Assignment 2 - HMM Questions

1. **This problem can be formulated in matrix form. Please specify the initial probability vector π, the transition probability matrix A, and the observation probability matrix B.**

Initial Probability Vector: [1, 2, 0.5, 0.5]

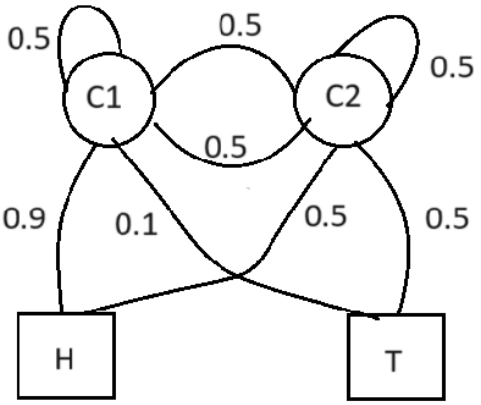
$$\pi = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$$

Transition Probability Matrix: [2, 2, 0.5, 0.5, 0.5, 0.5]

$$A = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Observation Probability Matrix: [2, 2, 0.9, 0.1, 0.5, 0.5]

$$B = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$



2. **“First, we need to multiply the transition matrix with our current estimate of states.” What is the result of this operation?**

$$\begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$$

Estimates the probability of the next step.

3. **“In the following, the result must be multiplied with the observation matrix.” What is the result of this operation?**

$$\begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$$

The probability distribution over the observable outputs. Distribution over the observation space for the next step (time t+1).

4. **Why is it valid to substitute O1:t = o1:t with Ot = ot when we condition on the state Xt = xi?**

The substitution O1:t = o1:t with Ot = ot when conditioning on the state Xt = xi is valid because of Markov’s assumption in HMMs. The Markov assumption states that the future state only depends on the current state, *not* on the sequence of events that preceded it.

Given the current state Xt, the past observations O1:t-1 do not provide additional information regarding the future observations Ot. The Markov assumption simplifies the computation, allowing us to express the probability in terms of the current state and the probability of transitioning from the previous state to the current state.

5. **How many values are stored in the matrices δ and δidx respectively?**

Matrix δ: N x T dimensions, where N is the number of hidden states and T is the length of the observation sequence. Each entry δt(i) stores the probability of the most likely path up to time t that ends in state i. Thus, N x T values are stored in the matrix δ.

Matrix δidx: N x T dimensions, same as Matrix δ. Each entry δt(i) stores the index of the most like predecessor state at time t for state i. There are therefore also N x T values stored in this matrix.

6. **Why we do we need to divide by the sum over the final α values for the di-gamma function?**

This is done to normalize the probabilities. The sum we divide by is the total probability of observing the entire sequence O1:T. By normalizing with this sum, given the observed sequence, the di-gamma values are ensured to represent the probability of being in state i at time t and state j at time t+1. It is necessary since the raw joint probabilities might not sum to 1 due to the dynamic programming nature of the forward and backward algorithms.

7. **Does the algorithm converge? How many observations do you need for the algorithm to converge? How can you define convergence?**

Converged in: 26. The convergence can be defined as calculating normalization between the new matrix and the old. By specifying a convergence threshold we can break out of the calculations.

8. **Train an HMM with the same parameter dimensions as above, i.e. A is a 3x3 matrix, etc. The initialization is left up to you. How close do you get to the parameters above, i.e. how close do you get to the generating parameters in Eq. 3.1? What is the problem when it comes to estimating the distance between these matrices? How can you solve these issues?**

Input:
3 3 0.12 0.44 0.44 0.15 0.31 0.53 0.36 0.39 0.25
3 4 0.12 0.29 0.37 0.21 0.43 0.15 0 0.42 0.08 0.13 0.5 0.29
1 3 0.45 0.5 0.05

1000 input:
3 3 0.818663 0.103323 0.078014 0.011501 0.625672 0.362826 0.24001 0.230423 0.529567
3 4 0.051195 0.40194 0.27966 0.267206 0.747057 0.252935 0.0 8e-06 0.001834 0.030947 0.413927 0.553293
Converged in: 225

10000 input:
3 3 0.612414 0.078377 0.309209 0.195686 0.54998 0.254334 0.1754 0.189521 0.635079
3 4 0.073184 0.170923 0.165993 0.589901 0.881868 0.118126 0.0 7e-06 0.06495 0.425283 0.353802 0.155965
Converged in: 338

An improvement to this would be to create better start parameters rather than randomly choosing.

9. **Train an HMM with different numbers of hidden states. What happens if you use more or less than 3 hidden states? Why? Are three hidden states and four observations the best choice? If not, why? How can you determine the optimal setting? How does this depend on the amount of data you have?**

I found that using 4 hidden states yielded better results than using three hidden states. A general rule is that for more data you should use more states to avoid overfitting and for less data, you should use fewer states.

Fewer states can lead to underfitting (missing patterns in the data) and more states can lead to overfitting.

10. **Initialize your Baum-Welch algorithm with a uniform distribution. How does this affect the learning? Initialize your Baum-Welch algorithm with a diagonal A matrix and π= [0, 0, 1]. How does this affect the learning? Initialize your Baum-Welch algorithm with matrices that are close to the solution. How does this affect the learning?**

Using uniform initialization leads to a non-biased initial starting point which can in many cases positively affect the result to avoid a bad starting point. Using the diagonal matrix leads to faster convergence and will have a hard time escaping the initial data. When setting the parameters close to the solution led to also giving a final solution that was extremely close to the solution. Generally speaking, it should lead to a fast convergence.