HIDDEN MARKOV MODEL

SPEECH RECOGNITION

UNIVERSITY OF SURREY

VAISHALI SHIVAKUMAR

DEPARTMENT OF COMPUTER SCIENCE AND ELECTRONICS ENGINEERING

UNIVERSITY OF SURREY

GUILDFORD

Abstract

As a part of speech processing and recognition, a discrete Hidden Markov Model (HMM) Is implemented. Initially, the model has 'V' visible states which internally depend on 'N' hidden states. This model involves three scenarios evaluation, decoding and training. In the evaluation stage, the likelihood of the occurrence of a given sequence of visible states is computed. The decoding state results are already provided for verification purposes. Thus, only the evaluation and training are implemented using the Baum Welch algorithm and the parameters are fine-tuned for the best possible sequence of the visible states.

Table of Contents

HIDDEN MARKOV MODEL

1. Introduction

The discrete Hidden Markov Model is an Unsupervised Machine Learning Algorithm. The discrete hidden Markov model is a Markov chain used to determine the sequence of observations which internally depends on a sequence of hidden values. The Markov assumption is that the current output depends only on the previous output and not on the series of previous outputs. In a discrete Hidden Markov Model, the state of a system is hidden (invisible), however, each state is responsible for an observation given in the observation sequence given over a time frame. HMM works with both discrete and continuous data values. The HMM mainly find usage in Speech recognition, handwriting recognition, gesture recognition, bioinformatics, parts-of-speech tagging etc.

2. Markov Model

2.1. Intuition

In a basic Markov Model, there are a set of N states which are visible. The intuition of this model is that a future state of the system depends only on the previous state of the system, meaning the next state depends only on the current state and not on any other previous states. This is known as Markov Property.

The probability of the current state is s(t) and the probability of a state at a time say (t+1) can be given as P(s(t+1) | s(t)). This is the first-order Markov Model. In case, the probability of the state ‘s’ at time ‘t’ depends on time steps t-1 and t-2, it’s known as the 2nd Order Markov Model. Thus, in the second case the probability of the current state depends on the previous s= two states so, it is the second order Markov Model. Thus, as the dependability increases the order of the model also increases. As you increase the dependency of past-time events the order increases.

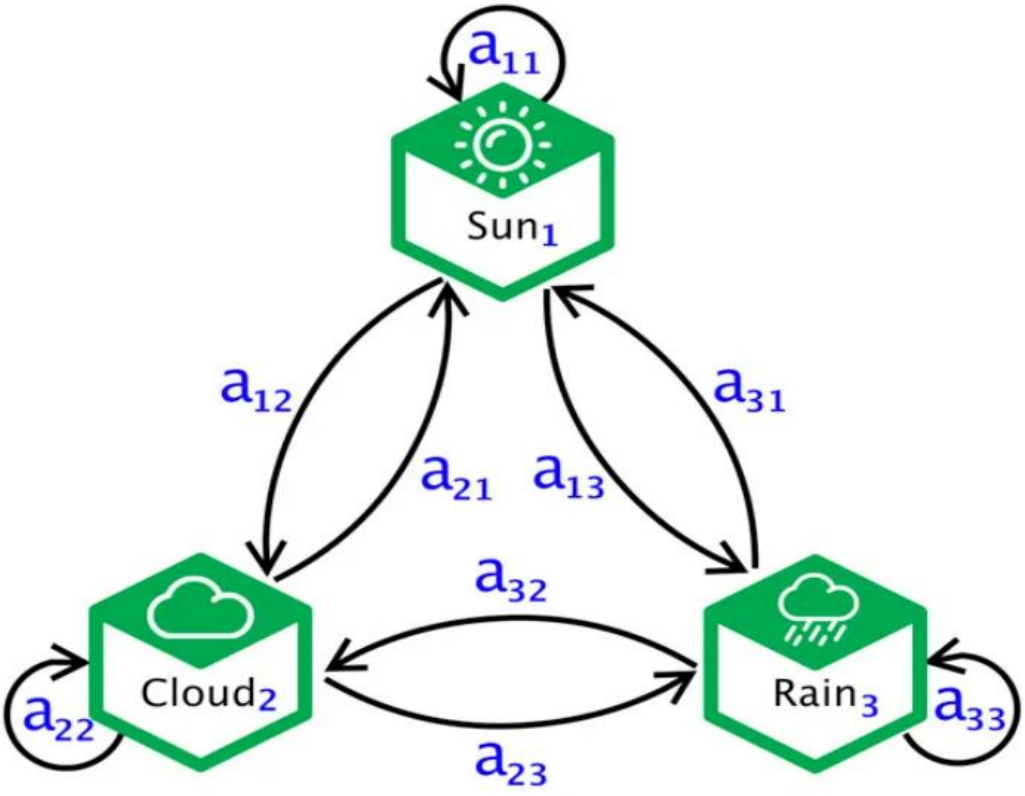

Figure 1 HMM Model Source.[1]

Suppose there is a set sequences in a set of state, the probability of occurrence of the given sequence can be determined a as joint probability. Consider a set of states is given as $S = \{S_1, S_2, S_3\}$ the probability of getting the sequence can be given as,

$$P(S) = P(S_1) * P(S_2|S_1) * P(S_3|S_2) \quad \text{_____ (1)}$$

3. Structure of HMM

A discrete Hidden Markov Model (θ) can be defined with the following parameters:

- Set of N Hidden states (S^N), $S = \{S_1, S_2, S_3,..S_N\}$

- A Transition Probability Matrix (A), A $\{a_{ij}\}$ = $\begin{bmatrix} 00 & 01 & & 0N \\ 10 & 11 & & 1N \\ & & & \\ N0 & N1 & & NN \end{bmatrix}$
- A sequence of T observations (V^T), V = $\{V_1, V_2, V_3, \ldots V_T\}$
- An Emission Probability Matrix (B), B $\{b_{ij}\}$ = $\begin{bmatrix} 00 & 01 & & 0N \\ 10 & 11 & & 1N \\ & & & \\ N0 & N1 & & NN \end{bmatrix}$
- Initial Probability Distribution (π),

Thus, Θ = {S, A, V, B, π}.

3.1. Set of Hidden States and Visible states

The hidden states are those states which determine the observable visible states. The system changes its state from one state to another in each time frame. The probability of a state plays a key role in the determination of the current state of the system. The whole system can be represented by a state topology diagram as given in Figure 1.

The Visible states are those which are the effects of the transition between the hidden states. Every state has a certain probability to generate a visible state which is given in Emission Probability Matrix. The relation between the Hidden states, Visible states is depicted in Figure 1. The set of hidden states is denoted by S which is given in equation (2) and set of visible states which as given in equation (3).

(S^N), S = $\{S_1, S_2, S_3, \ldots S_N\}$ _____ (2)

(V^T), V = $\{ V_1, V_2, V_3, \ldots V_T \}$ _____ (3)

3.2. Transition Probability Matrix

The probability of change from one state to another in each time frame is termed Transition Probability. Ex: if there are 3 states then there are 9 possible transitions in the model. The transition probability is denoted by a_{ij} which can be defined as the transition from state i to state j at time t+1.

The transition probability can be defined in terms of transition matrix A with dimensions (NxN) where 'N' is the number of states which is given as per equation (4). The state topology diagram given in figure 1 depicts the transition probability for a given model with 3 states. The transition can be within the same state also. In the figure the transition from State s_1 to s_1 is given as a_{11}, similarly, s_2 to s_2 is given as a_{22}, and the transition from state s_2 to s_3 is given as a_{23}. When the system transitions to another state, the sum of all transition probabilities given the current state should be 1.

$$A\{a_{ij}\} = \begin{bmatrix} 00 & 01 & & 0N \\ 10 & 11 & & 1N \\ & & & \\ N0 & N1 & & NN \end{bmatrix} \quad \text{(4)}$$

$$\sum_{j=0}^{N} aij = 1 \; \forall \; i \quad \text{(5)}$$

3.3. Initial Probability Distribution

The system must start from time t=0, which at that time has a state which is the initial state represented by an N-dimensional matrix which is given in equation (6). If π = 0, then the system has no initial state.

$$\Pi = [0 \quad 0 \quad N] \quad \text{(6)}$$

$$\sum_{i=0}^{N} \pi i = 1, \forall \; i \quad \text{(7)}$$

3.4. Final/ Absorbing State

When the transition probabilities of any step to other steps are zero except for itself, then it is the Final/Absorbing State. when the system enters the Final/Absorbing State, it never leaves.

3.5. Emission Probability

The set of observations given in as per equation (3) must be emitted from each of the given hidden states. The probability of emitting an observation from a hidden state is called Emission Probability, which is denoted as b_{jk}. The probability of emitting observation k from state j is given as,

$$b_{jk} = P(v_k(t) \mid s_j(t)) \quad \text{(8)}$$

The emission probability is given by the Emission Probability Matrix of dimensions (NxT) where N is the number of hidden states and T is the number of Visible states. Again the probability of emission for each of the hidden states to all of the observations must sum up to 1. The complete hidden Markov model is depicted in figure 1.

$$(B),\ B\{b_{ij}\} = \begin{bmatrix} 00 & 01 & & 0N \\ 10 & 11 & & 1N \\ & & & \\ N0 & N1 & & NN \end{bmatrix} \quad \text{(9)}$$

$$\sum_{k=0}^{T} bjk = 1, \forall\, j \quad \text{(10)}$$

4. Key Issues with Hidden Markov Model

The central issues with a Hidden Markov Model are:

a) Evaluation Problem: The evaluation problem is about finding the probability of occurrence of a sequence ‘V’ in an HMM model Θ. For this purpose, the forward-backwards Algorithm is used.

b) Decoding Problem: The decoding problem is about finding the best likely sequence of hidden states for the given observation sequence. This is implemented with the Viterbi Algorithm.

c) Learning Problem: The learning problem where the model is fine-tuned for the best set of transition probability and emission probability matrix for the given dataset. This is implemented with Baum-Welch algorithm.

5. Model Prototype and Observation Data

The task is to compute a series of likelihood calculations to perform one iteration of training on the parameters of a discrete hidden Markov model (HMM) with N=3 emitting states. Initial values of the

state-transition probabilities A and the output probabilities B are given in Tables 1 and 2. There are five observation sequences O^1, **O^2, O^3, O^4 and O^5** which are given below.

0	**0.41**	**0.23**	**0.36**	**0**
0	0.77	0.07	0.06	0.10
0	0.14	0.71	0.04	0.11
0	0.16	0.11	0.74	0.09
0	0	0	0	0

Table 1 State Transition Probability Matrix A = { π_i, a_{ij}, η_i } including entry and exit transitions

State/ Observations	1	2	3	4	5	6	7	8	9	10
1	0.30	0.21	0.24	0.10	0.06	0.04	0.01	0.03	0.00	0.01
2	0.00	0.00	0.03	0.04	0.05	0.08	0.14	0.20	0.25	0.21
3	0.00	0.05	0.05	0.23	0.32	0.24	0.03	0.04	0.02	0.02

Table 2 Output Probability Matrix B, with values for each state I and observation type k

O^1 = { 5, 4, 2, 1, 3 } with a duration of 5 frames.

O^2 = { 6, 4, 5, 7 } with a duration of 4 frames

O^3 = { 7, 9, 6, 8, 7, 3, 6, 3 } with a duration of 8 frames

O^4 = { 5, 7, 5, 5, 10, 7, 6 } with a duration of 7 frames

O^5 = { 3, 2, 2, 1, 2, 1, 8, 9, 7, 3 } with a duration of 10 frames

6. Method

For the given observation sequences first the forward and backward likelihood values are computed. Then the occupation likelihood matrix and the transition likelihood matrices are computed for each of the observation sequences. Later the optimum values for transition probability and emission probability are computed with Baum-Welch equations. The results obtained are then compared against the results of Viterbi and Naïve Bayes algorithm results.

7. Results

7.1. HMM's state topology and Trellis diagram for the first Observation Sequence O1.

For the given observation O^1 = {5, 4, 2, 1, 3} the probability for the sequence is given in table 1. The HMM state topology diagram for the sequence is given in Figure 2. The trellis intuition diagram for the forward procedure for the given sequence is given in the figure 3.

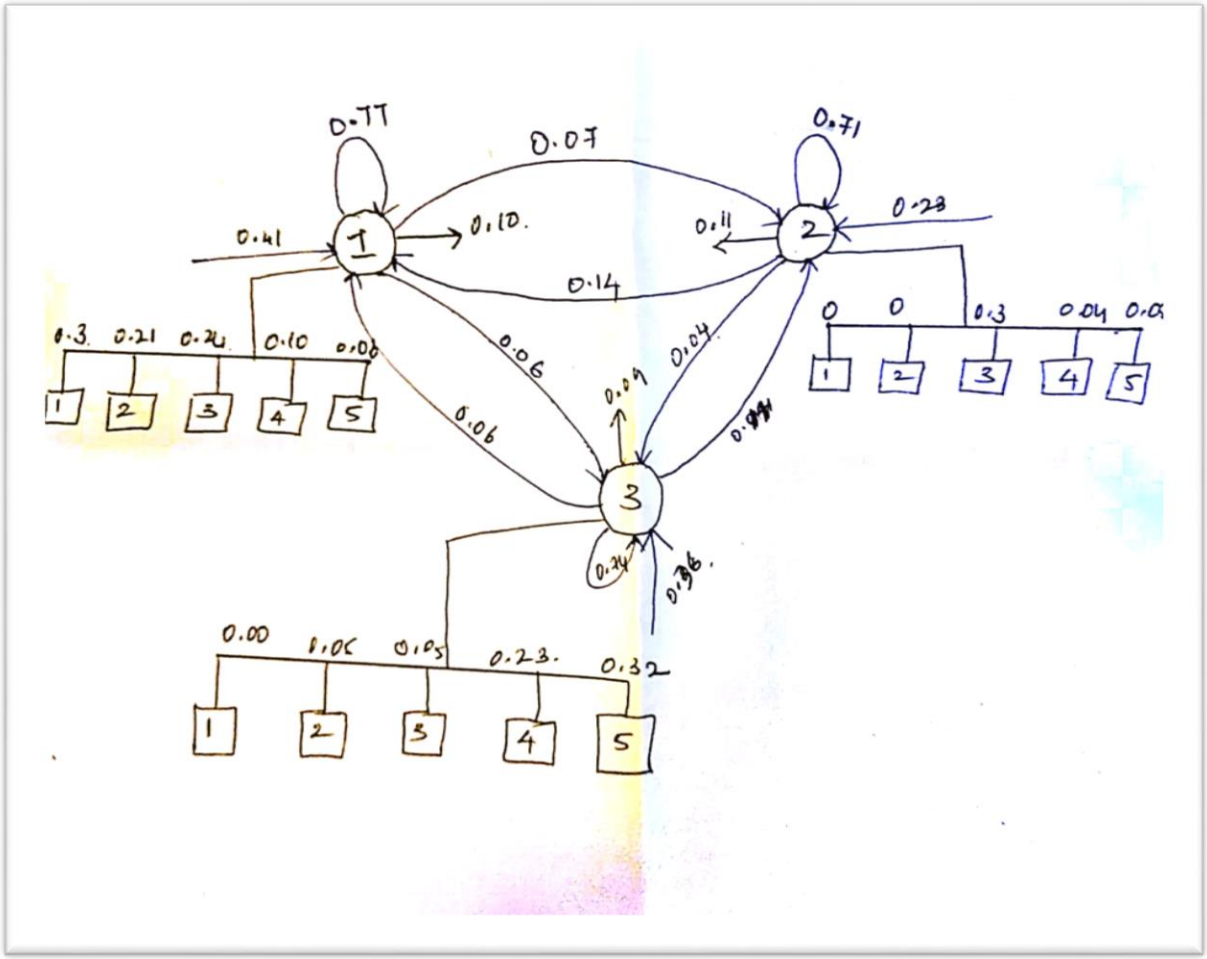

Figure 2 State Topology Diagram

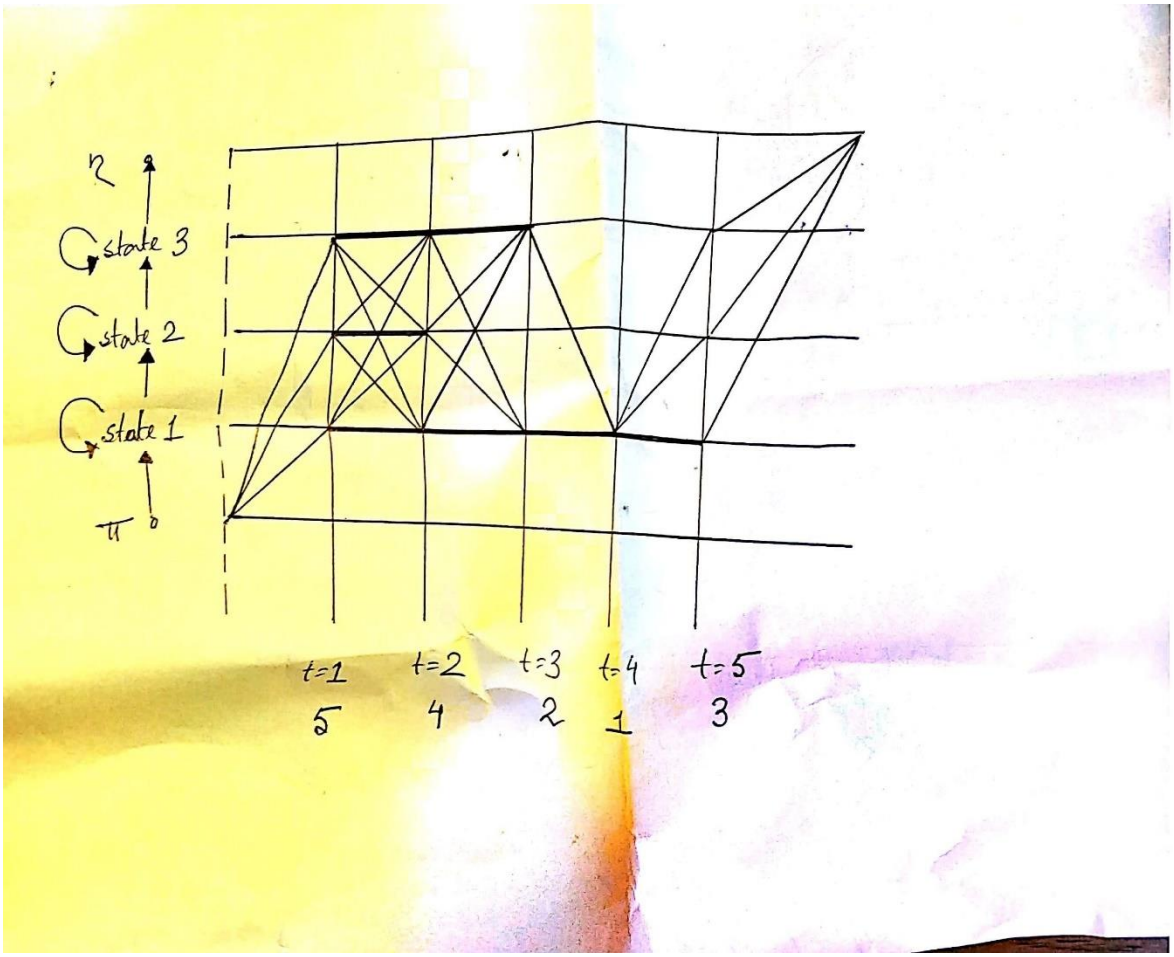

Figure 3 Trellis Diagram

7.2. Forward Procedure Computed for O^1

The forward likelihoods for the given observation sequence O^1 = {5, 4, 2, 1, 3} is given in table 2.

```
Forward Likelihood =  3.4282279043001356e-06
```

Out[79]:

Forward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5
1	0.024600	0.002746	0.000723	0.000181	0.000033	0.000003
2	0.011500	0.000902	0.000000	0.000000	0.000000	0.000000
3	0.115200	0.020052	0.000752	0.000000	0.000001	0.000000

Table 2 Forward likelihood Calculations for Observation Sequence O^1

The forward likelihoods for other observation sequences O^2, O^3, O^4 and O^5 are given in Appendix.

7.3. Backward Procedure Computed for O^1

The backward likelihoods for the given observation sequence O^1 = {5, 4, 2, 1, 3} are given in table 3.

```
Backward Likelihood =  2.2105224172080005e-06
```

Out[81]:

Backward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5
1	0.000001	0.000017	0.000219	0.001352	0.005000	0.100000
2	0.000000	0.000001	0.000000	0.000000	0.005500	0.110000
3	0.000001	0.000004	0.000004	0.000000	0.004500	0.090000

Table 3 Backward likelihood Calculations for Observation Sequence O^1

The backward likelihoods for other observation sequences O^2, O^3, O^4 and O^5 are given in Appendix.

7.4. Occupation Likelihoods for O1

The occupation likelihoods for the given observation sequence O^1 = {5, 4, 2, 1, 3} is given

Out[83]:

Occupation Likelihood

	t=0	t=1	t=2	t=3	t=4
1	0.120923	0.175073	0.285137	0.263421	0.973605
2	0.004127	0.000000	0.000000	0.000000	0.012170
3	0.124531	0.023715	0.000000	0.000000	0.014225

Table 4 Occupation likelihood for Observation sequence O^1

The backward likelihoods for other observation sequences O^2, O^3, O^4 and O^5 are given in Appendix.

8. Discussion

8.1. Comparison of Occupation likelihoods against Naïve Bayes

The occupation likelihood values computed for same set of data from Naïve Bayes and Viterbi algorithm is given in Table 5 and Table 6. The maximum likelihood values of the dataset which are obtained with reference to Table 4 are given in Table 7.

Observations| t	1	2	3	4	5	6	7	8	9	10
O^1	3	3	1	1	1					

Table 5 Naïve Bayes classification results for sequence O^1

Observations| t	1	2	3	4	5	6	7	8	9	10
O^1	1	1	1	1	1					

Table 6 Viterbi path alignments for observation sequence O^1

Observations| t	1	2	3	4	5	6	7	8	9	10
O^1	3	1	1	1	1					

Table 7 Maximum likelihood as per Table 4

Thus, form tables 5, 6 and 7 it can been seen that the HMM Prototype model implemented in this paper closely matches with that of NaïveBayes Classifier except for error in the time frame t=2.

8.2. Baum Welch Implementation

```
In [278]: res["a"]
```

Out[278]:

A^

	0	1	2
0	0.934564	0.002488	0.062948
1	0.898705	0.050914	0.050381
2	0.452623	0.004518	0.542860

```
In [279]: res["b"]
```

Out[279]:

B^

	0	1	2	3	4	5	6	7	8	9
0	0.275564	0.254913	0.228156	0.166051	0.075316	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.088196	0.506622	0.405182	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.057650	0.127519	0.277995	0.536836	0.000000	0.000000	0.000000	0.000000	0.000000

9. Conclusions

An investigation on hidden Markov models has been demostrated. Initially the forward and backward procedures were demostrated and found to produce almost near observation likelihood. The occupation likelihoods were calculated and used to provide suggestions for best possible sequence. Finally the model was demonstrated with single iteration of Baum-Welch training to produce new output parameters and were comapred to the original system parameters.

References

[1] http://www.adeveloperdiary.com/data-science/machine-learning/introduction-to-hidden-markov-model/

[2] http://www.adeveloperdiary.com/data-science/machine-learning/forward-and-backward-algorithm-in-hidden-markov-model/

[3] http://www.adeveloperdiary.com/data-science/machine-learning/derivation-and-implementation-of-baum-welch-algorithm-for-hidden-markov-model/

[4] http://www.adeveloperdiary.com/data-science/machine-learning/implement-viterbi-algorithm-in-hidden-markov-model-using-python-and-r/

[5] Rabiner L.R., tutorial on HMM and selected applications in speech recognition, In Proc. IEEE, Vol. 77, No. 2, pp. 257-286, Feb. 1989.

[6] Lecture slides EEEM030: speech & audio processing & recognition, Prof Philip Jackson, Centre for Vision, Speech & Signal Processing (CVSSP) Department of Electrical & Electronic Engineering University of Surrey, slides – eeeM030E, eeeM030H, eeeM030K, eeeM030N

[7] Images and Table data from source code file assignment_02_speech.ipynb

APPENDIX

A. Source Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np


# In[2]:


#V = np.array([2,1,1])
#V = np.array([4, 3, 1, 0, 2])
#V = np.array([6, 4, 5, 7])
#V = np.array([7, 9, 6, 8, 7, 3, 6, 3])
#V = np.array([5, 7, 5, 5, 10, 7, 6])
V = np.array([3, 2, 2, 1, 2, 1, 8, 9, 7, 3])
V


# In[4]:


# Transition Probabilitiy Matrix A
#a = np.array(((0.8, 0.2,0), (0, 0.6,0.4)))
a = np.array(((0.77, 0.07, 0.06, 0.10), (0.14, 0.71, 0.04, 0.11), (0.06, 0.11, 0.74, 0.09)))

# Emission Probabilitiy Matrix B
b = np.array(((0.3, 0.21, 0.24, 0.10, 0.06, 0.04, 0.01, 0.03, 0.00, 0.01), (0.00, 0.00, 0.03, 0.04, 0.05,
0.08, 0.14, 0.20, 0.25, 0.21), (0.00, 0.05, 0.05, 0.23, 0.32, 0.24, 0.03, 0.04, 0.02, 0.02)))
#b = np.array(((0.5, 0.2, 0.3), (0,0.9,0.1)))

# Equal Probabilities for the initial distribution
#initial_distribution = np.array((1,0))
initial_distribution = np.array((0.41, 0.23, 0.36))


# ## Forward Procedure

# In[5]:


def forward(V, a, b, initial_distribution):
    alpha = np.zeros((V.shape[0]+1, a.shape[0]))
    alpha[0, :] = initial_distribution * b[:, V[0]]
```

```
    for t in range(1, V.shape[0]+1): # I.e time distribution is 3 since there are only 3 visible states so the
loop will run 3 times
        for j in range(a.shape[0]): # i.e this is for 2 pass as there are only two states.
            if (t < V.shape[0]):
                alpha[t, j] = alpha[t - 1].dot(a[:, j]) * b[j, V[t]]
            else:
                alpha[t, j] = alpha[t - 1][j] * a[j][-1]
    return alpha


# In[6]:


alpha = forward(V, a, b, initial_distribution)
alpha_likelihood = np.sum(alpha[-1,:])
alpha = alpha.transpose()
print("Forward Likelihood = ", alpha_likelihood)
df_fwd = pd. DataFrame(alpha, columns=['t={}'.format(x) for x in range(alpha.shape[1])])
index = pd.Index([x for x in range(1,alpha.shape[0]+1)])
df_fwd = df_fwd.set_index(index)
df_fwd = df_fwd.style.set_caption("Forward Procedure Likelihood")
df_fwd


# ## Backward procedure

# In[7]:


def backward(V, a, b):
    T=V.shape[0]
    N = b.shape[0]
    beta = np.ones((V.shape[0]+1, b.shape[0]))
    beta[-1,:]=a[:,-1]
    for t in range(T-1,-1,-1):
        for i in range(N):
            s = 0
            for j in range(N):
                if(t==T-1):
                    beta[t][i] = a[i,-1]*b[j][V[t]]
                else:
                    s = s + (a[i][j]*b[i][V[t]]*beta[t+1][j])
                    beta[t][i] = s
    return beta


# In[8]:


beta = backward(V, a, b)
beta_likelihood = np.sum(beta[0,:])
beta = beta.transpose()
```

```
print("Backward Likelihood = ", beta_likelihood)
df_bck = pd.DataFrame(beta, columns=['t={}'.format(x) for x in range(beta.shape[1])])
index = pd.Index([x for x in range(1,alpha.shape[0]+1)])
df_bck = df_bck.set_index(index)
df_bck = df_bck.style.set_caption("Backward Procedure Likelihood")
df_bck


# ## Occupation Likelihood

# In[9]:


def occupation(alpha, beta, alpha_likelihood):
    t = alpha.shape[1]
    i = alpha.shape[0]

    gamma = np.zeros([i,t])

    for x in range(i): #0,1
        for y in range(t): #0,1,2
            gamma[x][y] = (alpha[x][y] * beta[x][y])/alpha_likelihood
    return gamma


# In[10]:


gamma = occupation(alpha[:, :-1 ], beta[:,1:], alpha_likelihood)
df_gamma = pd.DataFrame(gamma, columns=['t={}'.format(x) for x in range(gamma.shape[1])])
index = pd.Index([x for x in range(1,gamma.shape[0]+1)])
df_gamma = df_gamma.set_index(index)
df_gamma = df_gamma.style.set_caption("Occupation Likelihood")
df_gamma


# ## Transition Likelihood

# In[11]:


def transition(b, beta, alpha, a, alpha_likelihood):
    T =  3
    N = 2

    print(a)
    print(b)

    Estimated = []

    for p in range(4):
        E = np.zeros([3,3])
```

```
        for t in range(0,T-1):
            for i in range(N):
                for j in range(N):
                    if(t==0):
#                         print(a[i][j])
#                         print(b[j][V[t]])
#                         print(beta[t][j])
                        E[0][j] = (a[i][j]*b[j][V[t]]*beta[t][j])/alpha_likelihood
                    elif (t == T-1):
                        E[i][N] = (alpha[t][i]*a[i][N])/alpha_likelihood
                    else:
                        E[i][j] = (alpha[t-1][i]*a[i][j]*b[j][V[t]]*beta[t][j])/alpha_likelihood
    Estimated.append(E)
    return Estimated


# In[12]:


transition(b, beta, alpha, a , alpha_likelihood)


# ## Baum Welch Algorithm

# In[232]:


def baum_welch(V, a, b, initial_distribution, alpha, beta, n_iter=5):
    M = a.shape[0]
    T = len(V)
    print(T)

    for n in range(n_iter):
        #alpha = forward(V, a, b, initial_distribution)
        #beta = backward(V, a, b)

        xi = np.zeros((M, M, T - 1))
        for t in range(T - 1):
#             print("M: ", M)
#             print("T: ", T)
#             print("alpha: ", alpha)
#             print("beta: ", beta)
#             print("t: ", t)
#             print(alpha[t, :])
#             print(alpha[t, :].T)
#             print("a: ", a)
#             print(np.dot(alpha[t, :].T, a[:,:-1]))
#             print(b[:, V[t + 1]].T)
#             print(beta[t + 1, :])
            denominator = np.dot(np.dot(alpha[t, :].T, a[:,:-1]) * b[:, V[t + 1]].T, beta[t + 1, :])
            print(denominator)
            for i in range(M):
```

```
                numerator = alpha[t, i] * a[i, :-1] * b[:, V[t + 1]].T * beta[t + 1, :].T
                print(numerator)
                xi[i, :, t] = numerator / denominator

        print("xi: ",xi)
        gamma = np.sum(xi, axis=1)
        print("gamma: ", gamma)

        a = np.sum(xi, 2) / np.sum(gamma, axis=1).reshape((-1, 1))

        # Add additional T'th element in gamma
        gamma = np.hstack((gamma, np.sum(xi[:, :, T - 2], axis=0).reshape((-1, 1))))

        K = b.shape[1]
        denominator = np.sum(gamma, axis=1)
        for l in range(K):
            b[:, l] = np.sum(gamma[:, V == l], axis=1)

        b = np.divide(b, denominator.reshape((-1, 1)))

    return {"a":a, "b":b}


# In[233]:


a, b = baum_welch(V, a, b, initial_distribution, alpha, beta, n_iter=1)


# ## Viterbi Algorithm

# In[269]:


def viterbi(V, a, b, initial_distribution):
    T = V.shape[0]
    M = a.shape[0]

    omega = np.zeros((T, M))
    omega[0, :] = np.log(initial_distribution * b[:, V[0]])

    prev = np.zeros((T - 1, M))

    for t in range(1, T):
        for j in range(M):
            # Same as Forward Probability
            probability = omega[t - 1] + np.log(a[:, j]) + np.log(b[j, V[t]])

            # This is our most probable state given previous state at time t (1)
            prev[t - 1, j] = np.argmax(probability)

            # This is the probability of the most probable state (2)
```

```
            omega[t, j] = np.max(probability)

    # Path Array
    S = np.zeros(T)

    # Find the most probable last hidden state
    last_state = np.argmax(omega[T - 1, :])

    S[0] = last_state

    backtrack_index = 1
    for i in range(T - 2, -1, -1):
        S[backtrack_index] = prev[i, int(last_state)]
        last_state = prev[i, int(last_state)]
        backtrack_index += 1

    # Flip the path array since we were backtracking
    S = np.flip(S, axis=0)

    # Convert numeric values to actual hidden states
    result = []
    for s in S:
        if s == 0:
            result.append("A")
        else:
            result.append("B")

    return result


# In[ ]:


print(viterbi(V, a, b, initial_distribution))
```

B. Observation Sequence O^2 = { 6, 4, 5, 7 } with a duration of 4 frames

a) Forward Procedure

```
Forward Likelihood =  4.666986296544001e-06
```

Out[89]:

Forward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4
1	0.004100	0.000499	0.000029	0.000002	0.000000
2	0.032200	0.001217	0.000099	0.000027	0.000003
3	0.010800	0.003048	0.000560	0.000017	0.000002

b) Backward Procedure

```
Backward Likelihood =  7.5661132255999995e-06
```

Out[91]:

Backward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4
1	0.000000	0.000011	0.000144	0.004000	0.100000
2	0.000003	0.000014	0.000306	0.004400	0.110000
3	0.000005	0.000206	0.000813	0.003600	0.090000

c) Occupation Likelihood

Out[93]:

Occupation Likelihood

	t=0	t=1	t=2	t=3
1	0.009553	0.015407	0.025278	0.045091
2	0.093191	0.079848	0.093087	0.630699
3	0.476929	0.531103	0.432160	0.324210

C. Observation Sequence O^3 = { 7, 9, 6, 8, 7, 3, 6, 3 } with a duration of 8 frames

a) Forward Procedure

```
Forward Likelihood =  4.479722114949766e-10
```

Out[97]:

Forward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
1	0.012300	0.000168	0.000012	0.000000	0.000001	0.000000	0.000000	0.000000	0.000000
2	0.046000	0.007372	0.000739	0.000132	0.000019	0.000001	0.000000	0.000000	0.000000
3	0.014400	0.000265	0.000015	0.000001	0.000000	0.000000	0.000000	0.000000	0.000000

b) Backward Procedure

```
Backward Likelihood =  6.306447248690681e-09
```

Out[99]:

Backward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
1	0.000000	0.000000	0.000000	0.000000	0.000001	0.000041	0.000207	0.023000	0.100000
2	0.000000	0.000000	0.000000	0.000003	0.000015	0.000090	0.003082	0.025300	0.110000
3	0.000000	0.000000	0.000000	0.000000	0.000006	0.000180	0.000584	0.020700	0.090000

c) Occupation Likelihood

Out[101]:

Occupation Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7
1	0.006048	0.000769	0.000000	0.000000	0.050805	0.141647	0.166227	0.249705
2	4.330831	4.637228	4.662120	4.502914	3.744717	3.674948	3.359835	0.423655
3	0.025732	0.007111	0.004082	0.010765	0.094665	0.286900	0.280682	0.326639

D. Observation Sequence O^5 = { 3, 2, 2, 1, 2, 1, 8, 9, 7, 3 } with a duration of 10 frames

a) Forward Procedure

```
Forward Likelihood =  1.9675794250794863e-11
```

Out[109]:

Forward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
1	0.041000	0.009078	0.001742	0.000285	0.000053	0.000009	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.009200	0.000555	0.000041	0.000000	0.000001	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.082800	0.003205	0.000147	0.000011	0.000001	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

b) Backward Procedure

```
Backward Likelihood =  1.970512007317794e-10
```

Out[111]:

Backward Procedure Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000002	0.000000	0.000008	0.000622	0.023000	0.100000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000121	0.000681	0.004402	0.025300	0.110000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000001	0.000002	0.000022	0.000779	0.020700	0.090000

c) Occupation Likelihood

Out[113]:

Occupation Likelihood

	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9
1	4.033482	4.813625	4.983789	4.990123	4.850989	0.000000	0.000000	0.007138	0.122229	0.280700
2	0.023012	0.006748	0.000000	0.000000	0.000000	0.000000	5.388769	5.261307	4.306882	0.534616
3	0.176717	0.041294	0.017090	0.017830	0.046816	0.019195	0.014883	0.012747	0.050177	0.184684