Francisco Fierro

EE380

Lab 6

1a.) By simulation, the probability that Thursday is a medium day is 0.41233. Mathematically, that probability is 0.412

import numpy as np

P = np.array([[.5, .3 , .2], [.2, .6, .2], [.3, .2, .5]])

trials = 100000

mediumDays= 0

rolls = np.random.rand(trials,3)

ThursdayState = np.zeros(trials)

for i in range(trials):

state = 1

for j in range(3):

sum = 0

for k in range(3):

sum = sum + P[state, k]

#print(rolls[i,j])

#print(P[state,k])

if(rolls[i,j] <= sum):

state = k

#print(state)

break

#print(state)

#print(state)

if(state == 1):

mediumDays = mediumDays + 1

ThursdayState[i] = state

print(" By simulation, P(Thursday is a medium day) = " + str( mediumDays / trials))

MediumDay = np.array([0,1,0])

P3 = np.copy(P)

for i in range(2):

P3 = np.dot(P3,P)

Thursday = np.dot(MediumDay, P3)

print(" By math, P(Thursday is a medium day) = " + str( Thursday[1]))

1b.) By simulation, the expected number of good days is 2.28339. Mathematically, the answer is 2.0612.

import numpy as np

trials = 100000

P = np.array([[.5, .3 , .2], [.2, .6, .2], [.3, .2, .5]])

rolls = np.random.rand(trials,7)

numGoodDaysA = np.zeros(trials)

for i in range(trials):

state = 2

goodDays = 0

for j in range(7):

sum = 0

for k in range(3):

sum = sum + P[state, k]

if(rolls[i,j] <= sum):

state = k

break

if(state == 0):

goodDays = goodDays + 1

numGoodDaysA[i] = goodDays

mean = np.mean(numGoodDaysA)

print(" By simulation, the expected number of good days = " + str(mean))

P0 = np.array([[0, .3 , .2], [0, .6, .2], [0, .2, .5]])

I = np.identity(3)

diff = np.subtract(I,P0)

inverse = np.linalg.inv(diff)

M = [[1],[1],[1]]

m = np.dot(inverse,M)

mathmean = float((7 - m[2])/m[0] + 1)

print("By math, the expected number of good days = " + str(mathmean))

1.c) By simulation, the percentage of good days over the long run is 0.326602. Mathematically that answer is 0.32653

import numpy as np

trials = 1000000

P = np.array([[.5, .3 , .2], [.2, .6, .2], [.3, .2, .5]])

rolls = np.random.rand(trials)

numGoodDays = 0

state = 2

for i in range(trials):

sum = 0

for k in range(3):

sum = sum + P[state, k]

if(rolls[i] <= sum):

state = k

break

if(state == 0):

numGoodDays = numGoodDays + 1

percentGoodDays = numGoodDays / trials

print(" By simulation, the \% of good days over the long run = " + str(percentGoodDays))

P10000 = np.copy(P)

for i in range(9999):

P10000 = np.dot(P10000, P)

p = np.array([0,0,1])

A = np.dot(p,P10000)

ans = float(A[0])

print(" By math, the \% of good days over the long run = " + str(ans))

1d.) By simulation, the expected number of days until it gets good is 3.74306. Mathematically, the expected number of days until it gets good is 3.75

import numpy as np

trials = 1000000

P = np.array([[.5, .3 , .2], [.2, .6, .2], [.3, .2, .5]])

rolls = np.random.rand(trials)

daysElapsed = 0

daysElapsedA = list()

state = 2

for i in range(trials):

sum = 0

for k in range(3):

sum = sum + P[state, k]

if(rolls[i] <= sum):

state = k

break

daysElapsed = daysElapsed + 1

if(state == 0):

daysElapsedA.append(daysElapsed)

daysElapsed = 0

state = 2

mean = np.mean(daysElapsedA)

print(" By simulation, the expected number days until it gets good = " + str(mean))

P0 = np.array([[0, .3 , .2], [0, .6, .2], [0, .2, .5]])

I = np.identity(3)

diff = np.subtract(I,P0)

inverse = np.linalg.inv(diff)

M = [[1],[1],[1]]

m = np.dot(inverse,M)

mathmean = float(m[2])

print(" By math, the expected number days until it gets good= " + str(mathmean))

1e) By simulation, the expected number of days until it gets better is 1.99733. Mathematically, the expected number of days until it gets better is 1.9835

import numpy as np

trials = 1000000

P = np.array([[.5, .3 , .2], [.2, .6, .2], [.3, .2, .5]])

rolls = np.random.rand(trials)

daysElapsed = 0

daysElapsedA = list()

state = 2

for i in range(trials):

sum = 0

for k in range(3):

sum = sum + P[state, k]

if(rolls[i] <= sum):

state = k

break

daysElapsed = daysElapsed + 1

if(state != 2):

daysElapsedA.append(daysElapsed)

daysElapsed = 0

state = 2

mean = np.mean(daysElapsedA)

print(" By simulation, the expected number days until it gets better = " + str(mean))

P0 = np.array([[0, .3 , .2], [0, .6, .2], [0, .2, .5]])

I = np.identity(3)

diff = np.subtract(I,P0)

inverse = np.linalg.inv(diff)

M = [[1],[1],[1]]

m0 = np.dot(inverse,M)

P1 = np.array([[.5, 0 , .2], [.2, 0, .2], [.3, 0, .5]])

I = np.identity(3)

diff = np.subtract(I,P1)

inverse = np.linalg.inv(diff)

M = [[1],[1],[1]]

m1= np.dot(inverse,M)

mathmean = float(1/(1/m0[2] + 1/m1[2]))

print(" By math, the expected number days until it gets better= " + str(mathmean))

2.) By simulation, the probability of at least 8 flips until 3 consecutive heads is 0.63435 and the average number of flips until this condition is 13.998. Mathematically, the average number of flips until 3 consecutive heads is 14. And the probability of absorption vector is [[1][1][1]]. That is the probability of 3 consecutive heads approaches 1 as the number of flips approaches infinity.

import numpy as np

trials = 3000009

flips = np.random.rand(trials)

numFlipsUntilSuccss = 0

numFlipsUntilSuccssArr = list()

cnscHeads = 0

atLeast8 = 0

numSuccesses = 0

for i in range(trials):

numFlipsUntilSuccss = numFlipsUntilSuccss + 1

if(flips[i] < .5):

cnscHeads = cnscHeads + 1

if(cnscHeads == 3):

numSuccesses = numSuccesses + 1

if(numFlipsUntilSuccss >= 8):

atLeast8 = atLeast8 + 1

numFlipsUntilSuccssArr.append(numFlipsUntilSuccss)

numFlipsUntilSuccss = 0

cnscHeads = 0

if(flips[i] >= .5):

cnscHeads = 0

mean = np.mean(numFlipsUntilSuccssArr)

pAtLeast8 = atLeast8 / numSuccesses

print("By simulation, the probability of at least 8 flips until 3 consecutive heads = " + str(pAtLeast8))

print("By simulation, the average number of flips until 3 consecutive heads = " + str(mean))

Q = np.array([[1/2, 1/2, 0], [1/2, 0, 1/2], [1/2 , 0, 0]])

I = np.identity(3)

diff = np.subtract(I,Q)

inverse = np.linalg.inv(diff)

M = np.array([[1],[1],[1]])

A = np.dot(inverse, M)

mathmean = float(A[0])

print("By Markov Chain analysis, the average number of flips until 3 consective heads = " + str(mathmean))

R = np.array([[0],[0],[1/2]])

B = np.dot(inverse, R)

print("The probabities of absortion are given by the following vector")

print(B)