HMM

Introduction:

Hidden Markov Models are used in various applications of Artificial Intelligence like Part of Speech tagging etc. The Model contains Regular states and Hidden states and the output probability is based on the value of the observed hidden state which is Pro(O|lambda) where O is the observations and lambda defines the value of the state sequence observed that maximizes the probability on the provided hidden state sequence.

Model definition:

The model takes input as:

- **N** : Number of States
- **T** : Transitions from s_i to s_j given w_k
- **A** : State Transition Matrix
- **B** : Matrix containing the probabilities of observing o_k from s_i or Emission Matrix
- **O** : Observables / Number of hidden states
- **Sequence** : Observed state sequence of hidden states
- **Pi** : Initial State probabilities (In our case only one is initial thus it is one hot vector)

The model outputs as:

- **A** : Final Transition Matrix
- **B** : Final Emission Matrix
- **T** : Final Transitions from s_i to s_j given w_k

Algortithms used:

1. The viterbi algorithm is implemented as thought in class using the T values to get a State sequence with maximum probabilities.
2. The Backward/Beta and Forward/Alpha Algorithms and the Training Algorithms are implemented using two approaches.
    a. One used in the lecture using T.
    b. Using A and B Matrices (refer link)
3. POS tagging is done using the above defined algorithms.

Experiments and observations:

- While conducting the experiments on the defined algorithm it was found that the viterbi, Forward, and Backward are all giving correct answers and that when chaing the observed sequence the value of alpha and beta remains same.
- While in case of training the output was found correct in one case while in other the output was different than provided for reference and thus another algorithm using the reference link was derived that uses the A and B matrices for training and by using that the value was found near the correct value.
- Further while doing experiments it was observed that for higher value of observations the model was more accurate and reaches its optimal solution in 30-40 iterations still value was observed till 100 iteration in each case. Whereas for small training the value reached optimal point around 15-20 iterations.

Using Lectrure Algorithm
TEST 1:

```
Enter comma seprated States:
q,r
Enter initial state:
q
Enter comma seprated variables:
a,b
Enter the probability for q -> q for the observation a
0.4
Enter the probability for q -> r for the observation a
0.3
Enter the probability for q -> q for the observation b
0.2
Enter the probability for q -> r for the observation b
0.1
Enter the probability for r -> q for the observation a
0.2
Enter the probability for r -> r for the observation a
0.2
Enter the probability for r -> q for the observation b
0.1
Enter the probability for r -> r for the observation b
0.5
{('q', 'a', 'q'): 0.4, ('q', 'a', 'r'): 0.3, ('q', 'b', 'q'): 0.2,
('q', 'b', 'r'): 0.1, ('r', 'a', 'q'): 0.2, ('r', 'a', 'r'): 0.2, ('r',
'b', 'q'): 0.1, ('r', 'b', 'r'): 0.5}
```

OUTPUT:
For Observed Sequence : ababb

```
Best State Sequence for ababb is qrrrrr
Best State Sequence Probability for ababb is 0.0075

The Alpha vector for time tick are: [[1, 0], [0.4, 0.3],
[0.11000000000000001, 0.19], [0.08200000000000002,
0.07100000000000001], [0.023500000000000007, 0.0437],
[0.009070000000000002, 0.024200000000000003]]
The value of p1,w is: 0.03327000000000001

The Beta Vectors for time Tick are: [[1, 1], [0.30000000000000004,
0.6], [0.12000000000000001, 0.33], [0.14700000000000002,
0.09000000000000001], [0.038400000000000004, 0.05970000000000001],
[0.03327000000000001, 0]]
The probility of p1,w is: 0.03327000000000001
```

TEST 2:

```
Enter comma seprated States:
a,b
Enter initial state:
a
Enter comma seprated variables:
0,1
Enter the probability for a -> a for the observation 0
0.48
Enter the probability for a -> b for the observation 0
0.04
Enter the probability for a -> a for the observation 1
0.48
Enter the probability for a -> b for the observation 1
0
Enter the probability for b -> a for the observation 0
0
Enter the probability for b -> b for the observation 0
0
Enter the probability for b -> a for the observation 1
1.0
Enter the probability for b -> b for the observation 1
0
{('a', '0', 'a'): 0.48, ('a', '0', 'b'): 0.04, ('a', '1', 'a'): 0.48,
('a', '1', 'b'): 0.0, ('b', '0', 'a'): 0.0, ('b', '0', 'b'): 0.0, ('b',
'1', 'a'): 1.0, ('b', '1', 'b'): 0.0}
```

OUTPUT:

For Observed Sequence : 01011

```
Best State Sequence for 01011 is aaaaaa
Best State Sequence Probability for 01011 is 0.0254803968

The Alpha vector for time tick are: [[1, 0], [0.48, 0.04], [0.2704,
0.0], [0.129792, 0.010816], [0.07311615999999999, 0.0],
[0.03509575679999999, 0.0]]
The value of p1,w is: 0.03509575679999999

The Beta Vectors for time Tick are: [[1, 1], [0.48, 1.0], [0.2304,
0.48], [0.129792, 0.0], [0.06230015999999999, 0.129792],
[0.035095756799999996, 0]]
The probility of p1,w is: 0.035095756799999996

Final Values of T after training:
{('a', '0', 'a'): 0.16778500596293738, ('a', '0', 'b'):
0.6644299880741252, ('a', '1', 'a'): 0.16778500596293738, ('a', '1',
'b'): 0.0, ('b', '0', 'a'): 0.0, ('b', '0', 'b'): 0.0, ('b', '1', 'a'):
1.0, ('b', '1', 'b'): 0.0}
```

TEST 3:

```
Enter comma seprated States:
q,r,s
Enter initial state:
q
Enter comma seprated variables:
a,b
Enter the probability for q -> q for the observation a
0
Enter the probability for q -> r for the observation a
0.25
Enter the probability for q -> s for the observation a
0.25
Enter the probability for q -> q for the observation b
0
Enter the probability for q -> r for the observation b
0.25
Enter the probability for q -> s for the observation b
0.25
Enter the probability for r -> q for the observation a
0.5
Enter the probability for r -> r for the observation a
0
Enter the probability for r -> s for the observation a
0
Enter the probability for r -> q for the observation b
0.5
Enter the probability for r -> r for the observation b
0
Enter the probability for r -> s for the observation b
0
Enter the probability for s -> q for the observation a
0.5
Enter the probability for s -> r for the observation a
0
Enter the probability for s -> s for the observation a
0
Enter the probability for s -> q for the observation b
0.5
Enter the probability for s -> r for the observation b
0
Enter the probability for s -> s for the observation b
0
{('q', 'a', 'q'): 0.0, ('q', 'a', 'r'): 0.25, ('q', 'a', 's'): 0.25,
('q', 'b', 'q'): 0.0, ('q', 'b', 'r'): 0.25, ('q', 'b', 's'): 0.25,
('r', 'a', 'q'): 0.5, ('r', 'a', 'r'): 0.0, ('r', 'a', 's'): 0.0, ('r',
'b', 'q'): 0.5, ('r', 'b', 'r'): 0.0, ('r', 'b', 's'): 0.0, ('s', 'a',
'q'): 0.5, ('s', 'a', 'r'): 0.0, ('s', 'a', 's'): 0.0, ('s', 'b', 'q'):
0.5, ('s', 'b', 'r'): 0.0, ('s', 'b', 's'): 0.0}
```

OUTPUT:
For observed sequence : aabb

```
Best State Sequence for aabb is qrqrq
Best State Sequence Probability for aabb is 0.015625

The Alpha vector for time tick are: [[1, 0, 0], [0.0, 0.25, 0.25], [0.25, 0.0, 0.0], [0.0, 0.0625, 0.0625], [0.0625, 0.0, 0.0]]
The value of p1,w is: 0.0625

The Beta Vectors for time Tick are: [[1, 1, 1], [0.5, 0.5, 0.5], [0.25, 0.25, 0.25], [0.125, 0.125, 0.125], [0.0625, 0, 0]]
The probility of p1,w is: 0.0625

Final Value of T after Training:
{('q', 'a', 'q'): 0.0, ('q', 'a', 'r'): 0.25, ('q', 'a', 's'): 0.25, ('q', 'b', 'q'): 0.0, ('q', 'b', 'r'): 0.25, ('q', 'b', 's'): 0.25, ('r', 'a', 'q'): 0.5, ('r', 'a', 'r'): 0.0, ('r', 'a', 's'): 0.0, ('r', 'b', 'q'): 0.5, ('r', 'b', 'r'): 0.0, ('r', 'b', 's'): 0.0, ('s', 'a', 'q'): 0.5, ('s', 'a', 'r'): 0.0, ('s', 'a', 's'): 0.0, ('s', 'b', 'q'): 0.5, ('s', 'b', 'r'): 0.0, ('s', 'b', 's'): 0.0}
```

Using the Algorithm defined in the reference:

- Here the A, and B matrices were used.
- It was also observed that while assuming random values of A and B matrices the final probabilities always differ for each randomly generated A and B.

TEST 2:

```
A = np.array([[0.96,1.0],[0.04,0.0]])
B = np.array([[0.52,0.48],[1.0,0.0]])
Prob_Obervations = 0.03456
Initial state probabilities pi:
 [1. 0.]
Final A matrix:
 [[1.00000000e+00 1.30423689e-22]
 [1.00000000e+00 0.00000000e+00]]
Final B matrix:
 [[0.4 0.6]
 [1.  0. ]]
```

TEST 3:

```
A = np.array([[0,1.0,1.0],[0.5,0.0,0.0],[0.5,0.0,0.0]])
B =np.array([[0.5,0.5],[0.5,0.5],[0.5,0.5]])
Prob_Obervations = 0.0625
Initial state probabilities pi:
 [1. 0. 0.]
Final A matrix:
 [[0.  0.5 0.5]
 [1.  0.  0. ]
 [1.  0.  0. ]]
Final B matrix:
 [[0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]]
```

POS Tagging:

- For POS tagging the above defined algorithm is used.

Output:

```
Trainig Corpus:
gaurav jayesh can see will
spot will see gaurav
will jayesh spot gaurav?,gaurav will pat spot.

Observations:
 ('jayesh', 'will', 'spot', 'will')

Expected Output:
N M V N

           0        1        2        3
N: 0.16500 0.00199 0.00234 0.00084
M: 0.00000 0.04083 0.00000 0.00058
V: 0.00000 0.00000 0.00765 0.00000
The steps of states are N M V N with highest probability of
0.0008422734375000001
```

Note:

- More examples can be run similarly.
- By using random probabilities and running the algorithm.