EE381 Project 7

Alexander Fielding

Problem 1

Introduction

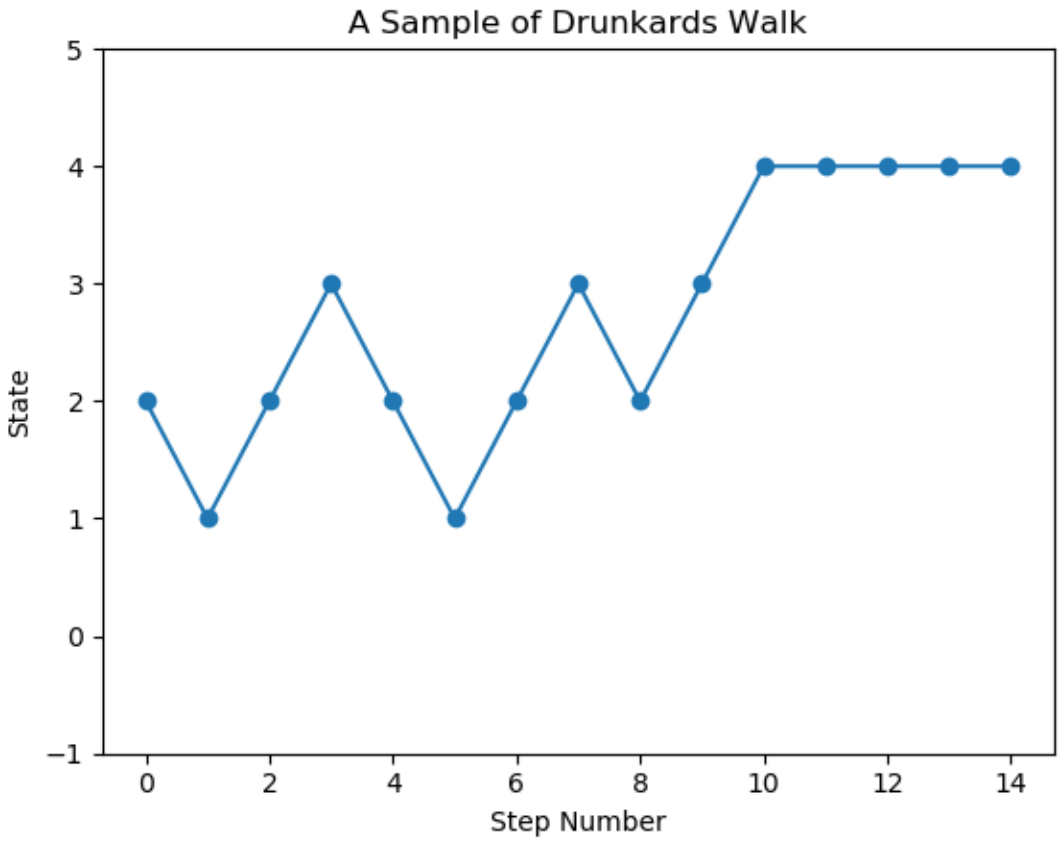
The purpose of the problem is to simulate both possible conclusions of an absorbing Markov Chain also known as the Drunkard's Walk or Random Walk problem.

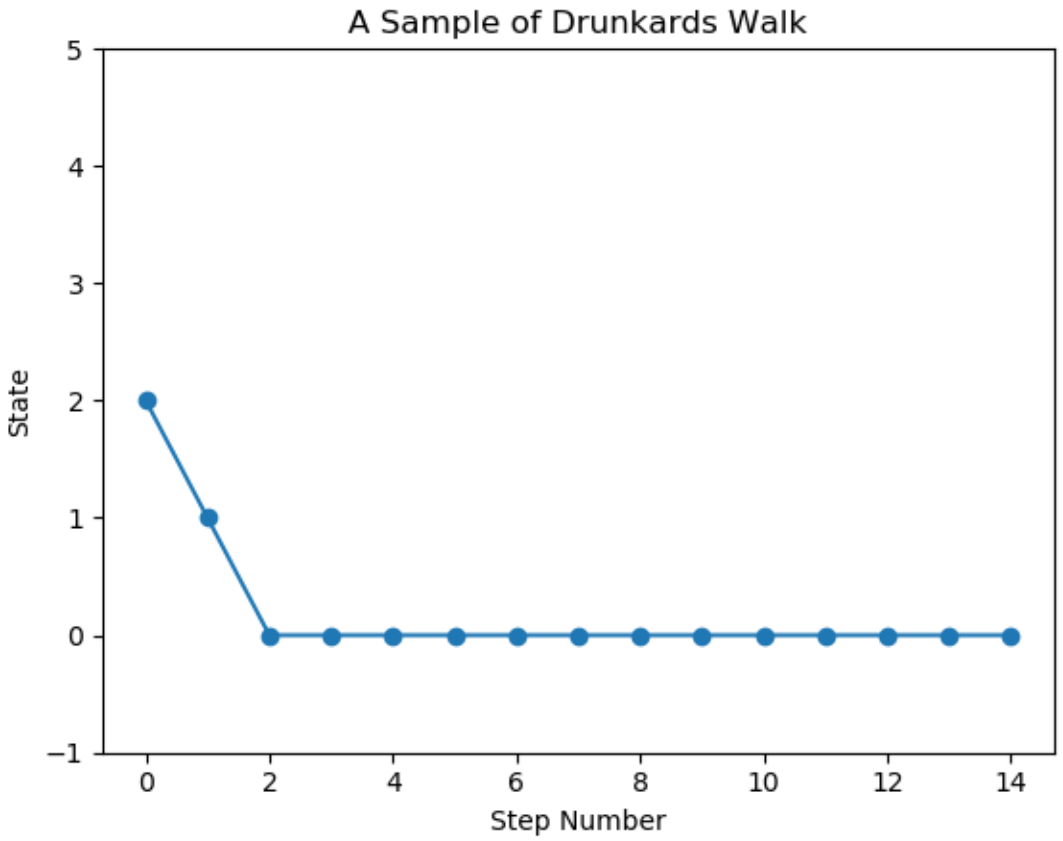
Methodology

To implement this simulation, we use the numpy library's choice function to handle a uniform random choice as a starting state. Then using the various probabilities for each transition state from the state transition matrix and the choice function the next state is chosen. This process is repeated 15 times.

Results

NOTE: 5 and -1 are **not** states and are shown for visualization/spacing.





Appendix

```
#EE381 Alexander Fielding
# Project 7 Drunken Random Walk
# problem 1

import numpy as np
from matplotlib import pylab as pyplt
from matplotlib.legend_handler import HandlerLine2D

#p =
np.matrix([1,0.3,0,0,0],[0,0,0.5,0,0],[0,0.7,0,0.6,0],[0,0,0.5,0,0],[0,0,0,0.
4,1])
statezero = [1,0,0,0,0]
stateone = [0.3,0,0.7,0,0]
statetwo = [0,0.5,0,0.5,0]
statethree = [0,0,0.6,0,0.4]
statefour = [0,0,0,0,1]

states = ['0','1','2','3','4']

n = 15 #number of steps

def initialstate():
    return np.random.choice(states,1)
```

```
def nextstate(sample,probs):
    return np.random.choice(sample,1,p = probs)

#Main
statelist = []
visited = []
current = initialstate()
print(current)
for i in range(n):
    if current == '0':
        current = nextstate(states,statezero)
        statelist.append(0)
    elif current == '1':
        current = nextstate(states,stateone)
        statelist.append(1)
    elif current == '2':
        current = nextstate(states,statetwo)
        statelist.append(2)
    elif current == '3':
        current = nextstate(states,statethree)
        statelist.append(3)
    elif current == '4':
        current = nextstate(states,statefour)
        statelist.append(4)
    print(current)
    visited.append(current)
pyplt.top = 0.9
axes = pyplt.gca()
axes.set_ylim([-1,5])
pyplt.plot([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14], statelist,'-o')
pyplt.title('A Sample of Drunkards Walk')
pyplt.ylabel('State')
pyplt.xlabel('Step Number')
pyplt.show()
```

Problem 2

Introduction

Now we continue to illustrate the Drunkards walk, specifically to determine the absorption probabilities of the two absorbing states.

Methodology

The same choice function is utilized from the numpy library, now altered to ensure state two is always the initial state. Each time an absorbing state is reached we record the count of that absorbing state and then restart the simulation. The simulation is repeated 10,00 times to achieve an accurate average for the probabilities.

Results

Absorption Probabilities (via simulations)			
b_{20}	39.6%	b_{24}	55.4%

Appendix

```
#EE381 Alexander Fielding
# Project 7 Drunken Random Walk
# problem 2

import numpy as np
from matplotlib import pylab as pyplt
from matplotlib.legend_handler import HandlerLine2D

#p =
np.matrix([1,0.3,0,0,0],[0,0,0.5,0,0],[0,0.7,0,0.6,0],[0,0,0.5,0,0],[0,0,0,0.
4,1])
statezero = [1,0,0,0,0]
stateone = [0.3,0,0.7,0,0]
statetwo = [0,0.5,0,0.5,0]
statethree = [0,0,0.6,0,0.4]
statefour = [0,0,0,0,1]

states = ['0','1','2','3','4']

initialprobs = [0,0,1,0,0] # intial state alwasy 2

n = 15 #number of steps
N = 10000 # number of simulations

def initialstate():
```

```
    return np.random.choice(states,1, p = initialprobs)

def nextstate(sample,probs):
    return np.random.choice(sample,1,p = probs)

#Main
zerocount = 0
fourcount = 0
for j in range(N):
    current = initialstate()
    for i in range(n):
        if current == '0':
            current = nextstate(states,statezero)
        elif current == '1':
            current = nextstate(states,stateone)
        elif current == '2':
            current = nextstate(states,statetwo)
        elif current == '3':
            current = nextstate(states,statethree)
        elif current == '4':
            current = nextstate(states,statefour)
    if current == '0':
        zerocount = zerocount + 1
    elif current == '4':
        fourcount = fourcount + 1

print("ending in zero:",zerocount)
print("ending in four:",fourcount)
```

Problem 3

Introduction

In this problem the goal is to further pour understanding of using state 2 as the initial state by using repeated simulatons to determine an average time of absorption to occur from starting at state 2.

Methodology

To accomplish this simulation, we utilize the same procedure in problem 2 and as soon as an absorbing state is reached the simulation records the number of steps necessary and repeats the process. We then determine the absorption time.

Results

Time to absorption for state 2 (via simulations)	
t_2	5.74

Appendix

```
#EE381 Alexander Fielding
# Project 7 Drunken Random Walk
# problem 3

import numpy as np
from matplotlib import pylab as pyplt
from matplotlib.legend_handler import HandlerLine2D

#p =
np.matrix([1,0.3,0,0,0],[0,0,0.5,0,0],[0,0.7,0,0.6,0],[0,0,0.5,0,0],[0,0,0,0.
4,1])
statezero = [1,0,0,0,0]
stateone = [0.3,0,0.7,0,0]
statetwo = [0,0.5,0,0.5,0]
statethree = [0,0,0.6,0,0.4]
statefour = [0,0,0,0,1]

states = ['0','1','2','3','4']

initialprobs = [0,0,1,0,0] # intial state alwasy 2

n = 80 #number of steps
N = 10000 # number of simulations

def initialstate():
    return np.random.choice(states,1, p = initialprobs)

def nextstate(sample,probs):
    return np.random.choice(sample,1,p = probs)

#Main
count = 0
numsteps = 0
for j in range(N):
    i = 0
    current = initialstate()
    while i < 80:
        if current == '0':
            current = nextstate(states,statezero)
            count = count + 1
            i = 80
```

```
        elif current == '1':
            current = nextstate(states,stateone)
        elif current == '2':
            current = nextstate(states,statetwo)
        elif current == '3':
            current = nextstate(states,statethree)
        elif current == '4':
            current = nextstate(states,statefour)
            count = count + 1
            i = 80
        numsteps = numsteps + 1

print(count/numsteps)
```

Problem 4

Introduction

To check the accuracy of the results of the simulation we can calculate the theoretical values using the canonical and fundamental matrices.

Methodology

To accomplish this, we refactor the transition matrix into the conical format and then find Q sub matrix from the transitive states. We subtract Q by the identity matrix and divide by the determinant to find the fundamental matrix. We multiply the fundamental matrix by a 1 filled column vector to find the time of absorption and multiple the fundamental by the R submatrix from the canonical form to find the absorption possibilities.

Results

Absorption Probabilities (via Fundamental Matrix)			
b_{20}	0.6	b_{24}	0.684

Time to absorption for state 2 (via Fundamental Matrix)	
t_2	6.57

Appendix

$$Q = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.7 & 0 & 0.6 \\ 0 & 0.5 & 0 \end{bmatrix}$$

$$N = \begin{bmatrix} 2 & 1.43 & 0.857 \\ 2 & 2.86 & 1.71 \\ 1 & 1.43 & 1.857 \end{bmatrix}$$

$$N*c = \begin{bmatrix} 4.287 \\ 6.57 \\ 4.287 \end{bmatrix}$$

$$N * R = \begin{bmatrix} 0.6 & 0.343 \\ 0.6 & 0.684 \\ 0.3 & 0.3 \end{bmatrix}$$