第5章 模式识别基础

班级： 无61 姓名： 李溢 学号： 2016011235

1. **贝叶斯决策 (Bayes Decision Theory)**

考虑二维的两类分类问题，其中



1. 假设，请尝试计算器贝叶斯决策边界。
2. 假设类别的概率分布不变，先验概率为， 请问贝叶斯决策边界会发生什么变化。
3. 假设测试样本为，请问依照(2)中的条件，请利用贝叶斯决策判别该样本的类别。

（1）两类概率相等，因此贝叶斯决策可以转化为最大似然估计，由于两个类别的协方差矩阵相同，因此又可以转化为马氏距离判决

决策边界即为

（2）判决准则变为

决策边界处满足

因此决策边界只是**沿着法线方向进行了平移**。

（3）

由于，最终判决为

1. **最大似然估计（Maximum Likelihood Estimation, MLE）**

假设有两类样本,分别是



假设样本分布服从高斯分布，请分别给出其基于最大似然估计的均值和协方差矩阵，并在此基础上结合先验概率估计给出贝叶斯决策边界。

类别

均值：

协方差：

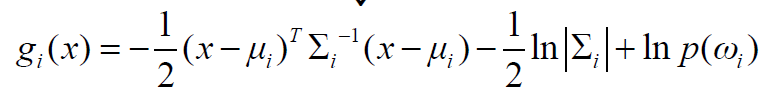
类别

均值：

协方差：

先验概率：

贝叶斯判决准则：



判决边界即为

1. **隐含马尔可夫模型（Hidden Markov Model, HMM）**

考虑采用HMM对DNA进行分析，DNA序列的碱基有腺嘌呤A、胞嘧啶C、鸟嘌呤G和胸腺嘧啶T等4种基本类型，假设有一个隐含状态控制着DNA序列的生成。有两个可能的状态，假定模型有如下的状态转移概率：

，，，

由状态到观测值的发射概率为：

，，，

，，，

初始状态分布为

假定观测序列，请计算：

(1)，要求采用前向变量法求解，请写出计算过程。

(2) 最可能的隐含状态序列，要求采用Viterbi算法求解，请写出计算过程。

（1）

（2）Viterbi算法

因此最有可能的序列是

1. **（上机题）隐含马尔可夫模型（Hidden Markov Model, HMM）**

考虑隐含马尔科夫模型对序列进行分类，序列的观测状态有四种，分别是A,B,C,D,每个序列有3个隐含状态，分别为S1, S2, S3; 状态间的转移是全连通的。下表示训练序列：

|  |  |  |
| --- | --- | --- |
| 样本 | 第一类 | 第二类 |
| 1 | AABBCCDD | DDCCBBAA |
| 2 | ABBCBBDD | DDABCBA |
| 3 | ACBCBCD | CDCDCBABA |
| 4 | AD | DDBBA |
| 5 | ACBCBABCDD | DADACBBAA |
| 6 | BABAADDD | CDDCCBA |
| 7 | BABCDCC | BDDBCAAAA |
| 8 | ABDBBCCDD | BBABBDDDCD |
| 9 | ABAAACDCCD | DDADDBCAA |
| 10 | ABD | DDCAA |

试结合隐马尔科夫模型的编程实现（**推荐使用python例程**，不限编程语言），完成如下操作：

1. 输出对应于两类的HMM模型参数，包括状态转移概率矩阵A，观测概率矩阵B；
2. 假设两类具备相同的先验概率，对以下序列评估结合上述两类的HMM模型参数评估其出现概率，并进行分类。序列包括：ABBBCDDD, DADBCBAA, CDCBABA, ADBBBCD。

Python例程见附件Easy\_HMM\_tiny.zip

matlab例程如下：

seqs = { double('AABBCCDD'), double('ABBCBBDD'), double('ACBCBCD'), …….. }; %输入序列

A\_o = []; %设置初始化状态转移概率矩阵

B\_o = []; %设置初始化观测概率矩阵

% 调用hmmtrain函数估计状态转移概率以及观测概率

[A\_e, B\_e] = hmmtrain(seqs, A\_o, B\_o, 'Symbols', double('ABCD') );

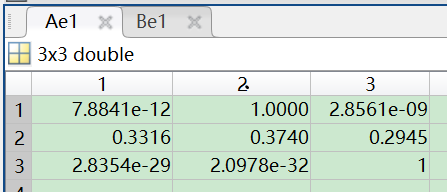
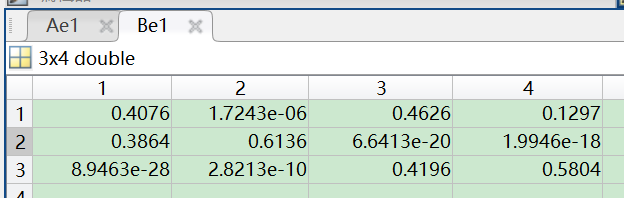
% 调用hmmdecode返回测试序列出现对数似然度（出现概率的对数）

[ ~, log\_prob\_seq ] = hmmdecode(double('ABBBCDDD'), A\_e, B\_e, 'Symbols', double('ABCD'));

% 输出测试序列由该HMM产生的概率

prob\_seq = exp( log\_prob\_seq );

（1）**第一个类别**

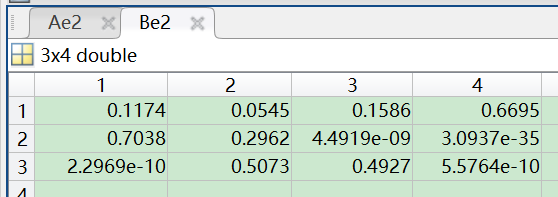
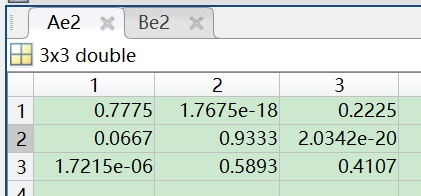
 

可以近似认为

转移矩阵：

发射矩阵：

**第二个类别**



可以近似认为

转移矩阵：

发射矩阵：

（2）对应于第1题目中算法得到的转移矩阵和发射矩阵，各个序列的出现概率及分类为

|  |  |  |  |
| --- | --- | --- | --- |
| 序列 |  |  | 分类 |
| ABBBCDDD | 1.1278e-04 | 1.6489e-06 |  |
| DADBCBAA | 9.0971e-17 | 9.4989e-05 |  |
| CDCBABA | 2.2439e-32 | 3.3066e-04 |  |
| ADBBBCD | 3.8515e-05 | 3.1676e-06 |  |

**实验过程中的一点发现：**

但是我发现一个问题就是matlab的hmmtrain函数**在多次运行时，最终输出的结果并不相同**。之后我使用了官方文档给出的示例程序进行测试，发现同样会给出不同的结果。

% 官方文档示例程序

tr = [0.95,0.05;

0.10,0.90];

e = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6;

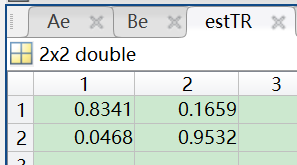
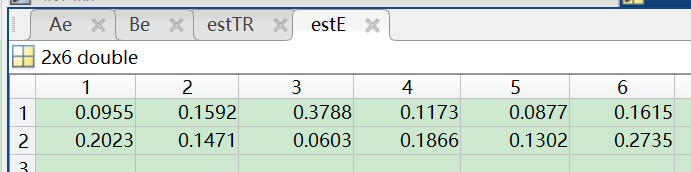
1/10, 1/10, 1/10, 1/10, 1/10, 1/2;];

seq1 = hmmgenerate(100,tr,e);

seq2 = hmmgenerate(200,tr,e);

seqs = {seq1,seq2};

[estTR, estE] = hmmtrain(seqs,tr,e);

我查看了hmmtrain.m源码之后，发现该函数设置了最大迭代次数500，如果500次后仍不收敛，则会在控制台输出一个未收敛的提示。但是我在每次代码运行中都没有看到这个提示，说明算法最终的确是收敛到了某一个结果，但是这个结果却不一定与真实值相同或者接近。除此之外我又用python示例程序测试，迭代次数改为500，发现运行多次时最终结果仍旧不同。

1）使用题目中的训练样本时，我使用unifrnd函数初始化两个矩阵，所以每次设置的初始化值都不同；

2）使用官方文档的示例程序时，每次设置的训练样本seq1，seq2都不相同；

由此看来**初始化值**和**训练样本的分布**对于Baum-Welch算法的最终结果有很大影响。

测试代码：

clear all; clc;

seq1 = { double('AABBCCDD'), double('ABBCBBDD'), double('ACBCBCD'), ...

double('AD'), double('ACBCBABCDD'), double('BABAADDD'), ...

double('BABCDCC'), double('ABDBBCCDD'), double('ABAAACDCCD'), ...

double('ABD')}; %输入序列

seq2 = { double('DDCCBBAA'), double('DDABCBA'), double('CDCDCBABA'), ...

double('DDBBA'), double('DADACBBAA'), double('CDDCCBA'), ...

double('BDDBCAAAA'), double('BBABBDDDCD'), double('DDADDBCAA'), ...

double('DDCAA')}; %输入序列

A0 = unifrnd(0,1,3,3); % 初始化状态转移概率矩阵

B0 = unifrnd(0,1,3,4); % 初始化观测概率矩阵

A0 = A0 ./ (sum(A0,2)\*ones(1,size(A0,2)));

B0 = B0 ./ (sum(B0,2)\*ones(1,size(B0,2)));

% 调用hmmtrain函数估计状态转移概率以及观测概率

[Ae1, Be1] = hmmtrain(seq1, A0, B0, 'Symbols', double('ABCD'));

% 调用hmmdecode返回测试序列出现对数似然度（出现概率的对数）

[ ~, log\_prob\_seq ] = hmmdecode(double('ABBBCDDD'), Ae1, Be1, 'Symbols', double('ABCD'));

% 输出测试序列由该HMM产生的概率

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('DADBCBAA'), Ae1, Be1, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('CDCBABA'), Ae1, Be1, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('ADBBBCD'), Ae1, Be1, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

disp('----------------------------')

% 调用hmmtrain函数估计状态转移概率以及观测概率

[Ae2, Be2] = hmmtrain(seq2, A0, B0, 'Symbols', double('ABCD'));

% 调用hmmdecode返回测试序列出现对数似然度（出现概率的对数）

[ ~, log\_prob\_seq ] = hmmdecode(double('ABBBCDDD'), Ae2, Be2, 'Symbols', double('ABCD'));

% 输出测试序列由该HMM产生的概率

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('DADBCBAA'), Ae2, Be2, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('CDCBABA'), Ae2, Be2, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);

[ ~, log\_prob\_seq ] = hmmdecode(double('ADBBBCD'), Ae2, Be2, 'Symbols', double('ABCD'));

prob\_seq = exp( log\_prob\_seq );

disp(prob\_seq);