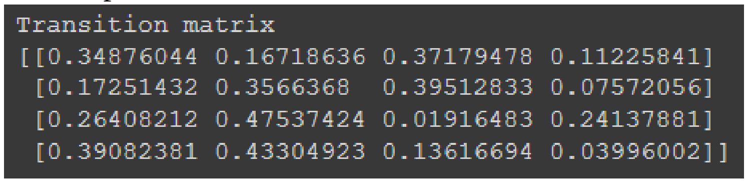
HMM RESULTS

Seed used - 200321513

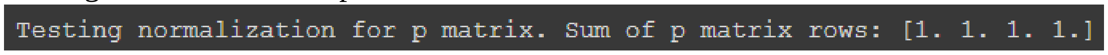
Task 1: Data set generation

One-step transition matrix:



```
Transition matrix
[[0.34876044 0.16718636 0.37179478 0.11225841]
 [0.17251432 0.3566368  0.39512833 0.07572056]
 [0.26408212 0.47537424 0.01916483 0.24137881]
 [0.39082381 0.43304923 0.13616694 0.03996002]]
```

Testing normalization for p-matrix:

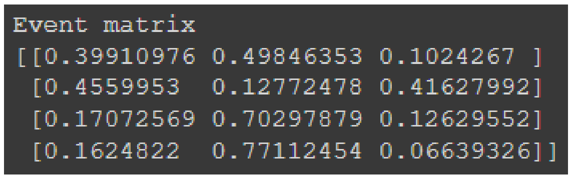


```
Testing normalization for p matrix. Sum of p matrix rows: [1. 1. 1. 1.]
```

From the p matrix results we can make the following conclusions:
1. When in state 1, the highest state probability is that the next state is state 3. The probability that is remains in the same state.
2. When in state 2, the next most probable state is again 3. Staying in the same state 2 is also highly probable.
3. The most probable next state when in state 3 is state 2. This time, there is very low probability that the state remains the same.
4. When in state 4, the most probable next state is state 2. The second most probable state is state 1. Similar to state 3, there is very low probability that it remains in the same state.

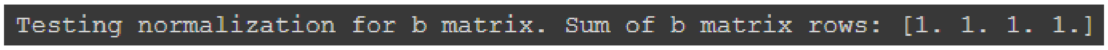
The most probable next states are 2, and 3.
All the rows add-up to 1. Indicating that the probabilities are normalized.

Event matrix:



```
Event matrix
[[0.39910976 0.49846353 0.1024267 ]
 [0.4559953  0.12772478 0.41627992]
 [0.17072569 0.70297879 0.12629552]
 [0.1624822  0.77112454 0.06639326]]
```

Normalization for b-matrix:

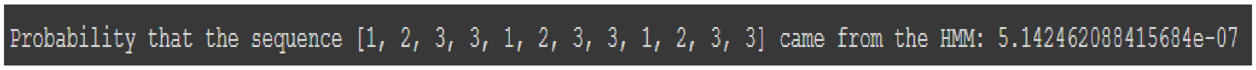


```
Testing normalization for b matrix. Sum of b matrix rows: [1. 1. 1. 1.]
```

1. If in state 1, the most probable observation is V = 2
2. If in state 2, the most probable observation is V = 1
3. If in state 3, the most probable observation is V = 2
4. If in state 4, the most probable observation is V = 2

Task 2: Estimate p(O|λ)

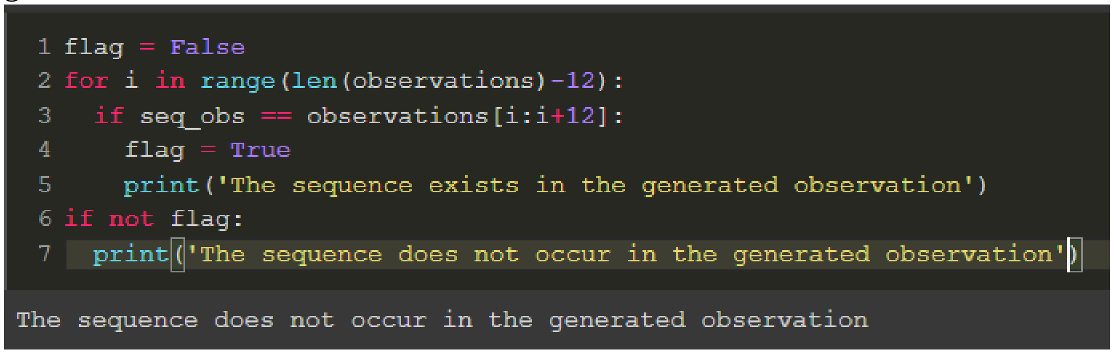
We use the forward algorithm to calculate the probability that a given observation came from a given HMM model.



```
Probability that the sequence [1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3] came from the HMM: 5.142462088415684e-07
```

The probability that the sequence 1,2,3,3,1,2,3,3,1,2,3,3 was generated from the given HMM is very low (5e-07), almost 0.

As a sanity check, we check if the given sequence ever occurred in the 1000 observations that we generated.



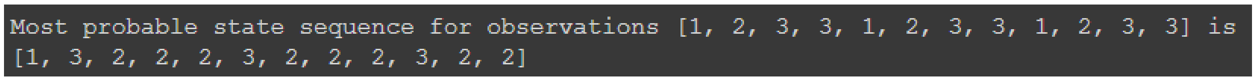
```
flag = False
for i in range(len(observations)-12):
  if seq_obs == observations[i:i+12]:
    flag = True
    print('The sequence exists in the generated observation')
if not flag:
  print('The sequence does not occur in the generated observation')
```

```
The sequence does not occur in the generated observation
```

We can confirm that the sequence does not appear in the observations that we generated. This justifies the very low probability that we get for the sequence.

Task 3: Estimate the most probable sequence Q

We use the Viterbi algorithm to generate the most probable state sequence for the given set of observation.



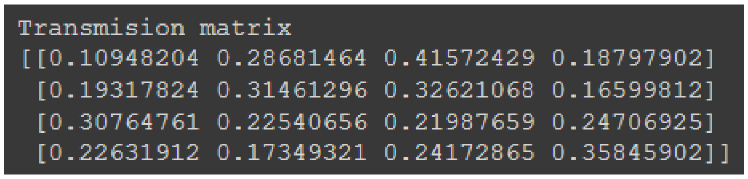
```
Most probable state sequence for observations [1, 2, 3, 3, 1, 2, 3, 3, 1, 2, 3, 3] is
[1, 3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2]
```

The initial distribution is (1,0,0,0) so it makes sense that the first state is 1. When in state 1, the next most probable state is 3. When in state three the most probable observed value, O[2], is 2. When in state 3 the next most probable state is 2 and that is what we observe. When in state 2 it is highly likely that the next state is 2. This is what we observe as the next 2 states are 2. When in state 2, the two most likely observation is 1 and 3 (they have very similar probability, 0.45, and 0.41). So, we see both of these observations when in state 2. In state 3 the most likely observation is V = 2. Looking at the p and the b matrices we can see that the probable states sequence generated using Viterbi is correct.

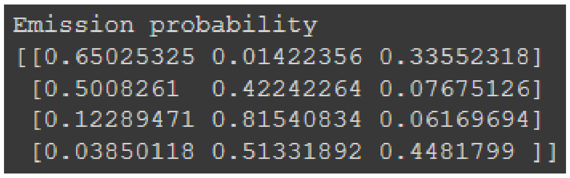
Reference: - The following link was used to get the forward and Viterbi algorithm used in the code.
http://www.adeveloperdiary.com/data-science/machine-learning/implement-viterbi-algorithm-in-hidden-markov-model-using-python-and-r/

Task 4: Train the HMM

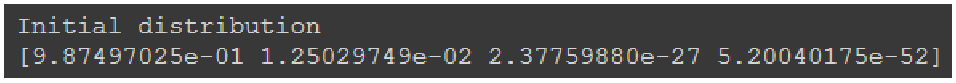
Estimated parameters:



```
Transmision matrix
[[0.10948204 0.28681464 0.41572429 0.18797902]
 [0.19317824 0.31461296 0.32621068 0.16599812]
 [0.30764761 0.22540656 0.21987659 0.24706925]
 [0.22631912 0.17349321 0.24172865 0.35845902]]
```



```
Emission probability
[[0.65025325 0.01422356 0.33552318]
 [0.5008261  0.42242264 0.07675126]
 [0.12289471 0.81540834 0.06169694]
 [0.03850118 0.51331892 0.4481799 ]]
```



```
Initial distribution
[9.87497025e-01 1.25029749e-02 2.37759880e-27 5.20040175e-52]
```

The initial distribution is similar to the initial distribution we generated. The estimated initial distribution is close to [1,0,0,0] same as the initial distribution we assumed [1,0,0,0].

The transmission matrix is different from the p-matrix we defined. It is more evenly distributed than the pre-defined p-matrix.
1. When in state 1, the highest state probability is that the next state is state 3. The probability that is remains in the same state. This is similar to the p-matrix we generated before
2. When in state 2, the next most probable state is again 3. Staying in the same state 2 is also highly probable. This is the same as p-matrix we generated earlier.
3. The most probable next state when in state 3 is state 1. This time, there is very low probability that the state remains the same. Different from the pre-generated p-matrix
4. When in state 4, the most probable next state is state 4. This is again, different from the pre-generated p-matirx.

The emission matrix is also different from the b-matrix we generated in task 1.
1. If in state 1, the most probable observation is V = 1
2. If in state 2, the most probable observation is V = 1
3. If in state 3, the most probable observation is V = 2
4. If in state 4, the most probable observation is V = 2
V=1 and V=2 are the two most probable observations according to the estimated parameters.

The hmmlearn api does not provide us with the p-values. From the estimated parameters, we can conclude that the estimation may not be a good fit for the generated observations.