Homework1

September 1, 2015

0.1 Problem 1

The transition matrix is given by:

$$\begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

0.1.1 Part (a)

$\eta = min(n > 0, X_n = 1)$ given $X_0 = 0$

To Prove $\eta \sim Geom(\alpha)$

$\eta = P(X_0 = 0, X_1 = 0, \ldots X_{n-1} = 1, X_n = 1)$

Using the Markov property this can be written as:

$$\eta = P(X_0 = 0)P(X_1 = 0|X_0 = 0)P(X_2 = 0|X_1 = 0)P(X_3 = 0|X_2 = 0)\ldots P(X_{n-1} = 0|X_{n-2} = 0)P(X_n = 1|X_{n-1} = 0)$$

And being time-homogenous, this simplifies to:

$$\eta = P(X_0 = 0)\big(P(X_1 = 0|X_0)\big)^{n-1} \times P(X_1 = 1|X_0 = 0)$$

$\implies$

$$\eta = P(X_0 = 0)\big(1-\alpha\big)^{n-1}\alpha = \big(1-\alpha\big)^{n-1}\alpha$$

And hence $\eta \sim Geom(\alpha)$

0.1.2 Part (b)

Spectral decomposition of P and value for $P(X_n = 1|X_0 = 0)$

Spectral decomposition of P:

$$det \begin{bmatrix} \alpha-\lambda & 1-\alpha \\ 1-\beta & \beta-\lambda \end{bmatrix} = 0$$

$$\lambda^2 + (\alpha+\beta-2)\lambda + (1-\alpha-\beta) = 0$$

Thus, $\lambda_1 = 1$ and $\lambda_2 = 1-\alpha-\beta$

Eigenvectors are given by:

$v_1^T = \begin{pmatrix} x_1 & x_1 \end{pmatrix} \ \forall \ x_1 \in R$

and for λ_2 , $v_2 = \begin{pmatrix} x_1 & \frac{-\beta x_1}{\alpha} \end{pmatrix}$

Now using Markov property: $P(X_n = 1|X_0 = 0) = (P^n)_{01}$

Now,

$P^n = VD^nV^{-1}$

where:

$$V = \begin{bmatrix} 1 & 1 \\ 1 & \frac{-\beta}{\alpha} \end{bmatrix}$$

and

$$D = \begin{bmatrix} 1 & 0 \\ 0 & (1-\alpha-\beta) \end{bmatrix}$$

$$V^{-1} = \frac{-1}{\frac{\beta}{\alpha}+1} \begin{bmatrix} -\frac{\beta}{\alpha} & -1 \\ -1 & 1 \end{bmatrix}$$

Thus,

$$P^n = \begin{bmatrix} 1 & 1 \\ 1 & \frac{-\beta}{\alpha} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & (1-\alpha-\beta)^n \end{bmatrix} \times \frac{-1}{\frac{\beta}{\alpha}+1} \begin{bmatrix} -\frac{\beta}{\alpha} & -1 \\ -1 & 1 \end{bmatrix}$$

$$P^n = \frac{1}{\alpha+\beta} \begin{bmatrix} \beta + \alpha(1-\alpha-\beta)^n & \alpha - \alpha(1-\alpha-\beta)^n \\ \beta - \beta(1-\alpha-\beta)^n & \alpha + \beta(1-\alpha-\beta)^n \end{bmatrix}$$

0.2 Part (c)

When $\alpha + \beta = 1$, the eigen values are $\lambda_1 = 1$ and $\lambda_2 = 0$ and hence

$$P^n = \begin{bmatrix} \beta & \alpha \\ \beta & \alpha \end{bmatrix}$$

Check:
Also consider the following identifiy: $P^{n+1} = PP^n$
then:

$$\begin{bmatrix} p_{00}^{n+1} & p_{01}^{n+1} \\ p_{10}^{n+1} & p_{11}^{n+1} \end{bmatrix} = \begin{bmatrix} p_{00}^{n} & p_{01}^{n} \\ p_{10}^{n} & p_{11}^{n} \end{bmatrix} \times \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

$\implies$

$$\begin{aligned} p_{11}^{n+1} &= p_{10}^{n}(\alpha) + p_{11}^{n}(1-\beta) \\ &= (1-p_{11}^{n})(\alpha) + (p_{11}^{n})(1-\beta) \\ &= \alpha + (1-\alpha-\beta)p_{11}^{n} \end{aligned}$$

Consider the recurrence:

$$x_{n+1} = \alpha + (1-\alpha-\beta)x_n$$

Constant solution $x_n = x_{n+1} = x$ is given by: $x = \frac{\alpha}{\alpha+\beta}$
Now let $y_n = x_n - x = x_n - \frac{\alpha}{\alpha+\beta}$ then,
$y_{n+1} = (1-\alpha-\beta)y_n$ and hence $y_n = (1-\alpha-\beta)^n y_0$
Thus,

$$p_{11}^{n} = (1-\alpha-\beta)^n p_{11}^{0} + \frac{\alpha}{\alpha+\beta}$$

Given $P_{00} = \frac{\beta}{\alpha+\beta}$ and $\alpha+\beta = 1$ and hence:
$p_{11}^{n} = \frac{\alpha}{\alpha+\beta} = \alpha$
and hence, $p_{10}^{n} = \beta$
Similary,
$p_{00}^{n} = \beta$ and $p_{01}^{n} = \alpha$

0.3 Problem 2

$P(X_1 = 0)\frac{\beta}{\alpha+\beta}$ and hence $P(X_1 = 1) = \frac{\alpha}{\alpha+\beta}$
$X = X_1X_2\dots X_n$ and $Y = Y_1Y_2\dots Y_n$ representes the reverse string $Y_k = X_{n+k-1}$

0.3.1 Part (a)

Given string of digits: $a_1, a_2, a_3 \dots a_n$ to find: $P(Y_1 = a_1, Y_2 = a_2, Y_3 = a_3 \dots Y_n = a_n)$

$$\begin{aligned} P(Y_1 = a_1, Y_2 = a_2, Y_3 = a_3 \dots Y_n = a_n) &= P(X_1 = a_n, X_2 = a_{n-1}, \dots X_n = a_1) \\ &= P(X_1 = a_n)P(X_2 = a_{n-1}|X_1 = a_n)P(X_3 = a_{n-2}|X_2 = a_{n-1}) \dots P(X_n = a_1|X_{n- \\ &= P(X_1 = a_n)(P_{a_n a_{n-1}})(P_{a_{n-1} a_{n-2}}) \dots (P_{a_2 a_1}) \end{aligned}$$

The problem asked about not using spectral decomposition, but I was not sure how spectral decomposition would have come in handy if the states a_i are not specified explicitly.

0.3.2 Part (b)

$$z = \begin{cases} X & if \theta = H \\ Y & otherwise \end{cases}$$

Given function f such that, $f : \{0, 1\}^n \longrightarrow \{H, T\}$ To show: $P(f(Z) = \theta) = 0.5$
$P(\theta = H) = P(\theta = T) = 0.5$
Given Z, guess θ:
$P(\theta = H|Z = X) = \frac{P(\theta=H,Z=X)}{P(Z=X)}$
Z, has only two possible values: H and T and hence assuming the guess function is unbiased:
$P(f(Z) = H) = P(f(Z) = T) = 0.5$

0.4 Problem 3

$$\tau = min\{n \geq 0 : X_n = \dagger\}$$

$$E[\tau] = E_a[E_a[\tau|X_n = a] \ where \ a \in \{\phi, \alpha, \beta, \alpha + \beta, pol, \dagger\}$$

Let $S = \{\phi, \alpha, \beta, \alpha + \beta, pol, \dagger\}$
Consider for $a \neq \dagger$:

$$h(a) = E[\tau|X_0 = a] = \sum_{s \in S} P_{as} \times (1) + P_{as} \times E[\tau|X_0 = s)$$

$$\implies$$

$$h(a) = ((I - P_{\_})^{-1})_a$$

where $P_{\_}$ represents the matrix with the row and column representng $X_i = \dagger$ removed.

```
In [59]: from __future__ import division
         import numpy as np
         k_a=0.2
         k_b=0.2
         k_p = 0.5
         P = np.array([[1-k_a-k_b,k_a,k_b,0,0,0],[k_a,1-k_a-k_b,0,k_b,0,0],[k_b,0,1-k_a-k_b,0,0],[0,k_b
         q = [[k_a-k_b,k_a,k_b,0,0],[k_a,k_a+k_b,0,k_b,0],[k_b,0,k_a+k_b,0,0],[0,k_b,k_a,k_a+k_b+k_p,k_
         qq = np.array(q)
```

```
In [60]: iq = np.linalg.inv(np.eye(5)-qq)
```

Define, $h(a) = E[\tau|X_0 = a]$ then $h(a) = \sum_b p_{ab}(1 + E[\tau|x_0 = b]) \implies$ d

```
In [61]: iq_phi = iq[0,0]
         print(iq_phi)
```

```
1.66666666667
```

```
In [62]: iq_alpha = iq[1,1]
         print(iq_alpha)
```

```
7.77777777778
```

```
In [63]: iq_beta = iq[2,2]
         print(iq_beta)
```

```
2.22222222222
```

```
In [64]: iq_alphabeta = iq[3,3]
         print(iq_alphabeta)
```

```
43.3333333333
```

```
In [65]: iq_pol = iq[4,4]
         print(iq_pol)
```

```
1.0
```

```
In [66]: a=[[k_a+k_b,-k_a,-k_b,0],[k_a,-k_a-k_b,0,k_b],[k_b,0,-k_a-k_b,k_a],[1,1,1,3]]
         b=[0,0,0,1]
         x=np.linalg.solve(a,b)
         print(x)
```

```
[ 0.16666667  0.16666667  0.16666667  0.16666667]
```

Stationary state is given by $\pi = (0.1667, 0.1667, 0.1667, 0.1667, 0.1667, 0.1667)$ We solve only for $\pi_i, \pi_2, \pi_3, \pi_4$ as $\pi_4 = \pi_5 = \pi_6$ and the mean number of visits per unit time to † are $\frac{1}{\pi_6} = 6$

0.5 Part (c)

Simulating the chain:

General strategy: Generate a random number $\longrightarrow$ Select a state $\longrightarrow$ Jump to state $\longrightarrow$ Repeat

```
In [67]: ## phi
         np.random.seed(1)
         a=np.random.uniform(high=1.4)
```

```
In [68]: print(a)

         P = {}
         P['phi']= [1-k_a-k_b,k_a,k_b,0,0,0]
         P['alpha'] = [k_a,1-k_a-k_b,0,k_b,0,0]
         P['beta'] = [k_b,0,1-k_a-k_b,0,0]
         P['ab']= [0,k_b,k_a,1-k_a-k_b-k_p,k_p,0]
         P['pol']= [0,0,0,0,0,1]
         P['d']= [0,0,0,1,0,0]

         states = ['phi', 'alpha', 'beta', 'ab', 'pol', 'd']
         def accumulate(lis):
             total = 0
             for x in lis:
                 total += x
                 yield total
```

```
        C= {}
        for key,value in P.iteritems():
            C[key] = list(accumulate(value))

        print(C)
```

```
0.583830806584
{'phi': [0.6000000000000001, 0.8, 1.0, 1.0, 1.0, 1.0], 'ab': [0, 0.2, 0.4, 0.5000000000000001, 1.0, 1.0
```

In [86]:

```
##For £h(\phi)£
x0='phi'
x='phi'
def h(x):
    s=0
    for i in range(1,10000):
        a = np.random.uniform()
        probs = P[x]
        for i in range(0,len(probs)):
            if i==0 and a<probs[0]:
                ## No state change
                continue
            if i<len(probs)-1:
                if int(probs[i])==1 and int(probs[i+1])==1:
                    continue
                if a>=probs[i] and a<probs[i+1]:
                    x=states[i+1]

            if a<=probs[i]:
                x=states[i]


        index = states.index(x)+1
        #print index
        s+=index
    return s/10000
```

```
2.9997
```

In [90]:

```
print(r'$h(\phi)$: From calculation: {}; From simulation: {}'.format(h('phi'),iq_phi))
```

```
$h(\phi)$: From calculation: 2.9987; From simulation: 1.66666666667
```

In [91]:

```
print(r'$h(\alpha)$: From calculation: {}; From simulation: {}'.format(h('alpha'),iq_alpha))
```

```
$h(\alpha)$: From calculation: 4.7572; From simulation: 7.77777777778
```

In [92]:

```
print(r'$h(\beta)$: From calculation: {}; From simulation: {}'.format(h('beta'),iq_beta))
```

```
$h(\beta)$: From calculation: 2.9997; From simulation: 2.22222222222
```

In [94]:

```
print(r'$h(\alpha+\beta)$: From calculation: {}; From simulation: {}'.format(h('ab'),iq_alphab
```

```
$h(\alpha+\beta)$: From calculation: 4.7488; From simulation: 43.3333333333
```

In [95]:

```
print(r'$h(\pol)$: From calculation: {}; From simulation: {}'.format(h('pol'),iq_pol))
```

```
$h(\pol)$: From calculation: 4.7432; From simulation: 1.0
```

In [ ]: