Viterbi_ee20resch11012

December 25, 2020

```
[163]: import numpy as np
```

```
[164]: pi=np.array([0.8,0.2])  #initial state
       A=np.array([[0.7,0.3],[0.2,0.8]])   #Transition probabilities
       Conditional_probability=np.array([[0.8,0.2],[0.3,0.7]])     #column: H,C. Row:␣
        ↪5,1
       observation_seq=np.array([5,5,5,1,1,1,1,5,5,5])  #Given sequence of fan speed
       #observation_seq=np.array([1,1,5])
       N=len(observation_seq)
```

1 Compute conditional probability (H=Hot, C=Cold)

```
[165]: def chose_conditional(current_observation,current_state):
           if (current_observation==5 and current_state=='H'):
               P_Xnplus1byZ_nplus1=0.8
           elif (current_observation==5 and current_state=='C'):
               P_Xnplus1byZ_nplus1=0.3
           elif (current_observation==1 and current_state=='H'):
               P_Xnplus1byZ_nplus1=0.2
           elif (current_observation==1 and current_state=='C'):
               P_Xnplus1byZ_nplus1=0.7
           return P_Xnplus1byZ_nplus1
```

2 Initialize w(Zn) for 1st time instant

```
[166]: def omega_init(pi,observation_seq):
           return (np.log(pi[0])+np.log(chose_conditional(observation_seq[0],'H')),\
                   np.log(pi[1])+np.log(chose_conditional(observation_seq[0],'C')))
```

3 Viterbi Algorithm implementation

[167]:

```
omega_z=np.zeros((N,2))
def omega(observation_seq,omega_z,pi,A,chose_conditional):
    #define intial omega
    omega_z[0,:]=omega_init(pi,observation_seq)
    for i in range(1,N):
        omega_z[i,:]=(np.log(chose_conditional(observation_seq[i],'H'))+\
        max((np.log(A[0][0])+omega_z[(i-1),0]),(np.
 ↪log(A[1][0])+omega_z[(i-1),1])),\
        np.log(chose_conditional(observation_seq[i],'C'))+\
        max((np.log(A[0][1])+omega_z[(i-1),0]),(np.
 ↪log(A[1][1])+omega_z[(i-1),1])))
    return omega_z
```

4 Backtrack through trellis to find hidden variables

[168]:

```
def bactrack_trellis(omega_z):
    hidden_var_opposite=[]
    output=[]
    for j in reversed(range(N)):
        if omega_z[j,0]>omega_z[j,1]:
            hidden_var_opposite.append('H')
            #print('H ')
        else:
            hidden_var_opposite.append('C')
            #print('C ')
    for o in reversed(hidden_var_opposite):   #As backtracking will give Zn is␣
 ↪reverse order, so we need to reverse the sequence
        output.append(o)
        #print(output)
    return output
```

[169]:

```
omega_Znplus1=omega(observation_seq,omega_z,pi,A,chose_conditional)
omega_Znplus1
```

[169]:

```
array([[-0.4462871 , -2.81341072],
       [-1.0261056 , -2.85423271],
       [-1.60592409, -3.43405121],
       [-3.57203695, -3.16657184],
       [-5.53814981, -3.74639034],
       [-6.96526616, -4.32620883],
       [-7.54508466, -4.90602733],
       [-6.73860879, -6.33314368],
       [-7.31842729, -7.76026004],
        [-7.89824578, -9.18737639]])
```

5 Print the hidden variables

[170]:

```
latent=bactrack_trellis(omega_Znplus1)
print(latent)
```

```
['H', 'H', 'H', 'C', 'C', 'C', 'C', 'C', 'H', 'H']
```