# **HW8** Dishonest Casino

## As always start with importing what we need and finding our data locations.

```
import numpy as np
import matplotlib.pyplot as plt

import os

os.chdir("C:/Users/rique/Downloads/datasets")
```

## For part 1a) and b) We're using the hmm_pb1 dataset

## For part 2 we'll use the hmm_pb2 dataset

### load the data.

```
def load_dataset(probChar):

    if(probChar == 'a'):
        X = np.loadtxt("hmm_pb1.csv", delimiter=',')
    if(probChar == 'b'):
        X = np.loadtxt("hmm_pb2.csv", delimiter=',')

    pi = np.array([0.5,0.5])
    a = np.array([[0.95,0.05],[0.05,0.95]])
    b = np.array([[1/6,1/6,1/6,1/6,1/6,1/6],
[1/10,1/10,1/10,1/10,1/10,1/2]])

    return X, pi, a, b

def viterbi(X, pi, a, b):
    c = np.zeros([2, X.shape[0]])
    p1 = np.zeros([2, X.shape[0]-1])
    p2 = np.zeros(X.shape[0])

    for i in range(X.shape[0]):

        if (i == 0):
            c[:,i] = np.log(b[:, int(X[i]-1)]*pi)
        else:
            c[:,i] = np.log(b[:, int(X[i]-1)]) + np.max(np.log(a)
+c[:,i-1], axis=1)
            p1[:,i-1] = np.argmax(np.log(a)+c[:,i-1], axis=1)
```

```
    for j in range(p2.shape[0]-1, -1, -1):

        if (j == p2.shape[0]-1):
            p2[j] = np.argmax(c[:, X.shape[0]-1])
        else:
            p2[j] = p1[int(p2[j+1]), j]

    return p2

def forward(X, pi, a, b):
    alpha = np.zeros([2, X.shape[0]])

    for i in range(X.shape[0]):

        if (i == 0):
            alpha[:,i] = b[:, int(X[i])-1] * pi
        else:
            alpha[:,i] = b[:,int(X[i])-1] * np.matmul(alpha[:,i-1], a)
            alpha[:,i] = alpha[:,i] / np.sum(alpha[:,i])

    return alpha

def backward(X, a, b):
    beta = np.zeros([2, X.shape[0]])

    for i in range(X.shape[0]-1, -1, -1):

        if ( i == X.shape[0]-1):
            beta[:,i] = np.ones(2)
        else:
            beta[:,i] = np.matmul(beta[:,i+1]*b[:, int(X[i+1]-1)],
a.T)

    return beta

def Baum_Welch(X,a,b,alpha,beta):

    # E STEP
    k = np.zeros([alpha.shape[0]-1,2,2])
    gamma = np.zeros([alpha.shape[0],2])

    for i in range(alpha.shape[0]-1):
        dot = beta[i+1,:]*b[int(X[i+1])-1,:]
        dot2 =
np.matmul(np.expand_dims(alpha[i,:],axis=1),np.expand_dims(dot2,axis=1
).T)*a
        k[i,:] = (1*dot2)/(np.sum(dot2))
        gamma[i,:] =
(1*alpha[i,:]*beta[i,:])/np.sum(alpha[i,:]*beta[i,:])
        dot3 = np.sum(alpha[alpha.shape[0]-1,:]*beta[alpha.shape[0]-
```

```
1,:])

    gamma[alpha.shape[0]-1,:] = (1*alpha[alpha.shape[0]-
1,:]*beta[alpha.shape[0]-1,:])/dot3

    # M STEP

    pi = gamma[0.:]
    a = np.sum(k,axis=0)/(np.expand_dims(np.sum(gamma[:-
1,:],axis=0),axis=1))
    b = np.zeros([6,2])

    for i in range(b.shape[0]):
        dot = gamma*np.expand_dims(np.where(x==[i+1],1,0),axis=1)
        b[i,:] = (1*np.sum(dot,axis=0))/np.sum(gamma, axis=0)

    return pi, a, b
```

# part 1a)

## for y; 1 = fair result, and 2 = loaded/rigged result

```
X, pi, a, b = load_dataset('a')

print("y =", viterbi(X, pi, a, b)+1)
```

```
y = [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
2.
 2. 2. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2.
2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
```

```
2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
```

# part 1b)

## alpha and beta ratios. for t=0 -> 135

```
print("Alpha Ratio α1 135/α2 135 =", forward(X, pi, a, b)[0,134] /
forward(X, pi, a, b)[1,134])
print("Beta Ratio β1 135/β2 135 =", backward(X, a, b)[0,134] /
backward(X, a, b)[1,134])
```

```
Alpha Ratio α1 135/α2 135 = 1.3868337520932614
Beta Ratio β1 135/β2 135 = 0.8614311418284931
```

# part 2)

```
X, pi, a, b = load_dataset('b')

for i in range(1000):
    alpha = forward(X, pi, a, b)
    beta = backward(X, a, b)
    pi_, a_, b_, = Baum_Welch(X,a,b,alpha,beta)

print("π:",pi_)
print("a:",a_)
print("b:",b_)
```

```
π: [1.e-100 1.e+100]
a: [[0.70872775 0.29127225][0.01192257 0.98807743]]
b: [[0.09529734 0.20085006][0.11215521 0.20567704][0.07141696
0.1933959][0.04335233 0.20162279][0.58099717 0.10540076][0.09678099
0.09305344]]
```