**Logo

Description automatically generated**

**San Francisco Bay University**

**CS483 - Fundamentals of Artificial Intelligence**

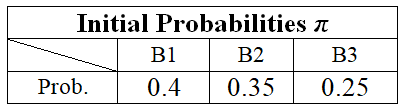
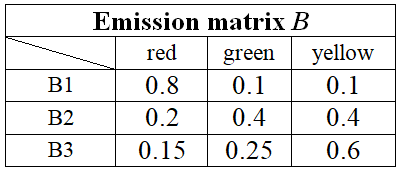
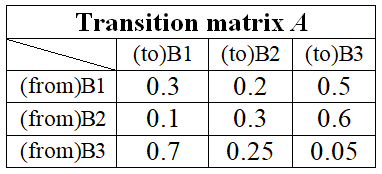
**Homework Assignment #6**

**Due day: 8/15/2022**

**Instruction:**

1. **Push the source code to Github**
2. **Overdue homework submission could not be accepted.**
3. **Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Assuming that there are 3 m&m candy bottles (labeled B1, B2, B3) with red, green and yellow colors, one of the candies from 3 bottles will be randomly taken out in a series. Given the transition probability matrix *A*, emission probability *B* and initial probability *π* as follows, if observation sequence is *red-red-yellow-green-yellow*, write python program by the functions from *"hmmlearn"* model to find probability of observation sequence *P(rrygy),* and what most likely bottles in series they are coming from.



**Solution:**

|  |
| --- |
| # %%  """      Given observations, find hidden state sequence by viterbi algorithm  """  import numpy as np  from hmmlearn import hmm  states = ["B1", "B2", "B3"]  n\_states = len(states)   # =3  observations = ["red", "red", "yellow", "green", "yellow"]  n\_observations = len(observations)  start\_probability = np.array([0.4, 0.35, 0.25])  transition\_probability = np.array([    [0.3, 0.2, 0.5],    [0.1, 0.3, 0.6],    [0.7, 0.25, 0.05]  ])  emission\_probability = np.array([    [0.8, 0.1, 0.1],    [0.2, 0.4, 0.4],    [0.15, 0.25, 0.6]  ])  model = hmm.MultinomialHMM(n\_components=n\_states)  # MultinomialHMM: observation distribution in Multinomial  model.startprob\_=start\_probability  model.transmat\_=transition\_probability  model.emissionprob\_=emission\_probability  seen = np.array([[0,0,1,2,1]]).T        # 0: red;     1: yellow   2: green => r r y g y  logprob, box = model.decode(seen, algorithm="viterbi")  seen = [0,0,1,2,1]  print("If Observation sequence is:", ", ".join(map(lambda x: observations[x], seen)))  print("Then the Bottle Series will be:", ", ".join(map(lambda x: states[x], box)))  """      Find probability of observation sequence  """  seen = np.array([[0,0,1,2,1]]).T                   # P(rrygy) =?  print(f"The model Score is: {model.score(seen)}") |

2. Suppose there are only two average annual temperatures, "Hot" and "Cold" in Fremont city of California, and also suppose that current observation in Mission San Jose district indicates a correlation between the tree size of palm, small(s)/medium(m) /large(l) and temperature, as follows are the transition matrix A, emission matrix B, and initial probabilities π.

|  |  |  |  |
| --- | --- | --- | --- |
| Emission matrix B | | | |
|  | s | m | l |
| H | 0.1 | 0.4 | 0.5 |
| C | 0.7 | 0.2 | 0.1 |

|  |  |  |
| --- | --- | --- |
| Transition matrix A | | |
|  | H | C |
| H | 0.7 | 0.3 |
| C | 0.4 | 0.6 |

|  |
| --- |
| Initial Prob. π |
| P(H) = 0.6 |
| P(C) = 0.4 |

**Solution:**

|  |
| --- |
| #Given Transition Probabilities  p\_11 = 0.7  p\_12 = 0.3  p\_21 = 0.4  p\_22 = 0.6  #Given Initial Probabilities  p\_1 = 0.6  p\_2 = 0.4  #Given Emission Probabilities  p\_1s = 0.1  p\_1m = 0.4  p\_1l = 0.5  p\_2s = 0.7  p\_2m = 0.2  p\_2l = 0.1  seq = ['S','M','L']  probabilities = []  out = []  if seq[0] == 'X':        probabilities.append((p\_1\*p\_1s, p\_2\*p\_2s))  else:       probabilities.append((p\_1\*p\_1l, p\_2\*p\_2l))  for i in range(1,len(seq)):      hot\_prev, cold\_prev = probabilities[-1]      #Given sequence has only X and Z, so we consider these cases only      if seq[i] == 'X':          hot\_today = max(hot\_prev\*p\_11\*p\_1s, cold\_prev\*p\_21\*p\_1s)          cold\_today = max(hot\_prev\*p\_12\*p\_2s, cold\_prev\*p\_22\*p\_2s)          probabilities.append((hot\_today, cold\_today))      else:          hot\_today = max(hot\_prev\*p\_11\*p\_1l, cold\_prev\*p\_21\*p\_1l)          cold\_today = max(hot\_prev\*p\_12\*p\_2l, cold\_prev\*p\_22\*p\_2l)            probabilities.append((hot\_today, cold\_today))  for p in probabilities:      if(p[0]>p[1]):          out.append('S')      elif(p[1]>p[0] ):          out.append('M')      else :          out.append('L')  #  (a) p = P(smsl HHCC) ?    p = p[0]\*p[0]\*p[1]\*p[1]        # where 0-> H and 1-> C  print("P(smsl HHCC) = ", p)  #  (b) Calculate 𝛼2(1) 𝑎𝑛𝑑 𝛼2(2) in forward algorithm    𝛼2\_1 = 0.06\*0.7\*0.4  𝛼2\_2 = 0.028\*0.6\*0.4  print("𝛼2(1) = ", 𝛼2\_1)  print("𝛼2(2) = ", 𝛼2\_2) |