NLP 中的概率图模型

很多机器学习的算法也常用于 NLP 的任务。例如，用朴素贝叶斯进行文本分类、用 SVM 进行语义角色标注，虽然它们在某些 NLP 任务中都实现了很好的效果，但它们都相互独立，没有形成体系。

随着近些年对智能推理和认知神经学的深入研究，人们对大脑和语言的内在机制了解得越来越多，也越来越能从更高层次上观察和认识思维（包括语言）的现象，由此形成一套完整的算法体系。目前最流行的算法思想包含如下两大流派：基于概率论和图论的概率图模型；基于人工神经网络的深度学习理论。

贝叶斯与朴素贝叶斯算法

贝叶斯公式最早是由英国神学家贝叶斯提出来的，用来描述两个条件概率之间的关系。在之前的条件概率定义中，我们知道

$$P(A|B)=\frac{P(A,B)}{P(B)} \qquad P(B|A)=\frac{P(A,B)}{P(A)}$$

由上式进一步推导得到：

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

由此，推广到随机变量的范畴，设 X，Y 为两个随机变量，得到贝叶斯公式：

$$P(X|Y)=\frac{P(Y|X)P(X)}{P(Y)}$$

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

其中，P(Y)叫作先验概率，P(Y|X)叫作后验概率，P(Y,X)是联合概率

在机器学习的视角下，我们把 X 理解成“具有某种特征”，把 Y 理解成“类别标签”：

$$P(\text{所属类别}|\text{某种特征})=\frac{P(\text{某种特征}|\text{所属类别})P(\text{所属类别})}{P(\text{某种特征})}$$

贝叶斯方法把计算“具有某特征的条件下属于某类”的概率转换成需要计算“属于某类

的条件下具有某特征”的概率，属于有监督学习。

下面以一个例子来解释朴素贝叶斯，给定数据如下：

帅？	性格好？	身高？	上进？	嫁与否
帅	不好	矮	不上进	不嫁
不帅	好	矮	上进	不嫁
帅	好	矮	上进	嫁
不帅	好	高	上进	嫁
帅	不好	矮	上进	不嫁
不帅	不好	矮	不上进	不嫁
帅	好	高	不上进	嫁
不帅	好	高	上进	嫁
帅	好	高	上进	嫁
不帅	不好	高	上进	嫁
帅	好	矮	不上进	不嫁
帅	好	矮	不上进	不嫁

现在给我们的问题是，如果一对男女朋友，男生想女生求婚，男生的四个特点分别是不帅，性格不好，身高矮，不上进，请你判断一下女生是嫁还是不嫁？

这是一个典型的分类问题，转为数学问题就是比较 p(嫁|(不帅、性格不好、身高矮、不上进))与 p(不嫁|(不帅、性格不好、身高矮、不上进))的概率，谁的概率大，我就能给出嫁或者不嫁的答案！这里我们联系到朴素贝叶斯公式：

$$P(\text{嫁}|\text{不帅},\text{性格不好},\text{矮},\text{不上进}) = \frac{P(\text{不帅},\text{性格不好},\text{矮},\text{不上进}|\text{嫁})P(\text{嫁})}{P(\text{不帅},\text{性格不好},\text{矮},\text{不上进})}$$

我们需要求 p(嫁|(不帅、性格不好、身高矮、不上进),这是我们不知道的，但是通过朴素贝叶斯公式可以转化为好求的三个量，这三个变量都能通过统计的方法求得。

等等，为什么这个成立呢？学过概率论的同学可能有感觉了，这个等式成立的条件需要特征之间相互独立吧！对的！这也就是为什么朴素贝叶斯分类有朴素一词的来源，朴素贝叶斯算法是假设各个特征之间相互独立，那么这个等式就成立了！

但是为什么需要假设特征之间相互独立呢？

1. 我们这么想，假如没有这个假设，那么我们对右边这些概率的估计其实是不可做的，这么说，我们这个例子有 4 个特征，其中帅包括{帅，不帅}，性格包括{不好，好，爆好}，身高包括{高，矮，中}，上进包括{不上进，上进}，那么四个特征的联合概率分布总共是 4 维空间，总个数为 2*3*3*2=36 个。

36 个，计算机扫描统计还可以，但是现实生活中，往往有非常多的特征，每一个特征的取值也是非常之多，那么通过统计来估计后面概率的值，变得几乎不可做，这也是为什么需要假设特征之间独立的原因。

2. 假如我们没有假设特征之间相互独立，那么我们统计的时候，就需要在整个特征空间中去找，比如统计 p(不帅、性格不好、身高矮、不上进|嫁),我们就需要在嫁的条件下，去找四种特征全满足分别是不帅，性格不好，身高矮，不上进的人的个数，这样的话，由于数据的稀疏性，很容易统计到 0 的情况。 这样是不合适的。

根据上面俩个原因，朴素贝叶斯法对条件概率分布做了条件独立性的假设，由于这是一

个较强的假设，朴素贝叶斯也由此得名！这一假设使得朴素贝叶斯法变得简单，但有时会牺牲一定的分类准确率。

朴素贝叶斯理论源于随机变量的独立性，之所以称之为朴素，是因为其思想基础的简单性：就文本分类而言，从朴素贝叶斯的角度来看，句子中的两两词之间的关系是相互独立的，即一个对象的特征向量中每个维度都是相互独立的。这是朴素贝叶斯理论的思想基础。在多维的情况下，朴素贝叶斯分类的表示为如下内容。

(1) 设 x={a1,a2,...,am}为一个待分类项，而每个 a 为 x 的一个属性特征。
(2) 有类别集合 C={y1,y2,...,yn}。
(3) 计算 p(y1|x),p(y2|x),p(y3|x),...,p(yn|x)。
(4) 如果 p(yk|x)=max{p(y1|x),p(y2|x),p(y3|x),...,p(yn|x)}，则 x∈yk。

那么，现在的关键就是如何计算第 3 步中的各个条件概率。

(1) 找到一个已知分类的待分类项集合，也就是训练集。
(2) 统计得到在各个类别下各个特征属性的条件概率估计，即

p(a1|y1),p(a2|y1),...p(am|y1);
p(a1|y2),p(a2|y2),...p(am|y2);
......
p(a1|yn),p(a2|yn),...p(am|yn);

(3) 如果各个特征属性是条件独立的（或者假设它们之间是相互独立的），则根据贝叶斯定理有如下推导：

$$p(y_i \mid x) = \frac{p(x \mid y_i)p(y_i)}{p(x)}$$

因为分母对于所有类别为常数，只要将分子最大化即可。又因为各特征属性是条件独立的，所以有：

$$p(x \mid y_i)p(y_i) = p(a_1 \mid y_i)p(a_2 \mid y_i)...p(a_m \mid y_i)p(y_i) = p(y_i)\prod_{j=1}^{m} p(a_j \mid y_i)$$

根据上述分析，朴素贝叶斯分类的流程可以表示如下。

第一阶段，训练数据生成训练样本集：TF-IDF。

第二阶段，对每个类别计算 P(yi)。

第三阶段，对每个特征属性计算所有划分的条件概率。

第四阶段，对每个类别计算 p(x|yi)p(yi)。

第五阶段，以 p(x|yi)p(yi)的最大项作为 x 的所属类别。

朴素贝叶斯优点：

算法逻辑简单,易于实现（算法思路很简单，只要使用贝叶斯公式转化即可！）

分类过程中时空开销小（假设特征相互独立，只会涉及到二维存储）

朴素贝叶斯缺点：

理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型假设属性之间相互独立，这个假设在实际应用中往往是不成

立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。

朴素贝叶斯模型(Naive Bayesian Model)的朴素(Naive)的含义是"很简单很天真"地假设样本特征彼此独立. 这个假设现实中基本上不存在, 但特征相关性很小的实际情况还是很多的, 所以这个模型仍然能够工作得很好。

文本分类

在处理文本分类之前，引入两个概念：

one-hot 编码

文本分类的结构化方法就是 one-hot 表达模型。它是最直观，也是目前为止最常用的词表示方法，虽然越来越多的实践已经证明，这种模型存在局限性，但它仍在文本分类中得到广泛应用。

假设把语料库中的所有词都收集为一个词典 D，词典容纳了语料库中所有句子的词汇。One-hot 方法就是把每个词表示为一个长长的向量。这个向量的维度是词典大小，其中绝大多数元素为 0，只有一个维度的值为 1。这个维度就代表了当前的词。

例如，一个语料库有三个文本。

文本 1：My dog ate my homework.

文本 2：My cat ate the sandwich.

文本 3：A dolphin ate the homework.

这三个文本生成的词典共有 9 个词：

[a, ate, cat, dolphin, dog, homework, my, sandwich, the]

这 9 个词依次表示为 One-hot 向量的形式如下：

‘a’ -> [1 0 0 0 0 0 0 0 0]

‘ate’ -> [0 1 0 0 0 0 0 0 0]

... ...

‘the’ -> [0 0 0 0 0 0 0 0 1]

其中，每个词都是”茫茫 0 海“中的一个 1。这种稀疏的表达方式就是 one-hot 表达。它的特点是相互独立地表示语料中的每个词。词与词在句子中的相关性被忽略了，这正符合朴素贝叶斯对文本的假设。

TF-IDF

由 one-hot 向量形式构成的句子就是一个词袋模型，它将文本中的词转换为数字，而整个文本集也都转换为维度相等的词向量矩阵。

仍旧使用上述三个文本，文本的词向量可以使用二值表示：

文本 1：0，1，0，0，1，1，1，0，0

文本 2：0，1，1，0，0，0，1，1，1

文本 3：1，1，0，1，0，1，0，0，1

这种方式的问题是忽略了一个句子中出现多个相同词的词频信息，增加这些词频信息，就变成了整型计数方式。

文本 1：0，1，0，0，1，1，2，0，0

文本 2：0，1，1，0，0，0，1，1，1

文本 3：1，1，0，1，0，1，0，0，1

接下来，对整型计数方式进行归一化。归一化可以避免句子长度不一致问题，便于算法计算，而且对于基于概率算法，词频信息就变为了概率分布。这就是文档的 TF 信息。

文本 1：0，1/5，0，0，1/5，1/5，2/5，0，0

文本 2：0，1/5，1/5，0，0，0，1/5，1/5，1/5

文本 3：1/5，1/5，0，1/5，0，1/5，0，0，1/5

但是这里还有一个问题，如何体现生成的词袋中的词频信息呢？

原信息：

a(1), ate(3), cat(1), dolphin(1), dog(1), homework(2), my(3), sandwich(1), the(2)

注意：由于词袋收集了所有文档中的词，这些词的词频是针对所有文档的词频，因此，词袋的统计基数是文档数。

词条的文档频率：

a(1/3), ate(3/3), cat(1/3), dolphin(1/3), dog(1/3), homework(2/3), my(2/3), sandwich(1/3), the(2/3)

IDF：a log(3/1), ate log(3/3), cat log(3/1), dolphin log(3/1), dog log(3/1), homework log(3/2), my log(3/2), sandwich log(3/1), the log(3/2)

TF-IDF 权重策略：计算文本的权重向量，应该选择一个有效的权重方案。最流行的方案是对 TF-IDF 权重的方法。TF-IDF 的含义是词频-逆文档频率，其含义是如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

在前面的例子中，在文本 1 中 my 的词频是 2，因为它在文档中出现了两次。而在这三个文件集合中，my 词条的文档频率也是 2。TF-IDF 的假设是，高频率词应该具有高权重，除非它也是高文档频率。my 这个词在文本中是经常出现的词汇。它不仅多次发生在单一的文本中，几乎也发生在每个文档中。逆文档频率就是使用词条的文档频率来抵消该词的词频

对权重的影响，而得到一个较低的权重。

词频（term frequency，TF）指的是某一个给定的词语在该文件中出现的频率。这个数字是对词数(term count)的归一化，以防止它偏向长的文件。（同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否。）对于在某一特定文件里的词语来说，它的重要性可表示为：

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子中分子是该词在文件中的出现次数，而分母则是在文件中所有字词的出现次数之和。

逆向文件频率（inverse document frequency，IDF）是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取以 10 为底的对数得到：

$$\mathrm{idf_i} = \lg \frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中，|D|：语料库中的文件总数，j：包含词语的文件数目（即的文件数目）如果该词语不在语料库中，就会导致分母为零，因此一般情况下使用 $1 + |\{d \in D : t \in d\}|$ 作为分母。然后再计算 TF 与 IDF 的乘积。

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$ 可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

文本分类的实现

```
Nbayes_lib.py
```

统计语言模型与 NLP 算法设计

在多年自然语言处理算法开发与建模的实践中，人们逐渐形成一套基于统计理论的 NLP 算法设计方法论。这套方法论最基础的环节就是统计语言模型(statistical language model)，它被广泛用于 NLP 的各项任务中。

那么，什么是语言模型呢？简单的说，统计语言模型是用来计算句子中某种语言模式出现概率的统计模型。一般自然语言的统计单位是句子，所以也看作句子的概率模型。假设 W=(w1,w2,...,wn)为一个句子，这个句子有 n 个词，也就是 n 个词汇按顺序构成的字符序列，这里表示为 w_1^n。前面讲过，利用贝叶斯公式进行链式分界，w1,w2,...wn 的联合概率

为：

$$p(W)=p(w_1^n)=p(w_1,w_2,...w_n)$$
$$=p(w_1)p(w_2\,|\,w_1)p(w_3\,|\,w_1^2)...p(w_n\,|\,w_1^{n-1})$$

需要设法计算 $p(w_1)p(w_2\,|\,w_1)p(w_3\,|\,w_1^2)...p(w_n\,|\,w_1^{n-1})$ 这些语言模型的参数，得到这些参数，即可得到 $p(w_1^n)$。

为了方便表示，将上述式子表述为语言模型的形式。考虑一个长度为 n 的句子，位置 k 的词出现的概率与其前面的所有的词都相关，也就是说与它前面的 k-1 个词都相关，其 k 元语言模型可以表示为

$p(w_k\,|\,w_1^{k-1})$ k>1

利用贝叶斯公式得到：

$$p(w_k\,|\,w_1^{k-1})=\frac{p(w_k)}{p(w_1^{k-1})}$$

假设有一个能够容纳所有语言现象的语料库，即统计样本足够大时，$p(w_k\,|\,w_1^{k-1})$ 就变为表达形式：

$$p(w_k\,|\,w_1^{k-1})=\frac{count(w_k)}{count(w_1^{k-1})}$$

其中, $count(w_k)$ 和 $count(w_1^{k-1})$ 分别表示 w_k 和 w_1^{k-1} 在语料中出现的次数。

考虑一个给定长度为 n 的句子就需要计算 n 个参数，不妨假设语料库对应词典 D 的大小（词汇量）为 T。那么，如果考虑长度为 n 的任意句子理论上就有 T^n 种可能。而每种可能都要计算 n 个参数，那么总共就需要计算 n T^n 个参数。而这些概率计算好后，还得保存下来。因此，存储这些信息也需要很大的内存开销。

如果一种语言现象出现的概率与它的 n-1 个词相关，而与其前面 n 个词和后面的词都无关，那么这就是一个 n-gram 语言模型。

$$p(w_k\,|\,w_1^{k-1})\approx p(w_k\,|\,w_{k-n+1}^{k-1})\approx\frac{count(w_{k-n+1}^{k})}{count(w_{k-n+1}^{k-1})}$$

如果 n=2，就有：

$$p(w_k\,|\,w_1^{k-1})\approx\frac{count(w_{k-1},w_k)}{count(w_{k-1})}$$

N-gram 语言模型中 n 的大小的选取，需要同时考虑计算复杂度和模型效果的两个因素。考虑汉语中词典大小 D=300000 个词汇。模型参数数量与 n 的关系：

n	模型参数数量
1(unigram)	$3x10^5$
2(bigram)	$9x10^{10}$
3(trigram)	$27x10^{15}$
4(4-gram)	$81x10^{20}$

模型参数的量级是模型长度 n 的指数函数{$O(N^n)$}。显然 n 不能取太大，实际应用中

最多的是采用 n=3 的三元模型。

需要特别说明的是语言模型的平滑问题，

- 若 $count(w_{k-n+1}^{k})=0$ ，是否认为 $p(w_k \mid w_1^{k-1})$ 就等于 0 呢？
- 若 $count(w_{k-n+1}^{k})=count(w_{k-n+1}^{k\text{-}1})$ ，是否认为 $p(w_k \mid w_1^{k-1})$ 就等于 1 呢？

显然不能！但这是一个无法回避的问题，无论你的语料库有多么大，平滑技术都不可避免，还好，在后面讲的算法中，这个问题已经得到解决。

总结起来，n-gram 模型的注意工作是在模型的训练阶段，统计语料中各种语言模式(比如词汇)出现的次数，并通过一定的统计算法，将它们计算好之后就存储起来。在预测阶段，当需要计算一个新的句子的概率时，只需查找到句子中可能出现的语言模型的概率参数，并将它们连乘起来即可得到最终的结果。

这是 NLP 算法的通用策略，这里涉及两个问题，1，如何设计语言模型，2，如何求解算法策略。几乎所有的 NLP 实践都是围绕着这两个问题展开的。

在机器学习领域最常用的策略为：对所考虑的问题建模后，先为其构造一个目标函数，然后对这个目标函数进行优化。从而求得一组最优的参数，最后利用这组最优参数--语言模型来做出预测。

极大似然估计

在求解机器学习、深度学习、概率图模型中，计算过程一般分为：模型的学习阶段（训练阶段）和模型的预测阶段。

在训练阶段，很多模型的学习函数都常被看作一个最优化问题。模型训练以训练集样本作为已知的总体分布，来计算模型参数过程。该阶段的计算特点是把训练集样本点的取值作为已知量，然后学习出样本点对应的参数，如神经网络中的神经元的权重向量、概率图模型的节点权重值等。

这样，模型的训练过程就变成了统计学中的参数估计过程。在统计学中，参数估计的方法有很多。如果已知某个随机样本的总体概率分布（训练集），但是其中具体的参数不清楚，则参数估计就是通过若干次实验，观察其结果，利用结果推断出参数大概值得方法，称为极大似然估计。

极大似然估计是建立在如下思想上的：已知某个参数能使这个样本出现的概率最大，我们当然不会再去选择其它小概率样本，所以干脆就把这个参数作为估计的真实值。

极大似然估计：设总体分布为 f(X,θ)，X1,X2,...,Xn 为该总体采样得到的样本（语料库中的样本集），θ为待估参数。因为 X1,X2,...,Xn 独立同分布，那么它们的联合概率分布为

$$L(x_1,x_2,\ldots,x_n;\theta_1,\theta_2,\ldots,\theta_k)=\prod_{i=1}^{n} f(x_i;\theta_1,\theta_2,\ldots,\theta_k)$$

因为样本已经存在，X1,X2,...,Xn 都被看作已知量，是固定的，θ被看作未知的、待估的参数；L(x,θ)就变成了θ的函数，也就是似然函数。求这个参数θ的值，使得似然函数取极大值，这种方法就是极大似然估计。

求极大似然估计值的一般步骤：

(1) 写出似然函数
(2) 对似然函数取对数，并整理
(3) 求导数
(4) 解似然方程

似然函数常写成若干式子连乘积的形式。在实践中，由于求导数的需要，最好先将似然函数取对数，因为函数的对数其极值点不发生变换，这样就把乘号变为了加号。得到的式子称为对数似然函数。若对数似然函数可导，可通过求到的方式解出下列方程组，得到驻点，然后分析该驻点的极值性。

对应统计语言模型而言，利用最大似然可把目标函数设为：

$$\prod_{w\in C} p(w\,|\,Context(w))$$

其中，C 表示语料 Corpus，Context(w)表示词 w 的上下文(Context 或 Window)，即 w 周边的词的集合，当 context(w)为空时，就取 p(w|context(w))=p(w)。特别的，对应前面说的 n-gram 模型有：

$$Context(w_i) = w_{i-n+1}^{i-1}$$

按照最大对数似然的管理，把目标函数设为：

$$L = \sum_{w\in C} \log p(w\,|\,Context(w))$$

然后对这个函数求最大化。由上式可知，概率 p(w|Context(w))已被视为关于 w 和 Context(w)的函数，即

$$p(w\,|\,Context(w)) = F(w, Context(w), \theta)$$

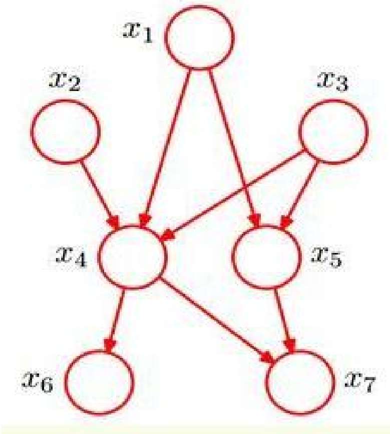
其中，θ为待定参数集。这样一来，一旦对上式进行优化得到最优参数集 θ^* 后，F 也就唯一被确定了，以后任何概率 p(w|Context(w))都可以通过函数 F(w,Context(w),θ*)来计算。

与 n-gram 相比，这种方法不需要事先计算并保存所有的概率值，而是通过直接计算来获取，且通过选取合适的模型可使得θ中参数的个数远小于 n-gram 中模型参数的个数。很显然，对于这种方法：最关键的地方就在于函数 F 的构造。下面介绍概率图模型中各种构造 F 的方法。

贝叶斯网络

贝叶斯网络(Bayesian network)，又称信念网络(Belief Network)，或有向无环图模型

(directed acyclic graphical model)，是一种概率图模型，于 1985 年由 Judea Pearl 首先提出。它是一种模拟人类推理过程中因果关系的不确定性处理模型，其网络拓朴结构是一个有向无环图(DAG)。



贝叶斯网络的有向无环图中的节点表示随机变量{X1,X2,...,Xn}

它们可以是可观察到的变量，或隐变量、未知参数等。认为有因果关系（或非条件独立）的变量或命题则用箭头来连接。若两个节点间以一个单箭头连接在一起，表示其中一个节点是“因(parents)”，另一个是“果(children)”，两节点就会产生一个条件概率值。

例如，假设节点 E 直接影响到节点 H，即 E→H，则用从 E 指向 H 的箭头建立结点 E 到结点 H 的有向弧(E,H)，权值(即连接强度)用条件概率 P(H|E)来表示，如下图所示：

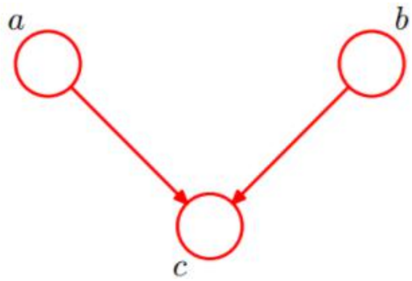
简言之，把某个研究系统中涉及的随机变量，根据是否条件独立绘制在一个有向图中，就形成了贝叶斯网络。其主要用来描述随机变量之间的条件依赖，用圈表示随机变量(random variables)，用箭头表示条件依赖(conditional dependencies)。

此外，对于任意的随机变量，其联合概率可由各自的局部条件概率分布相乘而得出：

$$P(x_1,\ldots,x_k) = P(x_k|x_1,\ldots,x_{k-1})\ldots P(x_2|x_1)P(x_1)$$

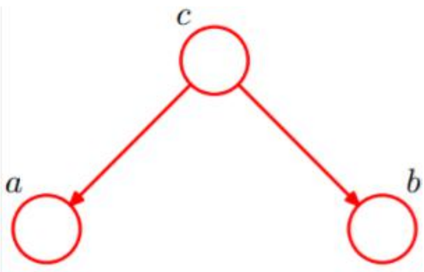
贝叶斯网络的结构形式

head-to-head



依上图，所以有：P(a,b,c) = P(a)P(b)P(c|a,b)成立，即在 c 未知的条件下，a、b 被阻断(blocked)，是独立的，称之为 head-to-head 条件独立。

tail-to-tail

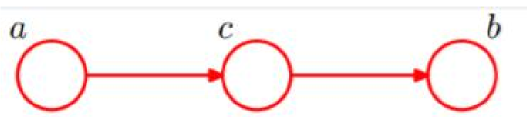


考虑 c 未知，跟 c 已知这两种情况：

(1) 在 c 未知的时候，有：P(a,b,c)=P(c)P(a|c)P(b|c)，此时，没法得出 P(a,b) = P(a)P(b)，即 c 未知时，a、b 不独立。

(2) 在 c 已知的时候，有：P(a,b|c)=P(a,b,c)/P(c)，然后将 P(a,b,c)=P(c)P(a|c)P(b|c)带入式子中，得到：P(a,b|c)=P(a,b,c)/P(c) = P(c)P(a|c)P(b|c) / P(c) = P(a|c)*P(b|c)，即 c 已知时，a、b 独立。

head-to-tail



还是分 c 未知跟 c 已知这两种情况：

(1) c 未知时，有：P(a,b,c)=P(a)P(c|a)P(b|c)，但无法推出 P(a,b) = P(a)P(b)，即 c 未知时，a、b 不独立。

(2) c 已知时，有：$P(a,b|c)=P(a,b,c)/P(c)$，且根据 $P(a,c) = P(a)P(c|a) = P(c)P(a|c)$，可化简得到：

$P(a,b|c)==P(a,b,c)/P(c)= P(a)P(c|a)P(b|c)/P(c)=P(a,c)P(b|c)/P(c)=P(a|c)P(b|c)$

所以，在 c 给定的条件下，a，b 被阻断(blocked)，是独立的，称之为 head-to-tail 条件独立。

这个 head-to-tail 其实就是一个链式网络，如下图所示：

根据之前对 head-to-tail 的讲解，我们已经知道，在 xi 给定的条件下，xi+1 的分布和 x1,x2…xi-1 条件独立。意味着啥呢？意味着：xi+1 的分布状态只和 xi 有关，和其他变量条件独立。通俗点说，当前状态只跟上一状态有关，跟上上或上上之前的状态无关。这种顺次演变的随机过程，就叫做马尔科夫链（Markov chain）。

**朴素贝叶斯可以看做是贝叶斯网络的特殊情况：即该网络中无边，各个节点都是独立的。
**

朴素贝叶斯朴素在哪里呢？

一个特征出现的概率与其他特征（条件）独立；

隐含马尔可夫模型

马尔可夫链

有向图模型（贝叶斯网络）：用有向图表示变量间的依赖关系；

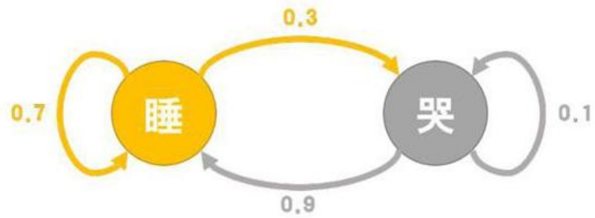
无向图模型（马尔可夫网）：用无向图表示变量间的相关关系。

HMM 就是贝叶斯网络的一种--虽然它的名字里有和“马尔可夫网”一样的马尔可夫。

对变量序列建模的贝叶斯网络又叫动态贝叶斯网络。HMM 就是最简单的动态贝叶斯网络。

注意，马尔可夫过程其原始模型是马尔可夫链。

该过程具有如下特性：在已知系统当前状态的条件下，它未来的演变不依赖于过去的演变。也就是说，一个马尔可夫过程可以表示为系统在状态转移过程中，第 T+1 次结果只受第 T 次结果的影响，即只与当前状态有关，而与过去状态，即与系统的初始状态和此次转移前的所有状态无关。



上图就是一个非常简单的马尔可夫链。两个节点分别表示睡和哭。几条边表示节点之间的转移概率。

睡之后，0.7 的可能又接着睡，只有 0.3 的可能变成哭。而哭之后，0.9 的可能是睡，也就有 0.1 的可能接着哭。

假设这是某个孩子的预报模型（这个孩子就只有哭和睡），则下一个状态只和上一个状态有关。和之前是在哭还是睡的状态没有关系。那么我们只要知道现在的状态，就可以推测接下来是睡还是哭的可能性了。

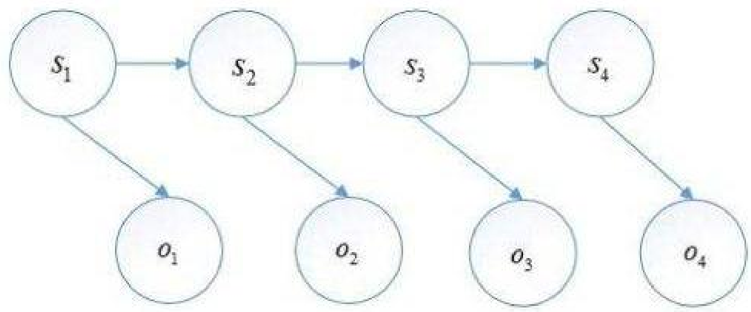
隐含马尔可夫模型（Hidden Markov Model, HMM）

HMM 定义

HMM 是一个关于时序的概率模型，它的变量分为两组：

- ✧ 状态变量{s1,s2,...,sr}，其中 st 属于 S，表示 t 时刻的系统状态；
- ✧ 观测变量{o1,o2,...,or}，其中 ot 属于 O，表示 t 实可的观测值。

状态变量和观测变量各自都是一个时间序列，每个状态/观测值都和一个时刻相对应，下图



一般假定状态序列是隐藏的，不能被观测到的，因此状态变量是隐变量，这就是 HMM 中的 hidden 的来源。

这个隐藏的，不可观测的状态序列是由一个马尔可夫链随机生成的，这是 HMM 中的第一个 M 的含义。

一条隐藏的马尔可夫链随机生成了一个不可观测的状态序列 state sequence，然后每个状态又对应生成了一个观测结果。这些观测值按照时序排列后就成了观测序列 observation sequence。这两个序列是一一对应的，每个对应的位置又对应着一个时刻。

一般而言，HMM 的状态变量取值是离散的，而观测变量的取值，则可以是离散的，也可以是连续的。不过为了方便讨论，也因为在大多数应用中观测变量也是离散的，因此，我们下面仅讨论状态变量和观测变量都是离散的情况。

HMM 基本假设

HMM 的定义建立在两个假设之上：

假设 1：假设隐藏的马尔可夫链在任意时刻 t 的状态只依赖于前一个时刻(t-1)的状态，与其它时刻的状态及观测无关，也与时刻 t 无关。

用公式表达就是：

$p(s_t \mid s_{t-1}, o_{t-1}, \ldots, s_1, o_1) = p(s_t \mid s_{t-1})$ t=1,2,...,T

这一假设又叫做齐次马尔可夫假设。

假设 2：假设任意时刻的观测只依赖于该时刻的马尔可夫链状态，与其它观测及状态无关。

用公式表达就是：

$p(o_t \mid s_r, o_r, \ldots, s_{t+1}, o_{t+1}, s_t, o_t, s_{t-1}, o_{t-1}, \ldots, s_1, o_1) = p(o_t \mid s_t)$

这叫观测独立性假设。

确定 HMM 的两个空间和三组参数

基于上述两个假设，我们可知：所有变量（包括状态变量和观测变量）的联合分布为：

$p(s_1, o_1, \ldots, s_r, o_r) = p(s_1)p(o_1 \mid s_1)\prod_{t=2}^{T} p(s_t \mid s_{t-1})p(o_t \mid s_t)$

1) 设 HMM 的状态变量（离散值），总共有 N 种取值，分别为：{S1,S2,...,Sn}
2) 观测变量，也是离散值，总共有 M 种取值，分别为：{O1,O2,...,Om}

那么，要确定一个 HMM，除了要指定其对应的状态空间 S 和观测空间 O 外，还需要三组参数，分别是：

✓ 状态转移概率：模型在各个状态间转换的概率，通常记作矩阵 A={a_{ij}}，N*N

其中 $a_{ij} = P(s_{t+1} = S_j \mid s_t = S_i)$,1≤i，j≤N 表示在任意时刻 t，若状态为 Si，则下一时刻状态为 Sj 的概率

✓ 输出观测概率：模型根据当前状态获得各个观测值的概率，通常记作矩阵 B={b_{ij}} ,N*M

其中 $b_{ij} = P(o_t = o_j \mid s_t = S_i)$ ，1≤i≤N，1≤j≤M 表示在任意时刻 t，若状态为 Si，则观测值 Oj 被获取的概率。

有些时候，Si 已知，但可能 Oj 是未知的，这个时候，b 就成了当时观测值的一个函数，因此也可以写作 $b_i(o) = P(o \mid s = S_i)$

✓ 初始状态概率：模型在初始时刻各状态出现的概率，通常记作 $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$

其中 $\pi_i = P(s_1 = S_i)$ ，1≤i≤N 表示模型的初始状态为 Si 的概率。

通常我们用λ=[A，B，Π]来指代这三组参数。

有了状态空间 S 和观测空间 O，以及参数λ，一个 HMM 就被确定了。

HMM 三个基本问题

概率计算问题

问题名称：概率计算问题，又称评价 evaluation 问题。
已知信息：

- ✓ 模型λ=[A，B，Π]
- ✓ 观测序列{o1,o2,...,or}

求解目标：计算在给定模型λ下，已知观测序列 O 出现的概率：P(O|λ)，也就是说，给定观测序列，求它和评估模型之间的匹配度。

预测问题

问题名称：预测问题，又称解码问题。
已知信息：

- ✓ 模型λ=[A，B，Π]
- ✓ 观测序列{o1,o2,...,or}

求解目标：计算在给定模型λ下，使已知观测序列 O 的条件概率 P(O|S)最大的状态序列{s1,s2,...,sr}，即给定观测序列，求最有可能与之对应的状态序列。

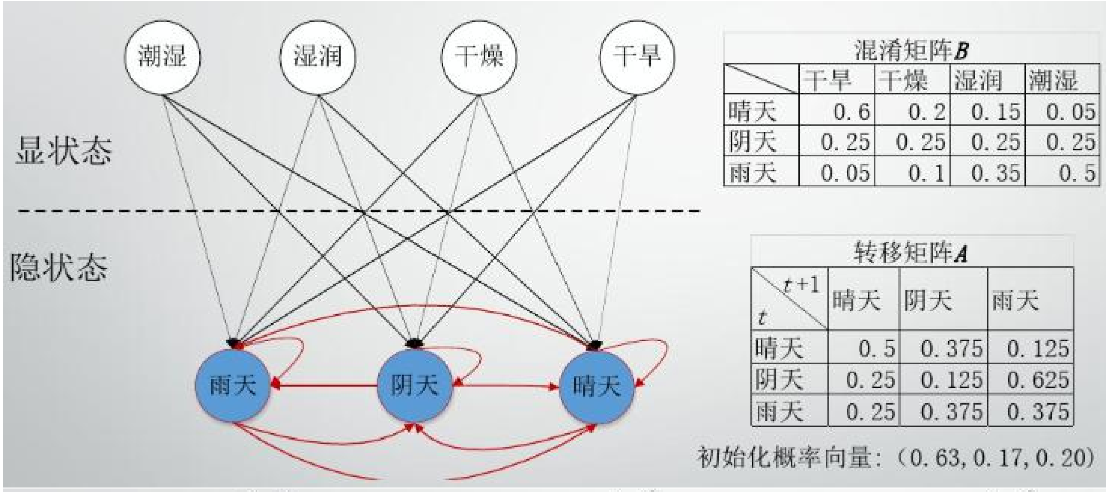
学习问题

问题名称：学习问题又称为训练问题。
已知信息：

- ✓ 观测序列{o1,o2,...,or}
- ✓ 或许也会给定与之对应的状态序列{s1,s2,...,sr}

求解目标：估计模型λ=[A，B，Π]参数，使得该模型下观测序列概率 P(O|λ)最大。也就是训练模型，使其最后地描述观测数据。
前两个问题是模型已经存在之后如何使用模型的问题，而最后一个则是如何通过训练得到模型的问题。

前向算法

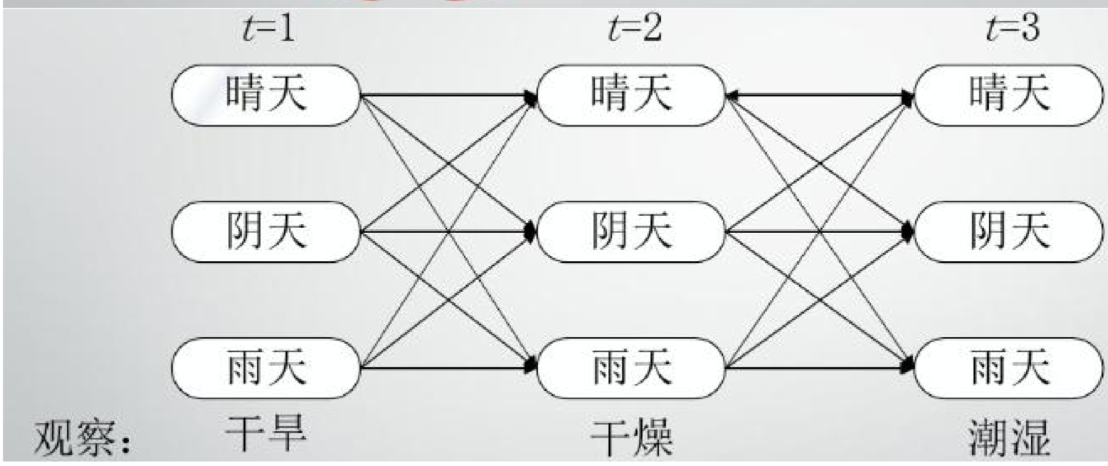


混淆矩阵*B*

	干旱	干燥	湿润	潮湿
晴天	0.6	0.2	0.15	0.05
阴天	0.25	0.25	0.25	0.25
雨天	0.05	0.1	0.35	0.5

转移矩阵*A*

t \ $t+1$	晴天	阴天	雨天
晴天	0.5	0.375	0.125
阴天	0.25	0.125	0.625
雨天	0.25	0.375	0.375



首先计算最初的状态，Π，初始 t0 是干旱：

$$\alpha_1\begin{pmatrix}\text{Sunny}\\\text{Cloudy}\\\text{Rain}\end{pmatrix}=\begin{pmatrix}0.6\\0.25\\0.05\end{pmatrix}\circ\begin{pmatrix}0.63\\0.17\\0.20\end{pmatrix}=\begin{pmatrix}0.63\times0.6\\0.17\times0.25\\0.20\times0.05\end{pmatrix}=\begin{pmatrix}\mathbf{0.378}\\0.0425\\0.01\end{pmatrix}$$

a1(晴天)=0.63x0.6=0.378

a1(阴天)=0.17x0.25=0.0425

a1(雨天)=0.20x0.05=0.01

第二天状态转移概率，因为后一天状态由前一天来决定，马尔可夫性，同时第二天转移概率也受第二天的显状态影响。

注意：这里因为第二时刻是干燥，所以不应该是 0.6，0.25，0.05 去算！！！

$$\alpha_2\begin{pmatrix}\text{Sunny}\\\text{Cloudy}\\\text{Rain}\end{pmatrix}=\begin{pmatrix}0.6\\0.25\\0.05\end{pmatrix}\circ\left|\begin{vmatrix}0.378 & 0.0425 & 0.01\end{vmatrix}\times\begin{vmatrix}0.5 & 0.375 & 0.125\\0.25 & 0.125 & 0.625\\0.25 & 0.375 & 0.375\end{vmatrix}\right|^{T}$$

$$=\begin{pmatrix}0.6\\0.25\\0.05\end{pmatrix}\circ\begin{pmatrix}0.378\times0.5+0.0425\times0.25+0.01\times0.25\\0.378\times0.375+0.0425\times0.125+0.01\times0.375\\0.378\times0.125+0.0425\times0.625+0.01\times0.375\end{pmatrix}$$

$$=\begin{pmatrix}0.6\\0.25\\0.05\end{pmatrix}\circ\begin{pmatrix}0.202125\\0.1508125\\0.0775625\end{pmatrix}$$

$$=\begin{pmatrix}\mathbf{0.121275}\\0.037703125\\0.003878125\end{pmatrix}$$

注意，这里每个 a2(j)都来自三个 a1(j)！！！

a2(晴天)=0.2*(0.378*0.5+0.0425*0.25+0.01*0.25)=0.040425

a2(阴天)=0.25*(0.378*0.375+0.0425*0.125+0.01*0.375)=0.037703125

a2(雨天)=0.1*(0.378*0.125+0.0425*0.625+0.01*0.375)=0.00775625

注意，这里中间的行向量，是错的，应该由上面的数值替代！！！

$$\alpha_3\begin{pmatrix}\text{Sunny}\\\text{Cloudy}\\\text{Rain}\end{pmatrix}=\begin{pmatrix}0.05\\0.25\\0.5\end{pmatrix}\circ\left|\begin{vmatrix}0.121275 & 0.037703125 & 0.003878125\end{vmatrix}\times\begin{vmatrix}0.5 & 0.375 & 0.125\\0.25 & 0.125 & 0.625\\0.25 & 0.375 & 0.375\end{vmatrix}\right|^{T}$$

$$=\begin{pmatrix}0.003551640625\\0.012911328125\\\mathbf{0.0200890625}\end{pmatrix}$$

a3(晴天)=0.05*(0.040425*0.5+0.037703125*0.25+0.00775625*0.25)

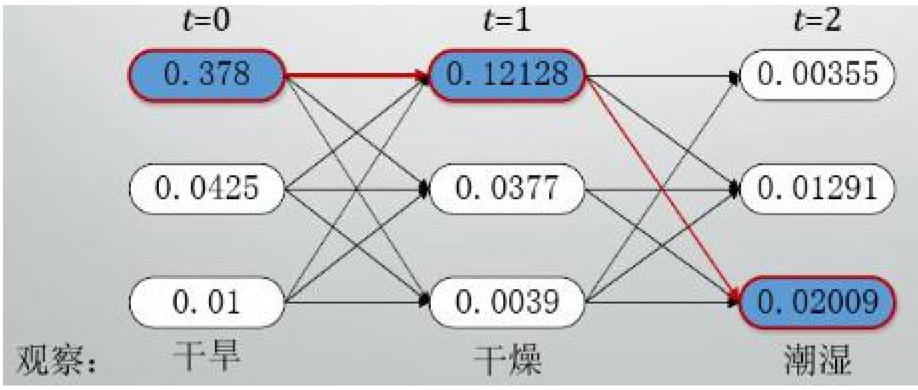
=0.0015788671875

a3(阴天)=0.25*(0.040425*0.375+0.037703125*0.125+0.00775625*0.375)

=0.00569521484375

a3(雨天)=0.5*(0.040425*0.125+0.037703125*0.625+0.00775625*0.375)

=0.0157630859375



最后结果是，晴天->晴天->雨天，但是注意，上面图中的数字是错误的！

缺点：随着矩阵的规模增大，算法的计算量是灾难性的。

hmm.py

维特比算法

可以发现 HMM 的计算结果呈现出如下的规律性：

- ✓ 最有路径是有网络中概率最大的节点构成的一条路径。
- ✓ 初始状态矩阵Π和 t0 时刻的转移概率决定了第一天天气状态的所有概率。但是，对后续各天气状态产生决定影响的只有最大概率的那个天气状态！
- ✓ 之后的每一天都如此，t+1 时刻的概率同时受到 t 时刻转移概率和当天的显状态影响。但是只有最大概率的那个天气状态才对后续状态产生决定性的影响。

定义δ(i,t)为所有序列中在 t 时刻以状态 i 终止时的最大概率。当然它所对应的那条路径称为部分最优路径。δ(i,t)对每个(i,t)都是存在的。这样我们就可以顺腾摸瓜找下去，在序列的最后一个状态找到整个序列的最优路径。

初始状态的计算与前面相同。

δ(晴天，1)=0.63x0.6=0.378

δ(阴天，1)=0.17x0.25=0.0425

δ(雨天，1)=0.20x0.05=0.01

然后计算最大概率的那个天气状态：概率为 0.378

接下来计算第二天的状态转移概率：

δ(晴天，2)=max(0.378*0.5，0.0425*0.25，0.01*0.25)*0.2=0.0378

δ(阴天，2)=max(0.378*0.375，0.0425*0.125，0.01*0.375)*0.25=0.0354375

δ(雨天，2)=max(0.378*0.125，0.0425*0.625，0.01*0.375)*0.1=0.004725

然后计算最大概率的那个天气状态：概率为 0.0378

最后计算第三天的状态转移概率：

δ(晴天，3)=max(0.0378*0.5，0.0354375*0.25，0.004725*0.25)*0.05=0.000945

δ(阴天 ,3)=max(0.0378*0.375 ,0.0354375*0.125 ,0.004725*0.375)*0.25=0.00354375

δ(雨天 ,3)=max(0.0378*0.125 ,0.0354375*0.625 ,0.004725*0.375)*0.5=0.011074218

最后结果是，晴天->晴天->雨天

案例

假设有三个盒子，编号为 1,2,3；每个盒子都装有黑白两种颜色的小球，球的比例。如下：

编号	白球	黑球
1	4	6
2	8	2
3	5	5

按照下列规则的方式进行有放回的抽取小球，得到球颜色的观测序列：

1、按照π的概率选择一个盒子，从盒子中随机抽取出一个球，记录颜色后放回盒子中；

2、按照某种条件概率选择新的盒子，重复该操作；

3、最终得到观测序列："白黑白白黑"

例如： 每次抽盒子按一定的概率来抽，也可以理解成随机抽。

第 1 次抽了 1 号盒子①，第 2 次抽了 3 号盒子③，第 3 次抽了 2 号盒子②.... ； 最终如下：

①→③→②→②→③ 状态值

白→黑→白→白→黑 观测值

1、状态集合： S={盒子 1，盒子 2，盒子 3}

2、观测集合： O={白，黑}

3、状态序列和观测序列的长度 T=5 (我抽了 5 次)

4、初始概率分布： π 表示初次抽时，抽到 1 盒子的概率是 0.2，抽到 2 盒子的概率是 0.5，抽到 3 盒子的概率是 0.3。

5、状态转移概率矩阵 A：a11=0.5 表示当前我抽到 1 盒子，下次还抽到 1 盒子的概率是 0.5；

6、观测概率矩阵 - 混淆矩阵 - 为了不和之前的混淆矩阵概念冲突，可以称之为发射矩阵，即从一个状态发射到另一个状态： B：如最初的图，b11=第一个盒子抽到白球概率 0.4，b12=第一个盒子抽到黑球概率 0.6;

$$A=\begin{bmatrix}0.5 & 0.4 & 0.1\\0.2 & 0.2 & 0.6\\0.2 & 0.5 & 0.3\end{bmatrix}\quad B=\begin{bmatrix}0.4 & 0.6\\0.8 & 0.2\\0.5 & 0.5\end{bmatrix}\quad \pi=\begin{pmatrix}0.2\\0.5\\0.3\end{pmatrix}$$

HMM 案例思考

在给定参数π、A、B 的时候，得到观测序列为"白黑白白黑"的概率是多少?

这个时候，我们不知道隐含条件，即不知道状态值：①→③→②→②→③ ；

我们如何根据π、A、B 求出测序列为"白黑白白黑"的概率？

下面给出解决方案。

概率计算问题

前向-后向算法 给定模型λ=(A,B,π)和观测序列 Q={q1,q2,...,qT}，计算模型λ下观测到序列 Q 出现的概率 P(Q|λ)；

回顾上面的案例，λ=(A,B,π)已知。观测到序列 Q=白→黑→白→白→黑，但我们不知道 状态序列 I=①→③→②→②→③；我们要求解 P(Q|λ)，即 Q=白→黑→白→白→黑 这个观测序列发生的概率。可以用前向-后向算法来实现。

预测问题

Viterbi 算法 给定模型λ=(A,B,π)和观测序列 Q={q1,q2,...,qT}，求给定观测序列条件概率 P(I|Q，λ)最大的状态序列 I。

已知观测到序列 Q=白→黑→白→白→黑，当我们得到λ=(A,B,π)后，我们用 Viterbi 算法 求出在哪一种状态序列发生的可能性最大，即，求出状态序列 I=①→③→②→②→③；即，抽取什么样的盒子顺序，更可能得到白→黑→白→白→黑这种结果。

学习问题

Baum-Welch 算法(状态未知) 已知观测序列 Q={q1,q2,...,qT}，估计模型λ=(A,B,π)的参数，使得在该模型下观测序列 P(Q|λ)最大。

Baum-Welch 算法是 EM 算法的一个特例，专门用来求解隐马尔科夫中隐状态参数λ=(A,B,π)。即：根据已知的观测到序列 Q=白→黑→白→白→黑，去寻找整个模型的一组隐状态参数λ=(A,B,π)，使得在模型中观测序列发生的可能性 P(Q|λ)最大。