Tópicos em probabilidade (Arquivo 2)

Doriedson Rodrigues Neres Junior

18/04/2024

Para o Arquivo 2

1º problema

Processo de nº de sucesso em um processo de Bernoulli com dependência

$\Big{\{}S_n,n\geq 0\Big{\}}$ se $S_n=m$ então $S_{n+k}=m, \forall k$

$S_0=0 \wedge S_1=1$

$N=min\{n, S_n=m\}$

$E(N)=?$

Para $i=1,2,..,m-1$

$$P(S_{n+1}=i+1 \vee S_n=i)=1-\frac{i}{m}=\frac{m-i}{m}$$

$$P(S_{n+1}=i \vee S_n=i)=\frac{i}{m}$$

```
import scipy.stats as stata
import numpy as np

def preencher(rep):
    """""""
    respostas = []
    for i in range(rep):
        m = 500
        album = [np.random.randint(1, m + 1)]
        cont = 0
        while len(album) < m:
            fig = np.random.randint(1, m + 1)
            if fig in album: pass
            else: album.append(fig)
            #print(sorted(album))
            cont += 1
        respostas.append(cont)

    return respostas[1:20], np.mean(respostas)

preencher(1000)
```

```
([3092,
  2876,
  3433,
```

```
  2841,
  3164,
  2477,
  3442,
  2796,
  2563,
  3339,
  2457,
  3216,
  4049,
  3595,
  3170,
  2502,
  4534,
  4393,
  3172],
 3374.798)
```

```
def preencher(rep):
    """
    Refazer usando Bernoulli.
    """
    respostas = []
    for i in range(rep):
        m = 500
        album = [(1, m + 1)]
        cont = 0
        while len(album) < m:
            fig = np.random.randint(1, m + 1)
            if fig in album: pass
            else: album.append(fig)
            #print(sorted(album))
            cont += 1
        respostas.append(cont)

    return respostas[1:20], np.mean(respostas)


260
```

2º problema

Muito semelhante ao primeiro mas agora cada carta possui certo peso.

```
def alg2(rep, especial):
    """
    """

    respostas = []
    for i in range(rep):
```

```
        m, scomum, srara = 50, 0, 0
        cont = 0
        while (scomum + srara) < m:
            rara = stata.bernoulli.rvs(especial/m)
            if rara == 1:
                p = (especial - srara)/m
                carta = stata.bernoulli.rvs(p)
                if carta == 1: srara += 1
                else: pass
            else:
                p = (m - especial - scomum)/m
                carta = stata.bernoulli.rvs(p)
                if carta == 1: scomum += 1
                else: pass
            #print(sorted(album))
            cont += 1

        respostas.append(cont)

    return respostas[1:20], np.mean(respostas)

alg2(20, 10)
```

```
([749,
  716,
  558,
  773,
  579,
  562,
  1321,
  419,
  638,
  566,
  962,
  947,
  1397,
  1001,
  368,
  663,
  707,
  539,
  567],
 744.95)
```

Aula 30/04/2024

Sistemas aleatórios dinâmicos

Cadeia de Markov

A matriz de transição P será dada por:

$P= \begin{pmatrix} 3 & 6 & 1 & 0 & 0 & 0 & 0 \\ 3 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 5 & 0 & 0 & 0 \\ 0 & 0 & 8 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 4 & 0 \\ 0 & 8 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix}*0,1 $

```
import scipy.stats as stata
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

matriz = np.array([[3, 6, 1, 0, 0, 0, 0],
                   [3, 7, 0, 0, 0, 0, 0],
                   [0, 0, 5, 5, 0, 0, 0],
                   [0, 0, 8, 1, 1, 0, 0],
                   [0, 0, 0, 0, 6, 4, 0],
                   [0, 8, 0, 0, 1, 0, 1],
                   [0, 0, 0, 0, 0, 0, 10]])


r=matriz[0,]>5
print(r)
np.where(matriz[0, ]>5)[0][0]
```

```
[False  True False False False False False]
```

```
1
```

```
# Arquivo 2

def sorteio():
    """
    Retorna uma uniforme.
    """
    return stata.uniform.rvs()


def movimento(m1, inicio=0):
    """
    Simula o movimento discreto de uma partícula, dada a matriz de
transição.
    Recebe a matriz de transição que deve estar com probabilidades
```

```
ACUMULADAS
    entre 0 e 10 (0 para evento impossível e 10 para o evento certo).
    Recebe também a posição de início (contagem a partir do zero).
    """
    estado_atual = inicio
    contador = 0
    while estado_atual != len(m1) - 1:
        x = sorteio()*10
        estado_proximo = np.where(m1[estado_atual, ] > x)[0][0]
        estado_atual = estado_proximo
        contador += 1

        #print(estado_atual, contador)
    return contador

%%time

m = np.array([[3, 9, 10, 0, 0, 0, 0],
              [3, 10, 0, 0, 0, 0, 0],
              [0, 0, 5, 10, 0, 0, 0],
              [0, 0, 8, 9, 10, 0, 0],
              [0, 0, 0, 0, 6, 10, 0],
              [0, 8, 0, 0, 9, 0, 10],
              [0, 0, 0, 0, 0, 0, 10]])

x, n = [], 10**3
for i in range(n):
    x.append(movimento(m1=m))

np.average(x)
```

```
CPU times: total: 18.3 s
Wall time: 29.5 s
```

```
590.183
```

```
%%time

m = np.array([[3, 9, 10, 0, 0, 0, 0],
              [3, 10, 0, 0, 0, 0, 0],
              [0, 0, 5, 10, 0, 0, 0],
              [0, 0, 8, 9, 10, 0, 0],
              [0, 0, 0, 0, 6, 10, 0],
              [0, 8, 0, 0, 9, 0, 10],
              [0, 0, 0, 0, 0, 0, 10]])

x, n = [], 10**4
for i in range(n):
    x.append(movimento(m1=m))

np.average(x)
```

```
CPU times: total: 3min 25s
Wall time: 4min 43s

585.1024

%%time

m = np.array([[3, 9, 10, 0, 0, 0, 0],
              [3, 10, 0, 0, 0, 0, 0],
              [0, 0, 5, 10, 0, 0, 0],
              [0, 0, 8, 9, 10, 0, 0],
              [0, 0, 0, 0, 6, 10, 0],
              [0, 8, 0, 0, 9, 0, 10],
              [0, 0, 0, 0, 0, 0, 10]])

x, n = [], 10**5
for i in range(n):
    if i%10000 == 0: print(f"Foram {i}")
    x.append(movimento(m1=m))

np.average(x)

Foram 0
Foram 10000
Foram 20000
Foram 30000
Foram 40000
Foram 50000
Foram 60000
Foram 70000
Foram 80000
Foram 90000
CPU times: total: 40min 35s
Wall time: 52min 59s

585.77534

m = np.array([[3, 9, 10, 0, 0, 0, 0],
              [3, 10, 0, 0, 0, 0, 0],
              [0, 0, 5, 10, 0, 0, 0],
              [0, 0, 8, 9, 10, 0, 0],
              [0, 0, 0, 0, 6, 10, 0],
              [0, 8, 0, 0, 9, 0, 10],
              [0, 0, 0, 0, 0, 0, 10]])

x, f, n = [], [], 10**2
for i in range(n):
    for j in range(n):
        x.append(movimento(m1=m))
    resp = np.mean(x)
    f.append(resp)
```

```
np.mean(f)
```

```
---------------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[10], line 12
     10 for i in range(n):
     11     for j in range(n):
---> 12         x.append(movimento(m1=m))
     13     resp = np.mean(x)
     14     f.append(resp)

Cell In[6], line 21, in movimento(m1, inicio)
     19 while estado_atual != len(m1) - 1:
     20     x = sorteio()*10
---> 21     estado_proximo = np.where(m1[estado_atual, ] > x)[0][0]
     22     estado_atual = estado_proximo
     23     contador += 1

File ~\anaconda3\envs\ambiente_zero\Lib\site-packages\numpy\core\
multiarray.py:346, in where(condition, x, y)
    256     """
    257     inner(a, b, /)
    258
   (...)
    341
    342     """
    343     return (a, b)
--> 346
@array_function_from_c_func_and_dispatcher(_multiarray_umath.where)
    347 def where(condition, x=None, y=None):
    348     """
    349     where(condition, [x, y], /)
    350
   (...)
    416             [ 0,  3, -1]])
    417     """
    418     return (condition, x, y)

KeyboardInterrupt:
```

Calcular:

$$E\left[T_6 \vee X_0 = 1\right)$$

$$E\left[T_1 \vee T_1 < \infty\right)$$

```
def andar_m(matriztr, estado_i, estado_fim):
    """
    Simula um passeio aleatório simples cuja matriz de transição é
dada.
    Encerra o passeio assim que ocorrer o alcance de do estado final
    pela primeira vez.

    Args:
    matriztr: um NumPy array que representa a matriz de transição.
    estado_i: estado inicial do passeio aleatório.
    estado_fim: estado final do passeio aleatório.

    Retorna:
    A lista de estados visitados e a quantidade de 'passos'
    até o fim do experimento.
    """

    estado_a = estado_i
    estados = [estado_a]

    contador = 0
    while contador == 0:
        # Get probabilities of transitioning to each state from the
current state
        probs = matriztr[estado_a]

        # Choose the next state based on the transition probabilities
        estado_prox = np.random.choice(len(probs), p=probs)

        estado_a = estado_prox

        if estado_a == estado_fim:
            estados.append(estado_a)
            contador += 1

        else:
            estados.append(estado_a)

    return estados, len(estados)-1

matriz = np.array([[3, 6, 1, 0, 0, 0, 0],
                   [3, 7, 0, 0, 0, 0, 0],
                   [0, 0, 5, 5, 0, 0, 0],
                   [0, 0, 8, 1, 1, 0, 0],
                   [0, 0, 0, 0, 6, 4, 0],
                   [0, 8, 0, 0, 1, 0, 1],
                   [0, 0, 0, 0, 0, 0, 10]])

matriz = matriz*.1

print("\t\t\t Matriz de transição:\n\n", pd.DataFrame(matriz))
```

```python
inicial = 1
final = 6

respostas = []
for i in range(2, 4):
    x = []
    for rep in range(10**i):
        # O tempo de retorno de cada passeio
        r_walk = andar_m(matriz, inicial, final)[1]
        x.append(r_walk)

    respostas.append(np.mean(x))
    print(f'\nPara 10^{i} repetições, o tempo médio de retorno foi
{np.mean(x)}')
```

```
                  Matriz de transição:

      0    1    2    3    4    5    6
0  0.3  0.6  0.1  0.0  0.0  0.0  0.0
1  0.3  0.7  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.5  0.5  0.0  0.0  0.0
3  0.0  0.0  0.8  0.1  0.1  0.0  0.0
4  0.0  0.0  0.0  0.0  0.6  0.4  0.0
5  0.0  0.8  0.0  0.0  0.1  0.0  0.1
6  0.0  0.0  0.0  0.0  0.0  0.0  1.0

Para 10^2 repetições, o tempo médio de retorno foi 538.25

Para 10^3 repetições, o tempo médio de retorno foi 586.578
```

09/07/2024

Baralho de 52 cartas: Extrair 3 cartas sem reposição.

- Quantas são as extrações onde
- na 1ª extração não sai carta de copas
- na 2ª extração sai carta de espadas
- na 3ª extração não sai dama

```
from scipy import stats
import numpy as np
import random as rd

def carta(exceto=[]):
    """
    Recebe uma lista de exceções, no padrão a lista é vazia.
    Faz uma carta do baralho.
    Combinação de uma das 4 pintas mais um dos 13 números.
    """

    while True:
        numero = str(np.random.randint(1, 13))
        pinta = rd.choice(["c","e","o","p"])
        x = numero + pinta

        if x in exceto:
            continue
        else:
            break

    return x

def baralho(n=1000):
    """
    """

    copas = [str(i)+"c" for i in range(1,14)]
    espadas = [str(i)+"e" for i in range(1,14)]
    damas = ["12c","12o","12e","12p"]

    sucesso, total = 0, 0
    while total < n:
        x1 = carta()
        x2 = carta([x1])
        x3 = carta([x1, x2])

        if (x1 not in copas) and (x2 in espadas) and (x3 not in
damas):
```

```
            sucesso += 1

        total += 1

    return (sucesso, total, sucesso/total, 132600*sucesso/total)

lista = []

for i in range(10**3):
    x = baralho(10**3)
    lista.append(x[2])

np.mean(lista)
```

```
0.17023399999999997
```

```
baralho(10**4)
```

```
(1702, 10000, 0.1702, 22568.52)
```

```
baralho(10**5)
```

```
(16801, 100000, 0.16801, 22278.126)
```

```
baralho(10**6)
```

```
(171017, 1000000, 0.171017, 22676.8542)
```

```
baralho(10**7)
```

```
(1708411, 10000000, 0.1708411, 22653.52986)
```