HW5_TiagoGoncalves_Exc3

January 6, 2019

3. Consider a HMM with continuous outputs.

Assume that the HMM only has two states; both states have the same initial probability; the probability of changing between different states is 0.1 (that is, the transition matrix is symmetric with 0.9 in the elements in the main diagonal). The emission density function for state 1 follows a Laplace distribution with mean 0 and standard deviation 0.2. For state 2 the emission density function is uniform in [0, 1]. Change the code provided in the class (file hmmTest.m / hmmTest.py) to compute the probability of the following sequence of length 10: {0.7, 0.7, 0.1, 0.2, 0.3, 0.6, 0.2, 0.3, 0.1, 0.2}.

Send the modified file and write the computed probability.

In [1]:
```
#Start by defining probability distribution functions
import math
import numpy as np
#Laplace Distribution Function
#ref: http://mathworld.wolfram.com/LaplaceDistribution.html
def laplace(x, mean, standard_deviation):
    scale = math.sqrt((standard_deviation**2)/2)
    return np.exp(-abs(x-mean)/scale)/(2.*scale)

#Uniform Distribution Function
#ref: https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.uniform
def uniform(x, low, high):
    probs = []
    for i in x:
        if i>=low and i<=high:
            probs.append(1/(high-low))
        else:
            probs.append(0)
    return np.array(probs)
```

In [2]:
```
#Change initial code
import math
import numpy as np

#xx = [int(c) for c in '1665626636111111122']
xx = [0.7, 0.7, 0.1, 0.2, 0.3, 0.6, 0.2, 0.3, -0.1, 0.2]
xx = np.array(xx)
```

```
        print("The given sequence is: \n", xx)

        AA = np.array([[0.9, 0.1], [0.1, 0.9]]) #transition matrix
        #print(AA)
        pi0 = np.array([0.5, 0.5])[:,None] #initial probs
        #print(pi0)

        #Probability Distributions
        #Laplace
        state1 = laplace(x=xx, mean=0, standard_deviation=0.2)
        #Uniform
        state2 = uniform(x=xx, low=0, high=1)

        Px = np.vstack((state1, state2)) # emission probs
        #print(Px)
        T = np.size(xx)
        #print(T)

        K = 2 # number of states

        # forward method - slide 31
        alpha=pi0*Px[:, 0]
        allAlpha=alpha; # only needed for slide 40
        for t in range(1,T):
            alpha=Px[:, t]*np.dot(AA.T,alpha)
            #print(Px[:, t])
            allAlpha = np.concatenate((allAlpha,alpha),1) # only needed for slide 40

        print("\nAll computed alpha: \n", allAlpha)
        SequenceProbability=np.sum(alpha)
        print("\nThe probability of the given sequence is: \n", SequenceProbability)
```

```
The given sequence is:
 [ 0.7  0.7  0.1  0.2  0.3  0.6  0.2  0.3 -0.1  0.2]

All computed alpha:
 [[1.25249788e-02 5.00000000e-01 3.13750187e-04 5.00000000e-01
  5.46948488e-04 5.00000000e-01 4.70128182e-04 5.00000000e-01
  1.99247818e-04 5.00000000e-01 1.01226248e-05 5.00000000e-01
  8.70087643e-06 5.00000000e-01 3.68757012e-06 5.00000000e-01
  6.42839746e-06 0.00000000e+00 5.52551270e-06 0.00000000e+00]
 [1.25249788e-02 5.00000000e-01 3.13750187e-04 5.00000000e-01
  5.46948488e-04 5.00000000e-01 4.70128182e-04 5.00000000e-01
  1.99247818e-04 5.00000000e-01 1.01226248e-05 5.00000000e-01
  8.70087643e-06 5.00000000e-01 3.68757012e-06 5.00000000e-01
  6.42839746e-06 0.00000000e+00 5.52551270e-06 0.00000000e+00]]

The probability of the given sequence is:
```

```
1.1051025390323771e-05
```