### 隐含马尔科夫模型（HMM，Hidden Markov Model）

到目前为止，本书所介绍的各种识别和分类方法所针对的都是以特征矢量方式描述的模式。但是在有些实际问题中，模式是以序列形式出现的，例如基因序列，消费者的一系列购买行为等等。还有一些模式具有明显的时间延续性，例如在语音识别和人体行为识别中，声音和人体的动作都是发生在一个时间段内，如果将这个时间段内的音频或视频信号采用特征矢量的方式描述，就会损失掉信号的先后次序变化信息。对于有时间延续性的模式，一种常用的描述方法是将信号划分为一系列短的时间段，在每一个短时间段内提取信号的特征（例如语音信号的各种频域特征），并将这些特征连接形成一个特征矢量的序列，这样就可以有效地描述信号在不同时间段内的变化信息。

如果我们希望采用贝叶斯分类器识别以序列形式出现的模式，就需要构建描述序列的概率密度函数，计算每个类别产生出需要识别序列的概率密度，隐含马尔科夫模型就是一种常用的序列概率密度描述模型。

我们将以序列形式描述的模式表示为：，一般将序列称作“观察序列”，上标表示序列的长度，序列中的元素称作“时刻”的观察值，可以是属于一个有限集合的离散符号（例如在基因序列中每个观察值只能是4个符号之一），也可以是一个特征矢量。为了方便问题的讨论，我们以观察值为离散的情况来介绍隐含马尔科夫模型，在本小节的最后介绍观察值为连续特征矢量情况下的处理方法。

**马尔科夫模型**

在介绍隐含马尔科夫模型之前，我们需要首先了解一下马尔科夫模型。马尔科夫模型由若干个状态构成，模型当前所处的状态只与之前的状态有关，与之后的状态无关，由此所产生的一类状态转移过程称为马尔科夫过程。这里，我们只讨论一种最简单的马尔科夫模型--离散时间有限状态一阶马尔科夫模型。

如图7.5所示，一阶马尔科夫模型包含个状态，在时刻模型处于状态，经过个时刻，模型可以产生出一个状态转移序列：。在一阶马尔科夫模型中，时刻模型处于什么状态只与时刻的状态有关，与其它时刻的状态都无关。将第时刻模型处于状态，而时刻处于状态的概率定义为：



称为状态转移概率，与时间无关，即在任何时刻，模型由状态转移到的概率是相同的。在初始时刻模型处于什么状态也是随机的，定义第一个时刻处于状态的概率为，这样就得到了描述一阶马尔科夫模型的参数：

， （7.40）

显然，参数应该满足如下关系：

， （7.41）



图7.5 4状态一阶马尔科夫模型状态转移图

对于给定参数的一阶马尔科夫模型，可以计算由这个模型产生出特定状态转移序列的概率。例如图7.5的一阶马尔科夫模型产生出状态转移序列的概率为：



**【例7.4】**某个城市天气的变化可以采用图7.6所示的一阶马尔科夫模型描述。每天的天气有4种状态{晴、阴、雨、雪}，分别按照1~4进行编号，每种天气发生的初始概率为：。计算连续7天是晴天的概率，以及前3天晴、后4天下雨的概率。

**解**：将图7.6中的状态转移概率表示为矩阵形式：



连续7个晴天的概率：



前3天晴、后4天下雨的概率：



■



图7.6 天气变化马尔科夫模型

**隐含马尔科夫模型**

隐含马尔科夫模型的内部是一个一阶马尔科夫模型，每一时刻都依据概率发生着状态转移，只不过状态转移的过程我们是观察不到的，是隐含的。我们所能够观察到的是每一时刻模型根据所处的状态产生出的一个“观察值”，经过个时刻之后，可以得到一个观察序列：。例如，图7.6中的马尔科夫模型描述的是其它城市的天气变化，你并不知道当地每天的天气情况，但是知道在当地一位朋友每天的活动情况，并且这些活动与天气密切相关，晴天出去散步或购物的可能性比较大，而雨天和雪天则很少外出。这位朋友每天的活动就可以看作是观察值，而天气就是隐含在马尔科夫模型中的状态。隐含马尔科夫模型中，不可见的状态是可见的观察值产生的内在“原因”。

在模式识别中，描述模式的序列是我们能够观察到的，而隐含马尔科夫模型描述的则是观察序列发生的概率。我们先从一种简单的情况入手，假设可能的观察值是有限的、离散的，第时刻的观察值。



图7.7 一阶隐含马尔科夫模型

隐含马尔科夫模型的参数除了初始状态概率和状态转移概率矩阵之外，增加了模型处于状态输出观察值的概率，所有的输出概率可以表示为状态输出概率矩阵：

，其中

隐含马尔科夫模型的参数为：。在隐含马尔科夫模型中存在着三个基本问题：估值问题，解码问题和学习问题。

**估值问题**

所谓估值问题是指，如何计算一个已知参数的HMM输出特定观察序列概率的问题。在模式识别中，待识模式以观察序列的形式出现，而每个类别的条件概率（密度）则由不同的隐含马尔科夫模型所描述。当输入一个待识模式时，需要计算每个类别对应的HMM输出这个观察序列的概率，然后根据贝叶斯判别准则进行分类。

估值问题的计算实际上并不复杂，我们先来看一下在已知HMM状态转移序列的条件下，输出观察序列的概率：

（7.42）

实际上，在隐含马尔科夫模型中，状态转移序列是不可见的、未知的，因此根据全概公式（参见附录C.4）可以计算HMM输出观察序列的概率：

 （7.43）

其中，求和式针对的是HMM在个时刻内所有可能的状态转移序列，是所有可能的状态转移数量。概率由一阶马尔科夫模型计算：

 （7.44）

公式（7.42~7.44）书写比较复杂，实际计算上并不是很困难，通过例7.5我们来看一下具体的计算过程。

【**例7.5**】 某城市的天气变化可以用例7.4中图7.6的马尔科夫模型描述。某人生活在这个城市中，每天的活动包括{散步、购物、做家务}，分别按照1~3进行编号。活动情况与天气状态之间的相关性由如下状态输出概率矩阵表示：



计算此人连续三天的活动分别为散步、做家务和购物的概率。

**解**：连续三天的天气状态包括种可能性：，，…，，计算每一种可能天气序列发生的概率以及在此天气条件下相应活动的概率。

连续三天晴天的概率：



三天晴天的条件下散步、做家务和购物的概率：



因此：

 （7.45）

同理：

 （7.46）

 （7.47）

 （7.48）

… … …





连续三天的活动为散步、做家务和购物的概率:



■

按照（7.43）式计算估值问题，计算复杂度为。当状态数较多，序列的长度较长时，计算量大。例如，计算5个状态的HMM产生长度为20序列的概率，需要计算次乘法。可以看出，随着序列长度的增加，计算复杂度呈指数增长。估值问题是否存在多项式时间复杂度的计算方法？

仔细观察例7.5中的计算过程就会发现，式（7.45）~（7.48）前4项的乘积是相同的，均为，不同的只是后两项。然而在计算过程中前4项的乘积我们计算了4次，实际上只需要计算1次即可，这样就可以有效地减小计算量。根据这样的思想，我们来介绍一种估值问题的有效算法--前向算法。



图7.8 前向算法示意图

首先我们将前向算法的过程用图7.8来表示，在图中每一列的节点表示同一时刻HMM有可能所处的状态，经过个时刻，因此共有列的节点。在每一个节点上我们定义一个值，表示当前时刻HMM处于该节点，并输出相应观察值的概率。例如，表示HMM在第时刻处于第个状态，并且输出序列的概率：



显然，第1列的节点可以直接计算出值：

， （7.49）

如果我们计算出了第列每个节点的值，就可以很容易得到HMM模型输出序列的概率。因为，HMM在最后一个时刻只可能处于最后一列个状态中的某一个，所以：

 （7.50）

下面的问题就是如何能够迭代计算出第2列到第列每个节点的值了。由于HMM中的状态转移是一阶马尔科夫过程，因此每一个节点的值只与前一列的节点有关。如果在第时刻模型处于状态，并且产生出，这个事件发生的概率为；在时刻转移到状态，并且输出，这个事件发生的概率是；上述两个事件同时发生的概率为。考虑到在前一个时刻HMM有可能处于个状态中的任何一个，因此根据全概公式可以得到递推公式：

，， （7.51）

总结（7.49）~（7.50）式，可以得到HMM估值问题的前向算法：

|  |
| --- |
| **前向算法** |
| * 初始化：； * 计算第1列每个节点的值：，； * 迭代计算第2至列每个节点的值：   ，  ；   * 输出： |

前向算法中每个节点需要计算次乘法，共有个节点。因此，忽略低阶项可以得到总的计算复杂度为，远远少于直接计算的方法。

**解码问题**

解码问题是指，给定已知参数的隐含马尔科夫模型，计算最有可能产生出特定观察序列的状态转移序列的问题。HMM中的状态转移过程是不可见的，解码问题实际上关心的是，如何根据模型输出的结果（观察序列）来推测产生出该结果的内部机理（状态转移序列）的问题。

解码问题需要求解的是如下优化问题：

 （7.52）

根据贝叶斯公式：



由于与优化变量无关，并且只存在有限的取值可能性，因此解码问题可以转化为求解如下的优化问题：

 （7.53）

其中，取所有可能的长度为的状态转移序列。同估值问题类似，只要我们计算出种可能的状态转移序列发生的概率，以及在状态转移序列发生条件下产生出观察序列的概率，寻找两项乘积的最大值就可以很容易地解决解码问题。

**【例7.6】** 在例7.5中，如果我们知道此人连续三天的活动分别为散步、做家务和购物，推测这三天中该城市最有可能的天气状况。

**解**：按照例7.5同样的方式，计算出所有种可能的连续三天天气状态发生的概率，以及在不同天气状态下此人完成散步、做家务和购物活动的概率；比较64种可能天气状况下两项概率乘积的大小，寻找最大值可得：





■

同估值问题一样，直接求解解码问题的计算量比较大，时间复杂度为。从对估值问题的分析可以看出，解码问题也存在多项式时间复杂度的求解算法，这就是著名的Viterbi算法。

重新观察图7.8我们会发现，解码问题实际上是要在每一列上选择且只选择一个节点，由这些节点构成一个开始于第1列，结束于第列的路径，使得路径上所有状态组成的状态转移序列，产生出观察序列的概率最大。代替前向算法中的值，Viterbi算法在每个节点上定义了一个值，表示在第时刻HMM处于第个状态，并且输出序列最优路径的概率值。利用HMM的一阶马尔科夫性可以得到类似于值的迭代公式，只不过将公式（7.51）的求和变为了最大值：

，， （7.54）

迭代计算出全部节点值之后，只需要在第列找到值最大的节点，就可以确定最优状态转移序列在第时刻的状态。为了找到1至时刻的最优状态，需要在迭代过程的每个节点上记录此节点最优路径上前一时刻的状态，保存在第时刻HMM处于第个状态的最优路径上时刻的状态。在迭代计算值之后，确定了第时刻的最优状态，可以根据节点的值回溯出整个的最优状态转移序列。

这样我们就可以已得到类似于前向算法的Viterbi解码算法：

|  |
| --- |
| **Viterbi算法** |
| * 初始化：； * 计算第1列每个节点的值：   ，，；   * 迭代计算第2至列每个节点的值：     ，  ；   * 最优路径的概率：      * 回溯最优路径：     ， |

分类问题一般并不关心HMM中状态是如何转移的，只需要计算模型输出观察序列的概率。但在有些应用中，也可以将最优状态转移序列输出的概率用于分类。

**学习问题**

学习问题解决的是，如何根据一组训练模式的观察序列集合，学习隐含马尔科夫模型参数的问题。HMM描述的是观察序列发生的概率，因此对它的学习仍然是一个参数估计问题，可以采用最大似然估计的方法求解如下的优化问题：

 （7.55）

同估值问题和解码问题比较起来，HMM的学习问题要复杂得多。与高斯混合模型参数的估计类似，解决学习问题存在的主要困难在于，我们并不知道HMM是经过一个什么样的状态转移过程，产生出的每一个观察序列。由于状态转移序列是隐含的，是“缺失的”，因此HMM的参数需要采用EM算法进行迭代估计。迭代公式具体的推导过程比较复杂，有兴趣的读者可以参考文献[10]的相关内容，下面简单介绍一下隐含马尔科夫模型参数估计的迭代算法--前向后向算法，也被称为Baum-Welch算法。为了简化问题，首先讨论只针对单个训练矢量的迭代公式。

在给出算法迭代公式之前，需要定义几个变量。

**α值**：在估值问题的前向算法中我们定义了一个值，表示在时刻HMM处于状态，并且在期间产生出观察序列的概率；

**β值**：类似的，在图7.8的每个节点上还可以定义一个值，表示在时刻HMM处于状态，并且在期间产生出观察序列的概率；

**γ值**：不同于和，是定义在相邻两列任意两个节点之间的。表示当HMM输出观察序列时，在时刻处于状态，时刻处于状态的概率。

值的计算同前向算法一样，可以根据公式（7.49）和（7.51）由第1列向最后一列迭代；值的计算与类似，只不过是由最后一列向第1列迭代：

， （7.56）

 （7.57）

根据的定义以及贝叶斯公式可以得到：

 （7.58）

其中概率表示，HMM输出观察序列，并且在时刻处于状态，时刻处于状态的概率。由图7.9可以看出，这样一个事件可以分解为3个独立的子事件：第一条虚线的左侧，HMM在时刻处于状态，并且输出序列的概率，由描述；两条虚线之间，在时刻由状态转移到，并且输出的概率，由计算；第二条虚线的右侧，HMM在时刻处于状态，并且输出序列的概率，由描述。



图7.9 值计算示意图

当时，值的相应计算公式为：

，

有了值可以很容易地得到模型参数估计的迭代公式。首先，根据全概公式，初始概率是在时刻模型处于状态的概率，因此的迭代估计公式为：

 （7.59）

在1至时刻之间，HMM由状态转移到的期望次数为：，而由状态转移到任意一个状态的期望次数为：，因此的迭代估计公式为：

 （7.60）

在1至时刻之间，HMM在状态上输出观察值的期望次数为：，因此的迭代估计公式为：

 （7.61）

总结上述过程可以得到HMM的学习算法：

|  |
| --- |
| **Baum-Welch算法** |
| * 随机初始化HMM参数，输入训练序列； * 前向计算：   ，，，；  ；   * 后向计算：   ，，，；   * 计算值：   ，，；  ，   * 重新估计HMM参数：   ， ，  ，，   * 迭代，直到满足收敛条件为止。 |

**HMM相关问题的讨论**

解决隐含马尔科夫模型学习问题的Baum-Welch算法是一种EM迭代算法，因此其收敛性是可以得到保证的。学习算法的收敛可以根据似然函数是否超过设定的阈值来判断，也可以根据两轮迭代中似然函数的差异满足一定的收敛精度来判断。同高斯混合模型的学习算法一样，前向后向算法并不能保证每次都收敛于似然函数的最大值点，只能保证收敛于极大值点，在实际应用中需要根据具体问题设置适合的初始值进行迭代，或者尝试不同初始值在多个学习结果中选择最优者。

公式（7.59）~（7.61）给出的是输入单个训练模式观察序列时的迭代计算方法，当输入为包含个观察序列的训练样本集时，可以每次输入一个序列进行单样本学习，也可以输入个样本进行批量学习。在批量学习时，、和值的计算与单样本学习相同，将第个序列的值以、和表示，，批量学习算法的迭代公式为：

， （7.62）

， （7.63）

，， （7.64）

在此之前我们讨论的都是离散型的HMM，输出的观察值为有限的离散值。当观察序列中的每一个元素是一个连续型的矢量时，需要使用连续隐马尔科夫模型描述每个类别产生每个模式的概率密度函数。连续HMM内部的状态转移仍然是一阶马尔科夫过程，不同的是由状态输出的是一个连续的观察矢量。因此对于每个状态来说，需要用函数来描述由状态输出观察矢量的概率密度。的函数形式需要在HMM的设计过程中做出假设，最常用的一种方式是假设为包含个分量的高斯混合模型：

 （7.65）

在估值问题和解码问题中，可以直接用每个状态的计算由此状态输出相应时刻观察矢量的概率密度；在学习问题中，对状态输出矩阵的估计需要转换为对每个状态GMM参数的估计。

，， （7.66）

 （7.67）

 （7.68）

 （7.69）

其中，是在时刻，由第个状态的第个分量高斯，输出观察矢量的概率。

隐含马尔科夫模型可以根据具体的应用问题使用不同的拓扑结构，图7.7是一种全连接模型，由一个状态可以依据概率转移到任意状态。图7.10和7.11给出了另外两种常用的拓扑结构：左-右模型和带跨越的左-右模型。



图7.10 左-右模型



图7.11 带跨越的左-右模型

不同的模型结构主要体现在状态转移矩阵上，例如语音识别中常用的左-右模型，就是限制矩阵中除主对角线及主对角线右侧元素之外均为0：



在左-右模型中还经常约束模型必须初始于状态，结束于。

下面分别给出解决HMM估值、解码和学习问题算法的实现代码：

|  |
| --- |
| 函数名称：HMMForward  参数：V--观察序列，PI，A，B--HMM模型参数  返回值：PV--HMM输出观察序列的概率  函数功能：隐含马尔科夫模型的前向估值算法 |
| function PV = HMMForward( V, PI, A, B )    T = length(V);  M = length(PI);  K = size(B,2);    Alpha = zeros(M,T);    Alpha(:,1) = PI'.\*B(:,V(1));    for t = 2:T  Alpha(:,t) = (Alpha(:,t-1)'\*A)'.\*B(:,V(t));  end    PV = sum(Alpha(:,T)); |

|  |
| --- |
| 函数名称：HMMViterbi  参数：V--观察序列，PI，A，B--HMM模型参数  返回值：S--HMM输出V可能性最大的状态转移序列，PV--由S输出V的概率  函数功能：隐含马尔科夫模型的Viterbi解码算法 |
| function [S, PV] = HMMViterbi( V, PI, A, B )    T = length(V);  M = length(PI);  K = size(B,2);    Delta = zeros(M,T);  Phi = zeros(M,T);    Delta(:,1) = PI'.\*B(:,V(1));    for t = 2:T  [y,id] = max(repmat(Delta(:,t-1),1,M).\*A);  Delta(:,t) = y'.\*B(:,V(t));  Phi(:,t) = id';  end    [PV,id] = max(Delta(:,T));    S = zeros(1,T);  S(T) = id;  for i = T-1:-1:1  S(i) = Phi(S(i+1),i+1);  end |

|  |
| --- |
| 函数名称：HMMTrain  参数：V—训练观察序列（Cell矩阵），IPI，IA，IB—模型参数的初始值  返回值：PI，A，B—模型参数的学习结果  函数功能：隐含马尔科夫模型的Baum-Welch学习算法 |
| function [PI A B] = HMMTrain( V, IPI, IA, IB )    PI = IPI; A = IA; B = IB;    n = length(V);  M = length(PI);  K = size(B,2);    iter = 0;  oldSPV = -10000000;    Theta = 0.01;    while true  TPI = zeros(1,M);  TA = zeros(M,M);  TB = zeros(M,K);    logSPV = 0;  for m = 1:n  T = length(V{m});  v = V{m};    Alpha = zeros(M,T);  Beta = zeros(M,T);  Gamma = zeros(M,M,T);    Alpha(:,1) = PI'.\*B(:,v(1));  for t = 2:T  Alpha(:,t) = (Alpha(:,t-1)'\*A)'.\*B(:,v(t));  end    PV = sum(Alpha(:,T));  logSPV = logSPV + log(PV);    Beta(:,T) = ones(M,1);  for t = T-1:-1:1  Beta(:,t) = A\*(Beta(:,t+1).\*B(:,v(t+1)));  end    Gamma(1,:,1) = PI.\*B(:,v(1))'.\*Beta(:,1)'/PV;  for t = 2:T  Gamma(:,:,t) = repmat(Alpha(:,t-1),1,M) .\* A .\* repmat(B(:,v(t))'.\*Beta(:,t)', M,1)/PV;  end    TPI = TPI + Gamma(1,:,1);  TA = TA + sum(Gamma(:,:,2:T),3);  for k = 1:K  id = find( v == k );  TB(:,k) = TB(:,k) + sum(sum(Gamma(:,:,id),3),1)';  end  end    logSPV = logSPV / n;    PI = TPI / sum(TPI);  A = TA ./ repmat(sum(TA,2),1,M);  B = TB ./ repmat(sum(TB,2),1,K);    iter = iter + 1;  fprintf( 'Iteration %d: %f\r', iter, logSPV );    if ( logSPV - oldSPV ) < Theta  break;  else  oldSPV = logSPV;  end  end |

在学习算法中，以两轮迭代训练观察序列集合的对数似然函数变化量是否超过收敛精度Theta作为收敛条件。由于对数似然函数的大小与训练序列的数量有关，因此变化量对样本数量做了平均。