，最后采用回归来预测那些没有被评分的电影。

当然还可以是同时优化得到每个用户对不同类型电影的热爱程度以及每部电影的特征。具体可以

参考Ng在coursera上的ml教程：<https://www.coursera.org/course/ml>

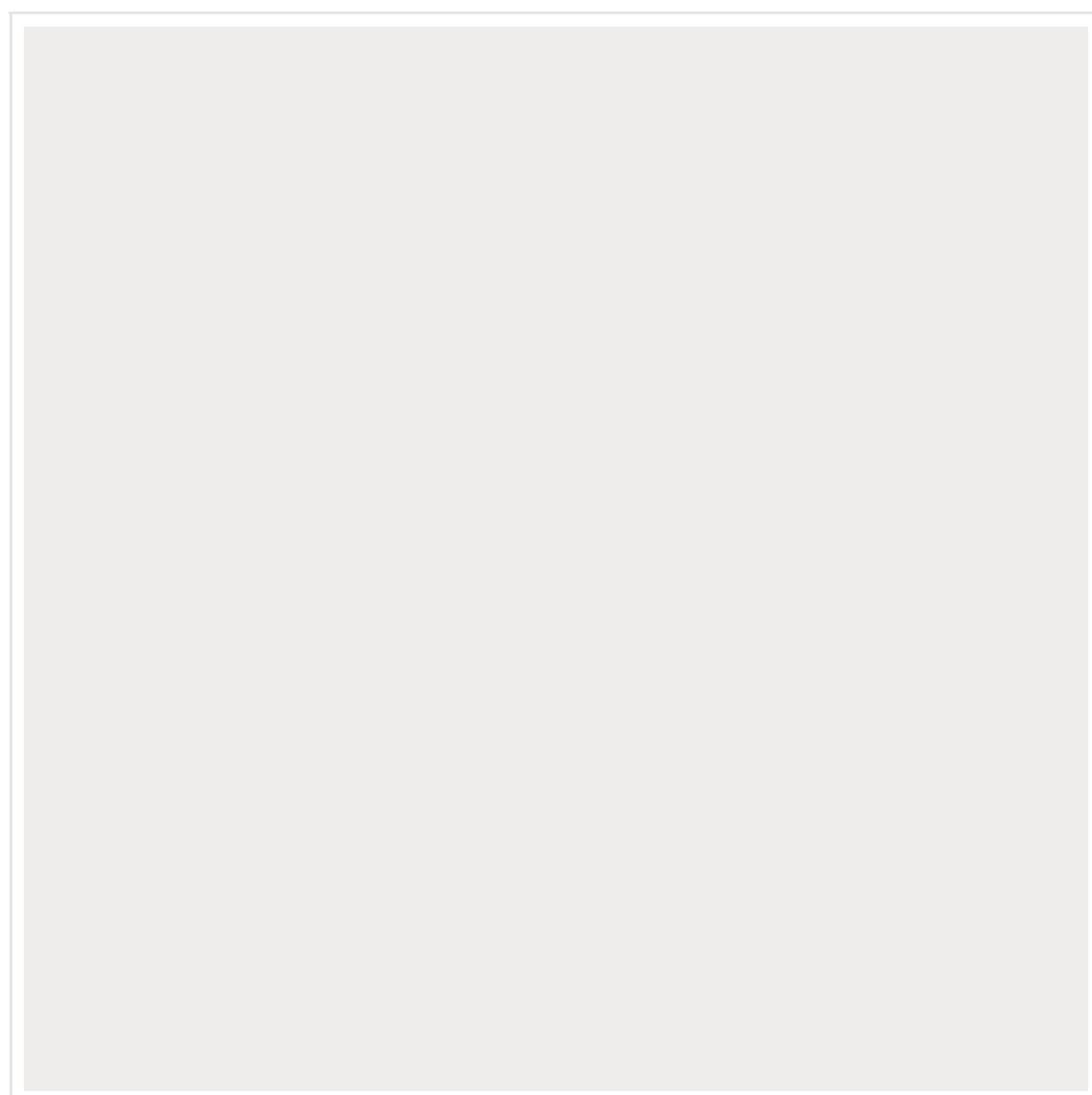
基于协同滤波的实现：

协同滤波（CF）可以看做是一个分类问题，也可以看做是矩阵分解问题。协同滤波主要是基于每个人自己的喜好都类似这一特征，它不依赖于个人的基本信息。比如刚刚那个电影评分的例子中，预测那些没有被评分的电影的分数只依赖于已经打分的那些分数，并不需要去学习那些电影的特征。

SVD将矩阵分解为三个矩阵的乘积，公式如下所示：

$$D = U \Sigma V^T$$

中间的矩阵sigma为对角矩阵，对角元素的值为Data矩阵的奇异值(注意奇异值和特征值是不同的)，且已经从大到小排列好了。即使去掉特征值小的那些特征，依然可以很好的重构出原始矩阵。如下图所示：



其中更深的颜色代表去掉小特征值重构时的三个矩阵。

果m代表商品的个数，n代表用户的个数，则U矩阵的每一行代表商品的属性，现在通过降维U矩阵（取深色部分）后，每一个商品的属性可以用更低的维度表示（假设为k维）。这样当新来一个用户的商品推荐向量X，则可以根据公式 $X' = U^T \cdot \text{inv}(S1)$ 得到一个k维的向量，然后在V' 中寻找最相似的那一个用户（相似度测量可用余弦公式等），根据这个用户的评分来推荐（主要是推荐新用户未打分

的那些商品)。具体例子可以参考网页：SVD在推荐系统中的应用。

另外关于SVD分解后每个矩阵的实际含义可以参考google吴军的《数学之美》一书（不过个人感觉吴军解释UV两个矩阵时好像弄反了，不知道大家怎样认为）。或者参考machine learning in action其中的svd章节。

pLSA:

pLSA由LSA发展过来，而早期LSA的实现主要是通过SVD分解。pLSA的模型图如下：



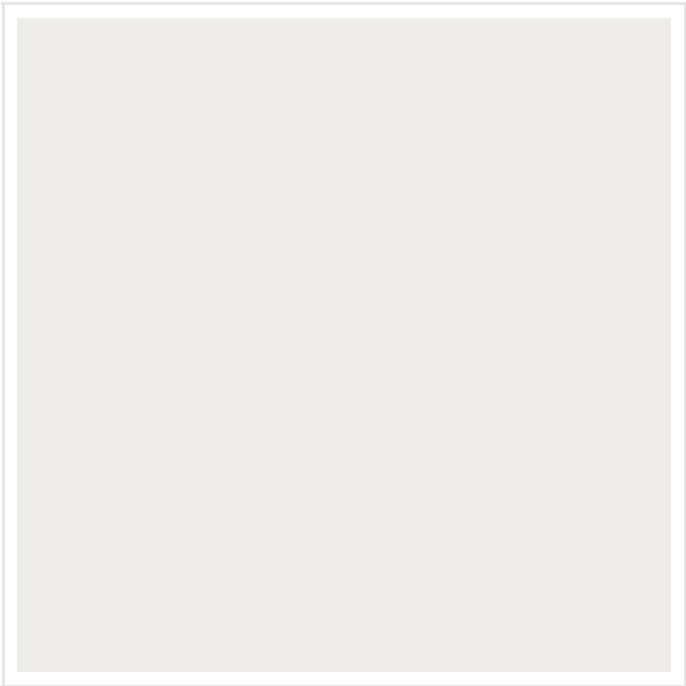
公式中的意义如下：



具体可以参考2010龙星计划：机器学习中对应的主题模型那一讲

LDA：

主题模型，概率图如下：



和pLSA不同的是LDA中假设了很多先验分布，且一般参数的先验分布都假设为Dirichlet分布，其原因是共轭分布时先验概率和后验概率的形式相同。

GBDT :

GBDT(Gradient Boosting Decision Tree) 又叫 MART (Multiple Additive Regression Tree) , 好像在阿里内部用得比较多 (所以阿里算法岗位面试时可能会问到) , 它是一种迭代的决策树算法 , 该算法由多棵决策树组成 , 所有树的输出结果累加起来就是最终答案。它在被提出之初就和SVM一起被认为是泛化能力 (generalization)较强的算法。近些年更因为被用于搜索排序的机器学习模型而引起大家关注。

GBDT是回归树，不是分类树。其核心就在于，每一棵树是从之前所有树的残差中来学习的。为了防止过拟合，和Adaboosting一样，也加入了boosting这一项。

关于GBDT的介绍可以参考：GBDT (MART) 迭代决策树入门教程 | 简介。

Regularization:

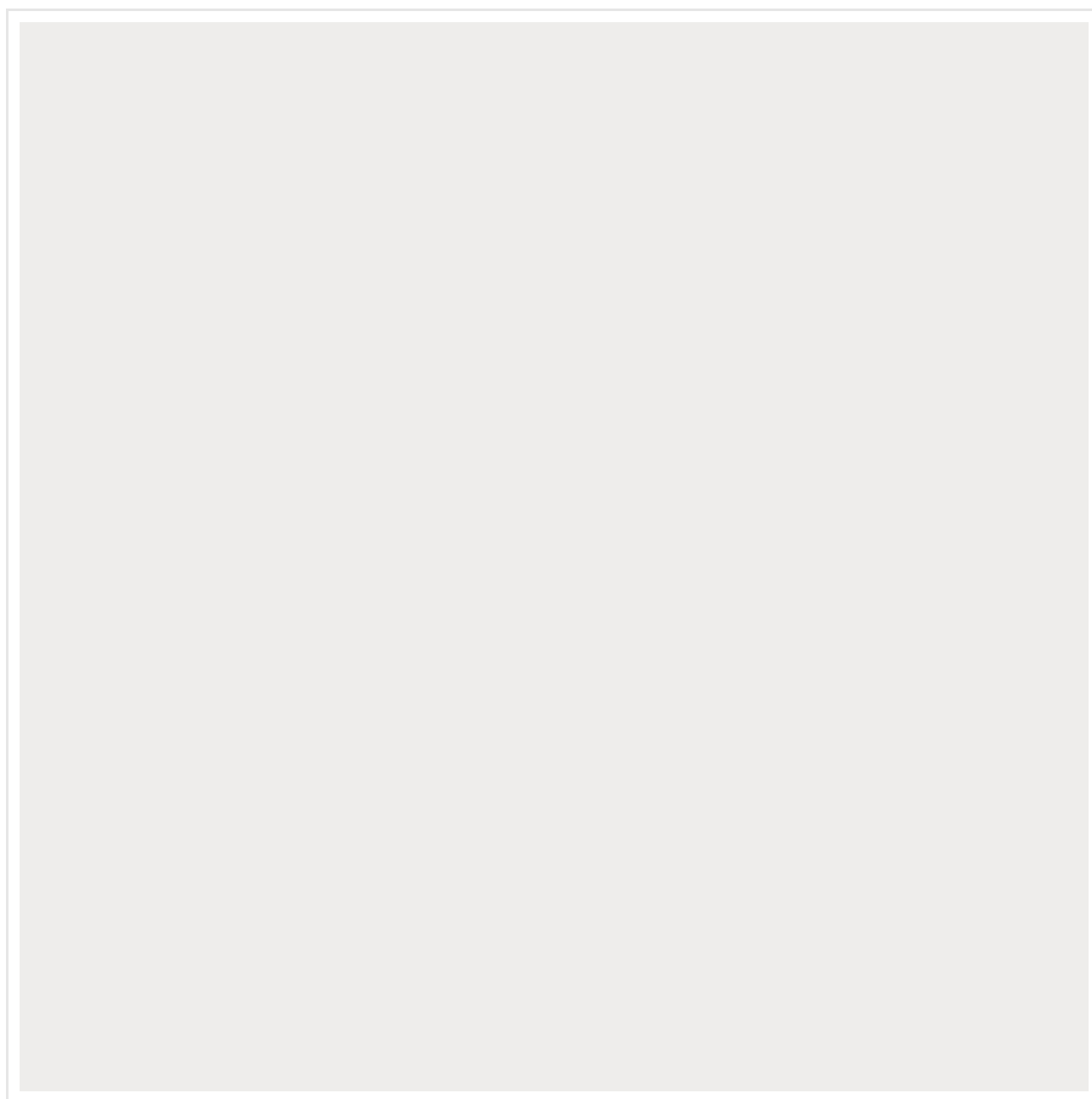
作用是 (网易电话面试时有问到) :

1. 数值上更容易求解；
2. 特征数目太大时更稳定；
3. 控制模型的复杂度，光滑性。复杂性越小且越光滑的目标函数泛化能力越强。而加入规则项能使目标函数复杂度减小，且更光滑。
4. 减小参数空间；参数空间越小，复杂度越低。
5. 系数越小，模型越简单，而模型越简单则泛化能力越强 (Ng宏观上给出的解释) 。
6. 可以看成是权值的高斯先验。

异常检测 :

可以估计样本的密度函数，对于新样本直接计算其密度，如果密度值小于某一阈值，则表示该样本异常。而密度函数一般采用多维的高斯分布。如果样本有n维，则每一维的特征都可以看作是符合高

斯分布的，即使这些特征可视化出来不太符合高斯分布，也可以对该特征进行数学转换让其看起来像高斯分布，比如说 $x = \log(x+c)$, $x = x^{(1/c)}$ 等。异常检测的算法流程如下：



其中的 ϵ 也是通过交叉验证得到的，也就是说在进行异常检测时，前面的 $p(x)$ 的学习是用的无监督，后面的参数 ϵ 学习是用的有监督。那么为什么不全部使用普通有监督的方法来学习呢（即把它看做是一个普通的二分类问题）？主要是在异常检测中，异常的样本数量非常少而正常样本数量非常多，因此不足以学习到好的异常行为模型的参数，因为后面新来的异常样本可能完全是与训练样本中的模式不同。

另外，上面是将特征的每一维看成是相互独立的高斯分布，其实这样的近似并不是最好的，但是它的计算量较小，因此也常被使用。更好的方法应该是将特征拟合成多维高斯分布，这时有特征之间的相关性，但随之计算量会变复杂，且样本的协方差矩阵还可能出现不可逆的情况（主要在样本数比特征数小，或者样本特征维数之间有线性关系时）。

上面的内容可以参考Ng的<https://www.coursera.org/course/ml>

EM算法：

有时候因为样本的产生和隐含变量有关（隐含变量是不能观察的），而求模型的参数时一般采用最大似然估计，由于含有了隐含变量，所以对似然函数参数求导是求不出来的，这时可以采用EM算法来求模型的参数的（对应模型参数个数可能有多个），EM算法一般分为2步：

E步：选取一组参数，求出在该参数下隐含变量的条件概率值；

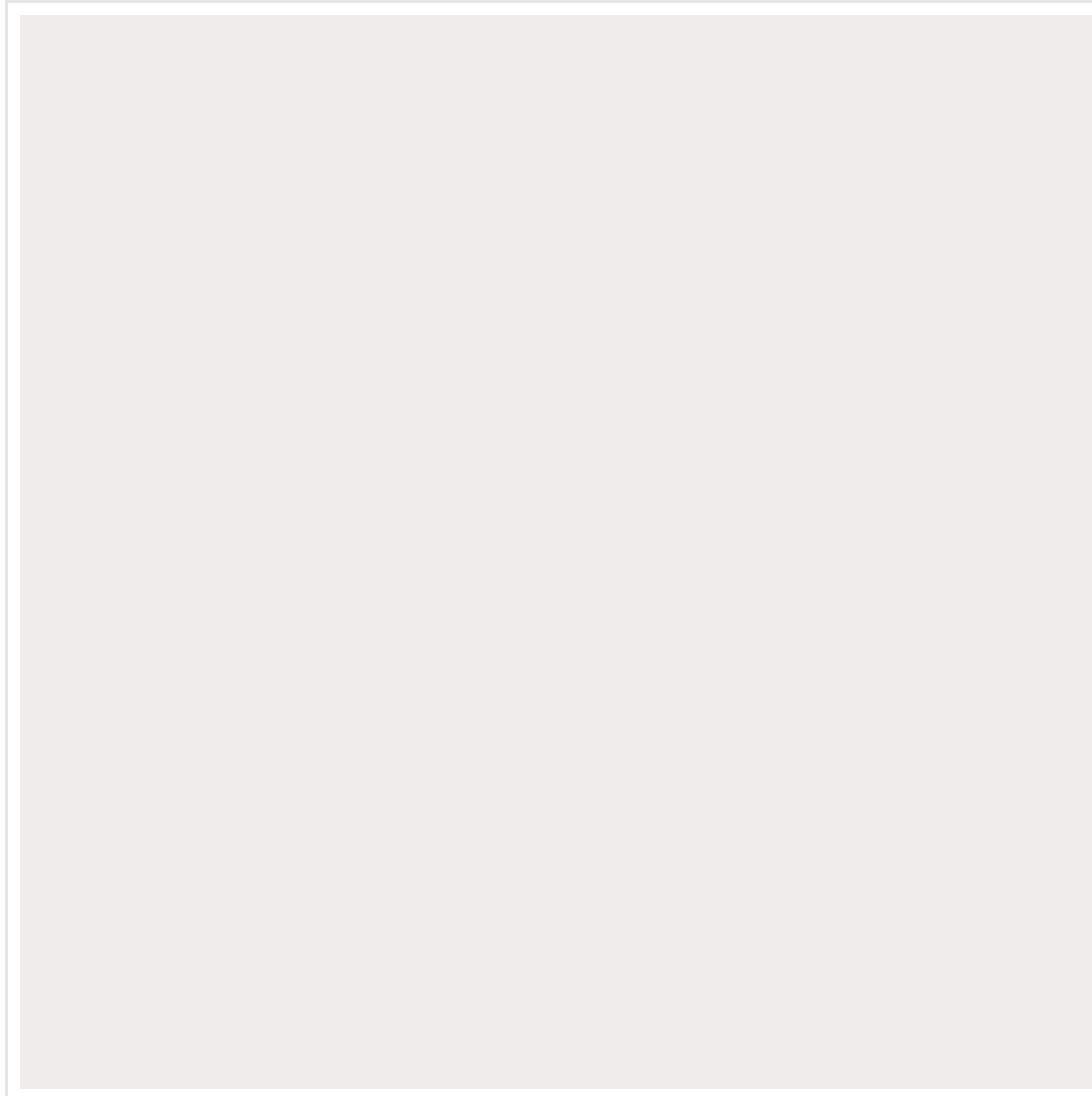
M步：结合E步求出的隐含变量条件概率，求出似然函数下界函数（本质上是某个期望函数）的最大值。

重复上面2步直至收敛。

公式如下所示：



M步公式中下界函数的推导过程：



EM算法一个常见的例子就是GMM模型，每个样本都有可能由k个高斯产生，只不过由每个高斯产生的概率不同而已，因此每个样本都有对应的高斯分布（k个中的某一个），此时的隐含变量就是每个样本对应的某个高斯分布。

GMM的E步公式如下（计算每个样本对应每个高斯的概率）：

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

更具体的计算公式为：

$$p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{\sum_{l=1}^k p(x^{(i)} | z^{(i)} = l; \mu, \Sigma) p(z^{(i)} = l; \phi)}$$

M步公式如下（计算每个高斯的比重，均值，方差这3个参数）：

(M-step) Update the parameters:

$$\begin{aligned}\phi_j &:= \frac{1}{m} \sum_{i=1}^m w_j^{(i)}, \\ \mu_j &:= \frac{\sum_{i=1}^m w_j^{(i)} x^{(i)}}{\sum_{i=1}^m w_j^{(i)}}, \\ \Sigma_j &:= \frac{\sum_{i=1}^m w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^m w_j^{(i)}}\end{aligned}$$

关于EM算法可以参考Ng的cs229课程资料 或者网易公开课：斯坦福大学公开课：机器学习课程。

Apriori:

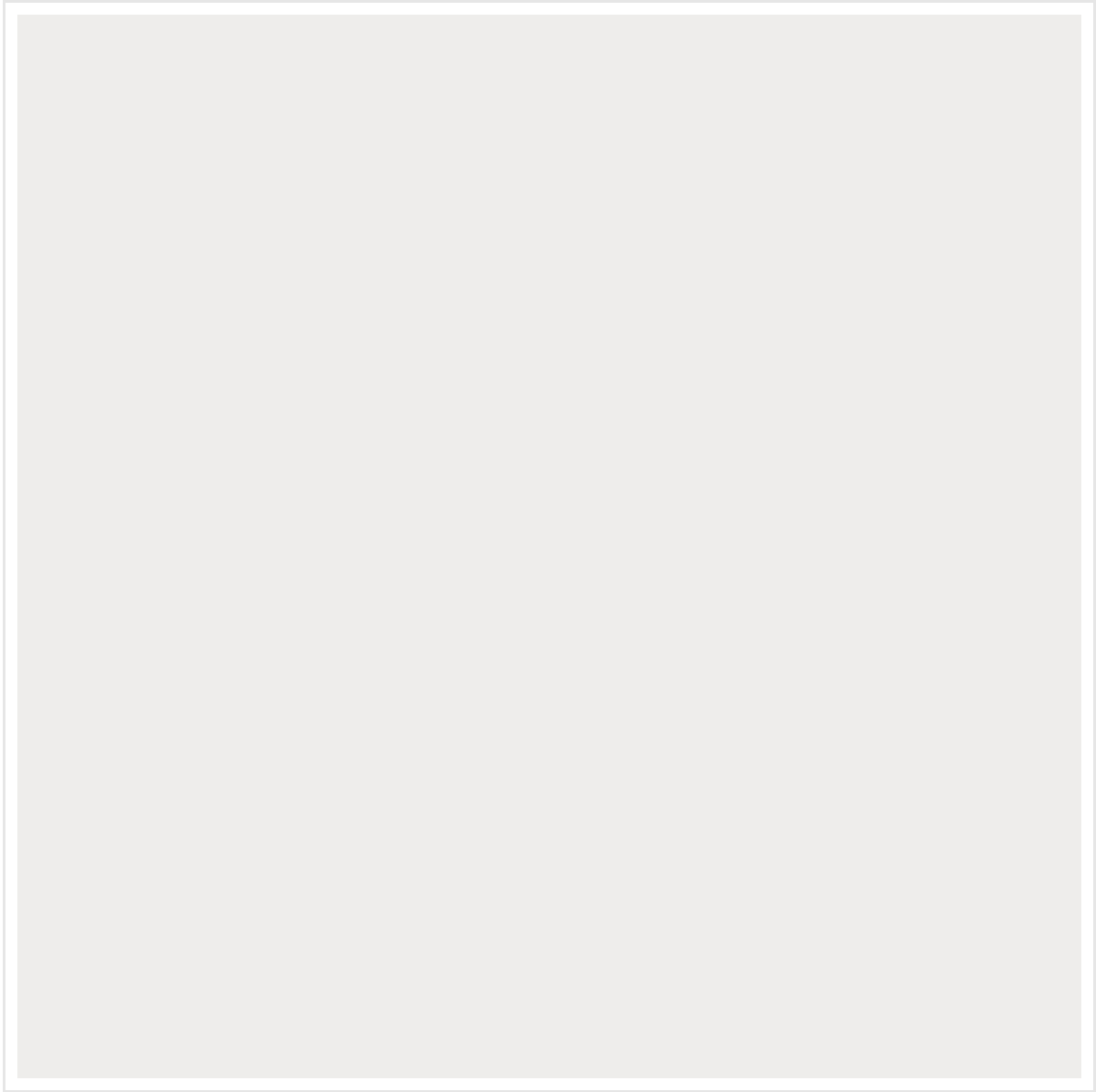
Apriori是关联分析中比较早的一种方法，主要用来挖掘那些频繁项集合。其思想是：

1. 如果一个项目集合不是频繁集合，那么任何包含它的项目集合也一定不是频繁集合；
2. 如果一个项目集合是频繁集合，那么它的任何非空子集也是频繁集合；

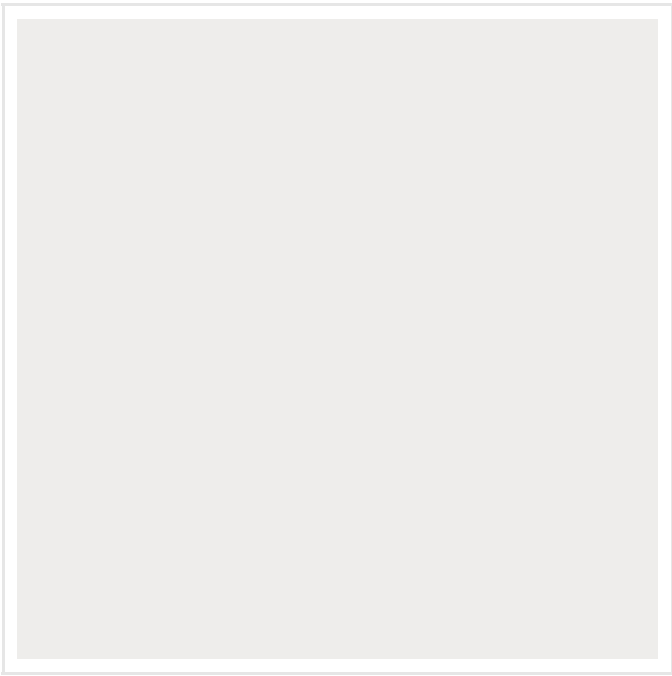
Apriori需要扫描项目表多遍，从一个项目开始扫描，舍去掉那些不是频繁的项目，得到的集合称为L，然后对L中的每个元素进行自组合，生成比上次扫描多一个项目的集合，该集合称为C，接着又扫描去掉那些非频繁的项目，重复...

看下面这个例子：

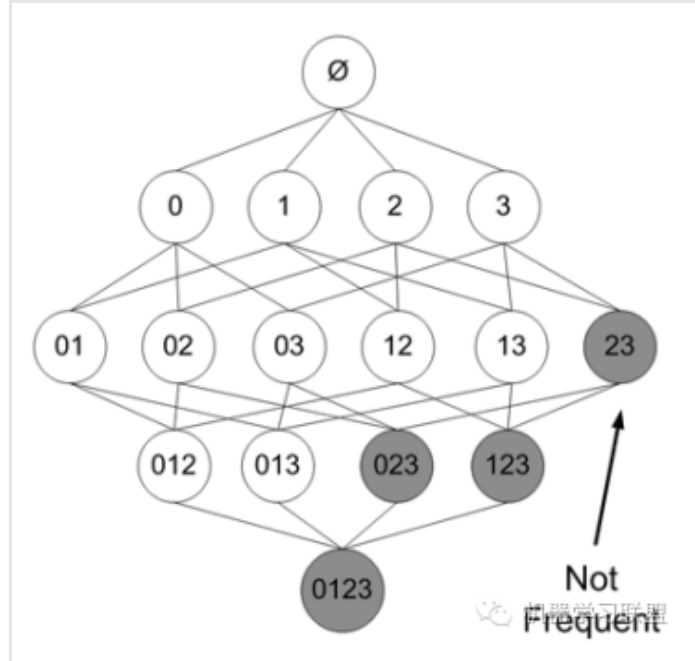
元素项目表格：



如果每个步骤不去掉非频繁项目集，则其扫描过程的树形结构如下：



在其中某个过程中，可能出现非频繁的项目集，将其去掉（用阴影表示）为：



上面的内容主要参考的是machine learning in action这本书。

FP Growth:

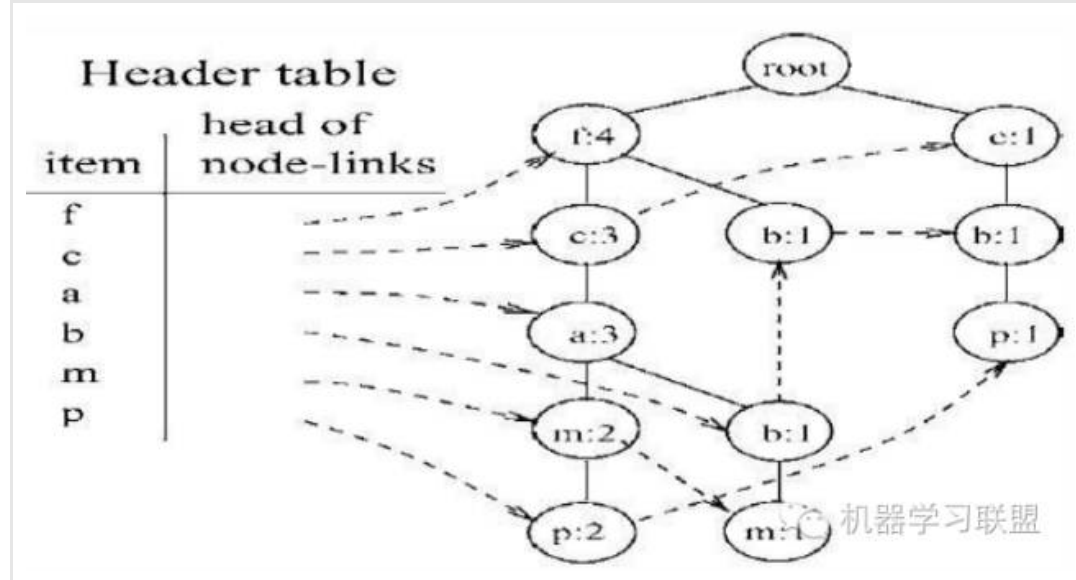
FP Growth是一种比Apriori更高效的频繁项挖掘方法，它只需要扫描项目表2次。其中第1次扫描获得当个项目的频率，去掉不符合支持度要求的项，并对剩下的项排序。第2遍扫描是建立一颗FP-Tree(frequent-patten tree)。

接下来的工作就是在FP-Tree上进行挖掘。

比如说有下表：

TID	Items bought	(Ordered) frequent items
100	<i>f, a, c, d, g, i, m, p</i>	<i>f, c, a, m, p</i>
200	<i>a, b, c, f, l, m, o</i>	<i>f, c, a, b, m</i>
300	<i>b, f, h, j, o</i>	<i>f, b</i>
400	<i>b, c, k, s, p</i>	<i>c, b, p</i>
500	<i>a, f, c, e, l, p, m, n</i>	<i>f, c, a, m, p</i>

它所对应的FP_Tree如下：



然后从频率最小的单项P开始，找出P的条件模式基，用构造FP_Tree同样的方法来构造P的条件模式基的FP_Tree，在这棵树上找出包含P的频繁项集。

依次从m,b,a,c,f的条件模式基上挖掘频繁项集，有些项需要递归的去挖掘，比较麻烦，比如m节点，具体的过程可以参考博客：Frequent Pattern 挖掘之二(FP Growth算法)，里面讲得很详细。

机器学习工作职位需要的7项技能

机器学习经常与人工智能紧密相连,在不考虑显式编程的情况下,机器学习可以使计算机具备完成特定任务的能力,例如识别,诊断,规划,机器人控制和预测等。它往往聚焦于算法创新,即在面对新数据时,其自身能够发生演化。

在某种程度上,机器学习与数据挖掘很相似。它们都是通过数据来获取模式。然而,与人类可理解的数据提取方式不同——通常是按照数据挖掘应用的方式——机器学习主要是使用数据去提升程序本身的理解能力。机器学习程序能够在数据中检测出相关模式并相应的进行程序行为的调整。

现在,你是否准备去了解一些获得机器学习工作必备的技术了呢?一个优秀的求职者应该对以下各方面知识都有很深的理解:算法和数学应用,问题解决能力和分析技巧,概率统计和诸如 Python/C++/R/Java 等编程语言。此外,机器学习还要求求职者具有与生俱来的好奇心,因此,如何你从来没有失去过自孩童时代就有的好奇心,那么,你就能顺理成章在机器学习领域取得成就。这里详细的列出一个的必备的技能清单。

1. Python/C++/R/Java

如果你希望在机器学习领域获得一份工作,那么在某种程度上,你很可能必须学习这里所列出的所有编程语言。C++ 能够加速代码执行速度。R 在统计绘图方面十分出色,Hadoop 是以 Java 为基础的,因此,你可能需要在 Java 中完成 Map/Reduce 算法。

2. Probability and Statistics (概率和统计)

概率和统计理论能够帮助你学习算法。很多常用的模型例如朴素贝叶斯、高斯混合模型和隐马尔可夫模型等,需要你有很好的概率和统计背景知识去理解。甚至你需要全身心的投入并且研究测度理论,同时需要理解一些统计指标,这些指标常作为模型评价标准,例如混淆矩阵,ROC曲线, P值等。

3. Applied Math and Algorithms(数学和算法)

对算法理论有相当深入的认识并且了解算法运行的机制,能够帮助你对模型加以区分,例如支持向量机模型 (译者注:支持向量机模型包括许多不同的核函数,核函数的不同,具体模型的原理、应用和结论也不同)。你需要理解一些数学方法,例如梯度下降,凸优化,拉格朗格方法,二次规划,偏微分方程等类似的理论和方法。同时,你也需要熟悉求和运算<http://en.wikipedia.org/wiki/Summation>。

4. Distributed Computing (分布式计算)

大多数时候,机器学习需要处理大型的数据集。使用单机无法处理这些数据,因此,你需要通过集群进行分布式计算。像 Apache Hadoop 架构和 Amazon 的 EC2 云服务等项目能够使这一过程更加容易,从而提高成本效益。

5. Expanding the Expertise in Unix Tools(使用Unix工具来拓宽你的专业知识)

你应该掌握专门为以下工作而设计的Unix命令或工具: cat, grep, nd, awk, sed, sort, cut, tr 等。由于所有这些处理过程都运行于基于linux平台的设备,因此,你需要熟悉这些工具。学习并很好的使用这些工具,会使你的工作更加轻松。

6. Learning more about Advanced Signal Processing techniques(学习一些信号处理技术)

特征提取是机器学习最重根据部分之一。不同问题需要不同的解决方案,你可以使用非常酷的高级信号处理算法,例如小波变换,剪切波变换,曲线波,轮廓波和 bandlets 变换等。学习时频分析技术,并用它来解决你的问题。如果你还不知道傅里叶分析和卷积原理,你同样也需要学习这些知识。二进制码信号处理技术是解决问题 的重要方法。

7. Other skills

(a) 提升自己:你必须时刻保持与新技术的同步以应对将要到来的挑战。这也意味着你必须注意以下几方面的最新动态:关于这些工具理论的变更日志和会议,算法的研究论文、博客和会议视频等。(b) 大量

阅读。阅读一些像 Google Map-Reduce, Google File System, Google Big Table,以及 e Unreasonable Effectiveness of Data 之类的 论文。此外,网上也有许多免费的机器学习书籍,你同样也应该读一读。

appy Machine Learning!

索引网站推荐：机器学习综述论文

这里推荐的网站是一个机器学习方面综述论文的索引网站，可惜的是，该网站好像2012年之后已经停止更新了。。。我个人对于综述文章是很有兴趣的，后面遇到有趣的新综述，会在微信平台分享给大家。

需要注意的是，并不是每一篇冠有综述之名的文章都具有综述的价值，有一些所谓的综述文章完全是凑数或赶工，请按照以下方式鉴别：

1、是否发表在水平较高的综述期刊

ACM Computing Surveys (CSUR)

Foundations and Trends in Machine Learning

The Knowledge Engineering Review

Artificial Intelligence Review

上面的几个是常见的本领域综述期刊，最后的AI Review文章数量较多，也存在一些水分。

当然，发表在并非综述期刊的文章，只要是高水平期刊，同样值得信赖。

2、作者是否是成熟的研究者

看到马来西亚或者印度作者的时候，尤其需要注意.....

(不是有意要黑的，对不起(⊙o⊙)...)

3、参考文献个数

作为综述，如果只有二十篇参考文献。。。。

4、很多时候，一篇高水平的博士论文，能起到比综述更好的作用。

附：

Machine Learning Surveys链接：

<http://www.mlsurveys.com/>

收集自网络，仅供学习参考使用！

技术大牛书单

- NGINX High Performance 下载
- The Book of R
- Reactive Design Patterns
- PHP 7: Real World Application Development
- Python 经典书籍：Black Hat Python
- Android Programming

slippers. reinvented.



the world's most
comfortable slippers

