

Assignment Information

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

In [14]:

```
!date  
import sys  
print(sys.version_info)
```

```
Fri Jul 1 00:14:10 CEST 2016  
sys.version_info(major=3, minor=5, micro=2, releaselevel='final', se  
rial=0)
```

Exercise 1

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

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

Row is start, column is end

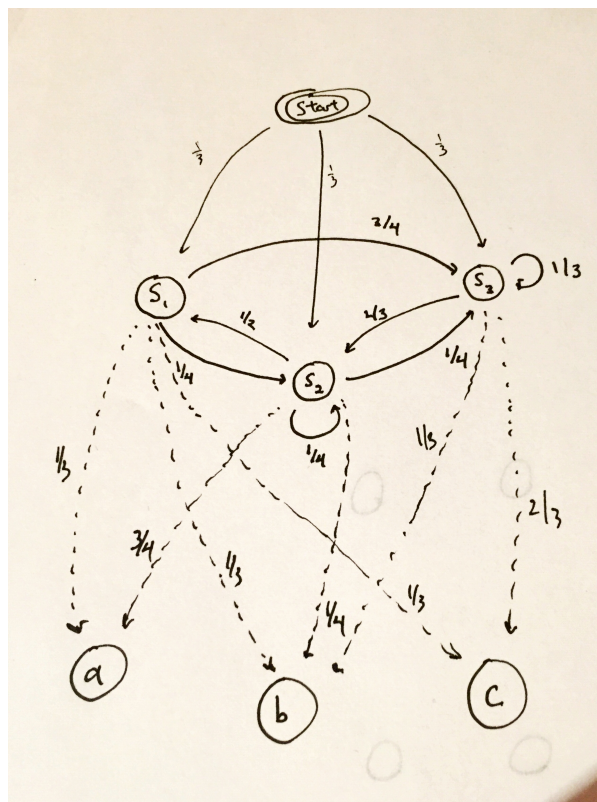
$$P = \begin{pmatrix} & S_1 & S_2 & S_3 \\ S_1 & 0 & \frac{1}{4} & \frac{3}{4} \\ S_2 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ S_3 & 0 & \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

and

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

Exercise 1A

Make a graphical representation of this HMM



Exercise 1B

What are the probability for observing sequences for emission $[a, b, c, b]$:

A) $S_1 S_3 S_3 S_2$

$$P(\text{Start} \rightarrow S_1) \times P(a | S_1) \times P(S_1 \rightarrow S_3) \times P(b | S_3) \times P(S_3 \rightarrow S_3) \times P(c | S_3) \times P(S_3 \rightarrow S_2) \times P(b | S_2) \\ = \frac{12}{2916} = \frac{1}{243}$$

B) $S_2 S_1 S_3 S_3$

$$P(\text{Start} \rightarrow S_2) \times P(a | S_2) \times P(S_2 \rightarrow S_1) \times P(b | S_1) \times P(S_1 \rightarrow S_3) \times P(c | S_3) \times P(S_3 \rightarrow S_3) \times P(b | S_3) \\ = \frac{18}{7776} = \frac{1}{432}$$

Exercise 2

Manually apply the Viterbi Algorithm for sequence \$\$

$$\begin{aligned} \delta_2(S_1) &= \max \begin{cases} P(a | S_1) \times P(S_1 | S_1) \times \delta_1(S_1) = \frac{1}{3} \times 0 \times \frac{1}{3} = 0 \\ P(a | S_2) \times P(S_1 | S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{2} \times \frac{1}{3} = \frac{1}{8} = 0.125 \\ P(a | S_3) \times P(S_1 | S_3) \times \delta_1(S_3) = 0 \times 0 \times \frac{1}{3} = 0 \end{cases} = 0.125 \\ \delta_2(S_2) &= \max \begin{cases} P(a | S_1) \times P(S_2 | S_1) \times \delta_1(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{36} \approx 0.0278 \\ P(a | S_2) \times P(S_2 | S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{16} = 0.0625 \\ P(a | S_3) \times P(S_2 | S_3) \times \delta_1(S_3) = 0 \times \frac{2}{3} \times \frac{1}{3} = 0 \end{cases} = \frac{1}{16} \\ \delta_2(S_3) &= \max \begin{cases} P(a | S_1) \times P(S_3 | S_1) \times \delta_1(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{3} = \frac{1}{12} \approx 0.08333 \\ P(a | S_2) \times P(S_3 | S_2) \times \delta_1(S_2) = \frac{3}{4} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{16} = 0.0625 \\ P(a | S_3) \times P(S_3 | S_3) \times \delta_1(S_3) = 0 \times \frac{1}{3} \times \frac{1}{3} = 0 \end{cases} = \frac{1}{12} \end{aligned}$$

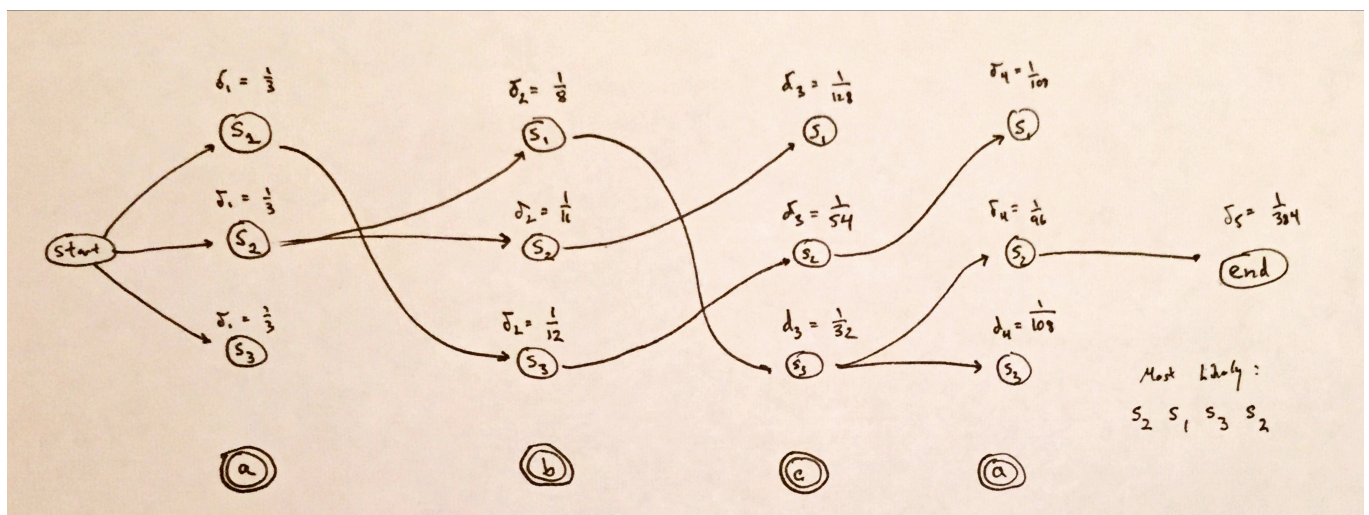
$$\begin{aligned} \delta_3(S_1) &= \max \begin{cases} P(b | S_1) \times P(S_1 | S_1) \times \delta_2(S_1) = \frac{1}{3} \times 0 \times \frac{1}{8} = 0 \\ P(b | S_2) \times P(S_1 | S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{2} \times \frac{1}{16} = \frac{1}{128} \approx 0.00787 \\ P(b | S_3) \times P(S_1 | S_3) \times \delta_2(S_3) = \frac{1}{3} \times 0 \times \frac{1}{12} = 0 \end{cases} = \frac{1}{128} \\ \delta_3(S_2) &= \max \begin{cases} P(b | S_1) \times P(S_2 | S_1) \times \delta_2(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{8} = \frac{1}{96} \approx 0.0104 \\ P(b | S_2) \times P(S_2 | S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{16} = \frac{1}{256} = 0.00390625 \\ P(b | S_3) \times P(S_2 | S_3) \times \delta_2(S_3) = \frac{1}{3} \times \frac{2}{3} \times \frac{1}{12} = \frac{1}{54} \approx 0.01852 \end{cases} = \frac{1}{54} \\ \delta_3(S_3) &= \max \begin{cases} P(b | S_1) \times P(S_3 | S_1) \times \delta_2(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{8} = \frac{1}{32} = 0.03125 \\ P(b | S_2) \times P(S_3 | S_2) \times \delta_2(S_2) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{16} = \frac{1}{256} = 0.00390625 \\ P(b | S_3) \times P(S_3 | S_3) \times \delta_2(S_3) = \frac{1}{3} \times \frac{1}{3} \times \frac{1}{12} = \frac{1}{108} \approx 0.009259 \end{cases} = \frac{1}{32} \end{aligned}$$

$$\begin{aligned}
\delta_4(S_1) &= \max \begin{cases} P(c | S_1) \times P(S_1 | S_1) \times \delta_3(S_1) = \frac{1}{3} \times 0 \times \frac{1}{128} = 0 \\ P(c | S_2) \times P(S_1 | S_2) \times \delta_3(S_2) = 0 \times \frac{1}{2} \times \frac{1}{54} = \frac{1}{108} \approx 0.01852 \\ P(c | S_3) \times P(S_1 | S_3) \times \delta_3(S_3) = \frac{1}{2} \times 0 \times \frac{1}{32} = 0 \end{cases} = \frac{1}{108} \\
\delta_4(S_2) &= \max \begin{cases} P(c | S_1) \times P(S_2 | S_1) \times \delta_3(S_1) = \frac{1}{3} \times \frac{1}{4} \times \frac{1}{128} = \frac{1}{1536} \approx 0.000651 \\ P(c | S_2) \times P(S_2 | S_2) \times \delta_3(S_2) = 0 \times \frac{1}{4} \times \frac{1}{54} = 0 \\ P(c | S_3) \times P(S_2 | S_3) \times \delta_3(S_3) = \frac{1}{2} \times \frac{2}{3} \times \frac{1}{32} = \frac{1}{96} \approx 0.0104 \end{cases} = \frac{1}{96} \\
\delta_4(S_3) &= \max \begin{cases} P(c | S_1) \times P(S_3 | S_1) \times \delta_3(S_1) = \frac{1}{3} \times \frac{3}{4} \times \frac{1}{128} = \frac{1}{512} \approx 0.001953125 \\ P(c | S_2) \times P(S_3 | S_2) \times \delta_3(S_2) = 0 \times \frac{1}{4} \times \frac{1}{54} = 0 \\ P(c | S_3) \times P(S_3 | S_3) \times \delta_3(S_3) = \frac{1}{2} \times \frac{1}{3} \times \frac{1}{32} = \frac{1}{192} \approx 0.005208333 \end{cases} = \frac{1}{192}
\end{aligned}$$

Take end transmissions to be:

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

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



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]:

```
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 = 'abc'
omissions = array([0, 1, 2, 1, 0])

n_states = 3
n_omissions = len(omissions)

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

Naive implementation that runs into numeric issues because of small numbers

In [4]:

```

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.          0.00173611  0.          ]
 [ 0.          0.00086806  0.00154321]
 [ 0.          0.00086806  0.0007716  ]]
[ 0.00173611  0.00154321  0.00086806]
[1 2 1]

4
[ 0.0001929  0.0003858  0.          ]
```

Out[4]:

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

In [5]:

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

Out[5]:

	a	b	c	b	a	End
0	0.333333	0.125000	0.007812	0.000000	0.001736	0.000386
1	0.333333	0.062500	0.018519	0.013889	0.001543	0.000386
2	0.333333	0.083333	0.031250	0.006944	0.000868	0.000386

Convert to using logs and addition. Still has same results

In [6]:

```

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 -6.35610766          -inf]
 [          -inf -7.04925484 -6.4738907 ]
 [          -inf -7.04925484 -7.16703788]]
[-6.35610766 -6.4738907  -7.04925484]
[1 2 1]
```

```
4
[-8.55333224 -7.86018506          -inf]
```

Out[6]:

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

In [7]:

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

Out[7]:

	a	b	c	b	a	End
0	-1.098612	-2.079442	-4.852030	-inf	-6.356108	-7.860185
1	-1.098612	-2.772589	-3.988984	-4.276666	-6.473891	-7.860185
2	-1.098612	-2.484907	-3.465736	-4.969813	-7.049255	-7.860185