The Backward Algorithm

EQ2341 Pattern Recognition and Machine Learning, Assignment 4

Jingxuan Mao
jmao@kth.se

Yuqi Zheng
yuqizh@kth.se

May 12, 2022

1 Implementation of the Backward Algorithm

@MarkovChain/ backward

```
def backward(self, px, c):
        """
        Implementation of the backward algorithm

        Input: px= The state-conditional probability mass or
                                        density values scaled
        by the largest probability of each frame in the observed
                                        sequence.
               c= The forward sclae factors

        Output: betaHat= The scaled forward variables

        """
        # Initialization of all backward parameters
        betaHat = np.zeros((self.nStates, px.shape[1]))

        # Step1: Intialization
        if self.is_finite == False:
            betaHat[:,-1] = np.array([ [1/c[-1]], [1/c[-1]] ])
        else:
            betaHat[:,-1] = self.A[:, -1]/(c[c.shape[0]-2]*c[-1])

        # Step2: Backward step
        for t in range(px.shape[1]-1, 0, -1):
            t = t-1
            for i in range(self.nStates):
                betaHat[i, t] = 1/c[t] * \
                np.sum(self.A[i, 0:self.nStates].T * \
                px[:, t+1] * betaHat[:, t+1])


        return betaHat
```

2 Verification of the Backward Algorithm

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

$$p = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; A = \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.9 & 0.1 \end{pmatrix}$$

The state-conditional output distributions for state 1 and state 2 are scalar Gaussians with means $\mu_1 = 0, \mu_2 = 3$ and standard deviations $\sigma_1 = 1, \sigma_2 = 2$. Assume that the observation sequence $x = (-0.2, 2.6, 1.3)$ and the scale factors given by Forward Algorithm is $c = (1, 0.1625, 0.8266, 0.0581)$. The main program is

```
# State generator
q = np.array( [ 1, 0 ] )
A = np.array( [ [ 0.9, 0.1, 0 ], [ 0, 0.9, 0.1 ] ] )

mc = MarkovChain(q, A)

g1 = GaussD( means=[0], stdevs=[1] )    # Distribution for state = 1
g2 = GaussD( means=[3], stdevs=[2] )    # Distribution for state = 2
h  = HMM( mc, [g1, g2])                 # The HMM

# Generate an output sequence
x = [-0.2, 2.6, 1.3]

# Generate px and scaler factors
px, scaler_px = h.Get_px(x)
np.set_printoptions(precision=4)
print("px is:")
print(px)
print()

# Print c
c = np.array([1, 0.1625, 0.8266, 0.0581])
print("c is:")
print(c)
print()

# The Backward Algorithm
betaHat = mc.backward(px, c)
print("betaHat is:")
print(betaHat)
```

With those inputs fed into The Backward Algorithm, the returned values of scaled backward variables are

```
px is:
[[1.     0.0695 1.    ]
 [0.1418 1.     0.8111]]

c is:
[1.     0.1625 0.8266 0.0581]

betaHat is:
[[1.0003 1.0393 0.    ]
 [8.4182 9.3536 2.0822]]
```