模式识别与机器学习 23-24

第十一章作业

韦诗睿

202328002509044

https://github.com/RuiNov1st/UCAS_PRML_2324

2024 年 1 月 2 日

第十一章 概率图模型

题 1：HMM 转移矩阵

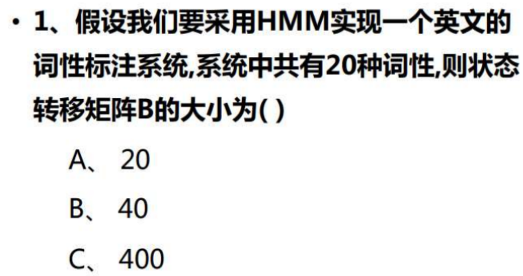

图 1: HMM 转移矩阵

答：C

由题, 词性即为 HMM 中的状态 M, 有 M=20。状态转移矩阵大小为 $M*M$, 因此为 $20\times 20=400$。

题 2：贝叶斯网络条件独立

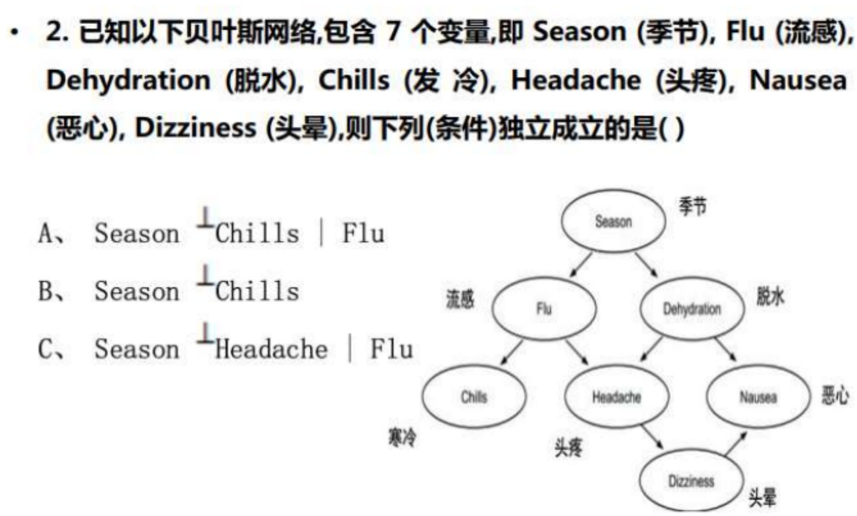

图 2: 贝叶斯网络条件独立

答：A

利用贝叶斯球:

- A）当 **Flu** 已知时，由父节点 **Season** 方向过来的球将被反弹，而由子结点 **Chills** 方向过来的球将被截止，因此结点 **Season** 和结点 **Chills** 之间不能联通，因此独立，A 成立；

- B）当 **Flu** 未知时总能使贝叶斯球通过，因此 **Season** 和结点 **Chills** 之间可以联通，因此不独立，B 不成立；
- C）当 **Flu** 已知时，由父节点 **Season** 方向过来的球将被反弹至 **Dehydration**，由于节点 **Dehydration** 未知总能使球通过，因此球可以到达 **Headache**，因此 **Season** 和 **Headache** 之间可以联通，不独立，C 不成立。

题 3：贝叶斯网络参数

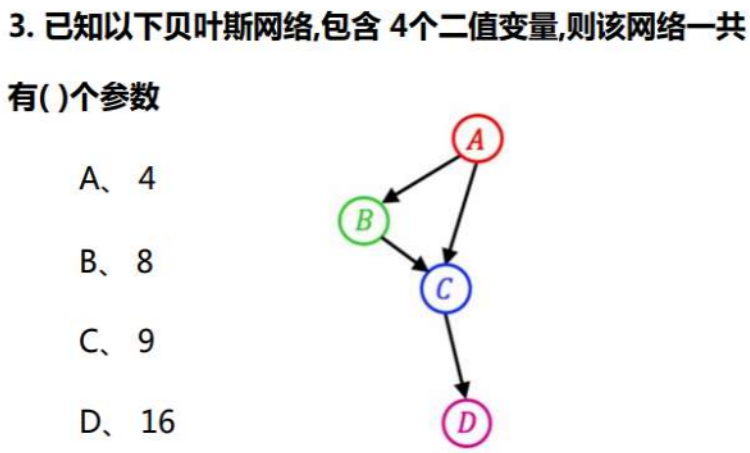

图 3: 贝叶斯网络参数

答：C

可以利用概率的归一性减少参数个数。由图的贝叶斯网络可写出联合概率分布为：

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|C)$$

由题，A、B、C、D 均为二值变量。若直接使用联合概率分布求，则需要 $2^4 - 1 = 15$ 个独立参数（可以利用概率相加之和为 1 去掉一个非独立的参数）。而分解为条件概率之后，有：
对于 $P(A)$，参数个数为 $2^1 - 1 = 1$；
对于条件概率也可以针对每一个条件组合利用概率的归一性，即：
$P(B|A)$ 的参数个数为 $2^1 \times (2^1 - 1) = 2$；
$P(C|A, B)$ 的参数个数为 $2^2 \times (2^1 - 1) = 4$；
$P(D|C)$ 的参数个数为 $2^1 \times (2^1 - 1) = 2$；
因此总参数个数为：$1 + 2 + 4 + 2 = 9$，选 C。

题 4：HMM

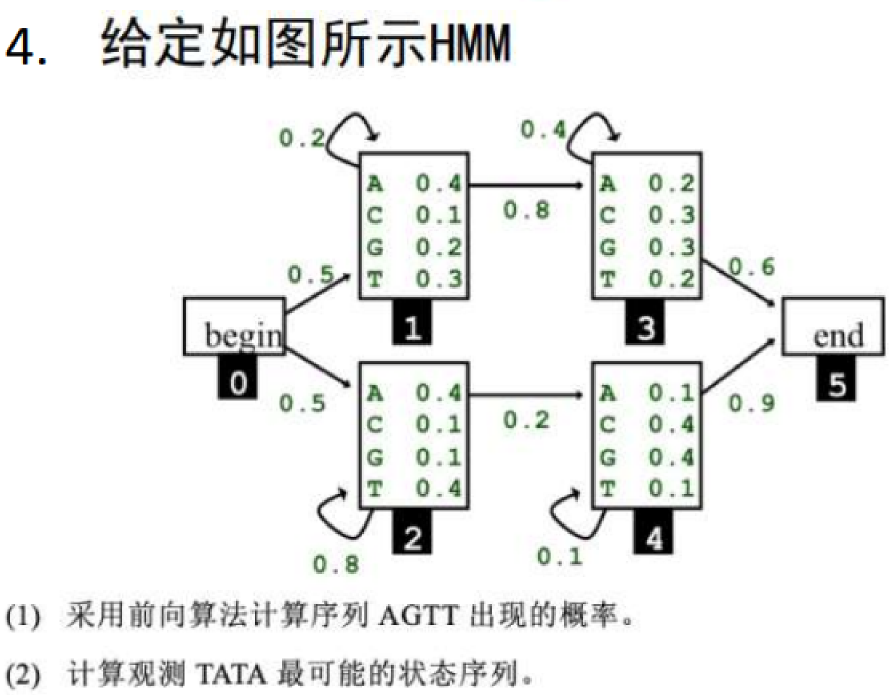

图 4: HMM

解：

由题，HMM 中的状态空间为 $[0,1,2,3,4,5]$，状态数 $M=6$；观测空间为 $[begin,A,C,G,T,end]$，观测数 $N=6$。

由图有状态转移矩阵 A、观测概率矩阵 B 和初始状态矩阵 π：

$$
A=\begin{array}{c} \\ 0\\1\\2\\3\\4\\5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}\\
\begin{pmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0\\
0 & 0.2 & 0 & 0.8 & 0 & 0\\
0 & 0 & 0.8 & 0 & 0.2 & 0\\
0 & 0 & 0 & 0.4 & 0 & 0.6\\
0 & 0 & 0 & 0 & 0.1 & 0.9\\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

$$
B=\begin{array}{c} \\ 0\\1\\2\\3\\4\\5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} begin & A & C & G & T & end \end{array}\\
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0\\
0 & 0.4 & 0.1 & 0.2 & 0.3 & 0\\
0 & 0.4 & 0.1 & 0.1 & 0.4 & 0\\
0 & 0.2 & 0.3 & 0.3 & 0.2 & 0\\
0 & 0.1 & 0.4 & 0.4 & 0.1 & 0\\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\end{array}
$$

$$
\pi=\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}\\
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}
$$

(1) 由题，观测序列为 $[begin, A, G, T, T, end]$，求 $P(begin, A, G, T, T, end|\lambda)$。
前向算法为：
初始化：$\alpha_1(y_1) = P(x_1, y_1|\lambda) = \pi(y_1)b_{y_1,x_1}$
前向概率：$\alpha_t(y_t) = P(x_1, \cdots, x_t, y_t|\lambda)$
递推：$\alpha_{t+1}(y_{t+1}) = \sum_{y_t}(\alpha_t(y_t) \cdot a_{y_t,y_{t+1}}) \cdot b_{y_{t+1},x_{t+1}}$
结束：$P(X|\lambda) = \sum_{y_T} \alpha_T(y_T)$

- 当 $t = 1$ 时，$y_1 = 0$，$x_1 = begin$，此时有：

$$\alpha_1(0) = \pi(0) \cdot b_{0,begin} = 1$$

- 当 $t = 2$ 时，$x_2 = A$，此时有：

$$\alpha_2(1) = \alpha_1(begin) \cdot a_{begin,1} \cdot b_{1,A} = 1 \times 0.5 \times 0.4 = 0.2$$

$$\alpha_2(2) = \alpha_1(begin) \cdot a_{begin,2} \cdot b_{2,A} = 1 \times 0.5 \times 0.4 = 0.2$$

$$\alpha_2(0) = \alpha_2(3) = \alpha_2(4) = \alpha_2(5) = 0$$

- 当 $t = 3$ 时，$x_3 = G$，此时有：

$$\alpha_3(1) = \alpha_2(1) \cdot a_{1,1} \cdot b_{1,G} = 0.2 \times 0.2 \times 0.2 = 0.008$$

$$\alpha_3(2) = \alpha_2(2) \cdot a_{2,2} \cdot b_{2,G} = 0.2 \times 0.8 \times 0.1 = 0.016$$

$$\alpha_3(3) = \alpha_2(1) \cdot a_{1,3} \cdot b_{3,G} = 0.2 \times 0.8 \times 0.3 = 0.048$$

$$\alpha_3(4) = \alpha_2(2) \cdot a_{2,4} \cdot b_{4,G} = 0.2 \times 0.2 \times 0.4 = 0.016$$

$$\alpha_3(0) = \alpha_3(5) = 0$$

- 当 $t = 4$ 时，$x_4 = T$，此时有：

$$\alpha_4(1) = \alpha_3(1) \cdot a_{1,1} \cdot b_{1,T} = 0.008 \times 0.2 \times 0.3 = 48 \times 10^{-5}$$

$$\alpha_4(2) = \alpha_3(2) \cdot a_{2,2} \cdot b_{2,T} = 0.016 \times 0.8 \times 0.4 = 512 \times 10^{-5}$$

$$\alpha_4(3) = (\alpha_3(1) \cdot a_{1,3} + \alpha_3(3) \cdot a_{3,3}) \cdot b_{3,T} = (0.008 \times 0.8 + 0.048 \times 0.4) \times 0.2 = 512 \times 10^{-5}$$

$$\alpha_4(4) = (\alpha_3(2) \cdot a_{2,4} + \alpha_3(4) \cdot a_{4,4}) \cdot b_{4,T} = (0.016 \times 0.2 + 0.016 \times 0.1) \times 0.1 = 48 \times 10^{-5}$$

$$\alpha_4(0) = \alpha_4(5) = 0$$

- 当 $t = 5$ 时，$x_5 = T$，此时有：

$$\alpha_5(1) = \alpha_4(1) \cdot a_{1,1} \cdot b_{1,T} = 48 \times 10^{-5} \times 0.2 \times 0.3 = 288 \times 10^{-7}$$

$$\alpha_5(2) = \alpha_4(2) \cdot a_{2,2} \cdot b_{2,T} = 512 \times 10^{-5} \times 0.8 \times 0.4 = 16384 \times 10^{-7}$$

$$\alpha_5(3) = (\alpha_4(1) \cdot a_{1,3} + \alpha_4(3) \cdot a_{3,3}) \cdot b_{3,T} = (48 \times 10^{-5} \times 0.8 + 512 \times 10^{-5} \times 0.4) \times 0.2 = 4864 \times 10^{-7}$$

$$\alpha_5(4) = (\alpha_4(2) \cdot a_{2,4} + \alpha_4(4) \cdot a_{4,4}) \cdot b_{4,T} = (512 \times 10^{-5} \times 0.2 + 48 \times 10^{-5} \times 0.1) \times 0.1 = 1072 \times 10^{-7}$$

$$\alpha_5(0) = \alpha_5(5) = 0$$

- 当 $t=6$ 时，$x_6=end$，$y_6=5$，此时有：

$$\alpha_6(5)=(\alpha_5(3)\cdot a_{3,5}+\alpha_5(4)\cdot a_{4,5})\cdot b_{5,end}=(4864\times10^{-7}\times0.6+1072\times10^{-7}\times0.9)\times1=38832\times10^{-8}$$

$$\alpha_6(0)=\alpha_6(1)=\alpha_6(2)=\alpha_6(3)=\alpha_6(4)=0$$

因此，$P(begin,A,G,T,T,end|\lambda)=\sum_{y_6}\alpha_6(y_6)=0.00038832$**，即序列** $AGTT$ **出现的概率为 0.00038832**。

（2）观测序列为 $[begin,T,A,T,A,end]$，求 $argmax_{\{y_1,\cdots,y_6\}}P(y_1,\cdots,y_6|begin,T,A,T,A,end,\lambda)$。
Viterbi 算法为：
初始化：$\delta_1(y1)=P(x_1,y_1|\lambda)=\pi(y_1)b_{y_1,x_1}$；$\varphi_1(y_1)=0$
递归：$\delta_{t+1}(i)=max_{\{y_t\}}[\delta_t(y_t)\cdot a_{y_t,i}]\cdot b_{i,x_{t+1}}$；$\varphi_{t+1}(i)=argmax_{\{y_t\}}[\delta_t(y_t)\cdot a_{y_t,i}]$
终止：$P=max_{\{y_T\}}\delta_T(y_T)$; $i_T=argmax_{\{y_T\}}\delta_T(y_T)$
最优路径回溯：对 $t=T-1,\cdots,1$,$i_t=\varphi_{t+1}(i_{t+1})$

- 当 $t=1$ 时，$x_1=begin$,$y_1=0$，此时有：

$$\delta_1(0)=\pi(0)b_{0,begin}$$

$$\varphi_1(0)=0$$

$$\delta_2(1)=\delta_2(2)=\delta_2(3)=\delta_2(4)=\delta_2(5)=0$$

- 当 $t=2$ 时，$x_2=T$，此时有：

$$\delta_2(1)=\delta_1(0)\cdot a_{0,1}\cdot b_{1,T}=1\times0.5\times0.3=0.15$$

$$\delta_2(2)=\delta_1(0)\cdot a_{0,2}\cdot b_{2,T}=1\times0.5\times0.4=0.2$$

$$\varphi_2(1)=\varphi_2(2)=0$$

$$\delta_2(0)=\delta_2(3)=\delta_2(4)=\delta_2(5)=0$$

- 当 $t=3$ 时，$x_3=A$，此时有：

$$\delta_3(1)=\delta_2(1)\cdot a_{1,1}\cdot b_{1,A}=0.15\times0.2\times0.4=0.012$$

$$\varphi_3(1)=1$$

$$\delta_3(2)=\delta_2(2)\cdot a_{2,2}\cdot b_{2,A}=0.2\times0.8\times0.4=0.064$$

$$\varphi_3(2)=2$$

$$\delta_3(3)=\delta_2(1)\cdot a_{1,3}\cdot b_{3,A}=0.15\times0.8\times0.2=0.024$$

$$\varphi_3(3)=1$$

$$\delta_3(4)=\delta_2(2)\cdot a_{2,4}\cdot b_{4,A}=0.2\times0.2\times0.1=0.004$$

$$\varphi_3(4)=2$$

$$\delta_3(0)=\delta_3(5)=0$$

- 当 $t=4$ 时，$x_4=T$，此时有：

$$\delta_4(1)=\delta_3(1)\cdot a_{1,1}\cdot b_{1,T}=0.012\times 0.2\times 0.3=72\times 10^{-5}$$

$$\varphi_4(1)=1$$

$$\delta_4(2)=\delta_3(2)\cdot a_{2,2}\cdot b_{2,T}=0.064\times 0.8\times 0.4=2048\times 10^{-5}$$

$$\varphi_4(2)=2$$

$$\delta_4(3)=max(\delta_3(1)a_{1,3},\delta_3(3)a_{3,3})\cdot b_{3,T}=max(0.012\times 0.8,0.024\times 0.4)\times 0.2=192\times 10^{-5}$$

$$\varphi_4(3)=1or3$$

$$\delta_4(4)=max(\delta_3(2)a_{2,4},\delta_3(4)a_{4,4})\cdot b_{4,T}=max(0.064\times 0.2,0.004\times 0.1)\times 0.1=128\times 10^{-5}$$

$$\varphi_4(4)=2$$

$$\delta_4(0)=\delta_4(5)=0$$

- 当 $t=5$ 时，$x_5=A$，此时有：

$$\delta_5(1)=\delta_4(1)\cdot a_{1,1}\cdot b_{1,A}=72\times 10^{-5}\times 0.2\times 0.4=576\times 10^{-7}$$

$$\varphi_5(1)=1$$

$$\delta_5(2)=\delta_4(2)\cdot a_{2,2}\cdot b_{2,A}=2048\times 10^{-5}\times 0.8\times 0.4=65536\times 10^{-7}$$

$$\varphi_5(2)=2$$

$$\delta_5(3)=max(\delta_4(1)a_{1,3},\delta_4(3)a_{3,3})\cdot b_{3,A}=max(72\times 10^{-5}\times 0.8,192\times 10^{-5}\times 0.4)\times 0.2=1536\times 10^{-7}$$

$$\varphi_5(3)=3$$

$$\delta_5(4)=max(\delta_4(2)a_{2,4},\delta_4(4)a_{4,4})\cdot b_{4,A}=max(2048\times 10^{-5}\times 0.2,128\times 10^{-5}\times 0.1)\times 0.1=4096\times 10^{-7}$$

$$\varphi_5(4)=2$$

$$\delta_5(0)=\delta_5(5)=0$$

- 当 $t=6$ 时，$x_6=end$，$y_6=5$，此时有：

$$\delta_6(5)=max(\delta_5(3)a_{3,5},\delta_5(4)a_{4,5})\cdot b_{5,end}=max(1536\times 10^{-7}\times 0.6,4096\times 10^{-7}\times 0.9)\times 1=36864\times 10^{-8}$$

$$\varphi_6(5)=4$$

回溯状态序列，可得在观测序列为 $[T,A,T,A]$ 时最可能的状态序列为 $[0,2,2,2,4,5]$，概率为 0.00036864。

附录

对于题 3HMM，编写相应程序验证：

AppendixI：计算观测序列对应概率

```
"""
使用前向算法计算序列AGTT出现的概率
状态: 0 1 2 3 4 5
观察: begin A C G T end
观察序列:[begin,A,G,T,T,end]
"""
import numpy as np
import decimal
# params:
M = 6
state_option = [0,1,2,3,4,5]
N = 6
obs_option = ['begin','A','C','G','T','end']
T = 6
s_begin = 0
s_end = 5
# 状态转移矩阵: M*M
A = np.array([[0,0.5,0.5,0,0,0],[0,0.2,0,0.8,0,0],[0,0,0.8,0,0.2,0],
              [0,0,0,0.4,0,0.6],[0,0,0,0,0.1,0.9],[0,0,0,0,0,0]])
# 发射矩阵: M*N
B = np.array([[1,0,0,0,0,0],[0,0.4,0.1,0.2,0.3,0],[0,0.4,0.1,0.1,0.4,0],
              [0,0.2,0.3,0.3,0.2,0],[0,0.1,0.4,0.4,0.1,0],[0,0,0,0,0,1]])
# 观察序列
obs_list = ['begin','A','G','T','T','end']

# 存储前向概率
for_pro = np.zeros((M,T),dtype=np.float)

# 前向算法
def forward():
    # 按时间走
    for t in range(T):
        # 初始值
        if t == 0:
            state_idx= state_option.index(s_begin)
            for_pro[state_idx,t] = 1
        else:
            # 目前观测值
            obs_idx = obs_option.index(obs_list[t])
            # t+1层的循环
            for s1 in range(M):
```

```
                    # t+2层的循环
                    for s2 in range(M):
                        for_pro[s1,t] = decimal.Decimal(str(for_pro[s1,t]))
                        +decimal.Decimal(str(for_pro[s2,t-1]))*\
                            decimal.Decimal(str(A[s2,s1]))
                    for_pro[s1,t] = decimal.Decimal(str(for_pro[s1,t])) *\
                          decimal.Decimal(str(B[s1,obs_idx]))
    # 最后输出序列概率
    pro = np.sum(for_pro[:,T-1],axis=0)
    return pro,for_pro

if __name__ == '__main__':
    pro,for_pro = forward()
    print(for_pro)
    print(pro)
```

输出结果:

```
前向概率计算结果:
[[1.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00]
 [0.0000e+00 2.0000e-01 8.0000e-03 4.8000e-04 2.8800e-05 0.0000e+00]
 [0.0000e+00 2.0000e-01 1.6000e-02 5.1200e-03 1.6384e-03 0.0000e+00]
 [0.0000e+00 0.0000e+00 4.8000e-02 5.1200e-03 4.8640e-04 0.0000e+00]
 [0.0000e+00 0.0000e+00 1.6000e-02 4.8000e-04 1.0720e-04 0.0000e+00]
 [0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 3.8832e-04]]
观测序列AGTT概率: 0.00038832
```

AppendixII：计算观测序列最可能的状态序列

```
"""
计算观测到TATA最可能的状态序列
状态: 0 1 2 3 4 5
观察: begin A C G T end
观察序列:[begin,T,A,T,A,end]
"""
import numpy as np
import decimal
# params:
M = 6
state_option = [0,1,2,3,4,5]
N = 6
obs_option = ['begin','A','C','G','T','end']
T = 6
s_begin = 0
s_end = 5
# 状态转移矩阵: M*M
A = np.array([[0,0.5,0.5,0,0,0],[0,0.2,0,0.8,0,0],[0,0,0.8,0,0.2,0],
```

```
                    [0,0,0,0.4,0,0.6],[0,0,0,0,0.1,0.9],[0,0,0,0,0,0]])
# 发射矩阵：M*N
B = np.array([[1,0,0,0,0,0],[0,0.4,0.1,0.2,0.3,0],[0,0.4,0.1,0.1,0.4,0],
                    [0,0.2,0.3,0.3,0.2,0],[0,0.1,0.4,0.4,0.1,0],[0,0,0,0,0,1]])
# 观察序列
obs_list = ['begin','T','A','T','A','end']

# 存储概率值
pro_arr = np.zeros((M,T),dtype=np.float)
# 存储路径
path_arr = np.zeros((M,T),dtype=int) # 初始值为-1

# Viterbi算法
def viterbi():
    # 按时间走
    for t in range(T):
        # 目前观测值
        obs_idx = obs_option.index(obs_list[t])
        # 初始值
        if t == 0:
            state_idx= state_option.index(s_begin)
            pro_arr[state_idx,t] = B[state_idx,obs_idx]*1
        else:
            # t+1层的循环
            for s1 in range(M):
                # t+2层的循环
                pro_value = []
                for s2 in range(M):
                    pro_value.append(decimal.Decimal(str(pro_arr[s2,t-1]))
                                     *decimal.Decimal(str(A[s2,s1])))
                pro_arr[s1,t] = max(pro_value)*\
                    decimal.Decimal(str(B[s1,obs_idx]))
                # 路径记录，概率不等于0才记录
                if pro_arr[s1,t]-0>1e-7:
                    max_idx = np.argmax(pro_value)
                    path_arr[s1,t] = max_idx

    # 最大可能状态对应的概率
    pro_max = np.max(pro_arr[s_end,T-1])

    # 路径回溯
    path_list = []
    path = s_end
    for t in range(T-1,-1,-1):
        path_list.append(path)
        path = path_arr[path,t]

```

```
    path_list = sorted(path_list,reverse=True)


    return pro_max,path_list,pro_arr,path_arr

if __name__ == '__main__':
    pro_max,path_list,pro_arr,path_arr = viterbi()
    print(pro_arr)
    print(path_list)
    print(pro_max)
    print(path_arr)
```

```
Viterbi算法概率计算结果：
[[1.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00]
 [0.0000e+00 1.5000e-01 1.2000e-02 7.2000e-04 5.7600e-05 0.0000e+00]
 [0.0000e+00 2.0000e-01 6.4000e-02 2.0480e-02 6.5536e-03 0.0000e+00]
 [0.0000e+00 0.0000e+00 2.4000e-02 1.9200e-03 1.5360e-04 0.0000e+00]
 [0.0000e+00 0.0000e+00 4.0000e-03 1.2800e-03 4.0960e-04 0.0000e+00]
 [0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 3.6864e-04]]
TATA最可能的状态序列： [5, 4, 2, 2, 2, 0]
TATA对应的状态序列的概率： 0.00036864
Viterbi算法路径计算结果：
[[0 0 0 0 0 0]
 [0 0 1 1 1 0]
 [0 0 2 2 2 0]
 [0 0 1 1 3 0]
 [0 0 2 2 2 0]
 [0 0 0 0 0 4]]
```

AppendixIII：使用 hmmlearn 库验证

```
import numpy as np
from hmmlearn import hmm

# params:
M = 6
state_option = [0,1,2,3,4,5]
N = 6
obs_option = ['begin', 'A', 'C', 'G', 'T', 'end']
T = 6
s_begin = 0
s_end = 5
# 状态转移矩阵：M*M
A = np.array([[0,0.5,0.5,0,0,0],[0,0.2,0,0.8,0,0],[0,0,0.8,0,0.2,0],
              [0,0,0,0.4,0,0.6],[0,0,0,0,0.1,0.9],[0,0,0,0,0,1]])
# 发射矩阵：M*N
```

```
B = np.array([[1,0,0,0,0,0],[0,0.4,0.1,0.2,0.3,0],[0,0.4,0.1,0.1,0.4,0],
              [0,0.2,0.3,0.3,0.2,0],[0,0.1,0.4,0.4,0.1,0],[0,0,0,0,0,1]])


model = hmm.CategoricalHMM(n_components = M)
model.startprob_ = np.array([1,0,0,0,0,0])
model.transmat_ = A
model.emissionprob_ = B

# 计算观测序列的概率：AGTT
pro1 = model.predict(np.array([[0,1,3,4,4,5]]).T)
print(np.power(np.e, model.score(np.array([[0,1,3,4,4,5]]).T)))

# 计算观测序列TATA的状态：
prob2,state2 = model.decode(np.array([[0,4,1,4,1,5]]).T)
print(state2)
print(np.power(np.e,prob2))
```

输出结果:

```
AGTT对应的概率： 0.00038832000000000027
TATA对应的状态： [0 2 2 2 4 5]
TATA对应的状态概率： 0.00036864000000000005
```