Ejercicio_Práctico_v31_01

January 31, 2024

1 Ejercicio Práctico : Soft Computing

Autores: Danel Arias y Rubén Pérez

Fecha: 31 de enero 2024

```
[ ]: # imports
     import numpy as np
```

1.1 Enunciado

Supongamos que cuatro expertos, $E = \{e_1, e_2, e_3, e_4\}$, deben elegir la mejor alternativa de entre un conjunto de cuatro posibles, $X = \{x_1, x_2, x_3, x_4\}$. Para ello, expresan sus opiniones mediante las siguientes relaciones de preferencia difusas:

$$p^1 = \begin{pmatrix} - & 0.5 & 0.7 & 0.8 \\ 0.5 & - & 0.6 & 0.1 \\ 0.2 & 0.3 & - & 0.5 \\ 0.2 & 0.8 & 0.5 & - \end{pmatrix}$$

$$p^2 = \begin{pmatrix} - & 0.9 & 0.2 & 0.2 \\ 0.1 & - & 0.3 & 0.9 \\ 0.7 & 0.7 & - & 0.6 \\ 0.7 & 0.1 & 0.4 & - \end{pmatrix}$$

$$p^3 = \begin{pmatrix} - & 0.7 & 0.8 & 0.6 \\ 0.3 & - & 0.7 & 0.2 \\ 0.1 & 0.3 & - & 0.3 \\ 0.3 & 0.7 & 0.6 & - \end{pmatrix}$$

$$p^4 = \begin{pmatrix} - & 0.2 & 0.8 & 0.8 \\ 0.7 & - & 0.6 & 0.3 \\ 0.1 & 0.3 & - & 0.5 \\ 0.1 & 0.7 & 0.5 & - \end{pmatrix}$$

```
[ ]: # Representación de las matrices en numpy
     p1 = np.array([[np.nan, 0.5, 0.7, 0.8],
                    [0.5, np.nan, 0.6, 0.1],
                    [0.2, 0.3, np.nan, 0.5],
```

```
                [0.2, 0.8, 0.5, np.nan]])
p2 = np.array([[np.nan, 0.9, 0.2, 0.2],
               [0.1, np.nan, 0.3, 0.9],
               [0.7, 0.7, np.nan, 0.6],
               [0.7, 0.1, 0.4, np.nan]])
p3 = np.array([[np.nan, 0.7, 0.8, 0.6],
               [0.3, np.nan, 0.7, 0.2],
               [0.1, 0.3, np.nan, 0.3],
               [0.3, 0.7, 0.6, np.nan]])
p4 = np.array([[np.nan, 0.2, 0.8, 0.8],
               [0.7, np.nan, 0.6, 0.3],
               [0.1, 0.3, np.nan, 0.5],
               [0.1, 0.7, 0.5, np.nan]])
pref = np.array([p1, p2, p3, p4])
n_x = p1.shape[1]
```

1.2 Preguntas a responder

> Nota: Use la relación de preferencia colectiva obtenida en el punto 2 para calcular los grados de dominancia y de no dominancia solicitados en los puntos 3 y 4.

1.2.1 Ejercicio 1

¿Cuál es el nivel de consenso alcanzado entre los cuatro expertos?

[ ]:
```
# Cálculo de las matrices de similitud
sms =[]
for i in range(len(pref)):
    for j in range(i+1, len(pref)):
        sm = 1 - np.abs(pref[i] - pref[j])
        print(f'SM_{i+1}_{j+1}:\n{sm}')
        sms.append(sm)
```

```
SM_1_2:
[[nan 0.6 0.5 0.4]
 [0.6 nan 0.7 0.2]
 [0.5 0.6 nan 0.9]
 [0.5 0.3 0.9 nan]]
SM_1_3:
[[nan 0.8 0.9 0.8]
 [0.8 nan 0.9 0.9]
 [0.9 1.  nan 0.8]
 [0.9 0.9 0.9 nan]]
SM_1_4:
[[nan 0.7 0.9 1. ]
 [0.8 nan 1.  0.8]
 [0.9 1.  nan 1. ]
```

```
 [0.9 0.9 1.  nan]]
SM_2_3:
[[nan 0.8 0.4 0.6]
 [0.8 nan 0.6 0.3]
 [0.4 0.6 nan 0.7]
 [0.6 0.4 0.8 nan]]
SM_2_4:
[[nan 0.3 0.4 0.4]
 [0.4 nan 0.7 0.4]
 [0.4 0.6 nan 0.9]
 [0.4 0.4 0.9 nan]]
SM_3_4:
[[nan 0.5 1.  0.8]
 [0.6 nan 0.9 0.9]
 [1.  1.  nan 0.8]
 [0.8 1.  0.9 nan]]
```

Una vez tenemos estas matrices, podemos calcular la matriz de consenso, que se obtiene como la media de las matrices de similitud.

```
[ ]: consensus = np.mean(sms, axis=0)

     print(f'Matriz de consenso:\n{consensus}')
```

```
Matriz de consenso:
[[       nan 0.61666667 0.68333333 0.66666667]
 [0.66666667        nan 0.8        0.58333333]
 [0.68333333 0.8               nan 0.85      ]
 [0.68333333 0.65       0.9               nan]]
```

Calculamos el nivel de consenso global como la media de los valores de la matriz de consenso.

```
[ ]: global_consensus = np.mean(consensus[~np.isnan(consensus)])
     print(f'Consenso global:\n{global_consensus}')
```

```
Consenso global:
0.7152777777777778
```

1.2.2 Ejercicio 2

Usando el operador OWA con el cuantificador lingüístico difuso «mayoría», definido por $Q(r) = r^{1/2}$, para obtener sus pesos, ¿cuál es la relación de preferencia colectiva?

Vamos a definir dos funciones, una para calcular el operador OWA (en este caso, $Q(r) = r^{1/2}$), y otra que, dado el número de pesos, calcule los valores automáticamente

```
[ ]: def q(r):
         return np.sqrt(r)
```

```python
def calc_w(n):
    w = np.ndarray(n)
    for i in range(1, n+1):
        w[i-1] = q(i / n) - q((i-1) / n)
    return w

n_exp = len(pref) # número de expertos
w4 = calc_w(n_exp)
```

Definimos también el operador de agregación: $\psi_W(p_1, ...p_n) = \sum_{i=1}^{n} w_i * p_{\sigma(i)}$

[ ]:
```python
def psi_w(array_p, w):
    # Ordenar de mayor a menor
    array_p = np.sort(array_p)[::-1]
    # Aplicar los pesos
    return np.sum(array_p * w)
```

[ ]:
```python
# Matriz de preferencia colectiva
pref_col = np.ndarray((n_x, n_x))

for i in range(n_x):
    for j in range(n_x):
        # Guardamos en pref_ij los elementos ij de cada matriz
        pref_ij = np.array([p[i, j] for p in pref])
        # Guardamos el valor de la función psi_w en la matriz de preferencia␣
 ↪agregada
        pref_col[i, j] = psi_w(pref_ij, w4)

print(f'Matriz de preferencia colectiva:\n{pref_col}')
```

```
Matriz de preferencia colectiva:
[[       nan 0.70122898 0.70372338 0.68783152]
 [0.51462644        nan 0.60980762 0.55731322]
 [0.42071068 0.5               nan 0.52320508]
 [0.45731322 0.66961524 0.53660254        nan]]
```

1.2.3 Ejercicio 3

¿Cuál es el grado de dominancia guiado por cuantificador asociado a cada alternativa? Indica el ranking de alternativas obtenido. Utilice de nuevo el operador OWA con el cuantificador lingüístico difuso «mayoría», definido por $Q(r) = r^{1/2}$, para obtener sus pesos.

[ ]:
```python
# Los pesos son los mismos que en el caso anterior
# Calculamos el grado de dominancia
dom = np.ndarray((n_x))
mask = np.ones(n_x, dtype=bool)
w3 = calc_w(n_exp - 1)
```

```python
for i in range(n_x):
    mask[i] = False
    mask[i-1] = True

    fila = pref_col[i, mask]

    dom[i] = psi_w(fila, w3)

print(f'Grado de dominancia:')
for i in range(n_x):
    print(f'\tX{i+1} = {dom[i]}')
```

```
Grado de dominancia:
        X1 = 0.7002106417598539
        X2 = 0.5797877056362575
        X3 = 0.49884759796040723
        X4 = 0.5988475979604071
```

Como podemos observar, la alternativa x_1 es la más dominante, seguida de x_4 y x_2, mientras que x_3 es la menos dominante.

1.2.4 Ejercicio 4

¿Cuál es el grado de **no** dominancia guiado por cuantificador asociado a cada alternativa? Indica el ranking de alternativas obtenido. Use de nuevo el operador OWA con el cuantificador lingüístico difuso «mayoría», definido por $Q(r) = r^{1/2}$, para obtener sus pesos.

[ ]:
```python
# Los pesos son los mismos que en el caso anterior
# Calculamos el grado de NO dominancia
no_dom = np.zeros((n_x))
# Calculamos haciendo el max(0, diferencia entre el elemento de la columna y su␣
↪simétrico respecto a la diagonal)
mask = np.ones(n_x, dtype=bool)
for col in range(n_x):
    # Actualizamos mask
    mask[col] = False
    mask[col-1] = True

    # Tomamos la fila y la columna col
    el = pref_col[mask, col]
    sim = pref_col[col, mask]

    # Calculamos los valores
    res = 1 - np.maximum(0, el - sim)

    # Operador de agregación
    no_dom[col] = psi_w(res, w3)
```

```python
print(f'Grado de NO dominancia:')
for i in range(n_x):
    print(f'\tX{i+1} = {no_dom[i]}')
```

```
Grado de NO dominancia:
	X1 = 1.0
	X2 = 0.9389011810468111
	X3 = 0.9140710870476838
	X4 = 0.9576991039281679
```

Tras realizar los cálculos, obtenemos que la alternativa x_1 es la más "no dominante", seguida de x_4 y x_2, mientras que x_3 es la menos "no dominante".