EQ2341 Pattern Recognition and Machine Learning

Assignment 4: Backward Algorithm

Hanqi Yang hanqi@kth.se

Qian Zhou qianzho@kth.se

May 13, 2022

1 Introduction

In this assignment we will implement one of two very important HMM algorithms: the Backward Algorithm. This algorithm calculates a matrix β of conditional probabilities of the final part of an observed sequence $(x_{t+1}, ..., x_T)$, given an HMM λ and the state $S_t = i$ at time t. The result is known as the backward variables, which has a slight difference between infinite and finite duration HMM. The backward algorithm consists of two steps: initialization, backward step.

2 Backward algorithm procedure

In this section we will discuss how to implement the backward algorithm, which is very similar to the forward algorithm. In this case, the starting point is at the end of the observed sequence and we work back towards the starting point. This algorithm calculates a matrix β of conditional state probabilities of the final part, given an observed feature sequence $(x_{t+1}, ..., x_T, ...)$, a known HMM parameter $\lambda = \{q, A, B\}$ and the state $S_t = i$ as

$$\beta_{i,t} = P(x_{t+1}, ..., x_T | S_t = i, \lambda) \tag{1}$$

for an infinite-furation HMM, and

$$\beta_{i,t} = P(x_{t+1}, ..., x_T, S_{T+1} = N + 1 | S_t = i, \lambda) \tag{2}$$

for a finite-furation HMM.

Equation 3 to 5 expose the mathematical methodology of the backward algorithm.

Initialization: When $t = T$, for an infinite-furation HMM:

$$\hat{\beta}_{i,T} = 1/c_T \tag{3}$$

for a finite-duration HMM:

$$\hat{\beta}_{i,T} = a_{i,N+1}/(c_T c_{T+1}) \tag{4}$$

Backward step: For $t = T - 1, T - 2, ..., 1$,

$$\hat{\beta}_{i,t} = \frac{1}{c_t} \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \hat{\beta}_{j,t+1}, \ t = T - 1, T - 2, ..., 1 \tag{5}$$

The derivation is exactly the same between the infinite-duration HMM and finite-duration HMM, except that the special exit condition $S_{T+1} = N + 1$ is not included.

Using the three steps above, the function `@MarkovChain/backward` is implemented and returns the results of the scaled backward variable $\hat{\beta}_{j,t}$.

3 Results

In order to implement the backward algorithm, we complete the `@MarkovChain/backward`. Then we just need a main function to test our code.

3.1 Finite duration HMM

We create a finite-duration test HMM with a Markov chain given by

$$q = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.9 & 0.1 \end{pmatrix} \tag{6}$$

and where the state-conditional output distribution is a scalar Gaussian with mean $\mu_1 = 0$ and standard deviation $\sigma_1 = 1$ for state 1, and another Gaussian with $\mu_2 = 3$ and $\sigma_2 = 2$ for state 2. From assignment 3, for this HMM and the observation sequence, the forward algorithm gives the scale factors $c_t = (1, 0.1625, 0.8266, 0.0581)$. Then we use our own backward algorithm to calculate $\hat{\beta}_{j,t},\ t = 1, 2, 3$, the results are

```
[[1.         1.03893571 0.        ]
 [8.41537925 9.35042138 2.08182773]]
```

Figure 1: Result of finite-duration HMM

The result corresponds with the calculation by hand. Therefore, our implementation of the backward algorithm works well for the finite-duration HMM.

3.2 Infinite duration HMM

For the infinite duration HMM, the parameters are given as

$$q = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} 0.3 & 0.7 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.4 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.6 \end{pmatrix} \tag{7}$$

The observed finite duration feature sequence is $x = (1, 2, 4, 4, 1)$. From assignment 3, we got $c_t = (1, 0.35, 0.35, 0.564, 0.099)$. Using our code to test, we get the result as

```
[[ 1.         0.33121019 0.17834395 5.35031847 10.06369427]
 [ 0.97634213 2.85714286 1.65605096 0.89171975 10.06369427]
 [ 0.52411283 5.2411283  3.05732484 1.78343949 10.06369427]]
```

Figure 2: Result of infinite-duration HMM

The results correspond with what they should be as

$$\hat{\beta} = \begin{pmatrix} 1 & 0.331 & 0.178 & 5.35 & 10.1 \\ 0.976 & 2.86 & 1.66 & 0.892 & 10.1 \\ 0.524 & 5.24 & 3.06 & 1.78 & 10.1 \end{pmatrix} \tag{8}$$

Therefore, our code works well also for infinite-duration HMM.

4 Conclusion

In this assignment, we implement the backward algorithm and test them with finite and infinite duration HMM. The results of $\hat{\beta}$ for both systems are almost the same as the theoretical values. In conclusion, our backward algorithm works well for both finite and infinite duration HMM.

Appendix

Python code

@MarkovChain/backward:

```
def backward(self, c, pX):
    T = len(pX[0, :])
    N = self.nStates
    A = self.A
    betaHat = np.zeros((N, T))
    finitestate = 0
    if N != len(A[0, :]):
    finitestate = 1

    # Initialization
    if finitestate == 0:
        for i in range(0, N):
            betaHat[i, T-1] = 1/c[T-1]

    if finitestate == 1:
        for i in range(0, N):
            betaHat[i, T-1] = A[i, N]/(c[T]*c[T-1])

    # Backward Step
    if finitestate == 0:
        for j in range(T-1, 0, -1):
            for i in range(0, N):
                betaHat[i, j-1] = (A[i, :]@(pX[:, j]*
                    betaHat[:, j]))/c[j-1]

    if finitestate == 1:
        A = np.delete(A, N, 1)
        for j in range(T-1, 0, -1):
            for i in range(0, N):
                betaHat[i, j-1] = (A[i, :]@(pX[:, j]*
                    betaHat[:, j]))/c[j-1]

    return betaHat
```