Assignment Information

- Assignment: Homework 9
- Author: Charles Hoyt
- Due: July 1, 2016
- Language: Python 3
- Posted: https://github.com/cthoyt/notebooks/blob/master/bit/AbiHomework9.ipynb (https://github.com/cthoyt/notebooks/blob/master/bit/AbiHomework9.ipynb)

In [1]:

```
import sys
sys.version_info
```

Out[1]:

```
sys.version_info(major=3, minor=5, micro=1, releaselevel='final', se
rial=0)
```

Exercise 1

$A = \{a, b, c\}$

$\pi = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$

Row is start, column is end

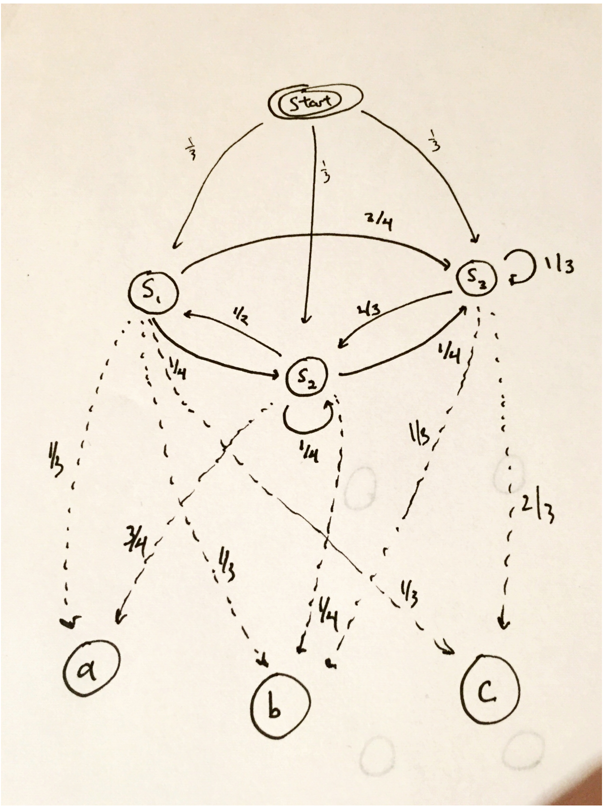
$$P = \begin{array}{cc} & \begin{array}{ccc} S_1 & S_2 & S_3 \end{array} \\ \begin{array}{c} S_1 \\ S_2 \\ S_3 \end{array} & \begin{pmatrix} 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \end{array}$$

and

$$B = \begin{array}{cc} & \begin{array}{ccc} a & b & c \end{array} \\ \begin{array}{c} S_1 \\ S_2 \\ S_3 \end{array} & \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} \end{pmatrix} \end{array}$$

Exercise 1A

Make a graphical representation of this HMM



Exercise 1B

What are the probability for observing sequences

A) $S_1 S_3 S_3 S_2$

$P(S_1 S_3 S_3 S_3) = P(Start \rightarrow S_1) \times P(S_1 \rightarrow S_3) \times P(S_3 \rightarrow S_3) \times P(S_3 \rightarrow S_2) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{3} \times \frac{2}{3} = \frac{1}{18}$

B) $S_2 S_1 S_3 S_3$

$P(S_2 S_1 S_3 S_3) = P(Start \rightarrow S_2) \times P(S_2 \rightarrow S_1) \times P(S_1 \rightarrow S_3) \times P(S_3 \rightarrow S_3) = \frac{1}{3} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{3} = \frac{1}{24}$

Exercise 2

Manually apply the Viterbi Algorithm for sequence $$

$$\delta_2(S_1) = \max \begin{cases} P(a \mid S_1) \times P(S_1 \mid S_1) \times \delta_1(S_1) = \frac{1}{3} \times 0 \times \frac{1}{3} = 0 \\ P(a \mid S_2) \times P(S_1 \mid S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{2} \times \frac{1}{3} = \frac{1}{8} = 0.125 \\ P(a \mid S_3) \times P(S_1 \mid S_3) \times \delta_1(S_3) = 0 \times 0 \times \frac{1}{3} = 0 \end{cases} = 0.125$$

$$\delta_2(S_2) = \max \begin{cases} P(a \mid S_1) \times P(S_2 \mid S_1) \times \delta_1(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{36} \approx 0.0278 \\ P(a \mid S_2) \times P(S_2 \mid S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{16} = 0.0625 \\ P(a \mid S_3) \times P(S_2 \mid S_3) \times \delta_1(S_3) = 0 \times \frac{2}{3} \times \frac{1}{3} = 0 \end{cases} = \frac{1}{16}$$

$$\delta_2(S_3) = \max \begin{cases} P(a \mid S_1) \times P(S_3 \mid S_1) \times \delta_1(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{3} = \frac{1}{12} \approx 0.08333 \\ P(a \mid S_2) \times P(S_3 \mid S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{16} = 0.0625 \\ P(a \mid S_3) \times P(S_3 \mid S_3) \times \delta_1(S_3) = 0 \times \frac{1}{3} \times \frac{1}{3} = 0 \end{cases} = \frac{1}{12}$$

$$\delta_3(S_1) = \max \begin{cases} P(b \mid S_1) \times P(S_1 \mid S_1) \times \delta_2(S_1) = \frac{1}{3} \times 0 \times \frac{1}{8} = 0 \\ P(b \mid S_2) \times P(S_1 \mid S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{2} \times \frac{1}{16} = \frac{1}{128} \approx 0.00787 \\ P(b \mid S_3) \times P(S_1 \mid S_3) \times \delta_2(S_3) = \frac{1}{3} \times 0 \times \frac{1}{12} = 0 \end{cases} = \frac{1}{128}$$

$$\delta_3(S_2) = \max \begin{cases} P(b \mid S_1) \times P(S_2 \mid S_1) \times \delta_2(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{8} = \frac{1}{96} \approx 0.0104 \\ P(b \mid S_2) \times P(S_2 \mid S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{16} = \frac{1}{256} = 0.00390625 \\ P(b \mid S_3) \times P(S_2 \mid S_3) \times \delta_2(S_3) = \frac{1}{3} \times \frac{2}{3} \times \frac{1}{12} = \frac{1}{54} \approx 0.01852 \end{cases} = \frac{1}{54}$$

$$\delta_3(S_3) = \max \begin{cases} P(b \mid S_1) \times P(S_3 \mid S_1) \times \delta_2(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{8} = \frac{1}{32} = 0.03125 \\ P(b \mid S_2) \times P(S_3 \mid S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{16} = \frac{1}{256} = 0.00390625 \\ P(b \mid S_3) \times P(S_3 \mid S_3) \times \delta_2(S_3) = \frac{1}{3} \times \frac{1}{3} \times \frac{1}{12} = \frac{1}{108} \approx 0.009259 \end{cases} = \frac{1}{32}$$

$$\delta_4(S_1) = \max\begin{cases} P(c \mid S_1) \times P(S_1 \mid S_1) \times \delta_3(S_1) = \frac{1}{3} \times 0 \times \frac{1}{128} = 0 \\ P(c \mid S_2) \times P(S_1 \mid S_2) \times \delta_3(S_2) = 0 \times \frac{1}{2} \times \frac{1}{54} = \frac{1}{108} \approx 0.01852 \\ P(c \mid S_3) \times P(S_1 \mid S_3) \times \delta_3(S_3) = \frac{1}{2} \times 0 \times \frac{1}{32} = 0 \end{cases} = \frac{1}{108}$$

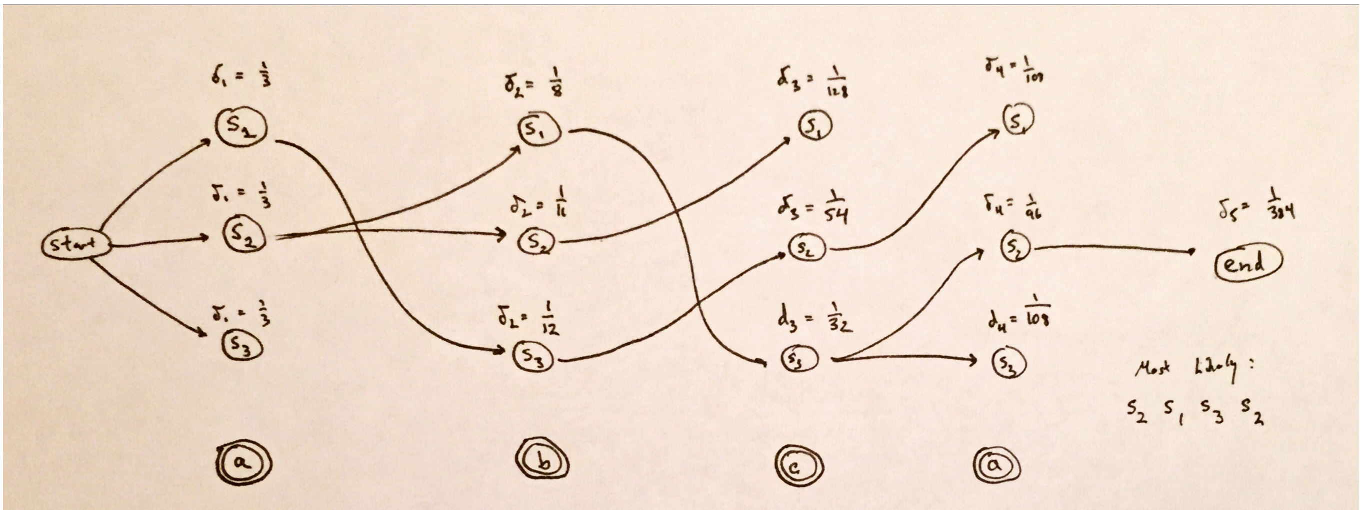
$$\delta_4(S_2) = \max\begin{cases} P(c \mid S_1) \times P(S_2 \mid S_1) \times \delta_3(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{128} = \frac{1}{1536} \approx 0.000651 \\ P(c \mid S_2) \times P(S_2 \mid S_2) \times \delta_3(S_2) = 0 \times \frac{1}{4} \times \frac{1}{54} = 0 \\ P(c \mid S_3) \times P(S_2 \mid S_3) \times \delta_3(S_3) = \frac{1}{2} \times \frac{2}{3} \times \frac{1}{32} = \frac{1}{96} \approx 0.0104 \end{cases} = \frac{1}{96}$$

$$\delta_4(S_3) = \max\begin{cases} P(c \mid S_1) \times P(S_3 \mid S_1) \times \delta_3(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{128} = \frac{1}{512} = 0.001953125 \\ P(c \mid S_2) \times P(S_3 \mid S_2) \times \delta_3(S_2) = 0 \times \frac{1}{4} \times \frac{1}{54} = 0 \\ P(c \mid S_3) \times P(S_3 \mid S_3) \times \delta_3(S_3) = \frac{1}{2} \times \frac{1}{3} \times \frac{1}{32} = \frac{1}{108} \approx 0.009259 \end{cases} = \frac{1}{108}$$

Take end transmissions to be:

$$\pi_{end} = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$$

$$\delta_5(End) = \max\begin{cases} P(a \mid S_1) \times P(End \mid S_1) \times \delta_4(S_1) = \frac{1}{3} \times \frac{1}{3} \times \frac{1}{108} = \frac{1}{972} = 0.00102880658436 \\ P(a \mid S_2) \times P(End \mid S_2) \times \delta_4(S_2) = \frac{3}{4} \times \frac{1}{3} \times \frac{1}{96} = \frac{1}{384} = 0.00260416666667 \\ P(a \mid S_3) \times P(End \mid S_3) \times \delta_4(S_3) = 0 \times \frac{1}{3} \times \frac{1}{108} = 0 \end{cases} =$$



Exercise 3

Implementation of the Viterbi Algorithm in Python3

In [2]:

```
import numpy as np
import pandas as pd
from numpy import array
from itertools import product
```

Starting parameters

In [3]:

```
n_states = 3
n_omissions = 4

pi =  array([1/3, 1/3, 1/3])
end = array([1/3, 1/3, 1/3])

# P[i,j] i is starting state, j is ending state
P = array([[0,   1/4, 3/4],
           [1/2, 1/4, 1/4],
           [0,   2/3, 1/3]])

# B[i,j] i is state, j is omission
B = array([[1/3, 1/3, 1/3],
           [3/4, 1/4, 0],
           [0,   1/3, 2/3]])

omission_chars = ['a', 'b', 'c']
omissions = array([0, 1, 2, 0])

labels = [omission_chars[i] for i in omissions] + ['End']
```

Naive implementation that runs into numeric issues because of small numbers

In [9]:

```
deltas      = np.empty(shape=(n_omissions+1, n_states))
breadcrumbs = np.empty(shape=(n_omissions+1, n_states), dtype=int)

# Initialization
for i in range(n_states):
    deltas[0, i] = pi[i]
    breadcrumbs[0, i] = -1

for j in range(n_omissions - 1):
    m = np.zeros(shape=(n_states, n_states))
    omission = omissions[j]

    # x is new delta, y is possibilities. argmax over each row
    for x, y in product(range(n_states), repeat=2):
        b = B[y, omission]
        p = P[y, x]
        d = deltas[j, y]
        m[x, y] = b * p * d

    print(j)
    print(m)

    win_values = np.max(m, axis=1)
    win_poses = np.argmax(m, axis=1)

    print(win_values) #in order of states
    print(win_poses)

    deltas[j + 1,:] = win_values
    breadcrumbs[j + 1, :] = win_poses

    print()

omission = omissions[-1]
last = []
for y in range(n_states):
    b = B[y, omission]
    p = end[y]
    d = deltas[n_omissions - 1, y]
    last.append(b * p * d)

last = array(last)

print(n_omissions - 1)
print(last)
deltas[n_omissions,:] = np.max(last)
breadcrumbs[n_omissions,:] = np.argmax(last)

pd.DataFrame(breadcrumbs, index=labels).T
```

```
0
[[ 0.          0.125       0.        ]
 [ 0.02777778  0.0625      0.        ]
 [ 0.08333333  0.0625      0.        ]]
[ 0.125       0.0625      0.08333333]
[1 1 0]

1
[[ 0.          0.0078125   0.        ]
 [ 0.01041667  0.00390625  0.01851852]
 [ 0.03125     0.00390625  0.00925926]]
[ 0.0078125   0.01851852  0.03125   ]
[1 2 0]

2
[[ 0.          0.          0.        ]
 [ 0.00065104  0.          0.01388889]
 [ 0.00195312  0.          0.00694444]]
[ 0.          0.01388889  0.00694444]
[0 2 2]

3
[ 0.          0.00347222  0.        ]
```

Out[9]:

	a	b	c	a	End
0	-1	1	1	0	1
1	-1	1	2	2	1
2	-1	0	0	2	1

In [5]:

```
pd.DataFrame(deltas, index=labels).T
```

Out[5]:

	a	b	c	a	End
0	0.333333	0.125000	0.007812	0.000000	0.003472
1	0.333333	0.062500	0.018519	0.013889	0.003472
2	0.333333	0.083333	0.031250	0.006944	0.003472

Convert to using logs and addition. Still has same results

In [8]:

```
deltas      = np.empty(shape=(n_omissions+1, n_states))
breadcrumbs = np.empty(shape=(n_omissions+1, n_states), dtype=int)

# Initialization
for i in range(n_states):
    deltas[0, i] = np.log(pi[i])
    breadcrumbs[0, i] = -1

for j in range(n_omissions - 1):
    m = np.zeros(shape=(n_states, n_states))
    omission = omissions[j]

    # x is new delta, y is possibilities. argmax over each row
    for x, y in product(range(n_states), repeat=2):
        b = np.log(B[y, omission])
        p = np.log(P[y, x])
        d = deltas[j, y]
        m[x, y] = b + p + d

    print(j)
    print(m)

    win_values = np.max(m, axis=1)
    win_poses = np.argmax(m, axis=1)

    print(win_values) #in order of states
    print(win_poses)

    deltas[j + 1,:] = win_values
    breadcrumbs[j + 1, :] = win_poses

    print()

omission = omissions[-1]
last = []
for y in range(n_states):
    b = np.log(B[y, omission])
    p = np.log(end[y])
    d = deltas[n_omissions - 1, y]
    last.append(b + p + d)

last = array(last)

print(n_omissions - 1)
print(last)
deltas[n_omissions,:] = np.max(last)
breadcrumbs[n_omissions,:] = np.argmax(last)

pd.DataFrame(breadcrumbs, index=labels).T
```

```
0
[[        -inf -2.07944154        -inf]
 [-3.58351894 -2.77258872        -inf]
 [-2.48490665 -2.77258872        -inf]]
[-2.07944154 -2.77258872 -2.48490665]
[1 1 0]

1
[[        -inf -4.85203026        -inf]
 [-4.56434819 -5.54517744 -3.98898405]
 [-3.4657359  -5.54517744 -4.68213123]]
[-4.85203026 -3.98898405 -3.4657359 ]
[1 2 0]

2
[[        -inf        -inf        -inf]
 [-7.33693691        -inf -4.27666612]
 [-6.23832463        -inf -4.9698133 ]]
[        -inf -4.27666612 -4.9698133 ]
[0 2 2]

3
[        -inf -5.66296048        -inf]
```

Out[8]:

	a	b	c	a	End
0	-1	1	1	0	1
1	-1	1	2	2	1
2	-1	0	0	2	1