EQ2341 Assignment 2

Yijie Li

May 22, 2025

1 Forward Algorithm

1.1 Implementation of the Scaled Forward Algorithm

In accordance with the formulation presented in equations (5.42) to (5.53), I implemented the scaled forward algorithm. The algorithm computes the scaled forward variable $\hat{\alpha}_{j,t}$ recursively to ensure numerical stability when dealing with long observation sequences.

Initialization ($t = 1$): For each state $j = 1, \ldots, N$, we compute the initial forward variable as:

$$\alpha_{j,1}^{\text{temp}} = P[X_1 = \mathbf{x}_1, S_1 = j \mid \lambda] = q_j \cdot b_j(\mathbf{x}_1) \tag{5.42}$$

$$c_1 = \sum_{k=1}^{N} \alpha_{k,1}^{\text{temp}} \tag{5.43}$$

$$\hat{\alpha}_{j,1} = \frac{\alpha_{j,1}^{\text{temp}}}{c_1} \tag{5.44}$$

Recursion ($t = 2, \ldots, T$): At each subsequent time step t, the scaled forward variable is computed by:

$$\alpha_{j,t}^{\text{temp}} = b_j(\mathbf{x}_t) \cdot \left(\sum_{i=1}^{N} \hat{\alpha}_{i,t-1} \cdot a_{ij} \right) \tag{5.50}$$

$$c_t = \sum_{k=1}^{N} \alpha_{k,t}^{\text{temp}} \tag{5.51}$$

$$\hat{\alpha}_{j,t} = \frac{\alpha_{j,t}^{\text{temp}}}{c_t} \tag{5.52}$$

Here, $b_j(\mathbf{x}_t)$ denotes the probability of observing $\mathbf{x}_t$ given the state j at time t.

Termination (finite-duration HMM): For finite-duration HMMs, we additionally compute the exit probability into the special END state $S_{T+1} = N + 1$, using the final forward step:

$$\begin{aligned} c_{T+1} &= P[S_{T+1} = N + 1 \mid \mathbf{x}_1, \ldots, \mathbf{x}_T, \lambda] \\ &= \sum_{k=1}^{N} \hat{\alpha}_{k,T} \cdot a_{k,N+1} \end{aligned} \tag{5.53}$$

This final scaling constant c_{T+1} is necessary for computing the complete likelihood of the observed sequence:

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T+1} c_t \quad \Rightarrow \quad \log P = \sum_{t=1}^{T+1} \log c_t$$

Code

```
    def forward(self, pX):
        """
        alpha_hat, c = forward(self, pX)
        Forward algorithm for Markov Chain.
        Input:
        pX= matrix of size (nStates, nSamples)
        pX(i,j)= P[X(j) | S(t)=i]

        Result:
        alpha_hat= matrix of size (nStates, nSamples)
        alpha_hat(i,j)= P[S(t)=i | X(1:j)]
        c= vector of size (nSamples)
        c(j)= P[X(1:j)]
        """

        # Initialize
        nSamples = pX.shape[1]
        alpha_hat = np.zeros((self.nStates, nSamples))
        c = np.zeros(nSamples)
        if self.is_finite:
            A_square = self.A[:,:-1]  # Exclude the last column (END state)
        else:
            A_square = self.A

        #t =1
        alpha_temp = self.q * pX[:, 0]
        c[0] = np.sum(alpha_temp)
        alpha_hat[:, 0] = alpha_temp / c[0]

        #t = 2..nSamples
        for t in range(1, nSamples):
            alpha_temp = A_square.T @ alpha_hat[:, t-1] * pX[:, t]
            c[t] = np.sum(alpha_temp)
            alpha_hat[:, t] = alpha_temp / c[t]

        # termination
        if self.is_finite:
            c= np.append(c, alpha_hat[:,-1] @ self.A[:,-1])

        return alpha_hat, c
```

1.2 Manual Calculation of the Scaled Forward Algorithm

Given a finite-duration HMM with the following parameters:

- Initial state distribution:

$$q = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Transition matrix including termination state:

$$A = \begin{bmatrix} 0.9 & 0.1 & 0.0 \\ 0.0 & 0.9 & 0.1 \end{bmatrix}$$

- State-dependent output distributions: - State 1: $x \sim \mathcal{N}(0, 1)$ - State 2: $x \sim \mathcal{N}(3, 2^2)$ - Observation sequence:

$$x = [-0.2,\ 2.6,\ 1.3]$$

We compute the scaled forward variable $\hat{\alpha}_{j,t}$ using the algorithm defined by equations (5.42)–(5.53), which includes scaling factors c_t for numerical stability.

Step 1: Compute Observation Likelihoods. Using the Gaussian PDF, the likelihoods $b_j(x_t)$ for each state and time step are:

$$pX = \begin{bmatrix} 0.39104269 & 0.01358297 & 0.17136859 \\ 0.05546042 & 0.19552135 & 0.13899244 \end{bmatrix}$$

Step 2: Initialization $(t = 1)$

$$\begin{aligned} \alpha_{1,1}^{\text{temp}} &= 1.0 \cdot 0.3910 = 0.3910 \\ \alpha_{2,1}^{\text{temp}} &= 0.0 \cdot 0.0555 = 0.0000 \\ c_1 &= 0.3910 + 0.0000 = 0.3910 \\ \hat{\alpha}_{1,1} &= 0.3910/0.3910 = 1.0000, \quad \hat{\alpha}_{2,1} = 0.0000 \end{aligned}$$

Step 3: Recursion $(t = 2)$

$$\begin{aligned} \alpha_{1,2}^{\text{temp}} &= 0.0136 \cdot (1.0 \cdot 0.9 + 0 \cdot 0) = 0.0122 \\ \alpha_{2,2}^{\text{temp}} &= 0.1955 \cdot (1.0 \cdot 0.1 + 0 \cdot 0.9) = 0.0196 \\ c_2 &= 0.0122 + 0.0196 = 0.0318 \\ \hat{\alpha}_{1,2} &= 0.0122/0.0318 \approx 0.3847, \quad \hat{\alpha}_{2,2} = 0.0196/0.0318 \approx 0.6153 \end{aligned}$$

Step 4: Recursion $(t = 3)$

$$\begin{aligned} \alpha_{1,3}^{\text{temp}} &= 0.1714 \cdot (0.3847 \cdot 0.9 + 0.6153 \cdot 0) = 0.0594 \\ \alpha_{2,3}^{\text{temp}} &= 0.1390 \cdot (0.3847 \cdot 0.1 + 0.6153 \cdot 0.9) = 0.0822 \\ c_3 &= 0.0594 + 0.0822 = 0.1417 \\ \hat{\alpha}_{1,3} &= 0.0594/0.1417 \approx 0.4189, \quad \hat{\alpha}_{2,3} = 0.0822/0.1417 \approx 0.5811 \end{aligned}$$

Step 5: Termination For finite-duration HMMs, we compute the exit probability into the END state:

$$c_4 = \hat{\alpha}_{1,3} \cdot 0.0 + \hat{\alpha}_{2,3} \cdot 0.1 = 0.0581$$

Final Results The scaled forward matrix $\hat{\alpha}_{j,t}$ is:

$$\hat{\alpha}_{j,t} = \begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{1,2} & \hat{\alpha}_{1,3} \\ \hat{\alpha}_{2,1} & \hat{\alpha}_{2,2} & \hat{\alpha}_{2,3} \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.3847 & 0.4189 \\ 0.0000 & 0.6153 & 0.5811 \end{bmatrix}$$

The scaling vector is:

$$c = [0.3910,\ 0.0318,\ 0.1417,\ 0.0581]$$

This confirms that the scaled forward algorithm produces numerically stable and theoretically correct results consistent with the textbook example.

1.3 Verification result

```
Alpha hat:
 [[1.         0.38470424 0.41887466]
 [0.         0.61529576 0.58112534]]
C:
 [0.39104269 0.03177681 0.14165001 0.05811253]
logp: -9.187726979475208
```

Figure 1: Result of forward algorithm

2 Backward Algorithm

2.1 Implementation of Algorithm

Initialization (Termination Step). At the final time step $t = T$, the backward variable is initialized as:

$$\hat{\beta}_{i,T} = \frac{a_{i,N+1}}{c_T \cdot c_{T+1}}$$

for the finite-duration HMM model, where $a_{i,\text{END}}$ denotes the transition probability from state i to the END state.

Recursion. For $t = T - 1$ down to 1, the scaled backward variable is computed recursively by:

$$\hat{\beta}_{i,t} = \frac{1}{c_{t+1}} \sum_{j=1}^{N} a_{ij} \cdot b_j(x_{t+1}) \cdot \hat{\beta}_{j,t+1}$$

Code

```
def backward(self, c, pX):
        """
        beta_hat = backward(self, c, pX)
        Backward algorithm for Markov Chain.
        Input:
        c= vector of size (nSamples)
        c(j)= P[X(1:j)]
        pX= matrix of size (nStates, nSamples)
        pX(i,j)= P[X(j) | S(t)=i]
        Result:
        beta_hat= matrix of size (nStates, nSamples)
        beta_hat(i,j)= P[X(j+1:nSamples) | S(t)=i]
        """
```

```
#initializa
T = pX.shape[1]
beta_hat = np.zeros((self.nStates, T))

if self.is_finite:
    A_square = self.A[:,:-1]
    beta_hat[:,-1] = self.A[:,-1] /(c[T]*c[T-1])
else:
    A_square = self.A
    beta_hat[:,-1] = 1/c[-1]

for t in range(T-2, -1, -1):
    beta_hat[:, t] = A_square @ (beta_hat[:, t+1] * pX[:, t+1])
    beta_hat[:, t] = beta_hat[:, t] / c[t]

return beta_hat
```

2.2 Theoretical Computation of Scaled Backward Variable $\hat{\beta}_{i,t}$

Given the parameters of a finite-duration Hidden Markov Model (HMM):

- Number of states: $N = 2$ - Sequence length: $T = 3$ - Transition matrix A (including termination state) is:

$$A = \begin{bmatrix} 0.9 & 0.1 & 0.0 \\ 0.0 & 0.9 & 0.1 \end{bmatrix}$$

- Observation likelihood matrix pX (columns represent time steps $t = 1, 2, 3$; rows represent states $i = 1, 2$):

$$pX = \begin{bmatrix} 0.39104269 & 0.01358297 & 0.17136859 \\ 0.05546042 & 0.19552135 & 0.13899244 \end{bmatrix}$$

- The scaling factors c obtained from the forward algorithm are:

$$c = [0.39104269,\ 0.03177681,\ 0.14165001,\ 0.05811253]$$

We now compute the scaled backward variable $\hat{\beta}_{i,t}$ for $t = 3,\ 2,\ 1$ in reverse order.

Step 1: Compute $\hat{\beta}_{i,3}$

Using the termination condition for finite-duration HMMs:

$$\hat{\beta}_{i,3} = \frac{a_{i,N+1}}{c_3 \cdot c_4}$$

Given: - $a_{1,N+1} = 0.0$, $a_{2,N+1} = 0.1$ - $c_3 = 0.14165001$, $c_4 = 0.05811253$

We have:

$$\hat{\beta}_{1,3} = \frac{0.0}{0.14165001 \cdot 0.05811253} = 0.0000$$

$$\hat{\beta}_{2,3} = \frac{0.1}{0.14165001 \cdot 0.05811253} = \frac{0.1}{0.00823698} \approx 12.1482$$

Step 2: Compute $\hat{\beta}_{i,2}$

The scaled backward recursion is:

$$\hat{\beta}_{i,2} = \frac{1}{c_2} \sum_{j=1}^{2} a_{ij} \cdot b_j(x_3) \cdot \hat{\beta}_{j,3}$$

With: - $c_2 = 0.03177681$ - $b_1(x_3) = 0.17136859$, $b_2(x_3) = 0.13899244$ - $\hat{\beta}_3 = [0.0,\ 12.1482]$
Calculations:

$$\hat{\beta}_{1,2} = \frac{1}{0.03177681} (0.1 \cdot 0.13899244 \cdot 12.1482) = \frac{0.1690}{0.03177681} \approx 5.3136$$

$$\hat{\beta}_{2,2} = \frac{1}{0.03177681} \cdot (0.9 \cdot 0.13899244 \cdot 12.1482) = \frac{1.5148}{0.03177681} \approx 47.8228$$

Step 3: Compute $\hat{\beta}_{i,1}$

$$\hat{\beta}_{i,1} = \frac{1}{c_1} \sum_{j=1}^{2} a_{ij} \cdot b_j(x_2) \cdot \hat{\beta}_{j,2}$$

Where: - $c_1 = 0.39104269$ - $b_1(x_2) = 0.01358297$, $b_2(x_2) = 0.19552135$ - $\hat{\beta}_2 = [5.3136,\ 47.8228]$
Calculations:

$$\hat{\beta}_{1,1} = \frac{1}{0.39104269} (0.9 \cdot 0.01358297 \cdot 5.3136 + 0.1 \cdot 0.19552135 \cdot 47.8228) = \frac{0.7947}{0.39104269} \approx 2.5573$$

$$\hat{\beta}_{2,1} = \frac{1}{0.39104269} (0.9 \cdot 0.19552135 \cdot 47.8228) = \frac{8.4143}{0.39104269} \approx 21.5203$$

Final Result

The full result is summarized in matrix form:

$$\hat{\beta}_{i,t} = \begin{bmatrix} \hat{\beta}_{1,1} & \hat{\beta}_{1,2} & \hat{\beta}_{1,3} \\ \hat{\beta}_{2,1} & \hat{\beta}_{2,2} & \hat{\beta}_{2,3} \end{bmatrix} = \begin{bmatrix} 2.5573 & 5.3136 & 0.0000 \\ 21.5203 & 47.8228 & 12.1482 \end{bmatrix}$$

2.3 Verification result

```
px:
 [[0.39104269 0.01358297 0.17136859]
 [0.05546042 0.19552135 0.13899244]]
c: [0.39104269 0.03177681 0.14165001 0.05811253]
beta_hat: [[ 2.55726552  5.31366894  0.        ]
 [21.52035922 47.82302049 12.14824553]]
```

Figure 2: Result of backward algorithm

The results show exact numerical agreement between the theoretical values and the computed results across all time steps and states, with differences within machine precision.