Métodos Montecarlo Fing 2022 - Entrega 4

Autor: Carlos M. Martinez, mayo 2022.

Email: carlosm@fing.edu.uy, carlos@cagnazzo.uy

Ejercicio 7.1

Problema: supongamos que un programa de posgrado incluye un conjunto de P profesores y otro de S estudiantes, y que se desea asignar a cada estudiante un profesor para consultas, pero que todas las personas son de diferentes países y comprenden distintos idiomas. Si cada estudiante tiene asignado un profesor para consultas, la cantidad total de formas de asignar profesores a los estudiantes es P^S (hay P opciones para cada estudiante, y S estudiantes en total, la cantidad total de opciones es el producto sobre los estudiantes de las opciones de cada estudiante)

.. recortado por brevedad ...

Para estimar la cantidad de formas distintas de realizar estas asignaciones de profesores y estudiantes, es posible aplicar el método Monte Carlo.

Se debe recibir en entrada el número de replicaciones a realizar, y el nivel de confianza; en salida, se debe dar la estimación del número de combinaciones NC, así como la desviación estándar y un intervalo de confianza (del nivel especificado) calculado en base al criterio de Agresti-Coull.

Parte a:

Escribir un programa para hacer el cálculo previamente descrito. Entregar seudocódigo y código.

Identificamos para este problema cuales son nuestros conjuntos base (X) y de objetos (F). X será el conjunto de todas las asignaciones posibles entre profesores y estudiantes. F será un conjunto compuesto por los subconjuntos de X que constituyen asignaciones válidas segun los criterios del problema. En la parte (a) el único criterio es el de que tengan un idioma en común.

X = {x | x es una asignación posible entre profesores y estudiantes, cada estudiante tiene un único profesor}

F = { S1 }

S1 = { x | x es una asignacón valida según el criterio de tener un idioma en común }

Cada elemento de X lo modelo como una matriz x de S filas y P columnas donde en cada fila hay un único 1 (indicando el profesor asignado) y el resto son ceros.

Los profesores se modelan como una mariz P de P filas y L columnas, con un 1 en cada idioma que el profesor habla.

Los estudiantes se modelan como una matriz S de S filas y L columnas, con un 1 en cada idioma que el estudiante habla.

Pseudocodigo

Primero tengo el procedimiento para estimar el conteo mismo, luego defino dos funciones auxiliares.

```
PROC: Montecarlo-Asignaciones
ENTRADAS: Matriz P, Matriz S, n (tam. muestra), r = ||X||, 1-delta
(int. confianza)
SALIDAS: Zest (estimacion del conteo), VZest (estimacion de la
desv.std.), (I1, I2) (intervalo de confianza AgC)

S = 0

FOR i = 1 to n
    a = sorteo aleatorio dentro de X
    IF PERTENECE_S1(a,P,S)==true THEN S = S + 1
ENDFOR

Zest = r*S/n
VZest = Zest*(r-Zest)/(n-1)
(I1, I2) = Agresti_Coull(S,n,delta,r)
```

La primer funcion auxiliar es la que determina si un elemento sorteado pertence a S1.

```
FN PERTENECE_S1
ENTRADAS: a (elemento sorteado en X), P (matriz profesores), S
(matriz estudiantes)
SALIDAS: true / false dependiendo si a pertenece a S1 o no

R1 = true
FOR i = 1 to S:
    pi = profesor asignado en a /* es el indice del unico 1 que esta
prendido en la fila i de a */
    idpi = fila pi de la matriz P
    idsi = fila i de la matriz S
    IF IDIOMA_EN_COMUN(id_pi, idsi) == false THEN
        /* si no tienen ningun idioma en comun esta asignacion ya no
es válida */
        R1 = false
        BREAK
    ENDIF
ENDFOR
```

```
    RETURN R1
```

La segunda función auxiliar es la que determina si hay un idioma en común entre profesor y estudiante.

```
    FN IDIOMA_EN_COMUN(idpi, idsi)
    ENTRADAS: idpi (linea de matriz de L elementos), idsi (linea de
    matriz de L elementos)
        R1 = false

        FOR j = 1 to L:
            IF idpi(j) == 1 AND idsi(j) == 1 THEN:
                R1 = true
                BREAK
            ENDIF
        ENDFOR

        RETURN R1
```

Parte b:

Problema: Sea el siguiente caso:

Estudiantes/lenguajes: Maria: español, ingles; Sophie: inglés, francés; Liliana: español, portugués; Lucia: inglés, portugués; Monique: francés; Rodrigo: español, inglés, francés; John: inglés; Neymar: portugués, español; Jacques: francés, portugués; Juan: español.

Profesores: Tom: inglés, francés, español; Luciana: inglés, portugués; Gerard: inglés, francés; Silvia: español, francés.

Usando el programa anterior, y empleando 1000 replicaciones de Monte Carlo, estimar los valores de cuantas combinaciones NC hay para asignar profesores a estudiantes y que tengan un idioma en común, con intervalos de confianza de nivel 95%

In [ ]:
```
import random
import math
import pandas
from IPython.core.display import HTML
random.seed()

import cm2c.fing.mmc.conteo as mmc
import cm2c.fing.mmc.utils as mmcutils
reloj_ppal = mmcutils.timeit()
mmc.version()
```

Out[ ]:
```
'Problemas de conteo MMC v0.1.1 - Carlos Martinez mayo 2022'
```

In [ ]:
```
# Defino estructuras de datos para los datos del problema

from enum import IntEnum, unique, auto
```

```
@unique
class Idiomas(IntEnum):
    espanol = 0
    ingles = 1
    frances = 2
    portugues = 3

@unique
class Profesores(IntEnum):
    tom = 0
    luciana = 1
    gerard = 2
    silvia = 3

@unique
class Estudiantes(IntEnum):
    maria = 0
    sophie = 1
    liliana = 2
    lucia = 3
    monique = 4
    rodrigo = 5
    john = 6
    neymar = 7
    jacques = 8
    juan = 9

# Defino la matriz S de estudiantes e Idiomas (tamaño S x L )
S = [ [ 0 for col in range(len(Idiomas))] for row in range(len(Estudiantes)) ]

S[Estudiantes.maria][Idiomas.espanol] = 1
S[Estudiantes.maria][Idiomas.ingles] = 1

S[Estudiantes.sophie][Idiomas.ingles] = 1
S[Estudiantes.sophie][Idiomas.frances] = 1

S[Estudiantes.liliana][Idiomas.espanol] = 1
S[Estudiantes.liliana][Idiomas.portugues] = 1

S[Estudiantes.lucia][Idiomas.ingles] = 1
S[Estudiantes.lucia][Idiomas.portugues] = 1

S[Estudiantes.monique][Idiomas.frances] = 1

S[Estudiantes.rodrigo][Idiomas.espanol] = 1
S[Estudiantes.rodrigo][Idiomas.ingles] = 1
S[Estudiantes.rodrigo][Idiomas.frances] = 1

S[Estudiantes.john][Idiomas.ingles] = 1

S[Estudiantes.neymar][Idiomas.portugues] = 1
S[Estudiantes.neymar][Idiomas.espanol] = 1

S[Estudiantes.jacques][Idiomas.frances] = 1
S[Estudiantes.jacques][Idiomas.portugues] = 1

S[Estudiantes.juan][Idiomas.espanol] = 1

# Defino la matriz P de profesores e Idiomas (tamaño P x L )
P = [ [ 0 for col in range(len(Idiomas))] for row in range(len(Profesores)) ]
```

```
P[Profesores.tom][Idiomas.ingles] = 1
P[Profesores.tom][Idiomas.frances] = 1
P[Profesores.tom][Idiomas.espanol] = 1

P[Profesores.luciana][Idiomas.ingles] = 1
P[Profesores.luciana][Idiomas.portugues] = 1

P[Profesores.gerard][Idiomas.ingles] = 1
P[Profesores.gerard][Idiomas.frances] = 1

P[Profesores.silvia][Idiomas.espanol] = 1
P[Profesores.silvia][Idiomas.frances] = 1

print(S)
```

```
[[1, 1, 0, 0], [0, 1, 1, 0], [1, 0, 0, 1], [0, 1, 0, 1], [0, 0, 1, 0], [1, 1, 1, 0],
[0, 1, 0, 0], [1, 0, 0, 1], [0, 0, 1, 1], [1, 0, 0, 0]]
```

In [ ]:

```
# defino la funcion para sortear en el espacio X de asignaciones de profesores a estud

def SortearPyS():
    """
    sorteo una asignacion de profesores a estudiantes
    """

    nfilas = len(Estudiantes)
    ncols = len(Profesores)

    random.seed()

    a = [ [0 for col in range(ncols)] for row in range(nfilas) ]

    # cada estudiante tiene un unico profesor asignado
    for i in range(len(Estudiantes)):
        a[i][ random.randint(0, len(Profesores)-1) ] = 1 #el -1 es porque randint incl

    return a

# end def

def IdiomaEnComun(id_pi, id_si):
    R1 = False

    for j in range(len(Idiomas)):
        if id_pi[j] == 1 and id_si[j] == 1:
            R1 = True
            break
        # end if
    # end for

    return R1
# end def

def AsignacionValidaIdioma(a):
    R1 = True

    for i in range(len(Estudiantes)):
        pi = a[i].index(1)
        idpi = P[pi]
```

```
        idsi = S[i]
        if IdiomaEnComun(idpi, idsi) == False:
            R1 = False
            break
        # end if
    # end for

    return R1
# end def
```

In [ ]:
```
# Ahora corro el Montecarlo con los parametros del problema
n = 10**4
delta = 0.05
r = len(Profesores)**len(Estudiantes)

resultado = mmc.MonteCarlo_Conteo(n, r, delta, SortearPyS, AsignacionValidaIdioma)

print("Resultado: \n")
print(f"Cantidad de muestras n          : {n:,} " )
print(f"Estimador NC                    : {resultado[0]:,} " )
print(f"Estimador DevStd                : {resultado[1]:.2f} " )
print(f"I de C Agresti-Coull al {1-delta} +/-: {resultado[3]:.2f} " )
```

```
Resultado:

Cantidad de muestras n          : 10,000
Estimador NC                    : 128,555
Estimador DevStd                : 3439.26
I de C Agresti-Coull al 0.95 +/-: 6742.63
```

Parte c:

Problema:

adaptar el programa para calcular el n´umero de combinaciones si adem´as queremos agregar como restricci´on que ning´un profesor atienda menos de un estudiante ni m´as de cuatro estudiantes.

Usando el programa modificado, y empleando 1000 replicaciones de Monte Carlo, estimar los valores de cuantas combinaciones NC hay para asignar profesores a estudiantes y que tengan un idioma en com´un y cuplan la restricci´on adicional mencionada, con intervalos de confianza de nivel 95%

Planteo

La modificacion que debo hacer es ahora tener en cuenta que tengo dos conjuntos Sj, S1 (definido como anteriormente, para el criterio del idioma común) y además un S2 definido según el criterio de carga docente de cada profesor.

El resultado que busco es la intersección de S1 y S2. Esto puedo lograrlo modificando la funcion que paso para determinar que es una asignación válida para que tenga en cuenta ambos criterios.

In [ ]:
```
def AsignacionValidaCargaProfesor(a):
```

```
    R1 = True

    for i in range(0, len(Profesores)):
        C = 0
        for j in range(0, len(Estudiantes)):
            C = C + a[j][i]
        # end for
        if C<1 or C>4:
            R1 = False
            break
    # end for

    return R1
# end for

def InterseccionS1S2(a):
    return AsignacionValidaIdioma(a) and AsignacionValidaCargaProfesor(a)
# end def

# Ahora corro el Montecarlo con los parametros del problema
n = 10**5
delta = 0.05
r = len(Profesores)**len(Estudiantes)

resultado = mmc.MonteCarlo_Conteo(n, r, delta, SortearPyS, InterseccionS1S2)

print("Resultado: \n")
print(f"Cantidad de muestras n          : {n:,} " )
print(f"Estimador NC                    : {resultado[0]:,} " )
print(f"Estimador DevStd                : {resultado[1]:.2f} " )
print(f"I de C Agresti-Coull al {1-delta} +/-: {resultado[3]:.2f} " )
```

```
Resultado:

Cantidad de muestras n          : 100,000
Estimador NC                    : 74,375
Estimador DevStd                : 851.22
I de C Agresti-Coull al 0.95 +/-: 1668.50
```

Datos adicionales y referencias

Información acerca del software y hardware utilizados

Software:

- Python 3.8.10 corriendo en Windows WSL2 (Windows Subsystem for Linux)
- Jupyter Notebook

Librerias:

- scipy norm
- pathos multiprocessing (para paralelizar ejecuciones)

Hardware:

- PC Windows 11, con WSL2

- CPU Intel Core i5 10400F (6 cores)
- 16 GB de RAM

In [ ]:
```
print(f"%% FIN - tiempo total de ejecución {reloj_ppal.lap():.3f}s")
```

```
%% FIN - tiempo total de ejecución 3.857s
```

Código de las funciones desarrolladas

Adjunto en el archivo "*conteo.py.pdf*".